Unit 5:
SPARQL1.1  ...

new! W3C Recommendation since 21 March 2013
This is where SPARQL1.1 started (2009)

Various SPARQL1.0 implementations out there, various extensions.

Missing common feature requirements in existing implementations or requested urgently by the community:
- Assignment/Project Expressions
- Aggregate functions (SUM, AVG, MIN, MAX, COUNT, …)
- Subqueries
- Property paths
  - complaint: SPARQL1.0 isn’t quite a “graph” query language

Ease of use:
- Why is Negation “hidden” in SPARQL1.0?

Interplay with other SW standards:
- SPARQL1.0 only defined for simple RDF entailment
- Other Entailment regimes missing:
  - RDF(S)
  - OWL2
  - (RIF)
Goals of SPARQL1.1

Per charter (http://www.w3.org/2009/05/sparql-phase-II-charter.html)

- “The scope of this charter is to extend SPARQL technology to include some of the features that the community has identified as both desirable and important for interoperability based on experience with the initial version of the standard.”

- No inclusion of new features that still require research
- Upwards compatible with SPARQL1.0
- The name SPARQL1.1 shall indicate an incremental change rather than any fundamental changes.
New in SPARQL1.1

SPARQL1.1 Query Language
- Project Expressions
- Aggregate functions
- Subqueries
- Negation
- Property Paths
- Extend the function library

SPARQL 1.1 Federated Query
- Basic federated Queries over different SPARQL endpoints

SPARQL 1.1 Entailment
- RDF(S), OWL, RIF

SPARQL 1.1 Update
- Full Update language

SPARQL 1.1 Graph Store HTTP Protocol
- simple RESTful update method to modify RDF graphs (HTTP methods)

SPARQL 1.1 Service Description
- Method for discovering a SPARQL endpoint’s capabilities
- Summary of its data

Plus several results formats (JSON, CSV/TSV, XML) and refurbished SPARQL Protocol
Where is SPARQL 1.1 in terms of becoming a standard?

Standardization process: Six types of documents

- **Note**
  - Not a component in the standardization process
  - No declaration that W3C stands behind

- **Working Draft (WD)**
  - Documentation of a discussion condition

- **Last Call WD**
  - When the goals are reached

- **Candidate Recommendation (CR)**
  - Confirmation of success

- **Proposed Recommendation**
  - Extension; partial implementation

- **Recommendation**
  - Official W3C standard
New query language features

- Project Expressions
- Aggregate functions
- Subqueries
- Negation
- Property Paths
Assignments, Creating new values...

```
PREFIX ex: <http://example.org/>
SELECT ?Item ?NewP
WHERE { ?Item ex:price ?Pr FILTER (?NewP = ?Pr * 1.1) }
```

Data:

- @prefix ex: <http://example.org/> .
- ex:lemonade1 ex:price 3.
- ex:beer1 ex:price 3.
- ex:wine1 ex:price 3.50.

Results:

<table>
<thead>
<tr>
<th>?Item</th>
<th>?NewP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Assignments, Creating new values...

PREFIX ex: <http://example.org/>
SELECT ?Item (?Pr * 1.1 AS ?NewP )
WHERE { ?Item ex:price ?Pr }

Data:

@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 .
ex:beer1 ex:price 3.
ex:wine1 ex:price 3.50 .

<table>
<thead>
<tr>
<th>?Item</th>
<th>?NewP</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemonade</td>
<td>3.3</td>
</tr>
<tr>
<td>beer</td>
<td>3.3</td>
</tr>
<tr>
<td>wine</td>
<td>3.85</td>
</tr>
</tbody>
</table>
Assignments, Creating new values...

PREFIX ex: <http://example.org/>
SELECT ?Item (?Pr * 1.1 AS ?NewP )
WHERE { ?Item ex:price ?Pr }

Data:

@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 .
ex:beer1 ex:price 3. 
ex:wine1 ex:price 3.50 .
ex:liqueur1 ex:price "n/a".

Results:

<table>
<thead>
<tr>
<th>?Item</th>
<th>?NewP</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemonade</td>
<td>3.3</td>
</tr>
<tr>
<td>beer</td>
<td>3.3</td>
</tr>
<tr>
<td>wine</td>
<td>3.85</td>
</tr>
</tbody>
</table>

Ignore entire row in result?
Assignments, Creating new values…

```sparql
PREFIX ex: <http://example.org/>
SELECT ?Item (?Pr * 1.1 AS ?NewP )
WHERE { ?Item ex:price ?Pr }
```

### Data:
```
@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 .
ex:beer1 ex:price 3.
ex:wine1 ex:price 3.50 .
ex:liqueur1 ex:price "n/a".
```

### Results:
```
?Item    ?NewP
----------
lemonade 3.3
beer      3.3
wine      3.85
liqueur1
```

SPARQL 1.1: Leaves “errors” unbound!
Alternative to Project Expressions – BIND:

Same meaning, different syntax **BIND**…

*Note: BIND is evaluated in-place*

PREFIX ex: <http://example.org/>

SELECT ?Item ?NewP
    BIND (?Pr * 1.1 AS ?NewP )}

---

**Data:**

```prefix ex: <http://example.org/> .
ex:lemonad1 ex:price 3 .
ex:beer1 ex:price 3 .
ex:wine1 ex:price 3.50 .
ex:liqueur1 ex:price "n/a".
```

**Results:**

<table>
<thead>
<tr>
<th>?Item</th>
<th>?NewP</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemonade</td>
<td>3.3</td>
</tr>
<tr>
<td>beer</td>
<td>3.3</td>
</tr>
<tr>
<td>wine</td>
<td>3.85</td>
</tr>
<tr>
<td>liqueur1</td>
<td></td>
</tr>
</tbody>
</table>
Alternative to Project Expressions – BIND:

Same meaning, different syntax BIND…

Note: BIND is evaluated in-place, cf. http://www.w3.org/TR/sparql11-query/#bind

PREFIX ex: <http://example.org/>
SELECT ?Item ?NewP
WHERE { BIND (?Pr * 1.1 AS ?NewP)
}

Data:

@prefix ex: <http://example.org/> .

ex:lemonade1 ex:price 3 .
ex:beer1 ex:price 3 .
ex:wine1 ex:price 3.50 .
ex:liqueur1 ex:price "n/a".

Results:

<table>
<thead>
<tr>
<th>?Item</th>
<th>?NewP</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemonade</td>
<td></td>
</tr>
<tr>
<td>beer</td>
<td></td>
</tr>
<tr>
<td>wine</td>
<td></td>
</tr>
<tr>
<td>liqueur1</td>
<td></td>
</tr>
</tbody>
</table>
Assignments, Creating new values…

PREFIX ex: <http://example.org/>
SELECT ?Item (?Pr * 1.1 AS ?NewP )
WHERE { ?Item ex:price ?Pr }

Semantics:

extend(\(\mu\), var, expr) = \(\mu\) if var not in \(\text{dom}(\mu)\) and eval(expr) is an error

\(extend(\mu, \text{var}, \text{expr}) = \mu \cup \{ \text{var} \rightarrow \text{value} | \text{var} \text{ not in dom(\mu)} \text{ and value} = \text{eval(expr)} \text{ is defined}\}

extend(\(\mu\), var, expr) undefined if var in \(\text{dom}(\mu)\)

For sets of solutions:

extend(M , var, term) = \{\{ extend(\(\mu\), var, term) | \(\mu\) in M \}
Assignments, Creating new values...

```
PREFIX ex: <http://example.org/>
SELECT ?Item (?Pr * 1.1 AS ?NewP)
WHERE { ?Item ex:price ?Pr }
```

Semantics:

- $\text{extend}(\mu, \text{var, expr}) = \mu$ if var not in $\text{dom}(\mu)$ and eval(expr) is an error
- $\text{extend}(\mu, \text{var, expr}) = \mu \cup \{ \text{var} \rightarrow \text{value} | \text{var not in } \text{dom}(\mu) \text{ and value } = \text{eval(expr)} \text{ is defined} \}$
- $\text{extend}(\mu, \text{var, expr})$ undefined if var in $\text{dom}(\mu)$ i.e., this is syntactically disallowed.

For sets of solutions:

$\text{extend}(M, \text{var, term}) = \{ \{ \text{extend}(\mu, \text{var, term}) | \mu \in M \} \}$
“Count items”

PREFIX ex: <http://example.org/>
SELECT (Count(?Item) AS ?C)
WHERE { ?Item ex:price ?Pr }

Data:

```
@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 ;
    rdf:type ex:Softdrink.
ex:beer1 ex:price 3 ;
    rdf:type ex:Beer.
ex:wine1 ex:price 3.50 ;
    rdf:type ex:Wine.
ex:wine2 ex:price 4 .
    rdf:type ex:Wine.
ex:wine3 ex:price "n/a" ;
    rdf:type ex:Wine.
```

Results:

```
?C
5
```
Aggregates

“Count categories”

PREFIX ex: <http://example.org/>
SELECT (Count(?T) AS ?C)
WHERE { ?Item rdf:type ?T }

Data:

@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 ;
    rdf:type ex:Softdrink.
ex:beer1 ex:price 3 ;
    rdf:type ex:Beer.
ex:wine1 ex:price 3.50 ;
    rdf:type ex:Wine.
ex:wine2 ex:price 4 .
    rdf:type ex:Wine.
ex:wine3 ex:price "n/a" ;
    rdf:type ex:Wine.

Results:

?C 5
“Count categories”

```
PREFIX ex: <http://example.org/>
SELECT (Count(DISTINCT ?T) AS ?C) 
WHERE { ?Item rdf:type ?T }
```

Data:

```
@prefix ex: <http://example.org/> .

ex:lemonade1 ex:price 3 ;
    rdf:type ex:Softdrink.

ex:beer1 ex:price 3 ;
    rdf:type ex:Beer.

ex:wine1 ex:price 3.50 ;
    rdf:type ex:Wine.

ex:wine2 ex:price 4 ;
    rdf:type ex:Wine.

ex:wine3 ex:price "n/a" ;
    rdf:type ex:Wine.
```

Results:

```
?C 3
```
Aggregates - Grouping

“Count items per categories”

PREFIX ex: <http://example.org/>
SELECT ?T (Count(?Item) AS ?C)
WHERE { ?Item rdf:type ?T }
GROUP BY ?T

Data:

@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 ;
    rdf:type ex:Softdrink.
ex:beer1    ex:price 3 ;
    rdf:type ex:Beer.
ex:wine1    ex:price 3.50 ;
    rdf:type ex:Wine.
ex:wine2    ex:price 4 ;
    rdf:type ex:Wine.
ex:wine3    ex:price "n/a" ;
    rdf:type ex:Wine.

Results:

<table>
<thead>
<tr>
<th>?T</th>
<th>?C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softdrink</td>
<td>1</td>
</tr>
<tr>
<td>Beer</td>
<td>1</td>
</tr>
<tr>
<td>Wine</td>
<td>3</td>
</tr>
</tbody>
</table>
"Count items per categories, for those categories having more than one item”

PREFIX ex: <http://example.org/>
SELECT ?T (Count(?Item) AS ?C)
WHERE { ?Item rdf:type ?T }
GROUP BY ?T
HAVING Count(?Item) > 1

Data:

@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 ;
    rdf:type ex:Softdrink.
ex:beer1 ex:price 3 ;
    rdf:type ex:Beer.
ex:wine1 ex:price 3.50 ;
    rdf:type ex:Wine.
ex:wine2 ex:price 4 ;
    rdf:type ex:Wine.
ex:wine3 ex:price "n/a";
    rdf:type ex:Wine.

Results:

<table>
<thead>
<tr>
<th>?T</th>
<th>?C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wine</td>
<td>3</td>
</tr>
</tbody>
</table>
Other Aggregates

SUM ... as usual
AVG ... as usual
MIN ... as usual
MAX ... as usual
SAMPLE ... “pick” one non-deterministically
GROUP_CONCAT ... concatenate values with a
designated separator string

...this list is extensible ... new built-ins will need to define
error-behaviour, extra-parameters
(like SEPARATOR in GROUP_CONCAT)
Example SUM

“Sum Prices per categories”

PREFIX ex: <http://example.org/>
SELECT ?T (Sum(?Pr) AS ?P)
WHERE { ?Item rdf:type ?T; ex:price ?Pr }
GROUP BY ?T

Data:

@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 ;
    rdf:type ex:Softdrink.
ex:beer1 ex:price 3;
    rdf:type ex:Beer.
ex:wine1 ex:price 3.50 ;
    rdf:type ex:Wine.
ex:wine2 ex:price 4 .
    rdf:type ex:Wine.
ex:wine3 ex:price "n/a";
    rdf:type ex:Wine.

Results:

<table>
<thead>
<tr>
<th>?T</th>
<th>?C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softdrink</td>
<td>3</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Wine</td>
<td></td>
</tr>
</tbody>
</table>

Note:
Important to know that Sum/Avg, just delegates to numeric operations (sum uses “+”, etc., so errors, unbounds, non-numerics need special handling!)
Example SUM

“Sum Prices per categories”

PREFIX ex: <http://example.org/>
SELECT ?T (Sum(?Pr) AS ?P)
WHERE { ?Item rdf:type ?T; ex:price ?Pr
FILTER( isNumeric(?Pr) ) }
GROUP BY ?T

Data:
@prefix ex: <http://example.org/> .
ex:lemonade1 ex:price 3 ;
rdf:type ex:Softdrink.
ex:beer1 ex:price 3;
rdf:type ex:Beer.
ex:wine1 ex:price 3.50 ;
rdf:type ex:Wine.
ex:wine2 ex:price 4 .
rdf:type ex:Wine.
ex:wine3 ex:price "n/a";
rdf:type ex:Wine.

Results:

<table>
<thead>
<tr>
<th>?T</th>
<th>?C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softdrink</td>
<td>3</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Wine</td>
<td>7.5</td>
</tr>
</tbody>
</table>
Example SUM

“Sum Prices per categories”

PREFIX ex: <http://example.org/>

SELECT ?T (Sum(COALESCE(xs:decimal(?Pr),0) AS ?C)
WHERE { ?Item rdf:type ?T; ex:price ?Pr }
GROUP BY ?T

<table>
<thead>
<tr>
<th>?T</th>
<th>?C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softdrink</td>
<td>3</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Wine</td>
<td>7.5</td>
</tr>
</tbody>
</table>

Note:
Important to know that Sum/Avg, just delegates to numeric operations (sum uses “+”, etc., so errors, unbounds, non-numerics need special handling!

Data:

@prefix ex: <http://example.org/> .

ex:lemonade1 ex:price 3 ;
   rdf:type ex:Softdrink.
ex:beer1 ex:price 3 ;
   rdf:type ex:Beer.
ex:wine1 ex:price 3.50 ;
   rdf:type ex:Wine.
ex:wine2 ex:price 4 .
   rdf:type ex:Wine.
ex:wine3 ex:price "n/a";
   rdf:type ex:Wine.
Example SUM

"Sum Prices per categories"

PREFIX ex: <http://example.org/>
SELECT ?T (Sum(IF(isNumeric(?Pr),?Pr,0) AS ?P) AS ?P)
WHERE { ?Item rdf:type ?T; ex:price ?Pr }
GROUP BY ?T

Note:
Important to know that Sum/Avg, just delegates to numeric operations (sum uses “+”, etc., so errors, unbounds, non-numerics need special handling!

Data:

@prefix ex: <http://example.org/> .

ex:lemonade1 ex:price 3 ;
    rdf:type ex:Softdrink.
ex:beer1 ex:price 3;
    rdf:type ex:Beer.
ex:wine1 ex:price 3.50 ;
    rdf:type ex:Wine.
ex:wine2 ex:price 4 .
    rdf:type ex:Wine.
ex:wine3 ex:price "n/a";
    rdf:type ex:Wine.

Results:

<table>
<thead>
<tr>
<th>?T</th>
<th>?C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softdrink</td>
<td>3</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Wine</td>
<td>7.5</td>
</tr>
</tbody>
</table>
Example GROUP_CONCAT, SAMPLE

“pick one sample name per person, plus a concatenated list of nicknames”

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ( SAMPLE(?N) as ?Name)
( GROUP_CONCAT(?M; SEPARATOR = "", "") AS ?Nicknames )
WHERE { ?P a foaf:Person ;
  foaf:name ?N ;
  foaf:nick ?M . }
GROUP BY ?P

@prefix ex: <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:alice a foaf:Person; foaf:name "Alice Wonderland";
  foaf:nick "Alice", "The real Alice".

ex:bob a foaf:Person;
  foaf:name "Robert Doe", "Robert Charles Doe",
  "Robert C. Doe";
  foaf:nick "Bob","Bobby","RobC","BobDoe".

ex:charles a foaf:Person;
  foaf:name "Charles Charles";
  foaf:nick "Charlie".

<table>
<thead>
<tr>
<th>Name</th>
<th>Nicknames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice Wonderland</td>
<td>The real Alice, Alice</td>
</tr>
<tr>
<td>Charles Charles</td>
<td>Charlie</td>
</tr>
<tr>
<td>Robert C. Doe</td>
<td>Bob, BobDoe, RobC, Bobby</td>
</tr>
</tbody>
</table>
Evaluate a list of (GROUP BY) expressions:

\[ \text{ListEval(ExprList, } \mu) \text{ returns a list } E, \text{ where } E[i] = \mu( \text{ExprList[i]} ) \]

Use these to partition a solution sequence:

\[ \text{Group((), } \Omega) = \{ 1 \rightarrow \Omega \} \]

\[ \text{Group(ExprList, } \Omega) = \{ \text{ListEval(ExprList, } \mu) \rightarrow \{ \mu' \mid \mu' \text{ in } \Omega, \text{ListEval(ExprList, } \mu) = \text{ListEval(ExprList, } \mu') \} \mid \mu \text{ in } \Omega \} \]

produces a *grouped solution sequence*

```
SELECT Sum(\(?y\)) AS \(?Sy\)
WHERE \{ :s :p \(?x\); :q \(?y\) \}
GROUP BY \(?x\)
```

Assume solution sequence \( S = \{ \{?x\rightarrow2, \ ?y\rightarrow3\}, \{?x\rightarrow2, \ ?y\rightarrow5\}, \{?x\rightarrow6, \ ?y\rightarrow7\}\} \),

\[ \text{Group(}(?x), S) = \{ (2) \rightarrow ( \{?x\rightarrow2, \ ?y\rightarrow3\}, \{?x\rightarrow2, \ ?y\rightarrow5\} ), \ (6) \rightarrow ( \{?x\rightarrow6, \ ?y\rightarrow7\} \} \} \]
Definition: Aggregation (simplified)

Aggregation applies set function “func” (e.g. sum, min, max, …) to the multiset obtained from applying a list of expressions to a grouped solution sequence, G as produced by the Group() function. It produces a single value for each key and partition for that key (key, X).

\[
\text{Aggregation(ExprList, func, G)} = \{ g \rightarrow F(\Omega) \mid g \rightarrow \Omega \text{ in } G \} \\
\text{where} \quad \text{M(}\Omega\text{)} = \{ \text{ListEval(ExprList, } \mu \mid \mu \text{ in } \Omega) \} \\
\text{F(}\Omega\text{)} = \text{func(M(}\Omega\text{)), for non-}\text{DISTINCT} \\
\text{F(}\Omega\text{)} = \text{func(Distinct(M(}\Omega\text{))), for DISTINCT}
\]

\[
G = \{ (2) \rightarrow (\{?x \rightarrow 2, ?y \rightarrow 3\}, \{?x \rightarrow 2, ?y \rightarrow 5\} ), \\
(6) \rightarrow (\{?x \rightarrow 6, ?y \rightarrow 7\}) \}
\]

Aggregation( ?y, Sum, G ) =  \{ (2) \rightarrow \text{Sum( (3), (5) )}, (6) \rightarrow \text{Sum( (7) )} \} \\
=  \{ (2) \rightarrow 8, (6) \rightarrow 7 \}

Aggregates - Semantics

Definition: Aggregation (simplified)

Aggregation applies set function “func” (e.g. sum, min, max, …) to the multiset obtained from applying a list of expressions to a grouped solution sequence, G as produced by the Group() function. It produces a single value for each key and partition for that key (key, X).

\[
\text{Aggregation(ExprList, func, G)} = \{ g \rightarrow F(\Omega) \mid g \rightarrow \Omega \text{ in } G \}
\]

where

\[
\begin{align*}
M(\Omega) &= \{ \text{ListEval(ExprList, } \mu) \mid \mu \text{ in } \Omega) \} \\
F(\Omega) &= \text{func}(M(\Omega)), \text{ for non-DISTINCT} \\
F(\Omega) &= \text{func}(\text{Distinct}(M(\Omega))), \text{ for DISTINCT}
\end{align*}
\]

G =
\[
\begin{align*}
\{ & (2) \rightarrow ( \{ ?x \rightarrow 2, ?y \rightarrow 3 \}, \{ ?x \rightarrow 2, ?y \rightarrow 3 \} ), \\
& (6) \rightarrow ( \{ ?x \rightarrow 6, ?y \rightarrow 7 \} ) \}
\end{align*}
\]

Aggregation( ?y, Sum, G ) =
\[
\begin{align*}
\{ & (2) \rightarrow \text{Sum( (3), (3) )}, (6) \rightarrow \text{Sum( (7) )} \}
\end{align*}
\]

= \{ (2) \rightarrow 6, (6) \rightarrow 7 \}
Definition: Aggregation (simplified)

Aggregation applies set function “func” (e.g. sum, min, max, …) to the multiset obtained from applying a list of expressions to a grouped solution sequence, G as produced by the Group() function. It produces a single value for each key and partition for that key (key, X).

Aggregation(ExprList, func, G) = \{ g \rightarrow F(\Omega) | g \rightarrow \Omega \text{ in } G \}

where 
- \( M(\Omega) = \{ \text{ListEval(ExprList, } \mu | \mu \text{ in } \Omega) \} \)
- \( F(\Omega) = \text{func}(M(\Omega)), \text{ for non-\text{DISTINCT}} \)
- \( F(\Omega) = \text{func}(\text{Distinct}(M(\Omega))), \text{ for DISTINCT} \)

\[ G = \{ (2) \rightarrow (\{?x\rightarrow2, ?y\rightarrow3\}, \{?x\rightarrow2, ?y\rightarrow3\} ), (6) \rightarrow (\{?x\rightarrow6, ?y\rightarrow7\}) \} \]

Aggregation( ?y, Sum(DISTINCT), G ) = \{ (2) \rightarrow \text{Sum(DISTINCT( (3), (3) ))}, (6) \rightarrow \text{Sum( DISTINCT(((7))))} \}

= \{ (2) \rightarrow 3, (6) \rightarrow 7 \}
Definition: Aggregation (simplified)

Aggregation applies set function “func” (e.g. sum, min, max, …) to the multiset obtained from applying a list of expressions to a grouped solution sequence, G as produced by the Group() function. It produces a single value for each key and partition for that key (key, X).

\[
\text{Aggregation} (\text{ExprList}, \text{func}, G) = \{ g \rightarrow F(\Omega) \mid g \rightarrow \Omega \text{ in } G \}
\]

where

\[
\begin{align*}
M(\Omega) &= \{ \text{ListEvalE(ExprList, } \mu) \mid \mu \text{ in } \Omega) \} \\
F(\Omega) &= \text{func}(M(\Omega)), \text{ for non-DISTINCT} \\
F(\Omega) &= \text{func}(\text{Distinct}(M(\Omega))), \text{ for DISTINCT}
\end{align*}
\]

Aggregations are subsequently mapped back via to solution multisets in the evaluation of SELECT expressions, cf. [http://www.w3.org/TR/sparql11-query/#sparqlSelectExpressions](http://www.w3.org/TR/sparql11-query/#sparqlSelectExpressions)

\[
G = \{ (2) \rightarrow (\{?x\rightarrow2, ?y\rightarrow3\}, \{?x\rightarrow2, ?y\rightarrow5\} ), \\
(6) \rightarrow (\{?x\rightarrow6, ?y\rightarrow7\} ) \}
\]

\[
\text{Aggregation( ?y, Sum, G )} = \{ (2) \rightarrow \text{Sum}( (3), (5) ) , (6) \rightarrow \text{Sum}( (7) ) \} \\
= \{ (2) \rightarrow 8, (6) \rightarrow 7 \}
\]

\[
\text{SELECT} \ ?x \ (\text{Sum(?y)} \ AS \ ?Sy) \\
\text{WHERE} \ \{ :s :p \ ?x; :q \ ?y \} \\
\text{GROUP BY} \ ?x
\]

\[
\{ \ ?x \rightarrow 2 , ?Sy \rightarrow 8 \}, \{?x\rightarrow6, ?Sy\rightarrow7\} \}
\]
Subqueries to realise complex mappings

- How to concatenate first name and last name?
- Wasn’t possible in SPARQL 1.0 … Now possible without problems per subqueries!

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>

CONSTRUCT{ ?P foaf:name ?FullName }  
WHERE { 
  WHERE { ?P foaf:firstName ?F ; foaf:lastName ?L. }  
}
Subqueries “Limit per resource”

Give me all titles of papers of 10 persons who co-authored with Tim Berners-Lee

```
SELECT ?T
WHERE {
  {
    SELECT DISTINCT ?P
      FILTER ( ?P != <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> )
    }
    LIMIT 10
  }
}
```
Subqueries - Semantics

Note: Before Solution Modifiers are applied, SPARQL semantics converts solution multisets to solution sequences

SELECT ?T
WHERE {
}

SELECT DISTINCT ?P
WHERE {
  FILTER ( ?P != <http://dblp.org/Tim_Berners-Lee> )
}
ORDER BY ?P
LIMIT 10

Subqueries require one additional algebra operator, toMultiset, which takes Sequences and returns Multisets
MINUS and NOT EXISTS
Negation as failure in SPARQL1.0 is “ugly”:

```
SELECT ?X
WHERE { ?X rdf:type foaf:Person
    MINUS { ?X foaf:homepage ?H } }
```

SPARQL1.1 has two alternatives to do the same

- NOT EXISTS in FILTERs
  - detect non-existence
- (P1 MINUS P2) as a new binary operator
  - Remove rows with matching bindings
  - only effective when P1 and P2 share variables

subtle difference, not relevant for most queries… but let’s look into it
MINUS and NOT EXISTS

May have different results, e.g.:

PREFIX ex: <http://example.org/>

SELECT *
WHERE{ ?S ?P ?O 
FILTER( NOT EXISTS { ex:a ex:b ex:c } ) }

@prefix ex: <http://example.org/> .

ex:a ex:b ex:c
MINUS and NOT EXISTS

May have different results, e.g.:

PREFIX ex: <http://example.org/>

SELECT *
WHERE{ ?S ?P ?O

MINUS { ex:a ex:b ex:c } }

@prefix ex: <http://example.org/> .

ex:a ex:b ex:c
Property Path Expressions
Property Path Expressions

Arbitrary Length paths, Concatenate property paths, etc.
E.g. names of people Tim Berners-Lee transitively co-authored papers with…

```sparql
SELECT DISTINCT ?N
WHERE {<http://dblp.../Tim_Berners-Lee>

    (^foaf:maker/foaf:maker)+/foaf:name ?N

}
```
## Path Expressions: full list of operators

### elt ... Path Element

<table>
<thead>
<tr>
<th>Syntax Form</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>iri</td>
<td>An IRI. A path of length one.</td>
</tr>
<tr>
<td>^elt</td>
<td>Inverse path (object to subject).</td>
</tr>
<tr>
<td>elt&lt;sub&gt;1&lt;/sub&gt; / elt&lt;sub&gt;2&lt;/sub&gt;</td>
<td>A sequence path of elt&lt;sub&gt;1&lt;/sub&gt; followed by elt&lt;sub&gt;2&lt;/sub&gt;.</td>
</tr>
<tr>
<td>elt&lt;sub&gt;1&lt;/sub&gt;</td>
<td>elt&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>elt*</td>
<td>A path that connects the subject and object of the path by zero or more occurrences of elt.</td>
</tr>
<tr>
<td>elt+</td>
<td>A path that connects the subject and object of the path by one or more occurrences of elt.</td>
</tr>
<tr>
<td>elt?</td>
<td>A path that connects the subject and object of the path by zero or one occurrences of elt.</td>
</tr>
<tr>
<td>!iri or ! (iri&lt;sub&gt;1&lt;/sub&gt;</td>
<td>...</td>
</tr>
<tr>
<td>!^iri or ! (^iri&lt;sub&gt;1&lt;/sub&gt;</td>
<td>...</td>
</tr>
<tr>
<td>!(iri&lt;sub&gt;1&lt;/sub&gt;</td>
<td>...</td>
</tr>
<tr>
<td>(elt)</td>
<td>A group path elt, brackets control precedence.</td>
</tr>
</tbody>
</table>
Semantics defined mostly in terms of rewriting:

/ ... rewrites to a sequence of patterns
| ... rewrites to UNION
^ ... rewrites to inverted path
? ... new algebra function ZeroOrOnePath()
* ... new algebra function ZeroOrMorePath()
+ ... new algebra function OneOrMorePath()

A lot of last-minute discussion about semantics (counting vs. non-counting) see also [Arenas, Conca, Pérez, WWW2012] and [Losemann, Martens, PODS2012] → Detailed presentation of Property Paths and their semantics: possible topic for a student presentation!
Many new functions as opposed to SPARQL1.0:

Mentioned a few already:
- coalesce
- if
- isNumeric

Many new functions for strings, e.g. strbefore(), strafter(), …

See full list (snapshot) at:
http://www.w3.org/2009/sparql/docs/query-1.1/rq25.xml#SparqlOps
Goals of SPARQL1.1

List of agreed features:

Additions to the Query Language:
- Project Expressions
- Aggregate functions
- Subqueries
- Negation
- Property Paths (time permitting)
- Extend the function library (time permitting)
- Basic federated Queries (time permitting)

Entailment (time permitting)

SPARQL Update
- Full Update language
- plus simple RESTful update methods for RDF graphs (HTTP methods)

Service Description
- Method for discovering a SPARQL endpoint’s capabilities
- Summary of its data

We will focus on these in this lecture
SPARQL Basic Federated Query

Allows you to query a remote endpoint from “WITHIN” your query…

Keyword SERVICE

Can be used e.g. to compute aggregates from an endpoint that doesn’t yet support them, e.g. SPARQL 1.1 for dbpedia, e.g. “How many inhabitants do Austria’s top-3 cities have in total (sum)?”

Using ARQ:

```
SELECT (SUM(?pop) AS ?P )
{ SERVICE <http://dbpedia.org/sparql/>
    { SELECT DISTINCT ?C ?pop
        WHERE {
            ?C <http://dbpedia.org/ontology/populationTotal> ?pop ;
            <http://dbpedia.org/ontology/country> <http://dbpedia.org/resource/Austria> .
        }
        ORDER BY DESC ( ?pop )
        LIMIT 3
    }
}
```
SPARQL 1.1 Entailment
SPARQL 1.1 Entailment:
Example where Reasoning is needed

Give me all facts about Tim Berners-Lee from DBPedia and DBLP?

```
SELECT ?P ?O
```

If I ask this query to DBPedia, I get quite some results…
… but not if I ask the same query to DBLP.

Because:

a) DBLP does not “know” that
http://dbpedia.org/resource/Tim_Berners-Lee
=
http://dblp.l3s.de/d2r/page/authors/Tim_Berners-Lee

b) SPARQL can’t follow links (more on that in the one of the next lectures)
SPARQL 1.1 Entailment: OWL

SPARQL 1.1 “understands” OWL:

```sparql
SELECT ?O
FROM <dbpedia.org>
FROM <dblp.l3s.de>
WHERE { <http://dbpedia.org/resource/Tim_Berners-Lee> foaf:name ?O }
```

*dbpedia.org*

```sparql
<http://dbpedia.org/resource/Tim_Berners-Lee>
  foaf:homepage
    <http://www.w3.org/People/Berners-Lee> .
```

*dblp.l3s.de*

```sparql
<http://dblp.l3s.de/d2r/page/authors/Tim_Berners-Lee>
  foaf:homepage
    <http://www.w3.org/People/Berners-Lee> ;
    foaf:name "Tim Berners-Lee".
```
SPARQL 1.1 Entailment: OWL

SPARQL 1.1 “understands” OWL:

```sparql
PREFIX foaf: <xmlns.com/foaf/1.0>
PREFIX owl: <xmlns.com/foaf/0.1>

SELECT ?O
FROM <dbpedia.org>
FROM <dblp.l3s.de>
WHERE { <http://dbpedia.org/resource/Tim_Berners-Lee> foaf:name ?O }
```

```
<Tim Berners-Lee>
```

```
“Tim Berners-Lee”
```
SPARQL 1.1 Entailment: OWL

SPARQL 1.1 “understands” OWL:

```
foaf:homepage a owl:InverseFunctionalProperty .

SELECT ?O
FROM <dbpedia.org>
FROM <dblp.l3s.de>
WHERE { <http://dbpedia.org/resource/Tim_Berners-Lee> foaf:name ?O }
```

```
?O

“Tim Berners-Lee”
```
SPARQL 1.1 Entailment: OWL

Defines which answers an OWL or RDF Schema-aware SPARQL engine should return ... a bit more on that in the next lecture, but also a possible topic for student presentation!
SQL has not only a query language, but also a Data manipulation language. 
→ SPARQL Update to fill this gap:

```
PREFIX ex: <http://example.org/>

DELETE { ?Item ex:price ?Pr }
INSERT { ?Item ex:price ?NewPr }
WHERE { ?Item ex:price ?Pr
          BIND (?Pr * 1.1 AS ?NewPr ) }
```

→ Allows to change/update an RDF Store from outside, again via standard HTTP protocol.

*Note: security issues are a separate issue, not prescribed yet by the standard!*
Some current (partial) SPARQL 1.1 implementations:

Jena ARQ
- [http://sourceforge.net/projects/jena/](http://sourceforge.net/projects/jena/)
- [http://sparql.org/sparql.html](http://sparql.org/sparql.html)

OpenAnzo
- [http://www.openanzo.org/](http://www.openanzo.org/)

Perl RDF
- [http://github.com/kasei/perlrdf/](http://github.com/kasei/perlrdf/)

Corese

etc.

Others probably forthcoming… Virtuoso (e.g. dbpedia) seems to support most of SPARQL 1.1 already.
References

Find all SPARQL 1.1 Drafts here: http://www.w3.org/2009/sparql/wiki/Main_Page

Papers:


[Arenas, Conca, Pérez, WWW2012] Marcelo Arenas, Sebastián Conca, Jorge Pérez: Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. WWW 2012: 629-638


[Fionda et al., WWW2012] Valeria Fionda, Claudio Gutierrez, Giuseppe Pirrò: Semantic navigation on the web of data: specification of routes, web fragments and actions. WWW 2012: 281-290
Assignment 2:

ATTENTION: If you have NOT received an email with feedback, I have NOT received your assignment!

Discuss Assignment 2 and common problems:

What is RDF Entailment (as opposed to Simple Entailment)?

Don’t interpret things into the OWL ontology that aren’t said there!

ill-typed literals alone don’t cause D-inconsistency

provide some examples of OWL inconsistencies

Time allowed:
If you have your solutions here, or have read my feedback already, we can also go through it.
Assignment 3:


Deadline: 31 May 2013

**ATTENTION:** Pick a topic for the final presentation until next time!
Student presentations:

I have time to discuss your proposals still, if you have some already, otherwise, more topics by next time.

Who has sent me a topic suggestion already?
- SPARQL GUIs (F. J. Ekaputra)
- Good Relations Ontology and use (B. Ege)

Who plans to still do?
Some suggested topics (which we can assign now already – first come, first serve:

- W3C RDF1.1 WG – status semantics, changes, semantics for named graphs, etc.
- W3C Linked Data Platform WG – [http://www.w3.org/2012/ldp/wiki/Main_Page](http://www.w3.org/2012/ldp/wiki/Main_Page)
- SPARQL1.1 spec parts which we don’t cover in detail (e.g. Entailment Regimes, Update, Semantics of Property Paths, etc.) - Jonas
- OWL2 and meta-modeling
- SKOS
- RDFa & schema.org
- Berlin SPARQL Benchmark, latest edition, see [http://lists.w3.org/Archives/Public/semantic-web/2013Apr/0237.html](http://lists.w3.org/Archives/Public/semantic-web/2013Apr/0237.html)

More own topics suggestions welcome!
Presentations

First slot: 24/06/2013
Second slot: 25/06/2013

Send me the slides at least 1 week in advance per email!
→ You should start to work on the topic soon!