An ECA Engine for Deploying Heterogeneous Component Languages in the Semantic Web

Erik Behrends Oliver Fritzen
Wolfgang May Daniel Schubert

Institut für Informatik, Universität Göttingen, Germany

Workshop “Reactivity on the Web”, München, Germany, 31.3.2006

Supported by the EU Network of Excellence

REWERSE©
reasoning on the web

Part of the “General Framework for Evolution and Reactivity in the Web” subproject in REWERSE I5 (Göttingen & Lisbon)

Thesis:
There is not a single formalism/language for describing and implementing behavior in the Semantic Web.

Hypothesis:
Semantical approaches (i.e., not “programming”, but based on an ontology of behavior) follow the *Event-Condition-Action* paradigm.

Contribution:
We show that a general framework approach with modular components covers many existing concepts that will prove useful for behavior in the Semantic Web.

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

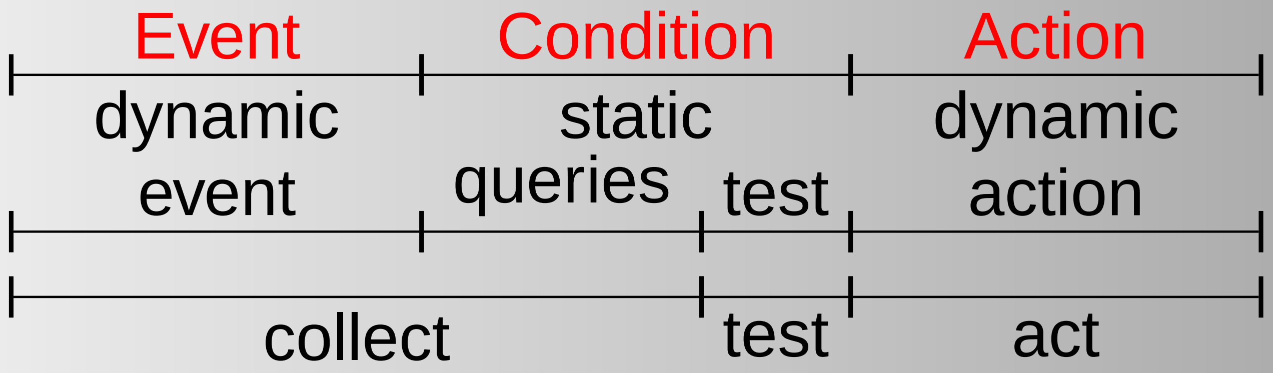
Event-Condition-Action Rules are suitable for both goals:

- operational semantics
- ontology of rules, events, actions

ECA Rules

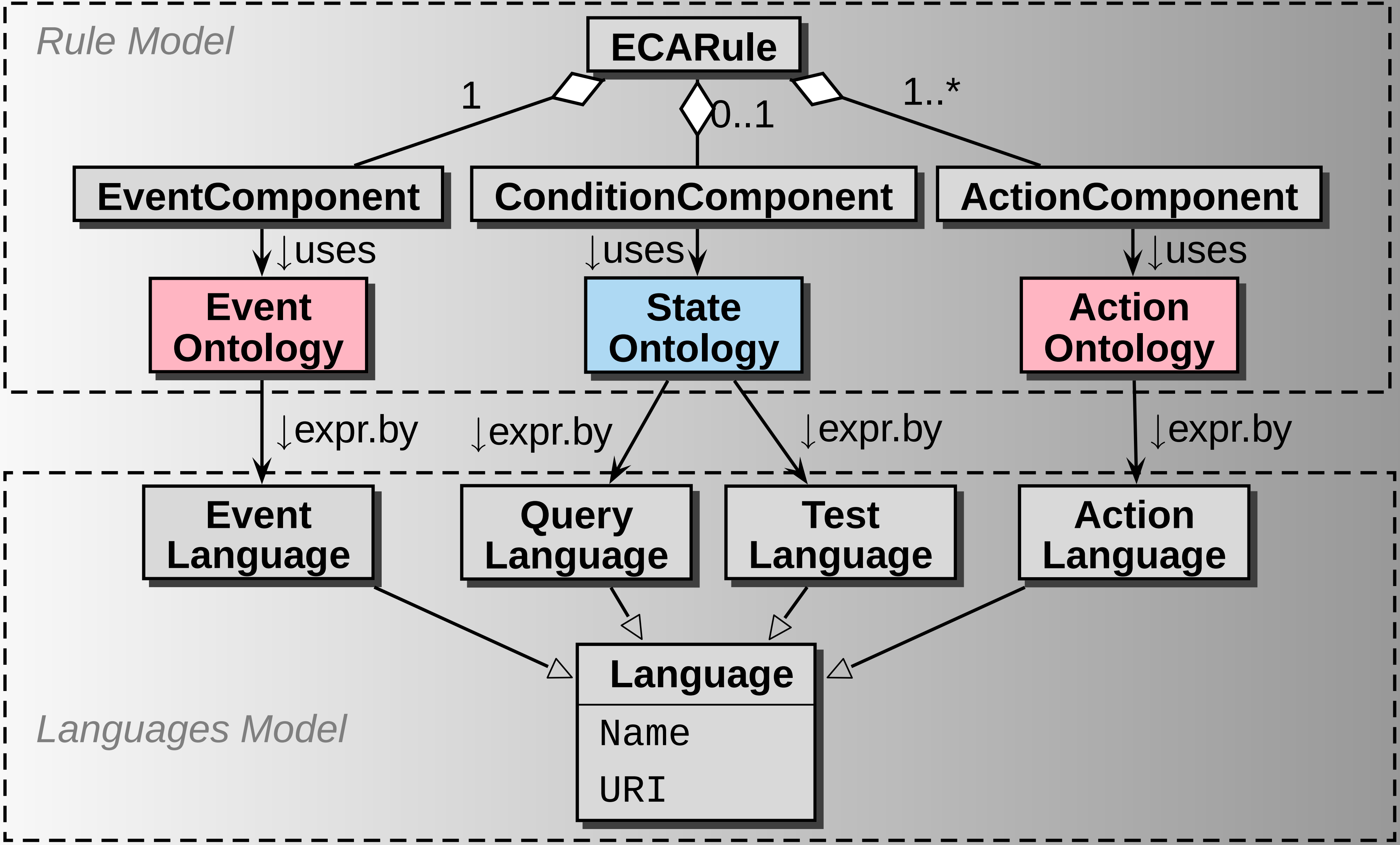
"On Event check Condition and then do Action"

- modular, declarative specification
- sublanguages for specifying *Events*, *Conditions*, *Actions*



- 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.

Modular ECA Concept: Rule Structure



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>
```

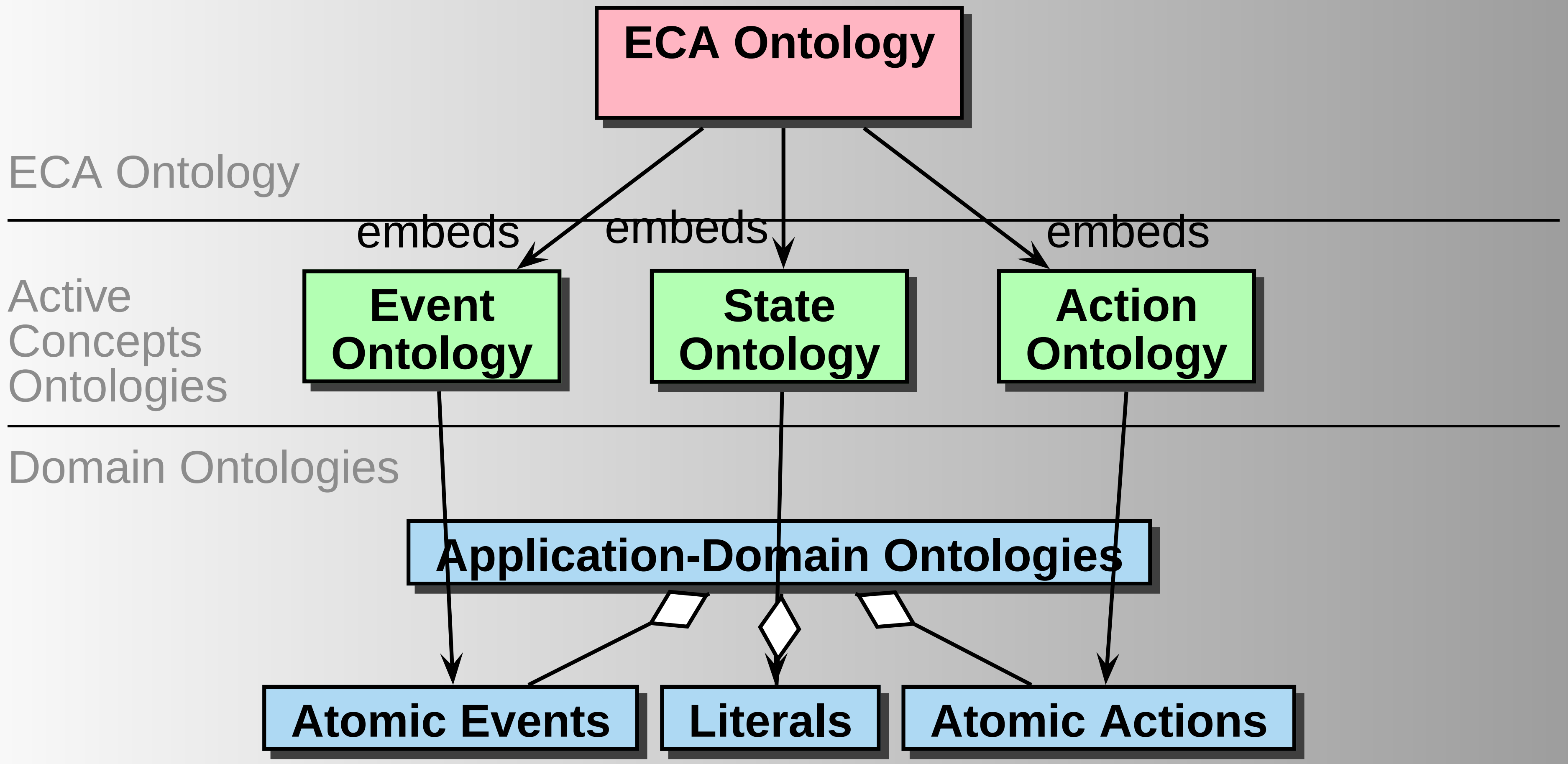
Example

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" >
    <travel:cancel-flight flight="$flight"/>
  </eca:event>
  <eca:query> get $email-address of all passengers of $flight </eca:query>
  <eca:test> . . .  </eca:test>
  <eca:action> tell each $email that $flight is cancelled  </eca:action>
</eca:rule>
```

Combination of Ontologies



Active Concepts Ontologies

- Domains specify atomic events, actions and static concepts

Composite [Algebraic] Active Concepts

- Event algebras: composite events
  - (when) E_1 and some time afterwards E_2 (then do A)
  - (when) E_1 happened and then E_2, but not E_3 after at least 10 minutes (then do A)
  - well-investigated in Active Databases (e.g. SNOOP).
- Process algebras (e.g. CCS)
- different languages, different expressiveness/complexity
- common structure: algebraic languages: natural term markup.

Rule Markup: Example (Stripped)

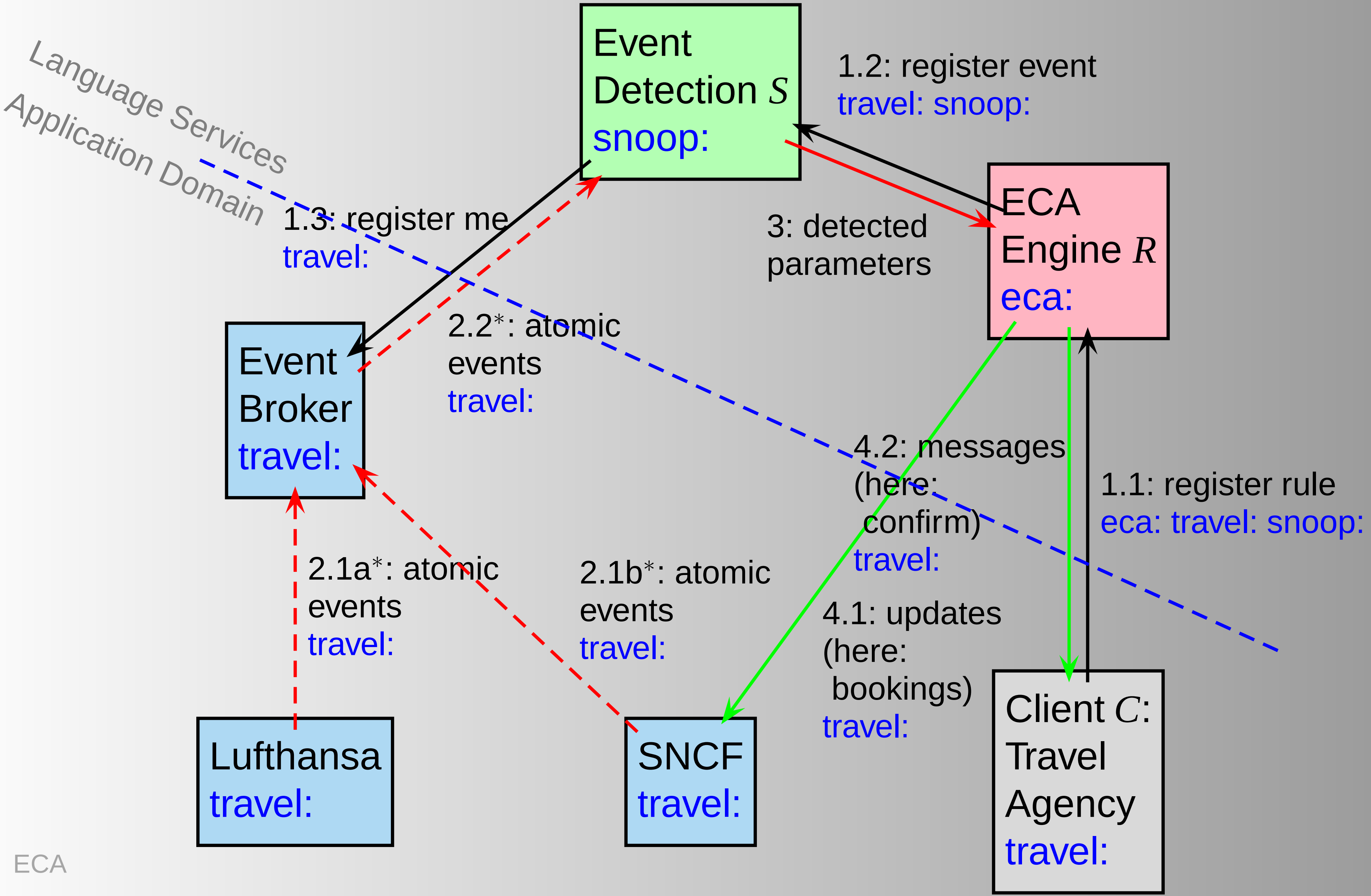
```
<!ELEMENT rule (event,query*,test?,action+) >
<eca:rule xmlns:travel="http://www.travel.de" >
  <eca:event xmlns:snoop="http://www.snoop.org" >
    <snoop:seq> <travel:delay-flight flight="$flight"/> <travel:cancel-flight flight="$flight"/>
    </snoop:seq>
  </eca:event>
  <eca:query>
    <eca:variable name="email">
      <eca:opaque lang="http://www.w3.org/xpath">
        doc("www.lufthansa.de")/flights[code="$flight"]/passenger/@e-mail
      </eca:opaque> </eca:variable> </eca:query>
  <eca:action xmlns:smtp="...">
    <smtp:send-mail to="$email" text="..."/>
  </eca:action>
</eca:rule>
```

Subconcepts and Sublanguages

- e/q/t/a subelements contain a language identification, and appropriate contents
- all rule components, subexpressions are associated with languages corresponding to the ontologies (event languages, action languages, domain languages):
- Algebraic languages:
  processing engine
- Domain Languages:
  Event Broker Services (subscribe) and processors for actions

⇒ Modular concepts with Web-wide services

Service-Oriented Architecture



Rule Semantics/Logical Variables

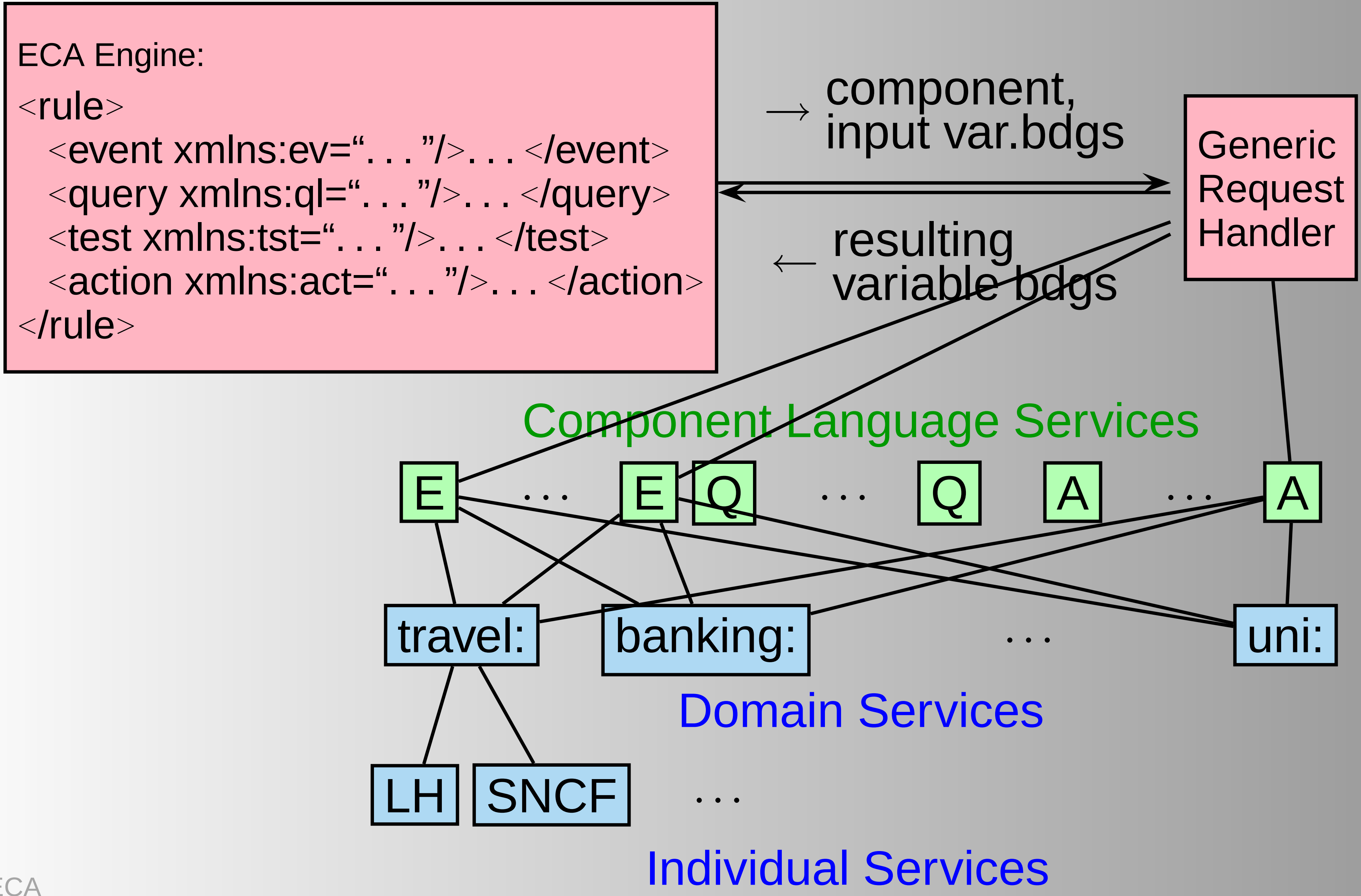
- (sub)terms (events, queries) must have a well-defined input and result/outcome: variable bindings
- information flow between components by logical variables.

Information Flow in ECA Rules

- initial bindings from the event
- additional bindings from queries (+ restriction via join variables)
- restrict by the test
- execute action for each tuple

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

ECA Engine Architecture


ECA Engine:
<rule>
<event xmlns:ev="..."/>... </event>
<query xmlns:ql="..."/>... </query>
<test xmlns:tst="..."/>... </test>
<action xmlns:act="..."/>... </action>
</rule>
→ component, input var.bdgs
← resulting variable bdgs
Generic Request Handler
Component Language Services
E
...
E
Q
...
Q
A
...
A
travel:
banking:
...
uni:
Domain Services
LH
SNCF
...
Individual Services

Tasks

- ECA Engine: Rule Semantics
  - Control flow: registering event component, receiving “firing” answer, continuing with queries etc.
  - Variable Bindings, Join Semantics
- Generic Request Handler: Mediator with Component Engines
  - depending on Service Descriptions
- Component Engines: dedicated to certain Event Algebras, Query Languages, Action Languages
- Domain Services (Portals): atomic events, queries, atomic actions

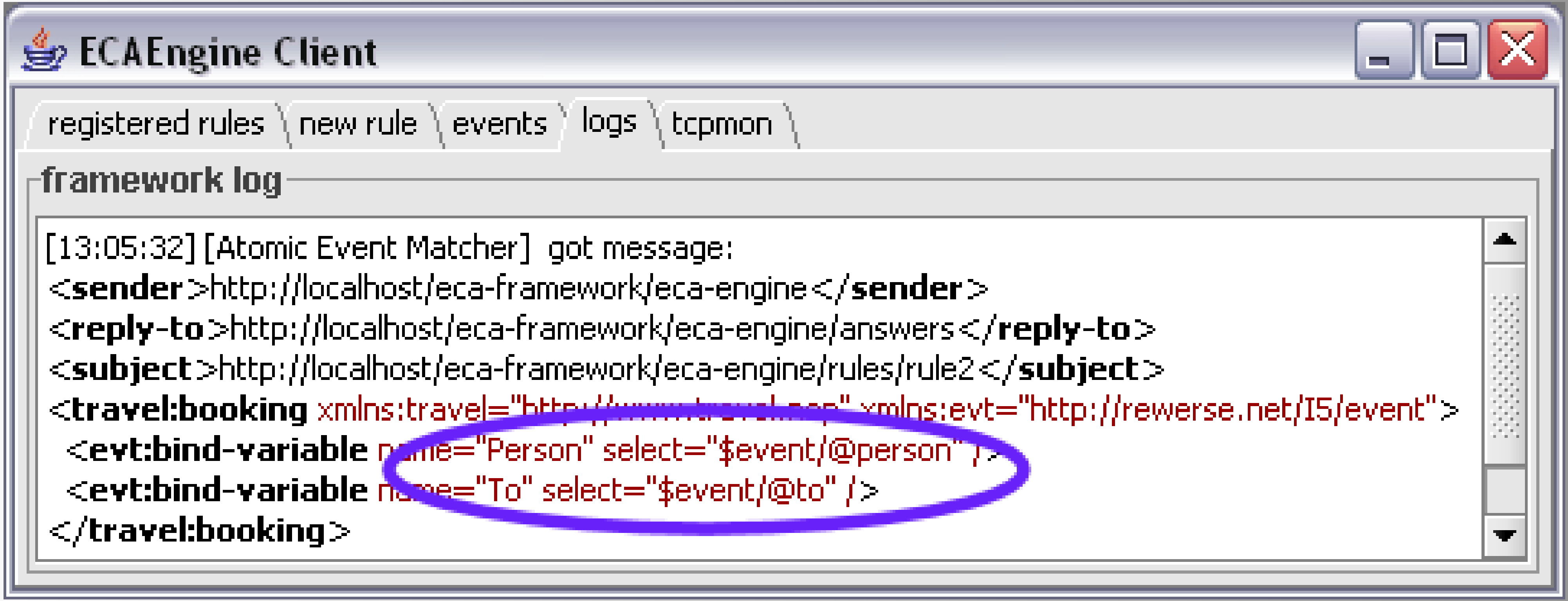
Sample Rule: Outline

```
<travel:booking name=”John Doe” from=”Munich” to=”Paris”/>
```

```
<eca:rule>
  <eca:event>
    <travel:booking xmlns:travel=“http://www.travel.nop”>
      <evt:bind-variable name=“Person” select=“$event/@person”/>
      <evt:bind-variable name=“To” select=“$event/@to”/>
    </travel:booking>
  </eca:event>
  <eca:query> which cars does $Person own at home? </eca:query>
  <eca:query> to which classes do these cars belong? </eca:query>
  <eca:query> which cars of these classes are available at $To </eca:query>
  <eca:query> how much money is $Person willing to spend per day </eca:query>
  <eca:test> which of the cars are available for less than this price? </eca:test>
  <eca:action> report recommended cars </eca:action>
</eca:rule>
```

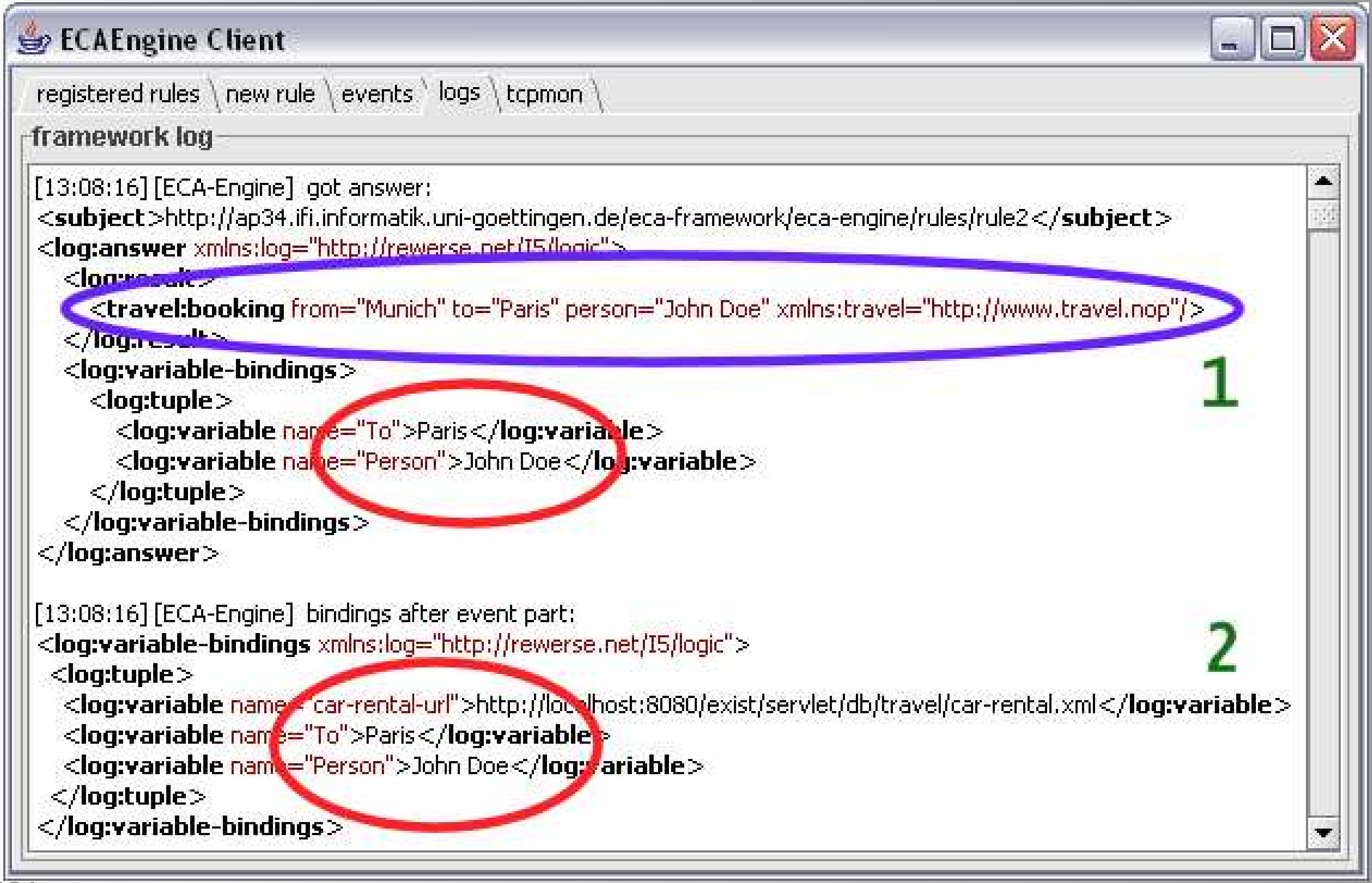
Registration of the Rule

- User registers rule at ECA service
- ECA registers event part at appropriate service (via GRH)
- then wait for an answer.



```
[13:05:32] [Atomic Event Matcher]  got message:
<sender>http://localhost/eca-framework/eca-engine</sender>
<reply-to>http://localhost/eca-framework/eca-engine/answers</reply-to>
<subject>http://localhost/eca-framework/eca-engine/rules/rule2</subject>
<travel:booking xmlns:travel="http://www.travel.org" xmlns:evt="http://rewerse.net/I5/event">
  <evt:bind-variable name="Person" select="$event/@person" />
  <evt:bind-variable name="To" select="$event/@to" />
</travel:booking>
```

Event has been Detected



```
[13:08:16] [ECA-Engine]  got answer:
<subject>http://ap34.ifi.informatik.uni-goettingen.de/eca-framework/eca-engine/rules/rule2</subject>
<log:answer xmlns:log="http://rewerse.net/I5/logic">
  <log:result>
    <travel:booking from="Munich" to="Paris" person="John Doe" xmlns:travel="http://www.travel.nop"/>
  </log:result>
  <log:variable-bindings>
    <log:tuple>
      <log:variable name="To">Paris</log:variable>
      <log:variable name="Person">John Doe</log:variable>
    </log:tuple>
  </log:variable-bindings>
</log:answer>

[13:08:16] [ECA-Engine]  bindings after event part:
<log:variable-bindings xmlns:log="http://rewerse.net/I5/logic">
 <log:tuple>
  <log:variable name="car-rental-url">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
  <log:variable name="To">Paris</log:variable>
  <log:variable name="Person">John Doe</log:variable>
 </log:tuple>
</log:variable-bindings>
```

Communication of Results

- result-bindings-pairs (semantics of expression)

```
<log:answers>
  <log:answer>
    <log:result>
      <!-- functional result -->
    </log:result>
    <log:variable-bindings>
      <log:tuple> . . . </log:tuple>
          :
      <log:tuple> . . . </log:tuple>
    </log:variable-bindings>
  </log:answer>
  <log:answer> . . .  </log:answer>
        :
  <log:answer> . . .  </log:answer>
</log:answers>
```

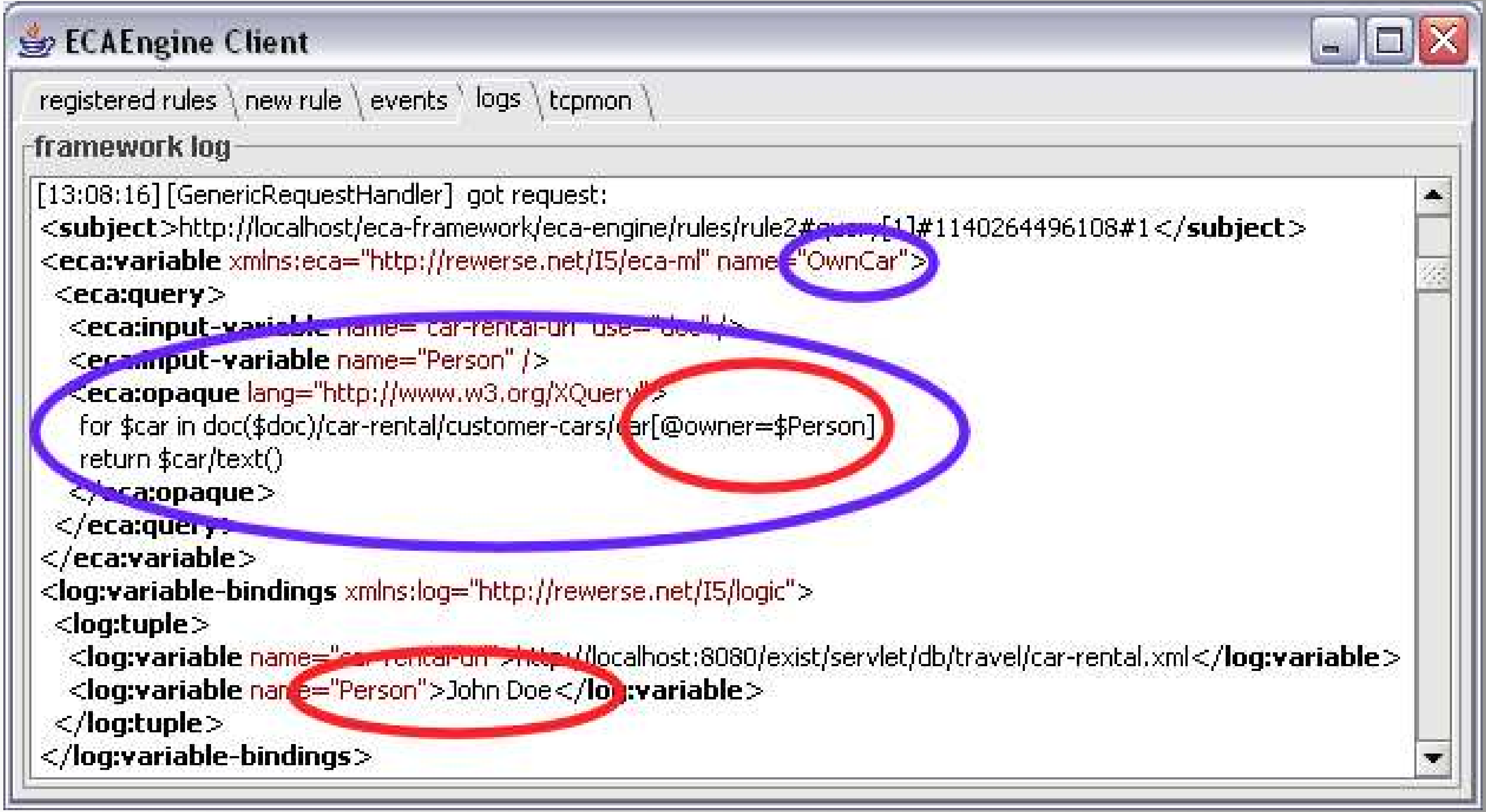
Communication

ECA engine sends component to be processed together with bindings of all relevant variables to GRH.

Generic Request Handler (GRH)

- Submits component (with relevant input/used variable bindings) to appropriate service (determined by namespace/language used in the component)
- if necessary: does some wrapping tasks (for non-framework-aware services)
- receives results and transforms them into flat variable bindings and sends them back to the ECA engine ...
- ... where they are joined with the existing tuples ...
- ... and the next component is processed.

Processing Queries

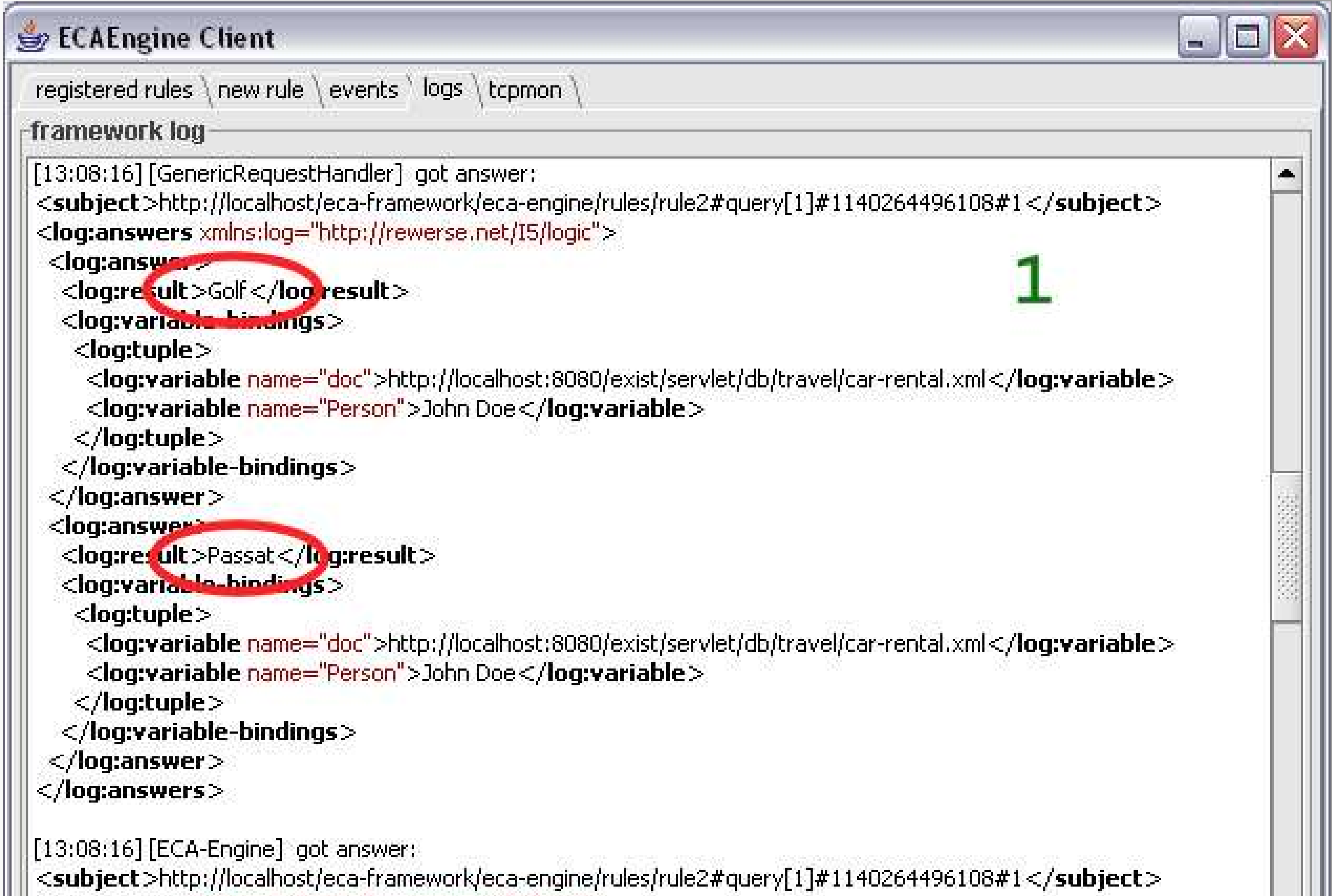


```
[13:08:16] [GenericRequestHandler]  got request:
<subject>http://localhost/eca-framework/eca-engine/rules/rule2#query[1]#1140264496108#1</subject>
<eca:variable xmlns:eca="http://rewerse.net/I5/eca-ml" name="OwnCar">
 <eca:query>
  <eca:input-variable name="car-rental-uri" use="doc" />
  <eca:input-variable name="Person" />
  <eca:opaque lang="http://www.w3.org/XQuery">
   for $car in doc($doc)/car-rental/customer-cars/car[@owner=$Person]
   return $car/text()
  </eca:opaque>
 </eca:query>
</eca:variable>
<log:variable-bindings xmlns:log="http://rewerse.net/I5/logic">
 <log:tuple>
  <log:variable name="car-rental-uri">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
  <log:variable name="Person">John Doe</log:variable>
 </log:tuple>
</log:variable-bindings>
```

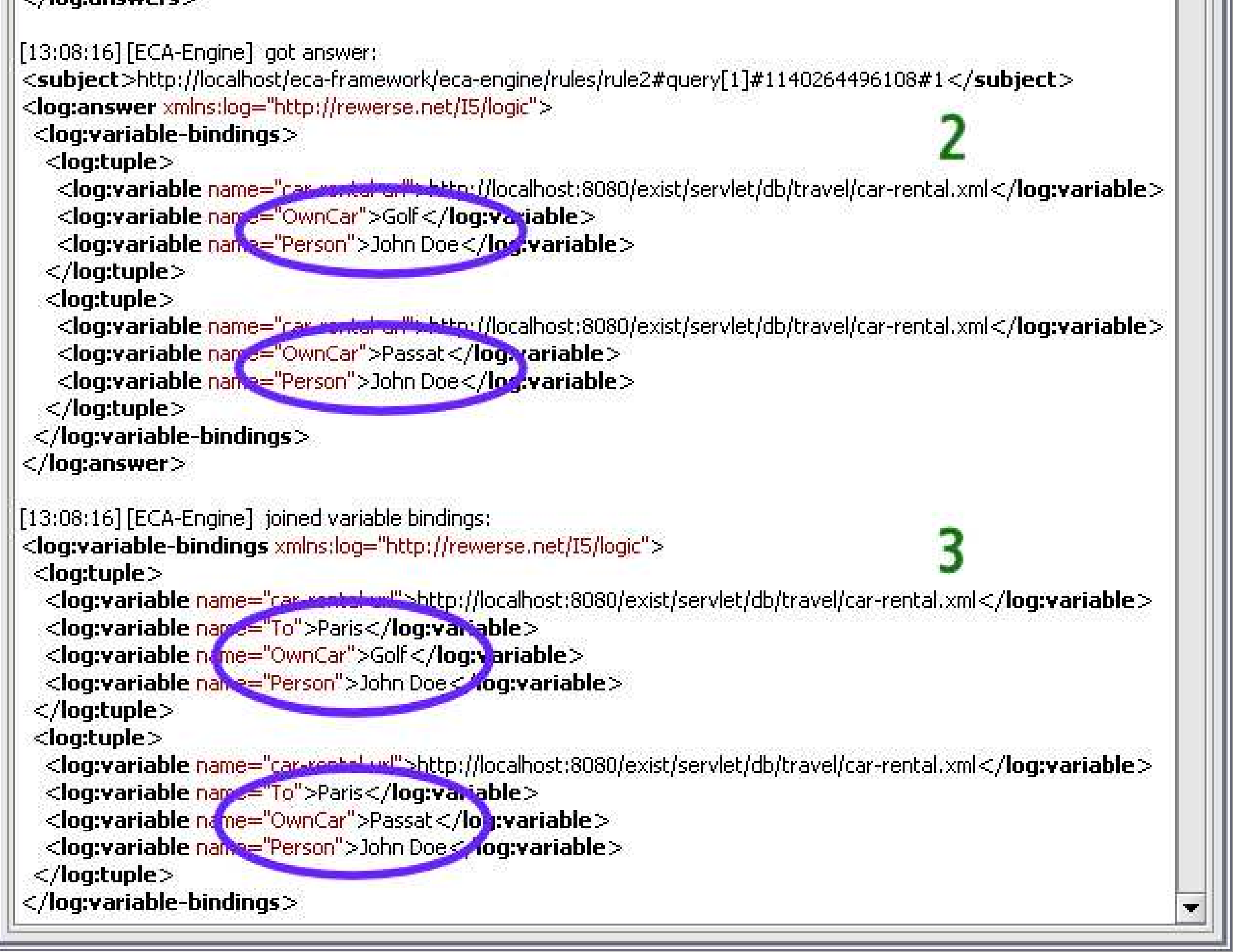
Queries

- query in XML Markup to appropriate processor (yet no engines available)
- common way in current Web environment:
  query in *opaque* program code without markup
  - language identifier (ECA Engine knows processors)
  - URI and HTTP GET/POST or SOAP

Answers

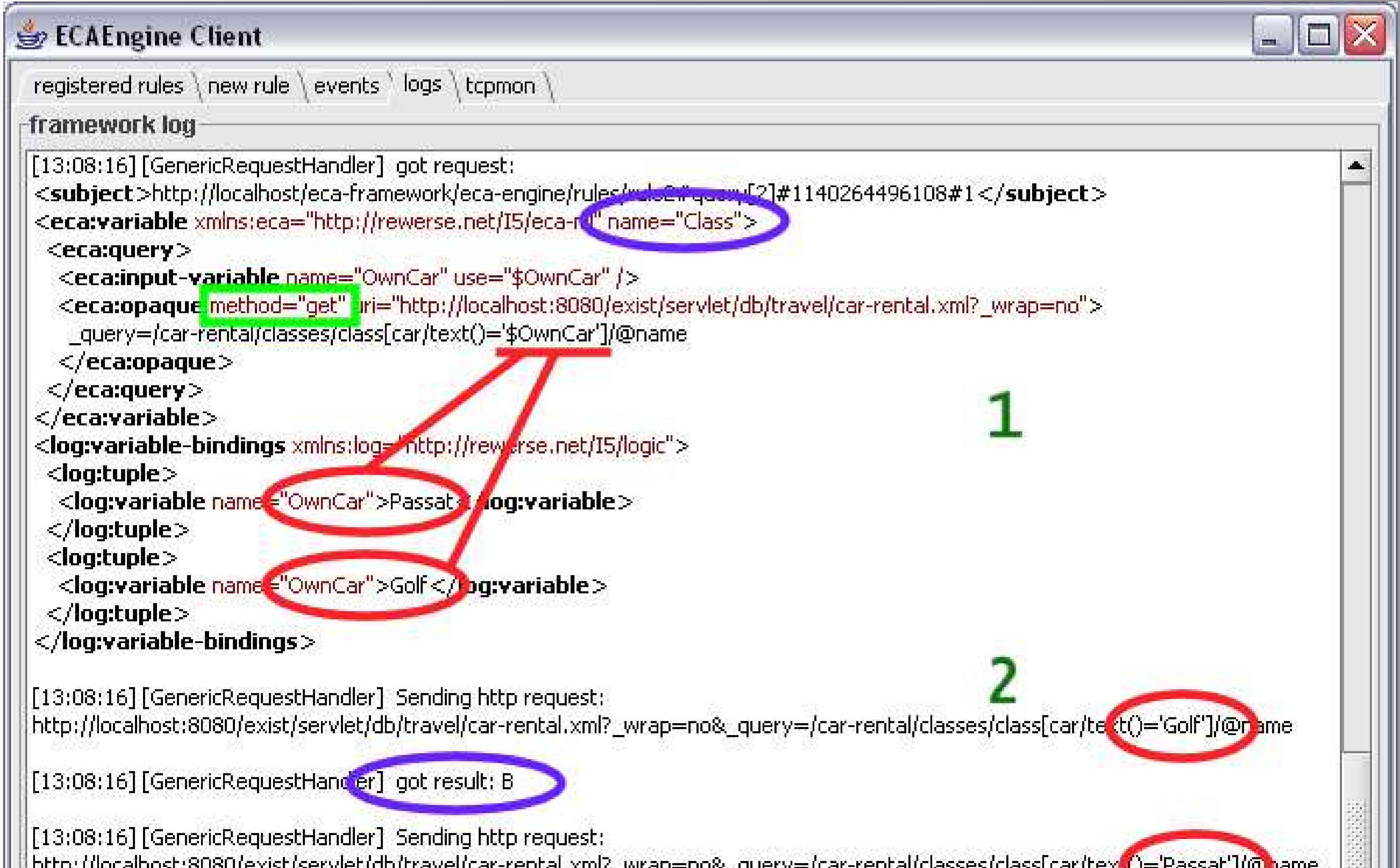


```
[13:08:16] [GenericRequestHandler]  got answer:
<subject>http://localhost/eca-framework/eca-engine/rules/rule2#query[1]#1140264496108#1</subject>
<log:answers xmlns:log="http://rewerse.net/I5/logic">
  <log:answer>
    <log:result>Golf</log:result>
    <log:variable-bindings>
      <log:tuple>
        <log:variable name="doc">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
        <log:variable name="Person">John Doe</log:variable>
      </log:tuple>
    </log:variable-bindings>
  </log:answer>
  <log:answer>
    <log:result>Passat</log:result>
    <log:variable-bindings>
      <log:tuple>
        <log:variable name="doc">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
        <log:variable name="Person">John Doe</log:variable>
      </log:tuple>
    </log:variable-bindings>
  </log:answer>
</log:answers>

[13:08:16] [ECA-Engine]  got answer:
<subject>http://localhost/eca-framework/eca-engine/rules/rule2#query[1]#1140264496108#1</subject>
```

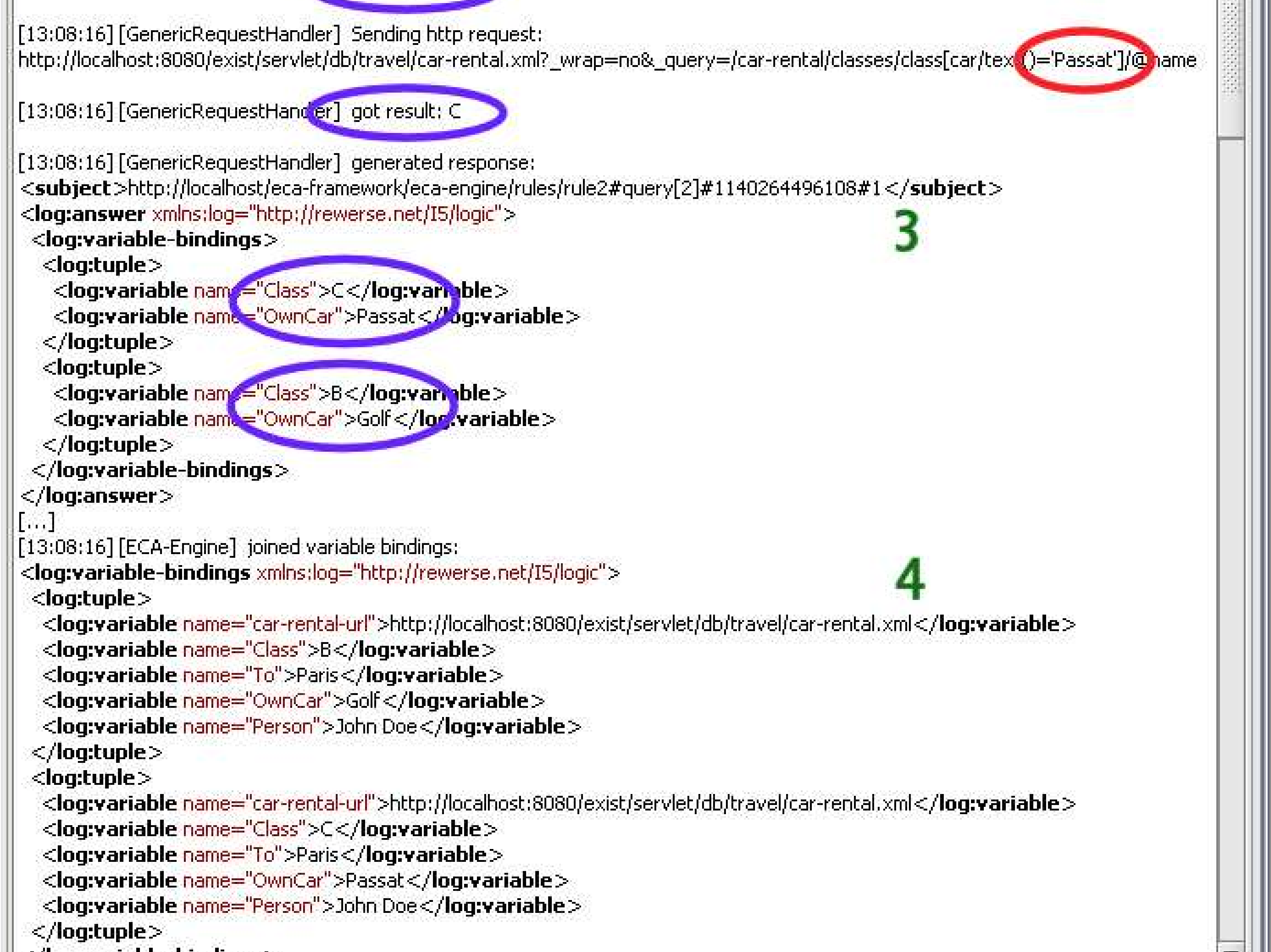


```
[13:08:16] [ECA-Engine]  got answer:
<subject>http://localhost/eca-framework/eca-engine/rules/rule2#query[1]#1140264496108#1</subject>
<log:answer xmlns:log="http://rewerse.net/I5/logic">
 <log:variable-bindings>
  <log:tuple>
   <log:variable name="car-rental-url">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
   <log:variable name="OwnCar">Golf</log:variable>
   <log:variable name="Person">John Doe</log:variable>
  </log:tuple>
  <log:tuple>
   <log:variable name="car-rental-url">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
   <log:variable name="OwnCar">Passat</log:variable>
   <log:variable name="Person">John Doe</log:variable>
  </log:tuple>
 </log:variable-bindings>
</log:answer>

[13:08:16] [ECA-Engine]  joined variable bindings:
<log:variable-bindings xmlns:log="http://rewerse.net/I5/logic">
 <log:tuple>
  <log:variable name="car-rental-url">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
  <log:variable name="To">Paris</log:variable>
  <log:variable name="OwnCar">Golf</log:variable>
  <log:variable name="Person">John Doe</log:variable>
 </log:tuple>
 <log:tuple>
  <log:variable name="car-rental-url">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
  <log:variable name="To">Paris</log:variable>
  <log:variable name="OwnCar">Passat</log:variable>
  <log:variable name="Person">John Doe</log:variable>
 </log:tuple>
</log:variable-bindings>
```

The Next Query



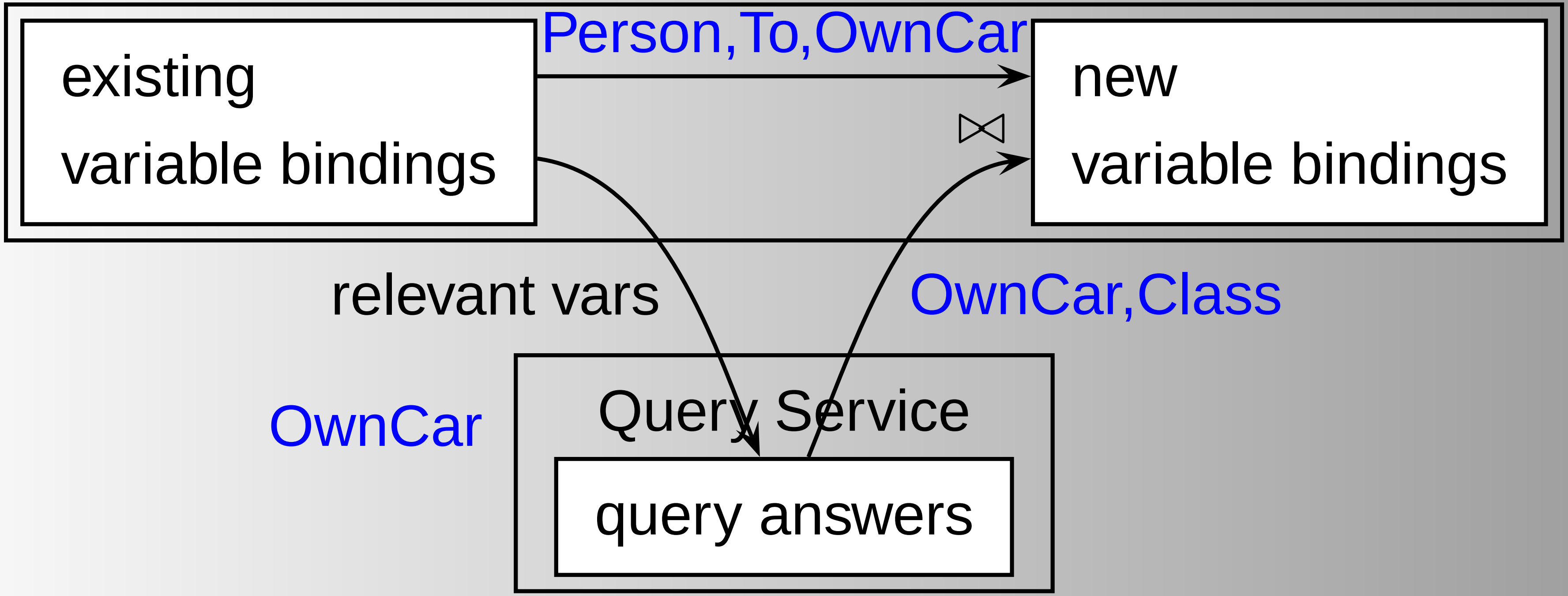
```
[13:08:16] [GenericRequestHandler]  got request:
<subject>http://localhost/eca-framework/eca-engine/rules/rule2#query[2]#1140264496108#1</subject>
<eca:variable xmlns:eca="http://rewerse.net/I5/eca-ml" name="Class">
  <eca:query>
    <eca:input-variable name="OwnCar" use="$OwnCar" />
    <eca:opaque method="get" uri="http://localhost:8080/exist/servlet/db/travel/car-rental.xml?_wrap=no">
     _query=/car-rental/classes/class[car/text()='$OwnCar']/@name
    </eca:opaque>
  </eca:query>
</eca:variable>
<log:variable-bindings xmlns:log="http://rewerse.net/I5/logic">
  <log:tuple>
    <log:variable name="OwnCar">Passat</log:variable>
  </log:tuple>
  <log:tuple>
    <log:variable name="OwnCar">Golf</log:variable>
  </log:tuple>
</log:variable-bindings>

[13:08:16] [GenericRequestHandler]  Sending http request:
http://localhost:8080/exist/servlet/db/travel/car-rental.xml?_wrap=no&_query=/car-rental/classes/class[car/text()='Golf']/@name

[13:08:16] [GenericRequestHandler]  got result: B

[13:08:16] [GenericRequestHandler]  Sending http request:
http://localhost:8080/exist/servlet/db/travel/car-rental.xml?_wrap=no&_query=/car-rental/classes/class[car/text()='Passat']/@name
```



```
[13:08:16] [GenericRequestHandler]  Sending http request:
http://localhost:8080/exist/servlet/db/travel/car-rental.xml?_wrap=no&_query=/car-rental/classes/class[car/text()='Passat']/@name

[13:08:16] [GenericRequestHandler]  got result: C

[13:08:16] [GenericRequestHandler]  generated response:
<subject>http://localhost/eca-framework/eca-engine/rules/rule2#query[2]#1140264496108#1</subject>
<log:answer xmlns:log="http://rewerse.net/I5/logic">
  <log:variable-bindings>
    <log:tuple>
      <log:variable name="Class">C</log:variable>
      <log:variable name="OwnCar">Passat</log:variable>
    </log:tuple>
    <log:tuple>
      <log:variable name="Class">B</log:variable>
      <log:variable name="OwnCar">Golf</log:variable>
    </log:tuple>
  </log:variable-bindings>
</log:answer>
[...]
[13:08:16] [ECA-Engine]  joined variable bindings:
<log:variable-bindings xmlns:log="http://rewerse.net/I5/logic">
  <log:tuple>
    <log:variable name="car-rental-url">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
    <log:variable name="Class">B</log:variable>
    <log:variable name="To">Paris</log:variable>
    <log:variable name="OwnCar">Golf</log:variable>
    <log:variable name="Person">John Doe</log:variable>
  </log:tuple>
  <log:tuple>
    <log:variable name="car-rental-url">http://localhost:8080/exist/servlet/db/travel/car-rental.xml</log:variable>
    <log:variable name="Class">C</log:variable>
    <log:variable name="To">Paris</log:variable>
    <log:variable name="OwnCar">Passat</log:variable>
    <log:variable name="Person">John Doe</log:variable>
  </log:tuple>
```

Language Heterogeneity

- Modules for arbitrary query (and event or action) languages
- Support for Sideways Information Passing Strategy



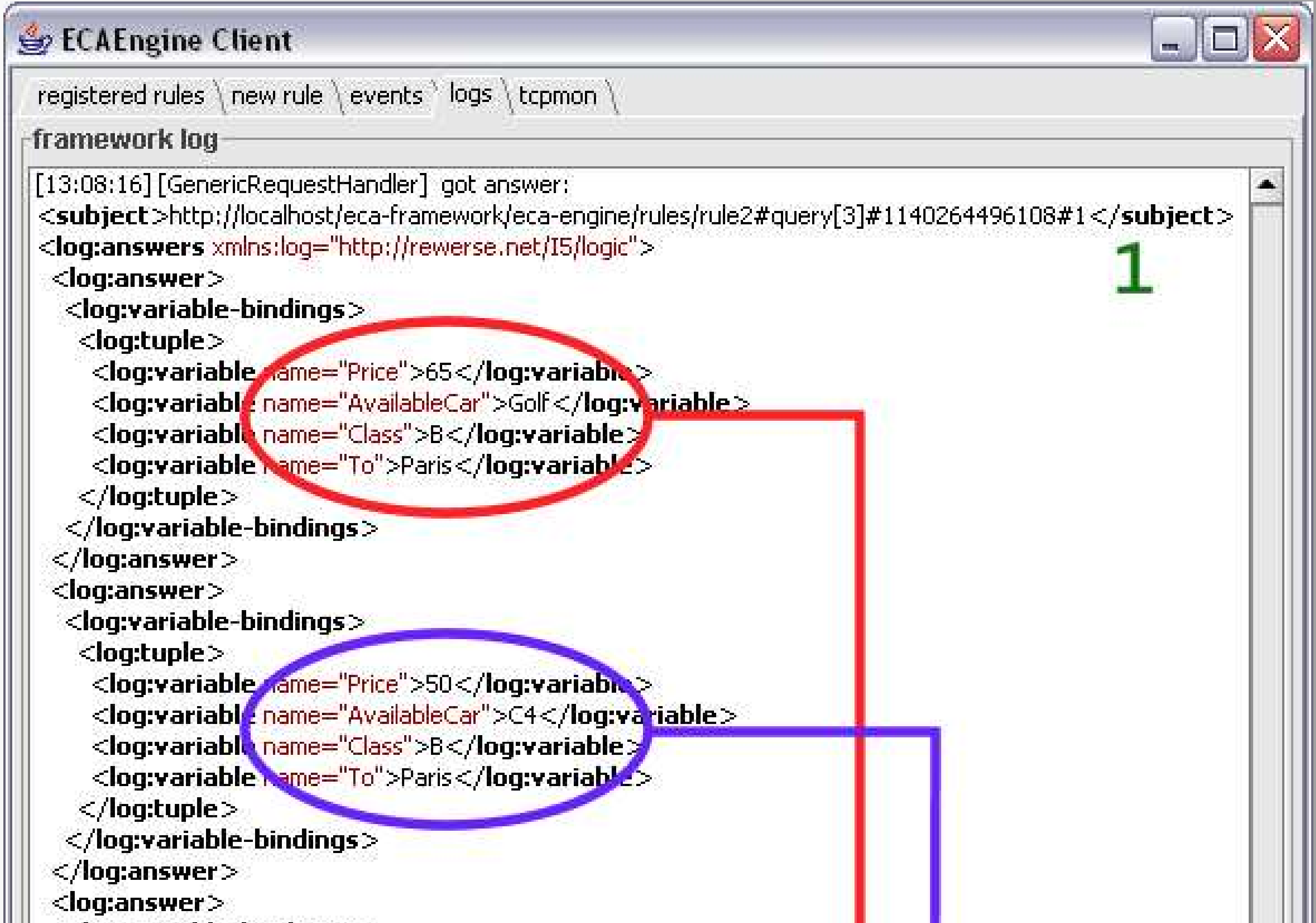
- Support for most simple HTTP GET services
  one tuple at a time, textual replacement of variable values
- (RDF) Service Descriptions

Next Query: Available Cars in Paris

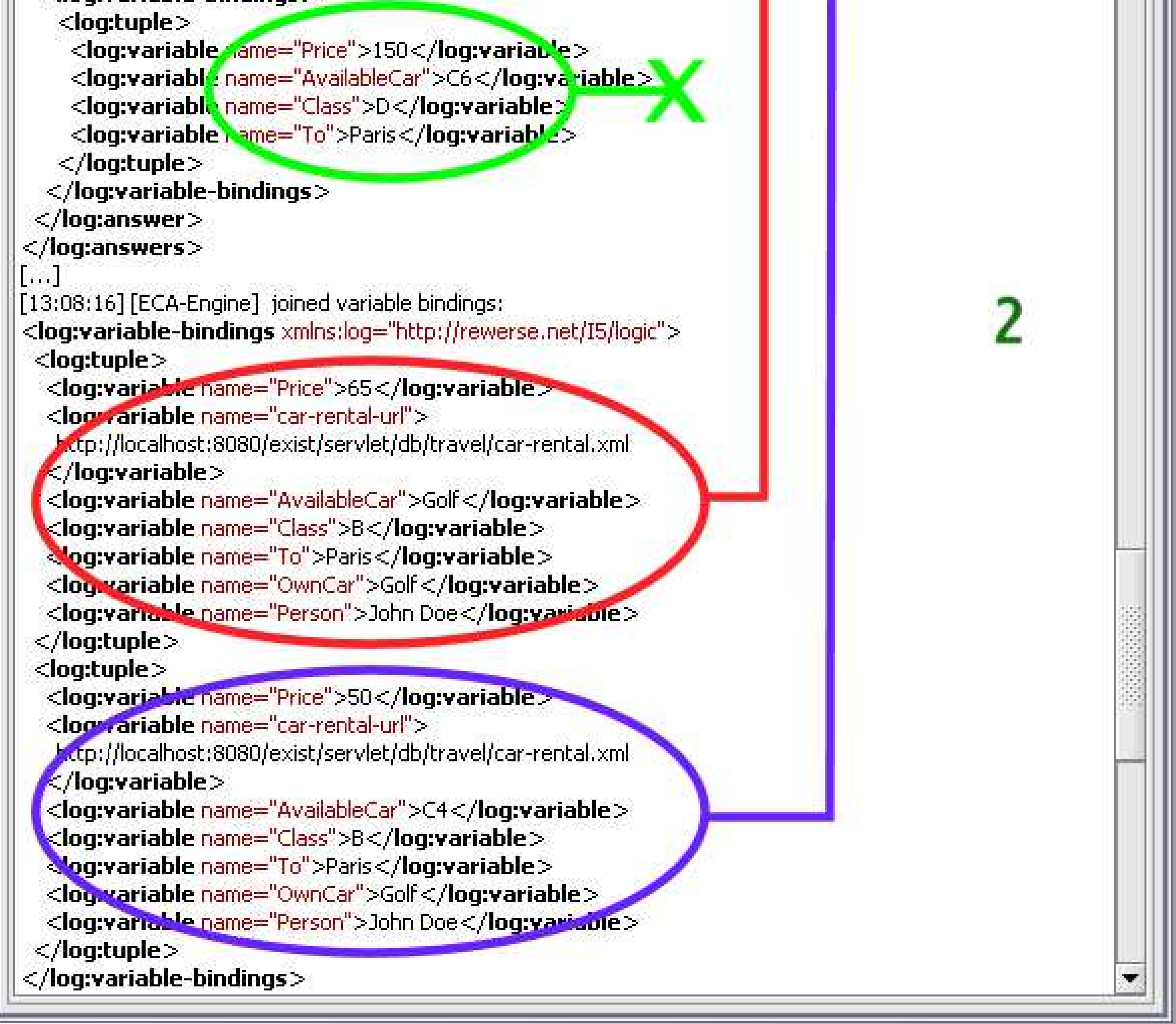
Assume an SQL wrapper.

```
<eca:query>
  <eca:input-variable name="$To"/>
  <eca:opaque uri="…/sql-interface/">
    SELECT
    Offers.Price AS Price,
    Offers.Type AS AvailableCar,
    Classes.Class,
    $To,
    FROM Offers, Classes
    WHERE city = $To AND Offer.Type = Classes.Type
  </eca:opaque>
</eca:query>
```

Next Answer: Available Cars in Paris



```
[13:08:16] [GenericRequestHandler] got answer:
<subject>http://localhost/eca-framework/eca-engine/rules/rule2#query[3]#1140264496108#1</subject>
<log:answers xmlns:log="http://rewerse.net/I5/logic">
  <log:answer>
    <log:variable-bindings>
      <log:tuple>
        <log:variable name="Price">65</log:variable>
        <log:variable name="AvailableCar">Golf</log:variable>
        <log:variable name="Class">B</log:variable>
        <log:variable name="To">Paris</log:variable>
      </log:tuple>
    </log:variable-bindings>
  </log:answer>
  <log:answer>
    <log:variable-bindings>
      <log:tuple>
        <log:variable name="Price">50</log:variable>
        <log:variable name="AvailableCar">C4</log:variable>
        <log:variable name="Class">B</log:variable>
        <log:variable name="To">Paris</log:variable>
      </log:tuple>
    </log:variable-bindings>
  </log:answer>
  <log:answer>
```



```
    <log:tuple>
      <log:variable name="Price">150</log:variable>
      <log:variable name="AvailableCar">C6</log:variable>
      <log:variable name="Class">D</log:variable>
      <log:variable name="To">Paris</log:variable>
    </log:tuple>
  </log:variable-bindings>
 </log:answer>
</log:answers>
[...]
[13:08:16] [ECA-Engine]  joined variable bindings:
<log:variable-bindings xmlns:log="http://rewerse.net/I5/logic">
 <log:tuple>
  <log:variable name="Price">65</log:variable>
  <log:variable name="car-rental-url">
   http://localhost:8080/exist/servlet/db/travel/car-rental.xml
  </log:variable>
  <log:variable name="AvailableCar">Golf</log:variable>
  <log:variable name="Class">B</log:variable>
  <log:variable name="To">Paris</log:variable>
  <log:variable name="OwnCar">Golf</log:variable>
  <log:variable name="Person">John Doe</log:variable>
 </log:tuple>
 <log:tuple>
  <log:variable name="Price">50</log:variable>
  <log:variable name="car-rental-url">
   http://localhost:8080/exist/servlet/db/travel/car-rental.xml
  </log:variable>
  <log:variable name="AvailableCar">C4</log:variable>
  <log:variable name="Class">B</log:variable>
  <log:variable name="To">Paris</log:variable>
  <log:variable name="OwnCar">Golf</log:variable>
  <log:variable name="Person">John Doe</log:variable>
 </log:tuple>
</log:variable-bindings>
```

Further Steps

- Query: query DB how much money $Person is willing to spend per day (extending join with $MaxPrice)
- Test:

```
<eca:test>
  <eca:input-variable name="Price" />
  <eca:input-variable name="MaxPrice" />
  <eca:opaque lang="http://www.w3.org/XPath">
    number($Price) < number($MaxPrice)
  </eca:opaque>
</eca:test>
```

  restricts set of variable bindings.

- action ...

Summary

- Prototype for language-independent ECA Engine
- based on exchanging and operating with variable bindings
- GRH as interface to component services

Further Work

- fix syntax of communication details
- Applications:
  - atomic events + opaque queries:
    integrating queries against different data sources in scientific workflows (Bioinformatics in Lisbon for REWERSE A2)
- framework-aware component modules under implementation
  - event algebras
  - action languages
  - RDF domain node infrastructure (Jena)

Thank You

Questions ??