Semantic Annotations and Querying of Web Data Sources

Thomas Hornung[1] and *Wolfgang May*[2]

[1] Institut für Informatik, Universität Freiburg, Germany

hornungt@informatik.uni-freiburg.de

[2] Institut für Informatik, Universität Göttingen, Germany

may@informatik.uni-goettingen.de

CoopIS 2009, Vilamoura, Portugal
November 4, 2009

Context of this work/Situation

- Many everyday's questions/problems can be answered by information that is available on the Web
  - what are the cheapest flights (by any airline) to a place near Vilamoura on October 31st or November 1st?
  - How to finally reach Vilamoura?
    How far ist the bus stop then from the hotel?
- Stefano Ceri, *Optimization of Multi-Domain Queries on the Web*, VLDB 2008:

  > *... at the current state-of-the-art, the above queries can be answered only by patient and expert users, whose strategy is to interact with specialized services, one at a time, and then feed the result of one search as input to another, reconstructing answers in their mind.*

Central Problem(s)

The Web is computer-based, but that does not mean that computers can yet help much for *using* it:

- actual data: (Deep) Web sources:
  - Deep Web: forms, wrappable into n-ary predicates $p(x_1, \ldots, x_n)$
  - sometimes databases, XML,
  - no common schema/ontology
  - restricted query capabilities (input $\rightarrow$ output)

Related Work

- wrapping of Deep Web sources to predicates:
  tools available [Lixto 2001-now]
- data integration/answering queries using views
  - Information Manifold/MiniCon [Halevy-VLDB-96,VLDBJ-01],
    Peer-to-Peer mappings [Piazza-01/02],
    LAV/GAV [Lenzerini-PODS-00]

  based on rule/view rewriting/syntax.
- dealing with access limitations (for given queries)
  - evaluation ordering under access limitations
    Algorithms: e.g. [Yang-Kifer-Chaudhri-PODS-06],
    [Cali-Martinenghi-ICDE-08]
  - using abstract domains for obtaining values [Cali-Martinenghi-ER-08]
  - optimization, e.g., [Braga-Ceri-Daniel-Martinenghi-VLDB-08]

Goal: Apply Semantic Technologies

provides a common data+metadata model: RDF/RDFS/OWL

- external, virtual schema: domain ontology given in RDF/RDFS/OWL
- answer (SPARQL) queries
- by using the actual (Deep) Web sources
- based on semantical and functional source descriptions
  - relate the (wrapped) data sources with the global ontology
  - themselves in RDF
- for finding appropriate overlappings (including reasoning).

Example: Searching for Flight Connections

"How to reach Vilamoura from Göttingen"

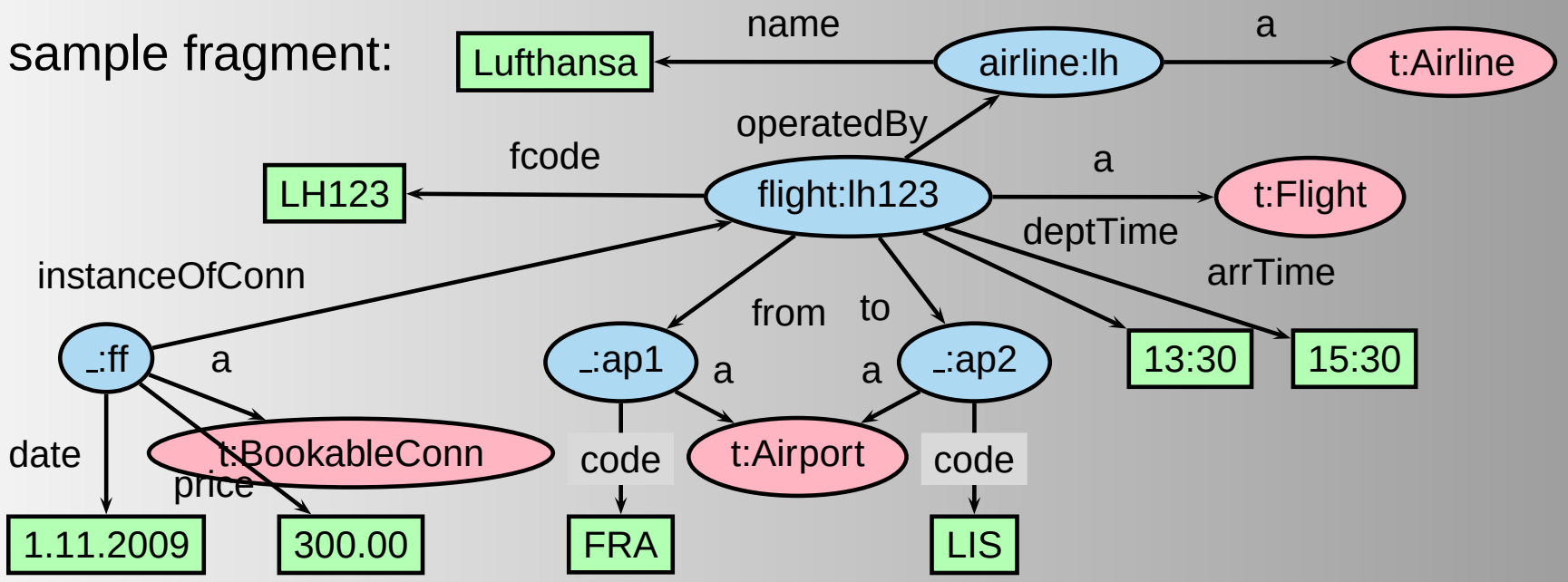
- set of potential departure airports: FRA, HAJ, LEJ
- geographical coordinates of the destination: 37°4' 40" N, 8° 6' 55" W
- probably some idea of nearby airports: LIS, FAO, SVQ
- manual evaluation of combinations takes some time ...
  experience: sometimes unexpected results are even better

Query/Workflow (simplified, flight issues only)

- check for all airports y reachable from $x \in \{\text{FRA, HAJ, LEJ}\}$
- compute distance from y to Vilamoura, take all $< 400km$
- ask for flight connections (x, y)
- output results, estimate/check manually for local transport

Example (flights part only)

RDF sample fragment:



Actual Sources

- (A) Flight portals:
  (dept.airport, dest.airport, date) → (flight code, dept.time, arr.time, price)
- (C) Sources like http://www.theairdb.com/:
  (dept.airport) → (dest.airport)
- sources (B) and (D) see proceedings
- relational paradigm: *sets* of tuples of inputs/answers/results

WDSDL: Annotation of Web Data Sources

Web Data Source Description Language

- technical level: how to address the source (or its wrapper).
  URI, (manually programmed) wrapper
- signature level: signature of views, specifying their input and output parameters.
  naming of variables
  source may offer several views with different i/o characteristics
- semantical level: relates the query services with the underlying domain terminology.
  relates variables with concepts of the application domain

Signature Level

- Web data source as *characteristic predicate*: $q(\overline{x}) = q(x_1, \ldots, x_n)$
- one or more views provided via restricted access patterns:
  input variables $\overline{qin} = \{xin_1, \ldots, xin_k\} \subseteq \{x_1, \ldots, x_n\}$,
  output variables $\overline{qout} = \{xout_1, \ldots, xout_m\} \subseteq \overline{x} \setminus \overline{qin}$
  signature of the view denoted by $\overline{qout} \leftarrow v(\overline{qin})$.
- set-oriented interface provided via wrappers
  - Lixto (TU Vienna): Deep Web Wrapper Generation and Execution Tool
  - WSQL: Web Service Query Language (via Java)
  - input r: sets of tuples over $\overline{qin}$
  - output: $\pi[\overline{qout} \cup \overline{qin}](q \rhd\!\lhd r)$

The annotations start on the level of $\overline{qout} \leftarrow v(\overline{qin})$ and annotate the wrapped source.

Signature Level

- *naming/tagging* the variables of the characteristic predicate

(A) connection(from, to, date, airline, fcode, deptTime, arrTime, price). (8 arguments)

view: (airline, fcode, deptTime, arrTime, price) ⟵ travenjoyConns(from, to, date).

(C) flies(carrier, origin, dest).

view: (carrier, dest) ⟵ flies(origin).

- human-readable since intuitive names are used,
- not machine-usable since tags are not (yet) formally associated to the domain notions
  (and do not match between different sources)
- note: predicates do usually not simply represent properties of a single object or relationship
  (e.g. (A): codes of two airports that are related by a flight (with departure and arrival time) carried out by an airline, where a booking is available on the given date – 5 entities)

Signature Level in RDF

- straightforward ontology (see paper):

  View – hasTag – Tag – hasName

  hasInputVariable ⊆ ⊇ hasOutputVariable

- annotations with dimensions (“time”, “price”, etc.), formats (e.g, “HH:mm”), and units (meters, miles, $, €) [Hornung-May-ADW-09]
  - conversion of distances, currencies etc.

Semantical Level: Correlating to the Domain Ontology

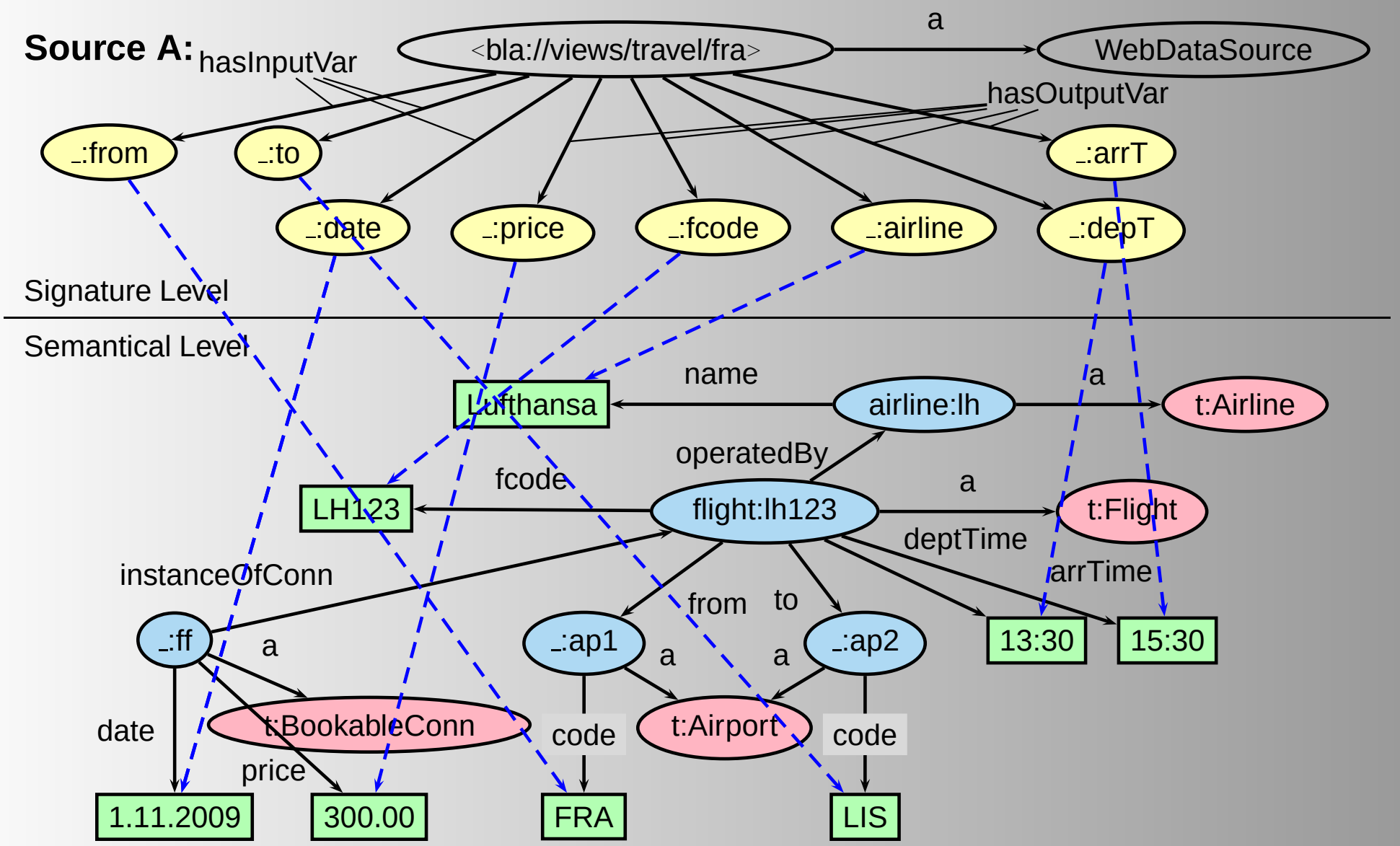
- Local-as-View-Mapping to RDF Ontology

Source A:

Literals (8) Things (5)

$$(*)\quad \text{flight}_A(a,b,c,d,e,f,g,h) \Leftrightarrow \exists v,w,x,y,z:$$
$$\text{Flight}(v) \wedge \text{flightCode}(v,e) \wedge \text{Airline}(w) \wedge \text{operatedBy}(v,w) \wedge \text{name}(w,d) \wedge$$
$$\text{Airport}(x) \wedge \text{from}(v,x) \wedge \text{code}(x,a) \wedge$$
$$\text{Airport}(y) \wedge \text{to}(v,y) \wedge \text{code}(y,b) \wedge$$
$$\text{deptTime}(v,f) \wedge \text{arrTime}(v,g) \wedge$$
$$\text{BookableConn}(z) \wedge \text{instanceOfConn}(z,v) \wedge \text{date}(z,c) \wedge \text{price}(z,h).$$

Semantical Level: Correlating to the Domain Ontology



Technical Issue: Representation of the Mapping in RD

Temptation: Canonical Instance as Prototype Triples

```
prefix t : <http://www.semwebtech.org/domains/2006/travel#>.
_:flightR a t:Flight; t:operatedBy [ a t:Airline; t:name "airlineTag" ];
  t:from [a t:Airport; t:code "fromTag" ];  t:to [a t:Airport; t:code "toTag" ] ;
  t:flightCode "fcodeTag" ; t:deptTime "depTTag" ; t:arrTime "arrTTag" .
_:bconnR a t:BookableConn; t:instanceOfConn _:flightR ; t:date "dateTag";
  t:price "priceTag".
```

Problem

- implicitly *contributes* its triples to the "world-wide RDF database"
- applications (e.g. search) cannot distinguish these triples from "real" triples:
  blank node local0815#id4711 is an airline that has a travel:name property with value "airlineTag"
  (note: using the tag's resource bla://views/sourceA/tags/airlineTag from the signature level is even worse since travel:name must be literal-valued)

WDSDL: Source Annotation Statements

- set of statements that *refer* to the tags and also refer to blank nodes acting as existential variables,
- use mechanism of *reification*: talk *about* statements by wdsdl:AnnotationStatements:
  - same strategy as in OWL-2 for asserting that a statement does not hold:
  - tags,
  - local variables,
  - constants if required,
  - the notions of the application domain.
- optional additional constraints (see paper)

Example: WDSDL Source Annotation

<bla://views/travel/travenjoy/> wdsdl:hasAnnotation
[wdsdl:localVar _:flightV, _:airlineV, _:airp1V, _:airp2V, _:bconnV ;
wdsdl:hasAnnotationStatement
[rdf:subject _:flightV; rdf:predicate rdf:type; rdf:object travel:Flight],
[rdf:subject _:flightV; rdf:predicate travel:operatedBy; rdf:object _:airlineV],
[rdf:subject _:airlineV; rdf:predicate rdf:type; rdf:object travel:Airline],
[rdf:subject _:airlineV; rdf:predicate travel:name; rdf:object _:airline],
[rdf:subject _:flightV; rdf:predicate travel:from; rdf:object _:airp1V],
[rdf:subject _:flightV; rdf:predicate travel:to; rdf:object _:airp2V],
[rdf:subject _:flightV; rdf:predicate travel:flightCode; rdf:object _:code],
[rdf:subject _:airp1V; rdf:predicate rdf:type:code; rdf:object travel:Airport],
[rdf:subject _:airp1V; rdf:predicate travel:code; rdf:object _:from],
[rdf:subject _:airp2V; rdf:predicate rdf:type:code; rdf:object travel:Airport],
[rdf:subject _:airp2V; rdf:predicate travel:code; rdf:object _:to], ... times ...,
[rdf:subject _:bconnV; rdf:predicate rdf:type; rdf:object travel:BookableConn],
[rdf:subject _:bconnV; rdf:predicate travel:instanceOfConn; rdf:object _:flightV],
[rdf:subject _:bconnV; rdf:predicate travel:date; rdf:object _:date, ... price ...]].

Source Selection and SPARQL Query Answering

- SPARQL: graph pattern (matching a fragment of the ontology graph),
- answers: match against data

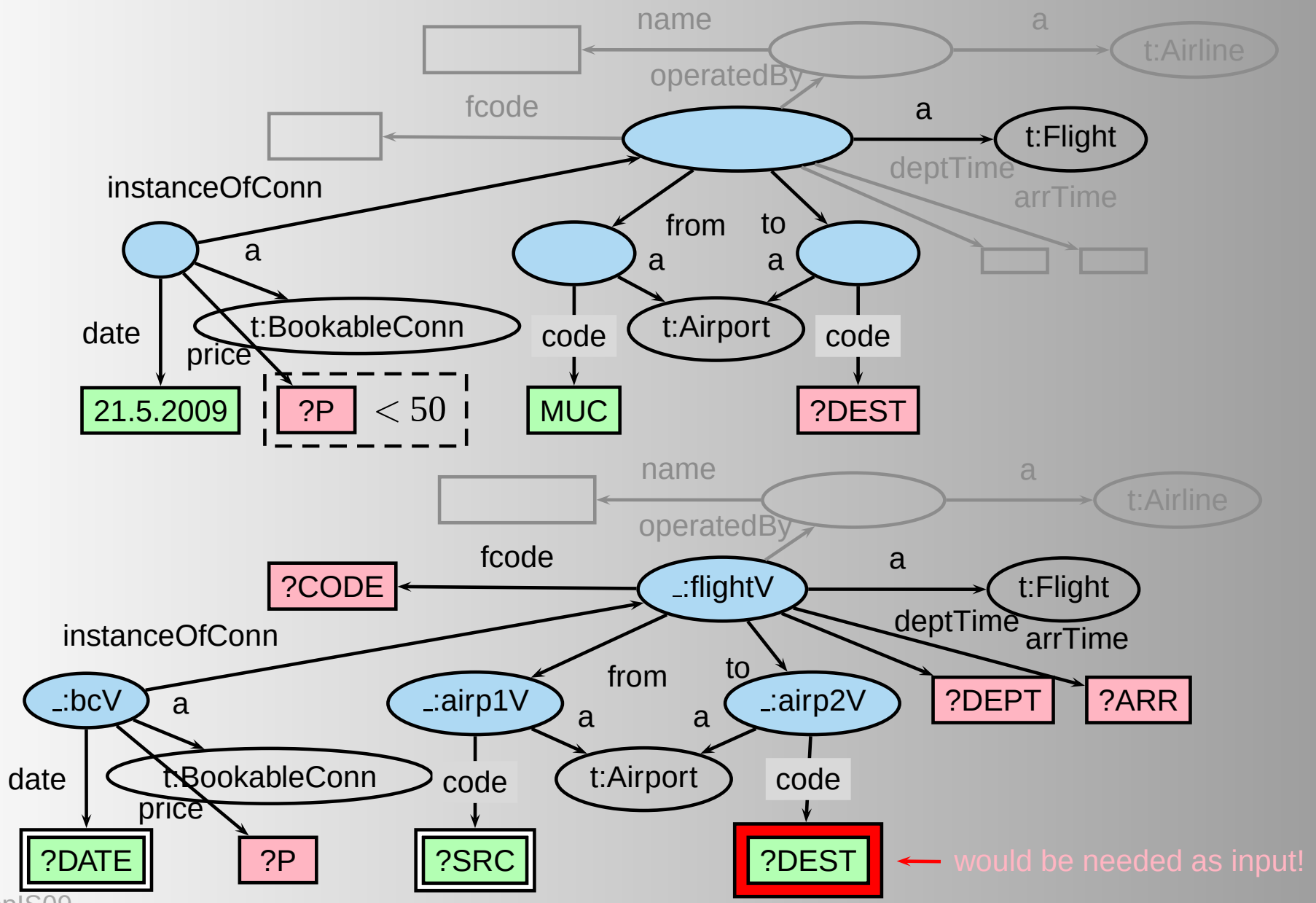
Example

“where can we go from MUC on 21.5.2009 for less than 50€?”

```
prefix t: <http://www.semwebtech.org/domains/2006/travel#>
select ?DEST ?P
where { ?C   a t:Flight;
             t:from [t:code “MUC”];
             t:to [ t:code ?DEST] .
        ?BC t:instanceOfConn ?C;
             t:date “21-05-2009”;
             t:price ?P .
        filter (?P < 50) }
```

- Match against views + in/out characteristics

Query-View-Coverage: Source A



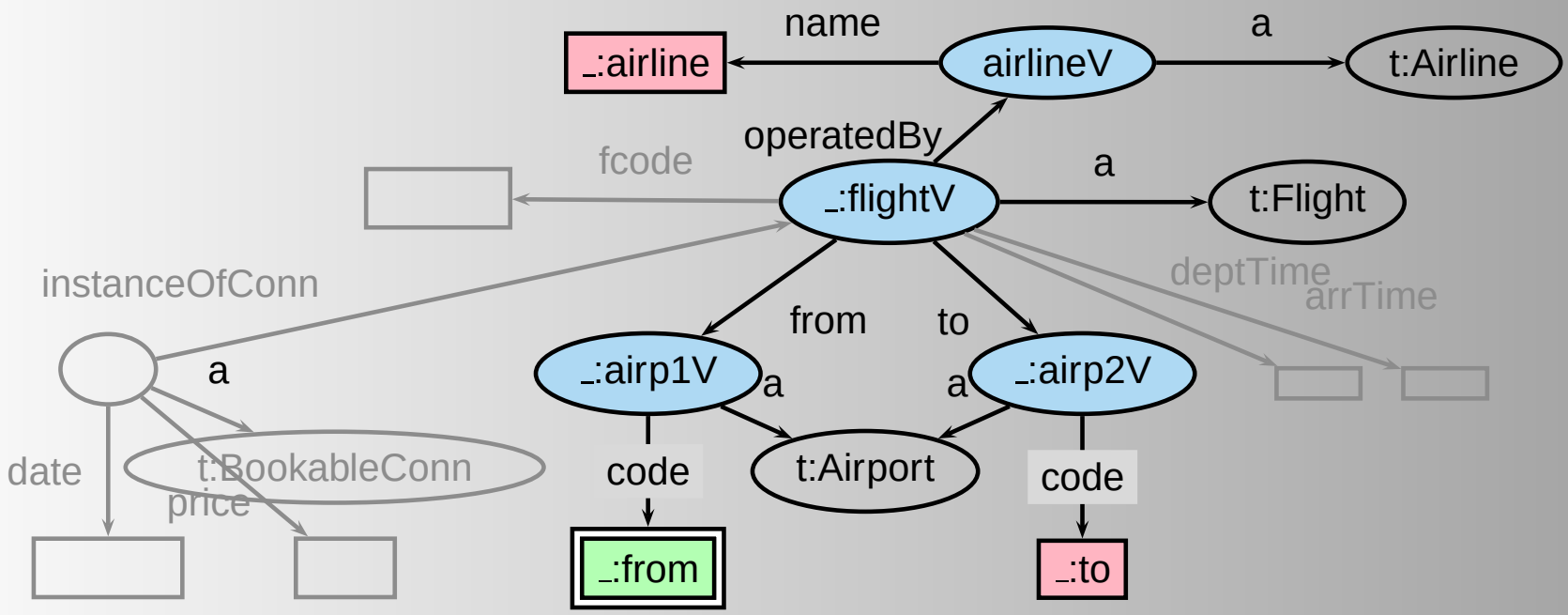
Source Selection and SPARQL Query Answering

Access Limitations

- Source (A) covers the notions required in the query.
- But: (A)'s view

  (airline, fcode, deptTime, arrTime, price) ← travenjoyConns(from, to, date)

  requires the destination (to) to be an input parameter.
- given from, Source (C) is able to return values for to, and also for airline.



Related Work

Query answering using views

- algorithms like MiniCon [Pottinger-Halevy-VLDB-01] construct a union of conjunctive queries over the available views that yields all obtainable answers.

1. q(To, Price) :-
   connection_A(“MUC”, To, “21.05.2009”, _Airl, _FCode, _DT, _AT, Price),
   Price $<$ 50.
2. q(To, Price) :-
   flight_C(_Airl, “MUC”, To),
   connection_A(“MUC”, To, “21.05.2009”, _Airl, _FCode, _DT, _AT, Price),
   Price $<$ 50.

- 1.) not executable under given access limitations (in/out)

$\Rightarrow$ adapt to general graph matching instead of flat predicates

Source Selection and SPARQL Query Answering

Querying data under access limitations

- Algorithms: [Yang-Kifer-Chaudhri-PODS-06], [Cali-Martinenghi-ER-08]
- apply to each solution,
- execution plans as processes on relational dataflow.

More General Case

Long Distance with Intermediate Flights

- Given (a set of) start airports, find paths to a target area,
- given departure airports, ask (C) for possible destinations,
- ask (A) for actual connections between dept.airport and dest.airport,
- generate graph and do A^*-search iteratively (poster at ODBASE'09)

Summary

- WDSDL ontology for annotating Web Data Sources to relate them to the terminology of their domain ontology,
- query broker selects appropriate sources,
- creates execution plans,
- and executes them.

$\Rightarrow$ The approach is currently under implementation.

$\Rightarrow$ Generalize to query workflows for e.g. transitive closure

Thank You

Questions?