Incomplete Information in RDF using Constraints

Manolis Koubarakis

Joint work with Charalampos Nikolaou

Department of Informatics and Telecommunications
National and Kapodistrian University of Athens

9th Panhellenic Logic Symposium 2013 (PLS9)
July 18, 2013

Outline

- Motivation
- Previous work
- The RDF^i framework
- SPARQL query evaluation over RDF^i databases
- Representation systems for RDF^i and SPARQL
- An algorithm for certain answer computation
- Preliminary complexity results
- Applications
- Conclusions and future work

Motivation

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

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/Cirangelo '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.

The RDF^i framework

RDF^i by example

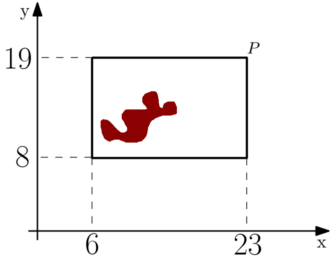
Example

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

RDF^i by example

Example

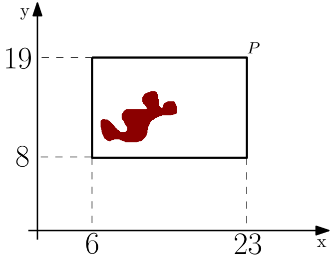
hotspot1	type	Hotspot	.
fire1	type	Fire	.
hotspot1	correspondsTo	fire1	.
fire1	occuredIn	_R1	.



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^{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 $\mathcal{L}$

Syntax

RDF graphs extended to RDF^{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 $\mathcal{L}$

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

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

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

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

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

Constraint languages $\mathcal{L}$ (cont'd)

Examples

ECL

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

Constraint languages $\mathcal{L}$ (cont'd)

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 $\mathcal{L}$ (cont'd)

Examples

ECL

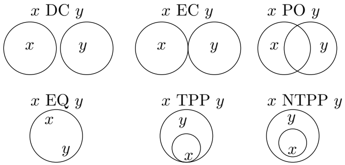
- Equality constraints interpreted over an infinite domain: x EQ y, x EQ c
- Blank nodes as existential variables

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 $\mathcal{L}$ (cont'd)

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^i: Vocabulary

RDF
I (IRIs) B (blank nodes) L (literals) M (datatype map)

RDF^{i}: Vocabulary

RDF	RDF^{i}
I (IRIs)	I
B (blank nodes)	B
L (literals)	L
M (datatype map)	M

RDF^i: Vocabulary

RDF	RDF^i
I (IRIs)	I
B (blank nodes)	B
L (literals)	L
	C (literals)
	U (e-literals)
M (datatype map)	M
	A (datatypes)

RDF^{i}: Vocabulary

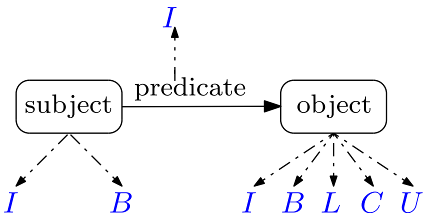
RDF	RDF^{i}	$\mathcal{L}$
I (IRIs)	I	
B (blank nodes)	B	
L (literals)	L	
	C (literals)	constants
	U (e-literals)	variables
M (datatype map)	M	
	A (datatypes)	set of sorts

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

RDF	RDF$^{\text{i}}$	$\mathcal{L}$
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^{i}: Syntax

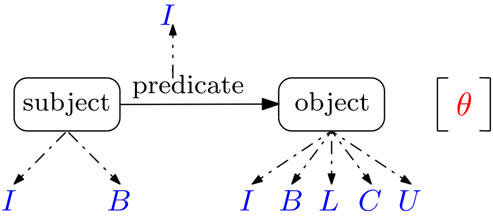


I : IRIs
B: blank nodes
L : literals
C: constants of $\mathcal{L}$
U: e-literals

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

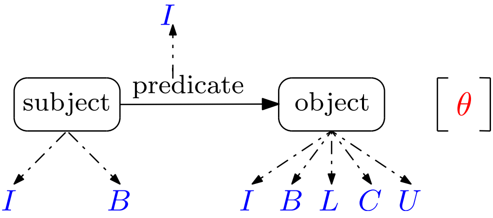


I : IRIs
B: blank nodes
L : literals
C: constants of $\mathcal{L}$
U: e-literals

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



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)

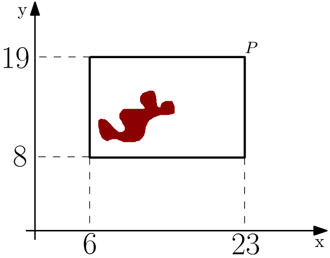
Definition

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

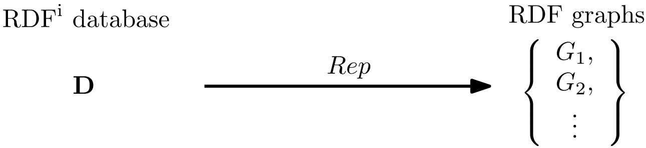


RDF$^{\text{i}}$: Semantics

RDF$^{\text{i}}$ database $\qquad\qquad$ RDF graphs

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

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



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$^{\text{i}}$ 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^{i} database D (compute $[\![q]\!]_D$)?

Semantic definition

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

In practice?

SPARQL query evaluation over RDF^i databases

Query evaluation highlights

- 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...

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

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

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

... to conditional mappings

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

... to conditional mappings

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

From compatible mappings to possibly compatible mappings

Join of conditional mappings

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

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

From compatible mappings to possibly compatible mappings

Join of conditional mappings

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

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

From compatible mappings to possibly compatible mappings

Join of conditional mappings

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

$$\bowtie$$

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

$$=$$

From compatible mappings to possibly compatible mappings

Join of conditional mappings

$$\Big(\{?F \rightarrow \mathit{fire}1, \;\; ?S \rightarrow \_R1\},\ \_R1\ EQ\ ''x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2''\Big)$$

$$\bowtie$$

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

$$=$$

$$\Big(\{?F \rightarrow \mathit{fire}1, \;\; ?S \rightarrow \_R1\},\ \mathit{true}\ \land\ \_R1\ EQ\ \_R2\ \land$$
$$\_R1\ EQ\ ''x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 2''\Big)$$

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^{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 $(P_1 \text{ OPT } P_2) \quad \rightarrow \quad [\![P_1]\!]_D ⟕ [\![P_2]\!]_D$

P is $(P_1 \text{ FILTER } R)$

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

Graph pattern evaluation

If D is an RDF^{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:

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

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

Query q

```
fire1 occuredIn _R1 .
```

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

```
?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 "$x \geq 6 \wedge x \leq 23 \wedge y \geq 8 \wedge y \leq 19$"

Query q

```
?F occuredIn ?R .
FILTER (?R NTPP
```
"$x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 2$")

Evaluation of FILTER graph patterns

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 .
FILTER (?R NTPP
```
"x $\geq$ 1 $\wedge$ x $\leq$ 2 $\wedge$ y $\geq$ 1 $\wedge$ y $\leq$ 2")

Answer

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

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

Query q

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

Answer (set of conditional mappings)

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

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

Query q

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

Answer (RDF^i database)

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

```
fire1 type Fire .
```

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

Closure property

Representation systems for RDF^i and SPARQL

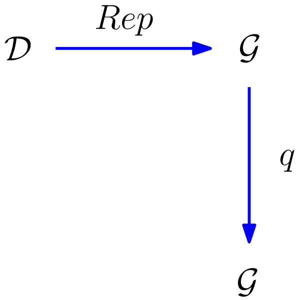
Correctness of SPARQL query evaluation for RDF^i

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)?

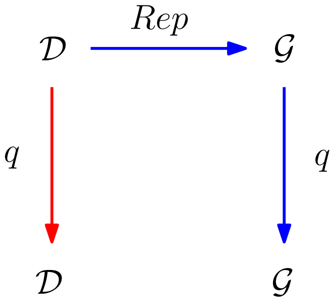
The following diagram should commute



Correctness of SPARQL query evaluation for RDF[i]

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

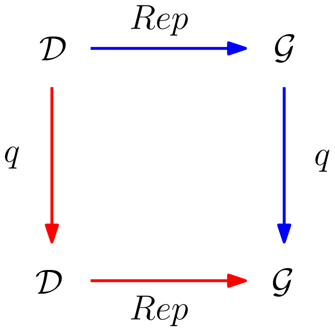
The following diagram should commute



Correctness of SPARQL query evaluation for RDF[i]

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

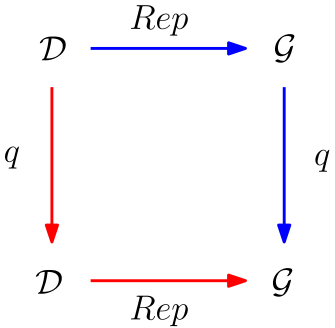
The following diagram should commute



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



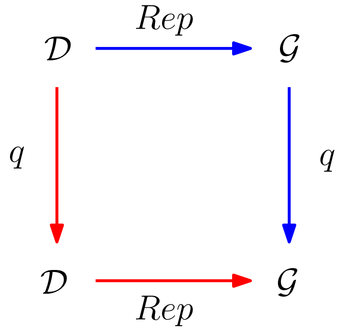
OR

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

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?



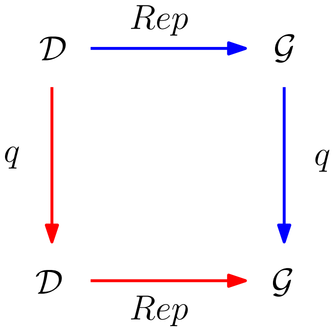
OR

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

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?



OR

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

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

An easy negative example

Example (classical RDF - OWA)

D	q
`s p o .`	`CONSTRUCT { s ?p ?o }` `WHERE { s ?p ?o }`

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

An easy negative example

Example (classical RDF - OWA)

D	q
`s p o .`	`CONSTRUCT { s ?p ?o }` `WHERE { s ?p ?o }`

Then,

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

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

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

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

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

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

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

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^{i} (cont'd)

An easy negative example

Example

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

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

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

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

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

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

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]

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*

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*

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 $[\![q]\!]_G \subseteq [\![q]\!]_H$.

Representation systems for RDF$^{\mathrm{i}}$ (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.

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 ($\sqsubseteq$).

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.

An algorithm for certain answer computation

Computing certain answers

- Representation systems guarantee correctness of query evaluation for RDF^i and SPARQL

Computing certain answers

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

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

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?

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

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)

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 (cont'd)

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

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

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

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

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*

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

Preliminary complexity results

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?

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

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?

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

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	

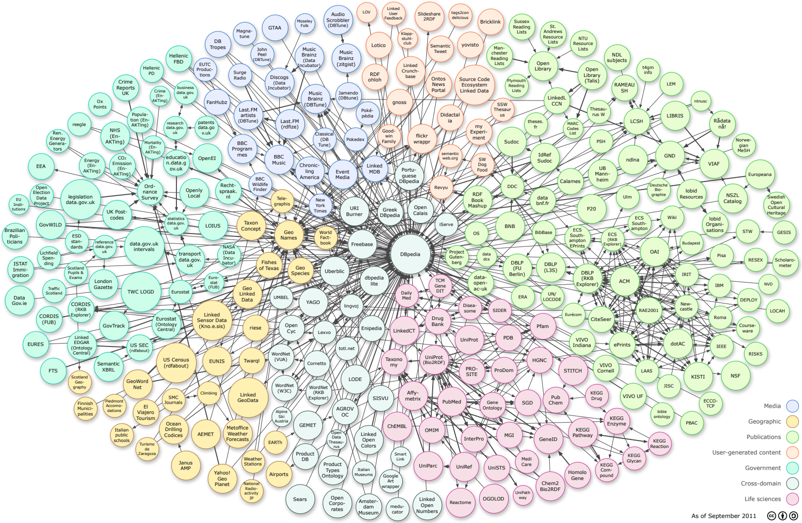
Computational complexity

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	

Applications

Linked geospatial data



Applications of RDF^i for TCL

Many linked geospatial datasets are populated with topological information

Examples:

- Administrative Geography of Great Britain (ADMGB)
- Greek Administrative Geography (GAG)
- Global Administrative Areas (GADM)
- Nomenclature of Territorial Units for Statistics (NUTS)

Applications of RDF^i for TCL (cont'd)

Dataset	triples	regions	RCC-8 relations
ADMGB	149,046	11,762	77,907
GAG	11,780	412	3,023
NUTS	316,246	2,236	3,176
GADM-EUROPE	355,656	23,037	51,309
GADM	9,896,532	27,6728	590,445

Applications of RDF^i for TCL (cont'd)

Dataset	triples	regions	RCC-8 relations
ADMGB	149,046	11,762	77,907
GAG	11,780	412	3,023
NUTS	316,246	2,236	3,176
GADM-EUROPE	355,656	23,037	51,309
GADM	9,896,532	27,6728	590,445

Can we do efficient reasoning for $\phi \models \theta$?

Reasoning example

Example

RDF graph

```
ex:a geo:rcc8tppi ex:b .
ex:b geo:rcc8tppi ex:c .
```

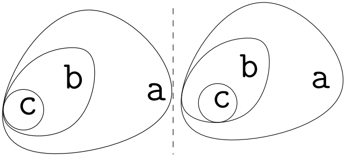
Reasoning example

Example

RDF graph

```
ex:a geo:rcc8tppi ex:b .
ex:b geo:rcc8tppi ex:c .
```

Spatial configuration



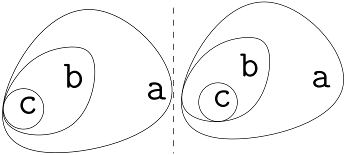
Reasoning example

Example

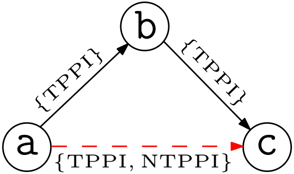
RDF graph

```
ex:a geo:rcc8tppi ex:b .
ex:b geo:rcc8tppi ex:c .
```

Spatial configuration



Constraint network



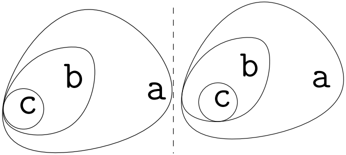
Reasoning example

Example

RDF graph

```
ex:a geo:rcc8tppi ex:b .
ex:b geo:rcc8tppi ex:c .
```

Spatial configuration



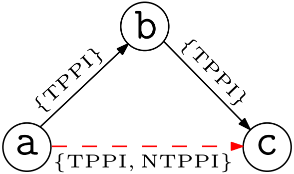
Representation

$TPPI(a, b)$,

$TPPI(b, c)$,

$\{TPPI, NTPPI\}(a, c)$

Constraint network



Reasoning algorithms

In general

- Backtracking algorithms

Reasoning algorithms

In general

- Backtracking algorithms

In particular (tractable cases)

- path-consistency algorithm

Reasoning algorithms

In general

- Backtracking algorithms

In particular (tractable cases)

- path-consistency algorithm
  Iterative execution:

$$\forall i, j, k \; R(i,j) \leftarrow R(i,j) \cap (R(i,k) \circ R(k,j))$$

Symbol $\circ$ is the composition of sets of RCC-8 relations (predefined)

Reasoning algorithms

In general

- Backtracking algorithms

In particular (tractable cases)

- path-consistency algorithm
  Iterative execution:

  $$\forall i, j, k \; R(i,j) \leftarrow R(i,j) \cap (R(i,k) \circ R(k,j))$$

  Symbol $\circ$ is the composition of sets of RCC-8 relations (predefined)
  Memory requirements: $\Theta(n^2)$
  Running time: $O(n^3)$

Implementations of path-consistency

RCC-8 reasoners

- Renz
- PyRCC8
- PPyRCC8

RDF systems

- PelletSpatial
- Strabon

Implementations of path-consistency

RCC-8 reasoners

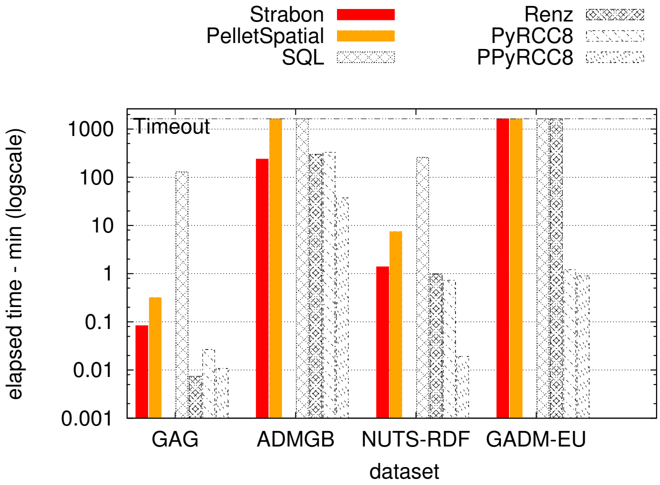
- Renz
- PyRCC8
- PPyRCC8

RDF systems

- PelletSpatial
- Strabon

How do they perform?

Experimental performance



Conclusions and future work

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

- Interesting representation systems

Future work

- Interesting representation systems
- More refined complexity results

Future work

- Interesting representation systems
- More refined complexity results
- Scalable implementation when $\mathcal{L}$ expresses topological constraints with/without constants (TCL/PCL)

Future work

- Interesting representation systems
- More refined complexity results
- Scalable implementation when $\mathcal{L}$ expresses topological constraints with/without constants (TCL/PCL)
- Probabilistic extension to RDF^i

Future work

- Interesting representation systems
- More refined complexity results
- Scalable implementation when $\mathcal{L}$ expresses topological constraints with/without constants (TCL/PCL)
- Probabilistic extension to RDF^i
- Data integration theory for linked data (only practice exists so far)

Future work

- Interesting representation systems
- More refined complexity results
- Scalable implementation when $\mathcal{L}$ expresses topological constraints with/without constants (TCL/PCL)
- Probabilistic extension to RDF^i
- Data integration theory for linked data (only practice exists so far)
- Connection to geospatial OBDA using DL logics

Future work

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

Related papers

Charalampos Nikolaou and Manolis Koubarakis.
Querying linked geospatial data with incomplete information.
In *5th International Terra Cognita Workshop*, 2012.

Charalampos Nikolaou and Manolis Koubarakis.
Incomplete information in RDF.
In *Web Reasoning and Rule Systems (RR'13)*, pages 138–152, 2013.

Charalampos Nikolaou and Manolis Koubarakis.
Incomplete information in RDF.
CoRR, abs/1209.3756, 2012.

Charalampos Nikolaou and Manolis Koubarakis.
Querying incomplete geospatial information in RDF.
In *13th International Symposium on Spatial and Temporal Databases (SSTD'13)*, pages 447–450, 2013.

Thank you