Incomplete Information in RDF

Charalampos Nikolaou and Manolis Koubarakis

charnik@di.uoa.gr koubarak@di.uoa.gr

Department of Informatics and Telecommunications
National and Kapodistrian University of Athens

Web Reasoning and Rule Systems (RR) 2013
July 27, 2013

Outline

Motivation

Previous work

The RDF^i framework

SPARQL query evaluation over RDF^i databases

An algorithm for certain answer computation

Preliminary complexity results

Conclusions and future work

Motivation

- Incomplete information is an important issue in many research areas: relational databases, knowledge representation and the semantic web.
- Incomplete information arises in many practical settings (e.g., sensor data). RDF is often used to represent such data.
- Even if initial information is complete, incomplete information arises later on (e.g., relational view updates, data integration, data exchange).
- Although there is much work recently on incomplete information in XML, not much has been done for incomplete information in RDF.

Previous work

Relational

- Relations extended to tables with various models of incompleteness [Imielinski/Lipski '84]
- Complexity results for the associated decision problems [Abiteboul/Kanellakis/Grahne '91]
- Dependencies and updates [Grahne '91]

Previous work

Relational

- Relations extended to tables with various models of incompleteness [Imielinski/Lipski '84]
- Complexity results for the associated decision problems [Abiteboul/Kanellakis/Grahne '91]
- Dependencies and updates [Grahne '91]

XML

- Dynamic enrichment of incomplete information [Abiteboul/Segoufin/Vianu '01,'06]
- General models of incompleteness, query answering, and computational complexity [Barceló/Libkin/Poggi/Sirangelo '09,'10]

Previous work (cont'd)

RDF

- Blank nodes as existential variables in the RDF standard
- SPARQL query evaluation under certain answer semantics (Open World Assumption) [Arenas/Pérez '11]
- Anonymous timestamps in general temporal RDF graphs [Gutierrez/Hurtado/Vaisman '05]
- General temporal RDF graphs with temporal constraints [Hurtado/Vaisman '06]

Previous work (cont'd)

RDF

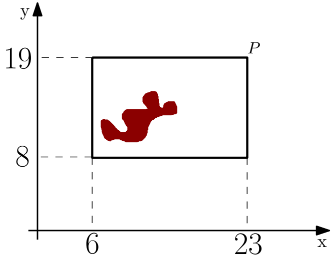
- Blank nodes as existential variables in the RDF standard
- SPARQL query evaluation under certain answer semantics (Open World Assumption) [Arenas/Pérez '11]
- Anonymous timestamps in general temporal RDF graphs [Gutierrez/Hurtado/Vaisman '05]
- General temporal RDF graphs with temporal constraints [Hurtado/Vaisman '06]

RDF$^{\mathbf{i}}$: It captures incomplete information for property values using constraints. It is for RDF what the c-tables model is for the relational model.

RDF^i by example

Example

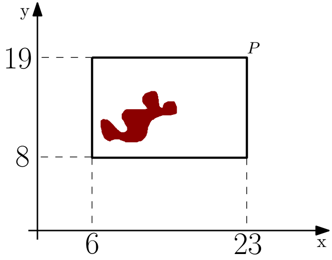
```
hotspot1     type       Hotspot .
   fire1     type       Fire    .
hotspot1 correspondsTo fire1    .
   fire1   occuredIn    _R1     .
```



RDF^i by example

Example

```
hotspot1       type       Hotspot .
   fire1       type       Fire    .
hotspot1 correspondsTo fire1   .
   fire1    occuredIn     _R1     .
```



_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

RDF$^{\mathsf{i}}$ in a nutshell

- Extension of RDF for capturing incomplete information for property values that exist but are unknown or partially known
- Partial knowledge captured by constraints using an appropriate constraint language $\mathcal{L}$ interpreted over a fixed structure $\mathbf{M}_{\mathcal{L}}$

Syntax

RDF graphs extended to RDF$^{\mathsf{i}}$ databases: pair (G, ϕ)

- G: RDF graph with a new kind of literals, called e-literals
- ϕ: quantifier-free formula of $\mathcal{L}$

Semantics

- Possible world semantics as in [Imielinski/Lipski '84] and [Grahne '91]

Constraint languages $\mathcal{L}$

Examples

ECL

- Equality constraints interpreted over an infinite domain: x EQ y, x EQ c
- Blank nodes as existential variables

Constraint languages $\mathcal{L}$

Examples

ECL

- Equality constraints interpreted over an infinite domain: x EQ y, x EQ c
- Blank nodes as existential variables

diPCL/dePCL

- Difference constraints of the form $x - y \leq c$ interpreted over the integers or rationals
- Incomplete temporal information [Koubarakis '94]

Constraint languages $\mathcal{L}$

Examples

ECL

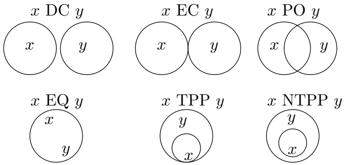
- Equality constraints interpreted over an infinite domain: x EQ y, x EQ c
- Blank nodes as existential variables

diPCL/dePCL

- Difference constraints of the form $x - y \leq c$ interpreted over the integers or rationals
- Incomplete temporal information [Koubarakis '94]

TCL

- Topological constraints of non-empty, regular closed subsets of topological space
- Six binary predicates: $\mathrm{DC}, \mathrm{EC}, \mathrm{PO}, \mathrm{EQ}, \mathrm{TPP}, \mathrm{NTPP}$



Constraint languages $\mathcal{L}$

Examples

ECL

- Equality constraints interpreted over an infinite domain: x EQ y, x EQ c
- Blank nodes as existential variables

diPCL/dePCL

- Difference constraints of the form $x - y \leq c$ interpreted over the integers or rationals
- Incomplete temporal information [Koubarakis '94]

TCL

- Topological constraints of non-empty, regular closed subsets of topological space
- Six binary predicates: $\mathrm{DC}, \mathrm{EC}, \mathrm{PO}, \mathrm{EQ}, \mathrm{TPP}, \mathrm{NTPP}$

PCL

- TCL plus constant symbols representing polygons in $\mathbb{Q}^2$
- e.g., r NTPP "$x - y \geq 0 \wedge x \leq 1 \wedge y \geq 0$"

RDF$^{\mathsf{i}}$: Vocabulary

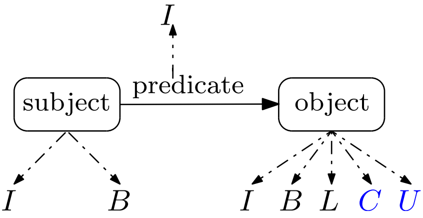
RDF	RDF$^{\mathsf{i}}$	$\mathcal{L}$
I (IRIs)	I	
B (blank nodes)	B	
L (literals)	L	
	C (literals)	constants
	U (e-literals)	variables
M (datatype map)	M	
	A (datatypes)	set of sorts

RDF$^{\mathsf{i}}$: Vocabulary

RDF	RDF$^{\mathsf{i}}$	$\mathcal{L}$
I (IRIs)	I	
B (blank nodes)	B	
L (literals)	L	
	C (literals)	constants
	U (e-literals)	variables
M (datatype map)	M	
	A (datatypes)	set of sorts

$\mathbf{M}_{\mathcal{L}}$ interprets the constants of $\mathcal{L}$ **in agreement with** function $L2V$ of M

RDF$^{\mathsf{i}}$: Syntax

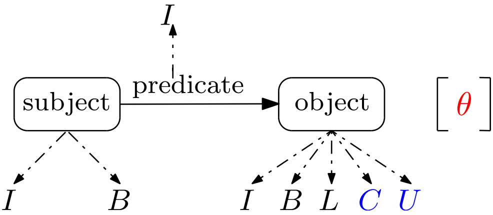


I : IRIs
B: blank nodes
L : literals
C: constants of $\mathcal{L}$
U: e-literals

Definition

- $(s, p, o) \in (I \cup B) \cup I \cup (I \cup B \cup L \cup C \cup U)$ is called an e-triple

RDF^{i}: Syntax

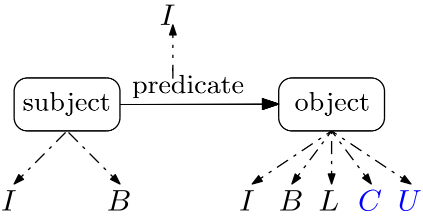


I : IRIs
B: blank nodes
L : literals
C: constants of $\mathcal{L}$
U: e-literals

Definition

- $(s, p, o) \in (I \cup B) \cup I \cup (I \cup B \cup L \cup C \cup U)$ is called an e-triple
- If t is an e-triple and θ a conjunction of $\mathcal{L}$-constraints, then the pair (t, θ) is called a conditional triple

RDF^i: Syntax



$\left[\ \theta\ \right]$

I : IRIs
B: blank nodes
L : literals
C: constants of $\mathcal{L}$
U: e-literals

Definition

- $(s, p, o) \in (I \cup B) \cup I \cup (I \cup B \cup L \cup C \cup U)$ is called an e-triple
- If t is an e-triple and θ a conjunction of $\mathcal{L}$-constraints, then the pair (t, θ) is called a conditional triple
- A set of conditional triples is called a conditional graph

RDF^{i}: Syntax (cont'd)

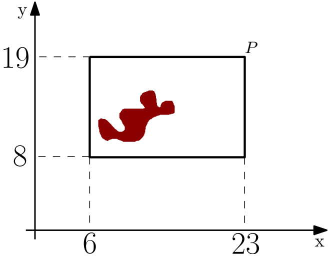
Definition

An RDF^{i} database D is a pair $D = (G, \phi)$ where G is a conditional graph and ϕ a Boolean combination of $\mathcal{L}$-constraints (global constraint)

Example

```
hotspot1      type       Hotspot .
   fire1      type       Fire    .
hotspot1 correspondsTo fire1     .
   fire1   occuredIn   _R1       .
```

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"



RDF$^{\mathrm{i}}$: Semantics

RDF$^{\mathrm{i}}$ database

set of RDF graphs

$$\mathbf{D} \xrightarrow{\quad Rep \quad} \left\{ \begin{array}{c} G_1, \\ G_2, \\ \vdots \end{array} \right\}$$

RDF$^{\mathrm{i}}$: Semantics

RDF$^{\mathrm{i}}$ database — set of RDF graphs

$$\mathbf{D} \xrightarrow{\quad Rep \quad} \left\{ \begin{array}{c} G_1, \\ G_2, \\ \vdots \end{array} \right\}$$

Definition

A valuation v is a function from U to C assigning to each e-literal from U a constant from C

Definition

Let G be a conditional graph and v a valuation. Then $v(G)$ denotes the RDF graph

$$\{v(t) \mid (t, \theta) \in G \text{ and } \mathbf{M}_{\mathcal{L}} \models v(\theta)\}$$

RDF^{i}: Semantics (cont'd)

From RDF^{i} databases to sets of RDF graphs

An RDF^{i} database $D = (G, \phi)$ corresponds to the following set of RDF graphs:

$$Rep(D) = \Big\{ H \mid \text{there exists valuation } v \text{ and RDF graph } H \text{ such that } \mathbf{M}_{\mathcal{L}} \models v(\phi) \text{ and } H \supseteq v(G) \Big\}$$

- Relation $\supseteq$ captures the OWA semantics
- An RDF^{i} database corresponds to an infinite number of RDF graphs

Question

How can we evaluate a query q over an RDF$^{\mathrm{i}}$ database D (compute $[\![q]\!]_D$)?

Question

How can we evaluate a query q over an RDF^i database D (compute $[\![q]\!]_D$)?

Semantic definition

$$[\![q]\!]_{Rep(D)} = \{[\![q]\!]_G \mid G \in Rep(D)\}$$

Question

How can we evaluate a query q over an RDF^i database D (compute $[\![q]\!]_D$)?

Semantic definition

$$[\![q]\!]_{Rep(D)} = \{[\![q]\!]_G \mid G \in Rep(D)\}$$

Question

How can we evaluate a query q over an RDF$^{\mathrm{i}}$ database D (compute $[\![q]\!]_D$)?

Semantic definition

$$[\![q]\!]_{Rep(D)} = \{[\![q]\!]_G \mid G \in Rep(D)\}$$

In practice?

- Start with SPARQL algebra of [Pérez/Arenas/Gutierrez '06] with set semantics
- Define SPARQL query evaluation for RDF$^{\mathrm{i}}$ databases

From mappings to e-mappings...

$$\{?\mathrm{F} \rightarrow \mathrm{fire1}, ?\mathrm{S} \rightarrow \text{"}\mathrm{x} \geq 1 \wedge \mathrm{x} \leq 2 \wedge \mathrm{y} \geq 1 \wedge \mathrm{y} \leq 2\text{"}\}$$

From mappings to e-mappings...

$$\{?\mathrm{F} \rightarrow \mathrm{fire1}, ?\mathrm{S} \rightarrow \mathrm{"x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2"}\}$$

$$\{?\mathrm{F} \rightarrow \mathrm{fire1}, ?\mathrm{S} \rightarrow \mathrm{\_R1}\}$$

... to conditional mappings

$$\{?\mathrm{F} \rightarrow \mathrm{fire1}, ?\mathrm{S} \rightarrow \text{"}\mathrm{x} \geq 1 \wedge \mathrm{x} \leq 2 \wedge \mathrm{y} \geq 1 \wedge \mathrm{y} \leq 2\text{"}\}$$

... to conditional mappings

$$\left(\{?\mathrm{F} \rightarrow \mathrm{fire1}, ?\mathrm{S} \rightarrow \text{"}\mathrm{x} \geq 1 \wedge \mathrm{x} \leq 2 \wedge \mathrm{y} \geq 1 \wedge \mathrm{y} \leq 2\text{"}\},\ \textit{true}\right)$$

… to conditional mappings

$$\Big(\{?\mathrm{F} \rightarrow \mathrm{fire1}, ?\mathrm{S} \rightarrow \_\mathrm{R1}\}, \_\mathrm{R1}\ \mathrm{EQ}\ "\mathrm{x} \geq 1 \wedge \mathrm{x} \leq 2 \wedge \mathrm{y} \geq 1 \wedge \mathrm{y} \leq 2"\Big)$$

From compatible mappings to possibly compatible mappings

Join of conditional mappings

$$\Big(\{?F \rightarrow \mathit{fire}1, \quad ?S \rightarrow \_R1\},\ \_R1\ \mathit{EQ}\ "x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2"\Big)$$

$$\Big(\{ \qquad ?S \rightarrow \_R2\},\ \mathit{true}\Big)$$

From compatible mappings to possibly compatible mappings

Join of conditional mappings

$$\Big(\{?F \rightarrow \mathit{fire}1, \quad ?S \rightarrow \_R1\},\ \_R1\ EQ\ \text{"}x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2\text{"}\Big)$$

$$\Big(\{ \qquad\qquad ?S \rightarrow \_R2\},\ \mathit{true}\Big)$$

From compatible mappings to possibly compatible mappings

Join of conditional mappings

$$\Big(\{?F \rightarrow \mathit{fire}1, \quad ?S \rightarrow \_R1\}, \_R1\ EQ\ "x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2"\Big)$$

$$\bowtie$$

$$\Big(\{ \qquad ?S \rightarrow \_R2\}, \mathit{true}\Big)$$

$$=$$

From compatible mappings to possibly compatible mappings

Join of conditional mappings

$$\Big(\{?F \rightarrow fire1, \quad ?S \rightarrow \_R1\},\ \_R1\ EQ\ "x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2"\Big)$$

$$\bowtie$$

$$\Big(\{\qquad ?S \rightarrow \_R2\},\ true\Big)$$

$$=$$

$$\Big(\{?F \rightarrow fire1, \quad ?S \rightarrow \_R1\},\ true\ \wedge\ \_R1\ EQ\ \_R2\ \wedge$$
$$\_R1\ EQ\ "x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2"\Big)$$

Operations on conditional mappings

Let Ω_1 and Ω_2 be sets of conditional mappings. We can define the operation of:

- Join ($\Omega_1 \bowtie \Omega_2$)
- Union ($\Omega_1 \cup \Omega_2$)
- Difference ($\Omega_1 \setminus \Omega_2$)
- Left-outer join ($\Omega_1 ⟕ \Omega_2$)

Graph pattern evaluation

If D is an RDF^i database and P a graph pattern, the evaluation of P over D is defined recursively:

Graph pattern evaluation

If D is an RDF^i database and P a graph pattern, the evaluation of P over D is defined recursively:

base case:

P is the triple pattern t

recursion:

$$
\begin{array}{lcl}
P \text{ is } (P_1 \text{ AND } P_2) & \rightarrow & [\![P_1]\!]_D \bowtie [\![P_2]\!]_D \\
P \text{ is } (P_1 \text{ UNION } P_2) & \rightarrow & [\![P_1]\!]_D \cup [\![P_2]\!]_D \\
P \text{ is } (P_1 \text{ OPT } P_2) & \rightarrow & [\![P_1]\!]_D ⟕ [\![P_2]\!]_D
\end{array}
$$

Graph pattern evaluation

If D is an RDF$^{\text{i}}$ database and P a graph pattern, the evaluation of P over D is defined recursively:

base case:

P is the triple pattern t

recursion:

$$
\begin{array}{lcl}
P \text{ is } (P_1 \text{ AND } P_2) & \rightarrow & [\![P_1]\!]_D \bowtie [\![P_2]\!]_D \\
P \text{ is } (P_1 \text{ UNION } P_2) & \rightarrow & [\![P_1]\!]_D \cup [\![P_2]\!]_D \\
P \text{ is } (P_1 \text{ OPT } P_2) & \rightarrow & [\![P_1]\!]_D ⟕ [\![P_2]\!]_D \\
P \text{ is } (P_1 \text{ FILTER } R) & & \\
\quad \text{where } R \text{ is a conjunction of } \mathcal{L}\text{-constraints} & &
\end{array}
$$

Graph pattern evaluation

If D is an RDF^{i} database and P a graph pattern, the evaluation of P over D is defined recursively:

base case:

P is the triple pattern t

recursion:

P is (P_1 AND P_2) $\rightarrow$ $[\![P_1]\!]_D \bowtie [\![P_2]\!]_D$

P is (P_1 UNION P_2) $\rightarrow$ $[\![P_1]\!]_D \cup [\![P_2]\!]_D$

P is (P_1 OPT P_2) $\rightarrow$ $[\![P_1]\!]_D ⟕ [\![P_2]\!]_D$

P is (P_1 FILTER R)

where R is a conjunction of $\mathcal{L}$-constraints

Triple pattern evaluation (case 1)

Example

Database D

```
fire1 occuredIn _R1 .
```

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

```
?F occuredIn ?R
```

Triple pattern evaluation (case 1)

Example

Database D

```
fire1 occuredIn _R1 .
```

$\_\mathrm{R1}\ \mathrm{NTPP}$ "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

```
?F occuredIn ?R
```

Answer (set of conditional mappings)

$$[\![q]\!]_D = \Big\{ \big(\{?\mathrm{F} \rightarrow \mathrm{fire1}, ?\mathrm{R} \rightarrow \_\mathrm{R1}\}, \mathrm{true} \big) \Big\}$$

Triple pattern evaluation (case 2)

Example

Database D

`fire1 occuredIn _R1 .`

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

`?F occuredIn`
"$x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2$"

Triple pattern evaluation (case 2)

Example

Database D

`fire1 occuredIn _R1 .`

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

`?F occuredIn`
"$x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2$"

Answer (set of conditional mappings)

$$[\![q]\!]_D = \Big\{ \big(\{?F \rightarrow \text{fire1}\}, \text{\_R1 EQ "}x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2\text{"} \big) \Big\}$$

Evaluation of FILTER graph patterns

Example

Database D

```
fire1 occuredIn _R1 .
```

_R1 NTPP "$\mathrm{x} \geq 6 \wedge \mathrm{x} \leq 23 \wedge \mathrm{y} \geq 8 \wedge \mathrm{y} \leq 19$"

Query q

```
?F occuredIn ?R .
FILTER (?R NTPP
```

"$\mathrm{x} \geq 1 \wedge \mathrm{x} \leq 2 \wedge \mathrm{y} \geq 1 \wedge \mathrm{y} \leq 2$")

Evaluation of FILTER graph patterns

Example

Database D

```
fire1 occuredIn _R1 .
```

$\_R1\ \mathrm{NTPP}\ "x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19"$

Query q

```
?F occuredIn ?R .
FILTER (?R NTPP
```

$"x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2")$

Answer

$$[\![q]\!]_D = \Big\{ \big(\{?F \rightarrow \mathrm{fire1},\ ?R \rightarrow \_R1\}, \ \_R1\ \mathrm{NTPP}\ "x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2" \big) \Big\}$$

SELECT queries

Example

Database D

```
fire1 occuredIn _R1 .
```

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

```
SELECT ?F
WHERE {
  ?F occuredIn ?R .
  FILTER (?R NTPP
  "x ≥ 1 ∧ x ≤ 2 ∧ y ≥ 1 ∧ y ≤ 2")}
```

SELECT queries

Example

Database D

```
fire1 occuredIn _R1 .
```

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

```
SELECT ?F
WHERE {
  ?F occuredIn ?R .
  FILTER (?R NTPP
  "x ≥ 1 ∧ x ≤ 2 ∧ y ≥ 1 ∧ y ≤ 2")}
```

Answer (set of conditional mappings)

$$[\![q]\!]_D = \Big\{\Big(\{?F \rightarrow \text{fire1}\}, \quad \text{\_R1 NTPP "} x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2\text{"}\Big)\Big\}$$

CONSTRUCT queries

Example

Database D

```
fire1 occuredIn _R1 .
```

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

```
CONSTRUCT { ?F type Fire }
WHERE {
    ?F occuredIn ?R
}
```

CONSTRUCT queries

Example

Database D

```
fire1 occuredIn _R1 .
```

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

```
CONSTRUCT { ?F type Fire }
WHERE {
    ?F occuredIn ?R
}
```

Answer (RDF^i database)

$D' = (G', \phi)$

```
fire1 type Fire .
```

_R1 NTPP "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

CONSTRUCT queries

Example

Database D

```
fire1 occuredIn _R1 .
```

_R1 NTPP "$\mathrm{x} \geq 6 \wedge \mathrm{x} \leq 23 \wedge \mathrm{y} \geq 8 \wedge \mathrm{y} \leq 19$"

Query q

```
CONSTRUCT { ?F type Fire }
WHERE {
    ?F occuredIn ?R
}
```

Answer (RDF^i database)

$D' = (G', \phi)$

```
fire1 type Fire .
```

_R1 NTPP "$\mathrm{x} \geq 6 \wedge \mathrm{x} \leq 23 \wedge \mathrm{y} \geq 8 \wedge \mathrm{y} \leq 19$"

Closure property

Correctness of SPARQL query evaluation for RDF^i

Does query evaluation compute the correct answer
(the answer agrees with the semantic definition)?

Correctness of SPARQL query evaluation for RDF[i]

Does query evaluation compute the correct answer
(the answer agrees with the semantic definition)?

The following diagram should commute

$$\begin{tikzcd}
\mathcal{D} \arrow[r, "Rep"] & \mathcal{G} \arrow[d, "q"] \\
 & \mathcal{G}
\end{tikzcd}$$

Correctness of SPARQL query evaluation for RDF[i]

Does query evaluation compute the correct answer
(the answer agrees with the semantic definition)?

The following diagram should commute

$$\begin{tikzcd}
\mathcal{D} \arrow[r, "Rep"] \arrow[d, "q"'] & \mathcal{G} \arrow[d, "q"] \\
\mathcal{D} & \mathcal{G}
\end{tikzcd}$$

Correctness of SPARQL query evaluation for RDF[i]

Does query evaluation compute the correct answer
(the answer agrees with the semantic definition)?

The following diagram should commute

$$
\begin{tikzcd}
\mathcal{D} \arrow[r, "Rep"] \arrow[d, "q"'] & \mathcal{G} \arrow[d, "q"] \\
\mathcal{D} \arrow[r, "Rep"'] & \mathcal{G}
\end{tikzcd}
$$

Correctness of SPARQL query evaluation for RDF[i]

Does query evaluation compute the correct answer
(the answer agrees with the semantic definition)?

The following diagram should commute

$$\begin{tikzcd} \mathcal{D} \arrow[r, "Rep"] \arrow[d, "q"'] & \mathcal{G} \arrow[d, "q"] \\ \mathcal{D} \arrow[r, "Rep"'] & \mathcal{G} \end{tikzcd}$$

OR

$$Rep([\![q]\!]_D) = [\![q]\!]_{Rep(D)}$$

Correctness of SPARQL query evaluation for RDF[i]

Does query evaluation compute the correct answer
(the answer agrees with the semantic definition)?

The following diagram should commute. Does it?

$$
\begin{tikzcd}
\mathcal{D} \arrow[r, "Rep"] \arrow[d, "q"'] & \mathcal{G} \arrow[d, "q"] \\
\mathcal{D} \arrow[r, "Rep"'] & \mathcal{G}
\end{tikzcd}
$$

OR

$$Rep(\llbracket q \rrbracket_D) = \llbracket q \rrbracket_{Rep(D)}$$

Correctness of SPARQL query evaluation for RDF^i

Does query evaluation compute the correct answer
(the answer agrees with the semantic definition)?

The following diagram should commute. Does it?

$$\begin{tikzcd}
\mathcal{D} \arrow[r, "Rep"] \arrow[d, "q"'] & \mathcal{G} \arrow[d, "q"] \\
\mathcal{D} \arrow[r, "Rep"'] & \mathcal{G}
\end{tikzcd}$$

OR

$$\bigcap Rep(\llbracket q \rrbracket_D) = \bigcap \llbracket q \rrbracket_{Rep(D)}$$

Certain answer to the rescue

Definition

The certain answer to query q over a set of RDF graphs $\mathcal{G}$ is set

$$\bigcap \{ [\![q]\!]_G \mid G \in \mathcal{G} \}$$

Certain answer to the rescue

Definition

The certain answer to query q over a set of RDF graphs $\mathcal{G}$ is set

$$\bigcap \{ [\![q]\!]_G \mid G \in \mathcal{G} \}$$

Using the notion of certain answer we can relax the earlier equality requirement to one that uses $\mathcal{Q}$-equivalence.

Certain answer to the rescue

Definition

The certain answer to query q over a set of RDF graphs $\mathcal{G}$ is set

$$\bigcap \{ [\![q]\!]_G \mid G \in \mathcal{G} \}$$

Using the notion of certain answer we can relax the earlier equality requirement to one that uses $\mathcal{Q}$-equivalence.

Definition

Let $\mathcal{Q}$ be a fragment of SPARQL. Two sets of RDF graphs $\mathcal{G}, \mathcal{H}$ will be $\mathcal{Q}$-equivalent (denoted by $\mathcal{G} \equiv_{\mathcal{Q}} \mathcal{H}$) if they give the same certain answer to every query $q \in \mathcal{Q}$

$$\bigcap \{ [\![q]\!]_G \mid G \in \mathcal{G} \} = \bigcap \{ [\![q]\!]_H \mid H \in \mathcal{H} \}$$

Representation system

Let

- $\mathcal{D}$ be the set of all RDF$^{\text{i}}$ databases
- $\mathcal{G}$ be the set of all RDF graphs
- $Rep : \mathcal{D} \to \mathcal{G}$ be a function determining the set of possible RDF graphs corresponding to an RDF$^{\text{i}}$ database, and
- $\mathcal{Q}$ be a fragment of SPARQL

$\langle \mathcal{D}, Rep, \mathcal{Q} \rangle$ is a representation system if for all $D \in \mathcal{D}$ and all $q \in \mathcal{Q}$, there exists an RDF$^{\text{i}}$ database $[\![q]\!]_D$ such that

$$Rep([\![q]\!]_D) \equiv_{\mathcal{Q}} [\![q]\!]_{Rep(D)}$$

Representation system

Let

- $\mathcal{D}$ be the set of all RDF^i databases
- $\mathcal{G}$ be the set of all RDF graphs
- $Rep : \mathcal{D} \rightarrow \mathcal{G}$ be a function determining the set of possible RDF graphs corresponding to an RDF^i database, and
- $\mathcal{Q}$ be a fragment of SPARQL

$\langle \mathcal{D}, Rep, \mathcal{Q} \rangle$ is a representation system if for all $D \in \mathcal{D}$ and all $q \in \mathcal{Q}$, there exists an RDF^i database $[\![q]\!]_D$ such that

$$Rep([\![q]\!]_D) \equiv_{\mathcal{Q}} [\![q]\!]_{Rep(D)}$$

Are there interesting fragments $\mathcal{Q}$ of SPARQL that lead to a representation system?

Representation systems for RDF^i

Theorem

The following fragments of SPARQL can give us representation systems for RDF^i (with D and Rep as defined):

- $\mathcal{Q}^{C}_{AUF}$: CONSTRUCT *queries using only AND, UNION, and FILTER graph patterns, and without blank nodes in their templates*
- $\mathcal{Q}^{C}_{WD}$: CONSTRUCT *queries using only well-designed graph patterns, and without blank nodes in their templates*

Well-designed graph patterns [Pérez/Arenas/Gutierrez '06]

- AND, FILTER, OPT fragment
- P FILTER R: **safe**
- P_1 OPT P_2: variables in P_2 are **properly scoped**

Representation systems for RDF^{i} (cont'd)

Monotonicity

Definition

A fragment $\mathcal{Q}$ of SPARQL is monotone if for every $q \in \mathcal{Q}$ and RDF graphs G and H such that $G \subseteq H$, it is $[\![q]\!]_G \subseteq [\![q]\!]_H$.

Proposition [Arenas/Pérez '11]

- The fragment of SPARQL corresponding to AND, UNION, and FILTER graph patterns is monotone.
- The fragment of SPARQL corresponding to well-designed graph patterns is weakly-monotone ($\sqsubseteq$).

Proposition

Fragments $\mathcal{Q}^{C}_{AUF}$ and $\mathcal{Q}^{C}_{WD}$ are monotone.

Computing certain answers

- Representation systems guarantee correctness of query evaluation for RDF^{i} and SPARQL
- Query evaluation computes an RDF^{i} database

$$[\![q]\!]_D = D' = (G', \phi)$$

- How could we compute the certain answer?

$$\bigcap Rep([\![q]\!]_D)$$

- $Rep([\![q]\!]_D)$ is infinite!

Computing certain answers (cont'd)

Theorem

For $D = (G, \phi)$ and q from $\mathcal{Q}^C_{AUF}$ or $\mathcal{Q}^C_{WD}$, the certain answer of q over D can be computed as follows:

i) *compute $[\![q]\!]_D = D_q = (G_q, \phi)$,*

ii) *compute the RDF^i database $(H_q, \phi) = ((D_q)^{\mathrm{EQ}})^*$, and*

iii) *return the set of RDF triples*

$$\{(s,p,o) \mid ((s,p,o),\theta) \in H_q \text{ such that } \phi \models \theta \text{ and } o \notin U\}$$

The certainty problem

$$CERT(q, H, D)$$

Input

An RDF graph H, a $\mathrm{CONSTRUCT}$ query q, and an RDF^i database D

Question

Does H belong to the certain answer of q over D?

$$H \subseteq \bigcap [\![q]\!]_{Rep(D)}?$$

The certainty problem

$$CERT(q, H, D)$$

Input

An RDF graph H, a $\mathrm{CONSTRUCT}$ query q, and an RDF$^{\mathrm{i}}$ database D

Question

Does H belong to the certain answer of q over D?

$$H \subseteq \bigcap [\![q]\!]_{Rep(D)}?$$

We study the data complexity of $CERT(q, H, D)$

- H and D are part of the input
- q is fixed

Deciding the certainty problem

Theorem

$CERT(q, H, D)$ is equivalent to deciding whether formula

$$\bigwedge_{t \in H} (\forall \_\mathbf{l})(\phi(\_\mathbf{l}) \supset \Theta(t, q, D, \_\mathbf{l}))$$

is true

- *$\_\mathbf{l}$ is the vector of all e-literals in D*
- *$\Theta(t, q, D, \_\mathbf{l})$ is of the form $\theta_1 \vee \cdots \vee \theta_k$, where θ_i is a conjunction of $\mathcal{L}$-constraints*

Computational complexity

Problem	$\mathcal{L}$	**data complexity**
$CERT(q, H, D)$	ECL/diPCL/dePCL/RCL	coNP-complete
	TCL/PCL (RCC-5)	EXPTIME

Computational complexity

Problem	$\mathcal{L}$	data complexity
$CERT(q, H, D)$	ECL/diPCL/dePCL/RCL	coNP-complete
	TCL/PCL (RCC-5)	EXPTIME

Problem	combined complexity	data complexity
SPARQL	PSPACE-complete	LOGSPACE
$\text{SPARQL}_{\text{AUF}}$	NP-complete	
$\text{SPARQL}_{\text{WD}}$	coNP-complete	

Conclusions

RDF^{i} framework

- Modeling of incomplete information for property values
- Formal semantics through possible worlds semantics
- SPARQL query evaluation and certain answer semantics
- Two representation systems for RDF^{i} and SPARQL
- Algorithm for certain answer computation
- Preliminary complexity analysis

Future work

- More general models of incomplete information (subject, predicate)
- More refined complexity results
- Scalable implementation when $\mathcal{L}$ expresses topological constraints with/without constants (TCL/PCL)
- Connection with query processing for the topology vocabulary extension of GeoSPARQL
- Probabilistic extension to RDF^i
- Data integration theory for linked data (only practice exists so far)
- Connection to geospatial OBDA using DL logics

Thank you

Constraint languages $\mathcal{L}$

Properties of $\mathcal{L}$

- Many-sorted first-order language
- Interpreted over a fixed (intended) structure $\mathbf{M}_{\mathcal{L}}$
- EQ: distinguished equality predicate
- $\mathcal{L}$-constraints: quantifier-free formulae of $\mathcal{L}$
- Weakly closed under negation: the negation of every atomic $\mathcal{L}$-constraint is equivalent to a disjunction of $\mathcal{L}$-constraints

Correctness of SPARQL query evaluation for RDF[i]

An easy negative example

Example (classical RDF - OWA)

D

```
s p o .
```

q

```
CONSTRUCT { s ?p ?o }
WHERE { s ?p ?o }
```

Correctness of SPARQL query evaluation for RDF[i]

An easy negative example

Example (classical RDF - OWA)

D

```
s p o .
```

q

```
CONSTRUCT { s ?p ?o }
WHERE { s ?p ?o }
```

Then,

$$[\![q]\!]_D = D$$

Correctness of SPARQL query evaluation for RDF$^{\mathrm{i}}$ (cont'd)

An easy negative example

Example

Let us compare the the set of graphs represented by $[\![q]\!]_D$ with $[\![q]\!]_{Rep(D)}$

Correctness of SPARQL query evaluation for RDF^{i} (cont'd)

An easy negative example

Example

Let us compare the the set of graphs represented by $[\![q]\!]_D$ with $[\![q]\!]_{Rep(D)}$

$$Rep([\![q]\!]_D) = \left\{ \left\{ \begin{array}{c} \text{(s, p, o)} \\ \ \end{array} \right\}, \left\{ \begin{array}{c} \text{(s, p, o)} \\ \text{(c, d, e)} \end{array} \right\}, \left\{ \begin{array}{c} \text{(s, p, o)} \\ \text{(s, b, c)} \end{array} \right\}, \dots \right\}$$

Correctness of SPARQL query evaluation for RDF$^{\mathsf{i}}$ (cont'd)

An easy negative example

Example

Let us compare the the set of graphs represented by $[\![q]\!]_D$ with $[\![q]\!]_{Rep(D)}$

$$Rep([\![q]\!]_D) = \left\{ \left\{ \begin{array}{c} (\mathsf{s},\ \mathsf{p},\ \mathsf{o}) \\ \end{array} \right\}, \left\{ \begin{array}{c} (\mathsf{s},\ \mathsf{p},\ \mathsf{o}) \\ (\mathsf{c},\ \mathsf{d},\ \mathsf{e}) \end{array} \right\}, \left\{ \begin{array}{c} (\mathsf{s},\ \mathsf{p},\ \mathsf{o}) \\ (\mathsf{s},\ \mathsf{b},\ \mathsf{c}) \end{array} \right\}, \ldots \right\}$$

$$[\![q]\!]_{Rep(D)} = \left\{ \left\{ \begin{array}{c} (\mathsf{s},\ \mathsf{p},\ \mathsf{o}) \\ \end{array} \right\}, \left\{ \begin{array}{c} (\mathsf{s},\ \mathsf{p},\ \mathsf{o}) \\ (\mathsf{s},\ \mathsf{b},\ \mathsf{c}) \end{array} \right\}, \ldots \right\}$$

Correctness of SPARQL query evaluation for RDF^{i} (cont'd)

An easy negative example

Example

Let us compare the the set of graphs represented by $[\![q]\!]_D$ with $[\![q]\!]_{Rep(D)}$

$$Rep([\![q]\!]_D) = \left\{ \left\{ \begin{array}{c} (s, p, o) \\ \end{array} \right\}, \left\{ \begin{array}{c} (s, p, o) \\ (c, d, e) \end{array} \right\}, \left\{ \begin{array}{c} (s, p, o) \\ (s, b, c) \end{array} \right\}, \ldots \right\}$$

$$[\![q]\!]_{Rep(D)} = \left\{ \left\{ \begin{array}{c} (s, p, o) \\ \end{array} \right\}, \left\{ \begin{array}{c} (s, p, o) \\ (s, b, c) \end{array} \right\}, \ldots \right\}$$

There is no $g \in [\![q]\!]_{Rep(D)}$ containing the triple (c, d, e)!

Correctness of SPARQL query evaluation for RDF$^{\mathrm{i}}$ (cont'd)

An easy negative example

Example

Let us compare the the set of graphs represented by $[\![q]\!]_D$ with $[\![q]\!]_{Rep(D)}$

$$Rep([\![q]\!]_D) = \left\{ \left\{ \begin{array}{c} (\mathrm{s,\ p,\ o}) \\ \\ \end{array} \right\}, \left\{ \begin{array}{c} (\mathrm{s,\ p,\ o}) \\ (\mathrm{c,\ d,\ e}) \end{array} \right\}, \left\{ \begin{array}{c} (\mathrm{s,\ p,\ o}) \\ (\mathrm{s,\ b,\ c}) \end{array} \right\}, \dots \right\}$$

$$[\![q]\!]_{Rep(D)} = \left\{ \left\{ \begin{array}{c} (\mathrm{s,\ p,\ o}) \\ \\ \end{array} \right\}, \left\{ \begin{array}{c} (\mathrm{s,\ p,\ o}) \\ (\mathrm{s,\ b,\ c}) \end{array} \right\}, \dots \right\}$$

There is no $g \in [\![q]\!]_{Rep(D)}$ containing the triple (c, d, e)!

- This would work if RDF made the CWA

Correctness of SPARQL query evaluation for RDF^{i} (cont'd)

An easy negative example

Example

Let us compare the the set of graphs represented by $[\![q]\!]_D$ with $[\![q]\!]_{Rep(D)}$

$$Rep([\![q]\!]_D) = \left\{ \left\{ \begin{array}{c} (s, p, o) \\ \end{array} \right\}, \left\{ \begin{array}{c} (s, p, o) \\ (c, d, e) \end{array} \right\}, \left\{ \begin{array}{c} (s, p, o) \\ (s, b, c) \end{array} \right\}, \dots \right\}$$

$$[\![q]\!]_{Rep(D)} = \left\{ \left\{ \begin{array}{c} (s, p, o) \\ \end{array} \right\}, \left\{ \begin{array}{c} (s, p, o) \\ (s, b, c) \end{array} \right\}, \dots \right\}$$

There is no $g \in [\![q]\!]_{Rep(D)}$ containing the triple (c, d, e)!

- This would work if RDF made the CWA
- We know this already from the relational case [Imielinski/Lipski '84]

Computing certain answers

Definitions

Definition (EQ-completion)

The EQ-completed form of $D = (G, \phi)$, denoted by $D^{EQ} = (G^{EQ}, \phi)$, is taken from D by replacing all e-literals $\_l \in U$ appearing in G by the constant $c \in C$ such that $\phi \models \_l$ EQ c

Computing certain answers

Definitions

Definition (EQ-completion)

The EQ-completed form of $D = (G, \phi)$, denoted by $D^{EQ} = (G^{EQ}, \phi)$, is taken from D by replacing all e-literals $\_l \in U$ appearing in G by the constant $c \in C$ such that $\phi \models \_l \text{ EQ } c$

Definition (Normalization)

The normalized form of D is the RDF^i database $D^* = (G^*, \phi)$ where G^* is the set

$$\{(t, \theta) \mid (t, \theta_i) \in G \text{ for all } i = 1 \ldots n, \text{ and } \theta \text{ is } \bigvee_i \theta_i\}$$

$$G = \{(t, \theta_1), (t, \theta_2), (t', \theta')\}$$

$$G^* = \{(t, \theta_1 \vee \theta_2), (t', \theta')\}$$