The RuleML Perspective on Deliberation-Reaction Standards

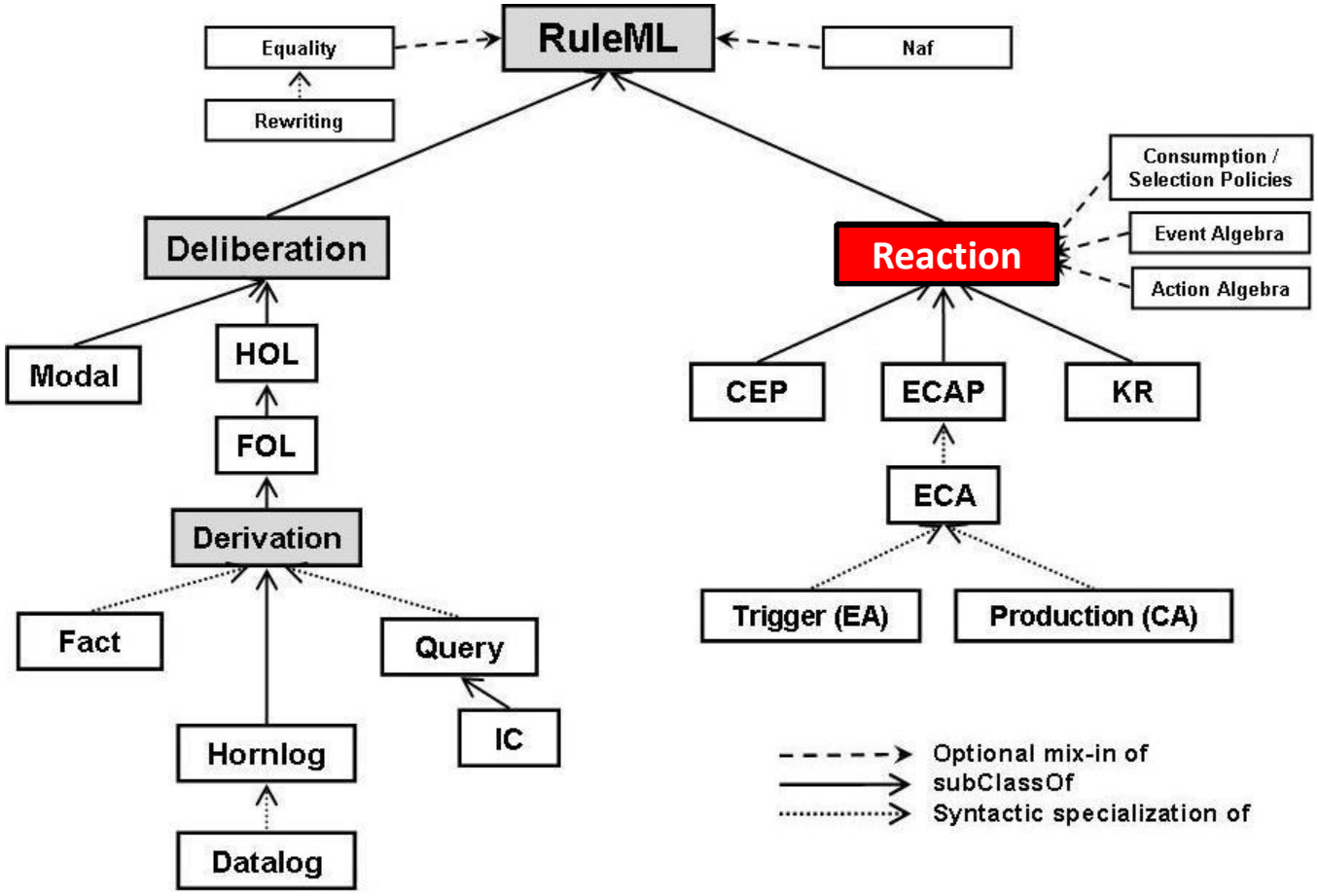
Adrian Paschke, Harold Boley, Tara Athan

In **Ontolog RulesReasoningLP: Series Session 5**
9 January 2014

Agenda

- Reaction RuleML
- Semantic Profiles
- Semantic Metamodel and External Ontologies (Semantic Sorts)
- Syntactic Customization

The RuleML Family



Reaction RuleML 1.0 Paper: http://link.springer.com/chapter/10.1007%2F978-3-642-32689-9_9
Reaction RuleML 1.0 Tutorial: http://www.slideshare.net/swadpasc/reaction-ruleml-ruleml2012paschketutorial
RuleML 1.0 Paper http://link.springer.com/chapter/10.1007/978-3-642-16289-3_15
RuleML Overview Slides: http://cs.unb.ca/~boley/talks/RuleML-Overarching-Talk.pdf

Reaction Rules: Four Sub-branches

- **PR** Production Rules
  (Condition-Action rules)
- **ECA** Event-Condition-Action (ECA) rules
- **CEP** Rule-based Complex Event Processing (complex event processing reaction rules, (distributed) event messaging reaction rules, query reaction rules, etc.)
- **DR and KR** Knowledge Representation AI Reasoning with Temporal/Event/Action/Situation/ Transition/Process Logics and Calculi

Quick Overview: Reaction RuleML Dialects

* + variants and alternatives

- **Spatio-Temporal Derivation RuleML** (*if-then*)*
  - Time, Spatial, Interval
- **KR RuleML** (*if-then* or *on-if-do*) *
  - Situation, Happens$_{(@type)}$, Initiates, Terminates, Holds, fluent
- **Production RuleML** (*if-do*) *
  - Assert, Retract, Update, Action
- **ECA RuleML** (*on-if-do*) *
  - Event, Action, + (event / action algebra operators)
- **CEP** (arbitrary combination of on, if, do)
  - Receive, Send, Message

(Reaction) Rules: Specializable Syntax

```
<Rule @key @keyref   @style>
  <meta>   <!–– descriptive metadata of the rule -->          </meta>
  <scope>   <!–– scope of the rule e.g. a rule module -->     </scope>
  <evaluation>    <!-- intended semantics -->                 </evaluation>
  <signature> <!– rule signature -->                          </signature>
  <qualification> <!–– e.g. qualifying rule metadata, e.g.
                     priorities, validity, strategy -->       </qualification>
  <quantification>  <!-- quantifying rule declarations,
                       e.g. variable bindings -->             </quantification>
  <oid>  <!– object identifier -->                            </oid>
  <on>      <!–– event part -->                               </on>
  <if>      <!–– condition part -->                           </if>
  <then>    <!–– (logical) conclusion part -->                </then>
  <do>      <!--  action part -->                             </do>
  <after>   <!–– postcondition part after action,
                e.g. to check effects of execution -->       </after>
  <else>  <!–– (logical) else conclusion -->                  </else>
  <elsedo>  <!-- alternative/else action,
                e.g. for default handling -->                </elsedo>
</Rule>
```

Info, Life Cycle Mgt.

Interface

Imple-mentation

Reaction RuleML – Example Rule Types

- **Derivation Rule:**
  (temporal/event/action/situation **reasoning**)

```
<Rule style="reasoning">
    <if>...</if>
    <then>...</then>
</Rule>
```

- **Production Rule:**

```
<Rule style="active">
    <if>...</if>
    <do>...</do>
</Rule>
```

- **Trigger Rule:**

```
<Rule style="active"> >
    <on>...</on>
    <do>...</do>
</Rule>
```

- **ECA Rule:**

```
<Rule style="active">
    <on>...</on>
    <if>...</if>
    <do>...</do>
</Rule>
```

Agenda

- Reaction RuleML
- Semantic Profiles
- Semantic Metamodel and External Ontologies (Semantic Sorts)
- Syntactic Customization

Integration of Semantic Profiles

1. **Include/Import external Semantic Profile**

```
<xi:include href="../../profiles/SituationCalculusProfile.rrml" xpointer="xpointer(/RuleML/*)"/>
<evaluation>
  <Profile keyref="&ruleml;ReifiedClassicalSituationCalculus" />
</evaluation>
```

2. **Reference pre-defined Semantic Profile as profile type**

```
<evaluation>
  <Profile type="&rif;RDFS“ iri="http://www.w3.org/ns/entailment/RDFS"/>
</evaluation>
```

Reference published external semantic profiles, e.g. RIF, OWL, profiles …

3. **Locally define Semantic Profile**

```
<Assert>
  <evaluation>
    <Profile key="&ruleml;ReifiedSituationCalculus" >
      <formula><Rulebase > ... RuleML definition… </Rulebase></formula>
      <content> … xs:any XML content, e.g. RIF, Common Logic XML… </content>
    </Profile>
  </evaluation>
  <Rulebase>
      <Rule> … </Rule>
      <Rule> … </Rule>
  </Rulebase>
</Assert>
```

Note: also other non RuleML content models are supported

Example – Reified Situation Calculus Axioms

(1) $\forall$(A1, A2, S1, S2) (do(A1,S1) = do(A2,S2)) ➔ (A1 = A2)

(2) $\forall$(S1, S2, A) (S1 < do(A,S2)) $\equiv$ (S1 <= S2)

(3) $\forall$(S1, S2) (S1 $\leqslant$ S2) $\equiv$ (S1 < S2) $\vee$ (S1 = S2)

(4) $\forall$(S1, S2) (S1 < S2) ➔ ¬ (S2 < S1)

(5) $\forall$(S, F) [holds(F,s0) $\wedge$ ($\forall$(A) (holds(F,S) ➔ holds(F,do(A,S)))] ➔ holds(F,S)

(6) ¬S < s0

…

Example: Axiomatization of Situation Calculus Profile in RuleML

```
<Assert>
 <evaluation>
  <Profile key="&ruleml;ReifiedSituationCalculus" >
    <Rulebase key="&ruleml;ReifiedSituationCalculusBasicAxioms" >
      <!-- Forall(A1, A2, S1, S2) (do(A1,S1) = do(A2,S2)) => (A1 = A2) -->
      <Rule key=„&ruleml;SituationCalculusBasicAxiom1">
          <quantification> <Forall>
             <declare><Var>A1</Var></declare>   <declare><Var>A2</Var></declare>
             <declare><Var>S1</Var></declare>   <declare><Var>S2</Var></declare>
           </Forall> </quantification>
          <if>
           <Equal>
            <Situation>
              <Do> <Action><Var>A1</Var></Action><Situation><Var>S1</Var></Situation>  </Do>
            </Situation>
            <Situation>
              <Do> <Action><Var>A2</Var></Action><Situation><Var>S2</Var></Situation></Do>
            </Situation>
           </Equal>
          </if>
          <then>
           <Equal> <Action><Var>A1</Var></Action> <Action><Var>A2</Var></Action>  </Equal>
          </then>
      </Rule>
       ....
    </Profile>
  </evaluation>
</Assert>
```

Complex Event Processing – Semantic Profiles

(defined in <evaluation> semantic profiles)

1. **Definition**
   - Definition of event/action pattern e.g. by event algebra
   - Based on declarative formalization or procedural implementation
   - Defined over an atomic instant or an interval of time, events/actions, situation, transition etc.
2. **Selection**
   - Defines selection function to select one event from several occurred events (stored in an event instance sequence e.g. in memory, database/KB) of a particular type, e.g. "*first*", "*last*"
   - Crucial for the outcome of a reaction rule, since the events may contain different (context) information, e.g. different message payloads or sensing information
3. **Consumption**
   - Defines which events are consumed after the detection of a complex event
   - An event may contribute to the detection of several complex events, if it is not consumed
   - Distinction in event messaging between "multiple receive" and "single receive"
   - Events which can no longer contribute, e.g. are outdated, should be removed
4. **Execution**
   - Actions might have an internal effect i.e. change the knowledge state leading to state transition from (pre-)condition state to post-condition state
   - The effect might be hypothetical (e.g. a hypothetical state via a computation) or persistent (update of the knowledge base),
   - Actions might have an external side effect

Example - Reaction Rules with Semantic Profiles

```
<Rule style="active">
  <evaluation>
    <Profile> e.g. selection and consumptions policies  </Profile>
  </evaluation>
  <signature>
    <Event key="#ce1">
        … event pattern definition (for event detection)
    </Event>
  </signature>

<on>
  <Event keyref="#ce1"/> <!– use defined event pattern for detecting events -->
</on>
<if>
    …
</if>
 <do>
   <Assert safety="transactional">  <!-- transactional update -->
     <formula>
          <Atom>
                    ….
          </Atom>
     </formula>
    </Assert>
  </do>
 </Rule>
```

Interface

(semantic profile + event pattern signature for event processing / detection)

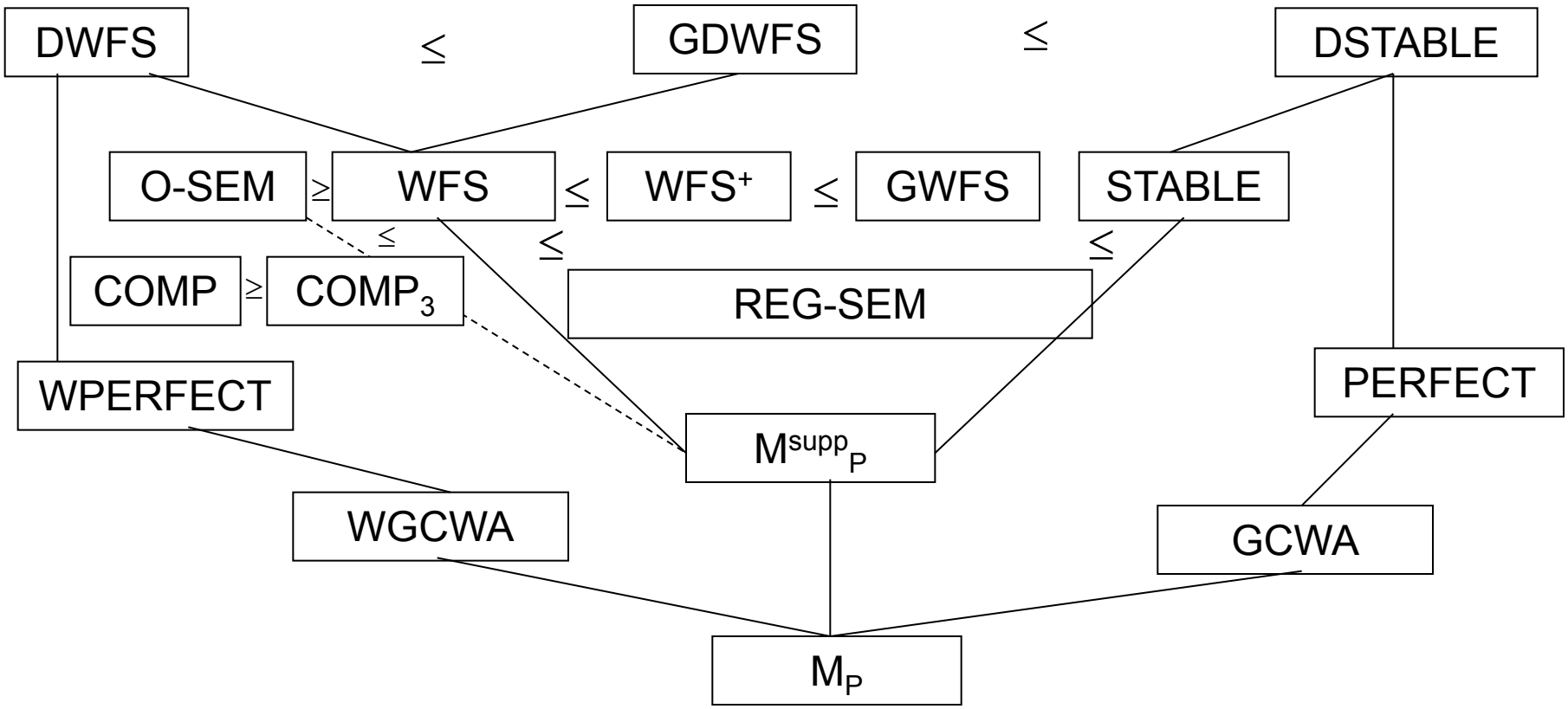
Implementation

(reaction rule triggered by event detection)

Example - Semantic LP Profiles

Class	Semantics
Definite LPs	Least Herbrand model: M_p
Stratified LPs	Supported Herbrand model: M_p^{supp}
Normal LPs	Clark's Completion: $COMP$
	3-valued Completion: $COMP_3$
	Well-founded Semantics: WFS
	WFS^+ and WFS'
	WFS_C
	Strong Well-founded Semantics: WFS_E
	Extended Well-founded Semantics: WFS_S
	Stable Model Semantics: $STABLE$
	Generalized WFS: $GWFS$
	$STABLE^+$
	$STABLE_C$
	$STABLE^{rel}$
	Pereira's $O-SEM$
	Partial Model Semantics: $PARTIAL$
	Regular Semantics: $REG-SEM$
	Preferred Semantics: $PREFERRED$
General Disjunctive	Disjunctive WFS: $DWFS$
	Generalized Disjunctive WFS: $GDWFS$
	Disjunctive Stable: $DSTABLE$
Stratified Disjunctive	Perfect model $PERFECT$
	Weakly Perfect: $WPERFECT$
	Generalized Closed World Assumption: $GCWA$
Positive Disjunctive	Weak generalized closed world assumption: $WGCWA$

Semantic Layering - Example LP Semantic Profiles



A semantic SEM‘ extends a semantic SEM:

SEM‘ ⩾ SEM iff ∀ (P, F) SEM(P) |= F ➔ SEM‘(P) |= F

Semantic Profiles for Evaluation

- A semantics ***SEM‘*** extends ***SEM*** for rule program ***P*** iff all formula ***F*** which are true in ***SEM(P)*** are also true in ***SEM‘(P),*** but in SEM‘(P) more formula can be true or false than with SEM(P).
- ***SEM'*** is defined for a class of programs that strictly includes the class of programs with the semantics ***SEM***
- ***SEM'*** coincides with ***SEM*** for all programs of the class of programs for which ***SEM*** is defined

➔A semantic profile ***SEM’*** (= intended evaluation semantics of the inference engine) can be used for all programs (or scoped modules) with **SEM(P) ⩽ SEM’(P)**

Example: Use of Semantic LP Profiles for Interpretation

```
<-- rule interface with two alternative interpretation semantics and a signature.
    The interface references the implementation identified by the corresponding key -->
<Rule keyref="#r1">
 <evaluation index="1">
   <!-- WFS semantic profile -->
   <Profile type="&ruleml;Well-Founded-Semantics" />
 </evaluation>
 <evaluation index="2">
  <!-- alternative ASS semantic profile define in the metamodel -->
  <Profile type="&ruleml;Answer-Set-Semantics" />
 </evaluation>
 <!-- the signature  defines the queryable head of the backward-reasoning rule -->
 <signature>
   <Atom><Rel>likes</Rel><Var mode="+"/><Var mode="-"/></Atom>
 </signature>
</Rule>

<!-- implementation of rule 1 which is interpreted either by WFS or by ASS semantics
     and only allows queries according to it's signature definition. -->
<Rule key="#r1" style="reasoning">
  <if>...  </if>
  <then>
    <Atom><Rel>likes</Rel><Var>X</Var><Var>Y</Var></Atom>
  </then>
</Rule>
```

Interface with evaluation semantics and public rule signature

Implementation

Example – RIF SWC Profiles

Profile	IRI of the Profile	Model	Satisfiability	Entailment
Simple	http://www.w3.org/ns/entailment/Simple	RIF-Simple-model	satisfiability	RIF-Simple-entailment
RDF	http://www.w3.org/ns/entailment/RDF	RIF-RDF-model	RIF-RDF-satisfiability	RIF-RDF-entailment
RDFS	http://www.w3.org/ns/entailment/RDFS	RIF-RDFS-model	RIF-RDFS-satisfiability	RIF-RDFS-entailment
D	http://www.w3.org/ns/entailment/D	RIF-D-model	RIF-D-satisfiability	RIF-D-entailment
OWL Direct	http://www.w3.org/ns/entailment/OWL-Direct	RIF-OWL Direct-model	RIF-OWL Direct-satisfiability	RIF-OWL Direct-entailment

Simple < RDF < RDFS < D < OWL RDF-Based

Example:

```
<evaluation>
        <Profile type="&rif;RDFS“ iri="http://www.w3.org/ns/entailment/RDFS"/>
</evaluation>
```

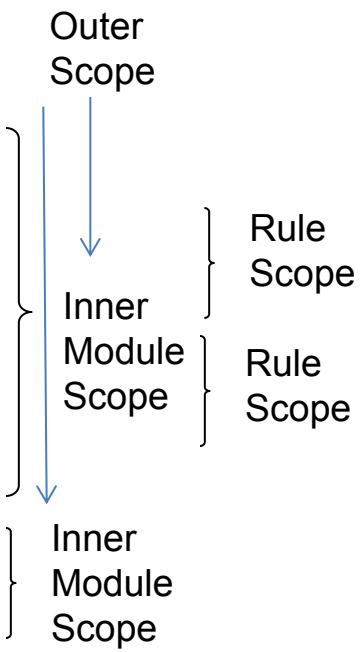
Semantic Profiles in Nested Program Scopes

```
<Assert key="#module1">
   <evaluation><Profile> ...SEM''… </Profile><evaluation>

<Rulebase key="#innermodule1.1">
      <evaluation><Profile> ...SEM'… </Profile><evaluation>
   <Rule key="#rule1">
      <evaluation><Profile> ...SEM… </Profile><evaluation>
   </Rule>
   <Rule key="#rule2">
      …
   </Rule>
 </Rulebase>

<Rulebase key="#innermodule1.2"> … </Rulebase>

 </Assert>
```



SEM''(#module1) ⩾ SEM'(#innermodule1.1) ⩾ SEM(#rule1)
SEM(#rule2) = SEM''(#module1)
SEM'(#innermodule1.2) = SEM''(#module1)

Dynamic Program Views with Scopes and Guards

```
<Assert key="#module1”>
 <meta> <!-- descriptive metadata -->
   <Atom><Rel iri="dc:creator"/><Ind>Adrian Paschke</Ind></arg></Atom>
 </meta>
 <qualification> <!-- qualifying metadata -->
   <Atom> <!-- the module is valid for one year from 2011 to 2012 -->
    <Rel>valid</Rel>
    <Interval type="&ruleml;TimeInstant">
       <Time type="&ruleml;TimeInstant"><Data xsi:type="xs:date">2011-01-01</Data></Time>
       <Time type="&ruleml;TimeInstant"><Data xsi:type="xs:date">2012-01-01</Data></Time>
    </Interval>
   </Atom>
 </qualification>
<Rulebase key="#innermodule1.1">
    <!– the rule base scopes apply to all knowledge in the rule base -->
   <scope> <!– scope : only knowledge authored by „Adrian Paschke“ -->
      <Atom><Rel iri="dc:creator"/><Ind>Adrian Paschke</Ind></arg></Atom>
   </scope>
   <scope> <!– scope : only valid knowledge; validity value will be bound to variable -->
      <Atom><Rel>valid</Rel><Var>Validity</Var></Atom>
   </scope>
   <Rule key=“#rule1”>
      <scope> <Atom><Rel>source</Rel><Ind>./module1.prova</Ind></Atom></scope>
      <guard> <!-- guard on the validity: “current date during validity of module1”
      <Operator type="&ruleml;During"><Expr iri="…getDate()“/><Var>Validity</Var></Operator>
      </guard>
      <if> … </if>  <then>   </then>
   </Rule>
 </Rulebase>
 </Assert>
```

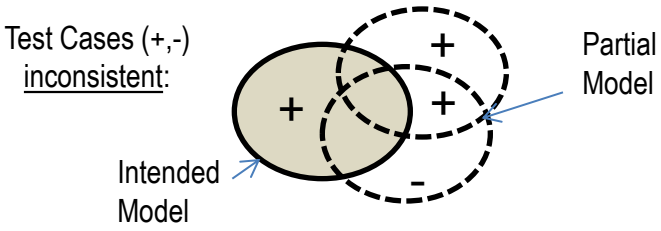
Outer Scope

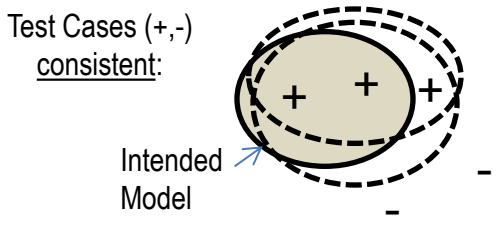
Inner Scope + Guard

Note: Local functions are interpreted with the intended semantics of the scope

Test Cases for Self-validating Rule Bases

- Test Cases constrain the possible models and **approximate the intended models** of the rule base
  - **Queries** are used to test the rule base





- A test case is defined by T := {X, A, N}, where
  - $X \subseteq L$ assertion base (input data, e.g. facts)
  - $A \in L$ a formula denoting a test query
  - $N := +, -$ a positive or negative label
- Semantics

$$M_0 \models_{TC} (X, A, +) \text{ iff } \forall m \in M_0 : m \in \sum(Mod(X), R) \Rightarrow m \in Mod(A)$$

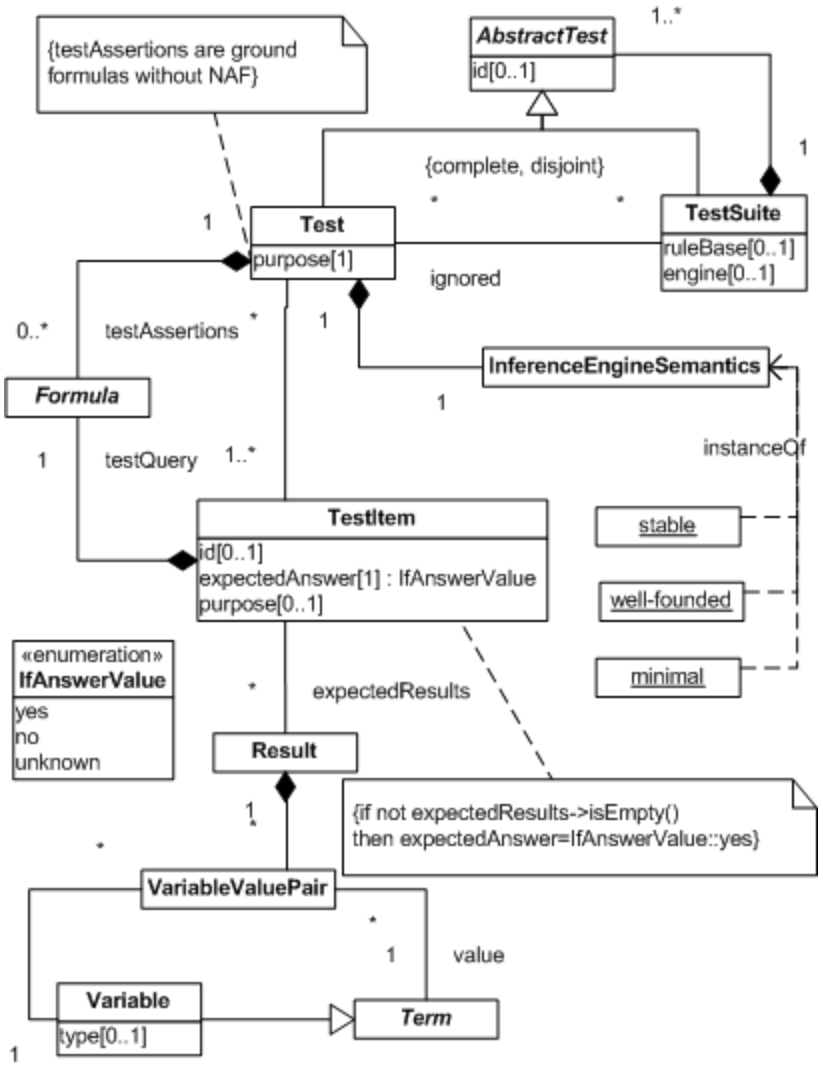
$$M_0 \models_{TC} (X, A, -) \text{ iff } \exists m \in M_0 : m \in \sum(Mod(X), R) \Rightarrow m \notin Mod(A)$$

  - $\models_{TC}$ compatibility relation
  - Mod association function between sets of formulas and sets of models
  - Σ model selection function

$A \notin C_R(X)$ for T:={X, A, +} and $A \in C_R(X)$ for T:={X,A,-}

  - $C_R(X)$ deductive closure of X. Decidable inference operator based on formal proofs

Rule Markup for Test Cases / Test Suites




```
<TestSuite ruleBase="SampleBase.xml">
      <Test id="ID001" purpose="...">
        <Assert><formula>
          <And>
              <Atom>
              <Rel>parent</Rel>
              <Ind>John</Ind>
              <Ind>Mary</Ind>
            </Atom>
          </And>
        </formula></Assert>
 <TestItem expectedAnswer="yes">
    <Query><formula>
     <Atom closure="universal">
      <Rel>uncle</Rel>
      <Ind>Mary</Ind>
      <Var>Nephew</Var>
     </Atom>
    </formula></Query>
    <expectedResults>
     <VariableValuePair>
        <Var>Nephew</Var>
        <Ind>Tom</Ind>
     </VariableValuePair>
     <VariableValuePair>
        <Var>Nephew</Var>
        <Ind>Irene</Ind>
     </VariableValuePair>
  </expectedResults>
 </TestItem>
 <InferenceEngineSemantics><Profile
      ="&ruleML;MinimalHerbrandSemantics"/></InferenceEngineSemantics>
</Test></TestSuite>
```

Testing Semantic Properties

- Entailment Tests for Classical Logics
  - *right weakening, reflexivity, and, or, left logical equivalence, cautious monotony, cut, rationality, negation rat., disjunction rat., …*
- Entailment Tests for Skeptical Logics
  - *cumulativity, rationality, …*
- Weak Semantic Properties Tests
  - *elimination of tautologies, generalized principle of partial evaluation, positive/negative reduction, elimination of non-minimal rules, independence, relevance, …*

Example – Cautious Montony Test

```
P:    a <- not b
      b <- not a
      c <- not c
      c <- a
```

```
P':   a <- not b
      b <- not a
      c <- not c
      c <- a
      c
```

```
T:{a=>true,c=>true}
```

- STABLE(P) $|=_{TC}$ {a, not b, c}, i.e. T succeeds.
- STABLE(P') $|=_{TC}$ {not a, b, c}, i.e. T fails.

➔Test Case: STABLE does not cautious monotony

Agenda

- Reaction RuleML
- Semantic Profiles
- Semantic Metamodel and External Ontologies (Semantic Sorts)
- Syntactic Customization

RuleML Types (Sorted Logic)

- Types (sorts) can be assigned by using the **@type** attribute
- External vocabularies / ontologies define types, e.g.,

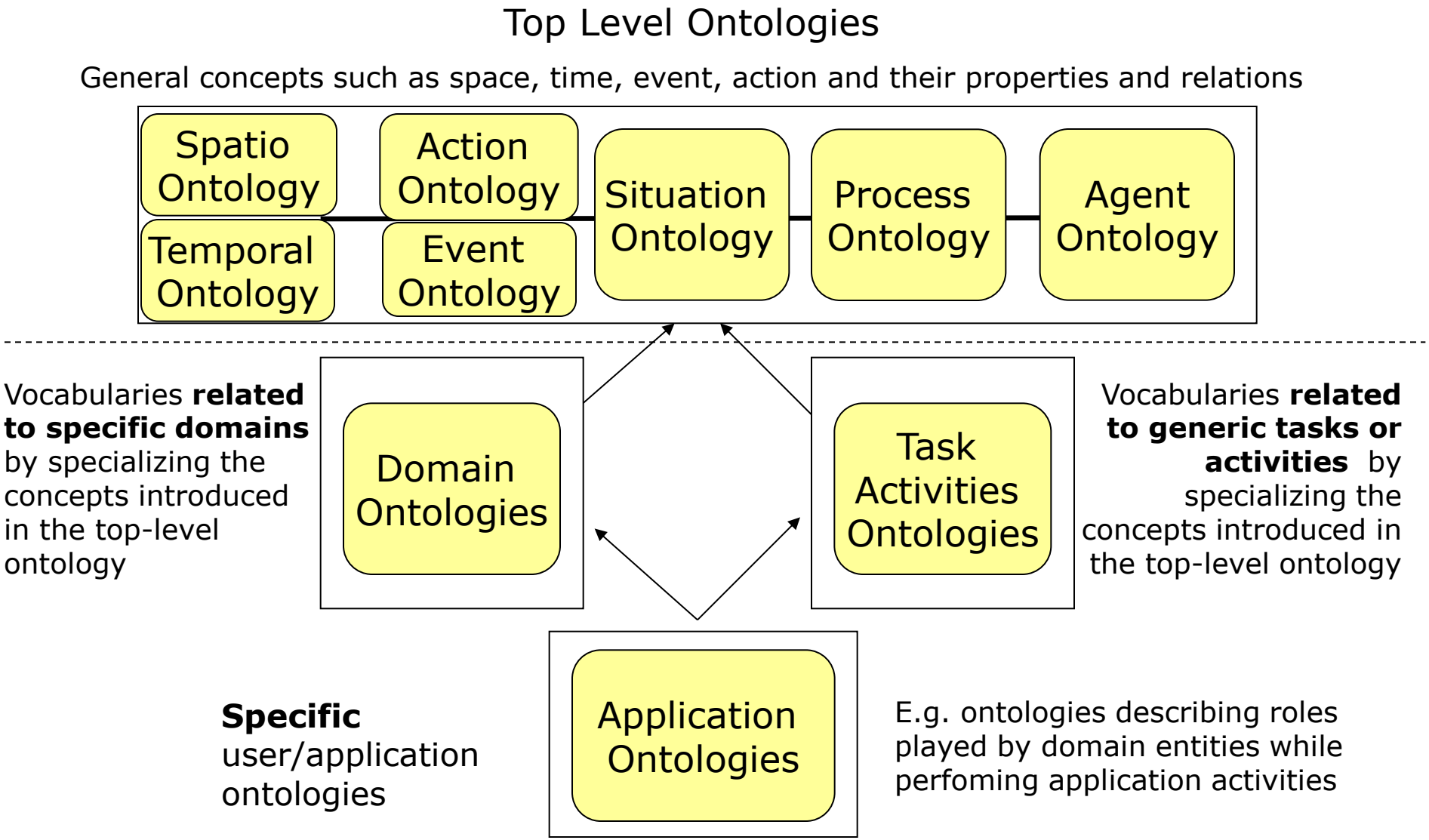
```
<Var type="&vo;Vehicle">Car</Var>
<Ind iri="&vo;Corolla" type="&vo;Sedan"/>
```

- Semantics defined / linked by Semantic Profile, e.g. order-sorted logic using RDFS or OWL entailment profiles

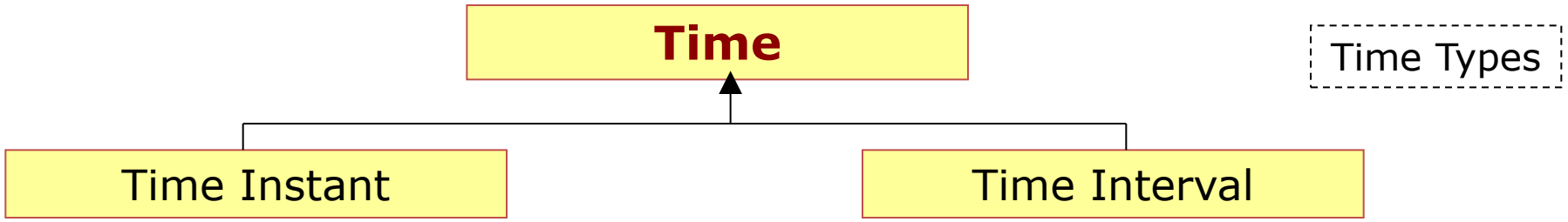
Reaction RuleML Examples with Types from RuleML Metamodel and External Ontologies

<Quantifier type="&ruleml;Forall"> == <Forall>
<Operator type="&ruleml;And"> == <And>
<Operator type="&ruleml;Conjunction"> == <Conjunction>
<Negation type="&ruleml;InflationaryNegation"> == <Naf>
<Action type="&ruleml;Assert"> == <Assert>
<Action type="&ruleml;Retract"> == <Retract>
<Event type="&ruleml;SimpleEvent"> == <Atom> … </Atom>
<Event type="ibm:CommonBaseEvent"> == IBM CBE
<Operator type="snoop:Squence"> == Snoop Algebra
== <Operator type="&ruleml;Sequence"> == <Sequence>
<Ind iri="person.xml#xpointer(//Person/LastName[1]/text())"/>
<Action iri="BPEL.xml#xpointer(//invoke[@name=checkHotel])">

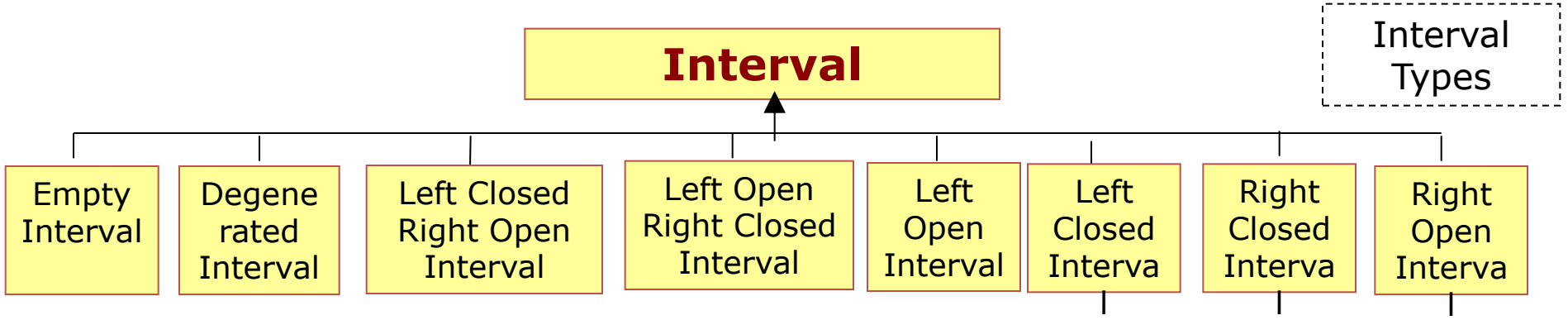
Reaction RuleML Metamodel



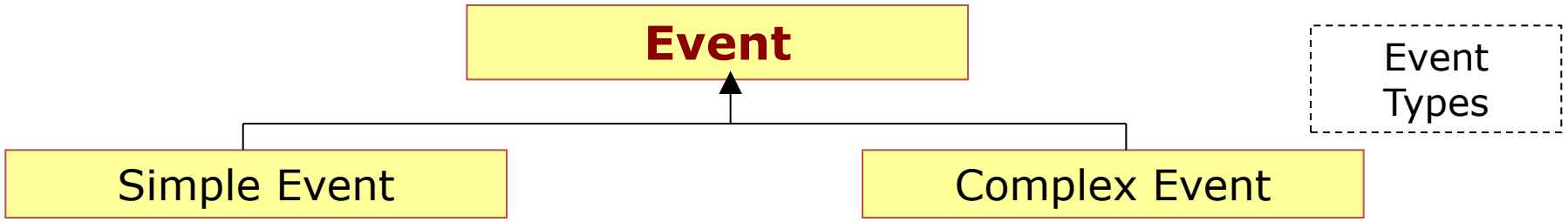
Time Top Ontology Metamodel



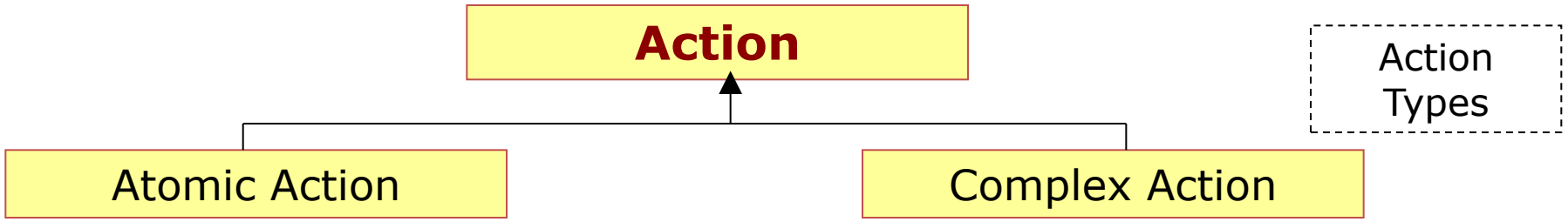
Interval Top Ontology Metamodel



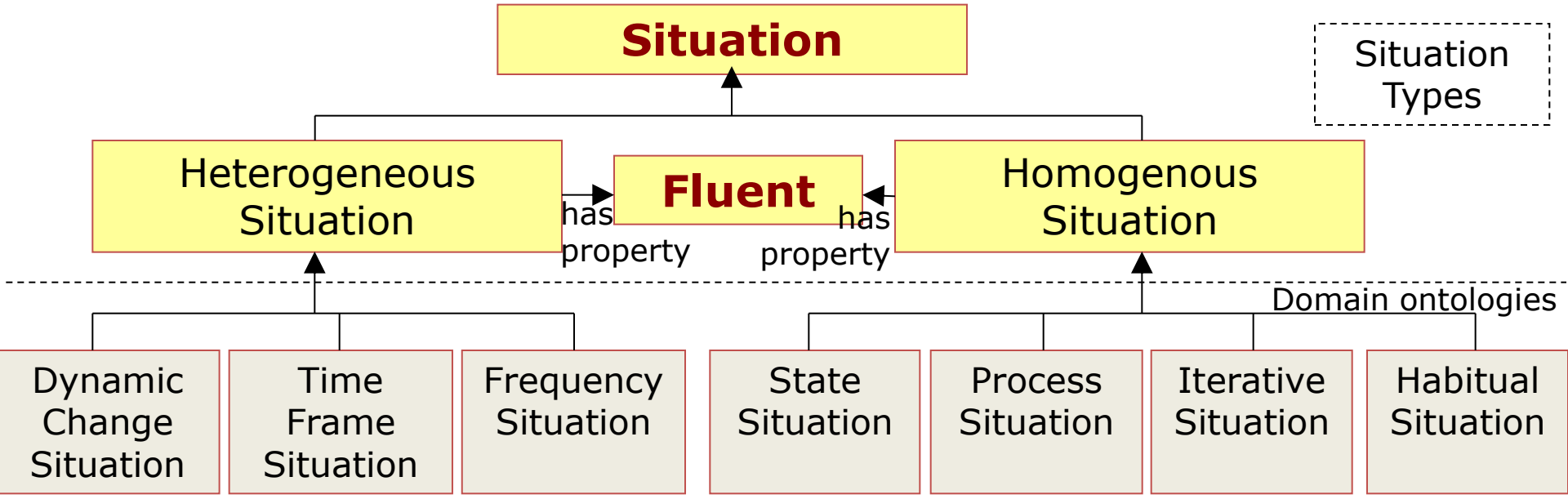
Event Top Ontology Metamodel



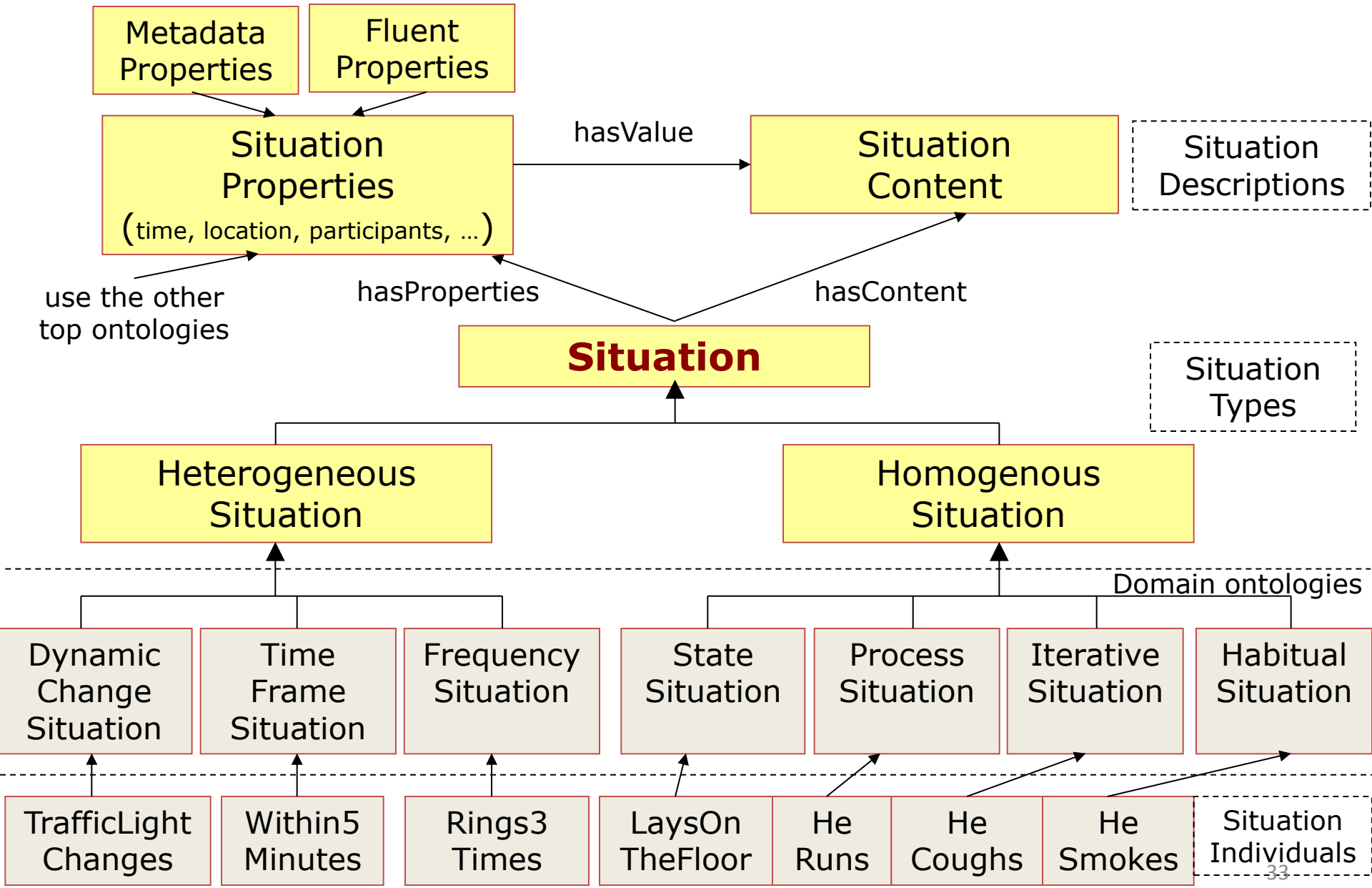
Action Top Ontology Metamodel



Situation Top Ontology Metamodel

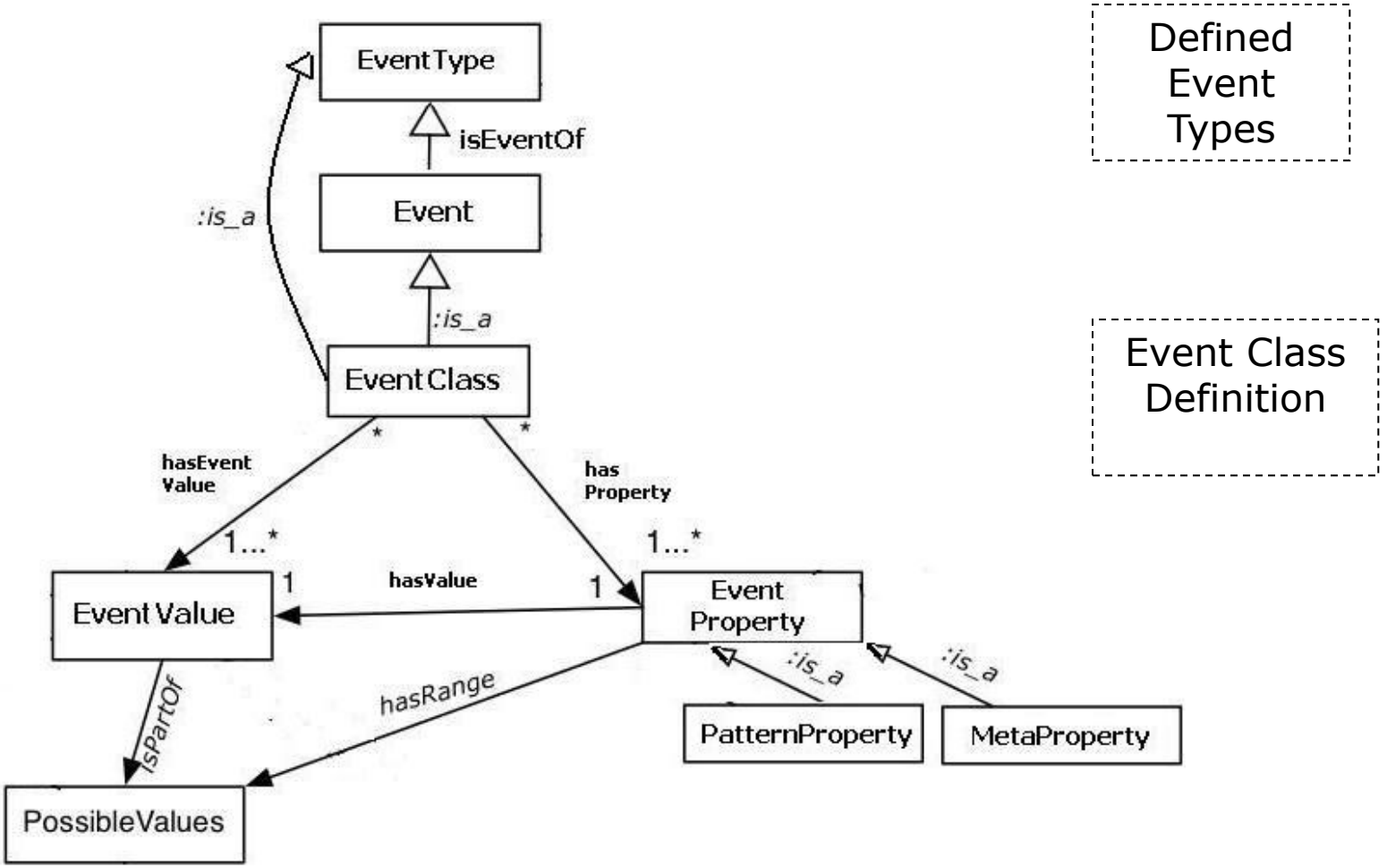


Example: Situation Top Ontology Metamodel



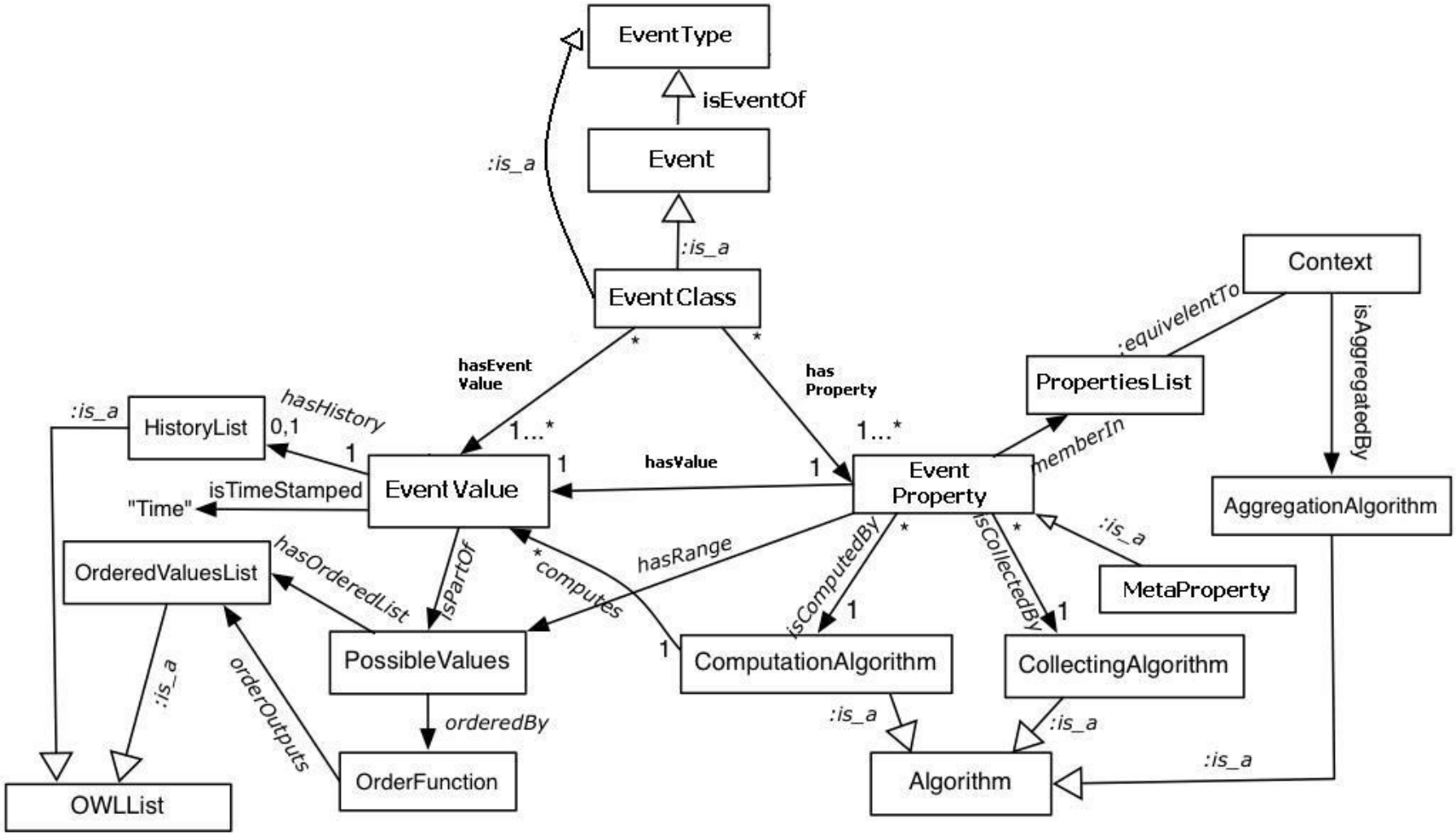
Example - Event MetaModel

(for defining Event Types as Instances of the MetaModel Event Class)



Extended Event Meta Model Ontology

(with computational properties and complex value definitions)



*Example - Typed Complex Event **Pattern** Definition*

```
<Event key="#ce2" type="&ruleml;ComplexEvent">
  <signature> <!-- pattern signature definition -->
    <Sequence>
      <signature>
        <Event type="&ruleml;SimpleEvent">
          <signature><Event>...event_A...</Event></signature>
        </Event>
      </signature>
      <signature><Event type="&ruleml;ComplexEvent" keyref="ce1"/></signature>
      <signature>
        <Event type="cbe:CommonBaseEvent" iri="cbe.xml#xpointer(//CommonBaseEvent)"/>
      </signature>
    </Sequence>
  </signature>
</Event>

<Event key="#ce1">
  <signature> <!-- event pattern signature -->
    <Concurrent>
      <Event><meta><Time>...t3</Time></meta><signature>...event_B</signature></Event>
      <Event><meta><Time>...t3</Time></meta><signature>...event_C</signature></Event>
    </Concurrent>
  </signature>
</Event>
```

Event Pattern defining **Event Templates** signature

```
<Event key="#e1" keyref="#ce2"><content>…</content></Event>
```

Event Instance

Agenda

- Reaction RuleML
- Semantic Profiles
- Semantic Metamodel and External Ontologies (Semantic Sorts)
- Syntactic Customization

RuleML Sublanguages Customized by MYNG as Relax NG Schemas (1)

Selection Form

Instructions

Make a selection from the form below, then click "Refresh Schema" to update the Schema URL. The main module is also displayed below the form. To reset the form to the default (supremum) values, click "Reset Form".

Reset Form | Refresh Schema

Schema URL = http://ruleml.org/1.0/relaxng/schema_rnc.php?backbone=x3f&default=x7&termseq=x7&lng=x1&propo=x3ff&implies=x7&terms=xf3f&quant=x7&expr=xf&serial=xf

Expressivity "Backbone" (Check One)
- ○ Atomic Formulas
- ○ Ground Fact
- ○ Ground Logic
- ○ Datalog
- ○ Horn Logic
- ○ Disjunctive Logic
- ◉ Full First-Order Logic

Treatment of Attributes With Default Values (Check One)
- ○ Required to be Absent
- ○ Required to be Present
- ◉ Optional

Term Sequences: Number of Terms (Check One)
- ○ None
- ○ Binary (Zero or Two)
- ◉ Polyadic (Zero or More)

Language (Check One)
- ◉ English Abbreviated Names
- ○ English Long Names
- ○ French Long Names

Serialization Options (Check Zero or More)
- ☑ Unordered Groups
- ☑ Stripe-Skipping
- ☑ Explicit Datatyping
- ☑ Schema Location Attribute

RuleML Sublanguages Customized by MYNG as Relax NG Schemas (2)

Propositional Options (Check Zero or More)
- ☑ IRIs
- ☑ Rulebases
  - ☑ Entailments
- ☑ Degree of Uncertainty
- ☑ Strong Negation
- ☑ Weak Negation (Negation as Failure)
- ☑ Node Identifiers
- ☑ In-Place Annotation
- ☑ XML base
- ☑ XML id

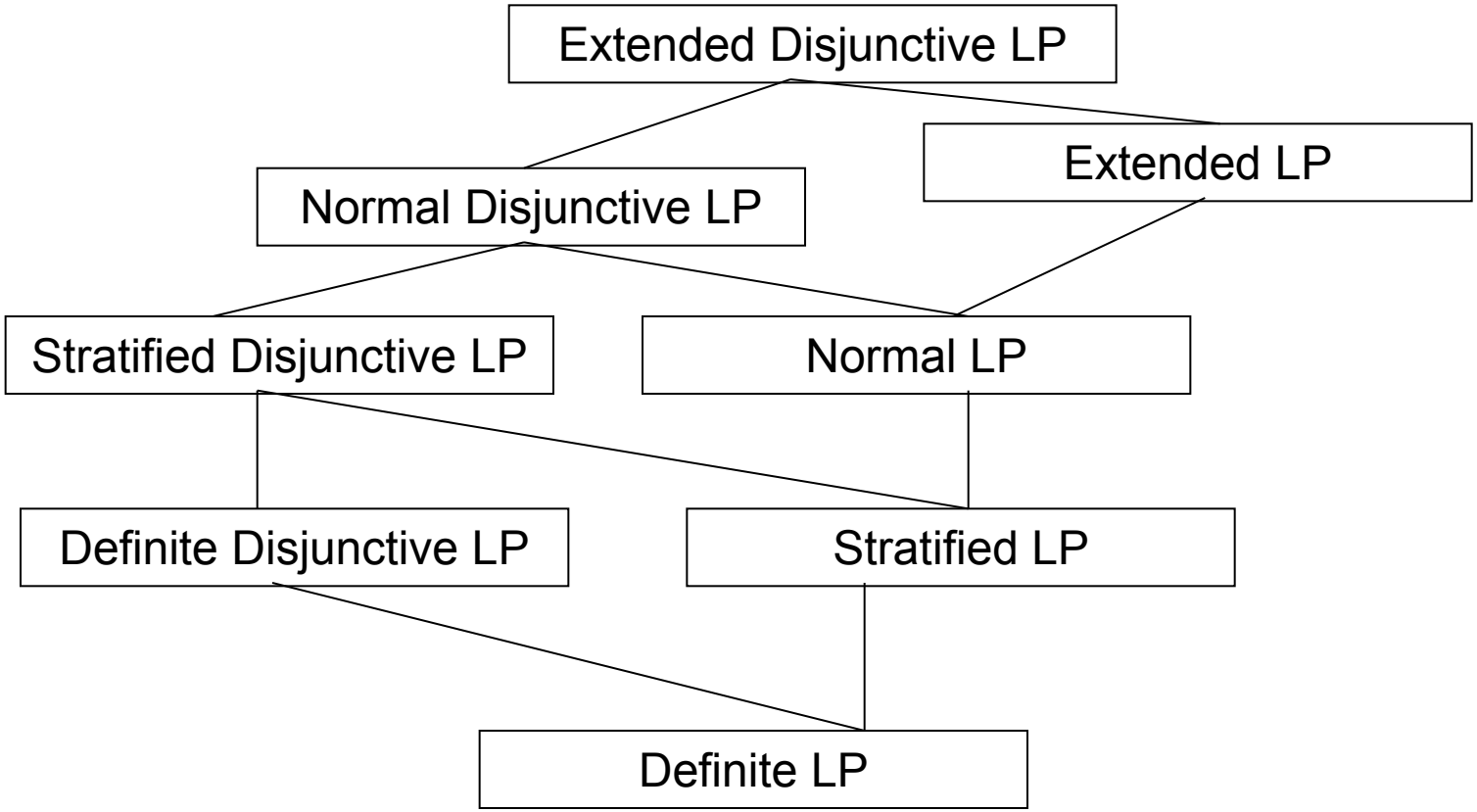
Implication Options (Check Zero or More)
- ☑ Equivalences
- ☑ Inference Direction
- ☑ Non-Material

Term Options (Check Zero or More)
- ☑ Object Identifiers
- ☑ Slots
  - ☑ Slot Cardinality
  - ☑ Slot Weight
- ☑ Equations
  - ☑ Oriented Equations
- ☑ Term Typing
- ☑ Data Terms
- ☑ Skolem Constants
- ☑ Reified Terms

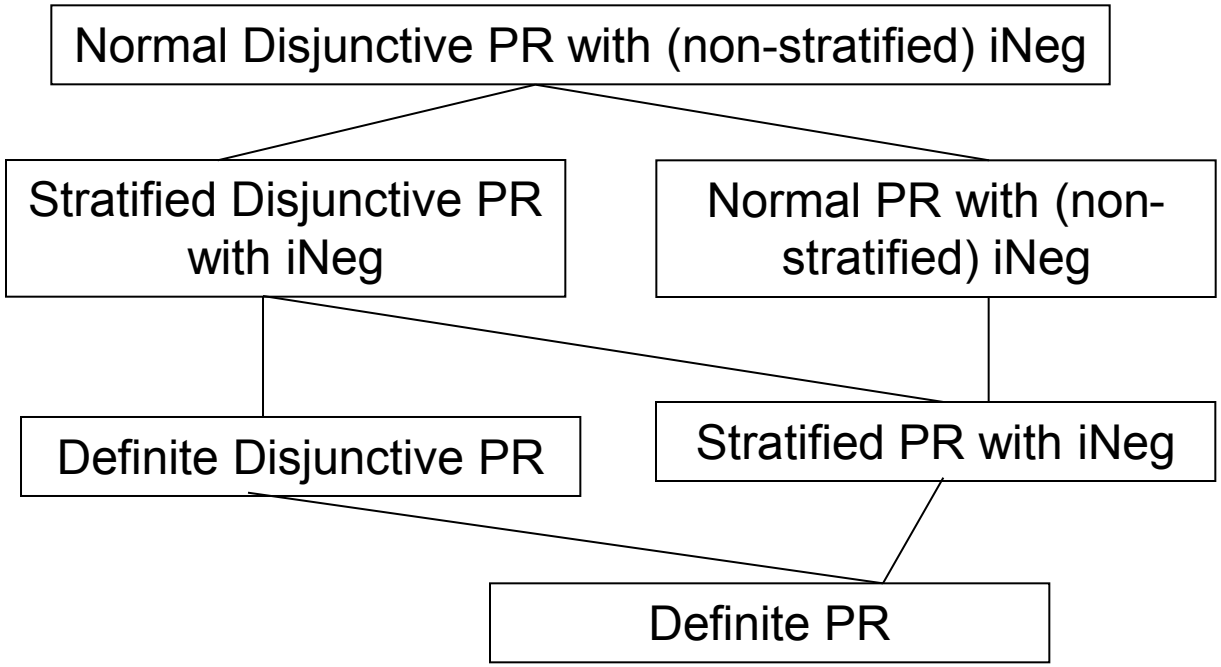
Quantification Options (Check Zero or More)
- ☑ Implicit Closure
- ☑ Slotted Rest Variables
- ☑ Positional Rest Variables

Expression Options (Check Zero or More)
- ☑ Generalized Lists
- ☑ Set-valued Expressions
- ☑ Interpreted Expressions

Example
Derivation RuleML Expressiveness Layering

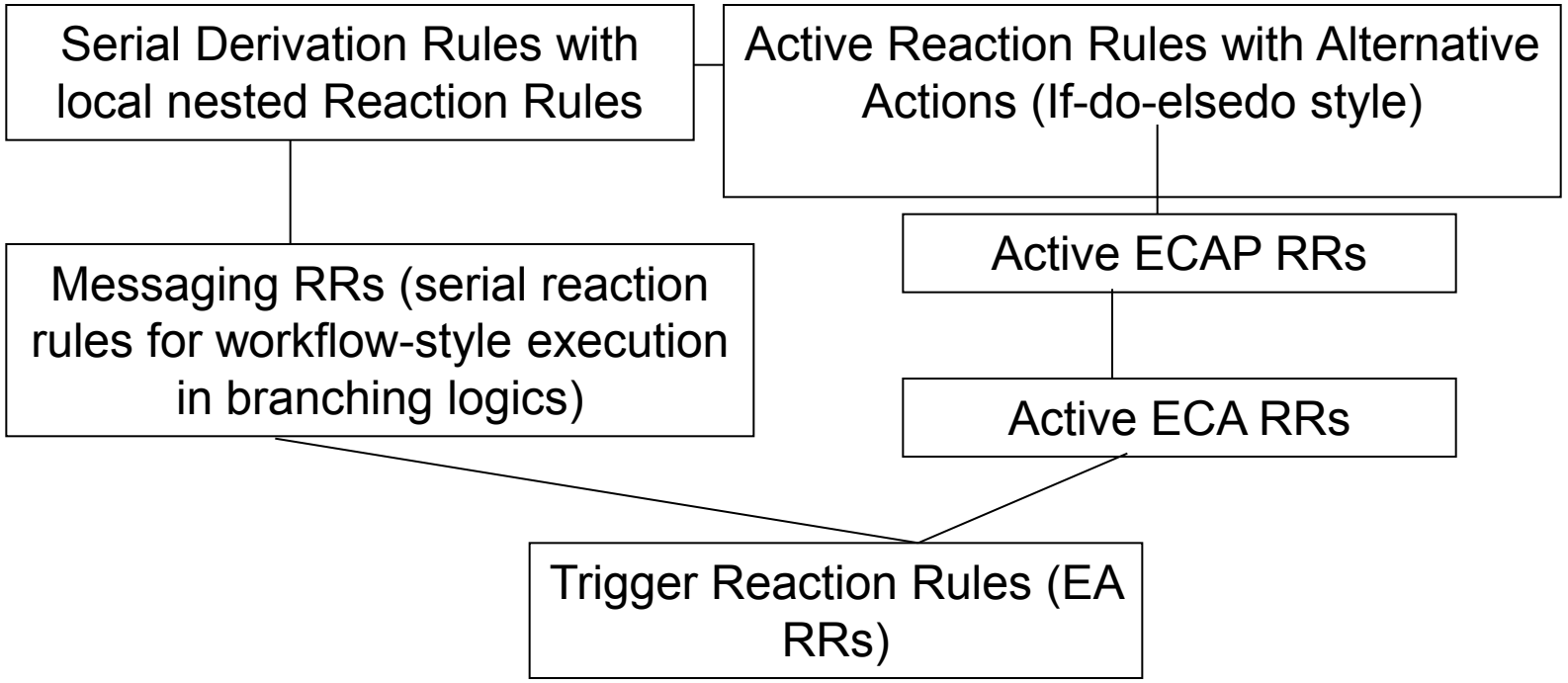


Example - Production RuleML Layering



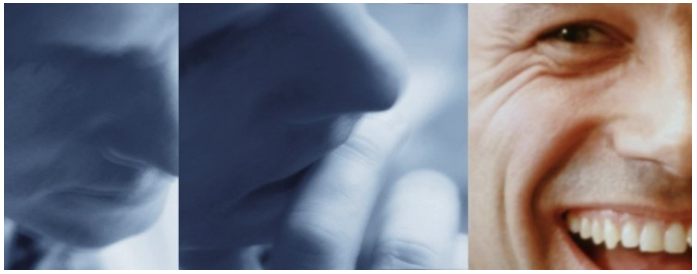
Example

ECA and CEP RuleML Expressiveness Layering



Summary

- Fine grained **syntactic language configuration** (expressiveness) with MYNG (as Relax NG schemas)
- Intended semantics defined by **Semantic Profiles**
- **Semantic Test Cases** for verification and validation of intended semantics for rule programs (and their scoped modules)
- Sorted Logic supports **Types** defined in external semantic ontologies and Meta Model vocabulary



Thank you !

Questions?

Acknowledgement to the members of the Reaction RuleML technical group

44

RuleML Online Community

- RuleML MediaWiki (http://wiki.ruleml.org)
- Mailing lists (http://ruleml.org/mailman/listinfo)
- Technical Groups
  (http://wiki.ruleml.org/index.php/Organizational_Structure#Technical_Groups)
  - Uncertainty Reasoning
  - Defeasible Logic
  - Reaction Rules
  - Multi-Agent Systems
  - …
- RuleML sources are hosted on Github
  (https://github.com/RuleML)

Further Reading – (Reaction) RuleML

- Harold Boley, Adrian Paschke, Omair Shafiq: RuleML 1.0: The Overarching Specification of Web Rules. RuleML 2010: 162-178
  http://dx.doi.org/10.1007/978-3-642-16289-3_15
  http://www.cs.unb.ca/~boley/talks/RuleML-Overarching-Talk.pdf
- Adrian Paschke, Harold Boley, Zhili Zhao, Kia Teymourian and Tara Athan: Reaction RuleML 1.0: Standardized Semantic Reaction Rules, 6th International Conference on Rules (RuleML 2012), Montpellier, France, August 27-31, 2012
  http://link.springer.com/chapter/10.1007%2F978-3-642-32689-9_9
  http://www.slideshare.net/swadpasc/reaction-ruleml-ruleml2012paschketutorial
- Adrian Paschke: Tutorial on Semantic Complex Event Processing and Reaction RuleML, atDemAAL 2013 - Dem@Care Summer School on Ambient Assisted Living, 16-20 September 2013, Chania, Crete, Greece
  http://www.slideshare.net/swadpasc/dem-aal-semanticceppaschke
- Paschke, A., Boley, H.: Rule Markup Languages and Semantic Web Rule Languages, in Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, ISBN:1-60566-402-2, 2009
  http://www.igi-global.com/chapter/rule-markup-languages-semantic-web/35852
  Sample chapter: http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=18533385
- Paschke, A., Boley, H.: Rules Capturing Event and Reactivity, in Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, ISBN:1-60566-402-2, 2009
  http://www.igi-global.com/book/handbook-research-emerging-rule-based/465
  Sample Chapter: http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=16435934&article=0&fd=pdf
- Adrian Paschke and Harold Boley: Rule Responder: Rule-Based Agents for the Semantic-Pragmatic Web, in Special Issue on Intelligent Distributed Computing in International Journal on Artificial Intelligence Tools (IJAIT), V0l. 20,6, 2011
  https://www.researchgate.net/publication/220160498

Further Reading – RuleML Test Cases

- Adrian Paschke: Verification, Validation, Integrity of Rule Based Policies and Contracts in the Semantic Web, 2nd International Semantic Web Policy Workshop (SWPW'06), Nov. 5-9, 2006, Athens, GA, USA
  http://ceur-ws.org/Vol-207/paper01.pdf

- Adrian Paschke, Jens Dietrich, Adrian Giurca, Gerd Wagner, Sergey Lukichev: *On Self-Validating Rule Bases,* Proceedings of 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006), Athens, Georgia, USA (6th November 2006), November 2006
  http://www.rewerse.net/publications/rewerse-description/REWERSE-RP-2006-166.html
  http://km.aifb.kit.edu/ws/swese2006/final/paschke_full.pdf
  http://www.researchgate.net/publication/228359327_On_self-validating_rule_bases

- Adrian Paschke and Jens Dietrich: On the Test-Driven Development and Validation of Business Rules. ISTA 2005: 31-48
  http://subs.emis.de/LNI/Proceedings/Proceedings63/article3604.html

- Adrian Paschke: The ContractLog Approach Towards Test-driven Verification and Validation of Rule Bases - A Homogeneous Integration of Test Cases and Integrity Constraints into Evolving Logic Programs and Rule Markup Languages (RuleML), in IBIS 10/2005
  http://rbsla.ruleml.org/docs/ContractLog_VVI.pdf