Semantic Web Data/RDF/SPARQL
<table>
<thead>
<tr>
<th>Relational</th>
<th>Semantic Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Tables</td>
<td><strong>RDF Graphs</strong></td>
</tr>
<tr>
<td>SQL</td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Tables</td>
<td><em>RDF Graphs</em></td>
</tr>
<tr>
<td>SQL</td>
<td><em>SPARQL</em></td>
</tr>
<tr>
<td>Relational</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Tables</td>
<td>RDF Graphs</td>
</tr>
<tr>
<td>SQL</td>
<td>SPARQL</td>
</tr>
<tr>
<td>Closed Data</td>
<td></td>
</tr>
<tr>
<td>(inside an organization)</td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Tables</td>
<td>RDF Graphs</td>
</tr>
<tr>
<td>SQL</td>
<td>SPARQL</td>
</tr>
<tr>
<td>Closed Data</td>
<td>Open Data</td>
</tr>
<tr>
<td>(inside an organization)</td>
<td>(available on the Web)</td>
</tr>
</tbody>
</table>
Semantic Web

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.”

[Tim Berners-Lee et al. 2001.]

Specific Goals:

► Build a description language with standard semantics
► Make semantics machine-processable and understandable
► Incorporate logical infrastructure to reason about resources
► W3C Proposal: Resource Description Framework (RDF)
RDF in a nutshell

- RDF is the W3C proposal framework for representing information in the Web
- Abstract syntax based on directed labeled graph
- Schema definition language (RDFS): Define new vocabulary (typing, inheritance of classes and properties)
- Extensible URI-based vocabulary
- Formal semantics
RDF formal model

\[ U \] = set of UrIs
\[ B \] = set of Blank nodes
\[ L \] = set of Literals
RDF formal model

\[ (s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L) \] is called an RDF triple.

\[ U = \text{set of } UrIs \]
\[ B = \text{set of Blank nodes} \]
\[ L = \text{set of Literals} \]
RDF formal model

\[(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)\] is called an RDF triple

A set of RDF triples is called an RDF graph

\[U = \text{set of Uris}\]
\[B = \text{set of Blank nodes}\]
\[L = \text{set of Literals}\]
An example of an RDF graph: DBLP

: <http://dblp.l3s.de/d2r/resource/authors/>
conf: <http://dblp.l3s.de/d2r/resource/conferences/>
inPods: <http://dblp.l3s.de/d2r/resource/publications/conf/pods/>
swrc: <http://swrc.ontoware.org/ontology#>
dc: <http://purl.org/dc/elements/1.1/>
dct: <http://purl.org/dc/terms/>
An example of a URI

http://dblp.l3s.de/d2r/resource/conferences/pods
URI can be used for any abstract resource

http://dblp.l3s.de/d2r/page/authors/Ronald_Fagin
RDF: Another example

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```

```
RDF: Another example

```
Some peculiarities of the RDF data model

- *Existential variables* as datavalues (null values)
- Built-in vocabulary with fixed semantics (RDFS)
- Graph model where nodes may also be edge labels
Previous example: A better representation

Messi works in person sportman soccer player

Barcelona plays in soccer team

Spain: lives in country address

_:b

country

person works in works_in company

rdf:sp rdf:sc rdf:dom rdf:range

rdf:sp rdf:sc rdf:dom rdf:range

rdf:sp rdf:sc rdf:dom rdf:range

rdf:sp rdf:sc rdf:dom rdf:range
RDF + RDFS

RDFS extends RDF with a schema vocabulary: subPropertyOf (\texttt{rdf:sp}), subClassOf (\texttt{rdf:sc}), domain (\texttt{rdf:dom}), range (\texttt{rdf:range}), type (\texttt{rdf:type}).

plus \textit{semantics} for this vocabulary
RDFS: Messi is a Person

Messi is a person. He works in a company as a sportman. Messi is a soccer player who plays in a soccer team. Messi lives in Spain and works in Barcelona.
Semantics of RDFS

Checking whether a triple $t$ is in a graph $G$ is the basic step when reasoning about RDF(S).

- For the case of RDFS, we need to check whether $t$ is implied by $G$. 
Semantics of RDFS

Checking whether a triple \( t \) is in a graph \( G \) is the basic step when reasoning about RDF(S).

- For the case of RDFS, we need to check whether \( t \) is implied by \( G \).

The notion of entailment in RDFS can be defined in terms of classical notions such as model, interpretation, etc.

- As for the case of first-order logic.
Semantics of RDFS

Checking whether a triple $t$ is in a graph $G$ is the basic step when reasoning about RDF(S).

- For the case of RDFS, we need to check whether $t$ is implied by $G$

The notion of entailment in RDFS can be defined in terms of classical notions such as model, interpretation, etc.

- As for the case of first-order logic

This notion can also be characterized by a set of inference rules.
Semantics of RDFS

Checking whether a triple $t$ is in a graph $G$ is the basic step when reasoning about RDF(S).

- For the case of RDFS, we need to check whether $t$ is implied by $G$.

The notion of entailment in RDFS can be defined in terms of classical notions such as model, interpretation, etc.

- As for the case of first-order logic.

This notion can also be characterized by a set of inference rules.

The closure of an RDFS graph $G$ ($\text{cl}(G)$) is the graph obtained by adding to $G$ all the triples that are implied by $G$.

A basic property of the closure:

- $G$ implies $t$ iff $t \in \text{cl}(G)$.
Example: \((Messi, \text{ rdf:type, person})\) over the closure
Does the blank node add some information?

```
Does the blank node add some information?

person rdfs:domain works_in rdfs:range company
  rdfs:subject
  sportman rdfs:subject
  soccer_player rdfs:domain plays_in rdfs:range soccer_team
    rdfs:domain
  Messi rdfs:cell
    rdfs:domain
      address
    rdfs:cell
     _:b
        country Spain
          lives_in
            :b
              employer
                works_in
                  employer rdfs:range company
                    rdfs:domain
```

What about now?
SPARQL
SPARQL is the W3C recommendation query language for RDF (January 2008).

> SPARQL is a recursive acronym that stands for **SPARQL Protocol and RDF Query Language**

SPARQL is a graph-matching query language.

A SPARQL query consists of three parts:

- Pattern matching: optional, union, filtering, . . .
- Solution modifiers: projection, distinct, order, limit, offset, . . .
- Output part: construction of new triples, . . .
Example: Authors that have published in PODS
Example: Authors that have published in PODS

SELECT ?Author
Example: Authors that have published in PODS

```
SELECT ?Author
WHERE
{
}
```
Example: Authors that have published in PODS

```
SELECT ?Author
WHERE
{
}
```
Example: Authors that have published in PODS

```
SELECT ?Author
WHERE {
}
```
Example: Authors that have published in PODS

```sparql
SELECT ?Author
WHERE
{
}
```
SPARQL: A Simple RDF Query Language

Example: Authors that have published in PODS

```
SELECT ?Author
WHERE
{
}
```

A SPARQL query consists of a:
Example: Authors that have published in PODS

```
SELECT ?Author
WHERE
{
}
```

A SPARQL query consists of a:

**Head:** Processing of the variables
Example: Authors that have published in PODS

```
SELECT ?Author
WHERE
{
}
```

A SPARQL query consists of a:

Head: Processing of the variables

Body: Pattern matching expression
Example: Authors that have published in PODS, and their Web pages if this information is available:

```
SELECT ?Author ?WebPage
WHERE
{

  OPTIONAL {
  }
}
```
Example: Authors that have published in PODS, and their Web pages if this information is available:

```
SELECT ?Author ?WebPage
WHERE
{

  OPTIONAL {
  }
}
```
But things can become more complex...

Interesting features of pattern matching on graphs

```
SELECT ?X1 ?X2 ...
{ P1 .
  P2 }
```
But things can become more complex...

Interesting features of pattern matching on graphs

► Grouping

```
SELECT ?X1 ?X2 ...
{{ P1 .
P2 }

{ P3 .
P4 }

}
```
But things can become more complex...

Interesting features of pattern matching on graphs

- Grouping
- Optional parts

```
SELECT ?X1 ?X2 ...
{{ P1 .
P2
OPTIONAL { P5 } }

{ P3 .
P4
OPTIONAL { P7 } }
}
```
But things can become more complex...

Interesting features of pattern matching on graphs

- Grouping
- Optional parts
- Nesting

```
SELECT ?X1 ?X2 ...
{{ P1 .
P2
OPTIONAL { P5 } }
{ P3 .
P4
OPTIONAL { P7
OPTIONAL { P8 } } }
}
```
But things can become more complex...

Interesting features of pattern matching on graphs

- Grouping
- Optional parts
- Nesting
- Union of patterns

```
SELECT ?X1 ?X2 ...
{{
  P1
  P2
  OPTIONAL { P5 }
}

{ P3
  P4
  OPTIONAL { P7
    OPTIONAL { P8 }
  }
}
}
UNION
{ P9 }
```
But things can become more complex...

Interesting features of pattern matching on graphs

- Grouping
- Optional parts
- Nesting
- Union of patterns
- Filtering

```sql
SELECT ?X1 ?X2 ...
{{
  {{
    P1 .
    P2
    OPTIONAL { P5 } }
  }

  { P3 .
    P4
    OPTIONAL { P7
      OPTIONAL { P8 } } }
}

UNION
{ P9
  FILTER ( R ) }
}}
```
But things can become more complex...

Interesting features of pattern matching on graphs

- Grouping
- Optional parts
- Nesting
- Union of patterns
- Filtering
- ...

+ several new features in the new version (March 2013): navigation, entailment regimes, federation, ...

```
SELECT ?X1 ?X2 ...
{{{ P1 .
  P2
  OPTIONAL { P5 } }

  { P3 .
    P4
    OPTIONAL { P7
      OPTIONAL { P8 } }
  }
}
UNION
{ P9
  FILTER ( R ) }
```
But things can become more complex...

Interesting features of pattern matching on graphs

- Grouping
- Optional parts
- Nesting
- Union of patterns
- Filtering
- ...

+ several new features in the new version (March 2013): navigation, entailment regimes, federation, ...

```
SELECT ?X1 ?X2 ...
{ {{ P1 .
    P2
    OPTIONAL { P5 } }

    { P3 .
    P4
    OPTIONAL { P7
    OPTIONAL { P8 } } }

    } }

UNION

{ P9
  FILTER ( R ) }
```

What is the (formal) *meaning* of a general SPARQL query?
SPARQL: An algebraic syntax

\( \mathcal{V} \): set of variables

Each variable is assumed to start with question mark (?)
SPARQL: An algebraic syntax

\( V \): set of variables

Each variable is assumed to start with ?

**Triple pattern:** \( t \in (U \cup V) \times (U \cup V) \times (U \cup L \cup V) \)

Examples: (?,name,john), (?,name,?Y)
SPARQL: An algebraic syntax

\( \mathbb{V} \): set of variables
Each variable is assumed to start with \(?\)

**Triple pattern:** \( t \in (U \cup V) \times (U \cup V) \times (U \cup L \cup V) \)
Examples: \((?X, \text{name}, \text{john})\), \((?X, \text{name}, ?Y)\)

**Basic graph pattern (bgp):** Finite set of triple patterns
Examples: \(\{(?X, \text{knows}, ?Y), (?Y, \text{name}, \text{john})\}\)
Recursive definition of SPARQL graph patterns:

- Every basic graph pattern is a graph pattern
- If $P_1$, $P_2$ are graph patterns, then $(P_1 \text{ AND } P_2)$, $(P_1 \text{ OPT } P_2)$, $(P_1 \text{ UNION } P_2)$ are graph pattern
- If $P$ is a graph pattern and $R$ is a built-in condition, then $(P \text{ FILTER } R)$ is a graph pattern

SPARQL query:

- If $P$ is a graph pattern and $W$ is a finite set of variables, then $(\text{SELECT } W P)$ is a SPARQL query
Standard versus algebraic notation

?X : name "john"  

(?X, name, john)
Standard versus algebraic notation

?X :name "john"

{ P1 . P2 }
Standard versus algebraic notation

?X :name "john"  

{ P1 . P2 }  

{ P1 OPTIONAL { P2 } }  

(?X, name, john)  

(P₁ AND P₂)  

(P₁ OPT P₂)
Standard versus algebraic notation

\(?X : \text{name} \ "\text{john}" \)

\(\{ \ P1 \ . \ P2 \ \} \)

\(\{ \ P1 \ \text{OPTIONAL} \ \{ \ P2 \ \}\} \)

\(\{ \ P1 \ \text{UNION} \ \{ \ P2 \ \}\} \)
## Standard versus algebraic notation

<table>
<thead>
<tr>
<th>Standard Notation</th>
<th>Algebraic Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>?X :name &quot;john&quot;</code></td>
<td><code>(?X, name, john)</code></td>
</tr>
<tr>
<td><code>{ P1 . P2 }</code></td>
<td><code>(P₁ AND P₂)</code></td>
</tr>
<tr>
<td><code>{ P1 OPTIONAL { P2 }}</code></td>
<td><code>(P₁ OPT P₂)</code></td>
</tr>
<tr>
<td><code>{ P1 } UNION { P2 }</code></td>
<td><code>(P₁ UNION P₂)</code></td>
</tr>
<tr>
<td><code>{ P1 FILTER ( R ) }</code></td>
<td><code>(P₁ FILTER R)</code></td>
</tr>
</tbody>
</table>
Standard versus algebraic notation

?-X :name "john"

{ P1 . P2 }

{ P1 OPTIONAL { P2 } }

{ P1 } UNION { P2 }

{ P1 FILTER ( R ) }

SELECT W WHERE { P }
Mappings: building block for the semantics

Definition

A mapping is a partial function:

$$\mu : V \rightarrow (U \cup L \cup B)$$
Mappings: building block for the semantics

Definition
A mapping is a partial function:

\[ \mu : V \rightarrow (U \cup L \cup B) \]

Given a mapping \( \mu \) and a triple pattern \( t \):
Mappings: building block for the semantics

Definition

A mapping is a partial function:

$$\mu : V \rightarrow (U \cup L \cup B)$$

Given a mapping $\mu$ and a triple pattern $t$:

$\mu(t)$: triple obtained from $t$ replacing variables according to $\mu$
Mappings: building block for the semantics

Definition
A mapping is a partial function:

\[ \mu : V \rightarrow (U \cup L \cup B) \]

Given a mapping \( \mu \) and a triple pattern \( t \):
- \( \mu(t) \): triple obtained from \( t \) replacing variables according to \( \mu \)

Example
Mappings: building block for the semantics

Definition
A mapping is a partial function:

\[ \mu : V \rightarrow (U \cup L \cup B) \]

Given a mapping \( \mu \) and a triple pattern \( t \):

\( \mu(t) \): triple obtained from \( t \) replacing variables according to \( \mu \)

Example

\[ \mu = \{ ?X \rightarrow R_1, ?Y \rightarrow R_2, ?Z \rightarrow \text{john} \} \]
Mappings: building block for the semantics

Definition
A mapping is a partial function:

$$\mu : V \rightarrow (U \cup L \cup B)$$

Given a mapping $\mu$ and a triple pattern $t$:

$\mu(t)$: triple obtained from $t$ replacing variables according to $\mu$

Example

$$\mu = \{ ?X \rightarrow R_1, ?Y \rightarrow R_2, ?Z \rightarrow john \}$$

$$t = (?X, name, ?Z)$$
Mappings: building block for the semantics

Definition
A mapping is a partial function:

\[ \mu : V \rightarrow (U \cup L \cup B) \]

Given a mapping \( \mu \) and a triple pattern \( t \):

- \( \mu(t) \): triple obtained from \( t \) replacing variables according to \( \mu \)

Example

\[ \mu = \{ ?X \rightarrow R_1, ?Y \rightarrow R_2, ?Z \rightarrow john \} \]

\[ t = ( ?X, \text{name}, ?Z ) \]

\[ \mu(t) = ( R_1, \text{name}, john ) \]
The semantics of triple patterns

Definition

The evaluation of triple pattern $t$ over a graph $G$, denoted by $\llbracket t \rrbracket_G$, is the set of all mappings $\mu$ such that:
The semantics of triple patterns

Definition

The evaluation of triple pattern $t$ over a graph $G$, denoted by $\llbracket t \rrbracket_G$, is the set of all mappings $\mu$ such that:

- $\text{dom}(\mu)$ is exactly the set of variables occurring in $t$
The semantics of triple patterns

Definition

The evaluation of triple pattern $t$ over a graph $G$, denoted by $[t]_G$, is the set of all mappings $\mu$ such that:

- $\text{dom}(\mu)$ is exactly the set of variables occurring in $t$
- $\mu(t) \in G$
Example

\[ G \\
(R_1, \text{name, john}) \\
(R_1, \text{email, J@ed.ex}) \\
(R_2, \text{name, paul}) \]

\[ [(?X, \text{name, ?N})]_G \]
Example

$G$

$(R_1, \text{name, john})$

$(R_1, \text{email, J@ed.ex})$

$(R_2, \text{name, paul})$

$\llbracket (?X, \text{name, ?N}) \rrbracket_G$

\[
\begin{align*}
\mu_1 &= \{ ?X \to R_1, ?N \to \text{john} \} \\
\mu_2 &= \{ ?X \to R_2, ?N \to \text{paul} \}
\end{align*}
\]
Example

\[ G \]
\[ (R_1, \text{name, john}) \]
\[ (R_1, \text{email, J@ed.ex}) \]
\[ (R_2, \text{name, paul}) \]

\[ \llbracket (?X, \text{name, ?N}) \rrbracket_G \]
\[
\mu_1 = \{ ?X \rightarrow R_1, ?N \rightarrow \text{john} \} \\
\mu_2 = \{ ?X \rightarrow R_2, ?N \rightarrow \text{paul} \} \\
\]

\[ \llbracket (?X, \text{email, ?E}) \rrbracket_G \]
Example

\[ G \]
\((R_1, \text{name, john})\)
\((R_1, \text{email, J@ed.ex})\)
\((R_2, \text{name, paul})\)

\[ \llbracket (?X, \text{name, ?N}) \rrbracket_G \]
\( \mu_1 = \{ ?X \to R_1, ?N \to \text{john} \} \)
\( \mu_2 = \{ ?X \to R_2, ?N \to \text{paul} \} \)

\[ \llbracket (?X, \text{email, ?E}) \rrbracket_G \]
\( \mu = \{ ?X \to R_1, ?E \to \text{J@ed.ex} \} \)
Example

\[
G
\]

\[
(R_1, \text{name}, \text{john})
\]

\[
(R_1, \text{email}, \text{J@ed.ex})
\]

\[
(R_2, \text{name}, \text{paul})
\]

\[
\llbracket (?X, \text{name}, ?N) \rrbracket_G
\]

\[
\begin{array}{|c|c|}
\hline
?X & ?N \\
\hline
R_1 & \text{john} \\
R_2 & \text{paul} \\
\hline
\end{array}
\]

\[
\llbracket (?X, \text{email}, ?E) \rrbracket_G
\]

\[
\begin{array}{|c|c|}
\hline
?X & ?E \\
\hline
R_1 & \text{J@ed.ex} \\
\hline
\end{array}
\]
Example

\[ G \]

\( (R_1, \text{name, john}) \)
\( (R_1, \text{email, J@ed.ex}) \)
\( (R_2, \text{name, paul}) \)

\[ [(R_1, \text{webPage, ?W})]_G \]

\[ [(R_3, \text{name, ringo})]_G \]

\[ [(R_2, \text{name, paul})]_G \]
Example

\[ G \\
(R_1, \text{name}, \text{john}) \\
(R_1, \text{email}, \text{J@ed.ex}) \\
(R_2, \text{name}, \text{paul}) \]

\[[\mathcal{I} (R_1, \text{webPage}, ?W)]_G \]

\{ \}

\[[\mathcal{I} (R_2, \text{name}, \text{paul})]_G \]

\[[\mathcal{I} (R_3, \text{name}, \text{ringo})]_G \]
Example

\[
G = (R_1, \text{name, john}) \\
(R_1, \text{email, J@ed.ex}) \\
(R_2, \text{name, paul})
\]

\[
[(R_1, \text{webPage, ?W})]_G \\
\{ \} \\
[(R_2, \text{name, paul})]_G
\]

\[
[(R_3, \text{name, ringo})]_G \\
\{ \} \\
\]
Example

\[ G \]

\((R_1, \text{name, john})\)
\((R_1, \text{email, J@ed.ex})\)
\((R_2, \text{name, paul})\)

\[\llbracket (R_1, \text{webPage, ?W}) \rrbracket_G\]
\[
\{ \}
\]

\[\llbracket (R_3, \text{name, ringo}) \rrbracket_G\]
\[
\{ \}
\]

\[\llbracket (R_2, \text{name, paul}) \rrbracket_G\]
\[
\{ \mu_\emptyset = \{ \} \}
\]
Semantics of SPARQL: Basic graph patterns

Let $P$ be a basic graph pattern

- $\text{var}(P)$: set of variables mentioned in $P$
Semantics of SPARQL: Basic graph patterns

Let $P$ be a basic graph pattern

- $\text{var}(P)$: set of variables mentioned in $P$

Given a mapping $\mu$ such that $\text{var}(P) \subseteq \text{dom}(\mu)$:

$$\mu(P) = \{\mu(t) \mid t \in P\}$$
Semantics of SPARQL: Basic graph patterns

Let $P$ be a basic graph pattern

- $\text{var}(P)$: set of variables mentioned in $P$

Given a mapping $\mu$ such that $\text{var}(P) \subseteq \text{dom}(\mu)$:

$$\mu(P) = \{\mu(t) \mid t \in P\}$$

**Definition**

The evaluation of $P$ over an RDF graph $G$, denoted by $[P]_G$, is the set of mappings $\mu$:

- $\text{dom}(\mu) = \text{var}(P)$
- $\mu(P) \subseteq G$
Semantics of basic graph patterns: An example

<table>
<thead>
<tr>
<th>graph</th>
<th>bgp</th>
<th>evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>((R_1, \text{name, john}))</td>
<td>{(?X, \text{name, ?Y}), (?X, \text{email, ?Z})}</td>
<td></td>
</tr>
<tr>
<td>((R_1, \text{email, <a href="mailto:J@ed.ex">J@ed.ex</a>}))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((R_2, \text{name, paul}))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Semantics of basic graph patterns: An example

<table>
<thead>
<tr>
<th>graph</th>
<th>bgp</th>
<th>evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(R_1, \text{name, john})$</td>
<td>${(?X, \text{name, ?Y}),$</td>
<td></td>
</tr>
<tr>
<td>$(R_1, \text{email, <a href="mailto:J@ed.ex">J@ed.ex</a>})$</td>
<td>$\text{(?X, email, ?Z)}}$</td>
<td></td>
</tr>
<tr>
<td>$(R_2, \text{name, paul})$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Semantics of basic graph patterns: An example

<table>
<thead>
<tr>
<th>graph</th>
<th>bgp</th>
<th>evaluation</th>
</tr>
</thead>
</table>
| \((R_1, \text{name, john})\) | \{(?X, \text{name, ?Y}), \?
| \((R_1, \text{email, J@ed.ex})\) | \(\text{(?X, email, ?Z)}\) \} |
Semantics of basic graph patterns: An example

<table>
<thead>
<tr>
<th>graph</th>
<th>bgp</th>
<th>evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1, \text{name, john} )</td>
<td>{(?X, \text{name, ?Y}), (?X, \text{email, ?Z})}</td>
<td>( \mu: R_1 )</td>
</tr>
<tr>
<td>( R_1, \text{email, <a href="mailto:J@ed.ex">J@ed.ex</a>} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( R_2, \text{name, paul} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Semantics of basic graph patterns: An example

<table>
<thead>
<tr>
<th>graph</th>
<th>bgp</th>
<th>evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(R_1, \text{name, john})$</td>
<td>${(?X, \text{name, } ?Y), (\mu: R_1, \text{john}, <a href="mailto:J@ed.ex">J@ed.ex</a>)}$</td>
<td>$\mu: R_1, \text{john}, <a href="mailto:J@ed.ex">J@ed.ex</a>$</td>
</tr>
</tbody>
</table>
| $(R_1, \text{email, J@ed.ex})$ | \n
**Notation**

$t$ is used to represent $\{t\}$
Compatible mappings: mappings that can be merged

Definition

Mappings $\mu_1$ and $\mu_2$ are compatible if they agree in their common variables:

If $\ ?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, then $\mu_1(\ ?X) = \mu_2(\ ?X)$
Compatible mappings: mappings that can be merged

**Definition**

Mappings $\mu_1$ and $\mu_2$ are compatible if they agree in their common variables:

If $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, then $\mu_1(?X) = \mu_2(?X)$

**Example**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_1$</td>
<td>$R_1$</td>
<td>john</td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
<td>$R_2$</td>
</tr>
<tr>
<td>$\mu_2$</td>
<td>$R_1$</td>
<td></td>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
<td></td>
</tr>
<tr>
<td>$\mu_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Compatible mappings: mappings that can be merged

**Definition**

Mappings $\mu_1$ and $\mu_2$ are compatible if they agree in their common variables:

\[
\text{If } ?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2), \text{ then } \mu_1(?X) = \mu_2(?X)
\]

**Example**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>john</td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
<td>$R_2$</td>
</tr>
<tr>
<td>$R_1$</td>
<td></td>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
<td></td>
</tr>
</tbody>
</table>
Compatible mappings: mappings that can be merged

Definition

Mappings $\mu_1$ and $\mu_2$ are compatible if they agree in their common variables:

If $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, then $\mu_1(?X) = \mu_2(?X)$

Example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>john</td>
<td>$<a href="mailto:J@edu.ex">J@edu.ex</a>$</td>
<td></td>
</tr>
<tr>
<td>$R_1$</td>
<td>$<a href="mailto:P@edu.ex">P@edu.ex</a>$</td>
<td></td>
<td>$R_2$</td>
</tr>
<tr>
<td>$R_1$</td>
<td>john</td>
<td>$<a href="mailto:J@edu.ex">J@edu.ex</a>$</td>
<td></td>
</tr>
</tbody>
</table>
Compatible mappings: mappings that can be merged

**Definition**

Mappings $\mu_1$ and $\mu_2$ are compatible if they agree in their common variables:

If $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, then $\mu_1(?X) = \mu_2(?X)$

**Example**

<table>
<thead>
<tr>
<th></th>
<th>$\mu_1$</th>
<th>$\mu_2$</th>
<th>$\mu_3$</th>
<th>$\mu_1 \cup \mu_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$?X$</td>
<td>$R_1$</td>
<td>$R_1$</td>
<td>$R_1$</td>
<td>$R_1$</td>
</tr>
<tr>
<td>$?Y$</td>
<td>john</td>
<td>john</td>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
<td>john</td>
</tr>
<tr>
<td>$?Z$</td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
<td></td>
</tr>
<tr>
<td>$?V$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Compatible mappings: mappings that can be merged

**Definition**
Mappings $\mu_1$ and $\mu_2$ are compatible if they agree in their common variables:

If $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, then $\mu_1(?X) = \mu_2(?X)$

**Example**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_1 :$</td>
<td>$R_1$</td>
<td>john</td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
</tr>
<tr>
<td>$\mu_2 :$</td>
<td>$R_1$</td>
<td>john</td>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
</tr>
<tr>
<td>$\mu_3 :$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mu_1 \cup \mu_2 :$</td>
<td>$R_1$</td>
<td>john</td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
</tr>
<tr>
<td>$\mu_1 \cup \mu_3 :$</td>
<td>$R_1$</td>
<td>john</td>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
</tr>
</tbody>
</table>
Compatible mappings: mappings that can be merged

Definition
Mappings $\mu_1$ and $\mu_2$ are compatible if they agree in their common variables:

If $\, ?X \in \operatorname{dom}(\mu_1) \cap \operatorname{dom}(\mu_2)$, then $\mu_1(\, ?X) = \mu_2(\, ?X)$

Example

<table>
<thead>
<tr>
<th>$\mu_1$</th>
<th>$\mu_2$</th>
<th>$\mu_3$</th>
<th>$\mu_1 \cup \mu_2$</th>
<th>$\mu_1 \cup \mu_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$R_1$</td>
<td>$R_1$</td>
<td>$R_1$</td>
<td>$R_1$</td>
</tr>
<tr>
<td>john</td>
<td>john</td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
<td><a href="mailto:J@edu.ex">J@edu.ex</a></td>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
</tr>
<tr>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
<td><a href="mailto:P@edu.ex">P@edu.ex</a></td>
<td>R_2</td>
<td>R_2</td>
<td></td>
</tr>
</tbody>
</table>

$\blacktriangleright \ \mu_2$ and $\mu_3$ are not compatible
Sets of mappings and operations

Let $\Omega_1$ and $\Omega_2$ be sets of mappings:

**Definition**

**Join:** $\Omega_1 \Join \Omega_2$

- $\{\mu_1 \cup \mu_2 | \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \text{ and } \mu_1, \mu_2 \text{ are compatibles}\}$
- extending mappings in $\Omega_1$ with compatible mappings in $\Omega_2$

will be used to define **AND**
Sets of mappings and operations

Let $\Omega_1$ and $\Omega_2$ be sets of mappings:

**Definition**

**Join:** $\Omega_1 \Join \Omega_2$

- $\{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \text{ and } \mu_1, \mu_2 \text{ are compatibles}\}$
- extending mappings in $\Omega_1$ with compatible mappings in $\Omega_2$

will be used to define **AND**

**Definition**

**Union:** $\Omega_1 \cup \Omega_2$

- $\{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\}$
- mappings in $\Omega_1$ plus mappings in $\Omega_2$ (the usual union of sets)

will be used to define **UNION**
Sets of mappings and operations

Definition

Difference: $\Omega_1 \setminus \Omega_2$

- $\{\mu \in \Omega_1 \mid \text{for all } \mu' \in \Omega_2, \mu \text{ and } \mu' \text{ are not compatibles}\}$
- mappings in $\Omega_1$ that cannot be extended with mappings in $\Omega_2$
Sets of mappings and operations

Definition

**Difference:** $\Omega_1 \setminus \Omega_2$

- $\{\mu \in \Omega_1 \mid \text{for all } \mu' \in \Omega_2, \mu \text{ and } \mu' \text{ are not compatibles}\}$
- mappings in $\Omega_1$ that cannot be extended with mappings in $\Omega_2$

Definition

**Left outer join:** $\Omega_1 \Join \Omega_2 = (\Omega_1 \Join \Omega_2) \cup (\Omega_1 \setminus \Omega_2)$

- extension of mappings in $\Omega_1$ with compatible mappings in $\Omega_2$
- plus the mappings in $\Omega_1$ that cannot be extended.

will be used to define $\text{OPT}$
Given an RDF graph $G$

**Definition**

\[
\begin{align*}
\llbracket \langle P_1 \text{ AND } P_2 \rangle \rrbracket_G &= \\
\llbracket \langle P_1 \text{ UNION } P_2 \rangle \rrbracket_G &= \\
\llbracket \langle P_1 \text{ OPT } P_2 \rangle \rrbracket_G &= \\
\llbracket \langle \text{SELECT } W \ P \rangle \rrbracket_G &=
\end{align*}
\]
Semantics of SPARQL: AND, UNION, OPT and SELECT

Given an RDF graph $G$

**Definition**

\[
\begin{align*}
\llbracket (P_1 \text{ AND } P_2) \rrbracket_G &= \llbracket P_1 \rrbracket_G \times \llbracket P_2 \rrbracket_G \\
\llbracket (P_1 \text{ UNION } P_2) \rrbracket_G &= \llbracket P_1 \rrbracket_G \cup \llbracket P_2 \rrbracket_G \\
\llbracket (P_1 \text{ OPT } P_2) \rrbracket_G &= \llbracket P_1 \rrbracket_G \Join \llbracket P_2 \rrbracket_G \\
\llbracket (\text{SELECT } W \ P) \rrbracket_G &= \{ \mu |_{W} \mid \mu \in \llbracket P \rrbracket_G \}
\end{align*}
\]
Given an RDF graph $G$

**Definition**

\[
\begin{align*}
[(P_1 \text{ AND } P_2)]_G &= [P_1]_G \times [P_2]_G \\
[(P_1 \text{ UNION } P_2)]_G &= [P_1]_G \cup [P_2]_G \\
[(P_1 \text{ OPT } P_2)]_G &= [P_1]_G \times [P_2]_G \\
[(\text{SELECT } W P)]_G &= \{ \mu\mid_W \mid \mu \in [P]_G \}
\end{align*}
\]

$\text{dom}(\mu\mid_W) = \text{dom}(\mu) \cap W$ and $\mu\mid_W(\?X) = \mu(\?X)$ for every $\?X \in \text{dom}(\mu\mid_W)$
Example (AND)

\[
\begin{align*}
(G_1, \text{name, john}) & \quad (G_2, \text{name, paul}) & \quad (G_3, \text{name, ringo}) \\
(G_1, \text{email, J@ed.ex}) & \quad (G_3, \text{email, R@ed.ex}) & \quad (G_3, \text{webPage, www.ringo.com})
\end{align*}
\]

\[
\llbracket ((?X, \text{name, ?N}) \ \text{AND} \ (?X, \text{email, ?E})) \rrbracket_G
\]
Example (AND)

\[ G : \begin{align*}
(R_1, \text{name, john}) & \quad (R_2, \text{name, paul}) & \quad (R_3, \text{name, ringo}) \\
(R_1, \text{email, J@ed.ex}) & \quad (R_3, \text{email, R@ed.ex}) & \quad (R_3, \text{webPage, www.ringo.com}) 
\end{align*} \]

\[ \llbracket((?X, \text{name, ?N}) \ \text{AND} \ (?X, \text{email, ?E}))\rrbracket_G \]

\[ \llbracket(?X, \text{name, ?N})\rrbracket_G \ \boxtimes \ \llbracket(?X, \text{email, ?E})\rrbracket_G \]
Example (AND)

\[
G : (R_1, \text{name}, \text{john}) \quad (R_2, \text{name}, \text{paul}) \quad (R_3, \text{name}, \text{ringo})
\]

\[
(R_1, \text{email}, \text{J@ed.ex}) \quad (R_3, \text{email}, \text{R@ed.ex}) \quad (R_3, \text{webPage}, \text{www.ringo.com})
\]

\[
\llbracket ((?X, \text{name}, ?N) \text{ AND } (?X, \text{email}, ?E)) \rrbracket_G
\]

\[
\llbracket (?X, \text{name}, ?N) \rrbracket_G \otimes \llbracket (?X, \text{email}, ?E) \rrbracket_G
\]

<table>
<thead>
<tr>
<th>( \mu_1 )</th>
<th>( \mu_2 )</th>
<th>( \mu_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>\text{john}</td>
<td></td>
</tr>
<tr>
<td>( R_2 )</td>
<td>\text{paul}</td>
<td></td>
</tr>
<tr>
<td>( R_3 )</td>
<td>\text{ringo}</td>
<td></td>
</tr>
</tbody>
</table>
Example (AND)

\( G : \)

\( (R_1, \text{name, john}) \)

\( (R_2, \text{name, paul}) \)

\( (R_3, \text{name, ringo}) \)

\( (R_1, \text{email, J@ed.ex}) \)

\( (R_3, \text{email, R@ed.ex}) \)

\( (R_3, \text{webPage, www.ringo.com}) \)

\[
\llbracket ((?X, \text{name, } ?N) \text{ AND } (?X, \text{email, } ?E)) \rrbracket_G
\]

\[
\llbracket (?X, \text{name, } ?N) \rrbracket_G \Join \llbracket (?X, \text{email, } ?E) \rrbracket_G
\]

\[
\begin{array}{c|c}
\mu_1 & ?X & ?N \\
R_1 & john \\
\mu_2 & R_2 & paul \\
\mu_3 & R_3 & ringo \\
\end{array}
\]

\[
\begin{array}{c|c}
\mu_4 & ?X & ?E \\
R_1 & J@ed.ex \\
\mu_5 & R_3 & R@ed.ex \\
\end{array}
\]
Example (AND)

\[ G : (R_1, \text{name}, \text{john}) \quad (R_2, \text{name}, \text{paul}) \quad (R_3, \text{name}, \text{ringo}) \]

\[ G : (R_1, \text{email}, \text{J@ed.ex}) \quad (R_3, \text{email}, \text{R@ed.ex}) \]

\[ (R_3, \text{webPage}, \text{www.ringo.com}) \]

\[
\left[\left( (\forall X. \text{name}, ?N) \land (\forall X. \text{email}, ?E) \right) \right]_G
\]

\[
\left[ (\forall X. \text{name}, ?N) \right]_G \land \left[ (\forall X. \text{email}, ?E) \right]_G
\]
Example (AND)

\[
G : (R_1, \text{name}, \text{john}) \quad (R_2, \text{name}, \text{paul}) \quad (R_3, \text{name}, \text{ringo}) \\
(R_1, \text{email}, \text{J@ed.ex}) \quad (R_3, \text{email}, \text{R@ed.ex}) \quad (R_3, \text{webPage}, \text{www.ringo.com})
\]

\[
\llbracket ((?X, \text{name}, ?N) \text{ AND } (?X, \text{email}, ?E)) \rrbracket_G \\
\llbracket (?X, \text{name}, ?N) \rrbracket_G \Join \llbracket (?X, \text{email}, ?E) \rrbracket_G
\]

\[
\begin{array}{ccc}
\mu_1 & ?X & ?N \\
R_1 & \text{john} \\
\mu_2 & R_2 & \text{paul} \\
\mu_3 & R_3 & \text{ringo}
\end{array} \Join \\
\begin{array}{cc}
\mu_4 & ?X & ?E \\
R_1 & \text{J@ed.ex} \\
\mu_5 & R_3 & \text{R@ed.ex}
\end{array}
\]

\[
\begin{array}{ccc}
\mu_1 \cup \mu_4 & ?X & ?N & ?E \\
R_1 & \text{john} & \text{J@ed.ex} \\
\mu_3 \cup \mu_5 & R_3 & \text{ringo} & \text{R@ed.ex}
\end{array}
\]
Example (OPT)

\[ G : (R_1, \text{name}, joh) (R_2, \text{name}, paul) (R_3, \text{name}, ringo) \]

\[ (R_1, \text{email}, J@ed.ex) (R_3, \text{email}, R@ed.ex) (R_3, \text{webPage}, www.ringo.com) \]

\[ \llbracket ((?X, \text{name}, ?N) \text{OPT} (?X, \text{email}, ?E)) \rrbracket_G \]
Example (OPT)

\( G : \)  
\( (R_1, \text{name, john}) \)  
\( (R_2, \text{name, paul}) \)  
\( (R_3, \text{name, ringo}) \)  
\( (R_1, \text{email, J@ed.ex}) \)  
\( (R_3, \text{email, R@ed.ex}) \)  
\( (R_3, \text{webPage, www.ringo.com}) \)

\[\begin{align*}
\llbracket((?X, \text{name, ?N}) \text{ OPT } (?X, \text{email, ?E}))\rrbracket_G \\
\llbracket(?X, \text{name, ?N})\rrbracket_G \boxtimes \llbracket(?X, \text{email, ?E})\rrbracket_G
\end{align*}\]
Example (OPT)

\[ G : \]

\[ (R_1, \text{name}, \text{john}) \]
\[ (R_2, \text{name}, \text{paul}) \]
\[ (R_3, \text{name}, \text{ringo}) \]

\[ (R_1, \text{email}, \text{J@ed.ex}) \]
\[ (R_3, \text{email}, \text{R@ed.ex}) \]
\[ (R_3, \text{webPage}, \text{www.ringo.com}) \]

\[ \llbracket \begin{array}{cc}
(\exists X \cdot \text{name} \in X) & \text{OPT} \\
(\exists X \cdot \text{email} \in X)
\end{array} \rrbracket_G \]

\[ \llbracket (\exists X \cdot \text{name} \in X) \rrbracket_G \times \llbracket (\exists X \cdot \text{email} \in X) \rrbracket_G \]

\[ \begin{array}{|c|c|}
\hline
\text{?X} & \text{?N} \\
\hline
R_1 & \text{john} \\
R_2 & \text{paul} \\
R_3 & \text{ringo} \\
\hline
\end{array} \]
Example (OPT)

\[ G : (R_1, \text{name, john}) (R_2, \text{name, paul}) (R_3, \text{name, ringo}) \]
\[ (R_1, \text{email, J@ed.ex}) (R_3, \text{email, R@ed.ex}) (R_3, \text{webPage, www.ringo.com}) \]

\[ \llbracket ((?X, \text{name, ?N}) \text{ OPT } (?X, \text{email, ?E})) \rrbracket_G \]
\[ \llbracket (?X, \text{name, ?N}) \rrbracket_G \Join \llbracket (?X, \text{email, ?E}) \rrbracket_G \]

\[
\begin{array}{|c|c|}
\hline
?X & ?N \\
\hline
R_1 & \text{john} \\
\hline
R_2 & \text{paul} \\
\hline
R_3 & \text{ringo} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
?X & ?E \\
\hline
R_1 & \text{J@ed.ex} \\
\hline
R_3 & \text{R@ed.ex} \\
\hline
\end{array}
\]
Example (OPT)

\[ G : (R_1, \text{name}, \text{john}) \quad (R_2, \text{name}, \text{paul}) \quad (R_3, \text{name}, \text{ringo}) \]

\[ (R_1, \text{email}, \text{J@ed.ex}) \quad (R_3, \text{email}, \text{R@ed.ex}) \quad (R_3, \text{webPage}, \text{www.ringo.com}) \]

\[
\llbracket ((?X, \text{name}, ?N) \ \text{OPT} \ (?X, \text{email}, ?E)) \rrbracket_G
\]

\[
\llbracket (?X, \text{name}, ?N) \rrbracket_G \ \boxtimes \ \llbracket (?X, \text{email}, ?E) \rrbracket_G
\]
Example (OPT)

\[ G : (R_1, \text{name, john}) (R_2, \text{name, paul}) (R_3, \text{name, ringo}) \]

\[ (R_1, \text{email, J@ed.ex}) (R_3, \text{email, R@ed.ex}) \]

\[ (R_3, \text{webPage, www.ringo.com}) \]

\[
\llbracket ((?X, \text{name, ?N}) \text{ OPT } (?X, \text{email, ?E})) \rrbracket_G
\]

\[
\llbracket (?X, \text{name, ?N}) \rrbracket_G \bigtriangleup \llbracket (?X, \text{email, ?E}) \rrbracket_G
\]

<table>
<thead>
<tr>
<th>( \mu_1 )</th>
<th>( \mu_2 )</th>
<th>( \mu_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?X)</td>
<td>(?N)</td>
<td>(R_1)</td>
</tr>
<tr>
<td>(R_2)</td>
<td>\text{paul}</td>
<td></td>
</tr>
<tr>
<td>(R_3)</td>
<td>\text{ringo}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \mu_4 )</th>
<th>( \mu_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?X)</td>
<td>(?E)</td>
</tr>
<tr>
<td>(R_3)</td>
<td>\text{<a href="mailto:R@ed.ex">R@ed.ex</a>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \mu_1 \cup \mu_4 )</th>
<th>( \mu_3 \cup \mu_5 )</th>
<th>( \mu_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?X)</td>
<td>(?N)</td>
<td>(R_1)</td>
</tr>
<tr>
<td>(R_3)</td>
<td>\text{ringo}</td>
<td>\text{<a href="mailto:R@ed.ex">R@ed.ex</a>}</td>
</tr>
</tbody>
</table>
Example (OPT)

\[ G : (R_1, \text{name, john}) \quad (R_2, \text{name, paul}) \quad (R_3, \text{name, ringo}) \]

\[ (R_1, \text{email, J@ed.ex}) \quad (R_3, \text{email, R@ed.ex}) \quad (R_3, \text{webPage, www.ringo.com}) \]

\[
\llbracket ((?X, \text{name, ?N}) \text{ OPT } (?X, \text{email, ?E})) \rrbracket_G
\]

\[
\llbracket (?X, \text{name, ?N}) \rrbracket_G \Join \llbracket (?X, \text{email, ?E}) \rrbracket_G
\]

\[
\begin{array}{c|c|c}
\mu_1 & ?X & ?N \\
\hline
R_1 & \text{john} \\
\mu_2 & R_2 & \text{paul} \\
\mu_3 & R_3 & \text{ringo}
\end{array}
\Join
\begin{array}{c|c|c}
\mu_4 & ?X & ?E \\
\hline
R_1 & \text{J@ed.ex} \\
\mu_5 & R_3 & \text{R@ed.ex}
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\mu_1 \cup \mu_4 & ?X & ?N & ?E \\
\hline
R_1 & \text{john} & \text{J@ed.ex} \\
\mu_3 \cup \mu_5 & R_3 & \text{ringo} & \text{R@ed.ex} \\
\mu_2 & R_2 & \text{paul}
\end{array}
\]
Example (UNION)

\[ G : \begin{align*}
(R_1, \text{name}, \text{john}) & \quad (R_2, \text{name}, \text{paul}) & \quad (R_3, \text{name}, \text{ringo}) \\
(R_1, \text{email}, J@ed.ex) & \quad (R_3, \text{email}, R@ed.ex) & \quad (R_3, \text{webPage}, \text{www.ringo.com})
\end{align*} \]

\[ [[[(?X, \text{email}, ?\text{Info}) \ \text{UNION} \ (\ ?X, \text{webPage}, ?\text{Info})]_G]_G \]
Example (UNION)

\[ G^* = (R_1, \text{name}, \text{john}) \quad (R_2, \text{name}, \text{paul}) \quad (R_3, \text{name}, \text{ringo}) \]

\[ G = (R_1, \text{email}, \text{J@ed.ex}) \quad (R_3, \text{email}, \text{R@ed.ex}) \quad (R_3, \text{webPage}, \text{www.ringo.com}) \]

\[
\llbracket (\langle ?X, \text{email}, ?\text{Info} \rangle \ \text{UNION} \ \langle ?X, \text{webPage}, ?\text{Info} \rangle) \rrbracket_G
\]

\[
\llbracket (\langle ?X, \text{email}, ?\text{Info} \rangle) \rrbracket_G \cup \llbracket (\langle ?X, \text{webPage}, ?\text{Info} \rangle) \rrbracket_G
\]
Example (UNION)

\( G : \)  
- \((R_1, \text{name}, \text{john})\)  
- \((R_2, \text{name}, \text{paul})\)  
- \((R_3, \text{name}, \text{ringo})\)  
- \((R_1, \text{email}, J@ed.ex)\)  
- \((R_3, \text{email}, R@ed.ex)\)  
- \((R_3, \text{webPage}, \text{www.ringo.com})\)

\[
\llbracket((?X, \text{email}, ?\text{Info}) \ \cup \ \text{UNION} \ \cup ((?X, \text{webPage}, ?\text{Info}))\rrbracket \_G
\]

\[
\llbracket((?X, \text{email}, ?\text{Info})\rrbracket \_G \ \cup \ \llbracket((?X, \text{webPage}, ?\text{Info})\rrbracket \_G
\]

\[
\begin{array}{|c|c|}
\hline
?X & ?\text{Info} \\
\hline
R_1 & J@ed.ex \\
\hline
R_3 & R@ed.ex \\
\hline
\end{array}
\]
Example (UNION)

\[ G : \]

\[
\begin{align*}
(R_1, \text{name, john}) & \quad (R_2, \text{name, paul}) & \quad (R_3, \text{name, ringo}) \\
(R_1, \text{email, J@ed.ex}) & \quad (R_3, \text{email, R@ed.ex}) & \quad (R_3, \text{webPage, www.ringo.com})
\end{align*}
\]

\[
\llbracket (\langle ?X, \text{email, } ?\text{Info} \rangle \cup \langle ?X, \text{webPage, } ?\text{Info} \rangle) \rrbracket_G
\]

\[
\llbracket (\langle ?X, \text{email, } ?\text{Info} \rangle) \rrbracket_G \cup \llbracket (\langle ?X, \text{webPage, } ?\text{Info} \rangle) \rrbracket_G
\]

\[
\begin{array}{|c|c|}
\hline
\mu_1 & \mu_2 \\
\hline
\langle ?X, \text{email, J@ed.ex} \rangle & \langle ?X, \text{email, R@ed.ex} \rangle \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\mu_3 & \\
\hline
\langle ?X, \text{webPage, www.ringo.com} \rangle & \\
\hline
\end{array}
\]
Example (UNION)

\[ G : \]
\[
\begin{array}{lll}
(R_1, \text{name, john}) & (R_2, \text{name, paul}) & (R_3, \text{name, ringo}) \\
(R_1, \text{email, J@ed.ex}) & & (R_3, \text{email, R@ed.ex}) \\
& & (R_3, \text{webPage, www.ringo.com}) \\
\end{array}
\]

\[
[[((?X, \text{email, ?Info}) \cup (?X, \text{webPage, ?Info}))]]_G
\]

\[
[[(?X, \text{email, ?Info})]]_G \cup [[(?X, \text{webPage, ?Info})]]_G
\]

<table>
<thead>
<tr>
<th>(\mu_1)</th>
<th>(\mu_2)</th>
<th>(\mu_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?X)</td>
<td>(\text{email, ?Info})</td>
<td>(\text{email, ?Info})</td>
</tr>
<tr>
<td>(R_1)</td>
<td><a href="mailto:J@ed.ex">J@ed.ex</a></td>
<td></td>
</tr>
<tr>
<td>(R_3)</td>
<td><a href="mailto:R@ed.ex">R@ed.ex</a></td>
<td></td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp; R_3 &amp; <a href="http://www.ringo.com">www.ringo.com</a></td>
<td></td>
</tr>
</tbody>
</table>
Example (UNION)

\[ G : (R_1, \text{name, john}) (R_2, \text{name, paul}) (R_3, \text{name, ringo}) \]

\[ (R_1, \text{email, J@ed.ex}) (R_3, \text{email, R@ed.ex}) \]

\[ (R_3, \text{webPage, www.ringo.com}) \]

\[
\llbracket (\exists X, \text{email, ?Info}) \cup (\exists X, \text{webPage, ?Info}) \rrbracket \]

\[
\llbracket (\exists X, \text{email, ?Info}) \rrbracket \cup \llbracket (\exists X, \text{webPage, ?Info}) \rrbracket
\]

\[
\begin{array}{c|c}
?X & \text{?Info} \\
\hline
R_1 & \text{J@ed.ex} \\
R_3 & \text{R@ed.ex} \\
\end{array}
\]

\[
\begin{array}{c|c}
?X & \text{?Info} \\
\hline
R_3 & \text{www.ringo.com} \\
\end{array}
\]

\[
\begin{array}{c|c}
?X & \text{?Info} \\
\hline
R_1 & \text{J@ed.ex} \\
R_3 & \text{R@ed.ex} \\
R_3 & \text{www.ringo.com} \\
\end{array}
\]
Example (SELECT)

\[
G : (R_1, \text{name, john}) \quad (R_2, \text{name, paul}) \quad (R_3, \text{name, ringo}) \\
(R_1, \text{email, J@ed.ex}) \quad (R_3, \text{email, R@ed.ex}) \quad (R_3, \text{webPage, www.ringo.com})
\]

\[
[(\text{SELECT } \{?N, ?E\} ((?X, \text{name, ?N}) \text{ AND } (?X, \text{email, ?E})))]_G
\]
Example (SELECT)

\[ G : (R_1, \text{name}, \text{john}) (R_2, \text{name}, \text{paul}) (R_3, \text{name}, \text{ringo}) \]

\[ (R_1, \text{email}, \text{J@ed.ex}) (R_3, \text{email}, \text{R@ed.ex}) (R_3, \text{webPage}, \text{www.ringo.com}) \]

\[ \llbracket (\text{SELECT} \{?N, ?E\} ((?X, \text{name}, ?N) \text{ AND } (?X, \text{email}, ?E)))) \rrbracket_G \]

\[ \text{SELECT}\{?N, ?E\} \]

<table>
<thead>
<tr>
<th>(?X)</th>
<th>(?N)</th>
<th>(?E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_1)</td>
<td>\text{john}</td>
<td>\text{<a href="mailto:J@ed.ex">J@ed.ex</a>}</td>
</tr>
<tr>
<td>(R_3)</td>
<td>\text{ringo}</td>
<td>\text{<a href="mailto:R@ed.ex">R@ed.ex</a>}</td>
</tr>
</tbody>
</table>
Example (SELECT)

\( G : (R_1, \text{name}, \text{john}) (R_2, \text{name}, \text{paul}) (R_3, \text{name}, \text{ringo}) \\
(R_1, \text{email}, \text{J@ed.ex}) (R_3, \text{email}, \text{R@ed.ex}) \)

\[
\begin{array}{|c|c|c|}
\hline
\hline
\mu_1 & \text{R}_1 & \text{john} & \text{J@ed.ex} \\
\mu_2 & \text{R}_3 & \text{ringo} & \text{R@ed.ex} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\mu_1|\{?N, ?E\} & ?N & ?E \\
\hline
\text{john} & \text{J@ed.ex} \\
\text{ringo} & \text{R@ed.ex} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\mu_2|\{?N, ?E\} & ?N & ?E \\
\hline
\end{array}
\]
Filter expressions (value constraints)

Filter expression: \((P \text{ FILTER } R)\)

- \(P\) is a graph pattern
- \(R\) is a built-in condition

We consider in \(R\):

- equality = among variables and RDF terms
- unary predicate bound
- boolean combinations \((\land, \lor, \neg)\)
A mapping $\mu$ satisfies a condition $R \ (\mu \models R)$ if:
A mapping $\mu$ satisfies a condition $R (\mu \models R)$ if:

- $R$ is $?X = c$, $?X \in \text{dom}(\mu)$ and $\mu(?X) = c$
- $R$ is $?X = ?Y$, $?X, ?Y \in \text{dom}(\mu)$ and $\mu(?X) = \mu(?Y)$
- $R$ is bound(?$X$) and $?X \in \text{dom}(\mu)$
Satisfaction of value constraints

A mapping $\mu$ satisfies a condition $R$ ($\mu \models R$) if:

- $R$ is $?X = c$, $?X \in \text{dom}(\mu)$ and $\mu(?X) = c$
- $R$ is $?X = ?Y$, $?X, ?Y \in \text{dom}(\mu)$ and $\mu(?X) = \mu(?Y)$
- $R$ is bound(?$X$) and $?X \in \text{dom}(\mu)$
- usual rules for Boolean connectives
Satisfaction of value constraints

A mapping \( \mu \) satisfies a condition \( R (\mu \models R) \) if:

- \( R \) is \( ?X = c, ?X \in \text{dom}(\mu) \) and \( \mu(?X) = c \)
- \( R \) is \( ?X = ?Y, ?X, ?Y \in \text{dom}(\mu) \) and \( \mu(?X) = \mu(?Y) \)
- \( R \) is bound(?X) and \( ?X \in \text{dom}(\mu) \)
- usual rules for Boolean connectives

**Definition**

**FILTER** : selects mappings that satisfy a condition

\[
\llbracket (P \ \text{FILTER} \ R) \rrbracket_G = \{ \mu \in \llbracket P \rrbracket_G \mid \mu \models R \}
\]
Example (FILTER)

\[ \begin{align*}
& (R_1, \text{name}, \text{john}) \\
& (R_2, \text{name}, \text{paul}) \\
& (R_3, \text{name}, \text{ringo}) \\
& (R_1, \text{email}, \text{J@ed.ex}) \\
& (R_3, \text{email}, \text{R@ed.ex}) \\
& (R_3, \text{webPage}, \text{www.ringo.com})
\end{align*} \]

\[ \llbracket (\exists X. \text{name}, \exists N. \text{FILTER (N = ringo} \lor \text{N = paul)}) \rrbracket_G \]
Example (FILTER)

\[
G : \quad (R_1, \text{name}, \text{john}) \quad (R_2, \text{name}, \text{paul}) \quad (R_3, \text{name}, \text{ringo}) \\
(R_1, \text{email}, \text{J@ed.ex}) \quad (R_3, \text{email}, \text{R@ed.ex}) \quad (R_3, \text{webPage}, \text{www.ringo.com})
\]

\[
\llbracket((?X, \text{name}, ?N) \ \text{FILTER} \ (\text{?N} = \text{ringo} \lor \text{?N} = \text{paul}))\rrbracket_G
\]

<table>
<thead>
<tr>
<th>(\mu_1)</th>
<th>(?X)</th>
<th>(?N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_1)</td>
<td>john</td>
<td></td>
</tr>
<tr>
<td>(\mu_2)</td>
<td>(R_2)</td>
<td>paul</td>
</tr>
<tr>
<td>(\mu_3)</td>
<td>(R_3)</td>
<td>ringo</td>
</tr>
</tbody>
</table>
Example (FILTER)

\[(R_1, \text{name, john}) \quad (R_2, \text{name, paul}) \quad (R_3, \text{name, ringo})\]

\[G : (R_1, \text{email, J@ed.ex}) \quad (R_3, \text{email, R@ed.ex}) \quad (R_3, \text{webPage, www.ringo.com})\]

\[
\llbracket((?X, \text{name, ?N}) \ \text{FILTER} \ (?N = \text{ringo} \ \lor \ ?N = \text{paul}))\rrbracket_G
\]

\[
\begin{array}{c|c}
\mu_1 & R_1 & \text{john} \\
\mu_2 & R_2 & \text{paul} \\
\mu_3 & R_3 & \text{ringo} \\
\end{array}
\]

\(?N = \text{ringo} \ \lor \ ?N = \text{paul}\)
Example (FILTER)

\[ ((\forall X, \text{name}, \forall N) \text{ FILTER } (\forall N = \text{ringo} \lor \forall N = \text{paul})) \] 

\[
\begin{array}{c|c|c}
\mu_1 & X & N \\
R_1 & john & \\
R_2 & paul & \\
R_3 & ringo & \\
\end{array}
\] 

\[ \forall N = \text{ringo} \lor \forall N = \text{paul} \]

\[
\begin{array}{c|c|c}
\mu_2 & X & N \\
R_2 & paul & \\
R_3 & ringo & \\
\end{array}
\]
Example (FILTER)

$G : (R_1, \text{name, john}) \quad (R_2, \text{name, paul}) \quad (R_3, \text{name, ringo})$

\[
\llbracket((?X, \text{name, } ?N) \text{ OPT } (?X, \text{email, } ?E)) \text{ FILTER } \neg \text{bound}(?E))\rrbracket_G
\]
Example (FILTER)

\[ ((\text{?X}, \text{name}, \text{?N}) \ \text{OPT} \ (\text{?X}, \text{email}, \text{?E})) \ \text{FILTER} \neg \text{bound(\text{?E}))) \]_G

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>john</td>
<td><a href="mailto:J@ed.ex">J@ed.ex</a></td>
</tr>
<tr>
<td>R3</td>
<td>ringo</td>
<td><a href="mailto:R@ed.ex">R@ed.ex</a></td>
</tr>
<tr>
<td>R2</td>
<td>paul</td>
<td></td>
</tr>
</tbody>
</table>

\[ \mu_1 \cup \mu_4 \\
\mu_3 \cup \mu_5 \\
\mu_2 \]

\( G : \ (R_1, \text{name, john}) \quad (R_2, \text{name, paul}) \quad (R_3, \text{name, ringo}) \quad (R_3, \text{email, R@ed.ex}) \quad (R_3, \text{webPage, www.ringo.com}) \)
Example (FILTER)

\[ G : (R_1, \text{name}, \text{john}) (R_2, \text{name}, \text{paul}) (R_3, \text{name}, \text{ringo}) (R_1, \text{email}, J@ed.ex) (R_3, \text{email}, R@ed.ex) (R_3, \text{webPage}, \text{www.ringo.com}) \]

\[ \llbracket (((?X, \text{name}, ?N) \text{ OPT } (?X, \text{email}, ?E)) \text{ FILTER } \neg \text{bound}(?E)) \rrbracket_G \]

\[
\begin{array}{c|c|c}
\mu_1 \cup \mu_4 & ?X & ?N \\
R_1 & \text{john} & J@ed.ex \\
\mu_3 \cup \mu_5 & R_3 & \text{ringo} \text{ R@ed.ex} \\
\mu_2 & R_2 & \text{paul} \\
\end{array}
\]

\[ \neg \text{bound}(?E) \]
Example (FILTER)

\[ G : (R_1, \text{name}, \text{john}) \quad (R_2, \text{name}, \text{paul}) \quad (R_3, \text{name}, \text{ringo}) \]

\[ (R_1, \text{email}, \text{J@ed.ex}) \quad (R_3, \text{email}, \text{R@ed.ex}) \quad (R_3, \text{webPage}, \text{www.ringo.com}) \]

\[ (((?X, \text{name}, ?N) \ \text{OPT} \ (?X, \text{email}, ?E)) \ \text{FILTER} \ \neg \text{bound(?E)})]_G \]

\[
\begin{array}{ccc}
\mu_1 \cup \mu_4 \\
R_1 & \text{john} & \text{J@ed.ex} \\
\mu_3 \cup \mu_5 \\
R_3 & \text{ringo} & \text{R@ed.ex} \\
\mu_2 \\
R_2 & \text{paul} \\
\end{array}
\]

\[ \neg \text{bound(?E)} \]

\[
\begin{array}{cc}
\mu_2 \\
R_2 & \text{paul} \\
\end{array}
\]
A new version of SPARQL was recently released (March 2013): SPARQL 1.1

Some new features in SPARQL 1.1:

- Entailment regimes for RDFS and OWL
- Navigational capabilities: Property paths
SPARQL provides limited navigational capabilities
SPARQL provides limited navigational capabilities

\[
\text{(SELECT } ?X \ (\text{(?X, friendOf, ?Y) AND (} ?Y, \text{ name, George}))\text{)}
\]
A possible solution: Property paths
A possible solution: Property paths

(\text{SELECT } ?X (\text{(friendOf)*, } ?Y) \text{ AND (} ?Y, \text{name, George}))
Syntax of property paths:

\[ \text{exp} \ := \ a \ | \ \text{exp}/\text{exp} \ | \ \text{exp}|\text{exp} \ | \ \text{exp}^* \]

where \( a \in U \)
Navigational capabilities in SPARQL 1.1: Property paths

Syntax of property paths:

\[ \text{exp} \ := \ a \mid \text{exp/\text{exp}} \mid \text{exp|\text{exp}} \mid \text{exp}^* \]

where \( a \in U \)

Other expressions are allowed:

\[ \hat{\text{exp}} \quad : \quad \text{inverse path} \]
\[ !(a_1|\ldots|a_n) \quad : \quad \text{a URI which is not one of } a_i \ (1 \leq i \leq n) \]
Evaluating property paths

The evaluation of a property path over an RDF graph $G$ is defined as follows:
Evaluating property paths

The evaluation of a property path over an RDF graph $G$ is defined as follows:

$$[a]_G = \{(x, y) \mid (x, a, y) \in G\}$$
Evaluating property paths

The evaluation of a property path over an RDF graph $G$ is defined as follows:

\[
\llbracket a \rrbracket_G = \{(x, y) \mid (x, a, y) \in G\}
\]

\[
\llbracket \text{exp}_1/\text{exp}_2 \rrbracket_G = \{(x, y) \mid \exists z \ (x, z) \in \llbracket \text{exp}_1 \rrbracket_G \text{ and } (z, y) \in \llbracket \text{exp}_2 \rrbracket_G\}
\]
Evaluating property paths

The evaluation of a property path over an RDF graph $G$ is defined as follows:

\[
[a]_G = \{(x, y) \mid (x, a, y) \in G\}
\]
\[
[exp_1/exp_2]_G = \{(x, y) \mid \exists z \ (x, z) \in [exp_1]_G \text{ and } (z, y) \in [exp_2]_G\}
\]
\[
[exp_1|exp_2]_G = [exp_1]_G \cup [exp_2]_G
\]
Evaluating property paths

The evaluation of a property path over an RDF graph $G$ is defined as follows:

$$[a]_G = \{(x, y) \mid (x, a, y) \in G\}$$

$$[\text{exp}_1/\text{exp}_2]_G = \{(x, y) \mid \exists z \, (x, z) \in [\text{exp}_1]_G \text{ and } (z, y) \in [\text{exp}_2]_G\}$$

$$[\text{exp}_1\mid\text{exp}_2]_G = [\text{exp}_1]_G \cup [\text{exp}_2]_G$$

$$[\text{exp}^*]_G = \{(a, a) \mid a \text{ is a URI in } G\} \cup [\text{exp}]_G \cup [\text{exp}/\text{exp}]_G \cup [\text{exp}/\text{exp}/\text{exp}]_G \cup \cdots$$
Property paths in SPARQL 1.1

New element in SPARQL 1.1: A triple of the form \((x, \text{exp}, y)\)

- \text{exp} is a property path
- \(x\) (resp. \(y\)) is either an element from \(U\) or a variable
Property paths in SPARQL 1.1

New element in SPARQL 1.1: A triple of the form $(x, exp, y)$

- $exp$ is a property path
- $x$ (resp. $y$) is either an element from $U$ or a variable

Example

- $(\?X, \text{friendOf}^*, \?Y)$: Checks whether there exists a path of friends of arbitrary length from $\?X$ to $\?Y$
New element in SPARQL 1.1: A triple of the form $(x, exp, y)$

- $exp$ is a property path
- $x$ (resp. $y$) is either an element from $U$ or a variable

Example

- $(?X, \text{friendOf}^*, ?Y)$: Checks whether there exists a path of friends of arbitrary length from $?X$ to $?Y$
- $(?X, \text{rdf:sc}^*, \text{person})$: Checks whether the value stored in $?X$ is a subclass of person
Property paths in SPARQL 1.1

New element in SPARQL 1.1: A triple of the form \((x, exp, y)\)

- \(exp\) is a property path
- \(x\) (resp. \(y\)) is either an element from \(U\) or a variable

Example

- \((?X, \text{(friendOf)}^*, ?Y)\): Checks whether there exists a path of friends of arbitrary length from \(?X\) to \(?Y\)
- \((?X, \text{(rdf:sc)}^*, \text{person})\): Checks whether the value stored in \(?X\) is a subclass of person
- \((?X, \text{(rdf:sp)}^*, ?Y)\): Checks whether the value stored in \(?X\) is a subproperty of the value stored in \(?Y\)
Semantics of property paths

Evaluation of $t = (?X, exp, ?Y)$ over an RDF graph $G$ is the set of mappings $\mu$ such that:
Semantics of property paths

Evaluation of $t = (?X, \text{exp}, ?Y)$ over an RDF graph $G$ is the set of mappings $\mu$ such that:

$\text{dom}(\mu) = \{?X, ?Y\}$
Semantics of property paths

Evaluation of $t = (?X, \text{exp}, ?Y)$ over an RDF graph $G$ is the set of mappings $\mu$ such that:

- $\text{dom}(\mu) = \{?X, ?Y\}$
- $(\mu(?X), \mu(?Y)) \in \llbracket \text{exp} \rrbracket_G$
Semantics of property paths

Evaluation of $t = (?X, \text{exp}, ?Y)$ over an RDF graph $G$ is the set of mappings $\mu$ such that:

- $\text{dom} (\mu) = \{ ?X, ?Y \}$
- $(\mu(?X), \mu(?Y)) \in \llbracket \text{exp} \rrbracket_G$

Other cases are defined analogously.
Evaluation of $t = (?X, \text{exp}, ?Y)$ over an RDF graph $G$ is the set of mappings $\mu$ such that:

- $\text{dom}(\mu) = \{?X, ?Y\}$
- $(\mu(?X), \mu(?Y)) \in \llbracket \text{exp} \rrbracket_G$

Other cases are defined analogously.

**Example**

- $((?X, \text{KLM}/(\text{KLM})^*, ?Y) \text{ FILTER } \neg(?X = ?Y))$: It is possible to go from $?X$ to $?Y$ by using the airline KLM, where $?X$, $?Y$ are different cities.
Comments on papers

- Jorge Perez, Marcelo Arenas, Claudio Gutierrez: Semantics and complexity of SPARQL. ACM Trans. Database Syst. 34(3) (2009)
- M. Arenas, J. Perez: Querying semantic web data with SPARQL. PODS 2011: 305-316
  In these two papers, your essays ought to concentrate on complexity, as semantics was already covered.
- Marcelo Arenas, Georg Gottlob, Andreas Pieris: Expressive languages for querying the semantic web. PODS 2014: 14-26
  Extend SPARQL with more expressive ontologies and recursion, and translation into datalog.
- Leonid Libkin, Juan L. Reutter, Domagoj Vrgoc: Trial for RDF: adapting graph query languages for RDF data. PODS 2013: 201-212
  Are graph data and RDF the same? Not really. This shows how to bridge them.
  Extending navigational capabilities, using some XPath ideas.
- Marcelo Arenas, Sebastian Conca, Jorge Perez: Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. WWW 2012: 629-638
  Two papers showing that bad things happen if one queries RDF according to SPARQL 1.1 standard, and different solutions for fixing the problem.