The Challenge of Optional Matching in SPARQL

Reinhard Pichler, TU Wien

(joint work with Shqiponja Ahmetaj, Wolfgang Fischl, Markus Kröll, Mantas Simkus, and Sebastian Skritek)

FoiKS 2016

Roadmap

Roadmap

SPARQL and optional matching

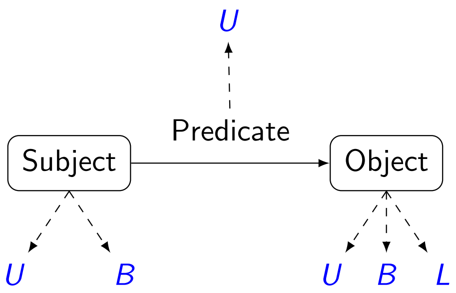
- SPARQL
  - W3C query language for the semantic web
  - important feature: OPTIONAL
  - significant extension of CQs: non-monotonicity

SPARQL and optional matching

- SPARQL
  - W3C query language for the semantic web
  - important feature: OPTIONAL
  - significant extension of CQs: non-monotonicity
- Various challenges:
  - query answering: becomes harder
  - static query analysis: containment vs. subsumption
  - SPARQL entailment regimes: rethinking of the semantics

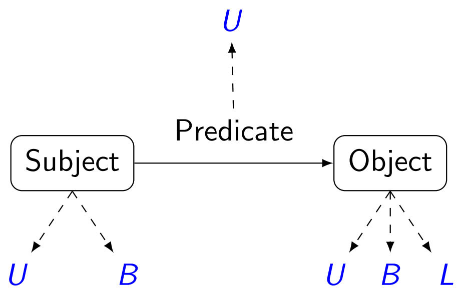
Roadmap

RDF: Resource Description Framework



U = set of Uris
B = set of Blank nodes
L = set of Literals

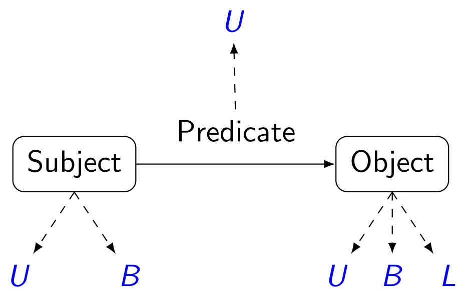
RDF: Resource Description Framework



U = set of Uris
B = set of Blank nodes
L = set of Literals

$(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an RDF triple.

RDF: Resource Description Framework



U = set of Uris

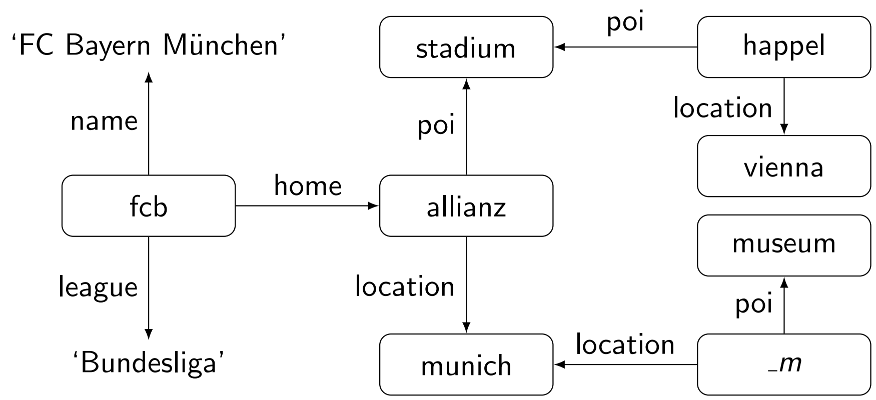
B = set of Blank nodes

L = set of Literals

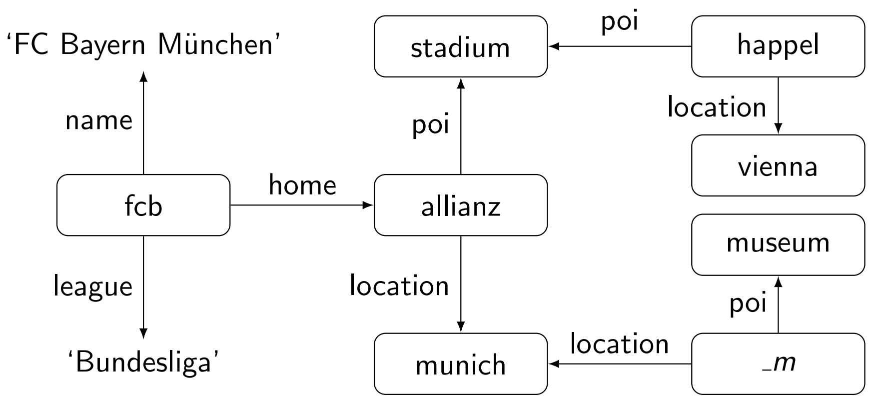
$(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an RDF triple.

A set of RDF triples is called an RDF graph.

RDF: Example

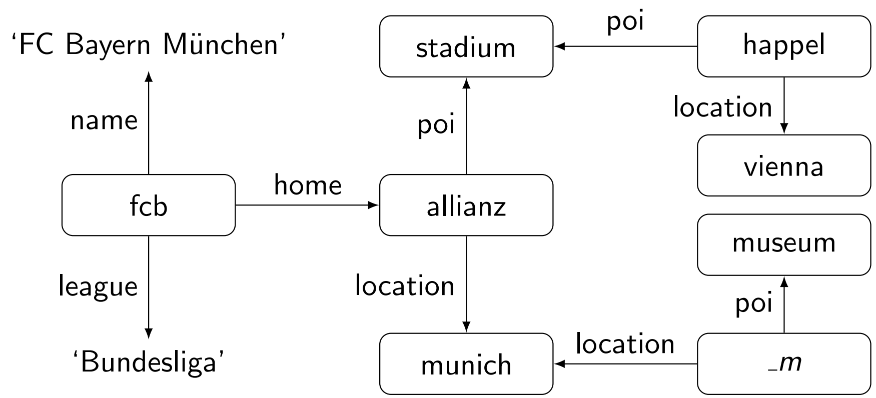


RDF: Example



$\{$(fcb, name, 'FC Bayern München'), (fcb, league, 'Bundesliga'), (fcb, home, allianz), (allianz, poi, stadium), (allianz, location, munich, ...)$\}$

RDF: Example



$\{$(fcb, name, 'FC Bayern München'), (fcb, league, 'Bundesliga'), (fcb, home, allianz), (allianz, poi, stadium), (allianz, location, munich, ...)$\}$

Remark. Use of algabraic-style notation from [Pérez, Arenas, Gutierrez 2009].

Querying RDF with triple patterns

- Triple patterns: RDF triple + variables

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

Querying RDF with triple patterns

- Triple patterns: RDF triple + variables

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

- The base case for RDF queries is a set of triple patterns

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

This is called basic graph pattern (BGP).

Querying RDF with triple patterns

- Triple patterns: RDF triple + variables

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

- The base case for RDF queries is a set of triple patterns

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

This is called basic graph pattern (BGP).

Example

$$\{\ (?X, \text{name}, ?N),\ \ (?X, \text{home}, ?S)\ \}$$

The semantics of triple patterns

Definition

The evaluation of a triple pattern t over an RDF graph G yields the set of mappings M such that $\mu \in M$ if

- μ has as domain the variables in t: $dom(\mu) = \text{var}(t)$
- μ makes t to match the graph: $\mu(t) \in G$

The semantics of triple patterns

Definition

The evaluation of a triple pattern t over an RDF graph G yields the set of mappings M such that $\mu \in M$ if

- μ has as domain the variables in t: $dom(\mu) = \text{var}(t)$
- μ makes t to match the graph: $\mu(t) \in G$

Example

triple pattern: $(?X, \text{location}, ?Y)$

graph

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

evaluation

?X	?Y

The semantics of triple patterns

Definition

The evaluation of a triple pattern t over an RDF graph G yields the set of mappings M such that $\mu \in M$ if

- μ has as domain the variables in t: $dom(\mu) = \text{var}(t)$
- μ makes t to match the graph: $\mu(t) \in G$

Example

triple pattern: $(?X, \text{location}, ?Y)$

graph

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

evaluation

	?X	?Y
μ_1:	allianz	munich
μ_2:	happel	vienna

SPARQL: "SPARQL Protocol and RDF Query Language"

- W3C recommendation for querying RDF:
  - SPARQL 1.0: January 2008
  - SPARQL 1.1: March 2013
- A SPARQL query consists of three parts:
  - Pattern matching: AND, OPTIONAL, UNION, FILTER, ...
  - Solution modifiers: projection, distinct, order, limit, offset,...
  - Output part: construction of new triples, boolean queries, ...

SPARQL: "SPARQL Protocol and RDF Query Language"

- W3C recommendation for querying RDF:
  - SPARQL 1.0: January 2008
  - SPARQL 1.1: March 2013
- A SPARQL query consists of three parts:
  - Pattern matching: AND, OPTIONAL, UNION, FILTER, ...
  - Solution modifiers: projection, distinct, order, limit, offset,...
  - Output part: construction of new triples, boolean queries, ...

SPARQL: "SPARQL Protocol and RDF Query Language"

- W3C recommendation for querying RDF:
  - SPARQL 1.0: January 2008
  - SPARQL 1.1: March 2013
- A SPARQL query consists of three parts:
  - Pattern matching: AND, OPTIONAL, UNION, FILTER, ...
  - Solution modifiers: projection, distinct, order, limit, offset,...
  - Output part: construction of new triples, boolean queries, ...

SPARQL Query

Example

```
SELECT ?Stadium ?League
WHERE
{
  { ?Club      ex:home     ?Stadium .
    ?Stadium   ex:poi      ex:stadium } OPTIONAL
  { ?Club      ex:league   ?League }
}
```


SPARQL Query

Example

```
SELECT ?Stadium ?League
WHERE
{
  { ?Club      ex:home     ?Stadium .
    ?Stadium   ex:poi      ex:stadium } OPTIONAL
  { ?Club      ex:league   ?League }
}
```

A SPARQL query consists of a:

Body: Pattern matching expression

SPARQL Query

Example

```
SELECT ?Stadium ?League
WHERE
{
  { ?Club      ex:home     ?Stadium .
    ?Stadium   ex:poi      ex:stadium } OPTIONAL
  { ?Club      ex:league   ?League }
}
```

A SPARQL query consists of a:

Body: Pattern matching expression

Head: Processing of the variable assignments

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y) \text{ OPT } (?X, \text{poi}, ?E))$

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y) \text{ OPT } (?X, \text{poi}, ?E))$

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y) \text{ OPT } (?X, \text{poi}, ?E))$

result:

$?X$	$?Y$
allianz	munich
happel	vienna

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y) \text{ OPT } (?X, \text{poi}, ?E))$

result:

$?X$	$?Y$
allianz	munich
happel	vienna

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y)$ OPT $(?X, \text{poi}, ?E))$

result:

?X	?Y
allianz	munich
happel	vienna

?X	?E
allianz	stadium

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y)$ OPT $(?X, \text{poi}, ?E))$

result:

?X	?Y
allianz	munich
happel	vienna

?X	?E
allianz	stadium

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y)$ OPT $(?X, \text{poi}, ?E))$

result:

?X	?Y
allianz	munich
happel	vienna

?X	?E
allianz	stadium

?X	?Y	?E

SPARQL graph pattern

Example

RDF graph:

(allianz, location, munich)
(allianz, poi, stadium)
(happel, location, vienna)

SPARQL graph pattern:

$((?X, \text{location}, ?Y)$ OPT $(?X, \text{poi}, ?E))$

result:

?X	?Y
allianz	munich
happel	vienna

?X	?E
allianz	stadium

?X	?Y	?E
allianz	munich	stadium
happel	vienna	

Complexity of SPARQL

Definition

The EVALUATION problem for SPARQL graph patterns:
INPUT: An RDF graph G, a graph pattern P and a mapping μ.
QUESTION: Is $\mu \in [\![P]\!]_G$?

Complexity of SPARQL

Definition

The EVALUATION problem for SPARQL graph patterns:
INPUT: An RDF graph G, a graph pattern P and a mapping μ.
QUESTION: Is $\mu \in [\![P]\!]_G$?

Theorem (Pérez, Arenas, Gutierrez 2009)

EVALUATION is PSPACE*-complete.*

Complexity of SPARQL

Definition

The EVALUATION problem for SPARQL graph patterns:
INPUT: An RDF graph G, a graph pattern P and a mapping μ.
QUESTION: Is $\mu \in [\![P]\!]_G$?

Theorem (Pérez, Arenas, Gutierrez 2009)

EVALUATION is PSPACE*-complete.*

Theorem (Schmidt, Meier, Lausen 2010)

EVALUATION remains PSPACE*-complete if P contains operators AND and OPT only.*

Observation

- High complexity due to unrestricted occurrences of OPT-operator.
- Arbitrary use of OPT-operator is highly unintuitive.

Observation

- High complexity due to unrestricted occurrences of OPT-operator.
- Arbitrary use of OPT-operator is highly unintuitive.

Example

RDF graph:

(fcb, home, allianz),
(fcb, league, ‘Bundesliga’),
(happel, location, vienna)

SPARQL graph pattern:

$(?C, \text{home}, ?S)\ \text{OPT}\ ((?C, \text{league}, ?L)\ \text{OPT}\ (?S, \text{location}, ?P))$

result:

$\{\{?C \to \text{fcb}, ?S \to \text{allianz}\}\}$

Observation

- High complexity due to unrestricted occurrences of OPT-operator.
- Arbitrary use of OPT-operator is highly unintuitive.

Example

RDF graph:

(fcb, home, allianz),
(fcb, league, ‘Bundesliga’),
(happel, location, vienna)

SPARQL graph pattern:

$(?C, \text{home}, ?S)\ \text{OPT}\ ((?C, \text{league}, ?L)\ \text{OPT}\ (?S, \text{location}, ?P))$

result:

$\{\{?C \to \text{fcb}, ?S \to \text{allianz}\}\}$

2. Introduction to RDF and SPARQL (14 / 47)

Observation

- High complexity due to unrestricted occurrences of OPT-operator.
- Arbitrary use of OPT-operator is highly unintuitive.

Example

RDF graph:

(fcb, home, allianz),
(fcb, league, ‘Bundesliga’),
(happel, location, vienna)

SPARQL graph pattern:

$(?C, \text{home}, ?S)\ \text{OPT}\ ((?C, \text{league}, ?L)\ \text{OPT}\ (?S, \text{location}, ?P))$

result:

$\{\{?C \to \text{fcb}, ?S \to \text{allianz}\}\}$

2. Introduction to RDF and SPARQL (14 / 47)

Observation

- High complexity due to unrestricted occurrences of OPT-operator.
- Arbitrary use of OPT-operator is highly unintuitive.

Example

RDF graph:

(fcb, home, allianz),
(fcb, league, ‘Bundesliga’),
(happel, location, vienna)

SPARQL graph pattern:

$(?C, \text{home}, ?S)\ \text{OPT}\ ((?C, \text{league}, ?L)\ \text{OPT}\ (?S, \text{location}, ?P))$

result:

$\{\{?C \to \text{fcb}, ?S \to \text{allianz}\}\}$

2. Introduction to RDF and SPARQL (14 / 47)

Well-designed graph patterns

Definition

A query in the AND-OPT-FILTER fragment of SPARQL is well-designed if for every OPT in the pattern:

$$(\cdots\cdots\cdots\cdots \quad (\quad P \quad \text{OPT} \quad Q \quad) \quad \cdots\cdots\cdots\cdots)$$

if a variable occurs

Well-designed graph patterns

Definition

A query in the AND-OPT-FILTER fragment of SPARQL is well-designed if for every OPT in the pattern:

$$(\cdots\cdots\cdots\cdots \quad (\quad P \quad \text{OPT} \quad \underset{\uparrow}{Q} \quad) \quad \cdots\cdots\cdots\cdots)$$

if a variable occurs inside Q

Well-designed graph patterns

Definition

A query in the AND-OPT-FILTER fragment of SPARQL is well-designed if for every OPT in the pattern:

$$(\underset{\uparrow}{\cdots\cdots\cdots\cdots} \quad (\quad P \quad \text{OPT} \quad \underset{\uparrow}{Q} \quad) \quad \underset{\uparrow}{\cdots\cdots\cdots\cdots})$$

if a variable occurs inside Q and anywhere outside the OPT expression,

Well-designed graph patterns

Definition

A query in the AND-OPT-FILTER fragment of SPARQL is well-designed if for every OPT in the pattern:

$$(\underset{\uparrow}{\cdots\cdots\cdots\cdots} \quad (\quad \underset{\uparrow}{P} \quad \text{OPT} \quad \underset{\uparrow}{Q} \quad) \quad \underset{\uparrow}{\cdots\cdots\cdots\cdots})$$

if a variable occurs inside Q and anywhere outside the OPT expression, then the variable must also occur inside P.

Well-designed graph patterns

Theorem (Pérez, Arenas, Gutierrez 2009)

EVALUATION is coNP-complete for well-designed graph pattern expressions constructed by using only AND and OPT operators.

Theorem (Letelier, Pérez, P., Skritek 2012)

EVALUATION is Σ_2P–complete for well-designed graph pattern expressions constructed by using only AND and OPT operators and allowing projection.

Summary so far

- Syntax of RDF
  - RDF triples (s, p, o)
  - underlying data model: labelled, directed graph
- Basic querying of RDF
  - RDF triple patterns
  - Basic graph patterns (BGPs)
  - Result of evaluation: set of mappings
- SPARQL
  - SPARQL graph patterns
  - principal operators: AND and OPT
  - complexity of SPARQL
  - well-designed SPARQL

Roadmap

OPT normal form [Pérez, Arenas, Gutierrez 2009]

OPT-normal form: No OPT occurs in the scope of an AND

OPT normal form [Pérez, Arenas, Gutierrez 2009]

OPT-normal form: No OPT occurs in the scope of an AND

- Every well-designed graph pattern can be transformed – in polynomial time – into an equivalent one in OPT-normal form

OPT normal form [Pérez, Arenas, Gutierrez 2009]

OPT-normal form: No OPT occurs in the scope of an AND

- Every well-designed graph pattern can be transformed – in polynomial time – into an equivalent one in OPT-normal form

$$(P_1 \text{ AND } (P_2 \text{ OPT } P_3)) \equiv ((P_1 \text{ AND } P_2) \text{ OPT } P_3)$$

for well-designed graph patterns

Pattern trees: representing well-designed graph patterns

Represent patterns in OPT normal form by trees:

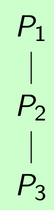
- nodes represent basic graph patterns (BGPs, CQs)
- tree structure represents optionality

Pattern trees: representing well-designed graph patterns

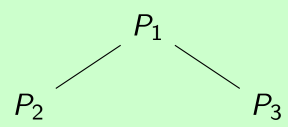
Represent patterns in OPT normal form by trees:

- nodes represent basic graph patterns (BGPs, CQs)
- tree structure represents optionality

Intuition



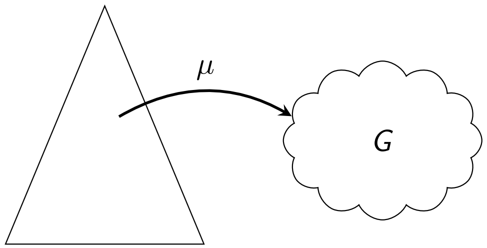
$$\Big(P_1 \text{ OPT } (P_2 \text{ OPT } P_3)\Big)$$



$$\Big((P_1 \text{ OPT } P_2) \text{ OPT } P_3\Big) \equiv \Big((P_1 \text{ OPT } P_3) \text{ OPT } P_2\Big)$$

Pattern trees: evaluation of well-designed SPARQL

Mapping $\mu \in [\![\mathcal{T}]\!]_G$ if and only if

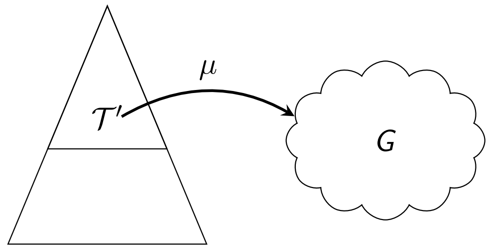


Pattern trees: evaluation of well-designed SPARQL

Mapping $\mu \in [\![\mathcal{T}]\!]_G$ if and only if

1. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$ containing the root s.t.

   $dom(\mu) = vars(\mathcal{T}')$ and $\mu(\mathcal{T}') \subseteq G$

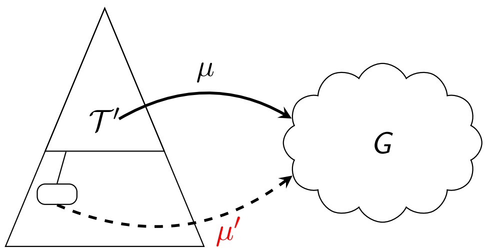


Pattern trees: evaluation of well-designed SPARQL

Mapping $\mu \in [\![\mathcal{T}]\!]_G$ if and only if

1. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$ containing the root s.t.

   $dom(\mu) = vars(\mathcal{T}')$ and $\mu(\mathcal{T}') \subseteq G$

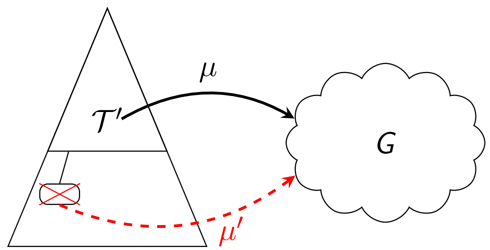


Pattern trees: evaluation of well-designed SPARQL

Mapping $\mu \in [\![\mathcal{T}]\!]_G$ if and only if

1. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$ containing the root s.t.

   $dom(\mu) = vars(\mathcal{T}')$ and $\mu(\mathcal{T}') \subseteq G$

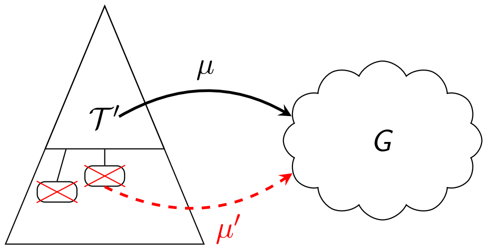


Pattern trees: evaluation of well-designed SPARQL

Mapping $\mu \in [\![\mathcal{T}]\!]_G$ if and only if

1. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$ containing the root s.t.

   $dom(\mu) = vars(\mathcal{T}')$ and $\mu(\mathcal{T}') \subseteq G$

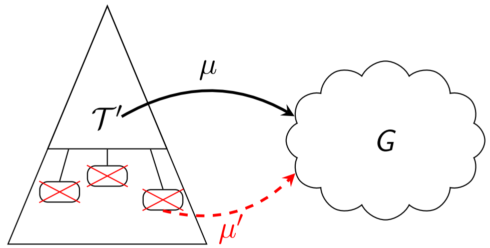


Pattern trees: evaluation of well-designed SPARQL

Mapping $\mu \in [\![\mathcal{T}]\!]_G$ if and only if

1. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$ containing the root s.t.

   $dom(\mu) = vars(\mathcal{T}')$ and $\mu(\mathcal{T}') \subseteq G$



Pattern trees: evaluation of well-designed SPARQL

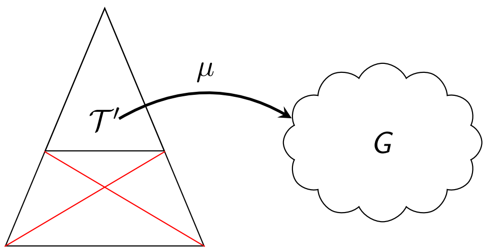
Mapping $\mu \in [\![\mathcal{T}]\!]_G$ if and only if

1. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$ containing the root s.t.

   $dom(\mu) = vars(\mathcal{T}')$ and $\mu(\mathcal{T}') \subseteq G$

2. there do not exist extensions μ' of μ and $\hat{\mathcal{T}}'$ of $\mathcal{T}'$ s.t.

   $dom(\mu') = vars(\hat{\mathcal{T}}')$ and $\mu'(\hat{\mathcal{T}}') \subseteq G$



Summary so far

- Introduction to RDF and SPARQL
- well-designed SPARQL
  - OPT-normal form
  - pattern trees
  - characterization of query evaluation

Roadmap

SPARQL: Subsumption

Definition

Let P_1 and P_2 be two SPARQL graph patterns. We say that P_1 is subsumed by P_2 ($P_1 \sqsubseteq P_2$) if, for every RDF graph G, every solution of P_1 over G can be extended to a solution of P_2 over G.

SPARQL: Subsumption

Definition

Let P_1 and P_2 be two SPARQL graph patterns. We say that P_1 is subsumed by P_2 ($P_1 \sqsubseteq P_2$) if, for every RDF graph G, every solution of P_1 over G can be extended to a solution of P_2 over G.

For SPARQL, subsumption is often more natural than containment.

SPARQL: Subsumption

Definition

Let P_1 and P_2 be two SPARQL graph patterns. We say that P_1 is subsumed by P_2 ($P_1 \sqsubseteq P_2$) if, for every RDF graph G, every solution of P_1 over G can be extended to a solution of P_2 over G.

For SPARQL, subsumption is often more natural than containment.

Example

- $G = \{(fcb, n, 'bayern'), (fcb, h, allianz)\}$
- $P_1 = (?C, n, ?N)$
- $P_2 = (?C, n, ?N)$ OPT $(?C, h, ?H)$

SPARQL: Subsumption

Definition

Let P_1 and P_2 be two SPARQL graph patterns. We say that P_1 is subsumed by P_2 ($P_1 \sqsubseteq P_2$) if, for every RDF graph G, every solution of P_1 over G can be extended to a solution of P_2 over G.

For SPARQL, subsumption is often more natural than containment.

Example

- $G = \{(fcb, n, \text{`}bayern'), (fcb, h, allianz)\}$
- $P_1 = (?C, n, ?N)$
- $P_2 = (?C, n, ?N)$ OPT $(?C, h, ?H)$
- $[\![P_1]\!]_G = \{\mu\}$ with $\mu = \{?C \to fcb, ?N \to \text{`}bayern'\}$,
- $[\![P_2]\!]_G = \{\mu'\}$ with $\mu' = \{?C \to fcb, ?N \to \text{`}bayern', ?H \to allianz\}$

SPARQL: Subsumption

Definition

Let P_1 and P_2 be two SPARQL graph patterns. We say that P_1 is subsumed by P_2 ($P_1 \sqsubseteq P_2$) if, for every RDF graph G, every solution of P_1 over G can be extended to a solution of P_2 over G.

For SPARQL, subsumption is often more natural than containment.

Example

- $G = \{(fcb, n, \text{`}bayern'), (fcb, h, allianz)\}$
- $P_1 = (?C, n, ?N)$
- $P_2 = (?C, n, ?N)$ OPT $(?C, h, ?H)$
- $[\![P_1]\!]_G = \{\mu\}$ with $\mu = \{?C \to fcb, ?N \to \text{`}bayern'\}$,
- $[\![P_2]\!]_G = \{\mu'\}$ with $\mu' = \{?C \to fcb, ?N \to \text{`}bayern', ?H \to allianz\}$

We have $P_1 \sqsubseteq P_2$ but $P_1 \not\subseteq P_2$.

Complexity

Language fragments considered

- well-designed SPARQL (AND, OPT): wd-SPARQL[$\emptyset$]
- extension by UNION: wd-SPARQL[$\{\cup\}$]
- extension by projection: wd-SPARQL[$\{\pi\}$] (i.e., via an appropriate SELECT clause)
- extension by both: wd-SPARQL[$\{\cup, \pi\}$]

Decision problems: for $S_1, S_2 \subseteq \{\cup, \pi\}$

CONTAINMENT[S_1,S_2]

INPUT: $Q_1 \in$ wd-SPARQL[S_1], $Q_2 \in$ wd-SPARQL[S_2]
QUESTION: Does $Q_1 \subseteq Q_2$ hold?

Decision problems: for $S_1, S_2 \subseteq \{\cup, \pi\}$

CONTAINMENT$[S_1, S_2]$

INPUT: $Q_1 \in$ wd-SPARQL$[S_1]$, $Q_2 \in$ wd-SPARQL$[S_2]$
QUESTION: Does $Q_1 \subseteq Q_2$ hold?

EQUIVALENCE$[S_1, S_2]$

INPUT: $Q_1 \in$ wd-SPARQL$[S_1]$, $Q_2 \in$ wd-SPARQL$[S_2]$
QUESTION: Does $Q_1 \equiv Q_2$ hold?

Decision problems: for $S_1, S_2 \subseteq \{\cup, \pi\}$

CONTAINMENT$[S_1, S_2]$

INPUT: $Q_1 \in$ wd-SPARQL$[S_1]$, $Q_2 \in$ wd-SPARQL$[S_2]$
QUESTION: Does $Q_1 \subseteq Q_2$ hold?

EQUIVALENCE$[S_1, S_2]$

INPUT: $Q_1 \in$ wd-SPARQL$[S_1]$, $Q_2 \in$ wd-SPARQL$[S_2]$
QUESTION: Does $Q_1 \equiv Q_2$ hold?

SUBSUMPTION$[S_1, S_2]$

INPUT: $Q_1 \in$ wd-SPARQL$[S_1]$, $Q_2 \in$ wd-SPARQL$[S_2]$
QUESTION: Does $Q_1 \sqsubseteq Q_2$ hold?

Complexity [P., Skritek 2014]

CONTAINMENT$[S_1, S_2]$

$\downarrow S_1$ / $S_2 \rightarrow$	$\emptyset$	$\{\cup\}$	$\{\pi\}$	$\{\cup, \pi\}$
$\emptyset$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup, \pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.

Complexity [P., Skritek 2014]

CONTAINMENT$[S_1, S_2]$

$\downarrow S_1$ / $S_2 \rightarrow$	$\emptyset$	$\{\cup\}$	$\{\pi\}$	$\{\cup, \pi\}$
$\emptyset$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup, \pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.

EQUIVALENCE$[S_1, S_2]$

$\downarrow S_1$ / $S_2 \rightarrow$	$\emptyset$	$\{\cup\}$	$\{\pi\}$	$\{\cup, \pi\}$
$\emptyset$	NP-compl.*	–	–	–
$\{\cup\}$	Π_2P-compl.	Π_2P-compl.	–	–
$\{\pi\}$	Π_2P-compl.	Π_2P-hard	undec.	–
$\{\cup, \pi\}$	Π_2P-compl.	undec.	undec.	undec.

*[Letelier, Pérez, P., Skritek 2012]

Complexity of subsumption

Theorem (Letelier, Pérez, P., Skritek 2012)

SUBSUMPTION$[S_1, S_2]$ *is* Π_2P*-hard even for* $S_1 = S_2 = \emptyset$.

Complexity of subsumption

Theorem (Letelier, Pérez, P., Skritek 2012)

SUBSUMPTION$[S_1, S_2]$ *is* Π_2P*-hard even for* $S_1 = S_2 = \emptyset$.

Theorem (P., Skritek 2014)

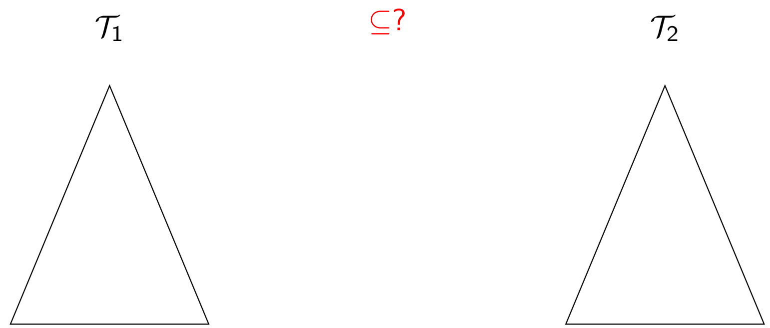
SUBSUMPTION$[S_1, S_2]$ *is in* Π_2P *even for* $S_1 = S_2 = \{\cup, \pi\}$.

Overview of Results

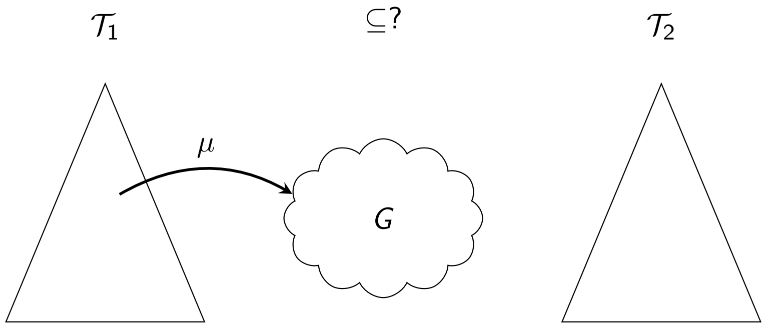
CONTAINMENT$[S_1, S_2]$

$\downarrow S_1$ / $S_2 \rightarrow$	$\emptyset$	$\{\cup\}$	$\{\pi\}$	$\{\cup, \pi\}$
$\emptyset$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup, \pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.

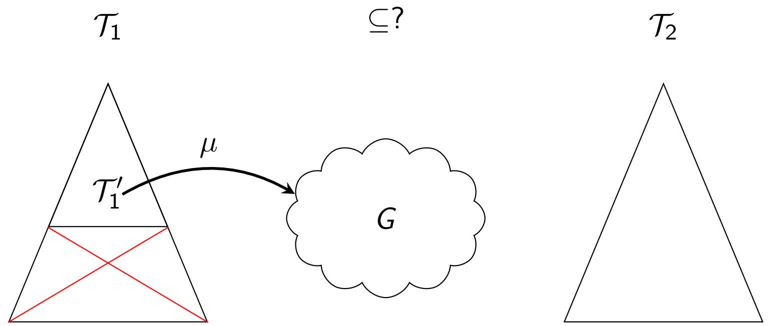
Decidable Containment



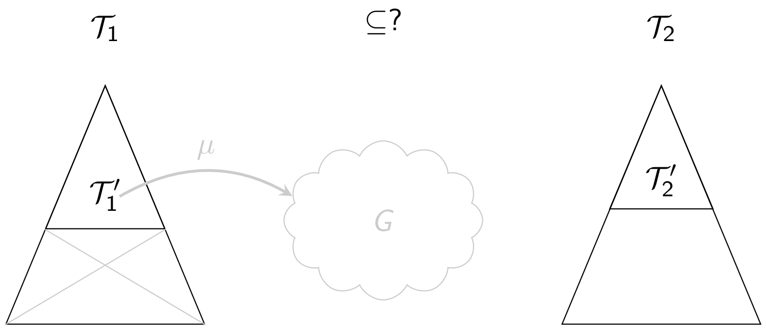
Decidable Containment



Decidable Containment

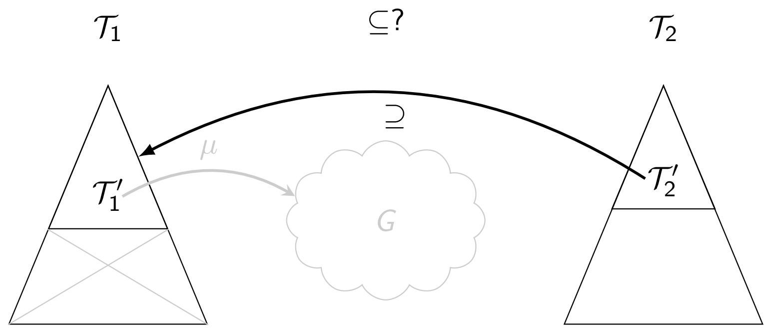


Decidable Containment



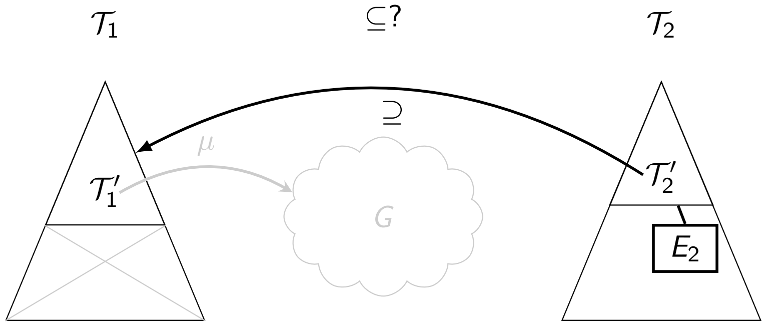
- For every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$ determine subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$
  - $vars(\mathcal{T}_2') = vars(\mathcal{T}_1')$,

Decidable Containment



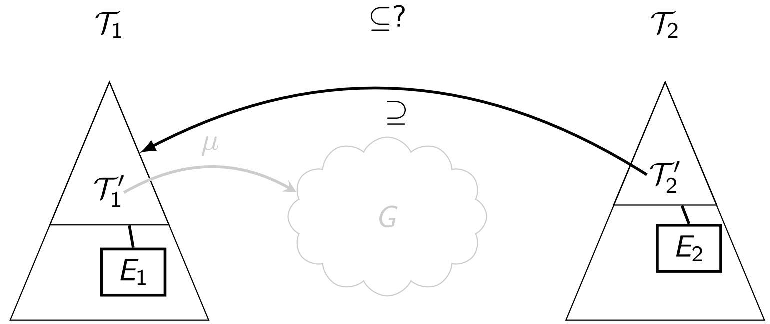
- For every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$ determine subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$
  - $vars(\mathcal{T}_2') = vars(\mathcal{T}_1')$, $triples(\mathcal{T}_2') \subseteq triples(\mathcal{T}_1')$

Decidable Containment



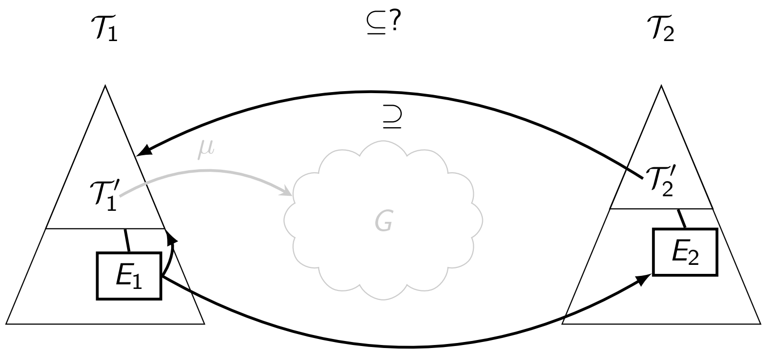
- For every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$ determine subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$
  - $vars(\mathcal{T}_2') = vars(\mathcal{T}_1')$, $triples(\mathcal{T}_2') \subseteq triples(\mathcal{T}_1')$
- For every extension E_2 of $\mathcal{T}_2'$,

Decidable Containment



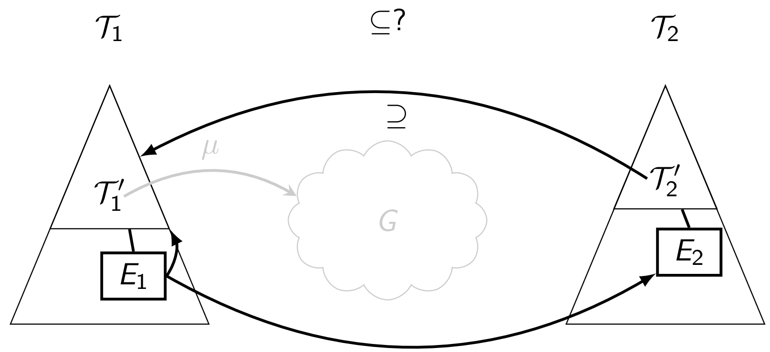
- For every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$ determine subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$
  - $vars(\mathcal{T}_2') = vars(\mathcal{T}_1')$, $triples(\mathcal{T}_2') \subseteq triples(\mathcal{T}_1')$
- For every extension E_2 of $\mathcal{T}_2'$,
- Exists extension E_1 of $\mathcal{T}_1'$

Decidable Containment



- For every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$ determine subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$
  - $vars(\mathcal{T}_2') = vars(\mathcal{T}_1')$, $triples(\mathcal{T}_2') \subseteq triples(\mathcal{T}_1')$
- For every extension E_2 of $\mathcal{T}_2'$,
- Exists extension E_1 of $\mathcal{T}_1'$ and exists homomorphism h
  - $h \colon E_1 \to \mathcal{T}_1' \cup E_2$, h is identity on $vars(\mathcal{T}_1')$

Decidable Containment



- For every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$ determine subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$
  - $vars(\mathcal{T}_2') = vars(\mathcal{T}_1')$, $triples(\mathcal{T}_2') \subseteq triples(\mathcal{T}_1')$
- For every extension E_2 of $\mathcal{T}_2'$,
- Exists extension E_1 of $\mathcal{T}_1'$ and exists homomorphism h
  - $h \colon E_1 \to \mathcal{T}_1' \cup E_2$, h is identity on $vars(\mathcal{T}_1')$

NP-completeness

Theorem (P., Skritek 2014)

Let $S_1 \subseteq \{\cup, \pi\}$ and $S_2 = \emptyset$.
Then the CONTAINMENT$[S_1, S_2]$ *problem is in NP.*

NP-completeness

Theorem (P., Skritek 2014)

Let $S_1 \subseteq \{\cup, \pi\}$ and $S_2 = \emptyset$.
Then the CONTAINMENT$[S_1, S_2]$ *problem is in NP.*

Proof idea.

- apply the algorithm sketched above
- show that it suffices to test polynomially many subtrees of $\mathcal{T}_1$

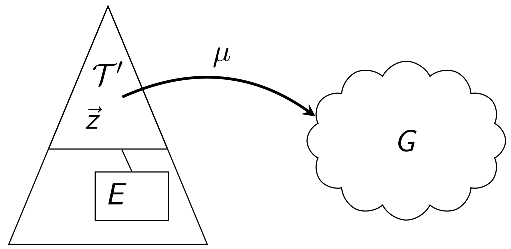
Overview of Results:

CONTAINMENT$[S_1, S_2]$

$\downarrow S_1$ / $S_2 \rightarrow$	$\emptyset$	$\{\cup\}$	$\{\pi\}$	$\{\cup, \pi\}$
$\emptyset$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.
$\{\cup, \pi\}$	NP-compl.	Π_2P-compl.	undec.	undec.

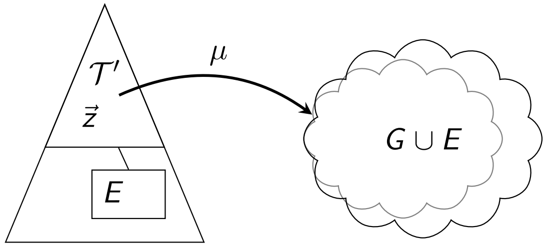
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL$[\{\pi\}]$:



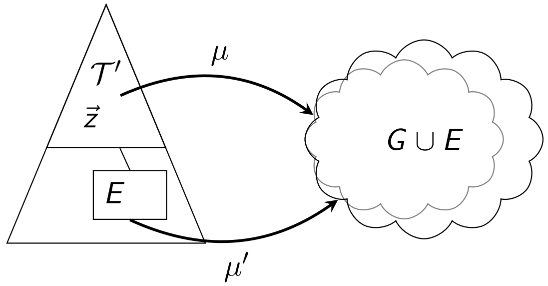
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL[$\{\pi\}$]:



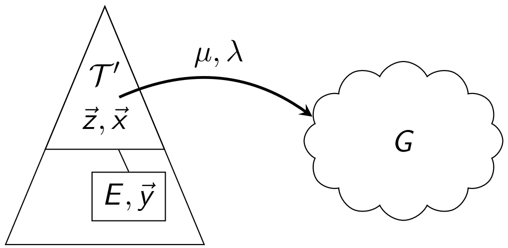
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL[$\{\pi\}$]:



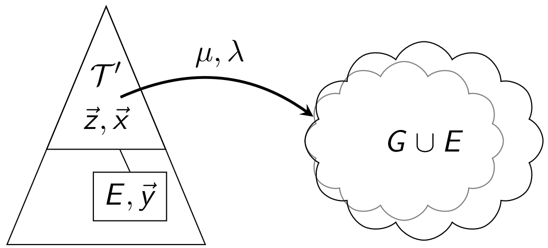
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL[$\{\pi\}$]:



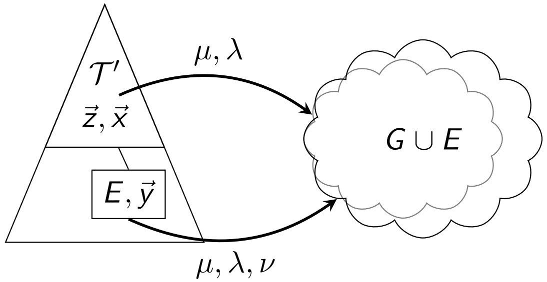
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL[$\{\pi\}$]:



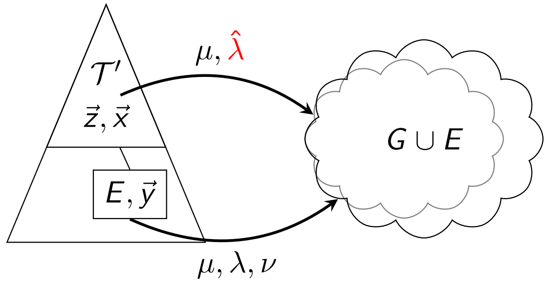
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL$[\{\pi\}]$:



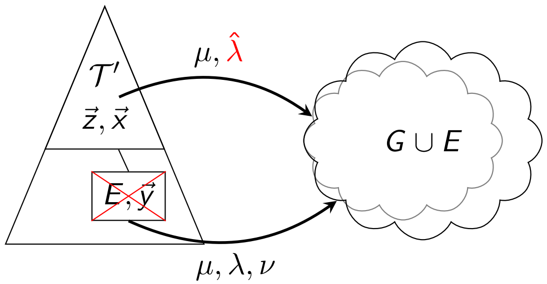
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL$[\{\pi\}]$:



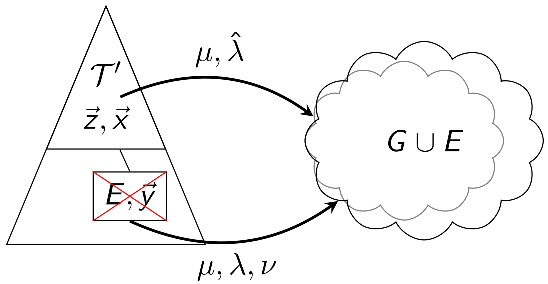
Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL$[\{\pi\}]$:



Undecidable Containment

Evaluation of $\mathcal{T} \in$ wd-SPARQL$[\{\pi\}]$:



similar to TGDs: $\forall \vec{x} \varphi(\mu(\vec{z}), \vec{x}) \rightarrow \exists \vec{y} \psi(\mu(\vec{z}), \vec{x}, \vec{y})$

Undecidability

BCQ-UNDER-TGDS

INPUT: Instance I, BCQ $\exists\vec{q}\, Q(\vec{q})$,
tgd $\tau\colon \forall\vec{x}(\varphi(\vec{x}) \rightarrow \exists\vec{y}\psi(\vec{x},\vec{y}))$

QUESTION: Does $I, \tau \models \exists\vec{q}\, Q(\vec{q})$ hold?

Undecidability

BCQ-UNDER-TGDS

INPUT: Instance I, BCQ $\exists\vec{q}\, Q(\vec{q})$,
tgd $\tau\colon \forall\vec{x}(\varphi(\vec{x}) \rightarrow \exists\vec{y}\psi(\vec{x},\vec{y}))$

QUESTION: Does $I, \tau \models \exists\vec{q}\, Q(\vec{q})$ hold?

Theorem (P., Skritek 2014)

Let $S_1 \subseteq \{\cup, \pi\}$ *and* $\pi \in S_2$.
Then the CONTAINMENT$[S_1,S_2]$ *problem is undecidable.*

Undecidability

BCQ-UNDER-TGDS

INPUT: Instance I, BCQ $\exists\vec{q}\, Q(\vec{q})$,
tgd $\tau\colon \forall\vec{x}(\varphi(\vec{x}) \rightarrow \exists\vec{y}\psi(\vec{x},\vec{y}))$

QUESTION: Does $I, \tau \models \exists\vec{q}\, Q(\vec{q})$ hold?

Theorem (P., Skritek 2014)

Let $S_1 \subseteq \{\cup, \pi\}$ *and* $\pi \in S_2$.
Then the CONTAINMENT$[S_1,S_2]$ *problem is undecidable.*

Proof idea.

by reduction from BCQ-UNDER-TGDS

Roadmap

1. Motivation
2. Introduction to RDF and SPARQL
3. SPARQL Pattern Trees
4. Static Analysis of SPARQL Queries
5. Certain answers for OWL2 Entailment
6. Conclusion and Future Work

Motivation

OBDA with SPARQL

- W3C defines various entailment regimes for SPARQL
  - examples of entailment regimes: RDF simple entailment, RDFS, OWL2 profiles (like OWL2-QL), etc.
  - W3C standard defines evaluation of BGPs
  - resulting semantics may not be intuitive
- Challenge with SPARQL
  - non-monotonicity of the OPT operator
  - adapt certain answer semantics to wd-SPARQL
  - adapt query rewriting to wd-SPARQL

Example 1: existentially quantified variables

Example

Consider RDF graph $G = \{(marc, \text{a}, Prof)\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

Example 1: existentially quantified variables

Example

Consider RDF graph $G = \{(marc, \text{a}, Prof)\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:
Q1: $SELECT * WHERE\ (?x, teaches, ?y)$
Q2: $SELECT\ ?x\ WHERE\ (?x, teaches, ?y)$
Q3: $SELECT * WHERE\ (?x, teaches, \_b)$

Example 1: existentially quantified variables

Example

Consider RDF graph $G = \{(marc, \text{a}, Prof)\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:
Q1: $SELECT * WHERE\ (?x, teaches, ?y)$
Q2: $SELECT\ ?x\ WHERE\ (?x, teaches, ?y)$
Q3: $SELECT * WHERE\ (?x, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):
Q1 returns the emtpy set.

Example 1: existentially quantified variables

Example

Consider RDF graph $G = \{(marc, \text{a}, Prof)\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ *\ WHERE\ (?x, teaches, ?y)$

Q2: $SELECT\ ?x\ WHERE\ (?x, teaches, ?y)$

Q3: $SELECT\ *\ WHERE\ (?x, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):

Q1 returns the emtpy set.

Q2 returns the emtpy set.

Example 1: existentially quantified variables

Example

Consider RDF graph $G = \{(marc, \text{a}, Prof)\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ *\ WHERE\ (?x, teaches, ?y)$

Q2: $SELECT\ ?x\ WHERE\ (?x, teaches, ?y)$

Q3: $SELECT\ *\ WHERE\ (?x, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):

Q1 returns the emtpy set.

Q2 returns the emtpy set.

Q3 returns $M = \{\mu\}$ with $\mu = \{?x \rightarrow marc\}$

Example 1: existentially quantified variables

Example

Consider RDF graph $G = \{(marc, \text{a}, Prof)\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ *\ WHERE\ (?x, teaches, ?y)$

Q2: $SELECT\ ?x\ WHERE\ (?x, teaches, ?y)$

Q3: $SELECT\ *\ WHERE\ (?x, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):

Q1 returns the emtpy set.

Q2 returns the emtpy set.

Q3 returns $M = \{\mu\}$ with $\mu = \{?x \rightarrow marc\}$

Remark. blank nodes are “immediately” projected out
$\Rightarrow$ binding to labelled nulls in the canonical model is allowed.

Example 2: join across the OPT operator

Example

Again, consider RDF graph $G = \{(\text{marc,a,Prof})\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

Example 2: join across the OPT operator

Example

Again, consider RDF graph $G = \{(\text{marc,a,Prof})\}$ with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, ?y)$ OPT $(?z, teaches, ?y)$

Q2: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Example 2: join across the OPT operator

Example

Again, consider RDF graph $G = \{(\text{marc,a,Prof})\}$ with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, ?y)$ OPT $(?z, teaches, ?y)$

Q2: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):

Q1 returns the emtpy set.

Example 2: join across the OPT operator

Example

Again, consider RDF graph $G = \{(\text{marc,a,Prof})\}$ with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, ?y)$ OPT $(?z, teaches, ?y)$

Q2: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):

Q1 returns the emtpy set.

Q2 returns $M = \{\mu\}$ with $\mu = \{?x \rightarrow marc\}$

Example 2: join across the OPT operator

Example

Again, consider RDF graph $G = \{(\text{marc,a,Prof})\}$ with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, ?y)$ OPT $(?z, teaches, ?y)$

Q2: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):

Q1 returns the emtpy set.

Q2 returns $M = \{\mu\}$ with $\mu = \{?x \rightarrow marc\}$

What about $\mu = \{?x \rightarrow marc, ?z \rightarrow marc\}$?

Example 2: join across the OPT operator

Example

Again, consider RDF graph $G = \{(\text{marc,a,Prof})\}$
with OWL2 ontology $\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$.

SPARQL Queries:

Q1: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, ?y)$ OPT $(?z, teaches, ?y)$

Q2: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Answers (tested using the OWL2 reasoner Pellet):
Q1 returns the emtpy set.
Q2 returns $M = \{\mu\}$ with $\mu = \{?x \rightarrow marc\}$

What about $\mu = \{?x \rightarrow marc, ?z \rightarrow marc\}$?

Remark. SPARQL does not allow the join of blank nodes across BGP boundaries.

Certain answers via subsumption?

Idea: following [Arenas, Pérez 2011]

Certain answers of query Q over RDF graph G w.r.t. ontology $\mathcal{O}$ as greatest lower bound w.r.t. $\sqsubseteq$ over all models of $G \cup \mathcal{O}$.

Certain answers via subsumption?

Idea: following [Arenas, Pérez 2011]

Certain answers of query Q over RDF graph G w.r.t. ontology $\mathcal{O}$ as greatest lower bound w.r.t. $\sqsubseteq$ over all models of $G \cup \mathcal{O}$.

Problem

- greatest lower bound w.r.t. $\sqsubseteq$ is not unique
- subsumed answers in the presence of projection and/or union

Certain answers via subsumption?

Idea: following [Arenas, Pérez 2011]

Certain answers of query Q over RDF graph G w.r.t. ontology $\mathcal{O}$ as greatest lower bound w.r.t. $\sqsubseteq$ over all models of $G \cup \mathcal{O}$.

Problem

- greatest lower bound w.r.t. $\sqsubseteq$ is not unique
- subsumed answers in the presence of projection and/or union

Example

Let $M_1 = \{\mu_1\}$, $M_2 = \{\mu_1, \mu_2\}$, and $M_3 = \{\mu_1, \mu_3\}$ with
$\mu_1 = \{?x \rightarrow a, ?y \rightarrow b, ?z \rightarrow c\}$,
$\mu_2 = \{?x \rightarrow a, ?y \rightarrow b\}$, and
$\mu_3 = \{?x \rightarrow a, ?z \rightarrow c\}$.

Certain answers via subsumption?

Idea: following [Arenas, Pérez 2011]

Certain answers of query Q over RDF graph G w.r.t. ontology $\mathcal{O}$ as greatest lower bound w.r.t. $\sqsubseteq$ over all models of $G \cup \mathcal{O}$.

Problem

- greatest lower bound w.r.t. $\sqsubseteq$ is not unique
- subsumed answers in the presence of projection and/or union

Example

Let $M_1 = \{\mu_1\}$, $M_2 = \{\mu_1, \mu_2\}$, and $M_3 = \{\mu_1, \mu_3\}$ with
$\mu_1 = \{?x \to a, ?y \to b, ?z \to c\}$,
$\mu_2 = \{?x \to a, ?y \to b\}$, and
$\mu_3 = \{?x \to a, ?z \to c\}$.
Then $M_1 \sqsubseteq M_2$, $M_2 \sqsubseteq M_1$, $M_2 \sqsubseteq M_3$, $M_3 \sqsubseteq M_1$, etc.

Certain answers via subsumption?

Example (continued)

Let $M_1 = \{\mu_1\}$ and $M_2 = \{\mu_1, \mu_2\}$ as before.
Suppose that in some possible worlds the set of solutions to a query Q over G w.r.t. $\mathcal{O}$ is M_1 and in some it is M_2.
Which of M_1 and M_2 should be the set of certain answers?

Certain answers via subsumption?

Example (continued)

Let $M_1 = \{\mu_1\}$ and $M_2 = \{\mu_1, \mu_2\}$ as before.
Suppose that in some possible worlds the set of solutions to a query Q over G w.r.t. $\mathcal{O}$ is M_1 and in some it is M_2.
Which of M_1 and M_2 should be the set of certain answers?

Answer: it depends!

Two reasons why in some possible worlds only μ_1 is a solution:
Case 1: μ_2 "gets lost";
Case 2: μ_2 can be extended to μ_1.

Proposed solution

Definition of certain answers

Given SPARQL pattern tree $\mathcal{T}$ (or, equivalently, wd-SPARQL graph pattern), RDF graph G, and ontology $\mathcal{O}$. We define:

$\mu \in \text{CertainAnswers}(\mathcal{T}, G, \mathcal{O})$ if

Proposed solution

Definition of certain answers

Given SPARQL pattern tree $\mathcal{T}$ (or, equivalently, wd-SPARQL graph pattern), RDF graph G, and ontology $\mathcal{O}$. We define:

$\mu \in \mathrm{CertainAnswers}(\mathcal{T}, G, \mathcal{O})$ if

1. for every model G' of $G \cup \mathcal{O}$, it holds that $\{\mu\} \sqsubseteq [\![\mathcal{T}]\!]_{G'}$;

Proposed solution

Definition of certain answers

Given SPARQL pattern tree $\mathcal{T}$ (or, equivalently, wd-SPARQL graph pattern), RDF graph G, and ontology $\mathcal{O}$. We define:

$\mu \in \mathrm{CertainAnswers}(\mathcal{T}, G, \mathcal{O})$ if

1. for every model G' of $G \cup \mathcal{O}$, it holds that $\{\mu\} \sqsubseteq [\![\mathcal{T}]\!]_{G'}$;
2. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$, s.t. $dom(\mu) = fvars(\mathcal{T}')$;

Proposed solution

Definition of certain answers

Given SPARQL pattern tree $\mathcal{T}$ (or, equivalently, wd-SPARQL graph pattern), RDF graph G, and ontology $\mathcal{O}$. We define:

$\mu \in \mathrm{CertainAnswers}(\mathcal{T}, G, \mathcal{O})$ if

1. for every model G' of $G \cup \mathcal{O}$, it holds that $\{\mu\} \sqsubseteq [\![\mathcal{T}]\!]_{G'}$;
2. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$, s.t. $dom(\mu) = fvars(\mathcal{T}')$;
3. μ is maximal with properties 1 and 2.

Computation of the certain answers

Idea: following [Calvanese et al., 2007]

Step 1: show how the certain answers of query $\mathcal{T}$ can be obtained from the canonical model of $G \cup \mathcal{O}$.

Computation of the certain answers

Idea: following [Calvanese et al., 2007]

Step 1: show how the certain answers of query $\mathcal{T}$ can be obtained from the canonical model of $G \cup \mathcal{O}$.

Essentially, this means that μ is a certain answer if the following conditions hold:

1. $\{\mu\} \sqsubseteq [\![\mathcal{T}]\!]_{Can(G \cup \mathcal{O})}$
2. $rg(\mu) \subseteq Voc(G)$
3. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$, s.t. $dom(\mu) = fvars(\mathcal{T}')$
4. μ is maximal with properties $1 - 3$.

Computation of the certain answers

Idea: following [Calvanese et al., 2007]

Step 1: show how the certain answers of query $\mathcal{T}$ can be obtained from the canonical model of $G \cup \mathcal{O}$.

Essentially, this means that μ is a certain answer if the following conditions hold:

1. $\{\mu\} \sqsubseteq [\![\mathcal{T}]\!]_{Can(G \cup \mathcal{O})}$
2. $rg(\mu) \subseteq Voc(G)$
3. there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$, s.t. $dom(\mu) = fvars(\mathcal{T}')$
4. μ is maximal with properties $1 - 3$.

Step 2: Extension of the "Perfect Reformulation" algorithm to allow joins over blank node _b between OPT-operands even if _b is mapped to a labelled null.
Idea: store "path information" via Skolem terms.

Computation of the certain answers

Example

$G = \{(marc, \text{a}, Prof)\}$
$\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$
Q: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Computation of the certain answers

Example

$G = \{(marc, \text{a}, Prof)\}$
$\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$
Q: $SELECT\ ?x, ?z\ WHERE\ (?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Rewriting:
$(?x, teaches, \_b) \Rightarrow (?x, \text{a}, Prof)$ with "path" $\_b \rightarrow f_{teaches}(?x)$,
$(?z, teaches, \_b) \Rightarrow (?z, \text{a}, Prof)$ with "path" $\_b \rightarrow f_{teaches}(?z)$.

Computation of the certain answers

Example

$G = \{(marc, \text{a}, Prof)\}$
$\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$
Q: *SELECT* $?x, ?z$ *WHERE* $(?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Rewriting:
$(?x, teaches, \_b) \Rightarrow (?x, \text{a}, Prof)$ with "path" $\_b \rightarrow f_{teaches}(?x)$,
$(?z, teaches, \_b) \Rightarrow (?z, \text{a}, Prof)$ with "path" $\_b \rightarrow f_{teaches}(?z)$.

Evaluation of the BGPs:
$\mu_1 = \{?x \rightarrow marc \mid \_b \rightarrow f_{teaches}(marc)\}$,
$\mu_2 = \{?z \rightarrow marc \mid \_b \rightarrow f_{teaches}(marc)\}$.

Computation of the certain answers

Example

$G = \{(marc, \text{a}, Prof)\}$
$\mathcal{O} = \{Prof \sqsubseteq \exists teaches\}$
Q: *SELECT* $?x, ?z$ *WHERE* $(?x, teaches, \_b)$ OPT $(?z, teaches, \_b)$

Rewriting:
$(?x, teaches, \_b) \Rightarrow (?x, \text{a}, Prof)$ with "path" $\_b \rightarrow f_{teaches}(?x)$,
$(?z, teaches, \_b) \Rightarrow (?z, \text{a}, Prof)$ with "path" $\_b \rightarrow f_{teaches}(?z)$.

Evaluation of the BGPs:
$\mu_1 = \{?x \rightarrow marc \mid \_b \rightarrow f_{teaches}(marc)\}$,
$\mu_2 = \{?z \rightarrow marc \mid \_b \rightarrow f_{teaches}(marc)\}$.

Evaluation of Q:
$\mu = \{?x \rightarrow marc, ?z \rightarrow marc\}$, because μ_1 and μ_2 are compatible.

Problems Studied

OWL 2 QL-Evaluation

Input: pattern tree $(\mathcal{T}, \mathcal{X})$, graph G, ontology $\mathcal{O}$, mapping μ.
Question: $\mu \in \text{CertainAnswers}((\mathcal{T}, \mathcal{X}), G, \mathcal{O})$?

Problems Studied

OWL 2 QL-Evaluation

Input: pattern tree $(\mathcal{T}, \mathcal{X})$, graph G, ontology $\mathcal{O}$, mapping μ.
Question: $\mu \in \text{CertainAnswers}((\mathcal{T}, \mathcal{X}), G, \mathcal{O})$?

OWL 2 QL-Subsumption/OWL 2 QL-Containment/OWL 2 QL-Equivalence

Input: pattern trees $(\mathcal{T}_1, \mathcal{X}_1)$, $(\mathcal{T}_2, \mathcal{X}_2)$, ontology $\mathcal{O}$.
Question: $(\mathcal{T}_1, \mathcal{X}_1) \sqsubseteq_{\mathcal{O}} / \subseteq_{\mathcal{O}} / \equiv_{\mathcal{O}} (\mathcal{T}_2, \mathcal{X}_2)$?

Complexity

[Ahmetaj, Fischl, P., Simkus, Skritek, 2015]

Theorem

The problem OWL 2 QL-EVALUATION *is DP-complete. Hardness holds even if the ontology is empty.*

Complexity

[Ahmetaj, Fischl, P., Simkus, Skritek, 2015]

Theorem

The problem OWL 2 QL-EVALUATION *is DP-complete. Hardness holds even if the ontology is empty.*

Theorem

The problems OWL 2 QL-SUBSUMPTION, OWL 2 QL-CONTAINMENT, *and* OWL 2 QL-EQUIVALENCE *are* $\Pi_2 P$*-complete. Hardness holds even for empty ontologies.*

Roadmap

Conclusion and Future Work

Conclusion

- OPTIONAL operator is crucial for SPARQL
- non-monotonicity poses various challenges
- query answering, static analysis, entailment regimes

Conclusion and Future Work

Conclusion

- OPTIONAL operator is crucial for SPARQL
- non-monotonicity poses various challenges
- query answering, static analysis, entailment regimes

Future Work

- closing gaps in complexity analyses
- extensions (patterns, ontologies)
- restrictions to achieve tractability