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\textsuperscript{i} framework

SPARQL query evaluation over RDF\textsuperscript{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\(^i\): It captures incomplete information for property values using constraints. It is for RDF what the c-tables model is for the relational model.
Example

```
hotspot1  type  Hotspot  .
f1        type  Fire    .
hotspot1  correspondsTo  f1  .
f1        occurredIn _R1  .
```

Diagram:

- hotspot1: located in the range x ≥ 6 ∧ x ≤ 23, y ≥ 8 ∧ y ≤ 19
- fire1: located within the area of hotspot1

C. Nikolaou and M. Koubarakis – Incomplete Information in RDF
Example

hotspot1 type Hotspot .
fire1 type Fire .
hotspot1 correspondsTo fire1 .
fire1 occurredIn _R1 .

_R1 NTPP "x \geq 6 \land x \leq 23 \land y \geq 8 \land y \leq 19"
RDF\textsuperscript{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\textsuperscript{i} databases: pair $(G, \phi)$

- $G$: RDF graph with a new kind of literals, called e-literals
- $\phi$: 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: $x \text{ EQ } y, x \text{ EQ } c$
- **Blank nodes as existential variables**
Constraint languages $\mathcal{L}$

Examples

ECL

- Equality constraints interpreted over an infinite domain: $x \text{ EQ } y, x \text{ EQ } c$
- 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: $x \text{ EQ } y, x \text{ EQ } c$
- 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: DC, EC, PO, EQ, TPP, NTPP
Constraint languages $\mathcal{L}$

Examples

**ECL**
- Equality constraints interpreted over an infinite domain: $x \text{ EQ } y, x \text{ EQ } c$
- 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: DC, EC, PO, EQ, TPP, NTPP

**PCL**
- TCL plus constant symbols representing polygons in $\mathbb{Q}^2$
- e.g.,
  \[ r \text{ NTPP } "x - y \geq 0 \land x \leq 1 \land y \geq 0" \]
RDF\textsuperscript{i}: Vocabulary

| RDF                  | RDF\textsuperscript{i} | \(\mathcal{L}\)             |
|----------------------|-------------------------|------------------------------|
| \(I\) (IRIs)         | \(I\)                   | constants                    |
| \(B\) (blank nodes)  | \(B\)                   | variables                    |
| \(L\) (literals)     | \(L\)                   | set of sorts                 |
| \(M\) (datatype map) | \(M\)                   |                              |
| \(C\) (literals)     |                         |                              |
| \(U\) (e-literals)   |                         |                              |
### RDF\(^i\): Vocabulary

| RDF          | RDF\(^i\) | \(\mathcal{L}\) |
|--------------|-----------|------------------|
| 1 (IRIs)     | 1         |                  |
| B (blank nodes) | B         |                  |
| L (literals) | L         |                  |
|               | C (literals) | constants       |
|               | U (e-literals) | variables      |
| M (datatype map) | M         |                  |
|               | A (datatypes) | set of sorts    |

\(M_{\mathcal{L}}\) interprets the constants of \(\mathcal{L}\) in agreement with function \(L2V\) of \(M\)
**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
RDF$^i$: Syntax

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 $\theta$ a conjunction of $\mathcal{L}$-constraints, then the pair $(t, \theta)$ is called a conditional triple

$I$: IRIs
$B$: blank nodes
$L$: literals
$C$: constants of $\mathcal{L}$
$U$: e-literals
**RDF⁰: Syntax**

![Diagram](image)

- **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 \(\theta\) a conjunction of \(\mathcal{L}\)-constraints, then the pair \((t, \theta)\) is called a **conditional triple**
- A set of conditional triples is called a **conditional graph**
**RDF\textsuperscript{i}: Syntax (cont’d)**

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

**Example**

- `hotspot1` type `Hotspot`.
- `fire1` type `Fire`.
- `hotspot1` correspondsTo `fire1`.
- `fire1` occurredIn `_R1`.

_R1 NTPP \( x \geq 6 \land x \leq 23 \land y \geq 8 \land y \leq 19 \)
RDF\textsuperscript{i}: Semantics

\textbf{RDF}\textsuperscript{i} database

\[ D \quad \xrightarrow{\text{Rep}} \quad \{ G_1, G_2, \ldots \} \]

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 \( \{ v(t) \mid (t, \theta) \in G \text{ and } M|_L = v(\theta) \} \).
**RDF^i: Semantics**

**RDF^i database**

\[ D \xrightarrow{Rep} \{ G_1, G_2, \ldots \} \]

**Definition**

A valuation \( \nu \) 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 \( \nu \) a valuation. Then \( \nu(G) \) denotes the RDF graph

\[ \{ \nu(t) \mid (t, \theta) \in G \text{ and } M_L \models \nu(\theta) \} \]
RDF\textsuperscript{i}: Semantics (cont’d)

From RDF\textsuperscript{i} databases to sets of RDF graphs

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

$$Rep(D) = \left\{ H \mid \text{there exists valuation } \nu \text{ and RDF graph } H \right. 
\left. \text{such that } M_\mathcal{L} \models \nu(\phi) \text{ and } H \supseteq \nu(G) \right\}$$

- Relation $\supseteq$ captures the OWA semantics
- An RDF\textsuperscript{i} database corresponds to an infinite number of RDF graphs
How can we evaluate a query \( q \) over an RDF\(^i \) database \( D \) (compute \( \llbracket q \rrbracket_D \))?
Question

How can we evaluate a query $q$ over an RDF database $D$ (compute $\llbracket q \rrbracket_D$)?

Semantic definition

$$\llbracket q \rrbracket_{Rep(D)} = \{ \llbracket q \rrbracket_G \mid G \in Rep(D) \}$$
Question

How can we evaluate a query $q$ over an RDF database $D$ (compute $\mathbb{D}_q D$)?

Semantic definition

$$\mathbb{D}_q Rep(D) = \{ \mathbb{D}_q G \mid G \in Rep(D) \}$$
Question

How can we evaluate a query \( q \) over an RDF\(^i \) database \( D \) (compute \( \llbracket q \rrbracket_D \))?

Semantic definition

\[
\llbracket q \rrbracket_{Rep(D)} = \{ \llbracket q \rrbracket_G \mid G \in Rep(D) \}
\]

In practice?

- Start with SPARQL algebra of [Pérez/Arenas/Gutierrez ’06] with set semantics
- Define SPARQL query evaluation for RDF\(^i \) databases
From mappings to e-mappings...

\{ ?F \rightarrow \text{fire1}, \ ?S \rightarrow "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \}
From mappings to e-mappings...

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

\{?F \rightarrow \text{fire1}, ?S \rightarrow _{\text{R1}}\}\}
... to conditional mappings

\{ ?F \rightarrow \text{fire1}, \ ?S \rightarrow "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \}
to conditional mappings

$$\left( \{ ?F \rightarrow \text{fire1}, ?S \rightarrow "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \} , \text{true} \right)$$
... to conditional mappings

\[
\{ ?F \rightarrow \text{fire1}, ?S \rightarrow \_R1 \}, \_R1 \text{ EQ "} x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2 \text{"} \}
\]
From compatible mappings to possibly compatible mappings

Join of conditional mappings

\[
\left( \{ ?F \to fire1, \ ?S \to _R1 \}, \ _R1 \ EQ \ "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \right)
\]

\[
\left( \{ \ ?S \to _R2 \}, \ true \right)
\]
From compatible mappings to possibly compatible mappings

Join of conditional mappings

\[
\left( \{ ?F \rightarrow \text{fire}_1, \ ?S \rightarrow \_R1 \}, \ _R1 \text{ EQ } "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \right)
\]

\[
\left( \{ ?S \rightarrow \_R2 \}, \ true \right)
\]
From compatible mappings to possibly compatible mappings

Join of conditional mappings

$$\left( \left\{ ?F \rightarrow \text{fire1}, \ ?S \rightarrow \_R1 \right\}, \_R1 \ \text{EQ} \ "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \right)$$

$\Join$

$$\left( \left\{ \ ?S \rightarrow \_R2 \right\}, \ \text{true} \right)$$

$$= \null$$
From compatible mappings to possibly compatible mappings

Join of conditional mappings

\[
\left( \{\ ?F \rightarrow \text{fire1}, \ ?S \rightarrow _R1 \}, \ _R1 \ EQ "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \right)
\]

\times

\left( \{\ ?S \rightarrow _R2 \}, \ true \right)

= 

\left( \{\ ?F \rightarrow \text{fire1}, \ ?S \rightarrow _R1 \}, \ true \land _R1 \ EQ _R2 \land _R1 \ EQ "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \right)
Operations on conditional mappings

Let \( \Omega_1 \) and \( \Omega_2 \) be sets of conditional mappings. We can define the operation of:

- **Join** \( (\Omega_1 \Join \Omega_2) \)
- **Union** \( (\Omega_1 \cup \Omega_2) \)
- **Difference** \( (\Omega_1 \setminus \Omega_2) \)
- **Left-outer join** \( (\Omega_1 \Join_{left} \Omega_2) \)
Graph pattern evaluation

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

- **base case:** $P$ is the triple pattern $t$
- **recursion:** $P$ is $(P_1 \text{AND} P_2) \rightarrow J P_1 K D \leftrightsquigarrow J P_2 K D$
- $P$ is $(P_1 \text{UNION} P_2) \rightarrow J P_1 K D \cup J P_2 K D$
- $P$ is $(P_1 \text{OPT} P_2) \rightarrow J P_1 K D \downarrow J P_2 K D$
- $P$ is $(P_1 \text{FILTER} R)$ where $R$ is a conjunction of $L$-constraints
Graph pattern evaluation

If $D$ is an RDF 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 $(P_1 \text{ AND } P_2) \rightarrow [P_1]_D \Join [P_2]_D$
- $P$ is $(P_1 \text{ UNION } P_2) \rightarrow [P_1]_D \cup [P_2]_D$
- $P$ is $(P_1 \text{ OPT } P_2) \rightarrow [P_1]_D \Join [P_2]_D$
Graph pattern evaluation

If $D$ is an RDF\textsuperscript{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 $(P_1 \text{ AND } P_2)$ \quad \rightarrow \quad [P_1]_D \bowtie [P_2]_D$
- $P$ is $(P_1 \text{ UNION } P_2)$ \quad \rightarrow \quad [P_1]_D \cup [P_2]_D$
- $P$ is $(P_1 \text{ OPT } P_2)$ \quad \rightarrow \quad [P_1]_D \varpi [P_2]_D$
- $P$ is $(P_1 \text{ FILTER } R)$
  where $R$ is a conjunction of $\mathcal{L}$-constraints
Graph pattern evaluation

If $D$ is an RDF 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 $(P_1 \text{ AND } P_2)$ \quad \rightarrow \quad [P_1]_D \, \boxtimes \, [P_2]_D$
- $P$ is $(P_1 \text{ UNION } P_2)$ \quad \rightarrow \quad [P_1]_D \, \cup \, [P_2]_D$
- $P$ is $(P_1 \text{ OPT } P_2)$ \quad \rightarrow \quad [P_1]_D \, \Join \, [P_2]_D$
- $P$ is $(P_1 \text{ FILTER } R)$

  where $R$ is a conjunction of $\mathcal{L}$-constraints
Triple pattern evaluation (case 1)

Example

Database $D$

```
fire1 occuredIn _R1 .
```

Query $q$

```
?f occurredIn ?R

_R1 NTPP "x ≥ 6 ∧ x ≤ 23 ∧ y ≥ 8 ∧ y ≤ 19"
```
Triple pattern evaluation (case 1)

Example

Database $D$

```
fire1 occurredIn _R1 .
```

Query $q$

```
?F occurredIn ?R
```

_R1 NTPP "x ≥ 6 ∧ x ≤ 23 ∧ y ≥ 8 ∧ y ≤ 19"

Answer (set of conditional mappings)

\[
[q]_D = \{(\{?F \rightarrow \text{fire1}, ?R \rightarrow _R1\}, \text{true})\}
\]
Triple pattern evaluation (case 2)

Example

Database $D$

fire1 occurredIn _R1 .

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

Query $q$

?f occurredIn
"x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2"
Triple pattern evaluation (case 2)

Example

Database $D$

```
fire1 occurredIn _R1 .
_R1 NTPP "x ≥ 6 ∧ x ≤ 23 ∧ y ≥ 8 ∧ y ≤ 19"
```

Query $q$

```
?F occurredIn
"x ≥ 1 ∧ x ≤ 2 ∧ y ≥ 1 ∧ y ≤ 2"
```

Answer (set of conditional mappings)

\[
[q]_D = \left\{ (\{?F \rightarrow \text{fire1}\}, \_R1 \text{ EQ } "x ≥ 1 ∧ x ≤ 2 ∧ y ≥ 1 ∧ y ≤ 2") \right\}
\]
Evaluation of FILTER graph patterns

Example

Database $D$

fire1 occurredIn _R1 .

_R1 NTPP "x ≥ 6 ∧ x ≤ 23 ∧ y ≥ 8 ∧ y ≤ 19"

Query $q$

?F occurredIn ?R .
FILTER (?R NTPP "x ≥ 1 ∧ x ≤ 2 ∧ y ≥ 1 ∧ y ≤ 2")
Evaluation of FILTER graph patterns

Example

Database $D$

fire1 occurredIn _R1 .

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

Query $q$

?F occurredIn ?R .
FILTER (?R NTPP "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2")

Answer

$$\llbracket q \rrbracket_D = \left\{ \{?F \rightarrow \text{fire1}, ?R \rightarrow \_R1\}, \right. \left. \text{ _R1 NTPP "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" } \right\}$$
SELECT queries

Example

Database $D$

$\text{fire1 occurredIn } _{R1}$ .

$\_R1 \text{ NTPP } "x \geq 6 \land x \leq 23 \land y \geq 8 \land y \leq 19"$

Query $q$

SELECT {?F}
WHERE {
  {?F occurredIn ?R .
  FILTER (?R NTPP
    "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2")}
}
**SELECT** queries

**Example**

Database \( D \)

\[
\text{fire1 occurredIn _R1}.
\]

\[
_R1 \text{ NTPP } "x \geq 6 \land x \leq 23 \land y \geq 8 \land y \leq 19"
\]

**Query \( q \)**

\[
\text{SELECT } ?F \\
\text{WHERE } \{ \\
?F \text{ occurredIn } ?R . \\
\text{FILTER } (?R \text{ NTPP } \\
"x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2") \}
\]

**Answer (set of conditional mappings)**

\[
[q]_D = \left\{ \left\{ ?F \rightarrow \text{fire1} \right\}, \\
\_R1 \text{ NTPP } "x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2" \right\}
\]
CONSTRUCT queries

Example

Database $D$

$\text{fire1 occuredIn } _R1$ .

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

Query $q$

CONSTRUCT $\{ \ ?F \text{ type Fire } \}$
WHERE $\{
\quad ?F \text{ occuredIn } ?R
\}$
CONSTRUCT queries

Example

Database $D$

fire1 occurredIn _R1 .

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

Answer (RDF$^i$ database)

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

Query $q$

CONSTRUCT { ?F type Fire }  
WHERE {
  ?F occurredIn ?R
}

fire1 type Fire .

_R1 NTPP "$x \geq 6 \land x \leq 23 \land y \geq 8 \land y \leq 19$"
CONSTRUCT queries

Example

Database \( D \)

\[
\text{fire1 occurredIn } _R1 .
\]

\[
_R1 \text{ NTPP } "x \geq 6 \land x \leq 23 \land y \geq 8 \land y \leq 19"
\]

Answer (RDF\(^i\) database)

\[
D' = (G', \phi) \quad \text{fire1 type Fire} .
\]

\[
_R1 \text{ NTPP } "x \geq 6 \land x \leq 23 \land y \geq 8 \land y \leq 19"
\]

Query \( q \)

\[
\text{CONSTRUCT} \{ \ ?F \text{ type Fire } \}
\]

\[
\text{WHERE} \{
\]

\[
?F \text{ occurredIn } ?R
\]

\[
\}
\]

Closure property
Correctness of SPARQL query evaluation for RDF

Does query evaluation compute the **correct answer** (the answer agrees with the semantic definition)?
Correctness of SPARQL query evaluation for RDF

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

The following diagram should commute:

\[ D \xrightarrow{\text{Rep}} G \]

\[ \downarrow q \]

\[ G \]
Correctness of SPARQL query evaluation for RDF

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

The following diagram should commute

\[ \text{Rep} \]

\[ D \rightarrow G \]

\[ q \]

\[ D \rightarrow G \]

\[ q \]
Correctness of SPARQL query evaluation for RDF

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

The following diagram should commute:

\[
\begin{array}{c}
  D \xrightarrow{Rep} G \\
  q \\
\end{array}
\]

\[
\begin{array}{c}
  D \xrightarrow{Rep} G \\
  q \\
\end{array}
\]
Correctness of SPARQL query evaluation for RDF

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

The following diagram should commute

\[ D \xrightarrow{Rep} G \]

\[ q \]

\[ OR \]

\[ Rep([q]_D) = [q]_{Rep(D)} \]
Correctness of SPARQL query evaluation for RDF

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

The following diagram should commute. Does it?

\[
\begin{align*}
D & \xrightarrow{Rep} G \\
q & \downarrow \quad & q & \uparrow \\
D & \xrightarrow{Rep} G
\end{align*}
\]

OR

\[Rep(\llbracket q \rrbracket_D) = \llbracket q \rrbracket_{Rep(D)}\]
Correctness of SPARQL query evaluation for RDF

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

The following diagram should commute. Does it?

\[
\begin{align*}
\neg D & \xrightarrow{Rep} G \\
q & \downarrow \quad \downarrow \\
D & \xrightarrow{Rep} G \\
\neg q & \uparrow \quad \uparrow \\
\neg D & \xrightarrow{Rep} G \\
\end{align*}
\]

OR

\[
\bigcap \text{Rep}(\llbracket q \rrbracket_D) = \bigcap \llbracket q \rrbracket_{\text{Rep}(D)}
\]
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 $G$ is set
\[
\bigcap \{ [q]_G \mid G \in G \}
\]

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

Definition
Let $Q$ be a fragment of SPARQL. Two sets of RDF graphs $\mathcal{G}, \mathcal{H}$ will be $Q$-equivalent (denoted by $\mathcal{G} \equiv_Q \mathcal{H}$) if they give the same certain answer to every query $q \in Q$
\[
\bigcap \{ [q]_G \mid G \in \mathcal{G} \} = \bigcap \{ [q]_H \mid H \in \mathcal{H} \}
\]
Representation system

Let

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

\( \langle \mathcal{D}, \text{Rep}, \mathcal{Q} \rangle \) 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

\[
\text{Rep}([q]_D) \equiv_{\mathcal{Q}} [q]_{\text{Rep}(D)}
\]
Representation system

Let

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

$\langle \mathcal{D}, \text{Rep}, \mathcal{Q} \rangle$ is a representation system if for all $D \in \mathcal{D}$ and all $q \in \mathcal{Q}$, there exists an RDF\textsuperscript{i} database $\llbracket q \rrbracket_D$ such that

$$\text{Rep}(\llbracket q \rrbracket_D) \equiv \mathcal{Q} \llbracket q \rrbracket_{\text{Rep}(D)}$$

Are there interesting fragments $\mathcal{Q}$ of SPARQL that lead to a representation system?
Theorem
The following fragments of SPARQL can give us representation systems for RDF (with $D$ and $Rep$ as defined):

- $Q_{AUF}^C$: CONSTRUCT queries using only AND, UNION, and FILTER graph patterns, and without blank nodes in their templates

- $Q_{WD}^C$: 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
Definition
A fragment \( Q \) of SPARQL is monotone if for every \( q \in Q \) 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 \( Q^C_{AUF} \) and \( Q^C_{WD} \) are monotone.
Computing certain answers

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

\[ \text{Rep}(\mathcal{Q}_D) = D' = (G', \phi) \]

- How could we compute the certain answer?

\[ \bigcap \text{Rep}(\mathcal{Q}_D) \]

- \text{Rep}(\mathcal{Q}_D) is infinite!
Theorem

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

i) compute $\mathbf{[q]}_D = D_q = (G_q, \phi)$,

ii) compute the RDF\textsuperscript{i} database $(H_q, \phi) = ((D_q)^{EQ})^*$, 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 CONSTRUCT query \( q \), and an RDF\(^i\) database \( D \)

**Question**
Does \( H \) belong to the certain answer of \( q \) over \( D \)?

\[ H \subseteq \bigcap \mathbb{R}_{\text{Rep}(D)}[\mathbb{R}_{\text{Rep}(D)}[q]] \]
The certainty problem

\[ \text{CERT}(q, H, D) \]

Input
An RDF graph \( H \), a CONSTRUCT query \( q \), and an RDF database \( D \)

Question
Does \( H \) belong to the certain answer of \( q \) over \( D \)?

\[ H \subseteq \bigcap [q]_{\text{Rep}(D)} \]

We study the data complexity of \( \text{CERT}(q, H, D) \)

- \( H \) and \( D \) are part of the input
- \( q \) is fixed
Deciding the certainty problem

Theorem

\( \text{CERT}(q, H, D) \) is equivalent to deciding whether formula

\[
\bigwedge_{t \in H} (\forall \_l)(\phi(\_l) \supset \Theta(t, q, D, \_l))
\]

is true

- \( \_l \) is the vector of all e-literals in \( D \)
- \( \Theta(t, q, D, \_l) \) is of the form \( \theta_1 \lor \cdots \lor \theta_k \), where \( \theta_i \) 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           |

C. Nikolaou and M. Koubarakis – Incomplete Information in RDF
## Computational complexity

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

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

**RDF\textsuperscript{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\textsuperscript{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

An easy negative example

Example (classical RDF - OWA)

\[
\begin{align*}
D & = q \\
\text{CONSTRUCT} & \{ \text{s ?p ?o} \} \\
\text{WHERE} & \{ \text{s ?p ?o} \}
\end{align*}
\]
Correctness of SPARQL query evaluation for RDF

An easy negative example

Example (classical RDF - OWA)

\[ D \rightarrow q \]

\[ s \, p \, o . \]

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

Then,

\[ [q]_D = D \]
Correctness of SPARQL query evaluation for RDF\textsuperscript{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 (cont’d)

An easy negative example

Example

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

$$Rep([q]_D) = \left\{ \left\{ (s, p, o) \right\}, \left\{ (s, p, o), (c, d, e) \right\}, \left\{ (s, p, o), (s, b, c) \right\}, \cdots \right\}$$
An easy negative example

Example

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

\[
Rep([q]_D) = \left\{ \left\{ (s, p, o) \right\}, \left\{ (s, p, o) \right\}, \left\{ (s, p, o) \right\}, \ldots \right\}
\]

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

There is no $g \in [q]_D$ containing the triple $(c, d, e)$!
Correctness of SPARQL query evaluation for RDF (cont’d)

An easy negative example

Example

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

$$Rep(\[ q \]_D) = \left\{ \left\{ (s, p, o) \right\}, \left\{ (s, p, o) \right\}, \left\{ (s, b, c) \right\}, \ldots \right\}$$

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

There is no $g \in \[ q \]_{Rep(D)}$ containing the triple $(c, d, e)$!
Correctness of SPARQL query evaluation for RDF

An easy negative example

Example

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

\[
Rep([q]_{D}) = \left\{ \{ (s, p, o) \}, \{ (s, p, o) \}, \{ (s, p, o) \}, \ldots \right\}
\]

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

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 (cont’d)

An easy negative example

Example

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

\[
\begin{align*}
\text{Rep}(\[[q]_D\]) &= \left\{ \left\{ (s, p, o) \right\}, \left\{ (s, p, o) \right\}, \left\{ (s, p, o) \right\}, \cdots \right\} \\
\text{Rep}(\[[q]_{Rep(D)}\]) &= \left\{ \left\{ (s, p, o) \right\}, \left\{ (s, p, o) \right\}, \left\{ (s, p, o) \right\}, \cdots \right\}
\end{align*}
\]

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 $\phi \models l \text{ EQ } c$.
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 $\phi \models l$ EQ $c$

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

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

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

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