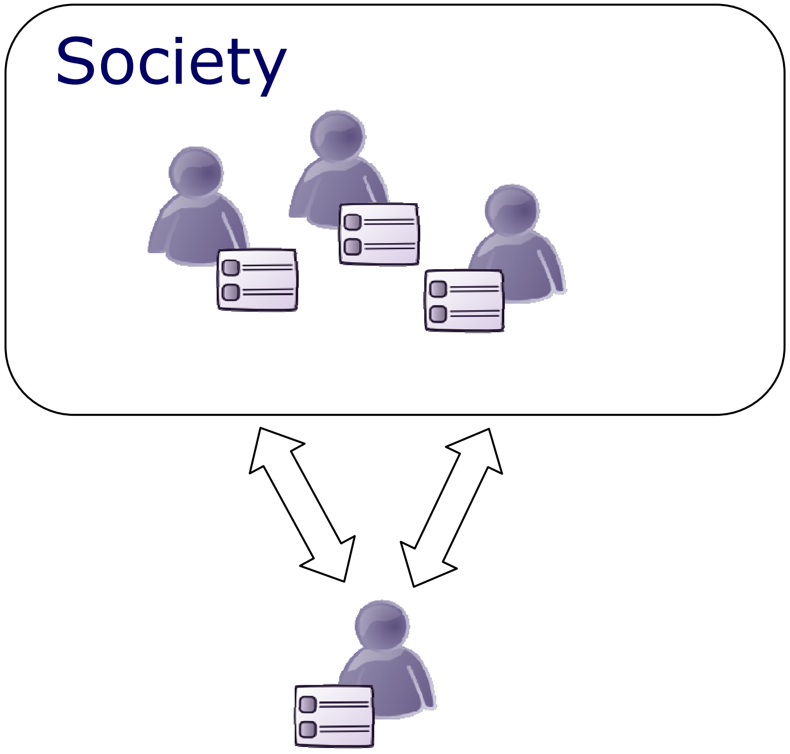
Reasoning on choreographies and capability requirements

M. Baldoni, C. Baroglio, A. Martelli, V. Patti, C.Schifanella

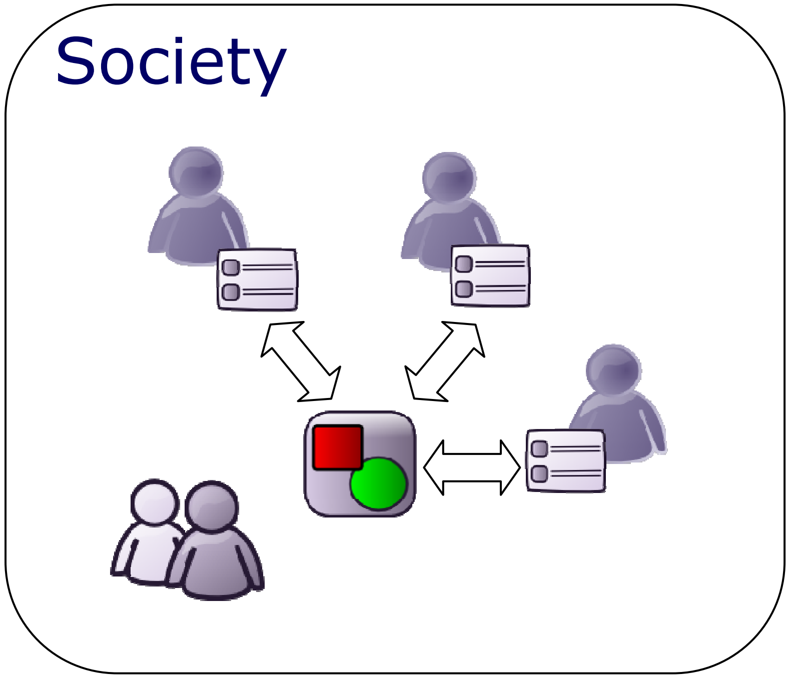
Bologna, 30-01-2007

Interoperability



- Heterogeneous and independent entities want to execute a shared task
- Interoperability is the capability of an entity of interacting with others
- Each entity must verify interoperability with other participants

Interoperability

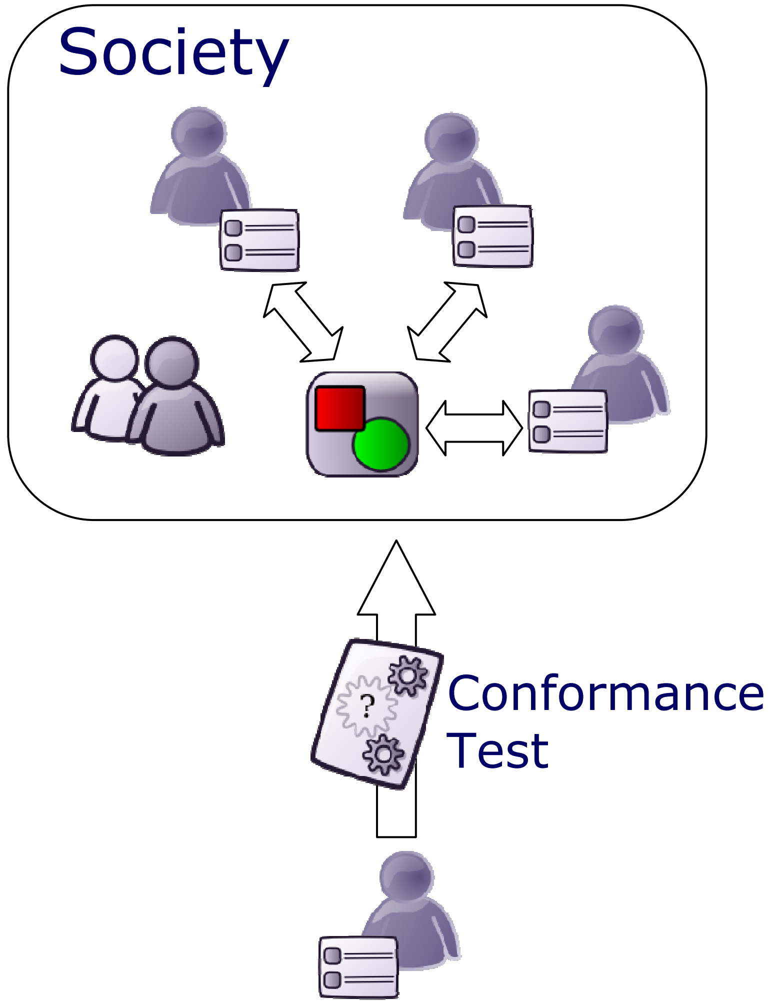


- We can introduce a description of the overall behavior

Choreography (Interaction protocol): global point of view by means of
  - roles
  - messages exchanged

Policy: local point of view of a single entity (orchestration)

Checking interoperability



- An entity that is conform to a protocol produces a legal (complete) conversation
- Conformance test entails a priori interoperability
- See Baldoni et al.
  - Agents: [Clima V, Clima VI]
  - Web Services: [WS-FM 05, ICSOC 06]

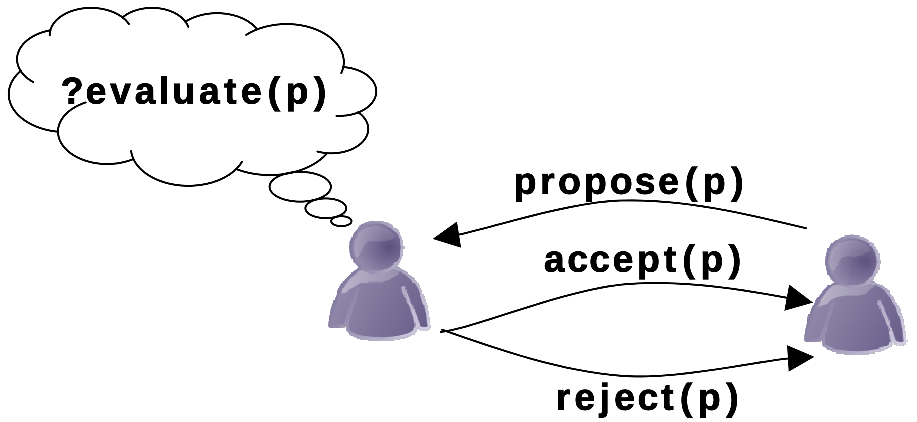
Interoperability

What happens if the entity has no valid policy (for example, the conformance test fails) ?

- (1) It may ask to other entities a correct interaction policy for the role that it wants to play
  - CooBDI, CooWS [Bozzo et al., ICWI 2005]
  - Knowledge exchange in DALI language [Costantini, Tocchio, WOA 05]
- (2) It may generate a conformant policy from a high-level description of the interaction

Interoperability

- Interaction protocols only concern roles and communicative behavior
- But an entity that wants to play a specific role must also execute actions that do not only concern communication
  - e.g.: producing a proposal, processing some data, checking if a product is in the store

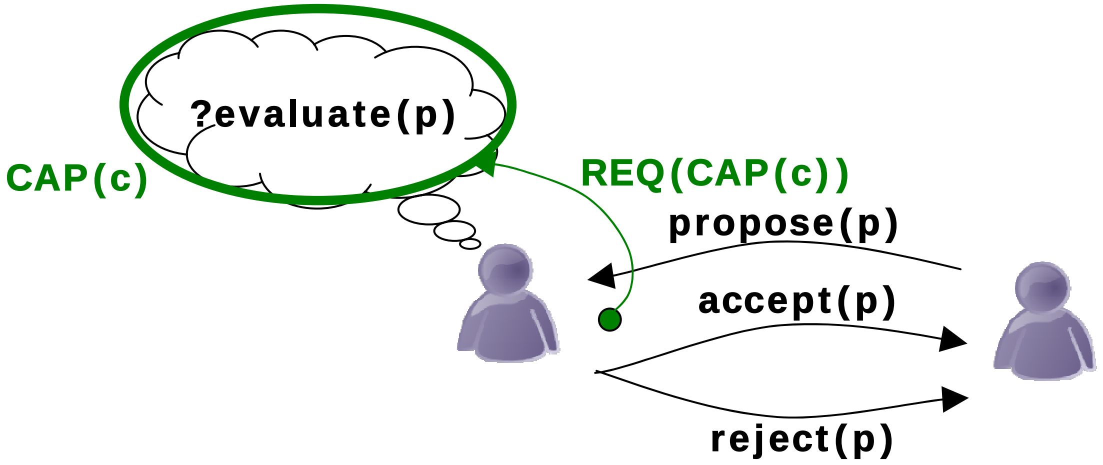


Capability requirements

- We can enrich Interaction Protocols with an high-level description of the actions that each entity must be able to execute if it wants to play a role
- The entity must own an implementation of these actions in order to generate in a (semi)automatic way an executable policy
- We call these skills **capability requirements**

Capability requirements

- We propose the extension of Interaction Protocols/choreographies with the notion of "requirement for a capability"

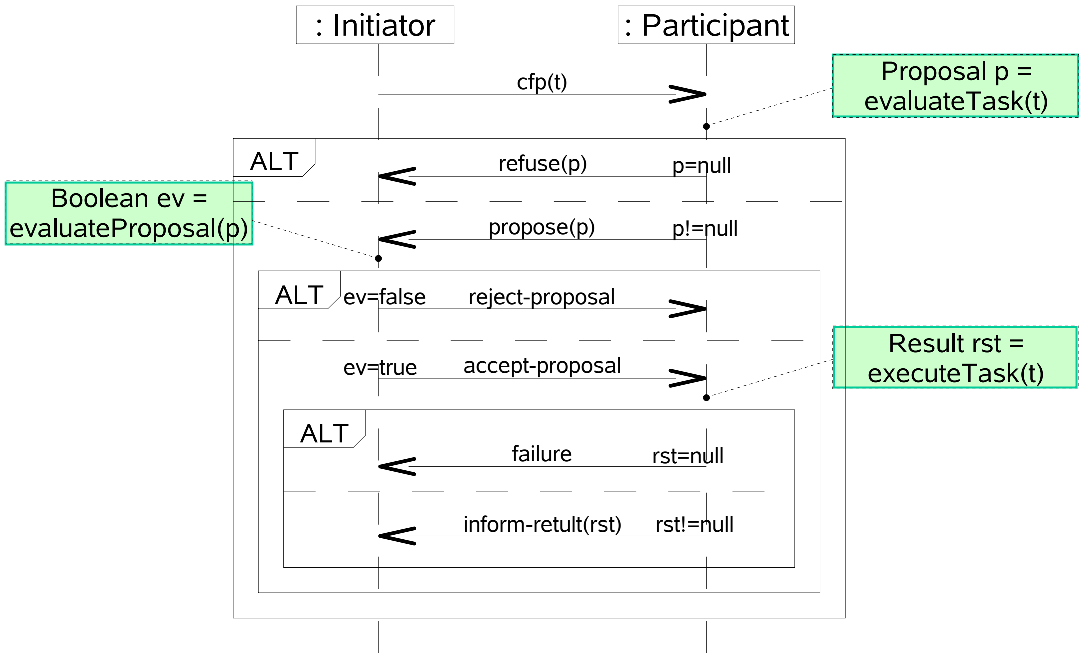


Capability requirements

- In literature we can find a similar concept:
  - Jade Platform [Bellifemine, Poggi, Rimassa]
  - PowerJava [Baldoni, Boella, Van der Torre - ProMAS 2005, SAC 2006]
- The term “capability” has been used (with a different meaning) by Padgham in the BDI framework (ability to achieve a goal)

Interaction Protocols and Capabilities: an example

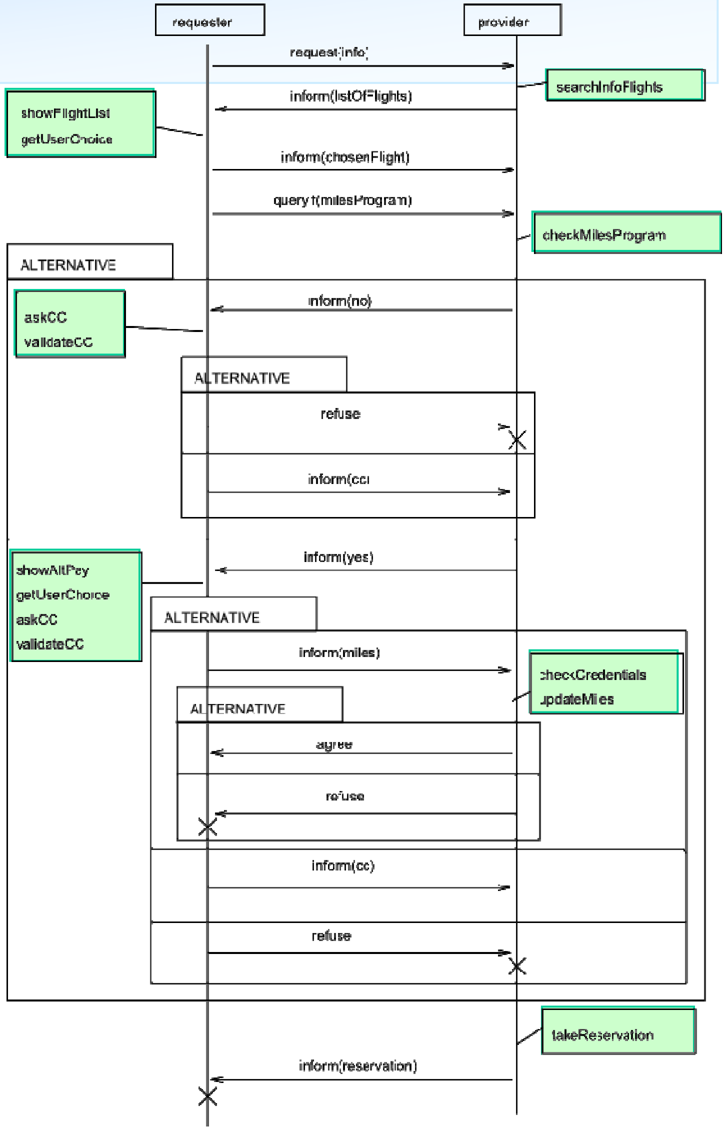
Fipa ConctractNet Protocol



- The initiator must be able to **evaluate a proposal**
- A participant must be able to **evaluate a task** and **execute a task**

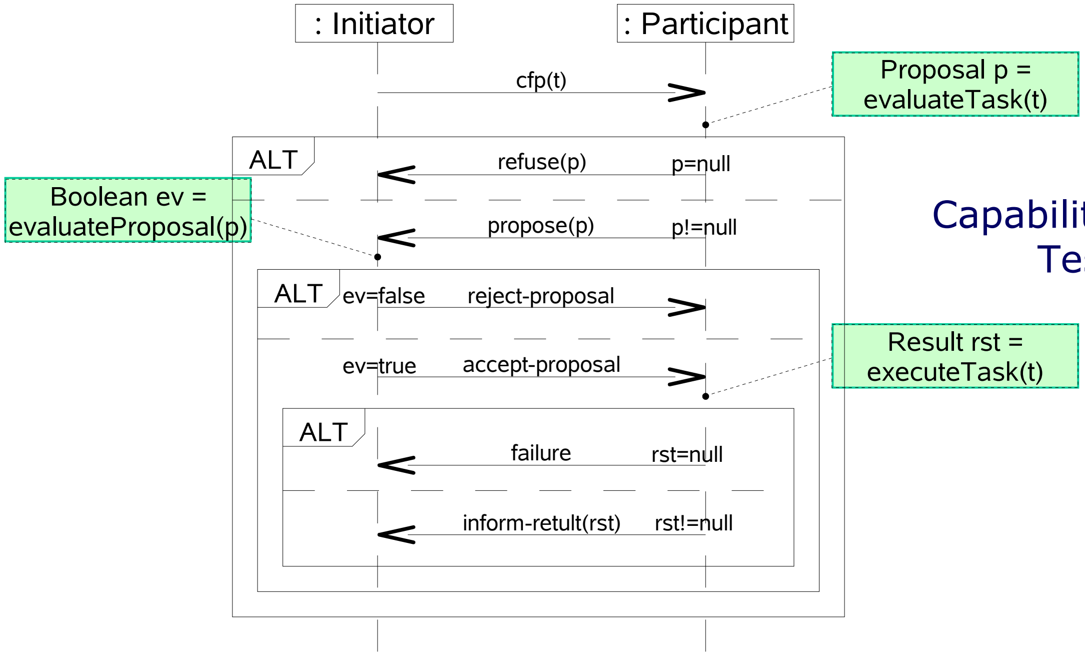
Interaction Protocols and Capabilities: an example

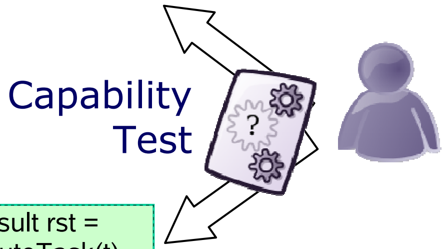
Flight company reservation



Checking capabilities

As for the conformance, an entity can execute a capability test





Checking capabilities

Different matching techniques can be used in the capability test

- Signature matching
  - simple
  - not-flexible
  - used also in PowerJava
- Semantic matchmaking
  - developed for semantic Web Services discovery
  - based on ontologies of concepts
  - support matching between different names and numbers/types of input/output parameters

Checking capabilities

Semantic matchmaking approaches:

- based on DAML-S proposed by Paolucci et al.
  - ontological reasoning is applied to input and output parameters
  - search is not goal-driven
- we ca use similar techniques for checking capabilities

Checking capabilities

Semantic matchmaking approaches:

- Web Services Modeling Ontology (WSMO) by Keller et al.
  - services are described by preconditions, assumptions, effects and postconditions ("capability" construct)
  - users can look for a service by specifying a goal described by means of the desired preconditions

Checking capabilities

WSMO can be used in our approach:

- a capability requirement inside an Interaction Protocol can be represented as a WSMO goal
- actions owned by an entity can be described by means of a WSMO capability construct

$\Downarrow$

We can apply existing matching techniques in capability test, enabling **goal-driven forms of reasoning**

Reasoning on capabilities

- The answer of a matchmaker has a local scope
- A goal-driven reasoning require the simulation of the execution of the policy, introducing a notion of state
- The choice of a capability could prevent the application of another capability

⇓

- We propose to combine the local matchmaking process with a global reasoning process

Reasoning on capabilities

- We propose to:
  - use a *declarative representation* for the policies and the roles
  - represent capabilities and capability requirements as *actions*

Reasoning on capabilities

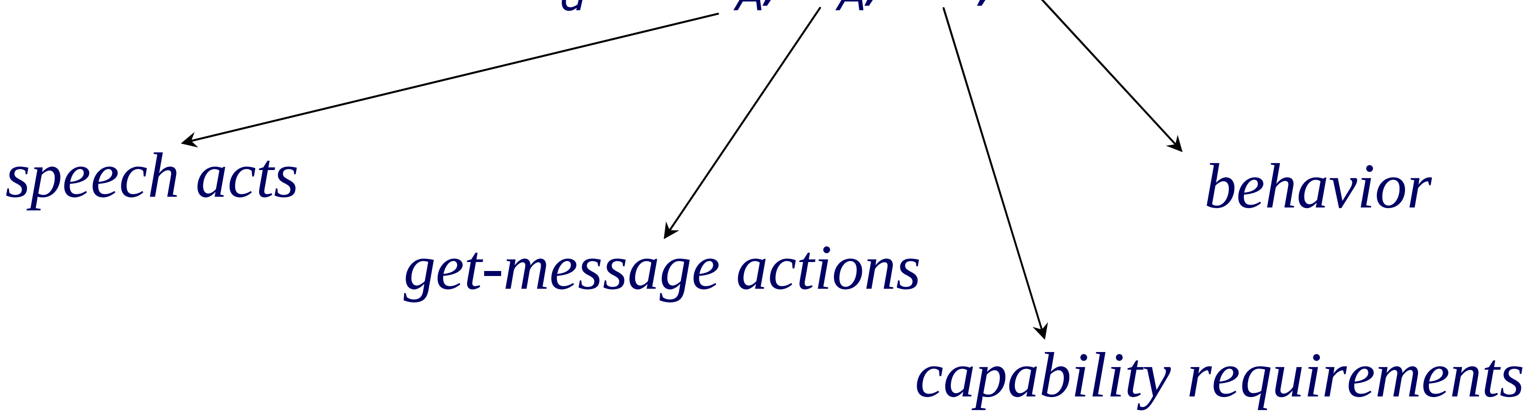
DyLOG:

- Language for programming communicating agents in a multi agent context
  - speech acts
  - get-message actions
  - policies
- Based on a logical theory for reasoning about actions and change in a modal logic programming setting

Representing a role

- Each role in a choreographies is represented as a subjective view by a DyLOG procedure R_d

$$R_d = \langle S_A, G_A, CR, P \rangle$$



Representing a role

- Speech acts

  performative(sender, receiver, l) causes E_1 if $Cond_1$
  …
  performative(sender, receiver, l) causes E_n if $Cond_n$
  performative(sender, receiver, l) possible if P

- Get-message actions

  receive_act(receiver, sender, l) receives I

- Capability requirements

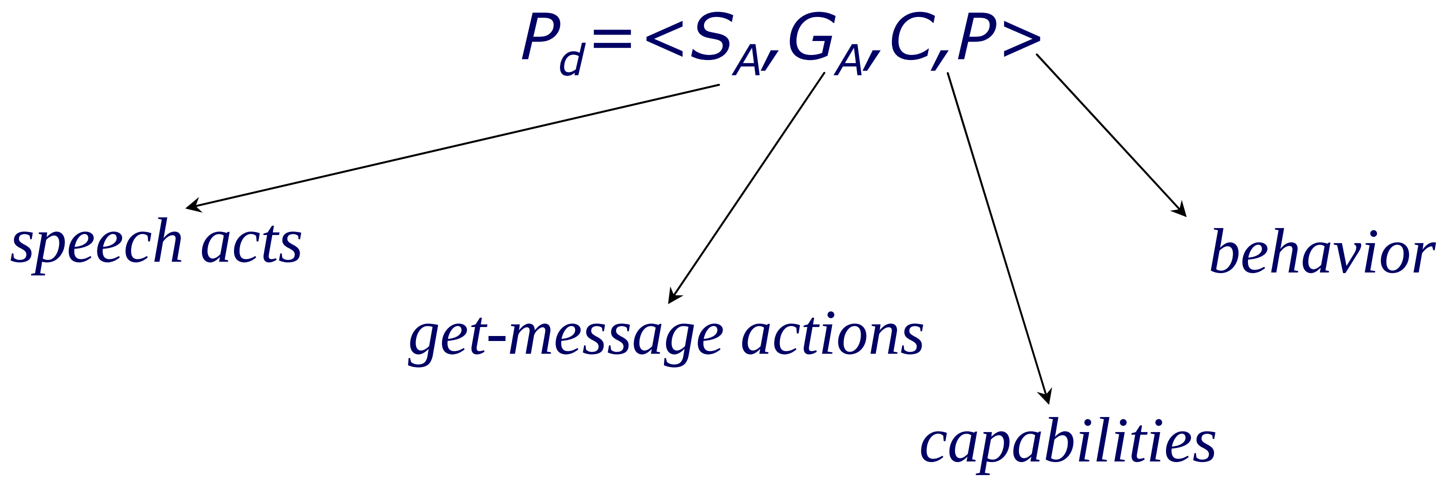
  c causes E_1 if $Cond_1$
  …
  c causes E_m if $Cond_m$
  c possible if P

- Behavior

  P_0 is $p_1,\ldots p_n$ $(n\geq 0)$

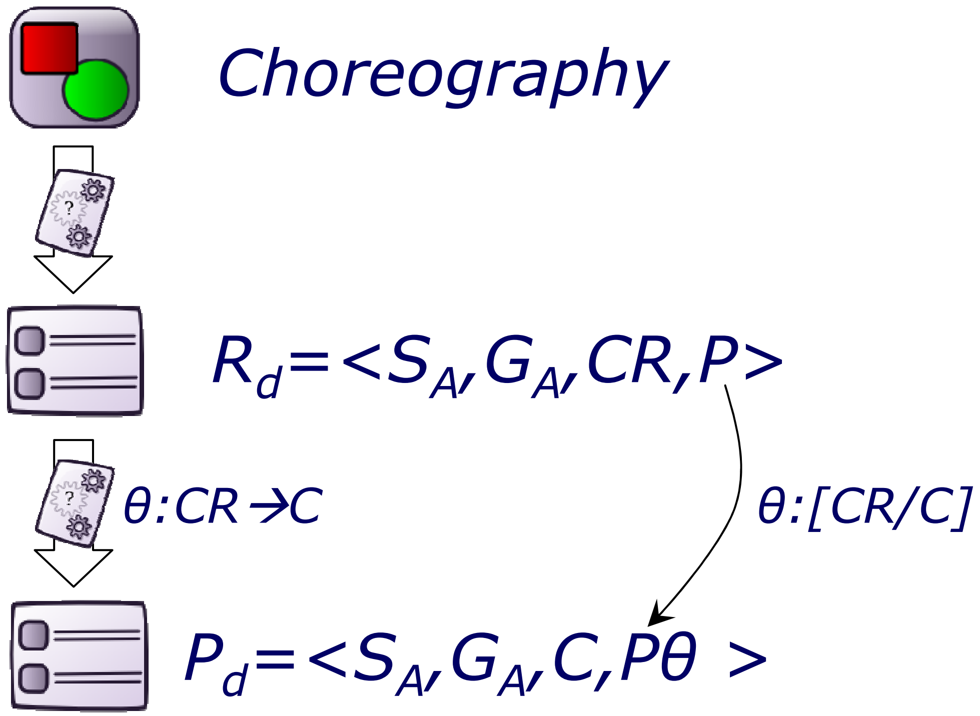
Representing a policy

- A policy represents an implementation of a role. We represent it in DyLOG as a procedure P_d



Reasoning on capabilities

- We want to build a policy P_d starting from a role description R_d in a (semi) automatic way:



Choreography

$$R_d = <S_A, G_A, CR, P>$$

$\theta : CR \rightarrow C$ $\qquad$ $\theta : [CR/C]$

$$P_d = <S_A, G_A, C, P\theta>$$

Reasoning on capabilities

- The matchmaking process has a local scope and it couldn't preserve global properties
- The selected *θ must also guarantee the achievement of the goal*

$$(R_d = \langle S_A, G_A, CR, P \rangle, S_0) \vdash G \xrightarrow{} F_s \text{ after } p$$

$$\exists \theta = [C/CR] \text{ s.t. } (\langle S_A, G_A, CR, P \rangle, S_0) \vdash G \Rightarrow (\langle S_A, G_A, C, P\theta \rangle, S_0) \vdash G$$

Reasoning on capabilities

- Must the entity have **all** the capabilities required for the role?

$$\exists\sigma, \theta' = [C/CR_\sigma],\ CR_\sigma \subseteq CR \text{ s.t.}$$

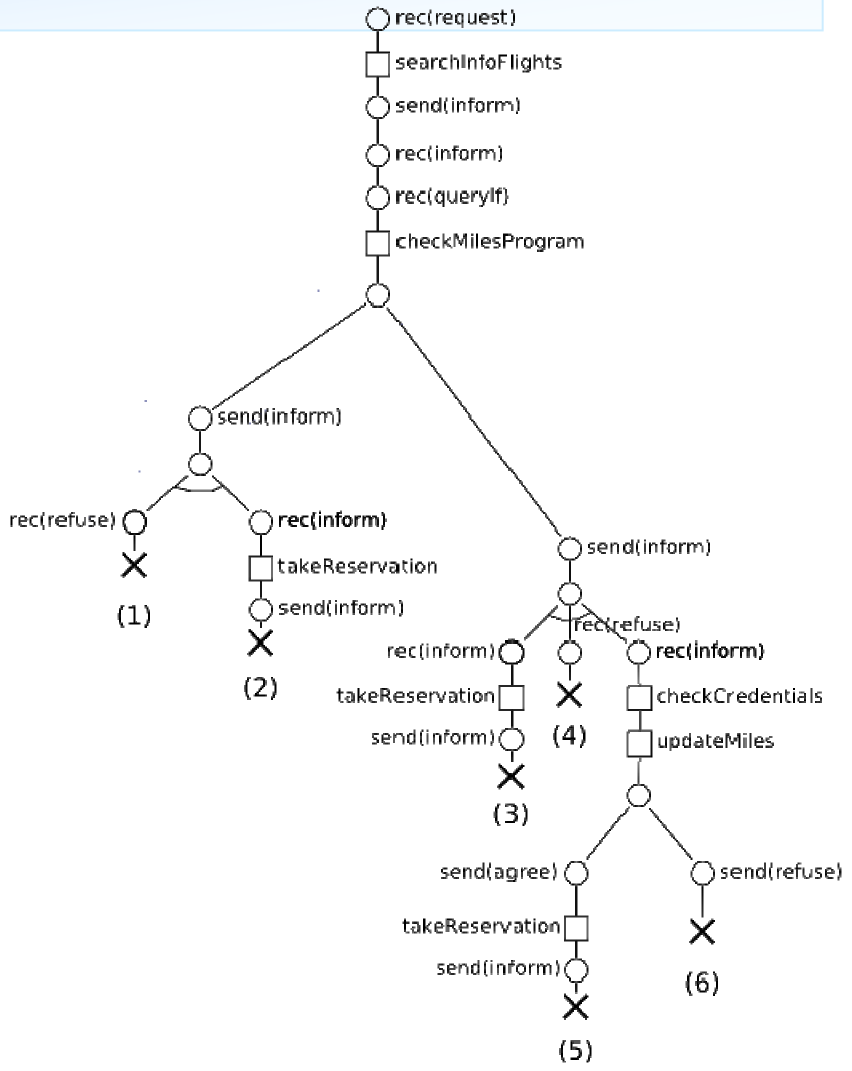
$$(\langle S_A, G_A, CR, P\rangle, S_0) \vdash G \text{ w.a. } \sigma \Rightarrow$$

$$(\langle S_A, G_A, C, P\theta'\rangle, S_0) \vdash G \text{ w.a. } \sigma\theta'$$

- Reasoning on global properties may also influence the matchmaking phase

Reasoning on capabilities

- A provider wants to sell some tickets
- It can only handle credit card payments



Reasoning on capabilities

- Moreover, the set of capabilities of a peer could depend on the context:

$$\exists \sigma, \theta'' = [C'/CR_\sigma],\ C' \subseteq C,\ CR_\sigma \subseteq CR \text{ s.t.}$$

$$(\langle S_A, G_A, CR, P\rangle, S_0) \vdash G \textit{ w.a. } \sigma \Rightarrow$$

$$(\langle S_A, G_A, C', P\theta''\rangle, S_0) \vdash G \textit{ w.a. } \sigma\theta''$$

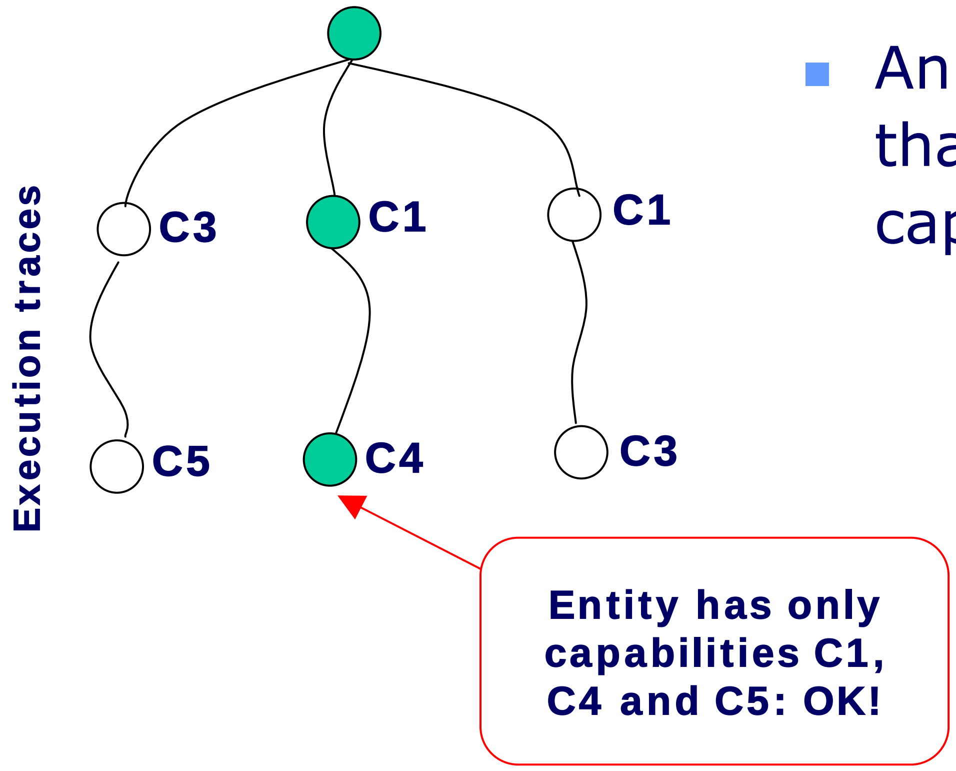
Reasoning on capabilities

Let us consider the interaction from the perspective of a given role, the role that the entity wants to play

- Must the entity synthesize a policy that implements **all** paths foreseen by it?
- Must the entity have **all** the capabilities required for the role?

Reasoning on capabilities

It is possible to perform some forms of customization



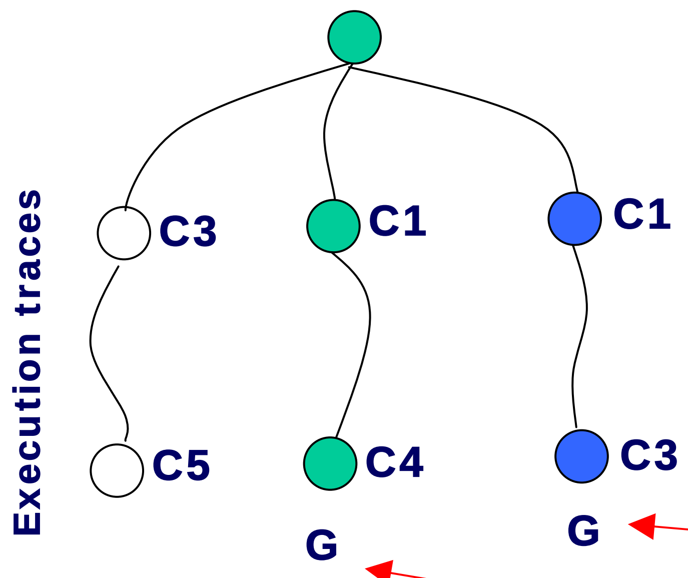
Entity has only capabilities C1, C4 and C5: OK!

- An entity can find paths that contains only the capabilities owned by it.

Reasoning on capabilities

It is possible to perform some forms of customization

- An entity can apply reasoning (e.g. Procedural planning) to choose paths that allow to reach a given goal and that contain only the owned capabilities



Entity wants to reach goal G and has only capabilities C1, C4 and C5: OK!

Capability requirements in WS-CDL

- **WS-CDL+C**: an extension of the WS-CDL language that includes capability requirements representation
- Capabilities represent operations performed by an entity which are non-observable by other entities, like SilentAction elements in WS-CDL
- Capability requirements are expressed (in a general way) by means of
  - input and output parameters
  - preconditions and goals

Capability requirements in WS-CDL

- Specified in a “capabilityRequirement” tag
- Gathered in the “capabilitySection” tag inside the package element

```
<cdl:capabilitySection>

  <cdl:capabilityRequirement name="evaluateTask">
    <cdl:input>?task</cdl:inputs>
    <cdl:output>?proposal</cdl:outputs>
    <cdl:preconditions>executable(?task)
    <cdl:effects>proposed(?proposal,?cost,?time)</cdl:effects>
  </cdl:capabilityRequirement>

  <cdl:capabilityRequirement name="executeTask">
    .......
  </cdl:capabilityRequirement>

  <cdl:capabilityRequirement name="evaluateProposal">
  ........
  </cdl:capabilityRequirement>

</cdl:capabilitySection>
```

Capability requirements in WS-CDL

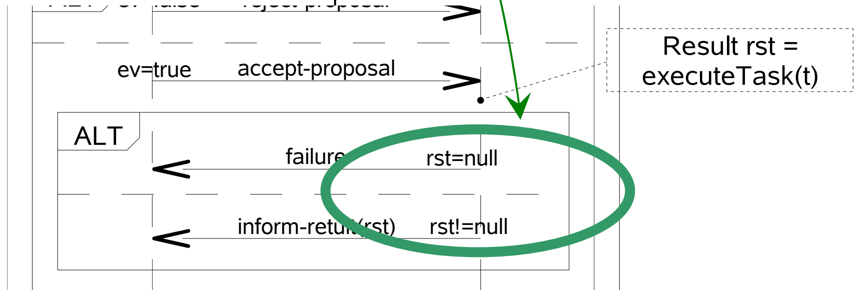
- Binding between capability requirement variables and ws-cdl variable are specified in a silent action element

```
<cdl:silentAction roleType="tns:Participant">
  <cdl:capabilityRequirementInstace name="evaluateTask">
    <cdl:variableBind>
      <cdl:cdlVariable>tns:task</cdl:cdlVariable>
      <cdl:capabilityVariable>?task</cdl:capabilityVariable>
    </cdl:variableBind>
    ...
  </cdl:capabilityRequirementInstace>
</cdl:silentAction>
```

Capability requirements in WS-CDL

- Input and output parameters, preconditions and effects variables can be used in the whole documents in standard ways (Interaction, Workunit, etc)

```
<choice>
  <workunit name="informResultWorkUnit"
    guard="cdl:getVariable('rst', '', '',
'Participant') != ‘null' ">
    <interaction name="informResultInteraction">
      ...
    </interaction>
  </workunit>
  <interaction name="failureExecuteInteraction">
    ...
  </interaction>
</choice>
```



Conclusions

This proposal:

- extends the specification of Interaction Protocols by means of “requirements of capabilities”
- allows an entity to improve its interoperability by synthesizing a new policy in a semi-automatic way
- permits entities to exploit reasoning techniques for customizing the policy synthesis

Future works

- More thorough formalization of the proposal
- Integration with a matchmaking (WSMO?) approach for checking capabilities
- Understand how the global reasoning can influence matchmaking process
- Design and implementation of a system in order to check the feasibility of the proposed approach