On the primitivity and satisfiability of SPARQL

Xiaowang Zhang

Co-work with Jan Van den Bussche

Hasselt University, Belgium
Navigating plan to DBDBD-14
Modeling in RDF graphs

Map

Triples

(subject, predicate, object)

- (Hasselt, E313, Antwerp)
- (Antwerp, E313, Hasselt)
- (Hasselt, E313, Liège)
- (Liège, E313, Hasselt)
- (Hasselt, E314, Leuven)
- (Leuven, E314, Hasselt)
- (Brussels, E19, Antwerp)
- (Antwerp, E19, Brussels)
- (Brussels, A12, Antwerp)
- (Antwerp, A12, Brussels)
- (Leuven, E40, Brussels)
- (Brussels, E40, Leuven)
- (Liège, E40, Leuven)
- (Leuven, E40, Liège)

DBDBD 2014
SPARQL for querying RDF graphs

- **Triple patterns**: a triple \((u, v, w)\) with variables \(?x, ?y, ?z\)...
- **Pattern operators**: AND, UNION, OPT, SELECT, FILTER

*Which highway connects Hasselt to Antwerp?*

Query: (Hasselt, \(?n\), Antwerp)?

Answer: \(?n = E313\).
SPARQL for querying RDF graphs

- Triple patterns: a triple \((u, v, w)\) with variables \(?x, ?y, ?z\)...
- Pattern operators: \(\text{AND}, \text{UNION}, \text{OPT}, \text{SELECT}, \text{FILTER}\)

**Which city connects to Antwerp through a city?**

**Query:** SELECT \(\{?x\}\) ((?x, ?n, ?y) AND (?y, ?m, Antwerp))

**Answer:**
- \(?x = \text{Liège}\), \(?n = \text{E313}\), \(?m = \text{E313}\), \(?y = \text{Hasselt}\).
- \(?x = \text{Leuven}\), \(?n = \text{E40}\), \(?m = \text{E19}\), \(?y = \text{Brussels}\).
SPARQL for querying RDF graphs

- Triple patterns: a triple \((u, v, w)\) with variables \(?x, ?y, ?z\)...
- Pattern operators: AND, UNION, OPT, SELECT, FILTER

For each highway leaving from Antwerp, give cities reachable by the same highway in two steps. If the highway does not continue, give only the first step.

Query:

\[
\text{SELECT} \{ ?x, ?y \} (\text{(Antwerp, } ?n, ?x) \text{ OPT } (\text{?x, } ?n, ?y)) \text{ FILTER } (\text{?y } \neq \text{Antwerp})
\]

Answer:

- \(?x =\text{Brussels, } ?n = \text{E19.}\)
- \(?x =\text{Brussels, } ?n = \text{A12.}\)
- \(?x =\text{Hasselt, } ?n = \text{E313, } ?y = \text{Liège.}\)
Outline

- Primitivity of SPARQL operators
- Satisfiability of SPARQL fragments
- Summary
Outline

- Primitivity of SPARQL operators
- Satisfiability of SPARQL fragments
- Summary
Primitivity of SPARQL operators

A fragment $W_1$ is *expressible* in a fragment $W_2$ if for any pattern $P$ in $W_1$ there exists some pattern $Q$ in $W_2$ such that $P$ is equivalent to $Q$. 

Result 1
Primitivity of SPARQL operators

\[ AFO \rightarrow FO \]

\[
P \text{ AND } Q = (P \text{ OPT } Q) \text{ MINUS } (P \text{ MINUS } Q)
\]

\[
P \text{ MINUS } Q = (P \text{ OPT } (Q \text{ OPT } (\text{?x, ?y, ?z})) \text{ FILTER } \neg \text{bound(?x)}, \]

where ?x, ?y, ?z are fresh variables.

R. Angles & C. Gutierrez, 2008
Primitivity of SPARQL operators

A fragment $W_1$ is **not expressible** in a fragment $W_2$ if there exists some pattern $P$ in $W_1$ such that for any pattern $Q$ in $W_2$, $P$ is **not** equivalent to $Q$. 

```
W1 → W2
```

Non-trivial results

```
FO       AFO

AO     AF

O  A  F

∅
```

```
W+SELECT     W+UNION

W+     W

W     W
```
Primitivity of SPARQL operators

Consider the pattern $P = (?x, ?y, ?z) \text{AND} (?u, ?v, ?w)$: each mapping contains 6 variables. However, for any pattern $Q$ in FSU, each mapping contains at most 3 variables.

Consider the pattern $P = (?x, p, ?y) \text{AND} (?x, q, ?y)$: not satisfied in any singleton graph. However, for any satisfiable pattern $Q$ in OSU, there must be a singleton graph satisfying it.

Both OPT and FILTER are necessary to express AND.
Primitivity of SPARQL operators

An operator \( X \) is primitive if for any \( X \)-free fragment \( W \), \( W + X \) is not expressible in \( W \).

Result 2

FILTER is primitive.

Consider the pattern \( P = (\dot{x}, \dot{y}, \dot{z}) \) FILTER \( \dot{x} = \dot{y} \) and the graph \( G = \{a, b\} \times \{a, b\} \times \{a, b\} \): each mapping satisfies \( \dot{x} = \dot{y} \). However, for any pattern \( Q \) in AOSU, it always contains all possible mappings from variables in \( Q \) to \( \{a, b\} \).
Primitivity of SPARQL operators

Consider the pattern $P = \text{SELECT}\{?x\} (?x, ?y, ?z)$. If there exists some pattern $Q$ in AOFU such that $P$ is expressible as $Q$. Then all variables in all triple patterns of $Q$ are $?x$. Consider the graph $G = \{(a, p, b)\}$, where $a, p, b$ are fresh constants. $P$ returns a mapping $?x = a$, however, $Q$ returns nothing.
Consider the pattern $P = (\text{x}, r, \text{y}) \cup (\text{y}, r, \text{x})$ and $G = \{(a, r, b)\}$, where there are two mappings to be returned. However, for any pattern $Q$ in AOFS, at most one mapping will be returned.
Primitivity of SPARQL operators

Only AND is not primitive.

(AND, FILTER, OPT, SELECT, UNION)
Primitivity of SPARQL operators

Well-designed patterns defined by Perez, Arenas and Gutierrez, 2008.

\[ P_1 \cup P_2 \cup \ldots \cup P_m \]

where each \( P_i \) is

- UNION-free
- Safe
- For every sub-pattern \( Q \) of the form \( Q_1 \) OPT \( Q_2 \), all variables occurring in both \( Q_2 \) and outside of \( Q \) must occur in \( Q_1 \).

\[ P \ \text{AND} \ Q = (P \ \text{OPT} \ Q) \ \text{MINUS} \ (P \ \text{MINUS} \ Q) \]

\[ P \ \text{MINUS} \ Q = (P \ \text{OPT} \ (Q \ \text{OPT} \ (?x, \ ?y, \ ?z)) \ \text{FILTER} \ \neg \text{bound}(?x). \]

Result 6

AND is primitive for well-designed patterns.

Non-well-designed
Outline

- Primitivity of SPARQL operators
- Satisfiability of SPARQL fragments
- Summary
Satisfiability of SPARQL fragments

A pattern P is **satisfiable** if there exists some graph G where P returns at least one solution.

The satisfiability problem of the full fragment (AFOSU) is **undecidable** because the full fragment has the same expressiveness as relational algebra whose satisfiability problem is well known to be undecidable.

**Fragments via Constraints**

- **SPARQL(bound)**
  - `bound(\(?x\))`
- **SPARQL(¬-bound)**
  - `¬bound(\(?x\))`
- **SPARQL(=c)**
  - `\(?x = c\)`
- **SPARQL(≠c)**
  - `\(?x ≠ c\)`
- **SPARQL(=)**
  - `\(?x = ?y\)`
- **SPARQL(≠)**
  - `\(?x ≠ ?y\)`
Satisfiability of SPARQL fragments

Result 7

The satisfiability problem of:

- $AFOSU(bound, =, \neq_c)$
- $AFOSU(bound, \neq, \neq_c)$

is decidable.

Consistency of constraints

For any set $S$ of constraints of forms:

- $bound(\text{?x}, \text{?x} = \text{?y}, \text{?x} \neq c)$
- $bound(\text{?x}, \text{?x} \neq \text{?y}, \text{?x} \neq c)$

there always exists some mapping that satisfies each constraints of $S$. 
The satisfiability problem of SPARQL fragments

**Result 8**

The satisfiability problem of:
- \( \text{AFOSU}(\neg \text{bound}) \)
- \( \text{AFOSU}(=, \neq) \)
- \( \text{AFOSU}(=, c) \)

is undecidable.

The set difference operator can be expressed in:
- \( \text{AFOSU}(\neg \text{bound}) \)
- \( \text{AFOSU}(=, \neq) \)
- \( \text{AFOSU}(=, c) \)

The satisfiability problem of the downward algebra (DA: union, composition, and difference) is undecidable.  
*T. Tan, J. Van den Bussche, X. Zhang 2014*
The satisfiability problem of well-designed patterns is decidable.

• Every UNION-free well-designed pattern can be reduced to a pattern in AF with preserving satisfiability.
• The satisfiability problem of AF is decidable.
Outline

- Primitivity of SPARQL operators
- Satisfiability of SPARQL fragments
- Summary
Summary

- Presenting the Hasse diagram of expressivity of all fragments with basic SPARQL operators.
- Investigating the satisfiability problem of all fragments with basic SPARQL operators.
- Future work: adding SPARQL 1.1 new operators, such as BIND, VALUE, MINUS etc.
Thanks

xiaowang.zhang@uhasselt.be
References

SPARQL for querying RDF graphs

• Triple patterns: a triple (u, v, w) with variables ?x, ?y, ?z...
• Pattern operators: AND, UNION, OPT, SELECT, FILTER

Constraints and its three-valued semantics (true, false, error)

• bound(?x)  
  • $\mu(\text{bound}(?x)) = \text{true}$ if ?x occurs in $\mu$; $\text{false}$ otherwise

• $\neg$ bound(?x)  
  • $\mu(\neg \text{bound}(?x)) = \text{true}$ if ?x occurs in $\mu$; $\text{false}$ otherwise

• ?x = c  
  • $\mu(?x = c) = \text{error}$ if ?x does not occur in $\mu$;
  • $\mu(?x = c) = \text{false}$ else if $\mu(?x) = c$;

• ?x ≠ c  
  • $\mu(?x = c) = \text{true}$ otherwise.
  • $\mu(?x ≠ c) = \text{error}$ if ?x does not occur in $\mu$;
  • $\mu(?x ≠ c) = \text{true}$ else if $\mu(?x) ≠ c$;

• ?x = ?y  
  • $\mu(?x ≠ c) = \text{false}$ otherwise.
  • $\mu(?x ≠ c) = \text{false}$ otherwise.

• ?x ≠ ?y  
  • $\mu(?x ≠ c) = \text{false}$ otherwise.

• ... ...
• disjunction: $\lor$
• conjunction: $\land$  
  • $\mu(?x ≠ c) = \text{false}$ otherwise.

... ...
Difference is expressible in three fragments.

Difference $(e_1 - e_2)$ is expressible in SPARQL ($\neg$bound)

$$\left( P_{e_1} \text{ OPT } (P_{e_2} \text{ AND } (?u, r, ?w)) \right) \text{ FILTER } \neg \text{bound}(?u).$$

Difference $(e_1 - e_2)$ is expressible in SPARQL $(=, \neq)$

$$\left( \left( P_{e_1} \text{ OPT } ((P_{e_2} \text{ AND } \text{adom}_?u \text{ AND } \text{adom}_?w') \text{ FILTER } ?u \neq ?u') \right) \text{ AND } \text{adom}_?u \text{ AND } \text{adom}_?w' \right) \text{ FILTER } ?u = ?u'.$$

Difference $(e_1 - e_2)$ is expressible in SPARQL $(\neq, \neq_c)$

$$\left( \left( P_{e_1} \text{ OPT } ((P_{e_2} \text{ AND } \text{adom}_?u) \text{ FILTER } ?u = a) \right) \text{ AND } \text{adom}_?u \right) \text{ FILTER } ?u = b.$$