Formal Introduction to SPARQL

Claudio Gutierrez
(Joint work with Marcelo Arenas and Jorge Perez)

Department of Computer Science
Universidad de Chile

Center for Web Research
http://www.cwr.cl

UPM - Madrid, Jan 2009

SPARQL: A simple RDF query language

```
SELECT ?Name ?Email
WHERE
{
  ?X :name ?Name
  ?X :email ?Email
}
```

- The *semantics* of simple SPARQL queries is easy to understand.

SPARQL: A simple RDF query language

```
SELECT ?Name ?Email
WHERE
{
  ?X :name ?Name
  ?X :email ?Email
}
```

- The *semantics* of simple SPARQL queries is easy to understand.

  > *"Give me the name and email of the resources in the datasource"*

But things can become more complex...

Interesting features of pattern matching on graphs

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

```
{ P1
  P2 }
```

But things can become more complex...

Interesting features of pattern matching on graphs

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

```
{ { P1
    P2 }


  { P3
    P4 }


}
```

But things can become more complex...

Interesting features of pattern matching on graphs

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

```
{ { 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
- Union of patterns
- Filtering
- .....

```
{ { 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
- Filtering
- .....

```
{ { 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
- .....

```
{ { 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
- .....

```
{ { P1
    P2
    OPTIONAL { P5 }  }

  { P3
    P4
    OPTIONAL { P7
      OPTIONAL { P8 }  }  }
}
UNION
{ P9
  FILTER ( R ) }
```

A formal semantics for SPARQL

A formal semantics for SPARQL

A formal approach is beneficial:

- Clarifies corner cases
- Helps in the implementation process
- Provides sound foundations

A formal semantics for SPARQL

A formal approach is beneficial:

- Clarifies corner cases
- Helps in the implementation process
- Provides sound foundations

We will present:

- A formal compositional semantics based on **[PAG06: Semantics and Complexity of SPARQL]**
- This formalization is the starting point of the official semantics of the SPARQL language by the W3C.
- Differences with W3C semantics are neglibible.
- Simple, clean, correct.

Outline

First of all, a simplified algebraic syntax

- Triple patterns: RDF triple + variables (no bnodes for now)

$$(?X, \text{name}, ?Name)$$

First of all, a simplified algebraic syntax

- Triple patterns: RDF triple + variables (no bnodes for now)

$$(?X, \text{name}, ?\mathit{Name})$$

- Basic graph pattern (BGP): a set of triple patterns

$$\{t_1, t_2, \ldots, t_k\}.$$

This is the base case of the algebra of patterns.

First of all, a simplified algebraic syntax

- Triple patterns: RDF triple + variables (no bnodes for now)

$$(?X, \text{name}, ?Name)$$

- Basic graph pattern (BGP): a set of triple patterns

$$\{t_1, t_2, \ldots, t_k\}.$$

This is the base case of the algebra of patterns.

Example

$$\{\ (?X, \text{name}, ?Name),\ \ (?X, \text{email}, ?Email)\ \}$$

First of all, a simplified algebraic syntax (cont.)

- There are three basic binary operators:

  AND, UNION, OPT.

- They are used to construct graph pattern expressions from basic graph patterns.

First of all, a simplified algebraic syntax (cont.)

- There are three basic binary operators:

  AND, UNION, OPT.

- They are used to construct graph pattern expressions from basic graph patterns.
- A SPARQL graph pattern is an expression built from the basic operators. Example:

  $$(((\{t_1, t_2\} \text{ AND } t_3) \text{ OPT } \{t_4, t_5\}) \text{ AND } (t_6 \text{ UNION } \{t_7, t_8\}))$$

  Note that it is a full parenthesized expression

First of all, a simplified algebraic syntax (cont.)

- There are three basic binary operators:

  AND, UNION, OPT.

- They are used to construct graph pattern expressions from basic graph patterns.
- A SPARQL graph pattern is an expression built from the basic operators. Example:

  $$(((\{t_1, t_2\} \text{ AND } t_3) \text{ OPT } \{t_4, t_5\}) \text{ AND } (t_6 \text{ UNION } \{t_7, t_8\}))$$

  Note that it is a full parenthesized expression

- Full parenthesized expressions give us explicit precedence/association.

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Given a mapping μ and a basic graph pattern P:

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Given a mapping μ and a basic graph pattern P:

- dom(μ): the domain of μ.

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Given a mapping μ and a basic graph pattern P:

- $\mathrm{dom}(\mu)$: the domain of μ.
- $\mu(P)$: the set obtained from P replacing the variables according to μ

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Given a mapping μ and a basic graph pattern P:

- dom(μ): the domain of μ.
- $\mu(P)$: the set obtained from P replacing the variables according to μ

Example

$$\mu = \{?X \to R_1, ?Y \to R_2, ?Name \to \text{john}, ?Email \to \text{J@ed.ex}\}$$

$$P = \{(?X, \text{name}, ?Name), (?X, \text{email}, ?Email)\}$$

$$\mu(P) = \{(R_1, \text{name}, \text{john}), (R_1, \text{email}, \text{J@ed.ex})\}$$

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Given a mapping μ and a basic graph pattern P:

- dom(μ): the domain of μ.
- $\mu(P)$: the set obtained from P replacing the variables according to μ

Example

$$\mu = \{?X \rightarrow R_1, ?Y \rightarrow R_2, ?Name \rightarrow \text{john}, ?Email \rightarrow \text{J@ed.ex}\}$$

$$P = \{(?X, \text{name}, ?Name), (?X, \text{email}, ?Email)\}$$

$$\mu(P) = \{(R_1, \text{name}, \text{john}), (R_1, \text{email}, \text{J@ed.ex})\}$$

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Given a mapping μ and a basic graph pattern P:

- dom(μ): the domain of μ.
- $\mu(P)$: the set obtained from P replacing the variables according to μ

Example

$$\mu = \{?X \rightarrow R_1, ?Y \rightarrow R_2, ?\mathit{Name} \rightarrow \text{john}, ?\mathit{Email} \rightarrow \text{J@ed.ex}\}$$

$$P = \{(?X, \text{name}, ?\mathit{Name}), (?X, \text{email}, ?\mathit{Email})\}$$

$$\mu(P) = \{(R_1, \text{name}, \text{john}), (R_1, \text{email}, \text{J@ed.ex})\}$$

Mappings: building block for the semantics

Definition

A mapping is a partial function from variables to RDF terms.

Given a mapping μ and a basic graph pattern P:

- dom(μ): the domain of μ.
- $\mu(P)$: the set obtained from P replacing the variables according to μ

Example

$$\mu = \{?X \to R_1, ?Y \to R_2, ?Name \to \text{john}, \; ?Email \to \text{J@ed.ex}\}$$

$$P = \{(?X, \text{ name}, \; ?Name), \; (?X, \text{ email}, \; ?Email)\}$$

$$\mu(P) = \{(R_1, \text{ name, john}), (R_1, \text{ email, J@ed.ex})\}$$

The semantics of basic graph pattern

Definition

The evaluation of the BGP P over a graph G, denoted by $[\![P]\!]_G$, is the set of all mappings μ such that:

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

The semantics of basic graph pattern

Definition

The evaluation of the BGP P over a graph G, denoted by $[[P]]_G$, is the set of all mappings μ such that:

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

The semantics of basic graph pattern

Definition

The evaluation of the BGP P over a graph G, denoted by $[\![P]\!]_G$, is the set of all mappings μ such that:

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

Example

G

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

$$[\![\{(?X, \text{name}, ?Y)\}]\!]_G$$

Example

$$G$$

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

$$[\![\{(?X, \text{name}, ?Y)\}]\!]_G$$

$$\left\{ \begin{array}{l} \mu_1 = \{?X \rightarrow R_1, ?Y \rightarrow \text{john}\} \\ \mu_2 = \{?X \rightarrow R_2, ?Y \rightarrow \text{paul}\} \end{array} \right\}$$

Example

$$G$$

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

$$[\![\{(?X, \text{name}, ?Y)\}]\!]_G$$

$$\left\{ \begin{array}{l} \mu_1 = \{?X \to R_1, ?Y \to \text{john}\} \\ \mu_2 = \{?X \to R_2, ?Y \to \text{paul}\} \end{array} \right\}$$

$$[\![\{(?X, \text{name}, ?Y), (?X, \text{email}, ?E)\}]\!]_G$$

Example

$$G$$

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

$$[\![\{(?X, \text{name}, ?Y)\}]\!]_G$$

$$\left\{ \begin{array}{l} \mu_1 = \{?X \to R_1, ?Y \to \text{john}\} \\ \mu_2 = \{?X \to R_2, ?Y \to \text{paul}\} \end{array} \right\}$$

$$[\![\{(?X, \text{name}, ?Y), (?X, \text{email}, ?E)\}]\!]_G$$

$$\left\{ \ \mu = \{?X \to R_1, ?Y \to \text{john}, ?E \to \text{J@ed.ex}\} \ \right\}$$

Example

G

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

$$[\![\{(?X, \text{name}, ?Y)\}]\!]_G$$

	$?X$	$?Y$
μ_1	R_1	john
μ_2	R_2	paul

$$[\![\{(?X, \text{name}, ?Y), (?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?Y$	$?E$
μ	R_1	john	J@ed.ex

Example

G

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

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

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

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

$[\![\{\ \}]\!]_G$

Example

$$G$$

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

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

$\{\quad\}$

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

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

$[\![\{\ \}]\!]_G$

Example

$$G$$

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

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

$\{ \quad \}$

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

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

$\{ \quad \}$

$[\![\{ \ \}]\!]_G$

Example

$$G$$

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

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

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

$$[\![\{(R_2, \text{name}, \text{paul})\}]\!]_G$$
$$\{\ \mu_\emptyset = \{\ \}\ \}$$

$$[\![\{\ \}]\!]_G$$

Example

G

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

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

$\{\quad\}$

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

$\{\quad\}$

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

$\{\ \mu_\emptyset = \{\ \}\ \}$

$[\![\{\ \}]\!]_G$

$\{\ \mu_\emptyset = \{\ \}\ \}$

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Example

	$?X$	$?Y$	$?U$	$?V$
μ_1	R_1	john		
μ_2	R_1		J@edu.ex	
μ_3			P@edu.ex	R_2

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Example

	$?X$	$?Y$	$?U$	$?V$
μ_1	R_1	john		
μ_2	R_1		J@edu.ex	
μ_3			P@edu.ex	R_2

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Example

	$?X$	$?Y$	$?U$	$?V$
μ_1	R_1	john		
μ_2	R_1		J@edu.ex	
μ_3			P@edu.ex	R_2
$\mu_1 \cup \mu_2$	R_1	john	J@edu.ex	

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Example

	$?X$	$?Y$	$?U$	$?V$
μ_1	R_1	john		
μ_2	R_1		J@edu.ex	
μ_3			P@edu.ex	R_2
$\mu_1 \cup \mu_2$	R_1	john	J@edu.ex	

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Example

	$?X$	$?Y$	$?U$	$?V$
μ_1	R_1	john		
μ_2	R_1		J@edu.ex	
μ_3			P@edu.ex	R_2
$\mu_1 \cup \mu_2$	R_1	john	J@edu.ex	
$\mu_1 \cup \mu_3$	R_1	john	P@edu.ex	R_2

Compatible mappings: mappings that can be merged.

Definition

The mappings μ_1, μ_2 are compatibles iff they agree in their shared variables:

- $\mu_1(?X) = \mu_2(?X)$ for every $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$.

$\mu_1 \cup \mu_2$ is also a mapping.

Example

	$?X$	$?Y$	$?U$	$?V$
μ_1	R_1	john		
μ_2	R_1		J@edu.ex	
μ_3			P@edu.ex	R_2
$\mu_1 \cup \mu_2$	R_1	john	J@edu.ex	
$\mu_1 \cup \mu_3$	R_1	john	P@edu.ex	R_2

$\mu_\emptyset = \{\ \}$ is compatible with every mapping.

Sets of mappings and operations

Let M_1 and M_2 be sets of mappings:

Definition

Join: $M_1 \bowtie M_2$

- $\{\mu_1 \cup \mu_2 \mid \mu_1 \in M_1, \mu_2 \in M_2,$ and μ_1, μ_2 are compatibles$\}$
- extending mappings in M_1 with compatible mappings in M_2

will be used to define AND

Sets of mappings and operations

Let M_1 and M_2 be sets of mappings:

Definition

Join: $M_1 \bowtie M_2$

- $\{\mu_1 \cup \mu_2 \mid \mu_1 \in M_1, \mu_2 \in M_2,$ and μ_1, μ_2 are compatibles$\}$
- extending mappings in M_1 with compatible mappings in M_2

will be used to define AND

Definition

Union: $M_1 \cup M_2$

- $\{\mu \mid \mu \in M_1$ or $\mu \in M_2\}$
- mappings in M_1 plus mappings in M_2 (the usual set union)

will be used to define UNION

Sets of mappings and operations

Definition

Difference: $M_1 \setminus M_2$

- $\{\mu \in M_1 \mid \text{ for all } \mu' \in M_2,\ \mu \text{ and } \mu' \text{ are not compatibles}\}$
- mappings in M_1 that cannot be extended with mappings in M_2

Sets of mappings and operations

Definition

Difference: $M_1 \setminus M_2$

- $\{\mu \in M_1 \mid \text{ for all } \mu' \in M_2,\ \mu \text{ and } \mu' \text{ are not compatibles}\}$
- mappings in M_1 that cannot be extended with mappings in M_2

Definition

Left outer join: $M_1 ⟕ M_2 = (M_1 \bowtie M_2) \cup (M_1 \setminus M_2)$

- extension of mappings in M_1 with compatible mappings in M_2
- plus the mappings in M_1 that cannot be extended.

will be used to define OPT

Semantics of general graph patterns

Definition

Given a graph G the evaluation of a pattern is recursively defined

the base case is the evaluation of a BGP.

Semantics of general graph patterns

Definition

Given a graph G the evaluation of a pattern is recursively defined

- $[\![(P_1 \text{ AND } P_2)]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$

the base case is the evaluation of a BGP.

Semantics of general graph patterns

Definition

Given a graph G the evaluation of a pattern is recursively defined

- $[\![(P_1 \text{ AND } P_2)]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$
- $[\![(P_1 \text{ UNION } P_2)]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$

the base case is the evaluation of a BGP.

Semantics of general graph patterns

Definition

Given a graph G the evaluation of a pattern is recursively defined

- $[[(P_1 \text{ AND } P_2)]]_G = [[P_1]]_G \bowtie [[P_2]]_G$
- $[[(P_1 \text{ UNION } P_2)]]_G = [[P_1]]_G \cup [[P_2]]_G$
- $[[(P_1 \text{ OPT } P_2)]]_G = [[P_1]]_G ⟕ [[P_2]]_G$

the base case is the evaluation of a BGP.

Example (AND)

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

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

Example (AND)

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

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

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \bowtie [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

Example (AND)

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

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

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \bowtie [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

Example (AND)

G :

$(R_1$, name, john)	$(R_2$, name, paul)	$(R_3$, name, ringo)
$(R_1$, email, J@ed.ex)		$(R_3$, email, R@ed.ex)
		$(R_3$, webPage, www.ringo.com)

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

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \bowtie [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

	$?X$	$?E$
μ_4	R_1	J@ed.ex
μ_5	R_3	R@ed.ex

Example (AND)

G :

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

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

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \bowtie [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

$\bowtie$

	$?X$	$?E$
μ_4	R_1	J@ed.ex
μ_5	R_3	R@ed.ex

Example (AND)

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

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

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \bowtie [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

$\bowtie$

	$?X$	$?E$
μ_4	R_1	J@ed.ex
μ_5	R_3	R@ed.ex

	$?X$	$?N$	$?E$
$\mu_1 \cup \mu_4$	R_1	john	J@ed.ex
$\mu_3 \cup \mu_5$	R_3	ringo	R@ed.ex

Example (OPT)

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

$$[\![\{(?X, \text{name}, ?N)\} \text{ OPT } \{(?X, \text{email}, ?E)\}]\!]_G$$

Example (OPT)

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

$$[\![\{(?X,\ \text{name},\ ?N)\}\ \text{OPT}\ \{(?X,\ \text{email},\ ?E)\}]\!]_G$$

$$[\![\{(?X,\ \text{name},\ ?N)\}]\!]_G \;⟕\; [\![\{(?X,\ \text{email},\ ?E)\}]\!]_G$$

Example (OPT)

G :

$(R_1$, name, john$)$	$(R_2$, name, paul$)$	$(R_3$, name, ringo$)$
$(R_1$, email, J@ed.ex$)$		$(R_3$, email, R@ed.ex$)$
		$(R_3$, webPage, www.ringo.com$)$

$$[\![\{(?X, \text{name}, ?N)\}\ \text{OPT}\ \{(?X, \text{email}, ?E)\}]\!]_G$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \ltimes\!\!\!\!\bowtie [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

Example (OPT)

G :

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

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

$(R_3, \text{name}, \text{ringo})$ $(R_3, \text{email}, \text{R@ed.ex})$ $(R_3, \text{webPage}, \text{www.ringo.com})$

$$[\![\{(?X, \text{name}, ?N)\} \text{ OPT } \{(?X, \text{email}, ?E)\}]\!]_G$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \mathbin{⟕} [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

	$?X$	$?E$
μ_4	R_1	J@ed.ex
μ_5	R_3	R@ed.ex

Example (OPT)

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

$$[\![\{(?X, \text{name}, ?N)\} \text{ OPT } \{(?X, \text{email}, ?E)\}]\!]_G$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G ⟕ [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	?X	?N
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

⟕

	?X	?E
μ_4	R_1	J@ed.ex
μ_5	R_3	R@ed.ex

Example (OPT)

G :

$(R_1$, name, john)	$(R_2$, name, paul)	$(R_3$, name, ringo)
$(R_1$, email, J@ed.ex)		$(R_3$, email, R@ed.ex)
		$(R_3$, webPage, www.ringo.com)

$$[\![\{(?X, \text{name}, ?N)\} \text{ OPT } \{(?X, \text{email}, ?E)\}]\!]_G$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \;⟕\; [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

⟕

	$?X$	$?E$
μ_4	R_1	J@ed.ex
μ_5	R_3	R@ed.ex

	$?X$	$?N$	$?E$
$\mu_1 \cup \mu_4$	R_1	john	J@ed.ex
$\mu_3 \cup \mu_5$	R_3	ringo	R@ed.ex
μ_2	R_2	paul	

Example (OPT)

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

$$[\![\{(?X, \text{name}, ?N)\} \text{ OPT } \{(?X, \text{email}, ?E)\}]\!]_G$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_G \;⟕\; [\![\{(?X, \text{email}, ?E)\}]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

⟕

	$?X$	$?E$
μ_4	R_1	J@ed.ex
μ_5	R_3	R@ed.ex

	$?X$	$?N$	$?E$
$\mu_1 \cup \mu_4$	R_1	john	J@ed.ex
$\mu_3 \cup \mu_5$	R_3	ringo	R@ed.ex
μ_2	R_2	paul	

Example (UNION)

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

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

Example (UNION)

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

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

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

Example (UNION)

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

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

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

	$?X$	$?Info$
μ_1	R_1	J@ed.ex
μ_2	R_3	R@ed.ex

Example (UNION)

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

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

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

	$?X$	$?Info$
μ_1	R_1	J@ed.ex
μ_2	R_3	R@ed.ex

	$?X$	$?Info$
μ_3	R_3	www.ringo.com

Example (UNION)

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

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

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

	$?X$	$?\mathit{Info}$
μ_1	R_1	J@ed.ex
μ_2	R_3	R@ed.ex

$\cup$

	$?X$	$?\mathit{Info}$
μ_3	R_3	www.ringo.com

Example (UNION)

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

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

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

	?X	?Info
μ_1	R_1	J@ed.ex
μ_2	R_3	R@ed.ex

$\cup$

	?X	?Info
μ_3	R_3	www.ringo.com

	?X	?Info
μ_1	R_1	J@ed.ex
μ_2	R_3	R@ed.ex
μ_3	R_3	www.ringo.com

Boolean filter expressions (value constraints)

In filter expressions we consider

- the equality $=$ among variables and RDF terms
- a unary predicate bound
- boolean combinations ($\wedge$, $\vee$, $\neg$)

A mapping μ satisfies

- $?X = c$ if $\mu(?X) = c$
- $?X = ?Y$ if $\mu(?X) = \mu(?Y)$
- bound($?X$) if μ is defined in $?X$, i.e. $?X \in \mathrm{dom}(\mu)$

Satisfaction of value constraints

- If P is a graph pattern and R is a value constraint then (P FILTER R) is also a graph pattern.

Definition

Given a graph G

- $[[(P \text{ FILTER } R)]]_G = \{\mu \in [[P]]_G \mid \mu \text{ satisfies } R\}$
  i.e. mappings in the evaluation of P that satisfy R.

Satisfaction of value constraints

- If P is a graph pattern and R is a value constraint then (P FILTER R) is also a graph pattern.

Definition

Given a graph G

- $[\![(P \text{ FILTER } R)]\!]_G = \{\mu \in [\![P]\!]_G \mid \mu \text{ satisfies } R\}$
  i.e. mappings in the evaluation of P that satisfy R.

Example (FILTER)

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

$$[\![(\{(?X, \text{name}, ?N)\} \text{ FILTER } (?N = \text{ringo} \vee ?N = \text{paul}))]\!]_G$$

Example (FILTER)

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

$$[\![(\{(?X, \text{name}, ?N)\} \text{ FILTER } (?N = \text{ringo} \vee ?N = \text{paul}))]\!]_G$$

	$?X$	$?N$
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

Example (FILTER)

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

$$[\![(\{(?X, \text{name}, ?N)\} \ \text{FILTER} \ (?N = \text{ringo} \vee ?N = \text{paul}))]\!]_G$$

	?X	?N
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

$$?N = \text{ringo} \vee ?N = \text{paul}$$

Example (FILTER)

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

$$[\![(\{(?X, \text{name}, ?N)\} \text{ FILTER } (?N = \text{ringo} \vee ?N = \text{paul}))]\!]_G$$

	?X	?N
μ_1	R_1	john
μ_2	R_2	paul
μ_3	R_3	ringo

$?N = \text{ringo} \vee ?N = \text{paul}$

	?X	?N
μ_2	R_2	paul
μ_3	R_3	ringo

Example (FILTER)

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

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

Example (FILTER)

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

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

	$?X$	$?N$	$?E$
$\mu_1 \cup \mu_4$	R_1	john	J@ed.ex
$\mu_3 \cup \mu_5$	R_3	ringo	R@ed.ex
μ_2	R_2	paul	

Example (FILTER)

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

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

	$?X$	$?N$	$?E$
$\mu_1 \cup \mu_4$	R_1	john	J@ed.ex
$\mu_3 \cup \mu_5$	R_3	ringo	R@ed.ex
μ_2	R_2	paul	

$\neg\,\text{bound}(?E)$

Example (FILTER)

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

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

	$?X$	$?N$	$?E$
$\mu_1 \cup \mu_4$	R_1	john	J@ed.ex
$\mu_3 \cup \mu_5$	R_3	ringo	R@ed.ex
μ_2	R_2	paul	

$\neg\,\text{bound}(?E)$

	$?X$	$?N$
μ_2	R_2	paul

FILTER: differences with the official specification

- We restrict to the case in which all variables in R are mentioned in P.
- This restriction is not imposed in the official specification by W3C.

FILTER: differences with the official specification

- We restrict to the case in which all variables in R are mentioned in P.
- This restriction is not imposed in the official specification by W3C.
- The semantics without the restriction does not modify the expressive power of the language.

SPARQL Datasets

- One of the interesting features of SPARQL is that a query may retrieve data from different sources.

Definition

A SPARQL dataset is a set

$$\mathcal{D} = \{G_0, \langle u_1, G_1\rangle, \langle u_2, G_2\rangle, \ldots, \langle u_n, G_n\rangle\}$$

- G_0 is the default graph, $\langle u_i, G_i\rangle$ are named graphs
- $\text{name}(\mathcal{D}) = \{u_1, u_2, \ldots, u_n\}$
- $d_{\mathcal{D}}$ is a function such $d_{\mathcal{D}}(u_i) = G_i$.

SPARQL Datasets

- One of the interesting features of SPARQL is that a query may retrieve data from different sources.

Definition

A SPARQL dataset is a set

$$\mathcal{D} = \{G_0, \langle u_1, G_1\rangle, \langle u_2, G_2\rangle, \ldots, \langle u_n, G_n\rangle\}$$

- G_0 is the default graph, $\langle u_i, G_i\rangle$ are named graphs
- $\text{name}(\mathcal{D}) = \{u_1, u_2, \ldots, u_n\}$
- $d_{\mathcal{D}}$ is a function such $d_{\mathcal{D}}(u_i) = G_i$.

SPARQL Datasets

- One of the interesting features of SPARQL is that a query may retrieve data from different sources.

Definition

A SPARQL dataset is a set

$$\mathcal{D} = \{G_0, \langle u_1, G_1\rangle, \langle u_2, G_2\rangle, \ldots, \langle u_n, G_n\rangle\}$$

- G_0 is the default graph, $\langle u_i, G_i\rangle$ are named graphs
- $\text{name}(\mathcal{D}) = \{u_1, u_2, \ldots, u_n\}$
- $d_{\mathcal{D}}$ is a function such $d_{\mathcal{D}}(u_i) = G_i$.

SPARQL Datasets

- One of the interesting features of SPARQL is that a query may retrieve data from different sources.

Definition

A SPARQL dataset is a set

$$\mathcal{D} = \{G_0, \langle u_1, G_1\rangle, \langle u_2, G_2\rangle, \ldots, \langle u_n, G_n\rangle\}$$

- G_0 is the default graph, $\langle u_i, G_i\rangle$ are named graphs
- $\text{name}(\mathcal{D}) = \{u_1, u_2, \ldots, u_n\}$
- $d_{\mathcal{D}}$ is a function such $d_{\mathcal{D}}(u_i) = G_i$.

The GRAPH operator

if u is an IRI, $?X$ is a variable and P is a graph pattern, then

- $(u$ GRAPH $P)$ is a graph pattern
- $(?X$ GRAPH $P)$ is a graph pattern

The GRAPH operator

if u is an IRI, $?X$ is a variable and P is a graph pattern, then

- (u GRAPH P) is a graph pattern
- ($?X$ GRAPH P) is a graph pattern

GRAPH will permit us to dynamically change the graph against which our pattern is evaluated.

Semantics of GRAPH

Definition

Given a dataset $\mathcal{D}$ and a graph pattern P

Semantics of GRAPH

Definition

Given a dataset $\mathcal{D}$ and a graph pattern P

$$[\![(u \text{ GRAPH } P)]\!]_G = [\![P]\!]_{d_{\mathcal{D}}(u)}$$

Semantics of GRAPH

Definition

Given a dataset $\mathcal{D}$ and a graph pattern P

$$[\![(u \text{ GRAPH } P)]\!]_G = [\![P]\!]_{d_{\mathcal{D}}(u)}$$

$$[\![(?X \text{ GRAPH } P)]\!]_G =$$

Semantics of GRAPH

Definition

Given a dataset $\mathcal{D}$ and a graph pattern P

$$[\![(u \text{ GRAPH } P)]\!]_G = [\![P]\!]_{d_{\mathcal{D}}(u)}$$

$$[\![(?X \text{ GRAPH } P)]\!]_G = \bigcup_{u \in \text{name}(\mathcal{D})}$$

Semantics of GRAPH

Definition

Given a dataset $\mathcal{D}$ and a graph pattern P

$$[\![(u \text{ GRAPH } P)]\!]_G = [\![P]\!]_{d_{\mathcal{D}}(u)}$$

$$[\![(?X \text{ GRAPH } P)]\!]_G = \bigcup_{u \in \text{name}(\mathcal{D})} [\![P]\!]_{d_{\mathcal{D}}(u)}$$

Semantics of GRAPH

Definition

Given a dataset $\mathcal{D}$ and a graph pattern P

$$[\![(u \text{ GRAPH } P)]\!]_G = [\![P]\!]_{d_{\mathcal{D}}(u)}$$

$$[\![(?X \text{ GRAPH } P)]\!]_G = \bigcup_{u \in \text{name}(\mathcal{D})} \left([\![P]\!]_{d_{\mathcal{D}}(u)} \bowtie \{\{?X \to u\}\} \right)$$

Semantics of GRAPH

Definition

Given a dataset $\mathcal{D}$ and a graph pattern P

$$[[(u \text{ GRAPH } P)]]_G = [[P]]_{d_{\mathcal{D}}(u)}$$

$$[[(?X \text{ GRAPH } P)]]_G = \bigcup_{u \in \text{name}(\mathcal{D})} \left([[P]]_{d_{\mathcal{D}}(u)} \bowtie \{\{?X \to u\}\} \right)$$

Definition

The evaluation of a general pattern P against a dataset $\mathcal{D}$, denoted by $[[P]]_{\mathcal{D}}$, is the set $[[P]]_{G_0}$ where G_0 is the default graph in $\mathcal{D}$.

Example (GRAPH)

$\mathcal{D}$

G_0:

Example (GRAPH)

$$\mathcal{D}$$

$$G_0:$$

$$\langle\ \text{tb},\ G_1: \quad \begin{array}{l}(R_1,\ \text{name},\ \text{john})\\ (R_1,\ \text{email},\ \text{J@ed.ex})\end{array} \qquad (R_2,\ \text{name},\ \text{paul}) \quad \rangle$$

Example (GRAPH)

$$\mathcal{D}$$

$$
\begin{array}{rll}
G_0: & & \\
\langle\ \text{tb},\ G_1: & (R_1,\ \text{name},\ \text{john}) \quad (R_2,\ \text{name},\ \text{paul}) & \rangle \\
 & (R_1,\ \text{email},\ \text{J@ed.ex}) & \\
\langle\ \text{trs},\ G_2: & (R_4,\ \text{name},\ \text{mick}) \quad (R_5,\ \text{name},\ \text{keith}) & \rangle \\
 & (R_4,\ \text{email},\ \text{M@ed.ex}) \quad (R_5,\ \text{email},\ \text{K@ed.ex}) &
\end{array}
$$

Example (GRAPH)

$$\mathcal{D}$$

G_0:		
$\langle$ tb, G_1:	(R_1, name, john) (R_1, email, J@ed.ex)	(R_2, name, paul) $\rangle$
$\langle$ trs, G_2:	(R_4, name, mick) (R_4, email, M@ed.ex)	(R_5, name, keith) (R_5, email, K@ed.ex) $\rangle$

$$[\![(\text{ trs } \text{ GRAPH } \{(?X, \text{name}, ?N)\})]\!]_{\mathcal{D}}$$

Example (GRAPH)

$$\mathcal{D}$$

G_0:

$\langle$ tb, G_1: $(R_1, \text{name}, \text{john})$ $(R_2, \text{name}, \text{paul})$ $(R_1, \text{email}, \text{J@ed.ex})$ $\rangle$

$\langle$ trs, G_2: $(R_4, \text{name}, \text{mick})$ $(R_5, \text{name}, \text{keith})$ $(R_4, \text{email}, \text{M@ed.ex})$ $(R_5, \text{email}, \text{K@ed.ex})$ $\rangle$

$$[\![(\ \text{trs}\ \ \text{GRAPH}\ \{(?X, \text{name}, ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_2}$$

Example (GRAPH)

$$\mathcal{D}$$

$$\begin{array}{llll} & G_0: & & \\ \langle\ \text{tb},\ G_1: & (R_1,\ \text{name},\ \text{john}) & (R_2,\ \text{name},\ \text{paul}) & \rangle \\ & (R_1,\ \text{email},\ \text{J@ed.ex}) & & \\ \langle\ \text{trs},\ G_2: & (R_4,\ \text{name},\ \text{mick}) & (R_5,\ \text{name},\ \text{keith}) & \rangle \\ & (R_4,\ \text{email},\ \text{M@ed.ex}) & (R_5,\ \text{email},\ \text{K@ed.ex}) & \end{array}$$

$$[\![(\ \text{trs}\ \ \text{GRAPH}\ \{(?X,\ \text{name},\ ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X,\ \text{name},\ ?N)\}]\!]_{G_2}$$

	$?X$	$?N$
μ_1	R_4	mick
μ_2	R_5	keith

Example (GRAPH)

$$\mathcal{D}$$

$$\begin{array}{llll}
 & G_0: & & \\
\langle\ \text{tb},\ G_1: & (R_1,\ \text{name},\ \text{john}) & (R_2,\ \text{name},\ \text{paul}) & \rangle \\
 & (R_1,\ \text{email},\ \text{J@ed.ex}) & & \\
\langle\ \text{trs},\ G_2: & (R_4,\ \text{name},\ \text{mick}) & (R_5,\ \text{name},\ \text{keith}) & \rangle \\
 & (R_4,\ \text{email},\ \text{M@ed.ex}) & (R_5,\ \text{email},\ \text{K@ed.ex}) &
\end{array}$$

Example (GRAPH)

$$\mathcal{D}$$

$$G_0:$$

$$\langle\ \mathrm{tb},\ G_1: \quad \begin{array}{ll} (R_1, \text{name}, \text{john}) & (R_2, \text{name}, \text{paul}) \\ (R_1, \text{email}, \text{J@ed.ex}) & \end{array} \ \rangle$$

$$\langle\ \mathrm{trs},\ G_2: \quad \begin{array}{ll} (R_4, \text{name}, \text{mick}) & (R_5, \text{name}, \text{keith}) \\ (R_4, \text{email}, \text{M@ed.ex}) & (R_5, \text{email}, \text{K@ed.ex}) \end{array} \ \rangle$$

$$[\![(?G \ \text{GRAPH} \ \{(?X, \text{name}, ?N)\})]\!]_{\mathcal{D}}$$

Example (GRAPH)

$$\mathcal{D}$$

$$\begin{array}{llll} & G_0: & & \\ \langle\ \text{tb},\ G_1: & (R_1,\ \text{name},\ \text{john}) & (R_2,\ \text{name},\ \text{paul}) & \rangle \\ & (R_1,\ \text{email},\ \text{J@ed.ex}) & & \\ \langle\ \text{trs},\ G_2: & (R_4,\ \text{name},\ \text{mick}) & (R_5,\ \text{name},\ \text{keith}) & \rangle \\ & (R_4,\ \text{email},\ \text{M@ed.ex}) & (R_5,\ \text{email},\ \text{K@ed.ex}) & \end{array}$$

$$[\![(?G\ \text{GRAPH}\ \{(?X,\ \text{name},\ ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X,\ \text{name},\ ?N)\}]\!]_{G_1} \bowtie \{\{?G \rightarrow \text{tb}\}\}$$

Example (GRAPH)

$$\mathcal{D}$$

$$\begin{array}{llll}
 & G_0: & & \\
\langle\ \text{tb},\ & G_1: & (R_1, \text{name}, \text{john}) & (R_2, \text{name}, \text{paul}) \\
 & & (R_1, \text{email}, \text{J@ed.ex}) & \quad\rangle \\
\langle\ \text{trs},\ & G_2: & (R_4, \text{name}, \text{mick}) & (R_5, \text{name}, \text{keith}) \\
 & & (R_4, \text{email}, \text{M@ed.ex}) & (R_5, \text{email}, \text{K@ed.ex}) \quad\rangle
\end{array}$$

$$[\![(?G \text{ GRAPH } \{(?X, \text{name}, ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_1} \bowtie \{\{?G \to \text{tb}\}\} \ \cup$$

Example (GRAPH)

$$\mathcal{D}$$

$$\begin{array}{lll} & G_0: & & \\ \langle\ \text{tb},\ G_1: & (R_1, \text{name}, \text{john}) & (R_2, \text{name}, \text{paul}) & \rangle \\ & (R_1, \text{email}, \text{J@ed.ex}) & & \\ \langle\ \text{trs},\ G_2: & (R_4, \text{name}, \text{mick}) & (R_5, \text{name}, \text{keith}) & \rangle \\ & (R_4, \text{email}, \text{M@ed.ex}) & (R_5, \text{email}, \text{K@ed.ex}) & \end{array}$$

$$[\![(?G \ \text{GRAPH}\ \{(?X, \text{name}, ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_1} \bowtie \{\{?G \to \text{tb}\}\} \ \cup$$
$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_2} \bowtie \{\{?G \to \text{trs}\}\}$$

Example (GRAPH)

$$\mathcal{D}$$

$$\begin{array}{lll} G_0: & & \\ \langle\ \text{tb},\ G_1: & \begin{array}{ll} (R_1,\ \text{name},\ \text{john}) & (R_2,\ \text{name},\ \text{paul}) \\ (R_1,\ \text{email},\ \text{J@ed.ex}) & \end{array} & \rangle \\ \langle\ \text{trs},\ G_2: & \begin{array}{ll} (R_4,\ \text{name},\ \text{mick}) & (R_5,\ \text{name},\ \text{keith}) \\ (R_4,\ \text{email},\ \text{M@ed.ex}) & (R_5,\ \text{email},\ \text{K@ed.ex}) \end{array} & \rangle \end{array}$$

$$[\![(?G\ \text{GRAPH}\ \{(?X,\ \text{name},\ ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X,\ \text{name},\ ?N)\}]\!]_{G_1} \bowtie \{\{?G \to \text{tb}\}\} \ \cup$$
$$[\![\{(?X,\ \text{name},\ ?N)\}]\!]_{G_2} \bowtie \{\{?G \to \text{trs}\}\}$$

$$\begin{array}{c|c|c|} \cline{2-3} & ?X & ?N \\ \cline{2-3} \mu_1 & R_1 & \text{john} \\ \cline{2-3} \mu_2 & R_2 & \text{paul} \\ \cline{2-3} \end{array} \bowtie \{\{?G \to \text{tb}\}\}$$

Example (GRAPH)

$\mathcal{D}$

G_0:

$\langle$ tb, G_1: $(R_1, \text{name}, \text{john})$ $(R_2, \text{name}, \text{paul})$ $(R_1, \text{email}, \text{J@ed.ex})$ $\rangle$

$\langle$ trs, G_2: $(R_4, \text{name}, \text{mick})$ $(R_5, \text{name}, \text{keith})$ $(R_4, \text{email}, \text{M@ed.ex})$ $(R_5, \text{email}, \text{K@ed.ex})$ $\rangle$

$$[\![(?G \text{ GRAPH } \{(?X, \text{name}, ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_1} \bowtie \{\{?G \to \text{tb}\}\} \;\cup$$
$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_2} \bowtie \{\{?G \to \text{trs}\}\}$$

	?X	?N
μ_1	R_1	john
μ_2	R_2	paul

$\bowtie \{\{?G \to \text{tb}\}\} \cup$

	?X	?N
μ_3	R_4	mick
μ_4	R_5	keith

$\bowtie \{\{?G \to \text{trs}\}\}$

Example (GRAPH)

$$\mathcal{D}$$

G_0:

$\langle$ tb, G_1: $(R_1, \text{name}, \text{john})$ $(R_2, \text{name}, \text{paul})$ $(R_1, \text{email}, \text{J@ed.ex})$ $\rangle$

$\langle$ trs, G_2: $(R_4, \text{name}, \text{mick})$ $(R_5, \text{name}, \text{keith})$ $(R_4, \text{email}, \text{M@ed.ex})$ $(R_5, \text{email}, \text{K@ed.ex})$ $\rangle$

$$[\![(?G \text{ GRAPH } \{(?X, \text{name}, ?N)\})]\!]_{\mathcal{D}}$$

$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_1} \bowtie \{\{?G \rightarrow \text{tb}\}\} \cup$$
$$[\![\{(?X, \text{name}, ?N)\}]\!]_{G_2} \bowtie \{\{?G \rightarrow \text{trs}\}\}$$

	?X	?N
μ_1	R_1	john
μ_2	R_2	paul

$\bowtie \{\{?G \rightarrow \text{tb}\}\} \cup$

	?X	?N
μ_3	R_4	mick
μ_4	R_5	keith

$\bowtie \{\{?G \rightarrow \text{trs}\}\}$

?G	?X	?N
tb	R_1	john
tb	R_2	paul
trs	R_4	mick
trs	R_5	keith

SELECT

- Up to this point we have concentrated in the body of a SPARQL query, i.e. in the graph pattern matching expression.
- A query can also process the values of the variables. The most simple processing operation is the selection of some variables appearing in the query.

SELECT

- Up to this point we have concentrated in the body of a SPARQL query, i.e. in the graph pattern matching expression.
- A query can also process the values of the variables. The most simple processing operation is the selection of some variables appearing in the query.

Definition

- A SELECT query is a tuple (W, P) where P is a graph pattern and W is a set of variable.
- The answer of a SELECT query against a dataset $\mathcal{D}$ is

$$\{\mu_{|W} \mid \mu \in [[P]]_{\mathcal{D}}\}$$

where $\mu_{|W}$ is the restriction of μ to domain W.

SELECT

- Up to this point we have concentrated in the body of a SPARQL query, i.e. in the graph pattern matching expression.
- A query can also process the values of the variables. The most simple processing operation is the selection of some variables appearing in the query.

Definition

- A SELECT query is a tuple (W, P) where P is a graph pattern and W is a set of variable.
- The answer of a SELECT query against a dataset $\mathcal{D}$ is

$$\{\mu_{|W} \mid \mu \in [\![P]\!]_{\mathcal{D}}\}$$

where $\mu_{|W}$ is the restriction of μ to domain W.

SELECT

- Up to this point we have concentrated in the body of a SPARQL query, i.e. in the graph pattern matching expression.
- A query can also process the values of the variables. The most simple processing operation is the selection of some variables appearing in the query.

Definition

- A SELECT query is a tuple (W, P) where P is a graph pattern and W is a set of variable.
- The answer of a SELECT query against a dataset $\mathcal{D}$ is

$$\{\mu_{|W} \mid \mu \in [\![P]\!]_{\mathcal{D}}\}$$

where $\mu_{|W}$ is the restriction of μ to domain W.

CONSTRUCT

- A query can also output an RDF graph.
- The construction of the output graph is based on a template.
- A template is a set of triple patterns possibly with bnodes.

Example

$$T_1 = \{(?X, \text{name}, ?Y), (?X, \text{info}, ?I), (?X, \text{addr}, B)\}$$

with B a bnode

Definition

- A CONSTRUCT query is a tuple (T, P) where P is a graph pattern and T is a template.

CONSTRUCT

- A query can also output an RDF graph.
- The construction of the output graph is based on a template.
- A template is a set of triple patterns possibly with bnodes.

Example

$$T_1 = \{(?X, \text{name}, ?Y), (?X, \text{info}, ?I), (?X, \text{addr}, B)\}$$

with B a bnode

Definition

- A CONSTRUCT query is a tuple (T, P) where P is a graph pattern and T is a template.

CONSTRUCT

- A query can also output an RDF graph.
- The construction of the output graph is based on a template.
- A template is a set of triple patterns possibly with bnodes.

Example

$$T_1 = \{(?X, \text{name}, ?Y), (?X, \text{info}, ?I), (?X, \text{addr}, B)\}$$

with B a bnode

Definition

- A CONSTRUCT query is a tuple (T, P) where P is a graph pattern and T is a template.

CONSTRUCT: Semantics

Definition

The answer of a CONSTRUCT query (T, P) against a dataset $\mathcal{D}$ is obtained by

- for every $\mu \in [[P]]_{\mathcal{D}}$ create a template T_μ with fresh bnodes
- take the union of $\mu(T_\mu)$ for every $\mu \in [[P]]_{\mathcal{D}}$
- discard the not valid RDF triples
  - some variables have not been instantiated.
  - bnodes in predicate positions

CONSTRUCT: Semantics

Definition

The answer of a CONSTRUCT query (T, P) against a dataset $\mathcal{D}$ is obtained by

- for every $\mu \in [\![P]\!]_{\mathcal{D}}$ create a template T_μ with fresh bnodes
- take the union of $\mu(T_\mu)$ for every $\mu \in [\![P]\!]_{\mathcal{D}}$
- discard the not valid RDF triples
  - some variables have not been instantiated.
  - bnodes in predicate positions

CONSTRUCT: Semantics

Definition

The answer of a CONSTRUCT query (T, P) against a dataset $\mathcal{D}$ is obtained by

- for every $\mu \in [\![P]\!]_{\mathcal{D}}$ create a template T_μ with fresh bnodes
- take the union of $\mu(T_\mu)$ for every $\mu \in [\![P]\!]_{\mathcal{D}}$
- discard the not valid RDF triples
  - some variables have not been instantiated.
  - bnodes in predicate positions

CONSTRUCT: Semantics

Definition

The answer of a CONSTRUCT query (T, P) against a dataset $\mathcal{D}$ is obtained by

- for every $\mu \in [\![P]\!]_{\mathcal{D}}$ create a template T_μ with fresh bnodes
- take the union of $\mu(T_\mu)$ for every $\mu \in [\![P]\!]_{\mathcal{D}}$
- discard the not valid RDF triples
  - some variables have not been instantiated.
  - bnodes in predicate positions

CONSTRUCT: Semantics

Definition

The answer of a CONSTRUCT query (T, P) against a dataset $\mathcal{D}$ is obtained by

- for every $\mu \in [[P]]_{\mathcal{D}}$ create a template T_μ with fresh bnodes
- take the union of $\mu(T_\mu)$ for every $\mu \in [[P]]_{\mathcal{D}}$
- discard the not valid RDF triples
  - some variables have not been instantiated.
  - bnodes in predicate positions

CONSTRUCT: Semantics

Definition

The answer of a CONSTRUCT query (T, P) against a dataset $\mathcal{D}$ is obtained by

- for every $\mu \in [\![P]\!]_{\mathcal{D}}$ create a template T_μ with fresh bnodes
- take the union of $\mu(T_\mu)$ for every $\mu \in [\![P]\!]_{\mathcal{D}}$
- discard the not valid RDF triples
  - some variables have not been instantiated.
  - bnodes in predicate positions

Blank nodes in graph patterns

- We allow now bnodes in triple patterns.
- Bnodes act as existentials scoped to the basic graph pattern.

Blank nodes in graph patterns

- We allow now bnodes in triple patterns.
- Bnodes act as existentials scoped to the basic graph pattern.

Definition

The evaluation of the BGP P with bnodes over the graph G denoted $[\![P]\!]_G$, is the set of all mappings μ such that:

- dom(μ) is exactly the set of variables occurring in P,
- there exists a function θ from bnodes of P to G such that

$$\mu(\theta(P)) \subseteq G.$$

Blank nodes in graph patterns

- We allow now bnodes in triple patterns.
- Bnodes act as existentials scoped to the basic graph pattern.

Definition

The evaluation of the BGP P with bnodes over the graph G denoted $[[P]]_G$, is the set of all mappings μ such that:

- $\text{dom}(\mu)$ is exactly the set of variables occurring in P,
- there exists a function θ from bnodes of P to G such that

$$\mu(\theta(P)) \subseteq G.$$

Blank nodes in graph patterns

- We allow now bnodes in triple patterns.
- Bnodes act as existentials scoped to the basic graph pattern.

Definition

The evaluation of the BGP P with bnodes over the graph G denoted $[\![P]\!]_G$, is the set of all mappings μ such that:

- $\mathrm{dom}(\mu)$ is exactly the set of variables occurring in P,
- there exists a function θ from bnodes of P to G such that

$$\mu(\theta(P)) \subseteq G.$$

Blank nodes in graph patterns

- We allow now bnodes in triple patterns.
- Bnodes act as existentials scoped to the basic graph pattern.

Definition

The evaluation of the BGP P with bnodes over the graph G denoted $[\![P]\!]_G$, is the set of all mappings μ such that:

- $\text{dom}(\mu)$ is exactly the set of variables occurring in P,
- there exists a function θ from bnodes of P to G such that

$$\mu(\theta(P)) \subseteq G.$$

- A natural extension of BGPs without bnodes.
- The algebra remains the same.

Bag/Multiset semantics

- In a bag, a mapping can have cardinality greater than one.
- Every mapping μ in a bag M is annotated with an integer $\mathrm{c}_M(\mu)$ that represents its cardinality ($\mathrm{c}_M(\mu) = 0$ if $\mu \notin M$).
- Operations between sets of mappings can be extended to bags maintaining duplicates:

Bag/Multiset semantics

- In a bag, a mapping can have cardinality greater than one.
- Every mapping μ in a bag M is annotated with an integer $\mathsf{c}_M(\mu)$ that represents its cardinality ($\mathsf{c}_M(\mu) = 0$ if $\mu \notin M$).
- Operations between sets of mappings can be extended to bags maintaining duplicates:

Definition

$$
\begin{aligned}
\mu \in M = M_1 \bowtie M_2, \quad \mathsf{c}_M(\mu) &= \sum_{\mu = \mu_1 \cup \mu_2} \mathsf{c}_{M_1}(\mu_1) \cdot \mathsf{c}_{M_2}(\mu_2), \\
\mu \in M = M_1 \cup M_2, \quad \mathsf{c}_M(\mu) &= \mathsf{c}_{M_1}(\mu) + \mathsf{c}_{M_2}(\mu), \\
\mu \in M = M_1 \smallsetminus M_2, \quad \mathsf{c}_M(\mu) &= \mathsf{c}_{M_1}(\mu).
\end{aligned}
$$

Bag/Multiset semantics

- In a bag, a mapping can have cardinality greater than one.
- Every mapping μ in a bag M is annotated with an integer $c_M(\mu)$ that represents its cardinality ($c_M(\mu) = 0$ if $\mu \notin M$).
- Operations between sets of mappings can be extended to bags maintaining duplicates:

Definition

$$
\begin{aligned}
\mu \in M = M_1 \bowtie M_2, &\quad c_M(\mu) = \sum_{\mu = \mu_1 \cup \mu_2} c_{M_1}(\mu_1) \cdot c_{M_2}(\mu_2), \\
\mu \in M = M_1 \cup M_2, &\quad c_M(\mu) = c_{M_1}(\mu) + c_{M_2}(\mu), \\
\mu \in M = M_1 \smallsetminus M_2, &\quad c_M(\mu) = c_{M_1}(\mu).
\end{aligned}
$$

- Intuition: we simply do not discard duplicates.

References

- R. Cyganiak, *A Relational Algebra for SPARQL*. Tech Report HP Laboratories, HPL-2005-170.
- E. Prud'hommeaux, A. Seaborne, *SPARQL Query Language for RDF*. W3C Working Draft, 2007.
- J. Pérez, M. Arenas, C. Gutierrez, *Semantics and Complexity of SPARQL*. In *Int. Semantic Web Conference 2006*.
- J. Pérez, M. Arenas, C. Gutierrez, *Semantics of SPARQL*. Tech Report Universidad de Chile 2006, TR/DCC-2006-17.