Active Rules in the Semantic Web: Dealing with Language Heterogeneity

Wolfgang May

José Júlio Alferes **Ricardo Amador**

Institut für Informatik, Universität Göttingen, Germany

CENTRIA, Universidade Nova de Lisboa, Portugal

Supported by the EU Network of Excellence

RuleML 2005, Galway, Ireland, Nov. 11, 2005

Motivation and Goals

(Semantic) Web:

- XML: bridge the heterogeneity of data models and languages
- RDF, OWL provide a computer-understandable semantics

... same goals for describing behavior:

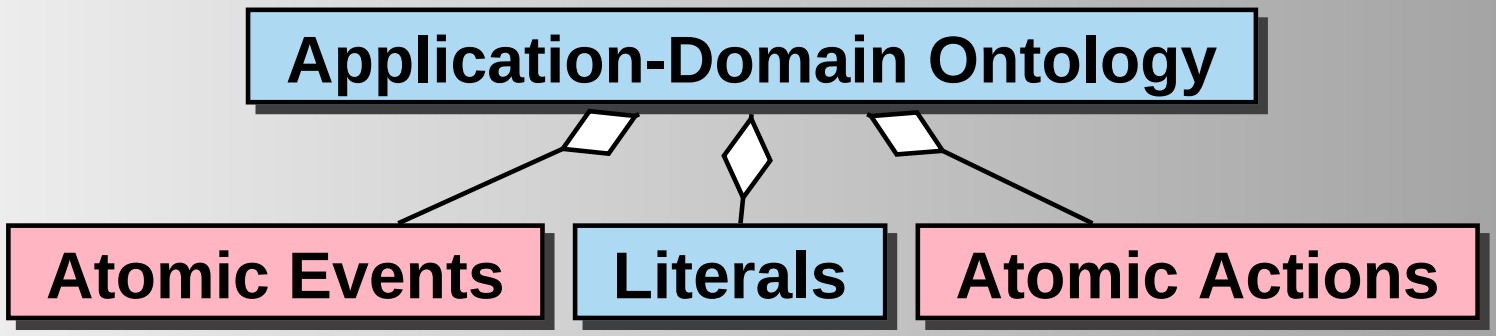
- description of behavior *in* the Semantic Web
- semantic description *of* behavior

Event-Condition-Action Rules are suitable for both goals:

- operational semantics
- ontology of rules, events, actions

Requirements

- Domain languages also describe behavior:



- combine application-dependent semantics with generic concepts of behavior
- Ontology
- Markup
- Operational Semantics

Analysis of Rule Components

... have a look at the clean concepts:
"On Event check Condition and then do Action"

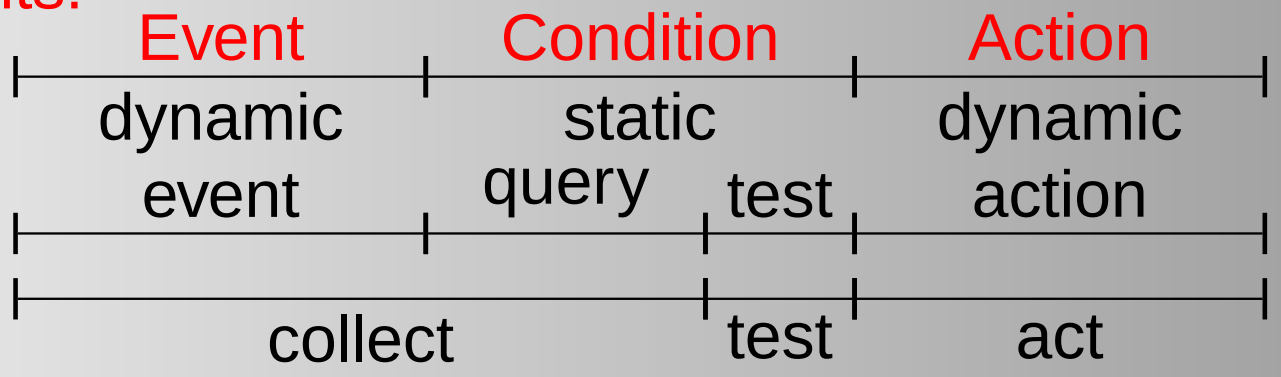
- Event: specifies a rough restriction on what *dynamic* situation probably something has to be done.
  Collects some parameters of the events.
- Condition: specifies a more detailed condition, including *static* data if actually something has to be done.
  $\Rightarrow$ evaluate a ((Semantic) Web) query.
- Action: actually *does* something.

Example

"if a flight is offered from FRA to LIS under 100E and I have no lectures these days then do ..."

Clean, Declarative “Normal Form”

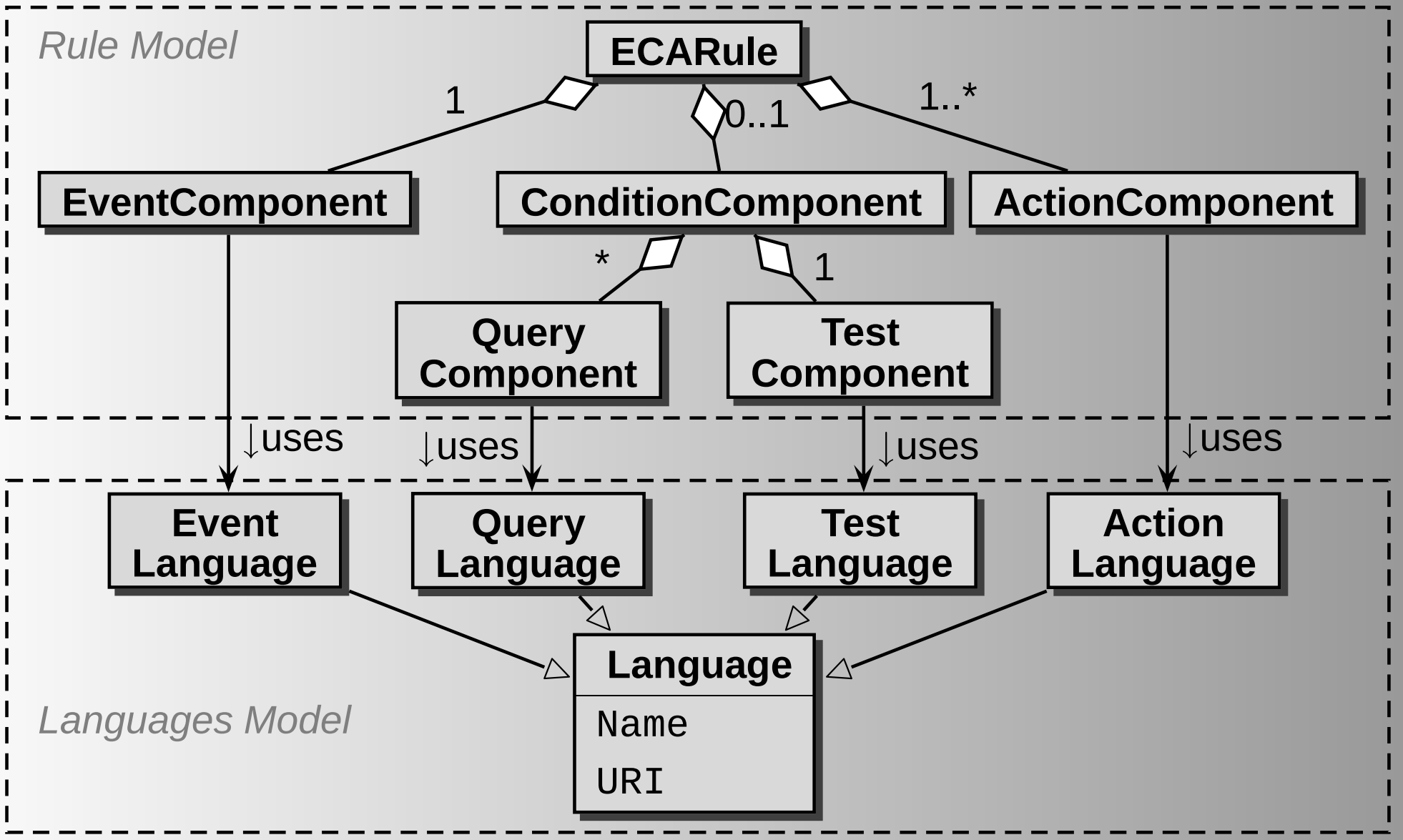
Rule Components:



- Event: detect just the dynamic part of a situation,
- Query: then obtain additional information by queries,
- Test: then evaluate a *boolean* condition,
- Action: then actually do something.

- Component sublanguages: heterogeneous
- Communication between components: logical variables

Modular ECA Concept: Rule Ontology



Rule Markup: ECA-ML

<!ELEMENT rule (event,query*,test?,action+) >

<eca:rule *rule-specific attributes*>
  <eca:event *identification of the language* >
    event specification, probably binding variables
  </eca:event>
  <eca:query *identification of the language* > <!-- there may be several queries -->
    query specification; using variables, binding others
  </eca:query>
  <eca:test *identification of the language* >
    condition specification, using variables
  </eca:test>
  <eca:action *identification of the language* > <!-- there may be several actions -->
    action specification, using variables, probably binding local ones
  </eca:action>
</eca:rule>

Example

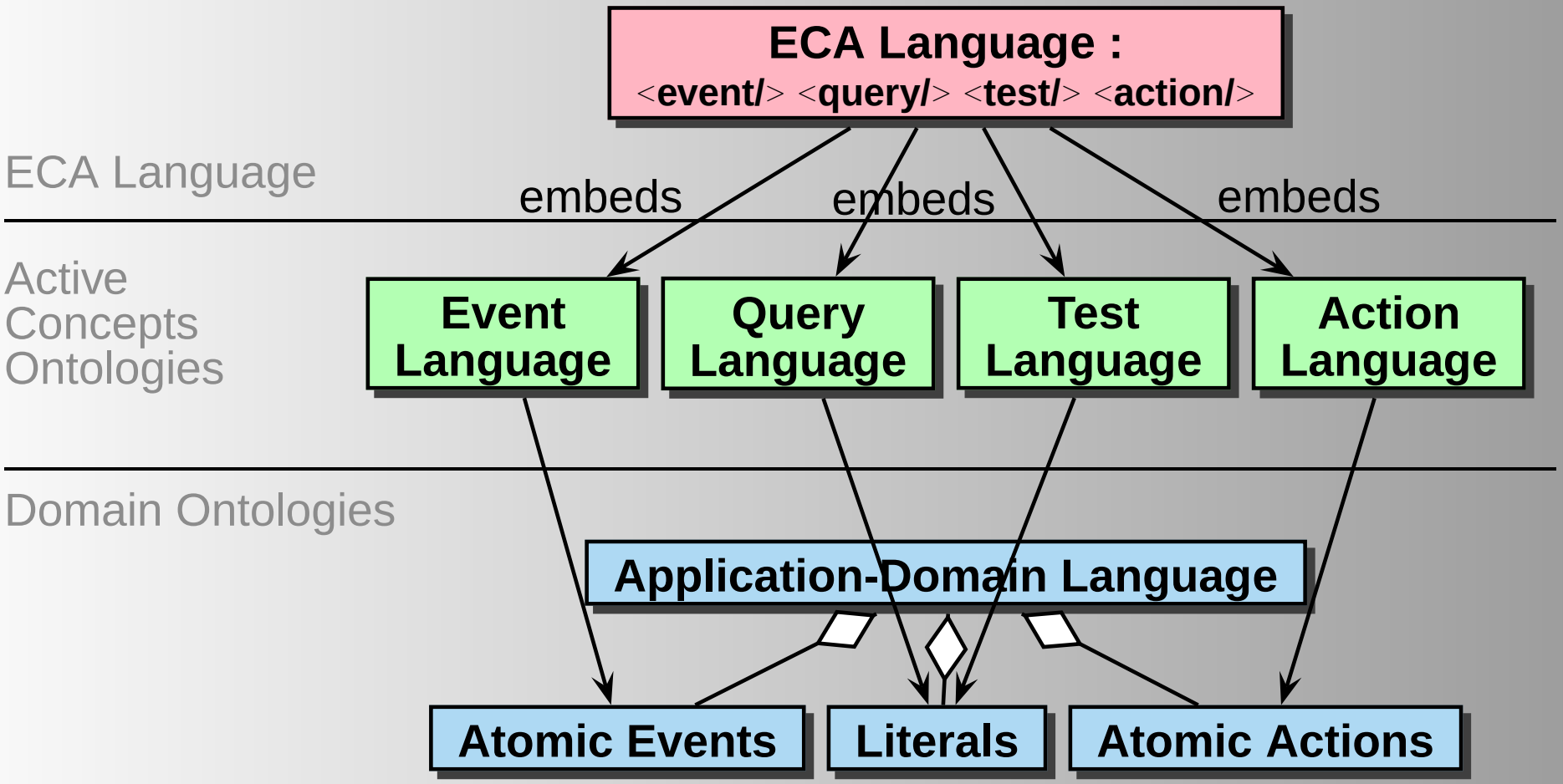
Sample Event:

```
<travel:cancel-flight flight="LH123">
    <travel:reason>bad weather</travel:reason>
</travel:cancel-flight>
```

```
<eca:rule>
  <eca:event xmlns:travel="www.travel.com" >
    <eca:atomic-event>
      <travel:cancel-flight flight="$flight"/>
    <eca:atomic-event>
  </eca:event>
  <eca:query> . . .  </eca:query>
  <eca:test> . . .  </eca:test>
  <eca:action > do something with $flight </eca:action>
</eca:rule>
```

Embedding of Languages

... there are not only atomic events and actions.

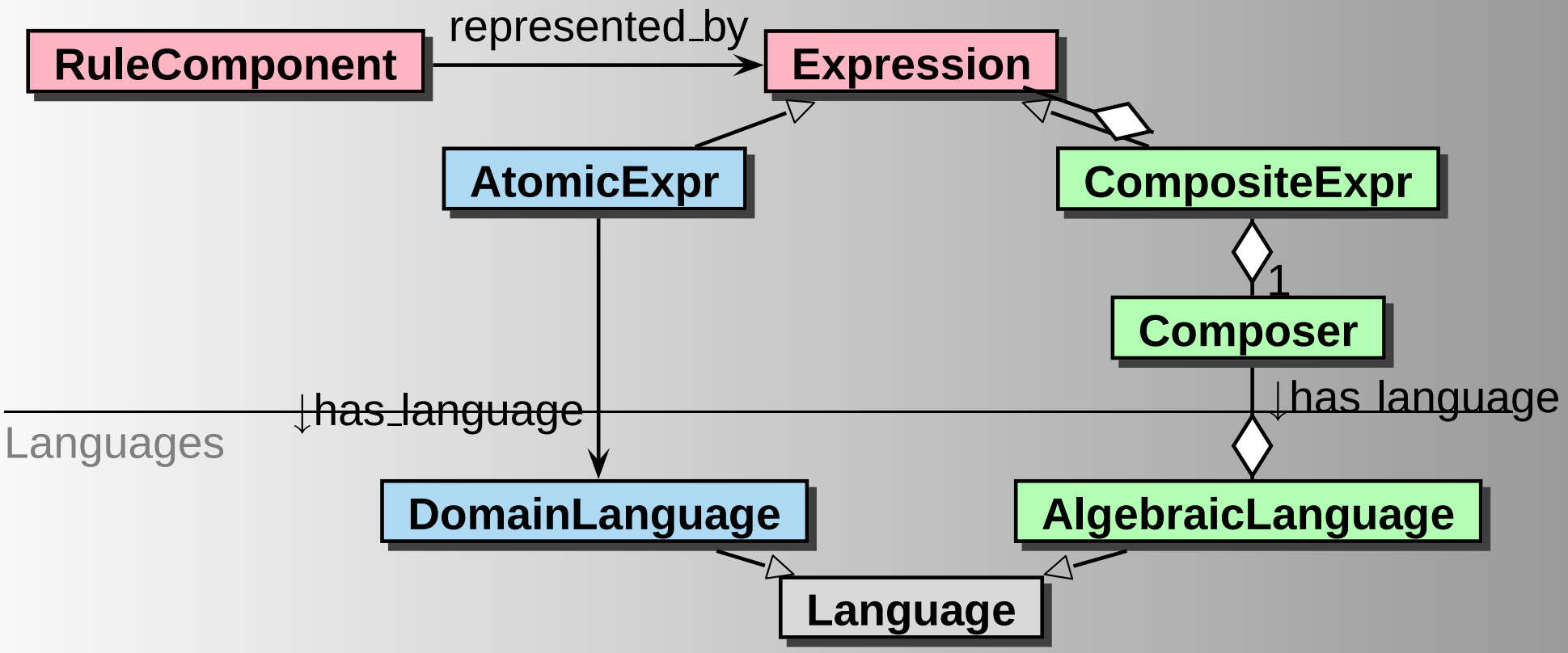


Sublanguages for Active Concepts

Algebraic Languages as a Generic Concept

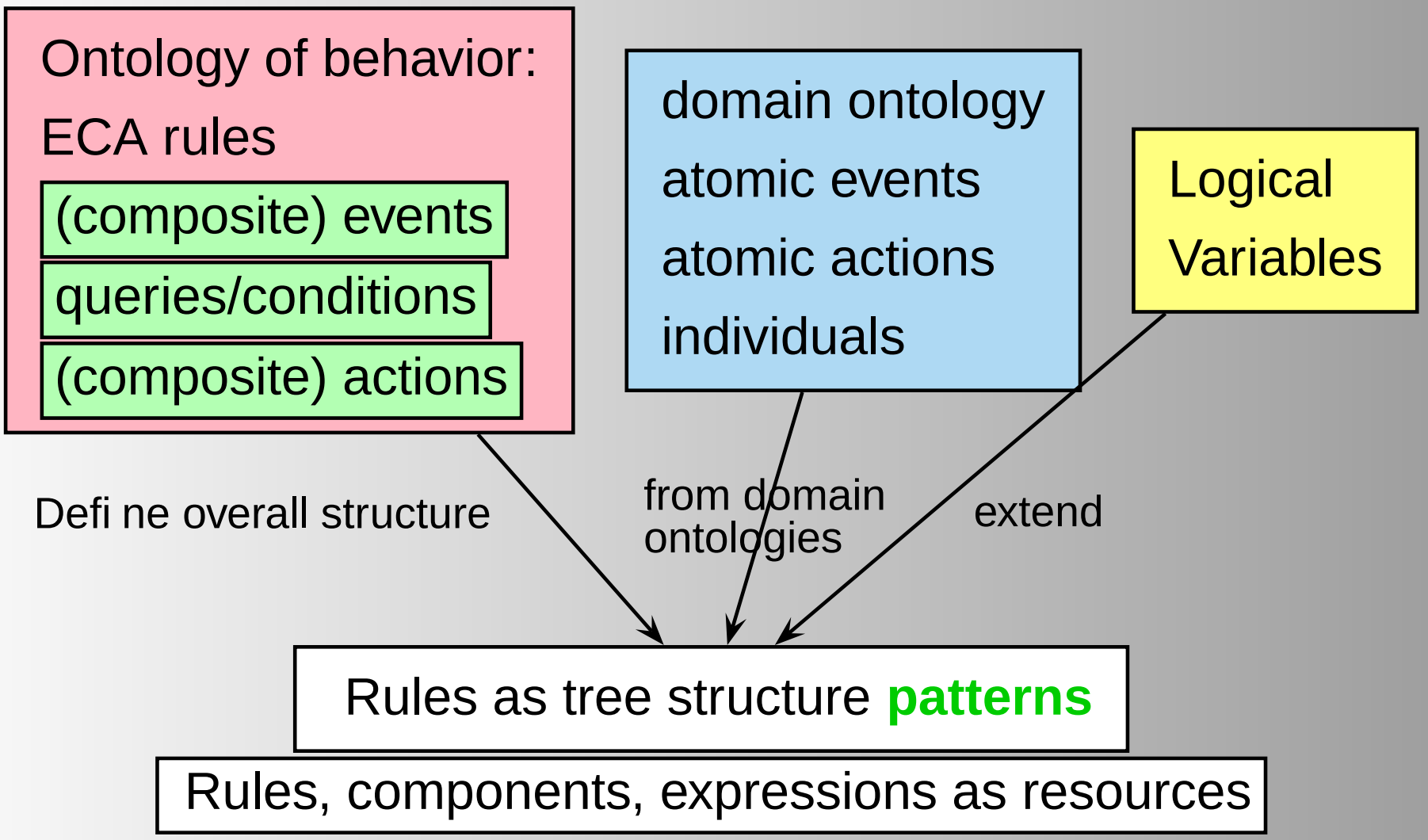
- Algebra terms: tree structures; expressions
  - events: event algebras (e.g. SNOOP)
  - queries
  - tests: boolean algebra
  - actions: process algebras (e.g. CCS)

Structure of Expressions



- tree corresponds to syntactical and semantical structure
- as operator trees: “standard” XML markup of terms
- RDF markup as languages
- every expression can be associated with its language

ECA Rule Markup



Sample Markup (Event Component)

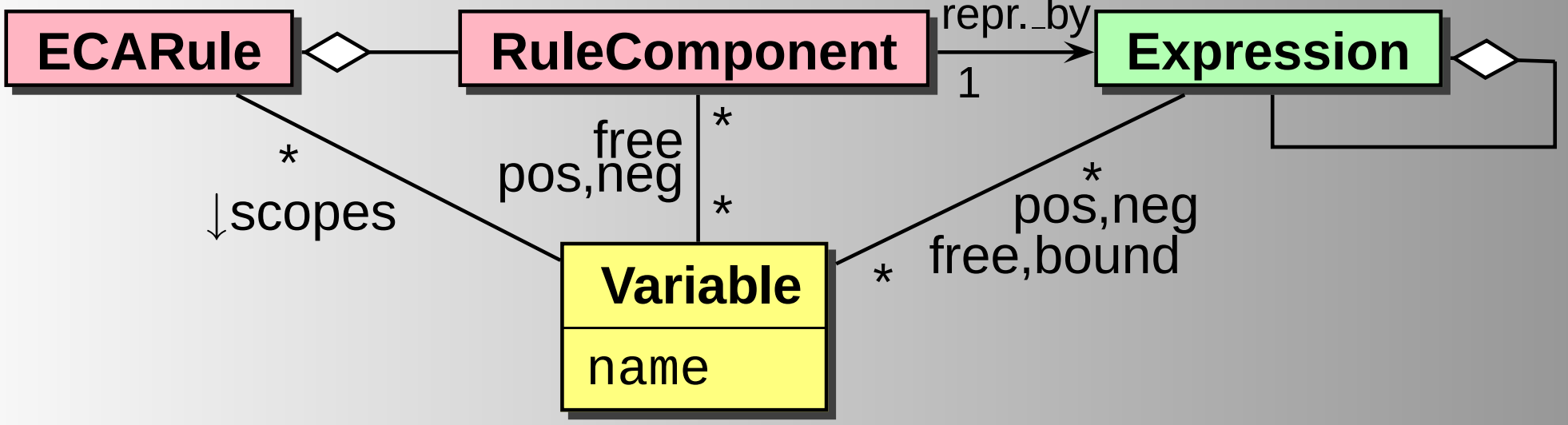
```
<eca:rule xmlns:travel="...">
  <eca:variable name="theSeq">
    <eca:event xmlns:snoop="...">
    <snoop:sequence>
        <eca:atomic-event>
          <travel:delay-flight flight="$Flight" minutes="$Minutes"/>
        </eca:atomic-event>
        <eca:atomic-event>
          <travel:cancel-flight flight="$Flight"/>
        </eca:atomic-event>
      </snoop:sequence>
    </eca:event>
  </eca:variable>
    :
</eca:rule>
```

binds variables:

- Flight, Minutes: by matching
- theSeq is bound to the sequence of events that matched the pattern

Rule Semantics

- Deductive rules: variable bindings Body$\rightarrow$Head
- communication/propagation of information by *logical variables*:
  $E \overset{+}{\rightarrow} Q \rightarrow T$ & A
- safety as usual (extended with technical details ...)



Operational Semantics of Rules

- Event: fires the rule
  - returns the sequence that matched the event
  - optional: variable bindings
- Query: obtain additional static information
  - returns the answer/set of answers
  - optional: for each answer, restrict/extend variable bindings (join semantics)
- Condition:
  - check a boolean condition, constrain variable bindings
- Action:
  - do something by using the variable bindings.

Binding and Use of Variables

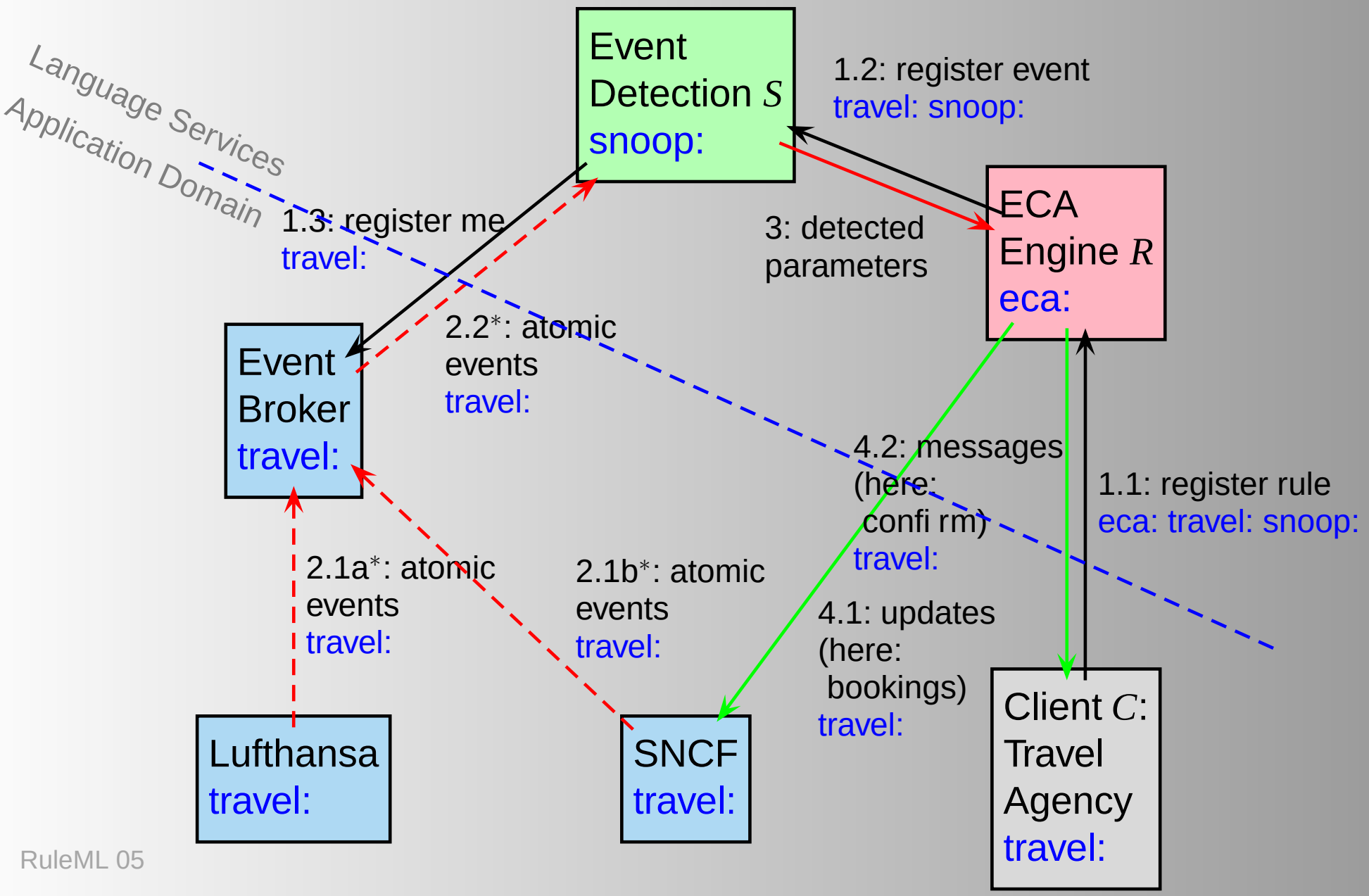
- Variables can be bound to values, XML fragments, RDF fragments, and (composite) events
- Logic Programming (Datalog, F-Logic): variables occur free in patterns.
  Markup uses XSLT-style
  <variable name="*var-name*"> and $*var-name*
  inside component expressions.
- functional style (Event algebras, SQL, OQL, XQuery): expressions return a value/fragment.
  ⇒ must be bound to a variable to be kept and reused.
  <variable name="*var-name*">*language-expr* </variable>
  on the rule level around component expression.

Engines – Service-Based Architecture

Language Processors as Web Services:

- ECA Rule Execution Engine employs other services for E/Q/T/A parts:
  nodes register their rules at the engines; processing is done by the engine
- dedicated services for each of the event/action languages
  e.g., composite event detection engines
- dedicated services for domain-specific issues:
  raising and communicating events, predicates, executing actions/updates
- query languages often implemented directly by the Web nodes (portals and data sources)

Architecture



Communication between Engines

ECA engine, event detection engines, query services etc.

- (sub)terms have well-defined results:
  variable bindings and/or result value
- XML markup for vertical communication of results and variable bindings:

```
<eca:variable-bindings>
  <eca:tuple>
    <eca:variable name="name" ref="URI"/>
    <eca:variable name="name"> any value </eca:variable>
      :
  </eca:tuple>
  <eca:tuple> . . . </eca:tuple>
      :
  <eca:tuple> . . . </eca:tuple>
</eca:variable-bindings>
```

Publications & Details

- REWERSE I5-D4: “Models and Languages for Evolution and Reactivity”: Everything + examples
- RuleML05: languages and their markup, communication and rule execution model
- ODBASE05: ontology of rules, rule components and languages, and the service-oriented architecture
- several additional details
- Prototype