Ontologies and Semantic Technologies

Izsó Benedek

Bergmann Gábor

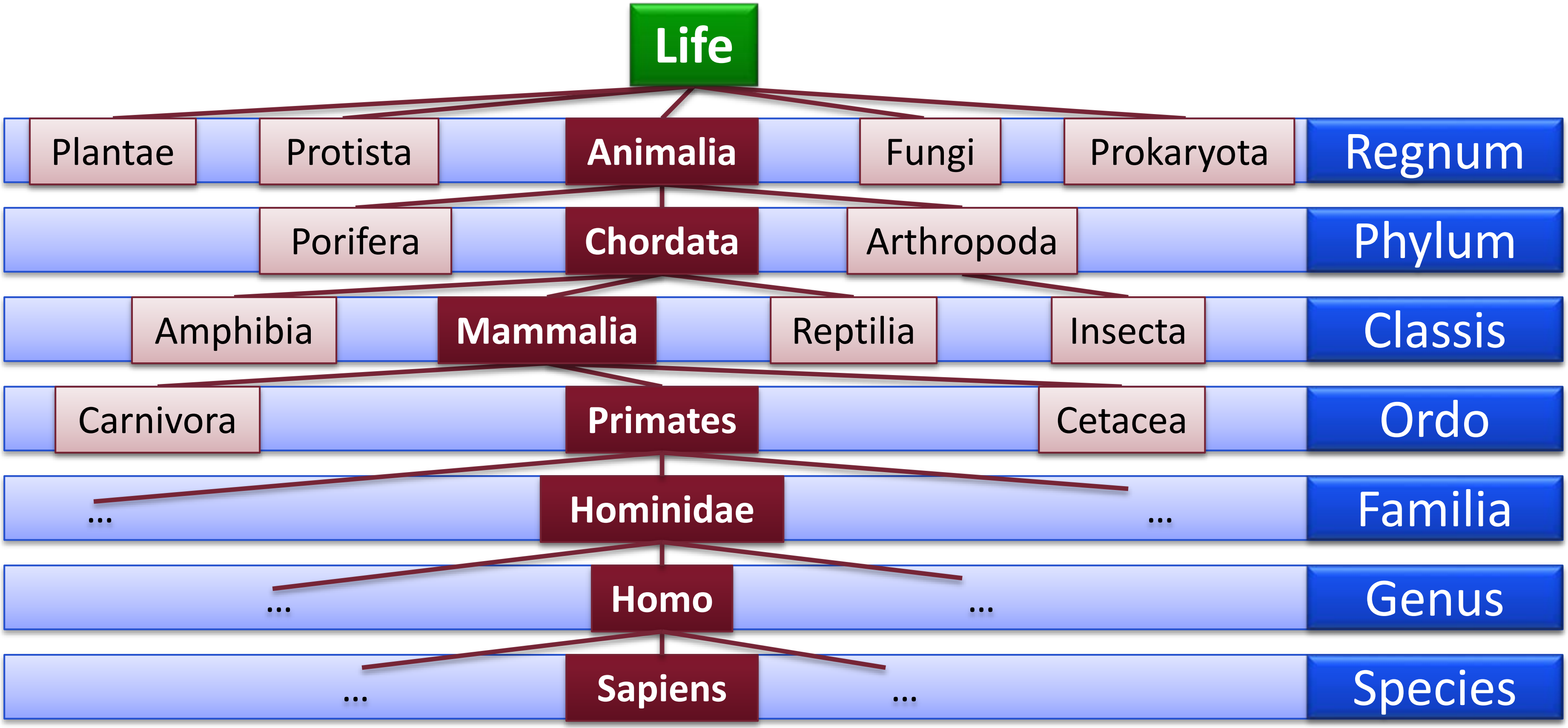
Agenda

- Ontologies
  - Resource Description Framework (RDF)
  - Querying ontologies (SPARQL)
  - Web Ontology Language (OWL)
- Semantic Technologies and Resources
- Modeling approaches
- Semantic Integration

ONTOLOGIES

Taxonomy

- **Taxonomy** = hierarchy of domain concepts



- Ontology ≅ taxonomy + relationships + definitions

Ontology

- **Ontology** = „the study of existence"
- Computer representation of **domain knowledge**
  - Identifying **concepts** to categorize **individuals**
  - **Relationships** that can hold between individuals
  - **Axioms** on concepts and their relationships
    - Including **taxonomy** of domain concepts (supertypes)
- Created by
  - **Domain experts, knowledge engineers**
  - People using annotation tools (e.g.: Annotator)
  - Natural language text analysis tools (e.g.: OpenCalais)

Domain Ontologies

- Open Biological and Biomedical Ontologies (OBO)
  - Chemical information
  - Cells, cell types, proteins, etc.
  - Anatomy (Upper/Human/…)
  - Medical software, imaging methods, spectrometry etc.
- National Center for Biomedical Ontology (NCBO)
  - National Drug File
  - International Classification of Deseases
  - SNOMED Clinical Terms
- data.gov (public access to US government data)
  - Documents categorized in an ontology

Open World Assumption

Can we enumerate all diseases?

- Traditional databases have Closed World Assumption
    - E.g. if not explicitly listed as a disease, then not a disease
- Most ontologies: **Open World Assumption** (OWA)
    - Not proven true/false → not treated as false or true
    - Why? Ontologies can never be complete
    - Examples
        - E.g. if not listed / implied as a viral disease, still can be one
        - Patient 42 has lepers. Does Patient 42 have a flu? Unknown!
        - Patient 2501 died of lepers. Did she die of flu? No!
          (by multiplicity 1 of cause of death)

No Unique Name Assumption

- Can two identifiers correspond to the same thing?
  - Patient 42 carries hereditary skin disease31.
  - Disease5 of patient 42 was found to be of viral nature.
  - Are they two different diseases? (→ unknown)
- Two things can be the same, unless contradicted
  - disjoint classes (hereditary and viral disease)
    - disease31 is a hereditary disease
  - explicit control: owl:sameAs, owl:differentFrom
    - disease31 owl:differentFrom disease5
- Usually NO **Unique Name Assumption** (UNA)
- Why? Distributed knowledge gathering

No Unique Name Assumption

- Can two identifiers correspond to the same thing?
  - Patient 42 carries hereditary skin disease31.
  - Disease5 of patient 42 was found to be of viral nature.
  - Are they two different diseases? (→ unknown)
- Two things can be the same, unless contradicted
  - disjoint classes (hereditary and viral disease)
    - disease31 is a I
  - explicit control:
    - disease31 owl:

John works at KeyWest.

Jill phoned Johnny.

- Usually NO **Unique Name Assumption** (UNA)
- Why? Distributed knowledge gathering

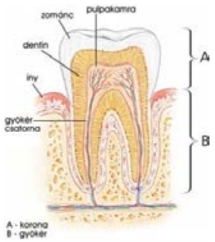
RESOURCE DESCRIPTION FRAMEWORK (RDF)

Metadata

- **Metadata**: description of data,
  - For people
  - For machines
- Example: image metadata
  - Generated partly automatically
  - „on this picture: John Doe, Jean-Baptiste Grenouille”
- Example: text document metadata
  - Author, literary category, year of publishing, etc.
- Metadata-based search

Syntactic Interpretation

- Can machines understand what we mean?
  - Textual / syntactic services can not
- Example: show me pictures depicting „fog”!



- Example: show me *poems* by *female authors*!
- Semantic solution
  - Machines should process the *meaning*, not the form
  - Use standardized concepts „fog”, „female”, „author”…
    - Refer to it in metadata and queries

Resource Description Framework

- W3C: Resource Description Framework (**RDF**)
- *Graph based* structure
  - Node: **rdf:Resource** → something we talk about
    - e.g. a document, this photo, a table or "something"
  - Edge: **rdf:Property** → relation type between resources
    - e.g depicts, taken_in, type etc.
- Node name and relation type name: **IRI** (Internationalized Resource Identifier)
- **Literal** nodes: 5^^xsd:integer, „John"

RDF Statements

- RDF statement = triple:
  - (subject, predicate, object)
  - subject is an IRI
  - predicate is an IRI
  - object: can be an IRI or a Literal
- Example triples
  - (this_photo, taken_in, Hungary)
  - (this_photo, file_name, „DSC0001.JPG”)
  - (this_photo, depicts, John Doe)

RDF Statements

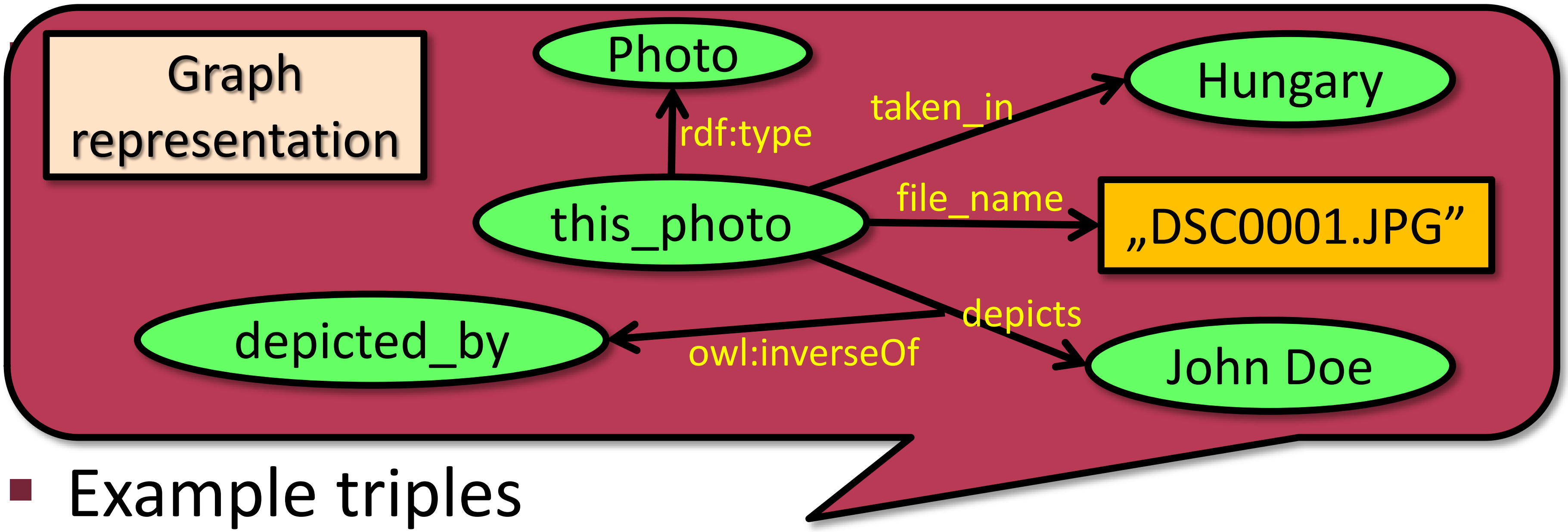
- RDF statement = triple:
  - (subject, predicate, object)
  - subject is an IRI
  - predicate is an IRI
  - object: can be an IRI or a Literal
- Example triples
  - (this_photo, taken_in, Hungary)
  - (this_photo, file_name, „DSC0001.JPG”)
  - (this_photo, depicts, John Doe)
  - (this_photo, has_type, Photo)
  - (rdf:type, rdf:type, rdf:Property)

RDF Statements

- RDF statement = triple:
  - (subject, predicate, object)
  - subject is an IRI
  - predicate is an IRI
  - object: can be an IRI or a Literal
- Example triples
  - (this_photo, taken_in, Hungary)
  - (this_photo, file_name, „DSC0001.JPG")
  - (this_photo, depicts, John Doe)
  - (this_photo, rdf:type, Photo)
  - (rdf:type, rdf:type, rdf:Property)

rdf:type is defined in the RDF standard

RDF Statements



- Example triples
  - (this_photo, taken_in, Hungary)
  - (this_photo, file_name, „DSC0001.JPG”)
  - (this_photo, depicts, John Doe)
  - (this_photo, rdf:type, Photo)
  - (rdf:type, rdf:type, rdf:Property)

Resource Description Framework

- RDF based modeling:
  - Graph based modeling with triples („triple stores”)
  - Graphs can be grouped: „quad stores”
- Languages:
  - RDF: rdf:type, graph based description
  - RDFS (RDF Schema): rdfs:subClassOf, rdfs:domain, rdfs:range
  - Other custom specifications (foaf, geo, ...)

Photo rdfs:subclassOf Picture
taken_in rdfs:domain Photo
taken_in rdfs:range geo:Feature

RDF Concrete Syntaxes

- **RDF+XML**

```
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:geo="http://www.geonames.org/ontology#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:foaf="http://xmlns.com/foaf/0.1/">

<foaf:Person rdf:about="http://www.w3.org/People/Ivan/">
 <foaf:firstName>Ivan</foaf:firstName>
 <foaf:surname>Herman</foaf:surname>
 <foaf:homepage rdf:resource="http://www.ivan-herman.net/"/>
 <foaf:depiction rdf:resource="http://www.ivan-herman.net/Images/me2003-small.png"/>

 <foaf:based_near>
  <geo:Feature>
   <geo:name>Hungary</geo:name>
  </geo:Feature>
 </foaf:based_near>

 <foaf:knows>
  <foaf:Person rdf:about="http://www.openlinksw.com/dataspace/oerling">
   <foaf:firstName>Orri</foaf:firstName>
   <foaf:surname>Erling</foaf:surname>
   <foaf:homepage rdf:resource="http://www.openlinksw.com/weblog/oerling/"/>
  </foaf:Person>
 </foaf:knows>
</foaf:Person>

</rdf:RDF>
```

RDF Concrete Syntaxes

- **RDF+XML**

```
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:geo="http://www.geonames.org/ontology#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:foaf="http://xmlns.com/foaf/0.1/">

<foaf:Person rd      ut="http://www.w3.org/People/Ivan/">
 <foaf:firstName       /foaf:firstName>
 <foaf:surname>He          surname>
 <foaf:homepage rd           tp://www.ivan-herman.net/"/>
 <foaf:depiction rd            //www.ivan-herman.net/Images/me2003-small.png"/>

 <foaf
  <geo
   <ge
  </ge
 </foa

 <foaf
  <foa
   <fo
   <foa
   <foaf:homepage rdf:resource="http://www.openlinksw.com/weblog/oerling/"/>
  </foaf:Person>
 </foaf:knows>
</foaf:Person>

</rdf:RDF>
```

xmlns:foaf=http://xmlns.com/foaf/0.1/

Organize concepts and relation types into namespaces

RDF Concrete Syntaxes

- **RDF+XML**

```
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:geo="http://www.geonames.org/ontology#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:foaf="http://xmlns.com/foaf/0.1/">

<foaf:Person rdf:about="http://
 <foaf:firstName>Ivan</foaf:fi
 <foaf:surname>Herman</foaf:su
 <foaf:homepage rdf:resource="
 <foaf:depiction rdf:resource=

 <foaf:based_near>
  <geo:Feature>
   <geo:name>Hungary</geo:
  </geo:Feature>
 </foaf:based_near>

 <foaf:knows>
  <foaf:Person rdf:about="http://www.openlinksw.com/dataspace/oerling">
   <foaf:firstName>Orri</foaf:firstName>
   <foaf:surname>Erling</foaf:surname>
   <foaf:homepage rdf:resource="http://www.openlinksw.com/weblog/oerling/"/>
  </foaf:Person>
 </foaf:knows>
</foaf:Person>

</rdf:RDF>
```

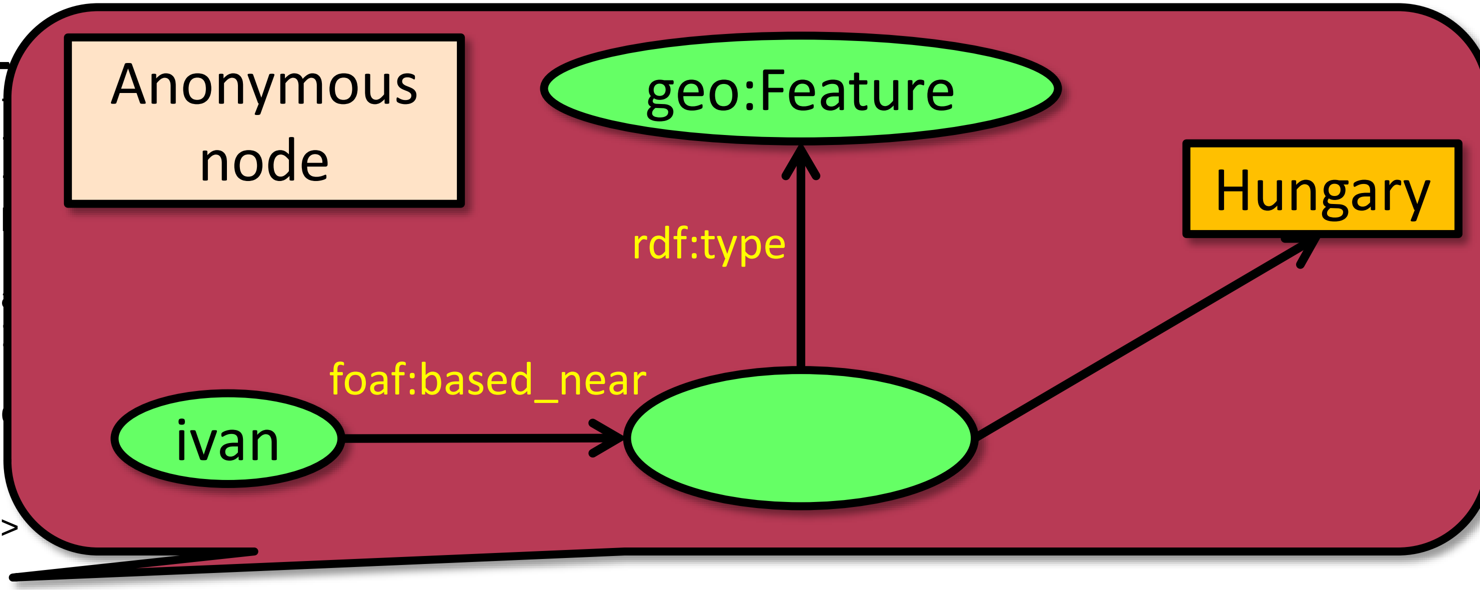
(oerling, ***rdf:type***, foaf:Person)

(oerling, ***foaf:firstName***, “Orri”)

The semantics of these concepts and relations are well understood, these means the same for everyone.

RDF Concrete Syntaxes

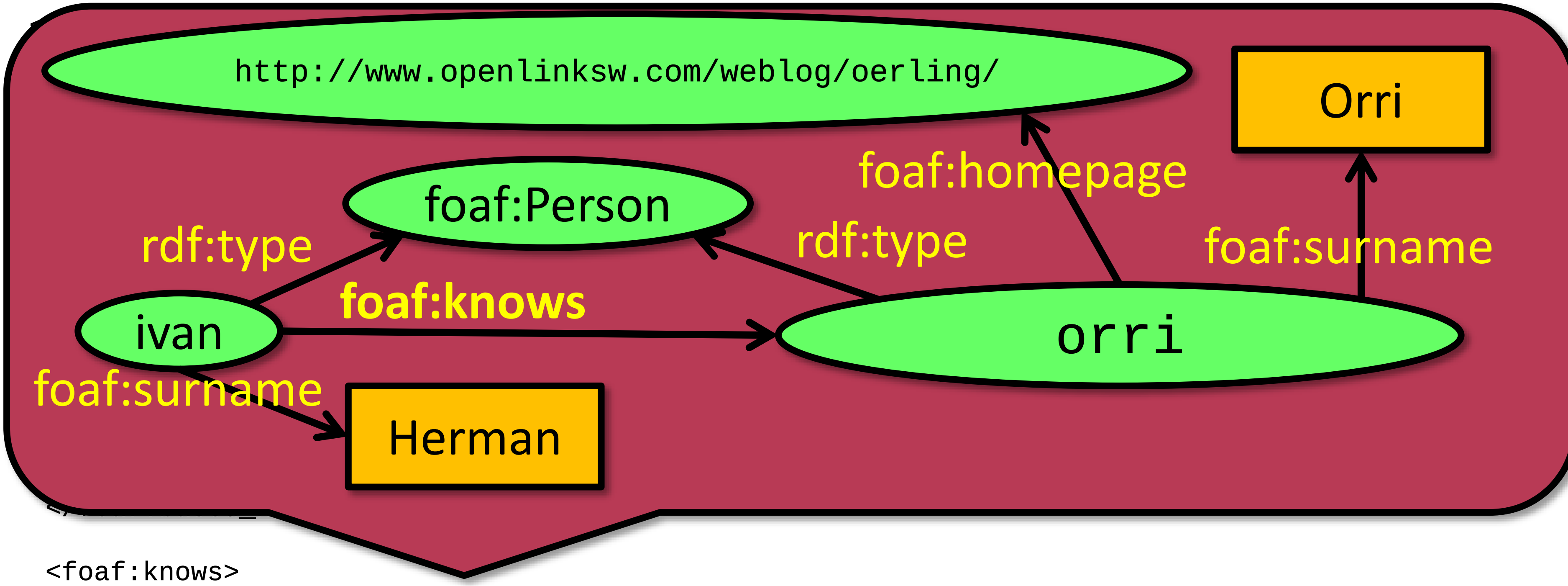
- **RDF+XML**




```
<rdf:RDF xmlns:rd
      xmlns:geo="h
      xmlns:rdf="h
      xmlns:foaf="

<foaf:Person rdf:
 <foaf:firstName>
 <foaf:surname>He
 <foaf:homepage r
 <foaf:depiction

 <foaf:based_near>
  <geo:Feature>
   <geo:name>Hungary</geo:name>
  </geo:Feature>
 </foaf:based_near>

 <foaf:knows>
  <foaf:Person rdf:about="http://www.openlinksw.com/dataspace/oerling">
   <foaf:firstName>Orri</foaf:firstName>
   <foaf:surname>Erling</foaf:surname>
   <foaf:homepage rdf:resource="http://www.openlinksw.com/weblog/oerling/"/>
  </foaf:Person>
 </foaf:knows>
</foaf:Person>

</rdf:RDF>
```

RDF Concrete Syntaxes

- **RDF+XML**




```
 <foaf:knows>
  <foaf:Person rdf:about="http://www.openlinksw.com/dataspace/oerling">
   <foaf:firstName>Orri</foaf:firstName>
   <foaf:surname>Erling</foaf:surname>
   <foaf:homepage rdf:resource="http://www.openlinksw.com/weblog/oerling/"/>
  </foaf:Person>
 </foaf:knows>
</foaf:Person>

</rdf:RDF>
```

RDF Concrete Syntaxes

- RDF+XML
- **Turtle** (Terse RDF Triple Language)

```
1. <ftsrg:ivan>   rdf:type        owl:NamedIndividual ,
2.                                foaf:Person ;
3.                foaf:firstName  "Ivan" ;
4.                foaf:homepage   <http://www.ivan-herman.net/> .
```

- **RDFa:** tag a HTML webpage with RDF attributes

```
<div xmlns:v="http://rdf.data-vocabulary.org/#"
typeof="v:Person">
  My name is <span property="v:name">Bob Smith</span>,
  but people call me <span property="v:nickname">Smithy</span>.
  Here is my homepage:
  <a href="http://www.example.com"
rel="v:url">www.example.com</a>.
</div>
```

RDF Tools

- RDF API: Sesame, Jena
- RDF engine for inference and query:
  OpenLink Virtuoso, Allegro Graph, 4store, Stardog

SPARQL RDF Query

- Recursive acronym:
  SPARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
- Graph Pattern based Query Language
  - Basic Graph Pattern (BGP) with attribute checks
  - Aggregates
  - Solution modifiers: ORDER BY, LIMIT, OFFSET
- Update protocol
  - HTTP based
  - INSERT, DELETE
- SQL-like keywords

SPARQL RDF Query

```
SELECT ?persA ?persB
WHERE {
   ?persA rdf:type foaf:Person .
   ?persB rdf:type foaf:Person .
   ?persA foaf:knows ?persB
}
```

SPARQL RDF Query

Return variables

```
SELECT ?persA ?persB
WHERE {
    ?persA rdf:type foaf:Person .
    ?persB rdf:type foaf:Person .
    ?persA foaf:knows ?persB
}
```

SQL-like keywords

RDF triples with variables (starting with ‚?’)

‚.’ character denotes and/join

SPARQL RDF Query

Return variables

```
SELECT ?persA ?persB
WHERE {
    ?persA rdf:type foaf:Person .
    ?persB rdf:type foaf:Person .
    ?persA foaf:knows ?persB
}
```

Give me pairs,
where
the first entity is a Person and
the second entity is a Person and
the first knows the second.

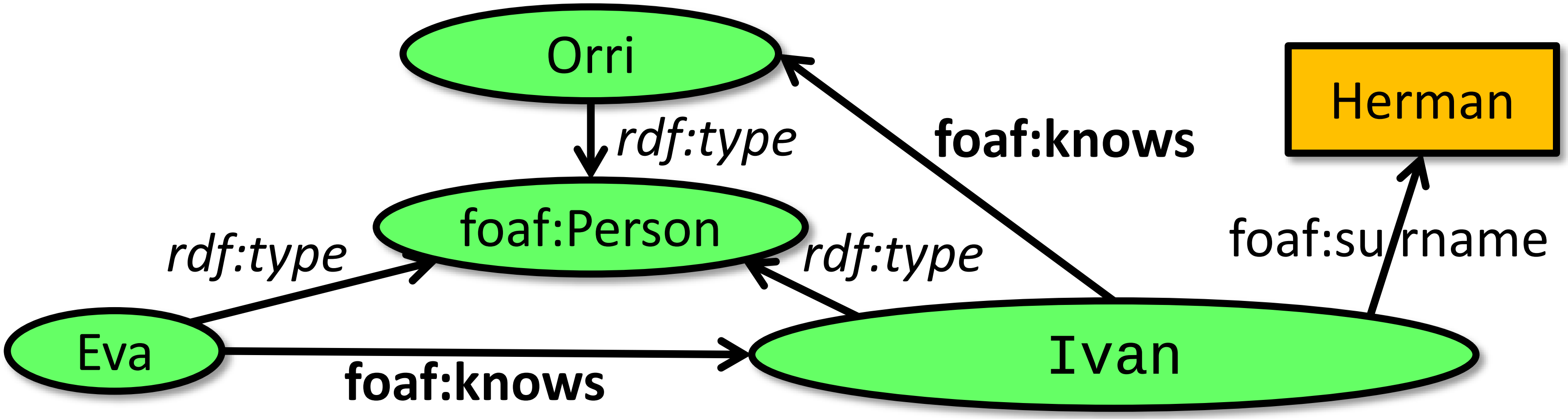
SQL-like keywords

RDF triples with variables (starting with ‚?')

‚.' character denotes and/join

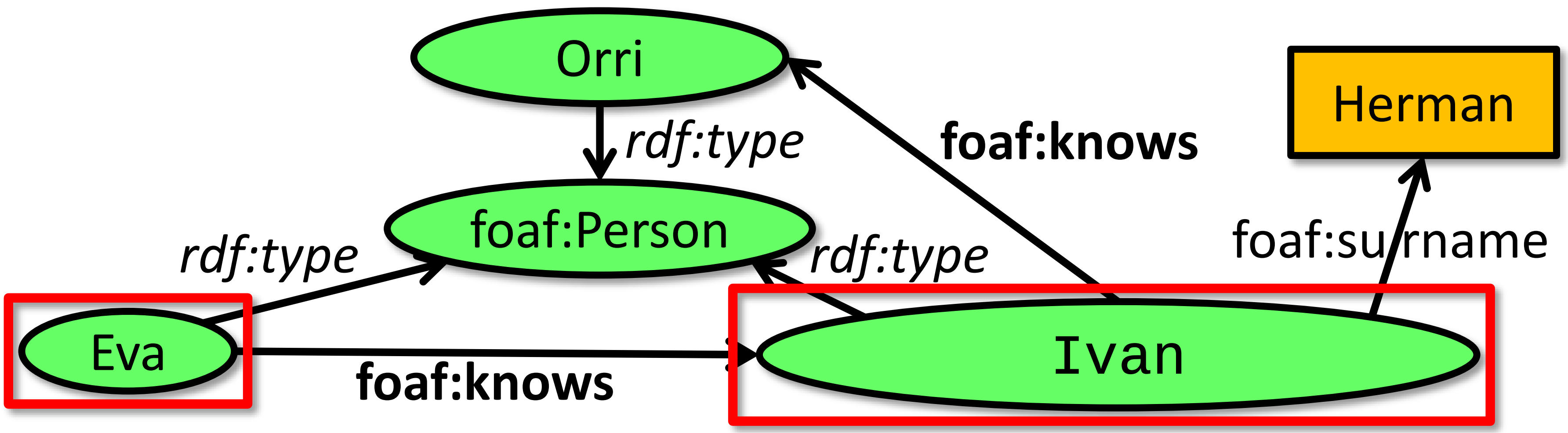
SPARQL RDF Query

```
SELECT ?persA ?persB
WHERE {
   ?persA rdf:type foaf:Person .
   ?persB rdf:type foaf:Person .
   ?persA foaf:knows ?persB
}
```



SPARQL RDF Query

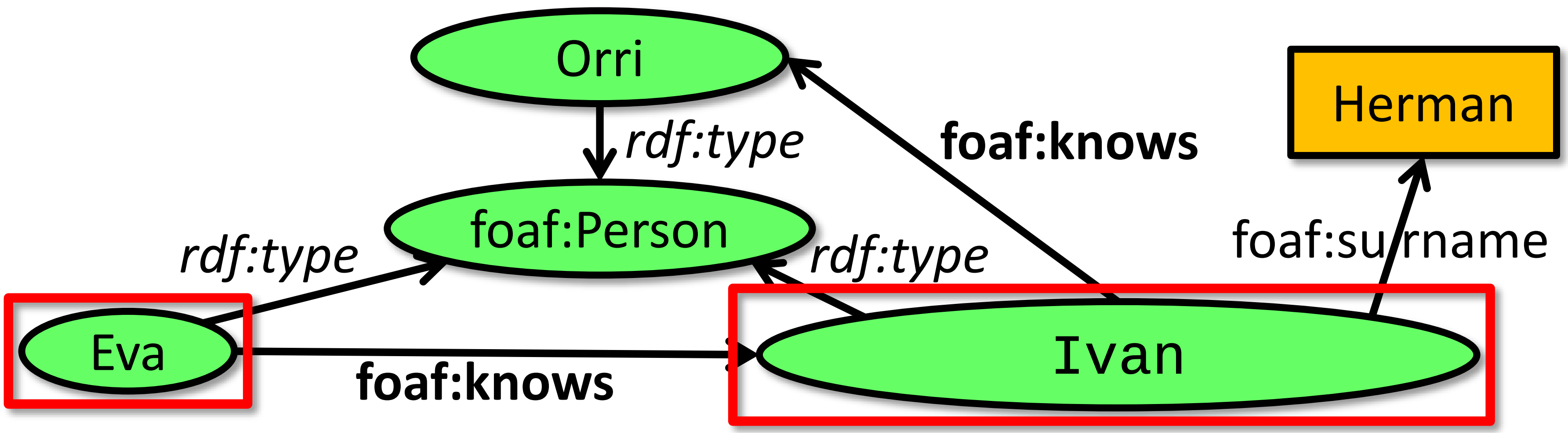
```
SELECT ?persA ?persB
WHERE {
   ?persA rdf:type foaf:Person .
   ?persB rdf:type foaf:Person .
   ?persA foaf:knows ?persB
}
```



SPARQL RDF Query

```
SELECT ?persA ?persB
WHERE {
   ?persA rdf:type foaf:Person .
   ?persB rdf:type foaf:Person .
   ?persA foaf:knows ?persB
}
```

?persA	?persB
Eva	Ivan
Ivan	Orri

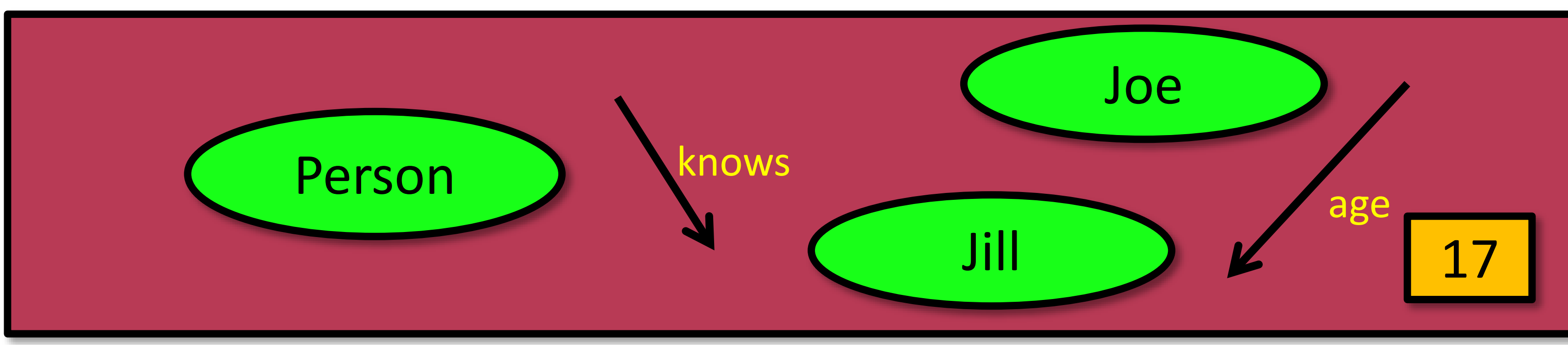


WEB ONTOLOGY LANGUAGE (OWL 2)

Formal Background

- Axiomatic language
  - Declaration (for tools): concepts, roles, individuals

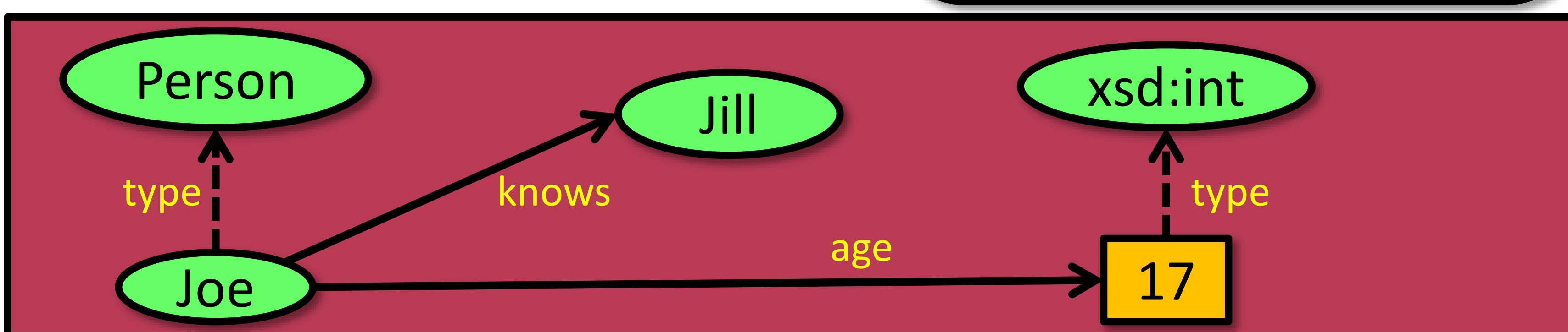
Class: Person
ObjectProperty: knows
DatatypeProperty: age
Individual: Joe, Jill



Formal Background

- Axiomatic language
  - Declaration (for tools): concepts, roles, individuals
  - A-Box (instance model level): typify individuals, data and object role assertions

Individual: Joe
    Types: Person
    Facts:
        knows Jill,
        age "17"^^xsd:int



Formal Background

- Axiomatic language
  - Declaration (for tools): concepts, roles, individuals
  - A-Box (instance model level): typify individuals, data and object role assertions
  - T-Box (metamodel level): subsumption between class expressions, which use role navigations and bool operators

Every male is a person:

$Male \sqsubseteq Person$

The intersection of males and females is the empty set:

$Male \sqcap Female \equiv \bot$ // $\bot$ sign means: empty set

Single child is a person, whose all parents have only one child:

$SingleChild \equiv Person \sqcap \forall parent{<=}1 child.Person$

Formal Background

- Axiomatic language
  - Declaration (for tools): concepts, roles, individuals
  - A-Box (instance model level): typify individuals, data and object role assertions
  - T-Box (metamodel level): subsumption between class expressions, which use role navigations and bool operators
- OWL Dialects:
  - OWL 2 RDF-Based Semantics: supports multi-level metamodeling, not decidable
  - OWL 2 Direct Semantics: $\mathcal{SROIQ}(\mathcal{D})$ description logic (DL)
    - decidable **reasoning** and **consistency checking**
  - OWL 2 profiles: ($\mathcal{EL}$, $\mathcal{QL}$, $\mathcal{RL}$)
    - reduced expressivity → more efficient algorithms

Formal Background

- Axiomatic language
  - o
  - o
  - o

$\mathcal{S}$: Mother ≡ ∃child.Person // have child who is a person

$\mathcal{R}$: ObjectProperty knows // everybody knows oneself

Characteristics: Reflexive

$\mathcal{O}$: Visibility ≡ {private, public} // nominals (enumeration)

$\mathcal{I}$: child ≡ parent^{-1} // Inverse

$\mathcal{Q}$: FootballCoach ⊑ ≥11 knows.FootballPlayer

// a football coach knows at least 11 football player

$(\mathcal{D})$: Individual: Joe

Facts: age "17"^^xsd:int // data assertion

- O
  - o
  - o OWL 2 Direct Semantics: $\mathcal{SROIQ}(\mathcal{D})$ description logic (DL)
    - decidable **reasoning** and **consistency checking**
  - o OWL 2 profiles: (Ɛ$\mathcal{L}$, $\mathcal{QL}$, $\mathcal{RL}$)
    - reduced expressivity → more efficient algorithms

Formal Background

- Axiomatic language
  - Declaration (for tools): concepts, roles, individuals
  - A-Box (instance model level): typify individuals, data and object role assertions
  - T-Box (metam
    expressions, w
- OWL Dialects:
  - OWL 2 RDF-Ba
    metamodeling
  - OWL 2 Direct S
    - decidable **reasoning** ... **nsistency checking**
  - OWL 2 profiles: ($\mathcal{EL}$, $\mathcal{QL}$, $\mathcal{RL}$)
    - reduced expressivity → more efficient algorithms

OWL2 EL profile:
- useful for ontologies that contain very large number of classes and relations
- only existential quantification is enabled
- polinomial reasoning algorithms can be used w.r.t. size of the ontology

Formal Background

- Axiomatic language
  - Declaration (for tools): concepts, roles, individuals
  - A-Box (instance model level): typify individuals, data and object role assertions
  - T-Box (metamodel level): subsumption between class expressions, which use role navigations and bool operators
- OWL Dialects:
  - OWL 2 RDF-Based Semantics: supports multi-level metamodeling, not decidable
  - OWL 2 Direct Semantics: $\mathcal{SROIQ(D)}$ description logic (DL)
    - decidable **reasoning** and **consistency checking**
  - OWL 2 profiles: ($\mathcal{EL}$, $\mathcal{QL}$, $\mathcal{RL}$)
    - reduced expressivity → more efficient algorithms

Concrete Syntaxes

- OWL is compatible with RDF, hence RDF tools can be used (at RDF semantics level)
  - RDF/XML is obligatory for OWL 2 tools
  - For data change, not human readable

```
<owl:Class rdf:about=“Young">
      <owl:equivalentClass>
         <owl:Restriction>
            <owl:onProperty rdf:resource=“age"/>
            <owl:someValuesFrom>
               <rdfs:Datatype>
                  <owl:onDatatype rdf:resource="&xsd;int"/>
                  <owl:withRestrictions rdf:parseType="Collection">
                     <rdf:Description>
                        <xsd:maxExclusive rdf:datatype="&xsd;integer">18</xsd:maxExclusive>
                     </rdf:Description>
                  </owl:withRestrictions>
               </rdfs:Datatype>
            </owl:someValuesFrom>
         </owl:Restriction>
      </owl:equivalentClass>
   </owl:Class>
```

Concrete Syntaxes

- OWL is compatible with RDF, hence RDF tools can be used (at RDF semantics level)
  - RDF/XML is obligatory for OWL 2 tools
  - For data change, not human readable
- Functional Syntax: close to AST, readable

```
EquivalentClasses(:Young
DataSomeValuesFrom(:age DatatypeRestriction(xsd:int xsd:maxExclusive "18"^^xsd:integer)))
```

Concrete Syntaxes

- OWL is compatible with RDF, hence RDF tools can be used (at RDF semantics level)
  - RDF/XML is obligatory for OWL 2 tools
  - For data change, not human readable
- Functional Syntax: close to AST, readable
- OWL/XML: more compact than RDF/XML, but not used widely

```
<EquivalentClasses>
  <Class IRI="#Young"/>
  <DataSomeValuesFrom>
    <DataProperty IRI="#age"/>
    <DatatypeRestriction>
      <Datatype abbreviatedIRI="xsd:int"/>
      <FacetRestriction facet="&xsd;maxExclusive">
        <Literal datatypeIRI="&xsd;integer">18</Literal>
      </FacetRestriction>
    </DatatypeRestriction>
  </DataSomeValuesFrom>
</EquivalentClasses>
```

Concrete Syntaxes

- OWL is compatible with RDF, hence RDF tools can be used (at RDF semantics level)
  - RDF/XML is obligatory for OWL 2 tools
  - For data change, not human readable
- Functional Syntax: close to AST, readable
- OWL/XML: more compact than RDF/XML, but not used widely
- Manchester: readable, used in ontology editors

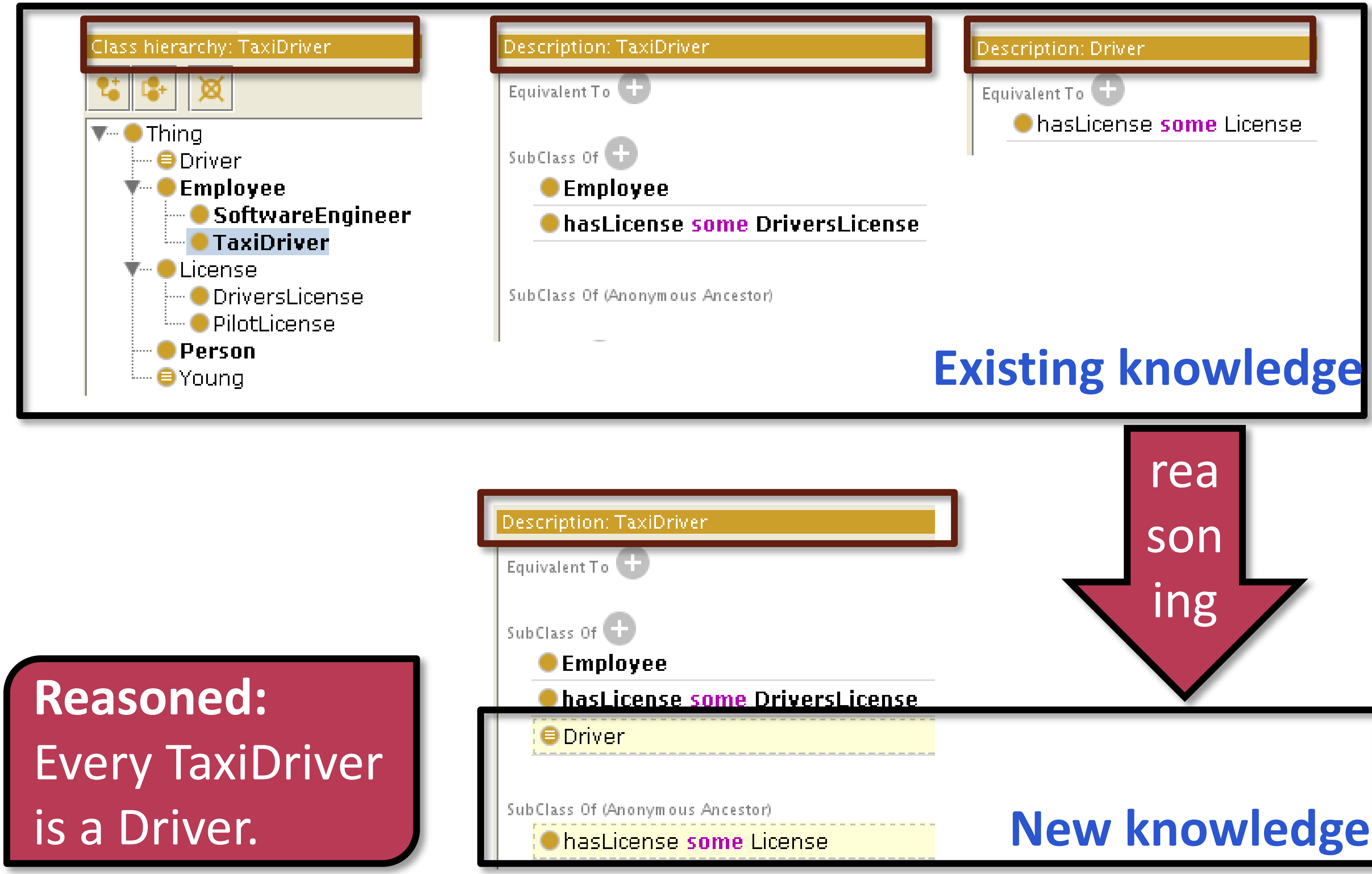
```
Class: Young
    EquivalentTo:
        age some xsd:int[< 18]
```

Tools

- Editors: Protégé
- Most usable part: DL subset for **reasoning** and **consistency checking**
  - Reasoning: infer new knowledge from existing ones
  - Reasoning algorithms: tableau calculi
  - Open World Assumption
  - No Unique Name Assumption,
    however reasoners can be configured to use
- Reasoners: Pellet, RacerPro, HermiT
- API: OWL API (owlapi.sf.net), Java based

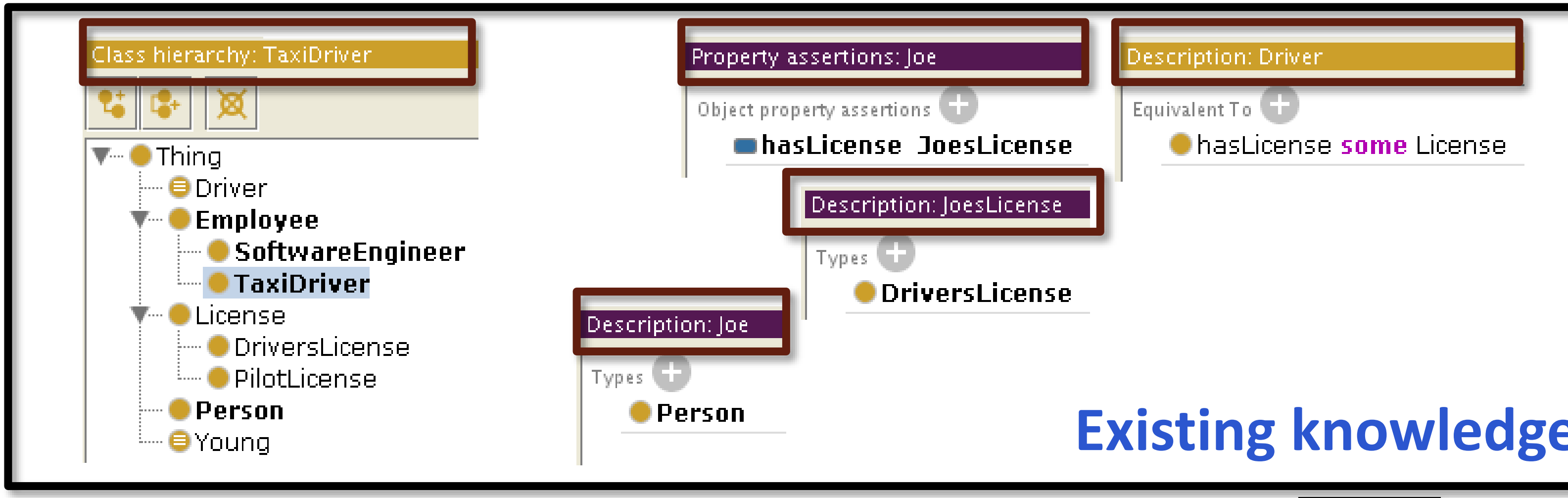
Ontology editing and reasoning demo

T-Box reasoning: new knowledge at meta level



Ontology editing and reasoning demo

A-Box reasoning: new assertion at instance level

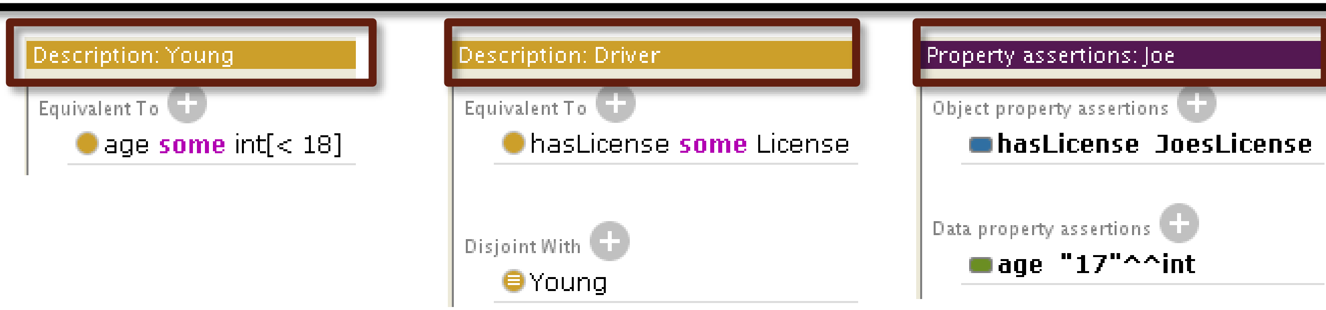


rea
son
ing

Reasoned:
Joe is a Driver.

Ontology editing and reasoning demo

Consistency checking: detect conflicting axioms



Existing knowledge

rea
son
ing

Reasoned:
Inconsistent ontology

$ pellet explain –inconsistent profession.owl
Axiom: Thing subClassOf Nothing

Explanation(s):
1) JoesLicense type DriversLicense
Young equivalentTo age some int[< 18]
Driver equivalentTo hasLicense some License
Joe hasLicense JoesLicense
DriversLicense subClassOf License
Joe age "17"^^int
Driver disjointWith Young

SEMANTIC TECHNOLOGIES AND RESOURCES

RDF Application

- **Semantic Web**
  - Is a photo of my Porsche a photo of a car?
  - Need standard IRIs for RDF resource/property types
  - Local metadata + ontologies = semantic web

RDF Application

- RDF Site Summary (**RSS**)
  - Items with title, description, link, creator, date, …
  - RSS 2.0 abandons RDF, backronym

RDF Application

- RDF Site Summary (**RSS**)
  - Items with title, description, link, creator, date, …
  - RSS 2.0 abandons RDF, backronym

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://purl.org/rss/1.0/"
        xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
        xmlns:dc="http://purl.org/dc/elements/1.1/" >
<title>Wikipedia Mobile Apps Switch To OpenStreetMap</title>
<link>http://rss.slashdot.org/~r/Slashdot/slashdot/~3/fQXcTk0g-aY/
      wikipedia-mobile-apps-switch-to-openstreetmap</link>
<dc:creator>timothy</dc:creator>
<dc:date>2012-04-08T14:31:00+00:00</dc:date>
<dc:subject>google</dc:subject>
<slash:department>location-aware</slash:department>
<slash:section>mobile</slash:section>
<slash:comments>125</slash:comments>
<slash:hit_parade>125,124,96,79,35,22,14</slash:hit_parade>
```

RDF Application

- RDF Site Summary (**RSS**)
  - Items with title, description, link, creator, date, …
  - RSS 2.0 abandons RDF, backronym

Standard IRIs

Defined vocabulary used to describe data

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://purl.org/rss/1.0/"
         xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
         xmlns:dc="http://purl.org/dc/elements/1.1/" >
<title>Wikipedia Mobile Apps Switch To OpenStreetMap</title>
<link>http://rss.slashdot.org/~r/Slashdot/slashdot/~3/fQXcTk0g-aY/
      wikipedia-mobile-apps-switch-to-openstreetmap</link>
<dc:creator>timothy</dc:creator>
<dc:date>2012-04-08T14:31:00+00:00</dc:date>
<dc:subject>google</dc:subject>
<slash:department>location-aware</slash:department>
<slash:section>mobile</slash:section>
<slash:comments>125</slash:comments>
<slash:hit_parade>125,124,96,79,35,22,14</slash:hit_parade>
```

Semantic Web Vocabularies

- Dublin Core (DC) ontology
  - Librarian metadata for documents
  - Title, publisher, language, format, date, creator, etc.
  - Widespread usage (e.g. as an RSS Module)
- Friend-of-a-Friend (FOAF) ontology
  - Classes: Person, Image, Document, OnlineAccount, etc.
  - Attributes: surname, birthday, title, etc.
  - Relationships: knows, made, depicts, weblog, topic, logo, openID, page, interest, etc.
  - Used / usable in Facebook, flickr, LauchPad…

Semantic Web Vocabularies

- **Opengraph Protocol (Facebook)**

A Like button on a movie page on imdb.co

* More info: The Open Graph Protocol: Understanding the design decisions

Semantic Web Vocabularies

- **Opengraph Protocol (Facebook)**

A Like button on a movie page on imdb.co

Me:
- I like The Rocks

* More info: The Open Graph Protocol: Understanding the design decisions

Semantic Web Vocabularies

- **Opengraph Protocol (Facebook)**

A Like button on a movie page on imdb.co

Me:
- I like The Rocks

Computer:
- What?? This HTML tag, or CSS?

* More info: The Open Graph Protocol: Understanding the design decisions

Semantic Web Vocabularies

- **Opengraph Protocol (Facebook)**

A Like button on a movie page on imdb.co

Me:
- I like The Rocks

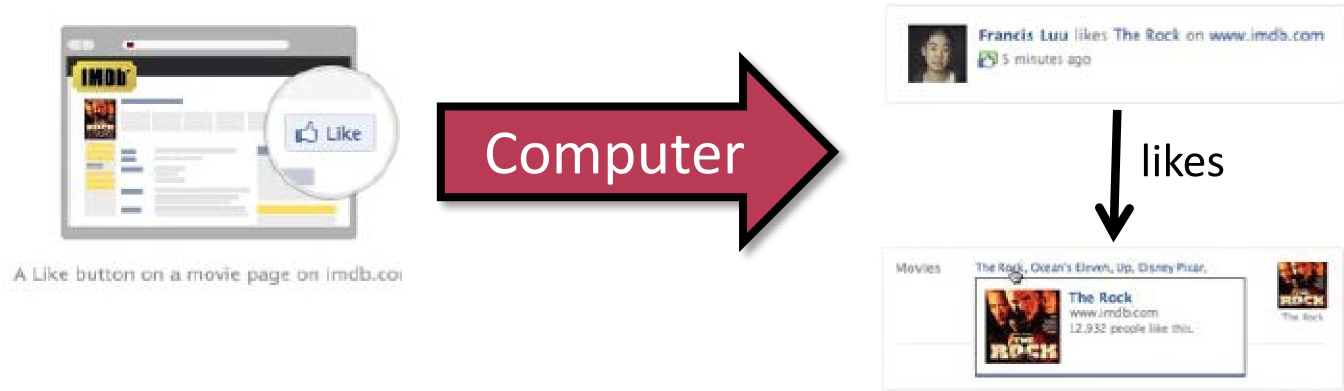
Computer:
- What?? This HTML tag, or CSS?

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:og="http://opengraphprotocol.org/schema/" >
<head>
<meta property="og:title" content="The Rock (1996)"/>
<meta property="og:type" content="video.movie"/>
<meta property="og:image" content="http://media-imdb.com/images/the_rocks.jpg"/>
<meta property="og:site_name" content="IMDb"/>
<meta property="fb:app_id" content="115109575169727"/>
</head>
```

* More info: The Open Graph Protocol: Understanding the design decisions

Semantic Web Vocabularies

- **Opengraph Protocol (Facebook)**

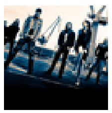



```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:og="http://opengraphprotocol.org/schema/" >
<head>
<meta property="og:title" content="The Rock (1996)"/>
<meta property="og:type" content="video.movie"/>
<meta property="og:image" content="http://media-imdb.com/images/the_rocks.jpg"/>
<meta property="og:site_name" content="IMDb"/>
<meta property="fb:app_id" content="115109575169727"/>
</head>
```

* More info: The Open Graph Protocol: Understanding the design decisions

- **Microformat**

Tuesday 1 April 2014



The Invincible Tour 2014
Amaranthe, Smash Into Pieces, Deals Death

Club 202
Budapest,
Hungary

Wednesday 2 April 2014

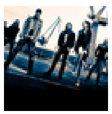
The Invincible Tour 2014
Amaranthe, Smash Into Pieces, Deals Death

Hudební klub Nová Chmelnice
Praha,
Czech Republic

Semantic Web Vocabularies

- **Microformat**

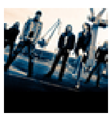
Tuesday 1 April 2014



The Invincible Tour 2014
Amaranthe, Smash Into Pieces, Deals Death

Club 202
Budapest,
Hungary

Wednesday 2 April 2014



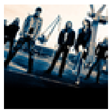
The Invincible Tour 2014
Amaranthe, Smash Into Pieces, Deals Death

Hudební klub Nová Chmelnice
Praha,
Czech Republic

Last.fm:
We collected the events for the band Amaranthe. How can we make data more searchable?

- **Microformat**

Tuesday 1 April 2014



The Invincible Tour 2014
Amaranthe, Smash Into Pieces, Deals Death

Club 202
Budapest,
Hungary

```
<abbr class="dtstart" title="2014-04-01T19:30:00">Tuesday 1 April 2014</abbr>
<td class="location vcard">
  <span class="fn org"><a href="/venue/10234205+Club+202">Club 202</a></span>
  <span class="adr"><br />
    <span class="locality">Budapest</span>,
    <span class="country-name">Hungary</span>
  </span>
  <div class="geo">
     <span class="latitude">47.440592</span>
     <span class="longitude">19.036936</span>
  </div>
</td>
```

- **Microformat**


```
<abbr class="dtstart" title="2014-04-01T19:30:00">Tuesday 1 April 2014</abbr>
<td class="location vcard">
  <span class="fn org"><a href="/venue/10234205+Club+202">Club 202</a></span>
  <span class="adr"><br />
    <span class="locality">Budapest</span>,
    <span class="country-name">Hungary</span>
  </span>
  <div class="geo">
     <span class="latitude">47.440592</span>
     <span class="longitude">19.036936</span>
  </div>
</td>
```

Semantic Web Vocabularies

- **Microdata: schema.org (Google)**

Semantic Web Vocabularies

- **Microdata: schema.org (Google)**

My homepage

[Benedek Izsó]
EMF-IncQuery is a great tool. Rating based on my opinion:
5 stars - based on 1 opinion

- **Microdata: schema.org (Google)**

Benedek Izsó
EMF-IncQuery is a great tool. Rating based on my opinion:
5 stars - based on 1 opinion

```
<body>
<div itemscope itemtype="http://schema.org/People">
 <span itemprop="name">
   <a href="https://plus.google.com/111902568077770766904?rel=author">
     Benedek Izsó</a>
</span>
</div>
<div itemscope itemtype="http://schema.org/Product">
  <span itemprop="name">EMF-IncQuery</span> is a great tool. Rating based on my opinion:
  <div itemprop="aggregateRating" itemscope itemtype="http://schema.org/AggregateRating">
    <span itemprop="ratingValue">5</span> stars -
    based on <span itemprop="reviewCount">1</span> review
  </div>
</div>
</body>
```

- **Microdata: schema.org (Google)**

Benedek Izsó
EMF-IncQuery is a great tool. Rating based on my opinion:
5 stars - based on 1 opinion

```
<body>
<div itemscope itemtype="http://schema.org/People">
 <span itemprop="name">
   <a href="https://plus.google.com/111902568077770766904?rel=author">
     Benedek Izsó</a>
</span>
</div>
<div itemscope itemtype="http://schema.org/Product">
  <span itemprop="name">EMF-IncQuery</span> is a great tool. Rating based on my opinion:
  <div itemprop="aggregateRating" itemscope itemtype="http://schema.org/AggregateRating">
    <span itemprop="ratingValue">5</span> stars -
    based on <span itemprop="reviewCount">1</span> review
  </div>
</div>
</body>
```

Only a product can be rated (online shops)

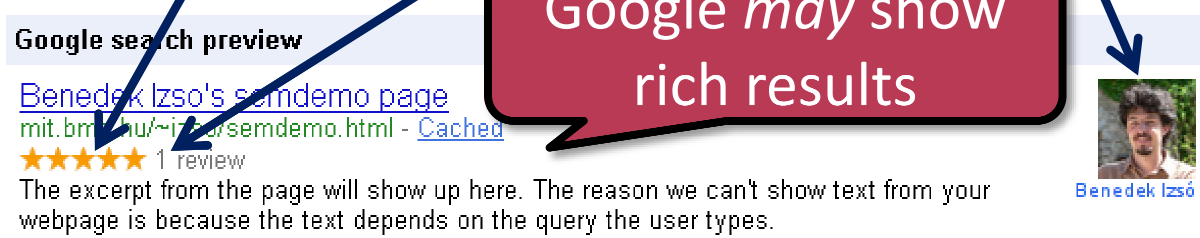
Semantic Web Vocabularies

- **Microdata: schema.org (Google)**

Benedek Izsó
EMF-IncQuery is a great tool. Rating based on my opinion:
5 stars - based on 1 opinion

```
<body>
<div itemscope itemtype="http://schema.org/People">
 <span itemprop="name">
   <a href="https://plus.google.com/111902568077770766904?rel=author">
    Benedek Izsó</a>
</span>
</div>
<div itemscope itemtype="http://schema.org/Product">
  <span itemprop="name">EMF-IncQuery</span> is a great tool. Rating based on my opinion:
  <div itemprop="aggregateRating" itemscope itemtype="http://schema.org/AggregateRating">
    <span itemprop="ratingValue">5</span> stars -
    based on <span itemprop="reviewCount">1</span> review
  </div>
</div>
</body>
```

Google *may* show rich results



Semantic Web Vocabularies

- Common vocabularies: schema.org (G), Open Graph Protocol (FB), Microformats
- Data searching and connecting services **needs data** and not the graphical look
  - A website will not upload its database
  - But may annotate their website content
- One easily usable, unified vocabulary
  - not reused existing ones which can be a big mess
  - Concepts can be the most popular ones, based on site statistics
- Data is more structured with annotations than natural language, but less structured than a DB.

Knowledge Bases

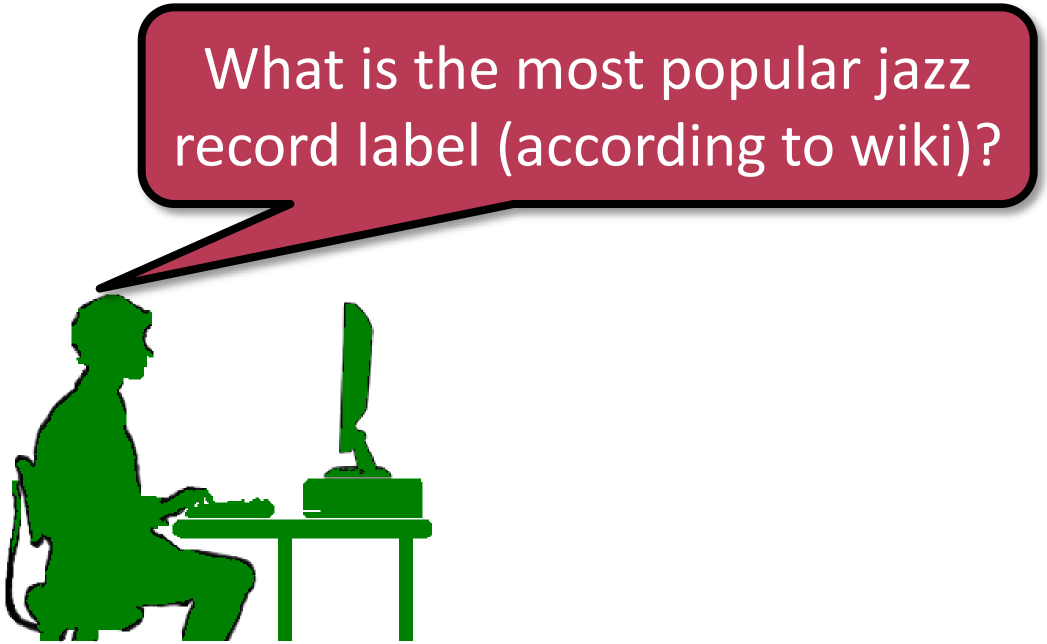
- WordNet: lexical knowledge base of english words
  - Synonym sets (synsets)
  - Semantic relations, including
    - Antonym (opposite)
    - Hypernym, hyponym (super/subtype)
- DBpedia Knowledge Base
  - RDF information
  - Automatically extracted from Wikipedia
  - Manual annotation, links to WordNet etc.
  - SPARQL endpoint

Knowledge Bases

- DBpedia SPARQL Query example with Sesame

Knowledge Bases

- DBpedia SPARQL Query example with Sesame



Knowledge Bases

- DBpedia SPARQL Query example with Sesame

What is the most popular jazz record label (according to wiki)?

Artists have genre and record label on wikipedia, how can I get and count it programmatically?

- DBpedia SPARQL Query example with Sesame

```
SELECT ?label (count(?label) AS ?labelCount)
WHERE {
  ?person dbp:genre dbr:Jazz .
  ?person dbp:label ?label .
  ?label rdf:type dbo:RecordLabel .
 } ORDER BY DESC(?labelCount)
```

Knowledge Bases

- DBpedia SPARQL Query example with Sesame

```
SELECT ?label (count(?label) AS ?labelCount)
WHERE {
  ?person dbp:genre dbr:Jazz .
  ?person dbp:label ?label .
  ?label rdf:type dbo:RecordLabel .
 } ORDER BY DESC(?labelCount)
```

Record Label	Count
Blue_Note_Records	739
ECM_Records	507
Columbia_Records	386
Verve_Records	374

- DBpedia SPARQL Query example with Sesame

```
HTTPRepository sparqlEndpoint = new HTTPRepository("http://live.dbpedia.org/sparql", "");
sparqlEndpoint.initialize();
conn = sparqlEndpoint.getConnection();
String sparqlQuery = " SELECT ?label (count(?label) AS ?labelCount) WHERE { " +
                     "   ?person dbp:genre dbr:Jazz . " +
                     "   ?person dbp:label ?label . " +
                     "   ?label rdf:type dbo:RecordLabel . " +
                     " } ";
TupleQuery query = conn.prepareTupleQuery(QueryLanguage.SPARQL, sparqlQuery);
TupleQueryResult result = query.evaluate();
while (result.hasNext()) {  // print answers
        BindingSet tuple = result.next();
        System.out.println(tuple.getBinding("label").getValue());
        Literal labelCount = (Literal) tuple.getBinding("labelCount").getValue();
        System.out.println(labelCount.intValue());
}
conn.close();
```

Knowledge Bases

- Linked GeoData: OpenStreetMap in RDF format
  - SPARQL endpoint
  - or downloadable RDF (10GB)

Knowledge Bases

- Linked GeoData: OpenStreetMap in RDF format
  - SPARQL endpoint
  - or downloadable RDF (10GB)

Knowledge Bases

- Linked GeoData: OpenStreetMap in RDF format
  - SPARQL endpoint
  - or downloadable RDF (10GB)

Knowledge Bases

- Linked GeoData: OpenStreetMap in RDF format
  - SPARQL endpoint
  - or downloadable RDF (10GB)

```
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
SELECT ?name
FROM <http://linkedgeodata.org> {
   ?poi rdf:type          lgdo:Cafe .
   ?poi rdfs:label        ?name .
   ?poi geo:geometry ?coord .
   filter(bif:st_intersects (?coord,
          bif:st_point (19.061667, 47.489722), 0.4)) .
};
```

Knowledge Bases

- Linked GeoData: OpenStreetMap in RDF format
  - SPARQL endpoint
  - or downloadable RDF (10GB)

```
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
SELECT ?name
FROM <http://linkedgeodata.o
    ?poi rdf:type          lgdo:Cafe
    ?poi rdfs:label        ?name .
    ?poi geo:geometry ?coo
    filter(bif:st_intersects (?coord,
            bif:st_point (19.061667, 47.489722), 0.4)) .
};
```

Filter near places: Intersect function not in SPARQL, LGD extension

Coordinate of Calvin Square

0.4 km

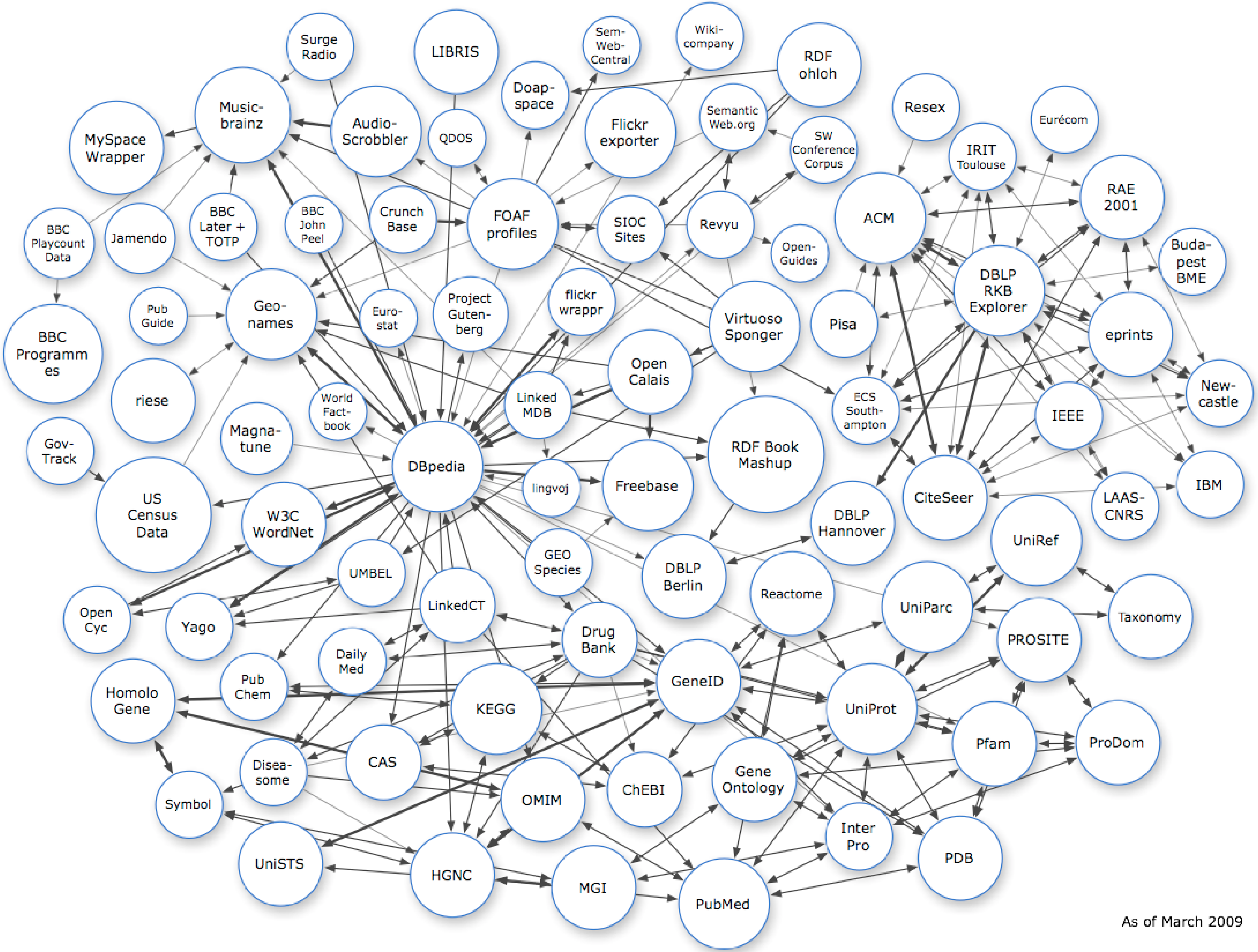
Knowledge Bases

- Linked GeoData: OpenStreetMap in RDF format
  - SPARQL endpoint
  - or downloadable RDF (10GB)

```
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
SELECT ?name
FROM <http://linkedgeodata.org> {
    ?poi rdf:type          lgdo:Cafe .
    ?poi rdfs:label        ?name .
    ?poi geo:geometry ?coord .
    filter(bif:st_intersects (?coord,
              bif:st_point (19.061667, 47.489722), 0.4)) .
};
```

Cafes near Calvin Square (?name)
Frank Zappa
LUMEN
California Coffee Bean

Linking Open Data (LOD)



MODELING APPROACHES

DSM and Ontology languages - Similarities

	Domain Specific Models	Ontology (DL)
Language	UML, **EMF**	RDF, OWL
1-ary predicate	EClass	Concept
2-ary predicate	EReference/EAttribute	Object/Datatype property
Model element	EObject	Instance
Enumeration	EEnum	Nominals
Cardinality	Role cardinality	Cardinality restriction
Inverse	EOpposite	owl:inverseOf
Type of	EType	rdf:type
Subsumption	Generalizaton	rdfs:subClassOf

DSM and Ontology languages - Differences

	Domain Specific Models	Ontology (DL)
Classifying instances	Enum, multiple inheritance	Multi domain metamodeling
Constraint description	Other auxiliary language (e.g.: OCL)	Built into the language
Types	One instance has 1 type	One instance can belong to multiple classes
Unique Name Assumption	Yes	No (by default)
Open World Assumption	No	Yes

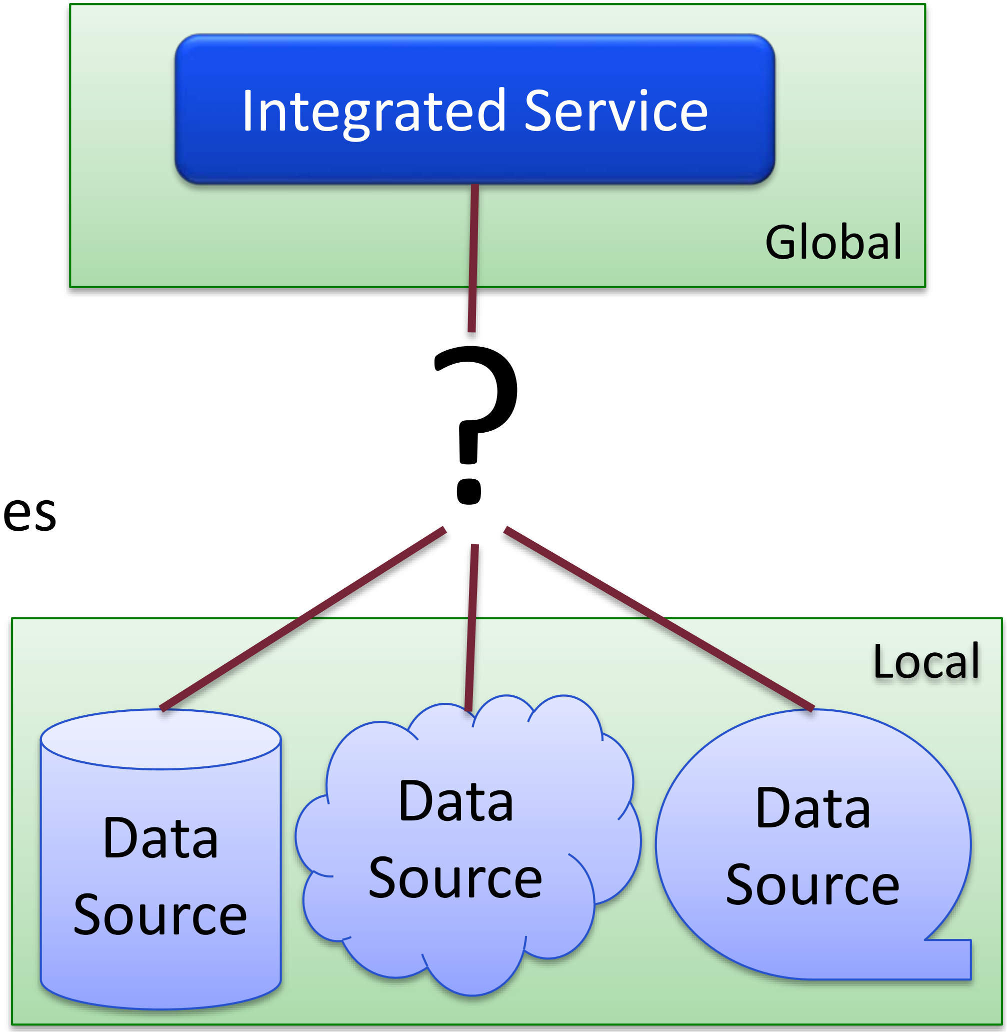
DSM and Ontology based modeling - Usage

	Domain Specific Models	Ontology (DL)
Metamodel description	Practical description -for engineers	Precise definitions - for mathematicians
Instance model	Complete data	Continuously evolving data
Strengths	• Code generation • Model transformation • Existing tools • Fast instance model query	• Reasoning • Consistency checking • Common vocabularies • Prototyping (OWA, UNA)
Modeling	Efficient instance model editing and querying	Efficient metamodel designing and reasoning
Language elements	Common OO terms	Mathematical constraints (e.g.: existential and universal quantification)
Structuredness	Highly structured	Usually not structured

SEMANTIC INTEGRATION

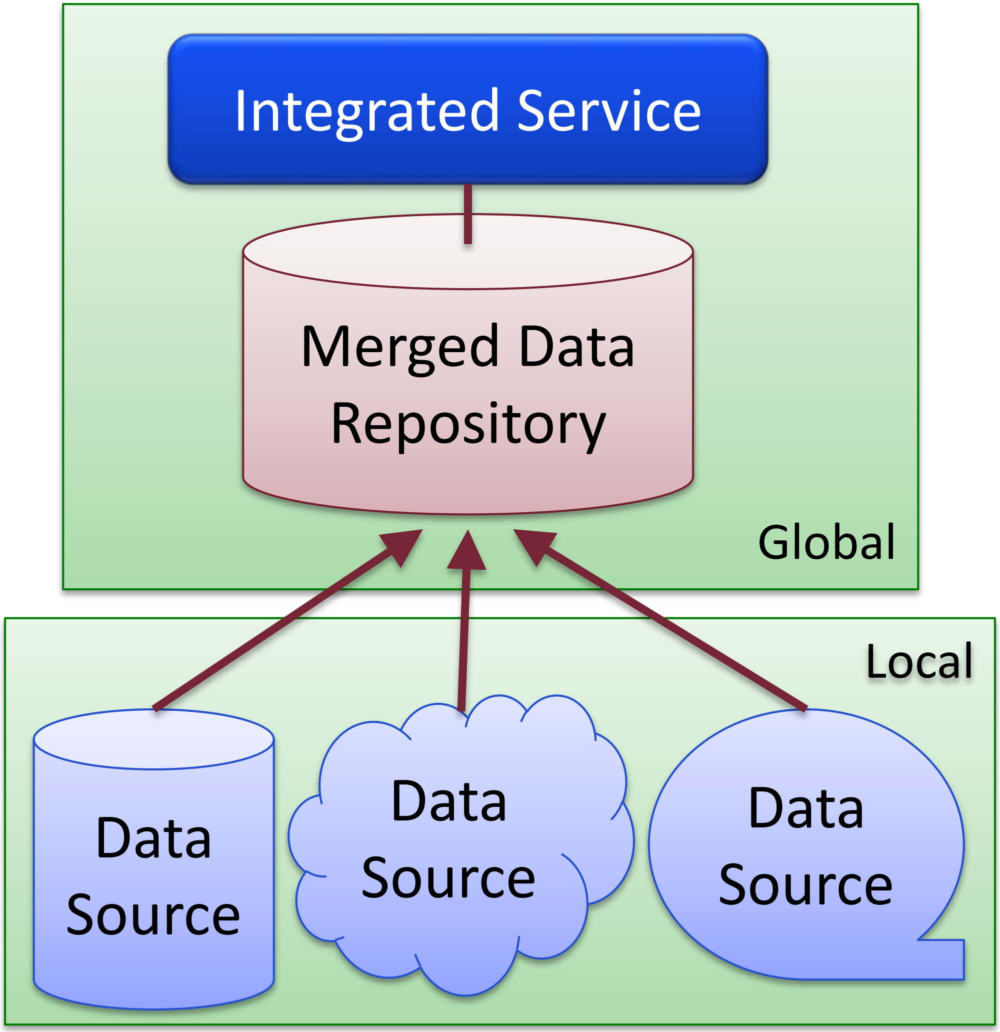
Data Integration

- Distributed, heterogenous sources
  - Relational DB
  - Web Service
  - Ad-hoc interfaces
- Unified global service
  - Usually query-only
  - Global query vs. local sources
- Terminology
  - Data Integration
  - Information Integration
  - Data Fusion



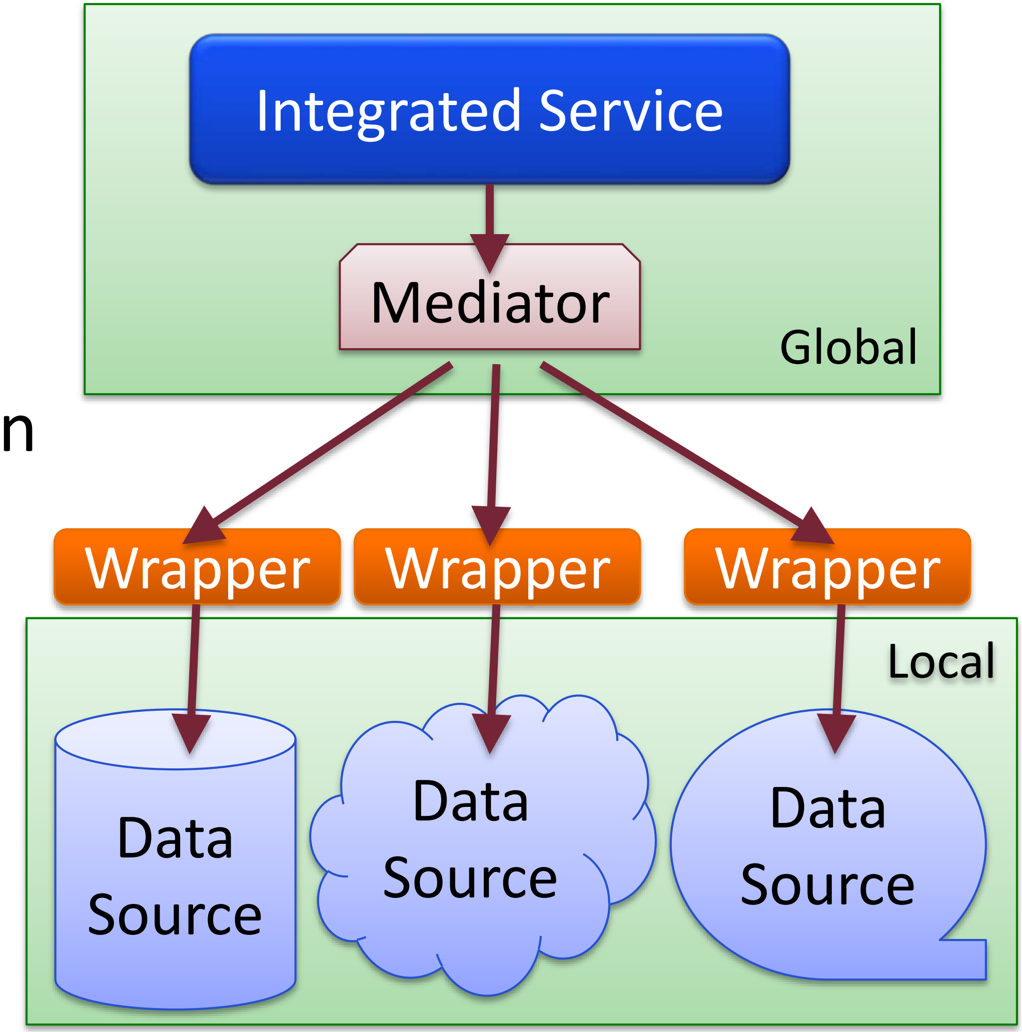
Merging (Data Warehousing)

- Solutions
  - **Merging (warehousing)**
  - Mediation (federated DB)
- Single central repository
  - Contains all merged data
- Straightforward
- Drawbacks
  - Merge cost
  - Outdated
  - Unless regularly refreshed
  - Maintainability issues



Mediation (Database Federation)

- Solutions
  - Merging (warehousing)
  - **Mediation (federated DB)**
- Query propagation
  - Result composition
- Problem: query translation
  - GaV
  - LaV
- Advantages
  - Always up-to-date
  - Lightweight



Mediation: Global as View

- Problem: query translation
  - **GaV (Global as View)**
  - LaV
- The global schema expressed as a view on locals
  - Transformations, projections, unions, joins
  - E.g. ‘ACME’$\times(\pi_{Name,Price}$AcmeProducts$)\cup$…
- Query execution: simple view evaluation
- The same basic design as in the merging case
  - Unmaintainable with too many sources

Mediation: Local as View

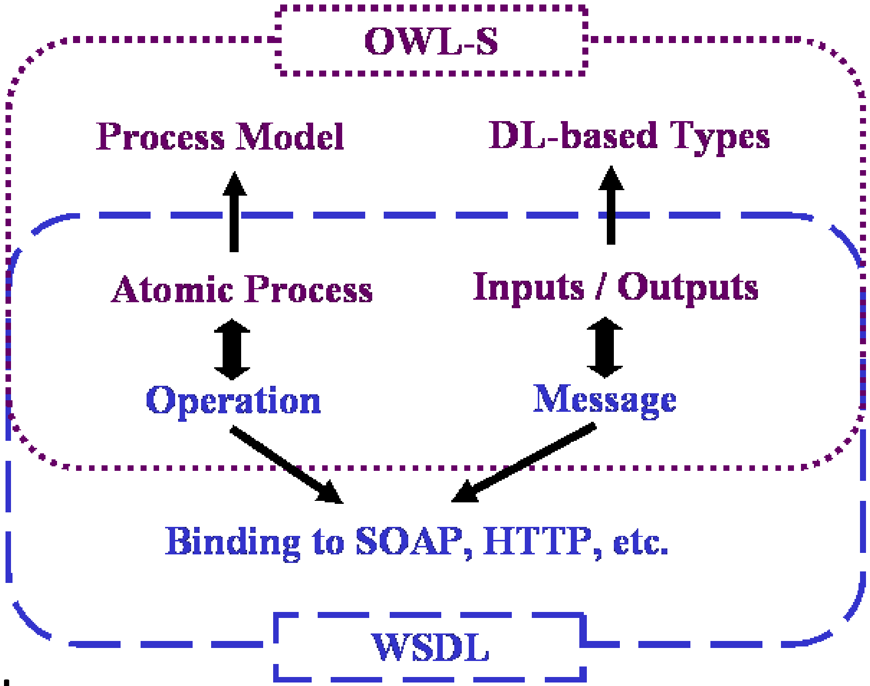
- Problem: query translation
  - GaV
  - **LaV (Local as View)**
- Each local schema as a view of the global
  - Projections, selections
  - E.g. only product data, only for ACME boxes
- Easier to add new sources
- Query execution
  - theory of federated databases
  - „answering queries using views”

Semantic Integration

- Heterogeneity
  - Non-relational sources
  - Different vocabulary and semantics
  - Different structure
  - Different representations
- Ontology-based Semantic Data Integration
  - Local scheme explained with linked ontologies
  - Semantic mapping between schemes
  - Query formulation based on ontology
  - Execution: automatic reasoning?

Semantic Service Integration

- Standards such as WSDL define the syntax
  - Currency? Unit of measurement?
  - The identifier of *what*?
  - Preconditions?
- Semantic Web Services
  - OWL-S
    - Process Model, DL types
  - WSMO
    - Goals of clients, mediators, etc.
- Dream: Semantic Service Discovery



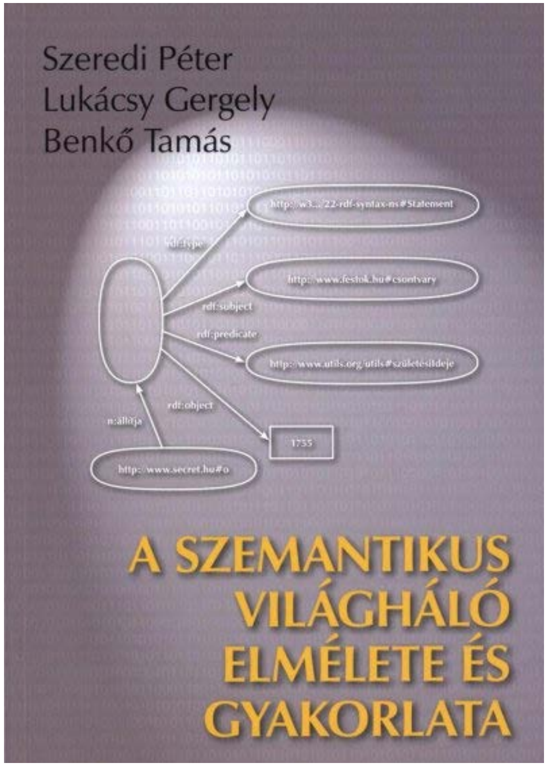
SUMMARY

Summary

- Semantic technologies
  - Metadata (RDF)
  - Ontologies (OWL)
  - Formal logic based reasoning
  - Querying with SPARQL
- Applications
  - Domain ontologies in expert systems
  - Semantic Web
  - Modeling Approaches
  - Semantic Integration

Recommended reading

Benkő-Szeredi-Lukácsy:
A szemantikus világháló elmélete és gyakorlata.
Typotex, 2005.



BMEVIMIM222
Információ- és tudásintegrálás
(MSc intelligens rendszerek szakirány)