Rule-Based Active Domain Brokering for the Semantic Web

Erik Behrends, Oliver Fritzen, Tobias Knabke,
Wolfgang May, Franz Schenk

Institut für Informatik, Universität Göttingen,
Germany

{behrends,fritzen,knabke,may,schenk}@informatik.uni-goettingen.de

Supported by the EU Network of Excellence

RR 2007, Innsbruck, Austria, June 8, 2007

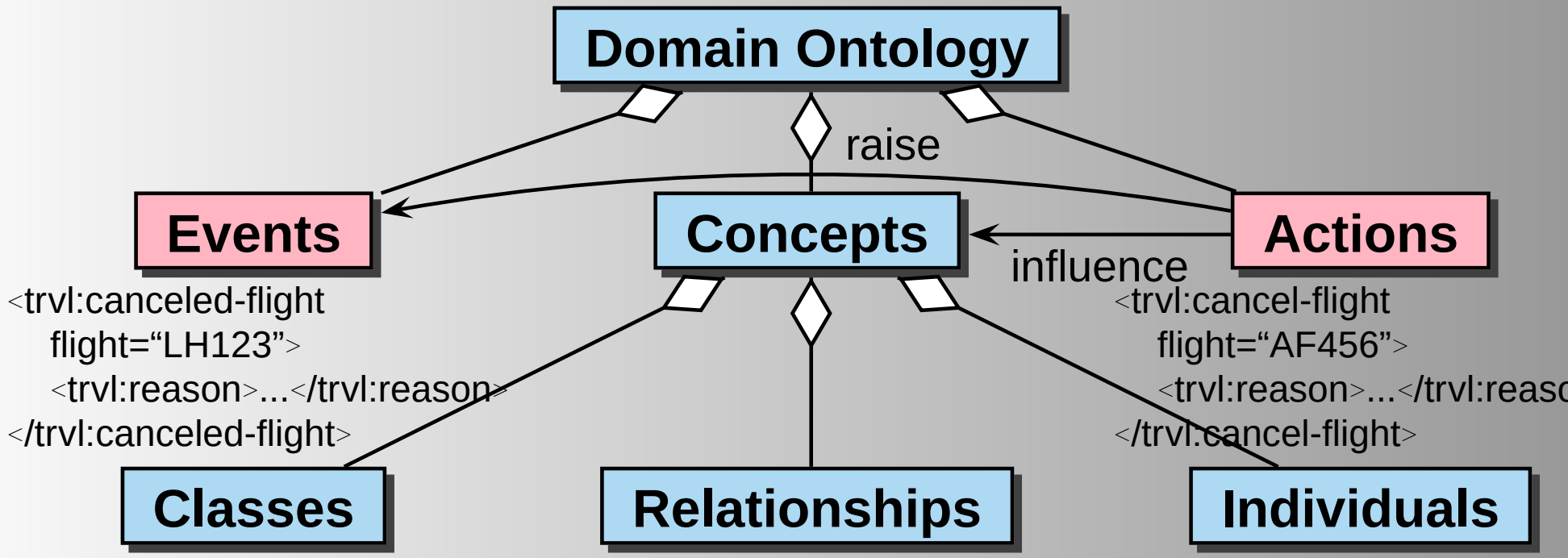
MARS

Modular Active Rules on the Semantic Web

- Rule-based description of behavior in the Semantic Web
- Rules are themselves objects in the Semantic Web
  - OWL Ontology of (Active) Rules
  - Rules as RDF data
  - which will finally allow for reasoning about rules.
- Paradigm: ECA Rules
  "On Event check Condition and then do Action"
- modular, declarative specification
- subontologies/-languages for specifying *Events*, *Conditions*, *Actions*,
- Services that implement these sublanguages.

Domain Ontologies with Active Notions

- Domain languages also describe behavior:



- Ontology of behavior aspects:
- correlate and axiomatize actions, events and state

Ontologies with Active Notions (Cont'd)

There are not only atomic events and actions.

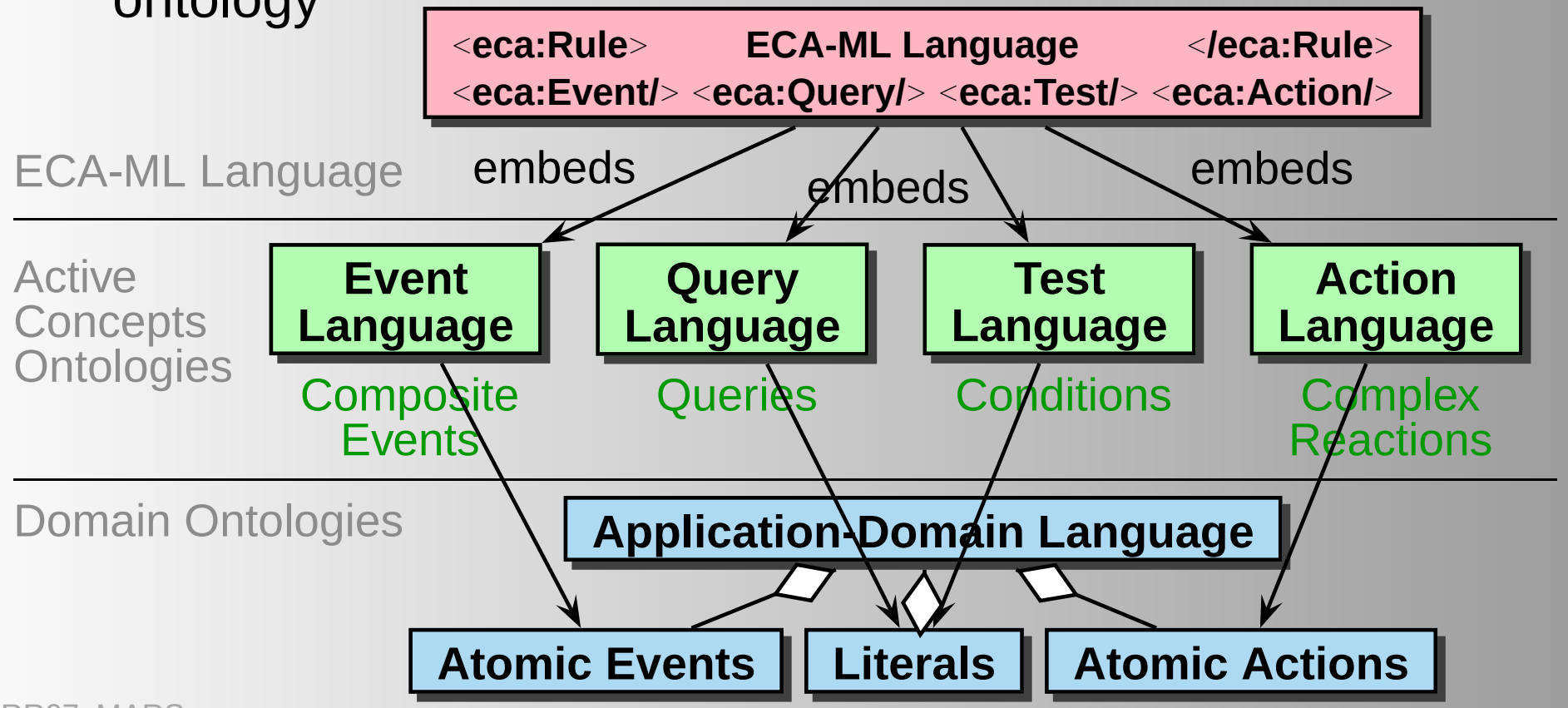
Ontologies also define the following:

- Derived/complex events, specified by some formalism over simpler events (usually an event algebra, e.g., SNOOP)
- composite actions = processes,
  specified by a process algebra over simpler actions, e.g. CCS

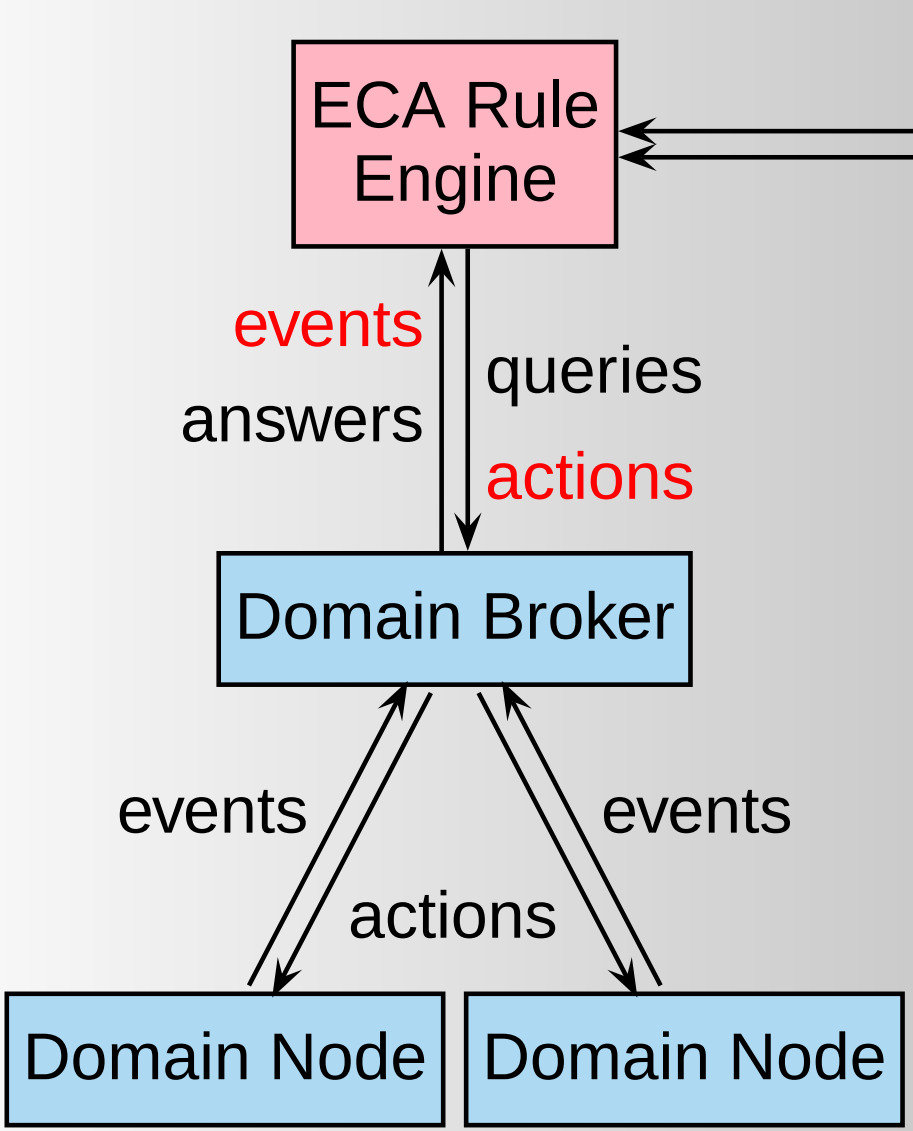
MARS' Underlying Paradigm: ECA Rules

"On Event check Condition and then do Action"

- paradigm of *Event-Driven Behavior*,
- modular, declarative specification in terms of the domain ontology



MARS: General Architecture (simplified)



Domain brokers forward actions a events, and process queries

- Derived Event Specifications: EC(raise-E)-Rules
- Composite Action Specification (on-A)CA-Rules

Domain nodes execute actions, raise events, and answer querie

- Action Implementation Spec's local (on-A)CA-Rules

Domain Broker

Initialize with an Ontology

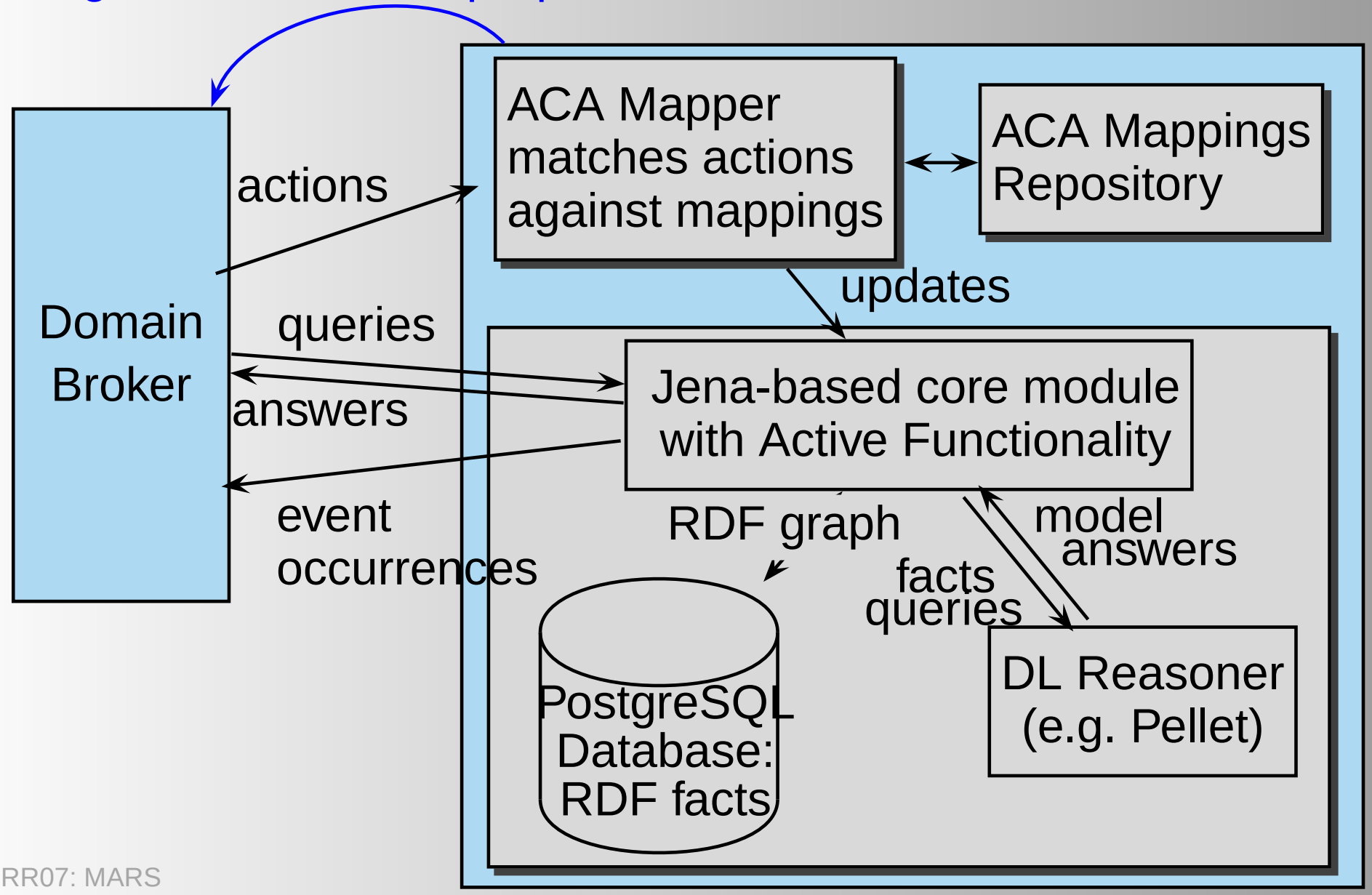
- complete ontology in terms of mars:Class, mars:Property, mars:Event, mars:Action
- the ontology's ECE and ACA rules (using the ECA-ML ontology+markup)
- domain broker registers ECE+ACA rules at the ECA Engine

Domain Nodes

- Each domain node registers at the domain broker which notions (classes, properties, actions) it mars:supports,
- runtime behavior: next slide ...

Architecture of the Domain Node

register for classes, properties, actions



Sample Local ACA Rule of the Domain Nod

- in: an action in XML
- or RDF (graph) fragment containing one {**?A rdf:type mars:Action**}
- implement the action on the local RDF database

```
## sample rule using XQuery-style
IMPLEMENT <travel:schedule-flight/> BY
let $flight := /travel:schedule-flight/@flight
let $captain := /travel:schedule-flight/@captain
return concat(
“INSERT ($flight has-captain $captain);”,
for $name in /travel:schedule-flight/cabincrew/@name
let $cabincrew := local:make-person-uri($name)
return “INSERT ($flight has-cabincrew $cabincrew);”)
```

Summary

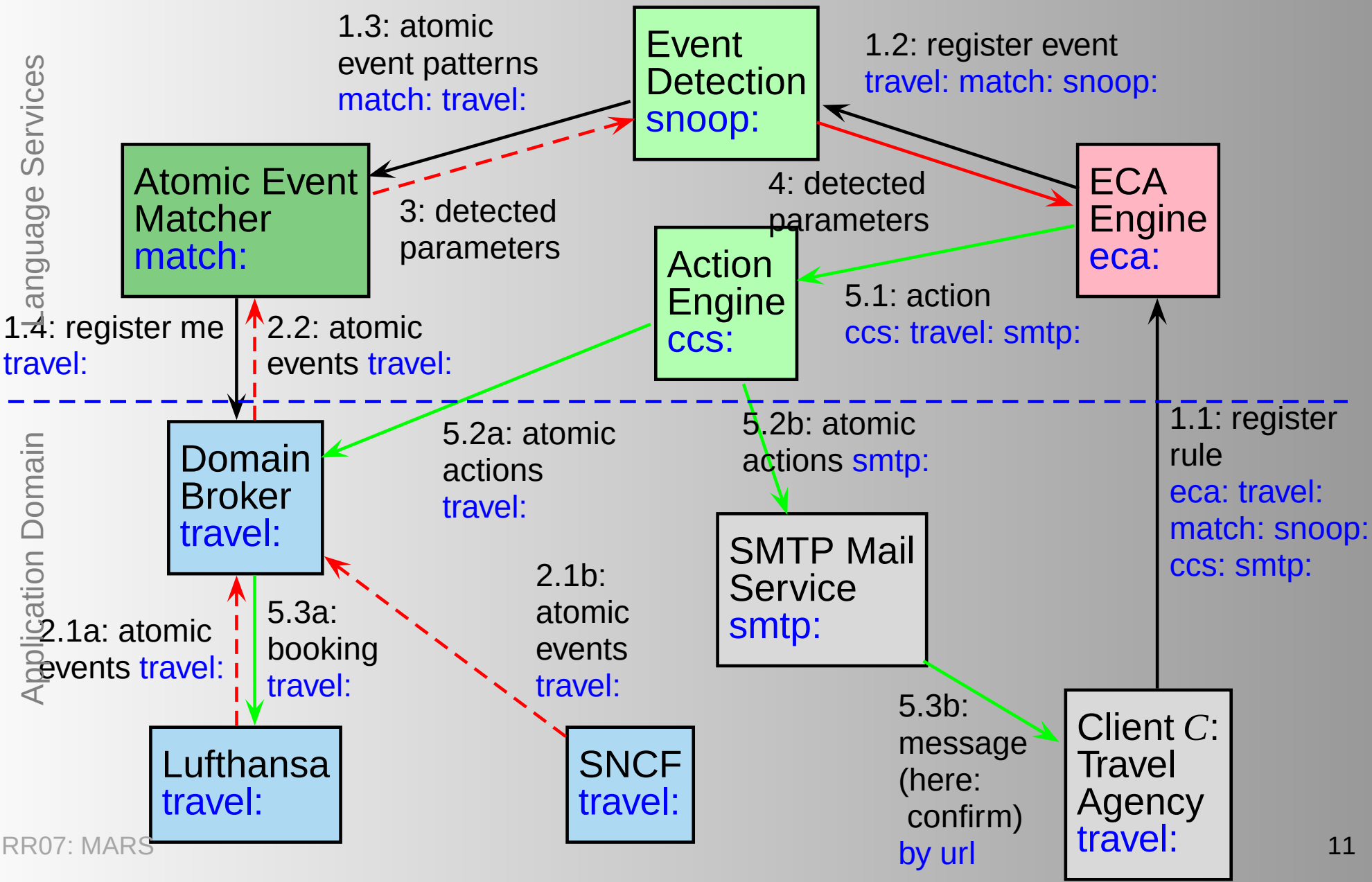
- describe events and actions of an application within its RDF/OWL ontology
- rules on different levels of abstraction/locality
- architecture: functionality provided by specialized nodes
- outsourcing ECE+ACA rules as much as possible to existing ECA infrastructure.

Further Information

- REWERSE Deliverable I5-D6: “An RDF/OWL-Level Spec. of Evolution and Behavior in the Semantic Web” and papers at ODBASE05, WebR06, RuleML05+06, PPSWR05+06
- MARS proof-of-concept experimental prototype: http://www.semwebtech.org

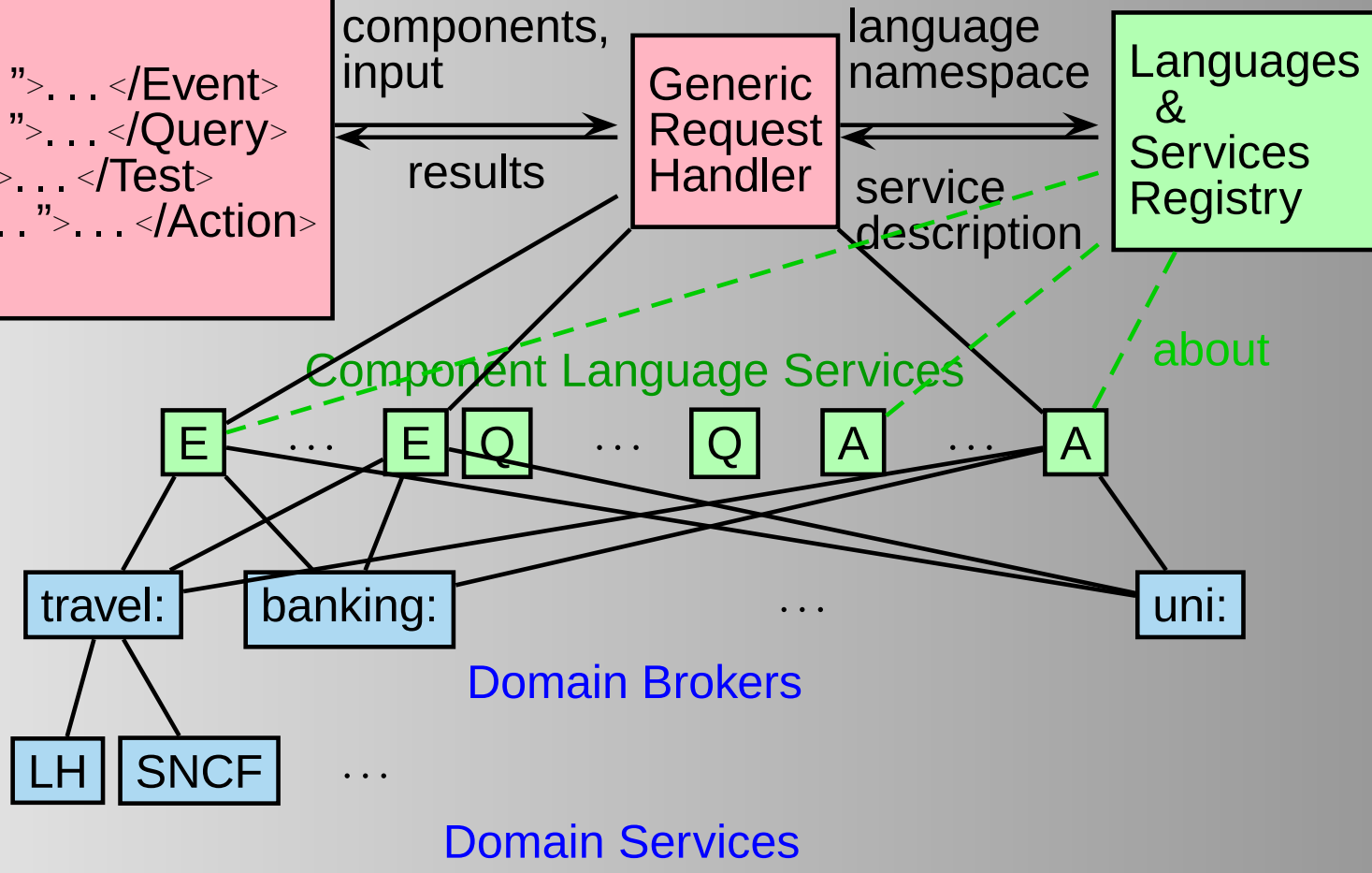
... following: backup slides

Architecture



ECA Architecture

ECA Engine:
<Rule>
  <Event xmlns:ev="...">...</Event>
  <Query xmlns:ql="...">...</Query>
  <Test xmlns:tst="...">...</Test>
  <Action xmlns:act="...">...</Action>
</Rule>



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>

Rule Semantics/Logical Variables

Deductive Rules: $head(X_1,\ldots,X_n) :- body(X_1,\ldots,X_n)$

- bind variables in the body
- instantiate/execute head for each tuple

ECA Rules

- initial bindings from the event
- additional bindings from queries
- restrict by the test
- execute action for each tuple

$$action(X_1,\ldots,X_n) \leftarrow$$
$$event(X_1,\ldots,X_k),\ query(X_1,\ldots,X_k,\ldots X_n),\ test(X_1,\ldots,X_n)$$

Binding and Use of Variables in ECA Rules

$action(X_1, \ldots, X_n) \leftarrow$

$event(X_1, \ldots, X_k),\ query(X_1, \ldots, X_k, \ldots X_n),\ test(X_1, \ldots, X_n)$