Web Rule Languages to Carry Policies

Nima Kaviani
Laboratory for Ontological Research (LORe)
Simon Fraser University Surrey, Canada

nkaviani@sfu.ca
http://www.sfu.ca/~nkaviani

June 15th, 2007

Outline

- Policy-based Trust Management
  - Web services and Policies
  - Policy Languages
    - PeerTrust, KAoS, and Rei
  - The communication issues
- Interchange Frameworks
  - What is RIF?
  - What is R2ML
  - Using R2ML to exchange policies
  - The technical difficulties
  - The obtained results
- Conclusions
- Future Directions

Policy-Based Trust Management

- **Web Services and Policy-Based Trust Management**
  - **Web services to facilitate collaboration**
  - **Trust Management to be used by web services**
  - **Policies to regulate Trust Management**

- **Dynamically regulate the behavior of the system without any need to manipulate the internal code**

- **Policies as *Guiding Plans* that restrict the behavior of the agents**

- **To protect the privacy of information by providing different levels of access**

- **Policy Management Approaches and the Languages that support it**
  - **Role Based (XACML, Cassandra)**
  - **Context Based (KAoS, Rei)**

Policy Languages

- **Existing Languages for Policy-based Trust Management**
  - *PeerTrust*
  - *Rei*
  - *KAoS*
    - A DAML/OWL based policy language (KPO)
    - Robust, Adaptable, Extensible
    - Policy Specification and Management
    - Enforcement
    - A GUI for policy manipulation
    - Stanford's JTP to perform static conflict resolution, intelligent lookup; and dynamic policy refinement

1. Trust Negotiation Engine
2. Text-based EBNF
3. Rules are defined in the form of definite horn clauses

   lit0 ← lit1, lit2, …, litn where

   liti is a predicate ***pj(t1, …, tn)***
4. Low Expandability
5. Easy to understand

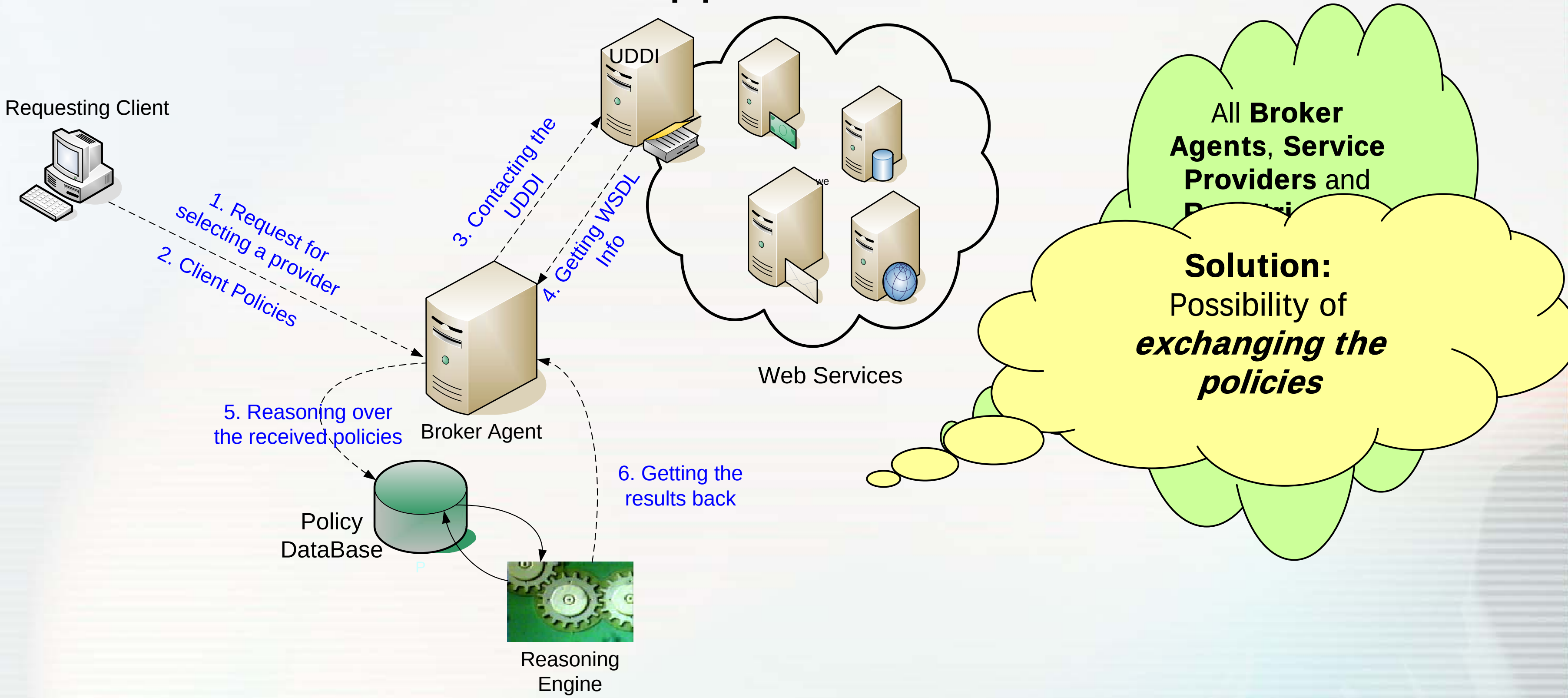
- Syntactically follows Description Logic (OWL-Lite)
- Semantically follows Computational Logic (Prolog)
- OWL as the meta-interpreter in the backend
- Global policy enforcement
- Speech Acts for message passing and dynamic
- Exchange of rights
- No policy disclosure possibility

Semantic Web Service Discovery & Composition

•The Current Proposals

- –*Combination of OWL-S and Rei* [Kagal, et. al, 2004]
- –*Combination of WSMO and PeerTrust* [Olmedilla et.al, 2004]

•Problems with the current approaches



[Kagal, et. al, 2004] Authorization and Privacy for Semantic Web services

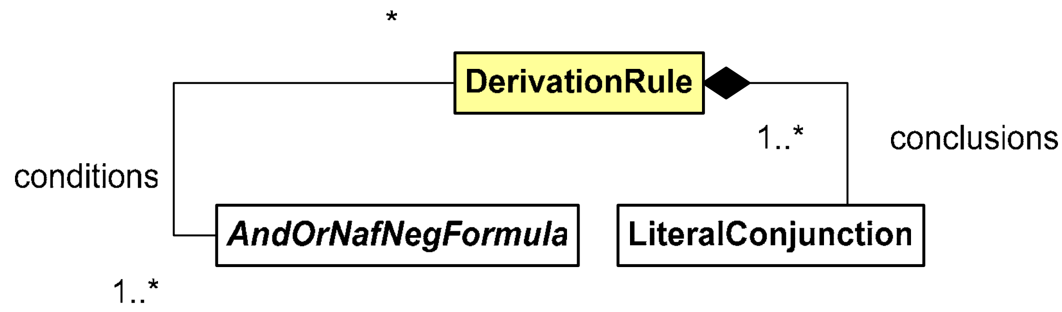
REWERSE Rule Markup Language (R2ML)

- **Rule Interchange Format (RIF)**
    - RIF working group: defining a rule interlingua based on W3C standards
    - *Develop a language to translate rules between rule languages and transform them between rule systems*
    - Goal: enabling existing rule technologies to interoperate

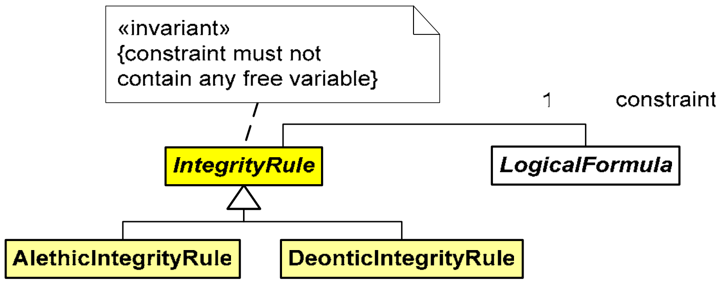
- **R2ML features**
    - – A general *rule interchange* language
    - – Admits to the RIF requirements
    - – http://rewerse.net/I1/
    - – Current version 0.4

R2ML cnt'd

- **Five General Rules**
  - *Integrity Rules*
  - *Derivation Rules*
  - *Production Rules*
  - *Reaction Rules*
  - *Transformation Rules*



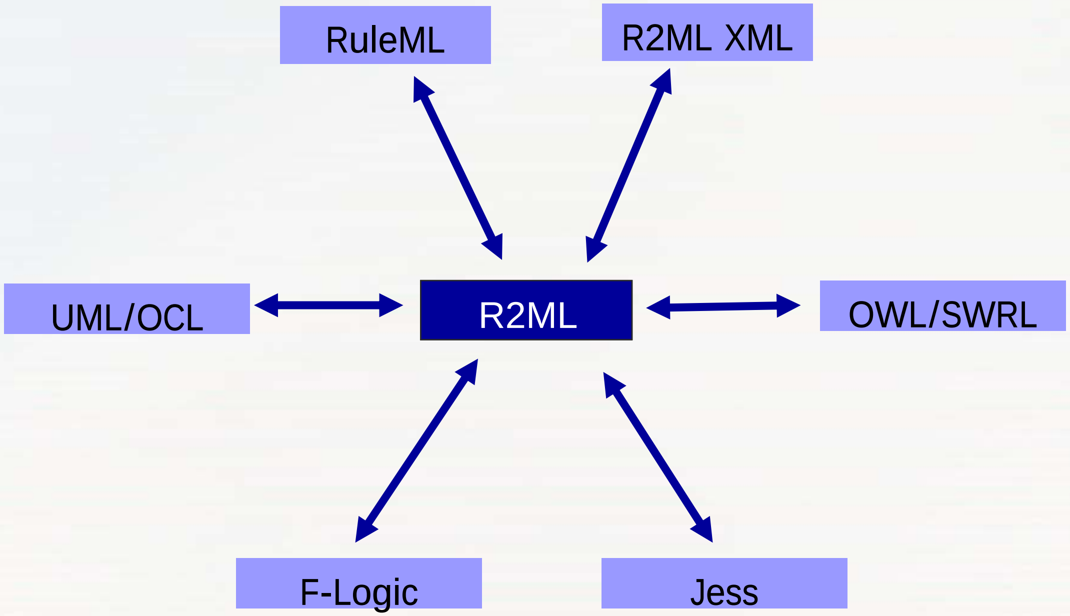
if the user is a faculty
then give him/her access to the meeting room



if a visitor is part of a patients family
then give him/her the allowance of visiting the patient

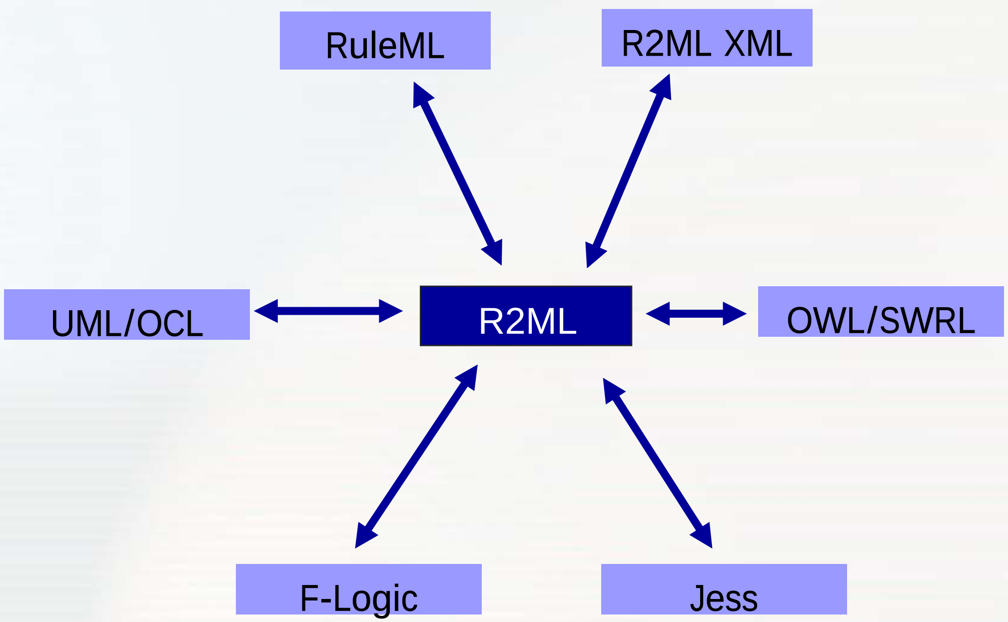
R2ML cnt'd

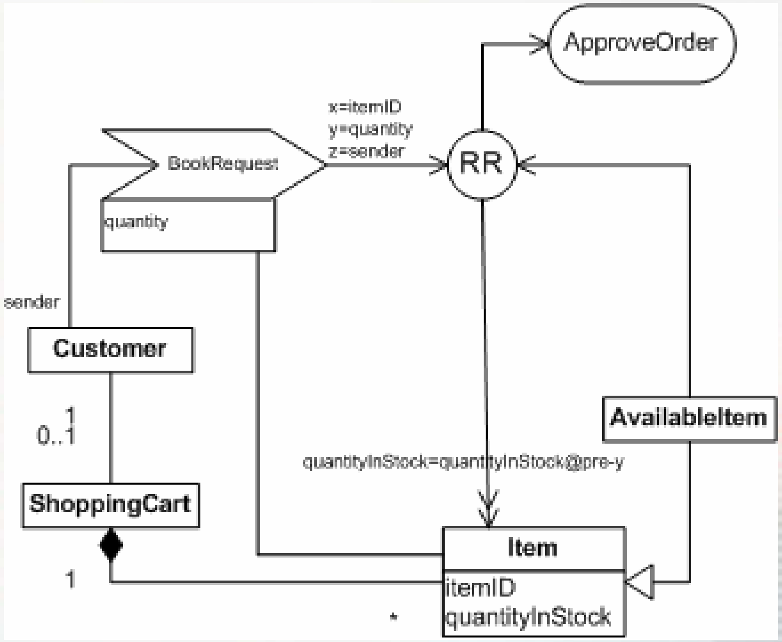
- **Current Transformations to/from R2ML**
  - **R2ML as a pivotal MetaModel**



R2ML cnt'd

- Current Transformations to/from R2ML
  - R2ML as a pivotal MetaModel
  - URML: UML based rule language with graphical notations



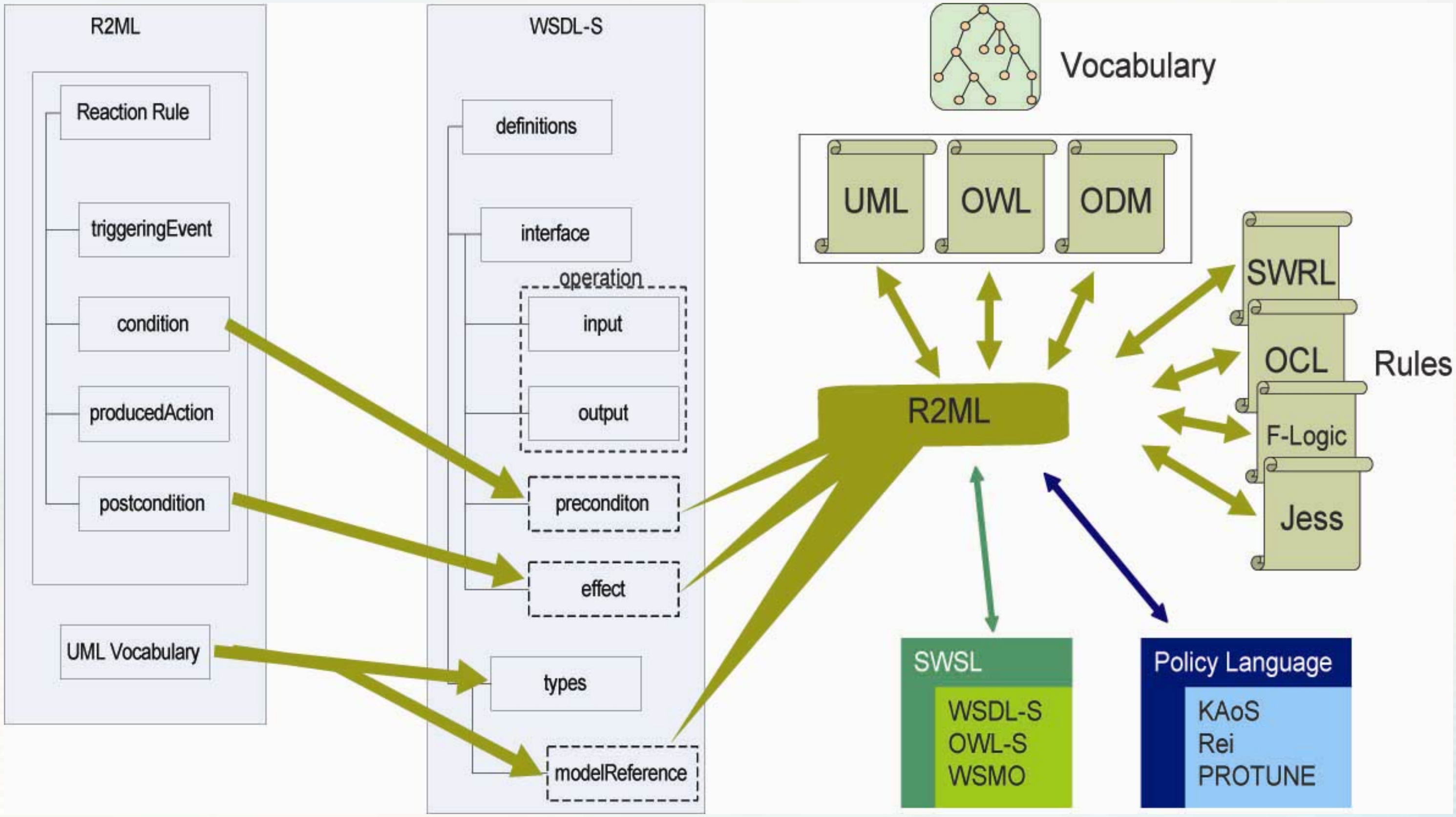


Semantic Web Service Discovery

Solution

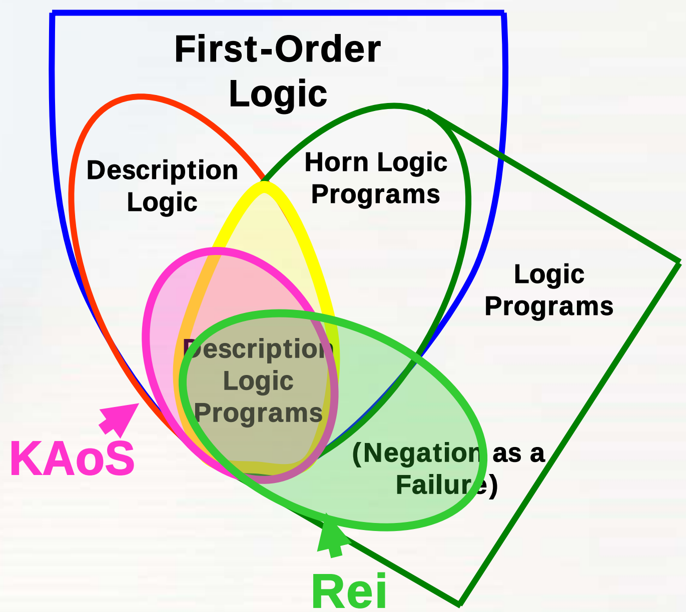
Enabling involved entities in Semantic Web Service discovery procedure to communicate

Policies can be defined in the form of R2ML rules



To get KAoS and Rei agents to communicate

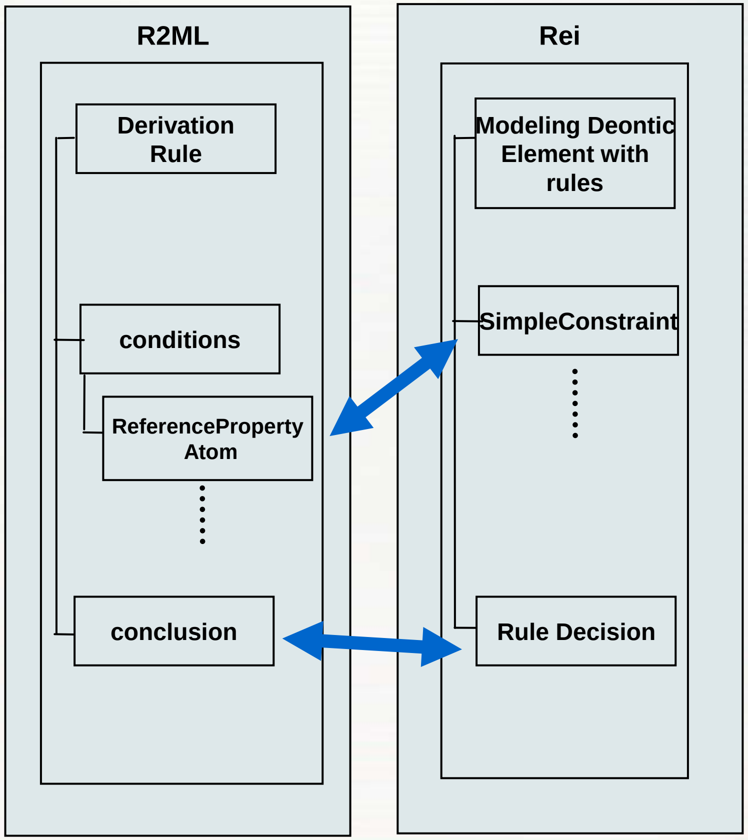
- Providing transformations between ***KAoS*** and ***Rei*** [Grosof, et. al, 2003]
  - Both are Context-Based policy languages
  - Both syntactically follow Ontology Languages
  - No straightforward mapping between **Rei** and **KAoS**
  - ***KAoS*** is based on ***Description Logic***
  - ***Rei*** follows ***Computational Logic (Logic Programs)***



OWL Constructor	DL Syntax	FOL Expressions
subClassOf	$C \subseteq D$	$D \leftarrow C$
transitiveProperty	$P^+ \subseteq P$	$\forall x,y,z(P(x,y) \wedge (P(y,z)) \rightarrow P(x,z)$
inverseOf	$P \equiv Q^-$	$\forall x,y P(x,y) \Leftrightarrow Q(y,x)$
intersectionOf	$C_1 \cap \ldots \cap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \cup \ldots \cup C_n$	$C_1(x) \vee \ldots \vee C_2(x)$
complementOf	$\neg C$	$\neg C(x)$
oneOf	$\{a_1, \ldots, a_n\}$	$x = a_1 \vee \ldots \vee x = a_n$
hasClass	$\exists P.C$	$\exists y(P(x,y) \wedge C(y))$
toClass	$\forall P.C$	$\forall y(P(x,y) \rightarrow C(y))$

Mapping R2ML & Rei

Rei	R2ML
Each Deontic Element	A *Derivation Rule*
Variable Definition	*ObjectClassificationAtoms*
OR	qf.Disjunction
AND	The *conclusion* in the rule is a conjunction of elemenets
NOT	Atom is Negated
SimpleConstraint	ReferencePropertyAtoms
SpeechActs	ObjectDescriptionAtoms
SubElements	Object- or Data-Slots



- We should get the identical Rei Policy:

prohibit our system from using data that is accepted by the members of a group called UserActor

Mapping R2ML & Rei – cnt'd

```
<entity:Variable rdf:ID="x"/>
<entity:Variable rdf:ID="y"/>
<entity:Variable rdf:ID="negAuth"/>
<constraint:SimpleConstraint rdf:ID="constraint1 ">          1
   <constraint:subject rdf:resource="#x "/>
   <constraint:predicate rdf:resource="&rdfs;type "/>
   <constraint:object rdf:resource="#AcceptData "/>
</constraint:SimpleConstraint>

<constraint:SimpleConstraint rdf:ID="constraint2 ">
   <constraint:subject rdf:resource="#y "/>
   <constraint:predicate rdf:resource="&rdfs;type "/>
   <constraint:object rdf:resource="#UserActors "/>
</constraint:SimpleConstraint>

<constraint:And rdf:ID="conditions ">
   <constraint:first rdf:resource="#constraint1 "/>
   <constraint:second rdf:resource="#constraint2 "/>
</constraint:And>

<constraint:SimpleConstraint rdf:ID="actor_value ">          3
   <constraint:subject rdf:resource="#y "/>
   <constraint:predicate rdf:resourc="#performedBy "/>
   <constraint:object rdf:resource="#x "/>
</constraint:SimpleConstraint>

<constraint:SimpleConstraint rdf:ID="actio_value ">          2
   <constraint:subject rdf:resource="#x "/>
   <constraint:predicate rdf:resource="controls "/>
   <constraint:object rdf:resource="#Plcy_Action "/>
</constraint:SimpleConstraint>

<deontic:Prohibition rdf:ID="AcpDataP">                      4
    <deontic:actor rdf:resource="#actor_value"/>
    <deontic:action rdf:resource="#action_value"/>
    <deontic:constraint rdf:resource="#conditions"/>
</deontic:Prohibition>
```

Rei

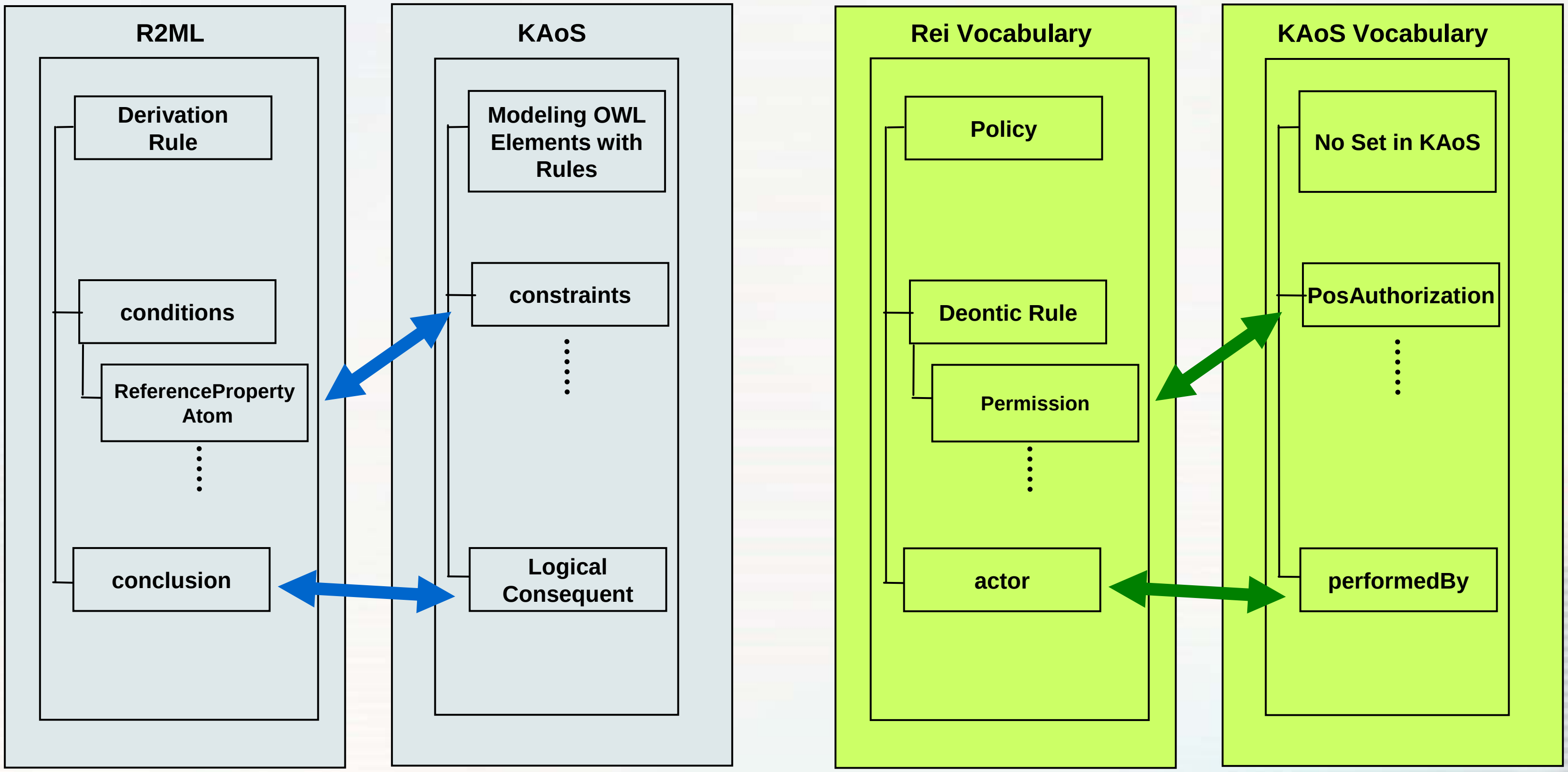
```
<r2ml:DerivationRule>
  <r2ml:conditions>
    <r2ml:ObjectClassificationAtom                           1
     r2ml:classID="#AcceptData">
        <r2ml:ObjectVariable r2ml:name="x"/>
    </r2ml:ObjectClassificationAtom >
    <r2ml:ObjectClassificationAtom
     r2ml:classID="#UserActor">
        <r2ml:ObjectVariable r2ml:name="y"/>
    </r2ml:ObjectClassificationAtom >
  </r2ml:conditions>
  <r2ml:conclusion>
    <r2ml:ObjectDescriptionAtom                              4
     r2ml:classID="Prohibition">
      <r2ml:subject>
           <r2ml:ObjectVariable r2ml:name="AcpDataP"/>
      </r2ml:subject>

      <r2ml:ObjectSlot                                       2
            r2ml:referencePropertyID="controls"/>
       <r2ml:ObjectVariable r2ml:name="x"
                      r2ml:classID="#Plcy_Action">
      </r2ml:ObjectSlot>

      <r2ml:ObjectSlot                                       3
            r2ml:referencePropertyID="performedBy">
        <r2ml:ObjectVariable r2ml:name="y"/>
      </r2ml:ObjectSlot>
    </r2ml:ObjectDescriptionAtom>
  </r2ml:conclusion>
</r2ml:DerivationRule>
```

R2ML

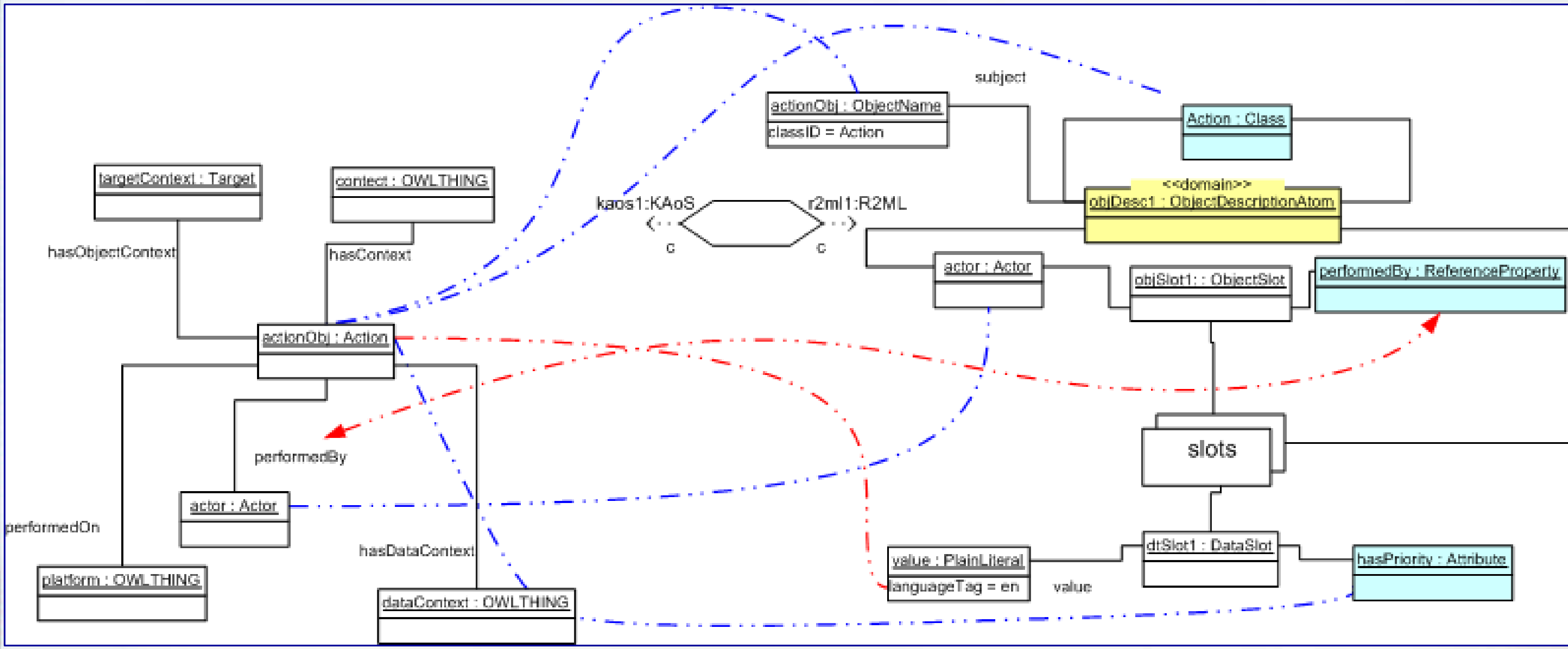
Mapping KAoS & R2ML

- The KAoS Policy:

  prohibit our system from using data that is accepted by the members of a group called UserActor

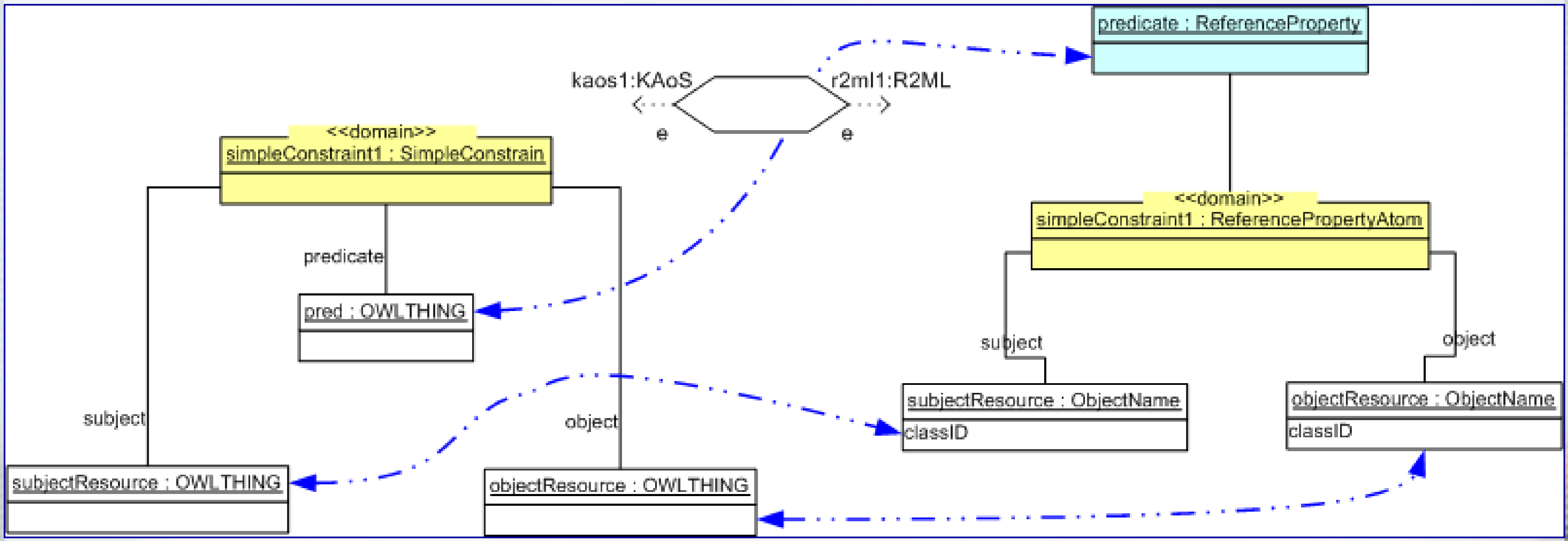


KAoS and Rei Meta-Models



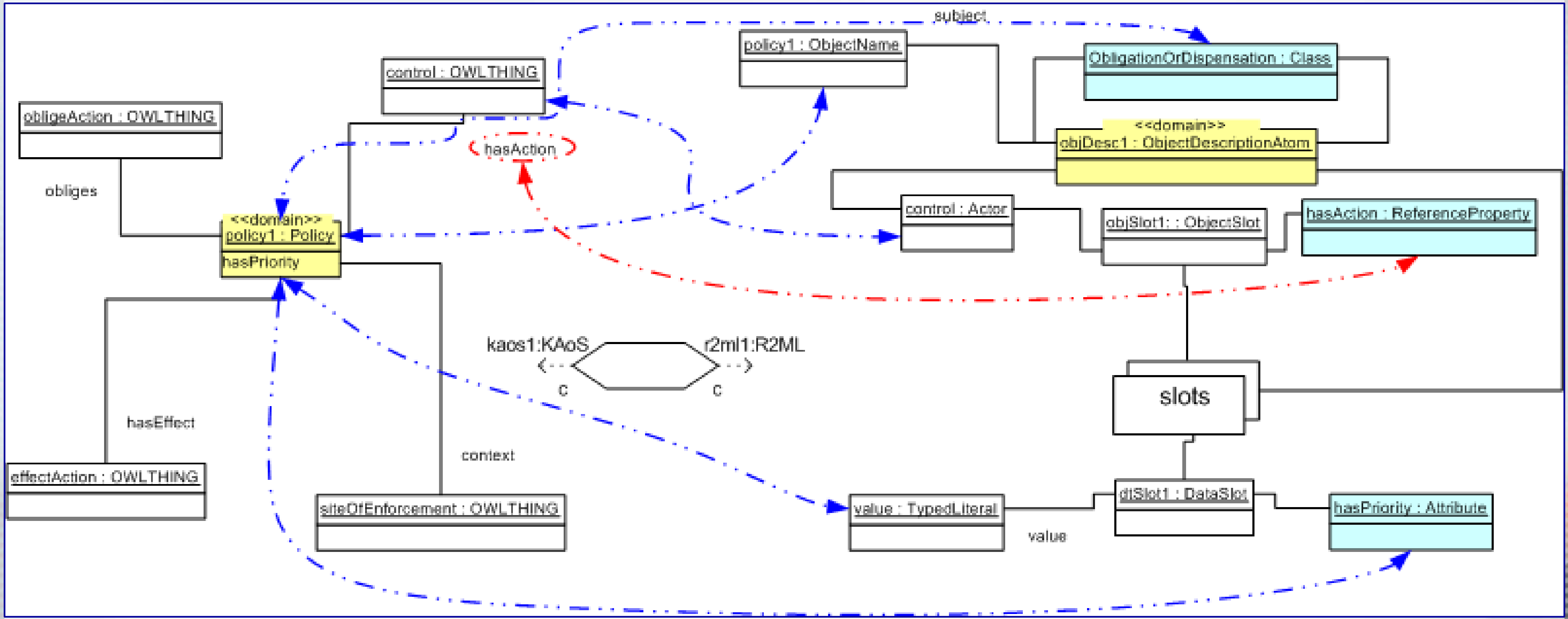
Rei Action to R2ML ObjectDescriptionAtom

KAoS and Rei Meta-Models



Rei SimpleConstraint to R2ML ObjectDescriptionAtom

KAoS and Rei Meta-Models



KAoS Policy Rule to R2ML ObjectDescriptionAtom

Mapping KAoS & R2ML - cnt'd

```
<r2ml:DerivationRule>
  <r2ml:conditions>
    <r2ml:ObjectClassificationAtom
     r2ml:classID="#AcceptData">
        <r2ml:ObjectVariable r2ml:name="x"/>
    </r2ml:ObjectClassificationAtom >
    <r2ml:ObjectClassificationAtom
     r2ml:classID="#UserActor">
        <r2ml:ObjectVariable r2ml:name="y"/>
    </r2ml:ObjectClassificationAtom >
  </r2ml:conditions>
  <r2ml:conclusion>
    <r2ml:ObjectDescriptionAtom
     r2ml:classID="Prohibition">
      <r2ml:subject>
           <r2ml:ObjectVariable r2ml:name="AcpDataP"/>
      </r2ml:subject>

      <r2ml:ObjectSlot
            r2ml:referencePropertyID="controls"/>
       <r2ml:ObjectVariable r2ml:name="x"
                      r2ml:classID="#Plcy_Action">
      </r2ml:ObjectSlot>

      <r2ml:ObjectSlot
            r2ml:referencePropertyID="performedBy">
        <r2ml:ObjectVariable r2ml:name="y"/>
      </r2ml:ObjectSlot>
    </r2ml:ObjectDescriptionAtom>
  </r2ml:conclusion>
</r2ml:DerivationRule>
```

1 4 2 3

```
<policy:NegAuthorizationPolicy rdf:ID="AcpDataP">
  <policy:controls rdf:resource="#Plcy _Action"/>
  <policy:hasPriority>2</policy:hasPriority>
</policy:NegAuthorizationPolicy>
<owl:Class rdf:ID="Plcy _Action ">
  <owl:intersectionOf>
    <owl:Class rdf:about="#AcceptData"/>
     <owl:Class>
        <owl:Restriction>
            <owl:onProperty rdf:resource="
                            #performedBy"/>
            <owl:allValuesFrom>
                 <owl:Class rdf:about="#UserActor"/>
            </owl:allValuesFrom>
        </owl:Restriction>
     </owl:Class>
   </owl:intersectionOf>
</owl:Class>
```

4 2 1 3 1

R2ML

KAoS

Evaluation of the information loss

- Reasoning on the obtained policies
  - The reasoner for Rei is not supported any more
  - No release for KAoS reasoner
- Derivation Rules or Integrity Rules
- The Difference in the underlying Logic
  - KAoS has both universal and existential quantifiers
  - Rei only has universal quantifiers
- Universal and Existential Quantifiers
- Cardinality Support for the Rules
- Language specific concepts
  - SpeechActs in Rei …. No equivalent concept in KAoS

Is it still effective when we perform the transformations?

Conclusions

- **Benefits**
  - Language Independence Policy Design
  - Architecture independent
  - Easier surfing of the web for broker agents

- **Known Issues**
  - Information loss during exchange
    - How it may affect the trust
  - Derived R2ML transformations from different languages do not exactly match
    - An internal exchange between R2ML rules might be required

Future Direction

- Towards Combining Model Driven Approaches and Policy Languages
  - Policy Modeling Language
  - Connecting various policy languages through their models
  - XACML as a widely recognized policy language
- Combining Service Oriented Architecture (SOA) with Policy Modeling
- Semantic Web and its ability to introduce context based concepts that facilitate the definition of TRUST.

Questions?

Thank you