Deep Web Queries in a Semantic Web Environment

Thomas Hornung (Universität Freiburg)

Wolfgang May (Universität Göttingen)

Motivation

- Machine-*accessible* vs. machine-*understandable* or what this talk is **not** about:
  - Deep Web Navigation (ADW `08),
  - Web Data Extraction and Labeling (ViPER - CIKM `05)
- *Goal:* Use Deep Web sources in a Semantic Web framework
- *Problem:* Deep Web sources have a primitive data model (only strings)
- *Proposed solution:* Semantic Annotation of Deep Web Sources

The MARS Framework (1/2)

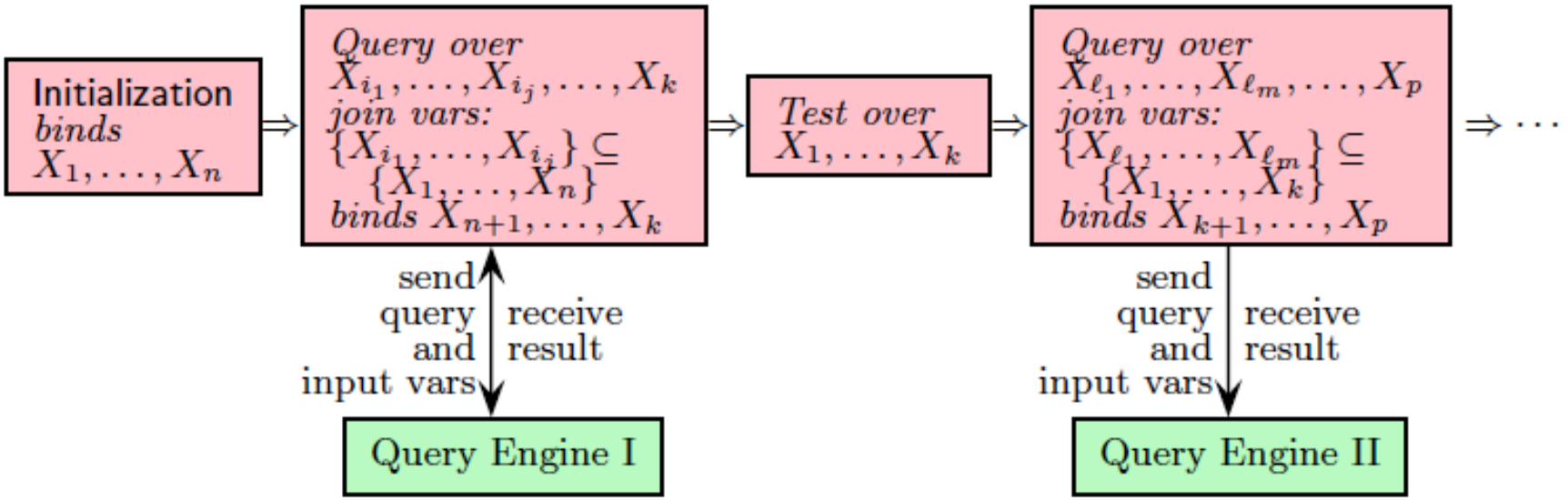
(Modular Active Rules for the Semantic Web)

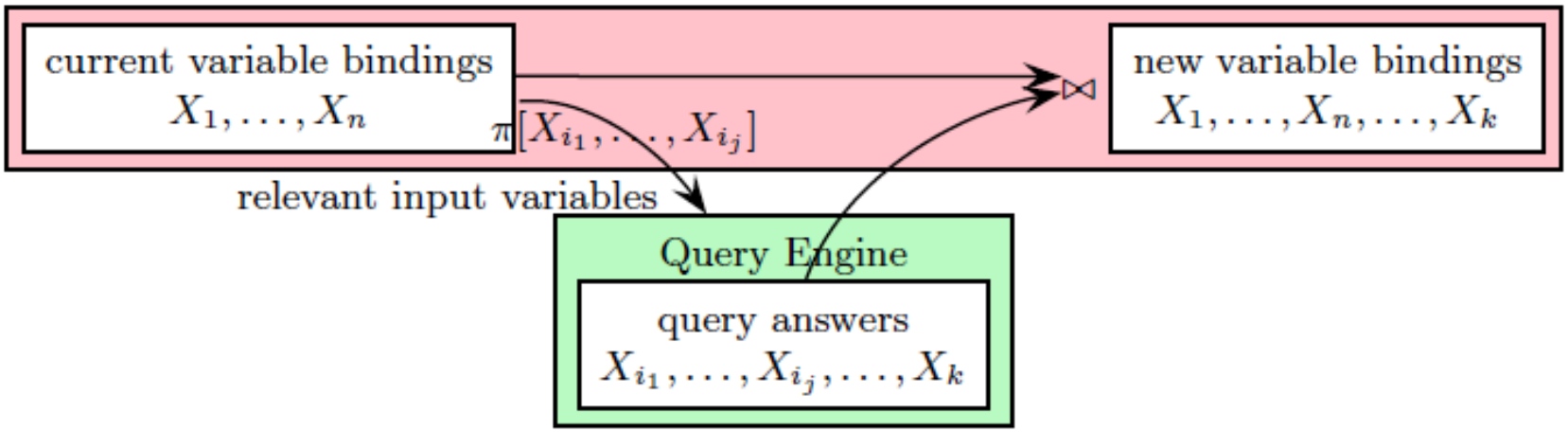
- Provides an open framework for ECA rules and for processes
- Supports *heterogeneous* event, query, and action languages:
  - Deep Web Query Language (DWQL)
  - RelCCS: CCS with relational dataflow (CAiSE '09)
- *Data model*: Set of tuples of variable bindings, i.e., every tuple is of the form $t = \{x_1/v_1, \ldots, x_n/v_n\}$
- DWQL example:

```
<dwql:Query xmlns:dwql="http://www.semwebtech.org/languages/2008/dwql#">
  <dwql:view dwql:resource="identifying URI of the DWQL view" />
  <dwql:inputVariable name="x"    ... further annotations ... />
  <dwql:outputVariable name="y"    ... further annotations ... />
  further specification in DWQL markup as element content
</dwql:Query>
```

The MARS Framework (2/2)

(Modular Active Rules for the Semantic Web)





Semantic Annotation of Deep Web Sources

Deep Web sources revisited (1/2)

(1) start, dest, date, desiredDeptTime →

(2) start, dest, date, desiredArrTime →

deptTime, arrTime, duration, price ←

Deep Web sources revisited (2/2)

(1) *(deptTime, arrTime, duration, price)* ←
germanRailwaysByDept(start, dest, date, desiredDeptTime)

(2) *(deptTime, arrTime, duration, price)* ←
germanRailwaysByArr(start, dest, date, desiredArrTime)

Example:

germanRailwaysByDept(
(start/"Freiburg", dest/"Göttingen", date/"03.02.2009", time/"08:00")) =
{ (deptTime/"08:57", arrTime/"13:07", duration/"4:10", price/"95.00"),
(deptTime/"09:03", arrTime/"14:48", duration/"5:45", price/"85.00"), ...}

Literals, Measurements, Dimensions, and Units

- *Schema*:
  - XML datatypes, such as xsd:date, xsd:time and xsd:dateTime
  - Syntactical representation, e.g. "dd.MM.yyyy"
- *Semantics*:
  - Physical vs. non-physical dimensions of properties, e.g. 100 km (distance) vs. 250€ (price)
    → (Value, Unit) pairs
  - Problem: Units may differ between automonous sources (e.g., miles vs. kilometers, or $ vs. €)
- *DeepWeb sources and variables in the query workflow are annotated with respect to the MARS annotation ontology*

MARS Annotation Ontology (Excerpt)

```
@prefix : <http://www.semwebtech.org/mars#> .
@prefix dim: <http://www.semwebtech.org/mars/dimensions#> .
@prefix unit: <http://www.semwebtech.org/mars/units#> .
@prefix curr: <http://www.semwebtech.org/mars/currencies#> .
dim:Length a :Dimension;
  :hasUnits unit:meter, unit:kilometer, unit:mile, ... .
dim:Price a :Dimension;
  :hasUnits curr:USD, curr:EUR, curr:PLN, ... .
  owl:equivalentClass
  [ a owl:Restriction; owl:onProperty :hasUnits;
    owl:allValuesFrom :Currency] .
[ a :FixedConversion;
  :from unit:kilometer; :to unit:mile;  :factor 1609.3 ] .
[ a :DynamicConversion; :from curr:EUR; :to curr:USD ] .
[ a :DynamicConversion; :from curr:EUR; :to curr:PLN ] .
```

Semantic Annotation of Railway Example (1/2)

(1) *(deptTime, arrTime, duration, price) ←*
germanRailwaysByDept(start, dest, date, desiredDeptTime)

(2) *(deptTime, arrTime, duration, price) ←*
germanRailwaysByArr(start, dest, date, desiredArrTime)

```
<bla://dwql-views/travel/germanRailways> a :DeepWebSource;
 :baseURL <http://www.bahn.de>;
 :providesView <bla://dwql-views/travel/germanRailwaysByDept>,
               <bla://dwql-views/travel/germanRailwaysByArr>;
```

(1)
```
<bla://dwql-views/travel/germanRailwaysByDept> a :DeepWebView;
  :hasInputVariable _:start, _:dest, _:dDepT, _:date;
  :hasOutputVariable _:deptT, _:arrT, _:dur, _:price.
```

(2)
```
<bla://dwql-views/travel/germanRailwaysByArr> a :DeepWebView;
  :hasInputVariable _:start, _:dest, _:dArrT, _:date;
  :hasOutputVariable _:deptT, _:arrT, _:dur, _:price.
```

Semantic Annotation of Railway Example (2/2)

```
_:start a :Tag; :name "start"; :datatype xsd:string; :denotes travel:City.
```

```
_:price a :Tag; :name "price"; :datatype xsd:decimal;
                               :dimension dim:price; :unit curr:EUR.
```

Tag	Datatype	Format	Unit
start, dest	xsd:string	–	–
deptTime, arrTime, desiredDeptTime, desiredArrTime	xsd:time	"HH:mm"	(internal)
duration	xsd:time	"HH:mm"	(internal)
date	xsd:date	"dd.MM.yyyy"	(internal)
price	xsd:decimal		curr:EUR

Java Simple Date Format

Deriving domains/units

```
select ?U
where {  ?S :providesView <bla://dwql-views/travel/germanRailwaysByDept> .
         ?S :hasTag [ :name "price"; :unit ?U ] }
```

→ *<http://www.semwebtech.org/mars/currencies#EUR>*

Such queries are used when the domains/units of variables of a process that contains a DWQL query are derived

Use Case:
Traveling to Poznan

Visiting ADW ´09

- *Goal:* Find an itinerary from Freiburg, Germany to Poznan, Poland
- *Problem:* Online railway portals only return pricing information for national travels
- *Proposed solution:*
  1. Use http://www.bahn.de for finding different routes from Freiburg to border towns*
  2. Use http://pkp.pl for finding different routes from each border town to the destination Poznan
  3. Return the cheapest/fastest connection

  * For a more generic solution an additional geo service for locating border towns could be used

Travel Query Workflow

```
<ccs:Sequence>
  assume variables start, startC, dest, destC, date, and time bound to initial values
  <ccsns:Query>
    binds variable borderStation by query hasBorderStation(startC, destC, borderStation)
  </ccsns:Query>
  <ccs:Query>
    <dwql:Query xmlns:dwql="http://.../languages/2008/dwql#">
      <dwql:view dwql:resource="bla://dwql-views/travel/germanRailwaysByDept" />
      ...
      <dwql:outputVariable dwql:name="P1" dwql:use="price" />
    </dwql:Query>
  </ccs:Query>
  ...
  <ccs:Alternative>
    <ccs:Sequence>
      <ccs:Test> <ccs:Equals ccs:variable="destC" ccs:withValue="PL" /></ccs:Test>
      <ccs:Query>
        <dwql:Query xmlns:dwql="http://.../languages/2008/dwql#">
          ...
          <dwql:outputVariable dwql:name="P2" dwql:use="price" />
        </dwql:Query>
      </ccs:Query>
      calculate Price := P1 + P2
    </ccs:Sequence>
    similar <ccs:Sequence> specifications for other destination countries
  </ccs:Alternative>
</ccs:Sequence>
```

Reasoning about Process Variables

```
...
_:startC a :Variable; :name "start"; ## ... derived from the first query
_:dest   a :Variable; :name "dest"; :datatype xsd:string;
         :denotes travel:City.
_:destC  a :Variable; :name "start"; ## ... derived from the first query
_:date   a :Variable; :name "date"; :datatype xsd:date;
         :format "dd.MM.yyyy".
_:border a :Variable; :name "borderStation"; :datatype xsd:string;
                                              :denotes travel:City.
_:time   a :Variable; :name "time"; :datatype xsd:time; :format "HH:mm".
_:arrBT  a :Variable; :name "arrBorderTime"; :datatype xsd:time;
         :format "HH:mm".
_:arrT a :Variable; :name "arrTime"; :datatype xsd:time; :format "HH:mm".
_:p1  a :Variable; :name "P1"; :datatype xsd:decimal;
                               :dimension dim:price; :unit curr:EUR.
_:p2  a :Variable; :name "P2"; :datatype xsd:decimal;
                               :dimension dim:price; :unit curr:PLN.
_:pr  a :Variable; :name "price"; :datatype xsd:decimal;
                               :dimension dim:price; :unit curr:EUR.
```

The TravelWorkflow Annotations are derived completely from the process structure and the DWQL Source Descriptions

Example: Price Calculation

- Freiburg → Frankfurt (Oder)

  P1: 127,00 €
- Frankfurt (Oder) → Poznan

  P2: 22 PLN
- Price Calculation:

1. Convert P2 to €: 4,87 €
2. *Price* = P1 + P2 = 127,00 € + 4,87 € = <u>131,87 €</u>

→ *Price (Any) conversions can now be handled transparently by the MARS Framework*

Discussion

Current Focus

- General Travel Planning application:
  - Consider trains, planes, etc.
  - Combine Web Services, Deep Web Sources, etc.
  - Optimize search heuristics, i.e. follow top-k best candidate routes first
  - Divide task into a graph management and a workflow problem
- Not mentioned in this talk:
  - Value tolerance
  - Range restriction

Conclusion

- Annotating Deep Web sources is *mandatory* for combining them in a *non-trivial* way
- Separation of general concerns vs. source-specific concerns:
  - Vocabulary (tags) vs. different units, formats, etc.
- Support for (automatic) handling of Literals, Measurements, Dimensions, and Units:
  - Annotate once, the MARS Framework assures interoperability
- Prototype available at:

  http://www.semwebtech.org/mars/frontend/

Questions/Comments

References

- Oliver Fritzen, Wolfgang May, Franz Schenk: *Markup and Component Interoperability for Active Rules.* RR 2008
- T. Hornung, W. May, and G. Lausen. *Process Algebra-based Query Workflows.* CAiSE 2009. to appear.
- Thomas Hornung, Kai Simon, Georg Lausen: *Mashups over the Deep Web*. In LNBIP, Volume 18, WebIST 2008 Revised Selected Papers, 2009
- Yang Wang, Thomas Hornung: *Deep Web Navigation by Example*. ADW 2008
- Kai Simon, Thomas Hornung, Georg Lausen: *Learning Rules to Pre-process Web Data for Automatic Integration*. RuleML 2006
- Kai Simon, Georg Lausen: *ViPER: Augmenting Automatic Information Extraction with Visual Perceptions*. CIKM 2005