

# A semantic based framework for supporting negotiation in Service Oriented Architectures

**Kyriakos Kritikos** and **Pierluigi Plebani**

Politecnico di Milano

&

**Marco Comuzzi**

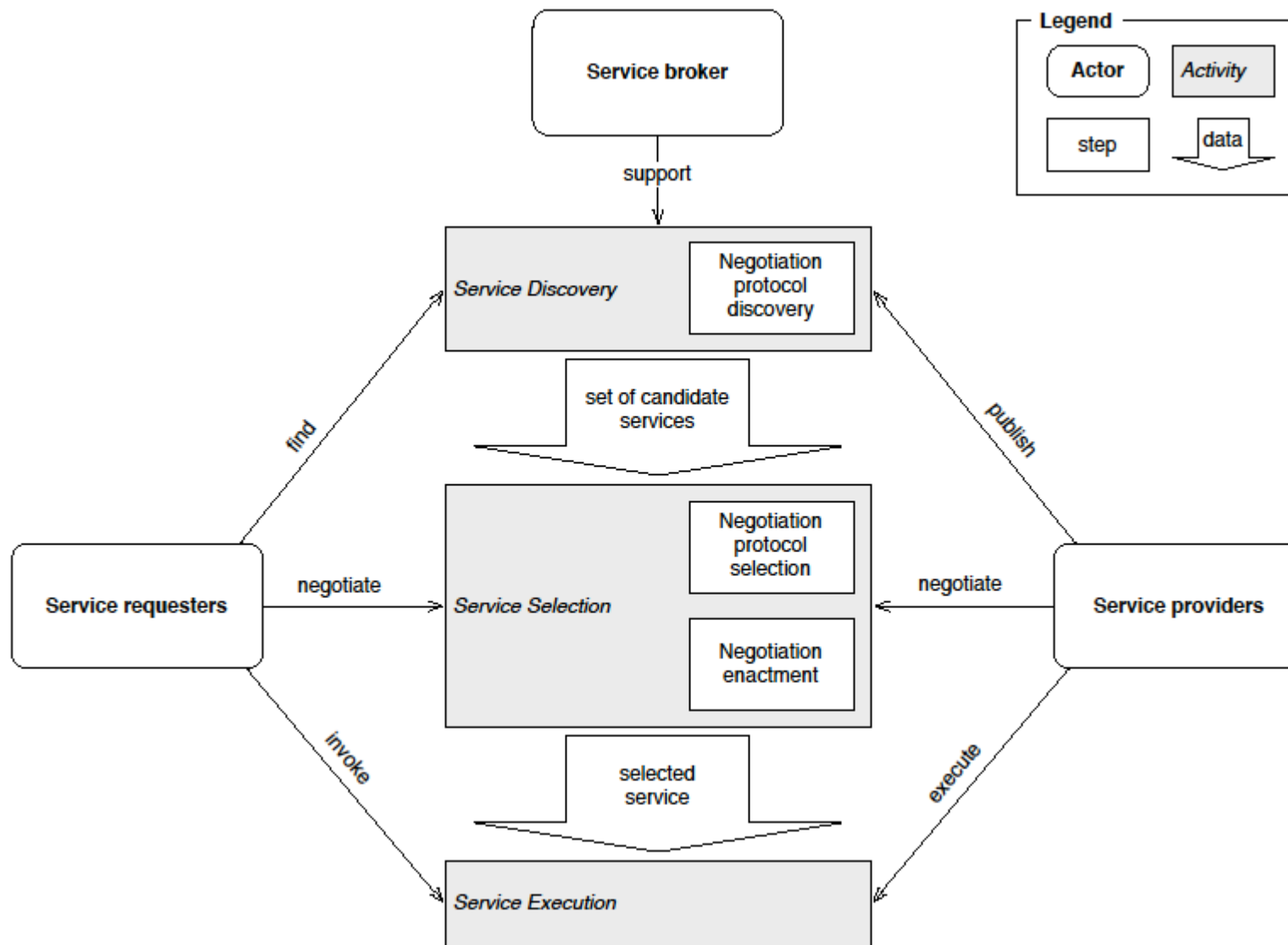
City University London

# Agenda



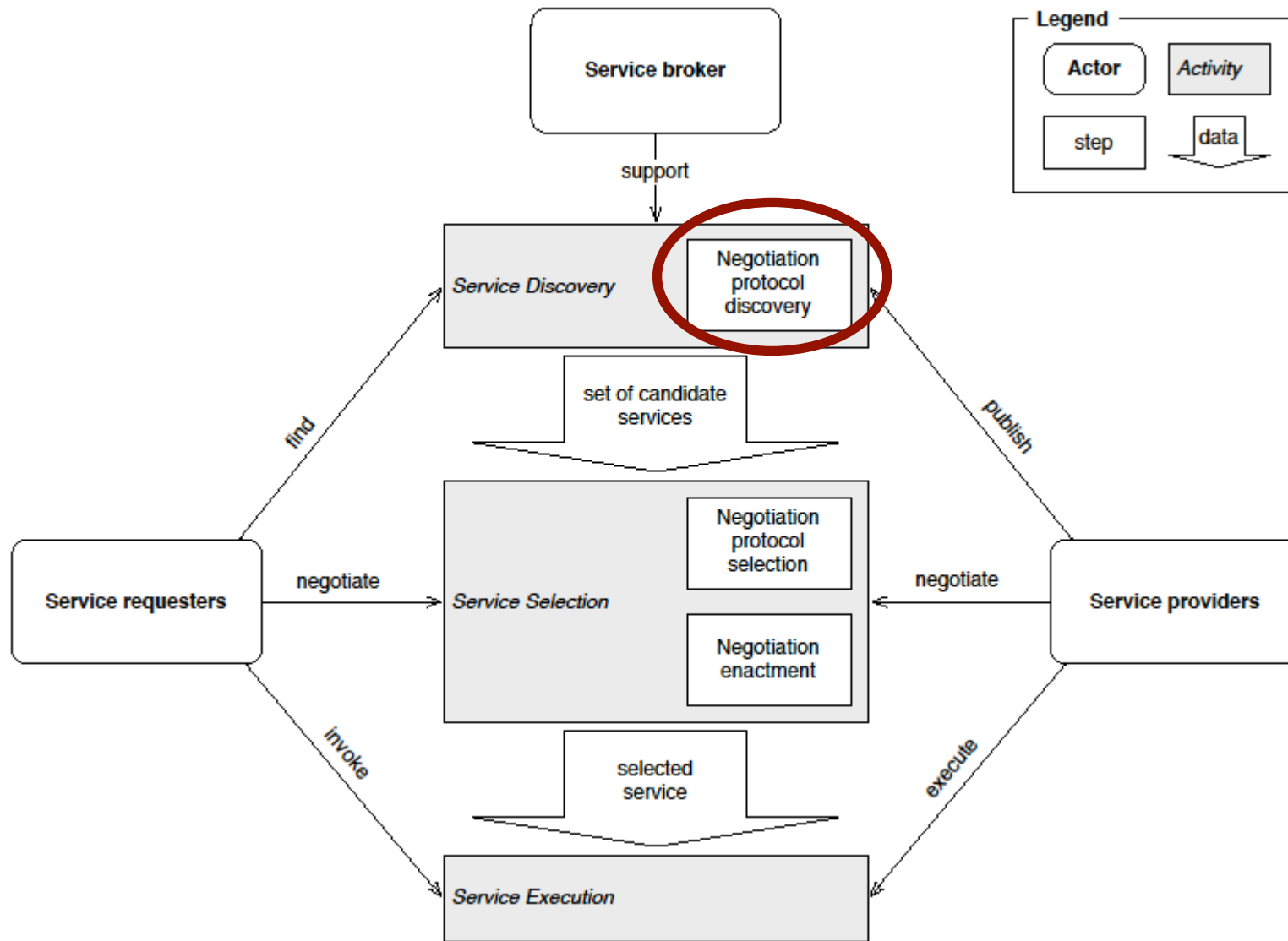
- Reference scenario
- The problem
- The negotiation framework
- An example in case of bilateral negotiation
- Concluding remarks and future work

# Reference scenario



- Facts:
  - Service discovery produces a set of available services for each requester
  - Negotiation is required before (or after) selecting the best service for each requester and establish the SLA
- Problems:
  - Current solutions assume all negotiation participants agree and are able to support the same negotiation protocol
    - Under a closed-world assumption the negotiation protocol can be selected at design time
    - Under a open-world assumption the negotiation protocol should be selected at run-time

# Focus of this work



# Starting point



- In our previous work [1], we extended OWL-Q [2] by focusing:
  - on the *negotiation objects*
  - on the negotiators' *decision models*

in order to:

- automate the service quality specification alignment and matchmaking processes
- assist the actual negotiation process through reasoning with rules
  - showed how to infer *negotiation compatibility* of service quality offers and demands



# Main idea



- Participants should advertise their negotiation capabilities in a more fine-grained way
- Matching of negotiation capabilities and of actions required by negotiation protocols is needed to find a negotiation protocol
- Three cases:
  - **Exact match:** a negotiation protocol is supported by all the participants
  - **Negotiation by delegation:** a negotiation protocol can be indirectly found if some negotiation participants can fully or partially delegate the missing capabilities to other cooperating actors
  - **Failed match:** a negotiation protocol cannot be found

# Negotiation Conceptual Model (1/2)



- Negotiation actors: user or agents
- Role skeleton in bilateral negotiation:
  - **Initiator**, able to generate the first offer.
  - **Participant**, able to send and receive offers, to generate counter offers, and to receive the outcome of the negotiation.
  - **Decision maker**, able to accept an offer and generate the outcome of a negotiation.
- - **send offer (so)**
  - **receive offer (ro)**
  - **create counter offer (cco)**

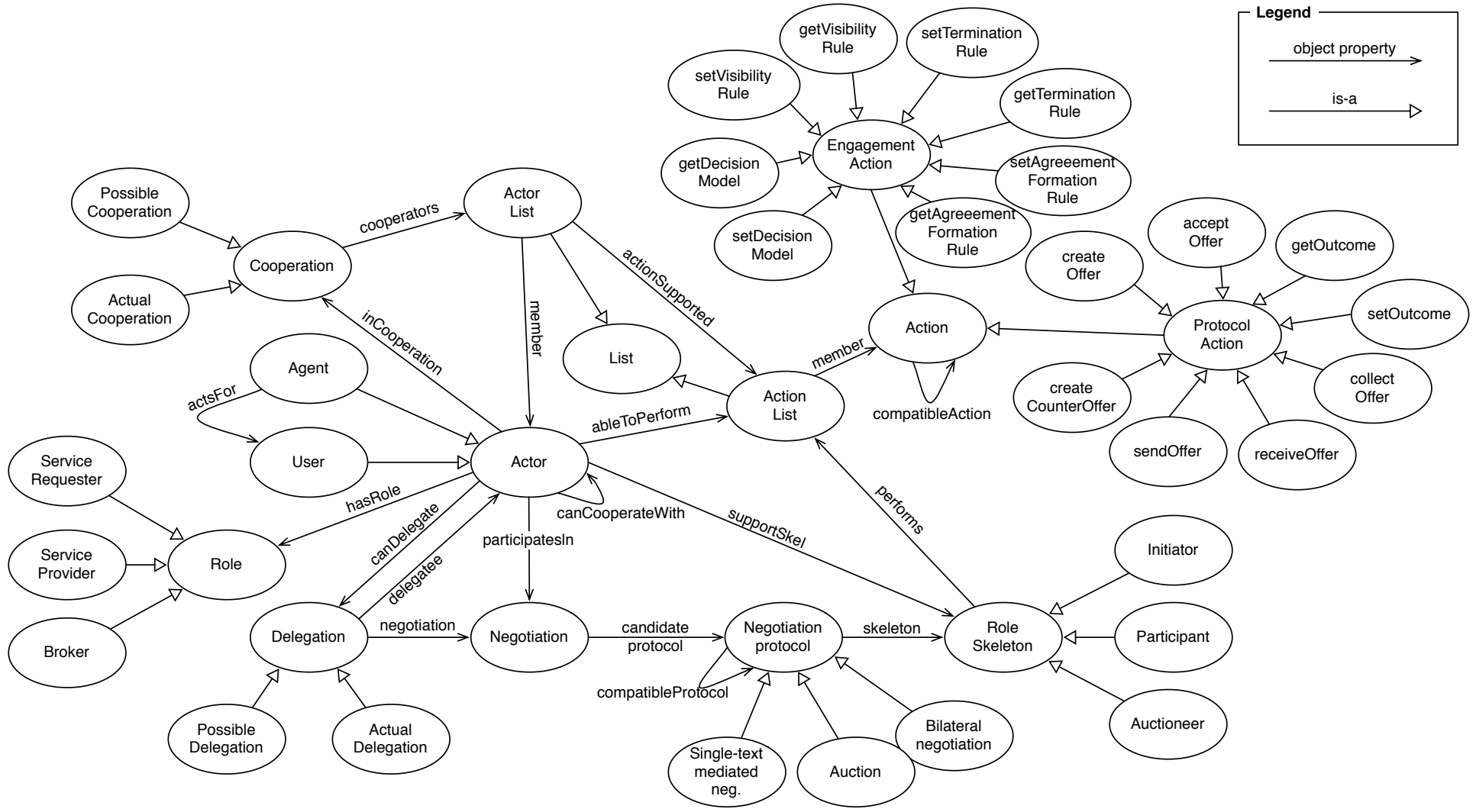
# Negotiation Conceptual Model (2/2)



- Two types of delegation
  - Full: An actor delegates all actions to another actor
  - Partial: An actor delegates some of the actions to one or more cooperating actors
  
- Two possible situations
  - Federation: relationships among the actors are more strict; we can assume that actors trust in some way
  - Marketplace: relationships among the actors are more loose; not all the data about negotiation could be exchanged

- Semantic framework for negotiation protocol discovery, consisting of:
  - A reference ontology: a new OWL-Q extensions for describing the negotiation capabilities of the participants
  - A set of basic rules: for matchmaking the capabilities of the participants with the actions required by a negotiation protocol irrespective of any negotiation
  - A set of additional rules: for selecting a specific protocol when participants are involved in a negotiation
    - Only for bilateral negotiation until now but plan for all other common negotiation protocols

# Reference ontology



1.  $fSupportsSkel(A, S) \leftarrow ableToPerform(A, L_1) \wedge performs(S, L_2) \wedge subList(L_2, L_1)$
2.  $delegatesSkel(A, S) \leftarrow actsFor(A_2, A) \wedge fSupportsSkel(A_2, S)$
3.  $PossibleDelegation(D) \wedge canDelegate(A, D) \wedge delegatee(D, A_2) \wedge skel(D, S) \leftarrow$   
 $\leftarrow cooperatesWith(A, A_2) \wedge supportsSkel(A_2, S)$
4.  $PossibleCooperation(C) \wedge inCooperation(A, C) \wedge cooperators(C, L) \wedge ofSkeleton(C, S) \leftarrow$   
 $\leftarrow supportSkel(L, S) \wedge contains(L, A) \wedge \forall X(member(L, X) \wedge differentFrom(X, A) \rightarrow$   
 $\rightarrow canCooperateWith(X, A))$
5.  $candidateProtocol(N, P) \leftarrow newList(LS) \wedge \forall R(ofNegotiation(R, N) \wedge \exists S(skeleton(P, S) \wedge$   
 $\neg member(LS, S) \wedge \forall X(participatesIn(X, N) \wedge takesRole(X, R) \wedge (supports(X, S) \vee ($   
 $canDelegate(X, D) \wedge delegatee(D, X_2) \wedge skel(D, S) \wedge \neg participatesIn(X_2, N))) \vee ($   
 $inCooperation(X, C) \wedge cooperators(C, L) \wedge ofSkeleton(C, S) \wedge \neg \exists X_2(member(L, X) \wedge$   
 $differentFrom(X_2, X) \wedge participatesIn(X_2, N)))) \wedge listAdd(LS, S)))$   
 $\wedge \forall S(skeleton(P, S) \wedge member(LS, S))$

## Rules for inferring the candidate protocols

6.  $pSupportsSkel(A, S) \leftarrow ableToPerform(A, L_1) \wedge performs(S, L_2) \wedge$   
 $\wedge listIntersection(L_3, L_1, L_2) \wedge strictlySubList(L_3, L_2)$
7.  $pSupportSkel(AL, S) \wedge actionsSupported(AL, L) \leftarrow pSupportsSkel(A, S) \wedge$   
 $\wedge ableToPerform(A, L) \wedge listAdd(AL, A)$
8.  $pSupportSkel(AL_2, S) \wedge actionsSupported(AL_2, L_4) \leftarrow pSupportSkel(AL_1, S) \wedge$   
 $\wedge pSupportsSkel(A, S) \wedge ableToPerform(A, L_1) \wedge actionsSupported(AL_1, L_3) \wedge$   
 $\wedge listConcatenation(L_4, L_1, L_3) \wedge performs(S, L_2) \wedge listIntersection(L_5, L_4, L_2) \wedge$   
 $\wedge strictlySubList(L_5, L_2) \wedge newList(AL_2, AL_1, A)$
9.  $supportSkel(AL_2, S) \wedge actionsSupported(AL_2, L_4) \leftarrow pSupportSkel(AL_1, S) \wedge$   
 $\wedge (A, S) \wedge ableToPerform(A, L_1) \wedge actionsSupported(AL_1, L_3) \wedge$   
 $\wedge listConcatenation(L_4, L_1, L_3) \wedge performs(S, L_2) \wedge subList(L_2, L_4) \wedge newList(AL_2, AL_1, A)$

## Rules for inferring supportSkel

# Basic rules



$fSupportsSkel(A, S) \leftarrow ableToPerform(A, L_1) \wedge performs(S, L_2) \wedge subList(L_2, L_1)$

An actor  $A$  fully support a skeleton  $S$  when he is able to perform all the actions ( $L_1$ ) that the role skeleton should perform ( $L_2$ )

$delegatesSkel(A, S) \leftarrow actsFor(A_2, A) \wedge fSupportsSkel(A_2, S)$

An actor  $A$  delegates a skeleton  $S$  when he has an agent  $A_2$  that sully supports the skeleton

# Basic rules



$candidateProtocol(N, P) \leftarrow newList(LS) \wedge \forall R(ofNegotiation(R, N) \wedge \exists S(skeleton(P, S) \wedge \neg member(LS, S) \wedge \forall X(participatesIn(X, N) \wedge takesRole(X, R) \wedge (supports(X, S) \vee (canDelegate(X, D) \wedge delegatee(D, X_2) \wedge skel(D, S) \wedge \neg participatesIn(X_2, N))) \vee (inCooperation(X, C) \wedge cooperators(C, L) \wedge ofSkeleton(C, S) \wedge \neg \exists X_2(member(L, X) \wedge differentFrom(X_2, X) \wedge participatesIn(X_2, N)))))) \wedge listAdd(LS, S)))) \wedge \forall S(skeleton(P, S) \wedge member(LS, S))$

- to assign to the participants taking the same role in the negotiation the same role skeleton;
  - to assign different role skeletons to participants of different roles;
  - to assign all role skeletons of a negotiation protocol.
- Thus, when this goal is satisfied, the negotiation protocol can be used for enacting the negotiation.

$$\begin{aligned} & \text{supportSkel}(AL_2, S) \wedge \text{actionsSupported}(AL_2, L_4) \leftarrow \text{pSupportSkel}(AL_1, S) \wedge \\ & \wedge (A, S) \wedge \text{ableToPerform}(A, L_1) \wedge \text{actionsSupported}(AL_1, L_3) \wedge \\ & \wedge \text{listConcatenation}(L_4, L_1, L_3) \wedge \text{performs}(S, L_2) \wedge \text{subList}(L_2, L_4) \wedge \text{newList}(AL_2, AL_1, A) \end{aligned}$$

- states that an actor list  $AL_2$  supports a role skeleton  $S$  and supports a set of actions  $L_4$ , when
  - it is produced by an actor list  $AL_1$  that partially supports  $S$  and
  - an actor  $A$  that also partially supports  $S$  and
  - the union  $L_4$  of the action sets that  $AL_1$  and
  - $A$  support is a super-set of the action set that  $S$  requires.

# Example for bilateral negotiation



*Action(co), Action(so), Action(ro), Action(go), Action(cco), Action(ao), Action(sto), Actor(p), ableToPerform(p, [co, so, ro, go]), Actor(h<sub>1</sub>), ableToPerform(h<sub>1</sub>, [cco, so, ro]), cooperatesWith(p, h<sub>1</sub>), Actor(c), ableToPerform(c, [cco, ro, so, ao, sto]), Actor(h<sub>2</sub>), ableToPerform(h<sub>2</sub>, [cco, ro, so, ao, sto]), cooperatesWith(c, h<sub>2</sub>), BilateralNegotiation(bn), Initiator(i), performs(i, [co, cco, so, ro, go]), Participant(pa), performs(pa, [cco, ro, so, ao, sto]), Role(provider), Role(requester)*

- The actor p participates in the negotiation and takes the role of the service provider. The actor p is able to perform the actions co, so, ro, go.
- The actor c participates in the negotiation and takes the role of the service requester. The actor c is able to perform the following actions: cco, so, ro, ao, sto.
- The actor h1 does not participate in the negotiation and is able to perform the actions cco, so, ro.
- The actor h2 does not participate in the negotiation and is able to perform the actions cco, so, ro, ao, sto.

# Example for bilateral negotiation



*fSupportsSkel(c, pa), fSupportsSkel(h<sub>2</sub>, pa), PossibleDelegation(d), canDelegate(c, d),  
delegatee(d, h<sub>2</sub>), skel(d, pa), pSupportsSkel(p, i), pSupportsSkel(h<sub>1</sub>, i),  
pSupportSkel([p], i), actionsSupported([p], [co, so, ro, go]), supportSkel([p, h<sub>1</sub>], i),  
actionsSupported([p, h<sub>1</sub>], [co, cco, so, ro, go]), PossibleCooperation(pc),  
inCooperation(p, pc), cooperators(pc, [p, h<sub>1</sub>]), ofSkeleton(pc, i)*

- The service provider  $p$  does not fully support the Initiator role skeleton, but it may support it through the cooperation with actor  $h_1$  .
- The service requester  $c$  fully supports the Participant role skeleton, but it may also support it through a full delegation to actor  $h_2$  .

# Example for bilateral negotiation



*Negotiation(n), participatesIn(p, n), takesRole(p, provider), participatesIn(c, n), takesRole(c, requester), of Negotiation(provider, n), of Negotiation(requester, n)*

*candidateProtocol(n, bn)*

- Thus, the negotiation n between p and c actors can be enacted through the use of the bilateral negotiation protocol.

# Future Work



- Develop a set of rules for matchmaking the most common negotiation protocols apart from bilateral negotiation
- Extend our framework for selecting the best negotiation protocol among the candidate ones
- Analyze how to involve humans in the negotiation
- Create and evaluate a prototype implementation of our framework

# References



1. M. Comuzzi, K. Kritikos, P. Plebani, Semantic-aware Service Quality Negotiation, in Proc. of First European Conference, ServiceWave 2008, Madrid, Spain, December 10-13, 2008.
2. Kyriakos Kritikos, Dimitris Plexousakis: Semantic QoS Metric Matching. ECOWS 2006: 265-274.