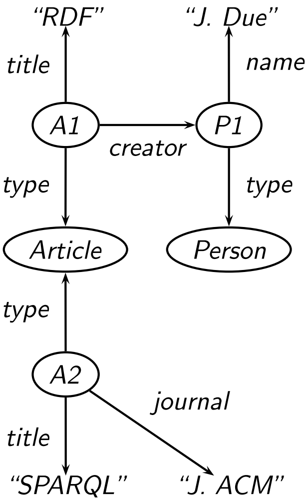
Foundations of SPARQL Query Optimization

Michael Schmidt, Michael Meier, Georg Lausen

Albert-Ludwigs-Universität Freiburg
– Database and Information Systems Group –

13th International Conference on Database Theory, March 2010

The RDF Data Format



The RDF Data Format

- W3C Recommendation
- **R**esource **D**escription **F**ramework
- Graph-structured data
- Recently developed to encode data in the Semantic Web
- Well-suited for semi-structured and unstructured data

"Triples of knowledge"

$$D := \{(A1, type, Article), (A1, title, \text{“RDF”}), (A1, creator, P1), (P1, type, Person), \ldots\}$$

SPARQL: A Query Language for RDF

Basic Construct

Triples that may contain variables in positions, e.g. (*?article*, *type*, *Article*)

SPARQL: A Query Language for RDF

Basic Construct

Triples that may contain variables in positions, e.g. $(\textit{?article}, \textit{type}, \textit{Article})$

SPARQL Evaluation

Evaluation result described by mapping sets, e.g.
$[\![(\textit{?article}, \textit{type}, \textit{Article})]\!]_D = \{\{\textit{?article} \mapsto A1\}, \{\textit{?article} \mapsto A2\}\}$

SPARQL: A Query Language for RDF

Basic Construct

Triples that may contain variables in positions, e.g. $(?article, type, Article)$

SPARQL Evaluation

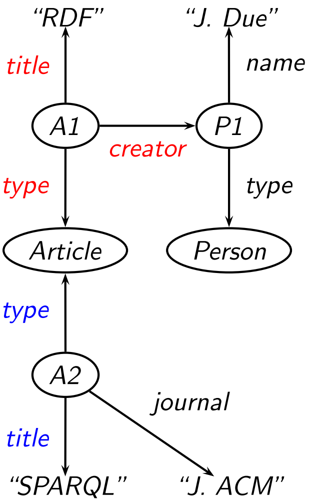
Evaluation result described by mapping sets, e.g.
$[\![(?article, type, Article)]\!]_D = \{\{?article \mapsto A1\}, \{?article \mapsto A2\}\}$

Operators to Build More Expressive Queries (will be explained later)

Select, And, Filter, Optional, Union

$\Rightarrow$ no explicit minus operator, but negation can be simulated using a combination of operators Optional and Filter

SPARQL Example



SPARQL Query (Abstract Syntax)

Select the titles of all articles and, optionally, the persons that authored the articles:

Q :

```
SELECT ?title ?person
WHERE {
  ((?article,type,Article) AND
   (?article,title,?title))
  OPTIONAL (?article,creator,?person)
}
```

Solution

$$[\![Q]\!]_D = \{\{?title \mapsto \text{"RDF"}, ?person \mapsto P1\}, \{?title \mapsto \text{"SPARQL"}\}\}$$

SPARQL Semantics

SPARQL Semantics Idea

- Semantics $[\![.]\!]_D$ maps SPARQL operators into algebraic operations over mapping sets
- SPARQL Algebra operations similar in idea to Relational Algebra

SPARQL Semantics

SPARQL Semantics Idea

- Semantics $[\![.]\!]_D$ maps SPARQL operators into algebraic operations over mapping sets
- SPARQL Algebra operations similar in idea to Relational Algebra

Translation of SPARQL Operators into SPARQL Algebra

- $\textsc{Select}\ ?x_1 \ldots ?x_n\ \textsc{Where}\ \{A\}$:
  $\Rightarrow$ mapped to a projection operation $\pi_{?x_1 \ldots ?x_n}([\![A]\!]_D)$

SPARQL Semantics

SPARQL Semantics Idea

- Semantics $[\![.]\!]_D$ maps SPARQL operators into algebraic operations over mapping sets
- SPARQL Algebra operations similar in idea to Relational Algebra

Translation of SPARQL Operators into SPARQL Algebra

- SELECT $?x_1 \ldots ?x_n$ WHERE $\{A\}$:
  $\Rightarrow$ mapped to a projection operation $\pi_{?x_1 \ldots ?x_n}([\![A]\!]_D)$
- A AND B (also denoted as "A . B"):
  $\Rightarrow$ mapped to an algebraic join operation $[\![A]\!]_D \bowtie [\![B]\!]_D$

SPARQL Semantics

SPARQL Semantics Idea

- Semantics $[\![.]\!]_D$ maps SPARQL operators into algebraic operations over mapping sets
- SPARQL Algebra operations similar in idea to Relational Algebra

Translation of SPARQL Operators into SPARQL Algebra

- SELECT $?x_1 \ldots ?x_n$ WHERE $\{A\}$:
  $\Rightarrow$ mapped to a projection operation $\pi_{?x_1 \ldots ?x_n}([\![A]\!]_D)$
- A AND B (also denoted as "A . B"):
  $\Rightarrow$ mapped to an algebraic join operation $[\![A]\!]_D \bowtie [\![B]\!]_D$
- A OPTIONAL B:
  $\Rightarrow$ mapped to a left outer join $[\![A]\!]_D ⟕ [\![B]\!]_D$

SPARQL Semantics

SPARQL Semantics Idea

- Semantics $[\![.]\!]_D$ maps SPARQL operators into algebraic operations over mapping sets
- SPARQL Algebra operations similar in idea to Relational Algebra

Translation of SPARQL Operators into SPARQL Algebra

- $\textsc{Select}\ ?x_1 \ldots ?x_n\ \textsc{Where}\ \{A\}$:
  $\Rightarrow$ mapped to a projection operation $\pi_{?x_1 \ldots ?x_n}([\![A]\!]_D)$
- $A\ \textsc{And}\ B$ (also denoted as "$A\ .\ B$"):
  $\Rightarrow$ mapped to an algebraic join operation $[\![A]\!]_D \bowtie [\![B]\!]_D$
- $A\ \textsc{Optional}\ B$:
  $\Rightarrow$ mapped to a left outer join $[\![A]\!]_D ⟕ [\![B]\!]_D$
- $A\ \textsc{Filter}\ R$:
  $\Rightarrow$ mapped to an algebraic selection $\sigma_R([\![A]\!]_D)$

SPARQL Semantics

SPARQL Semantics Idea

- Semantics $[\![.]\!]_D$ maps SPARQL operators into algebraic operations over mapping sets
- SPARQL Algebra operations similar in idea to Relational Algebra

Translation of SPARQL Operators into SPARQL Algebra

- SELECT $?x_1 \ldots ?x_n$ WHERE $\{A\}$:
  $\Rightarrow$ mapped to a projection operation $\pi_{?x_1 \ldots ?x_n}([\![A]\!]_D)$
- A AND B (also denoted as "A . B"):
  $\Rightarrow$ mapped to an algebraic join operation $[\![A]\!]_D \bowtie [\![B]\!]_D$
- A OPTIONAL B:
  $\Rightarrow$ mapped to a left outer join $[\![A]\!]_D ⟕ [\![B]\!]_D$
- A FILTER R:
  $\Rightarrow$ mapped to an algebraic selection $\sigma_R([\![A]\!]_D)$
- A UNION B:
  $\Rightarrow$ mapped to a set-theoretic union $[\![A]\!]_D \cup [\![B]\!]_D$

Contributions and Outline

Key Contributions of the Paper

1. Comprehensive study of SPARQL evaluation complexity
2. Novel techniques for SPARQL query optimization
   - Algebraic query optimization
   - Semantic query optimization

The SPARQL Evaluation Problem

EVALUATION Problem

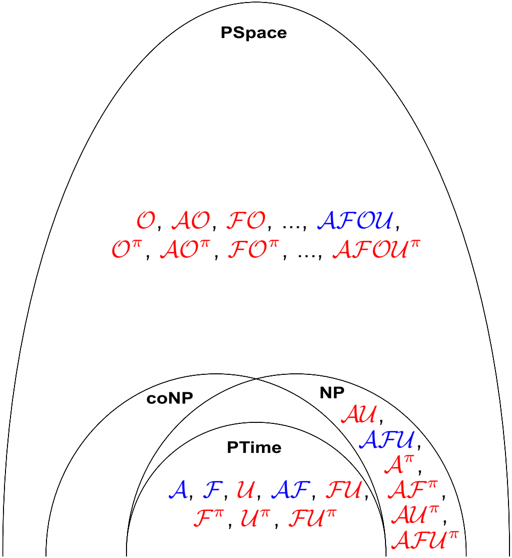
Input: SPARQL query Q, candidate mapping μ, and RDF document D
Question: Is $\mu \in [\![Q]\!]_D$?

Answer: *yes* or *no*

Motivation

- Default problem to understand the complexity of query evaluation and the expressiveness/limitations of query languages
- Study of the problem for different language fragments
- Results allow to understand relations between SPARQL fragments and established query languages (such as Relational Algebra)

Complete Classification



Operator Shortcuts

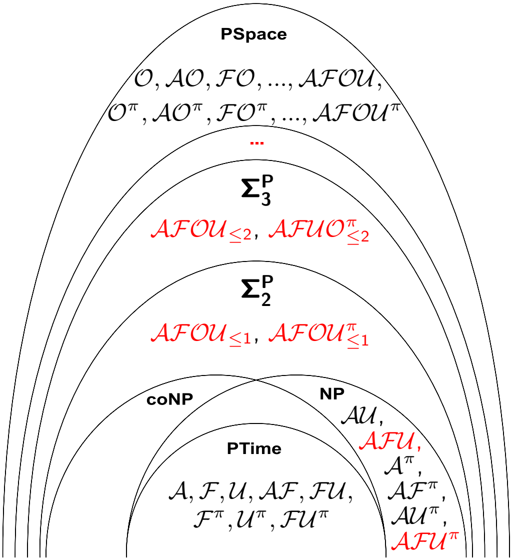
- $\mathcal{A}$:= AND, $\mathcal{O}$:= OPTIONAL, $\mathcal{F}$:= FILTER, $\mathcal{U}$:= UNION
- π := SELECT

Central Results[a]

[a] Results in blue from *J. Pérez, M. Arenas, C. Gutierrez: Semantics and Complexity of SPARQL. In ISWC 2006.*

- Already OPTIONAL alone leads to PSPACE-hardness
- Combinations AND/UNION and AND/SELECT cause NP-hardness
- Operator FILTER never affects complexity

The Source of Complexity



Operator Shortcuts

- $\mathcal{A}$:= AND, $\mathcal{O}$:= OPTIONAL, $\mathcal{F}$:= FILTER, $\mathcal{U}$:= UNION
- π := SELECT
- $\leq n$:= expression contains at most n nested OPTIONAL subexpressions

Central Result

- Nesting depth of OPTIONAL expressions fixes class in the polynomial hierarchy

Algebraic Rewritings for SPARQL

Central Results: Various Algebraic Equivalences over SPARQL Algebra

- Comprehensive investigation of algebraic laws
  - Associativity, Commutativity, Distributivity
  - Idempotence and Inverse
- Study of rewriting rules known from Relational Algebra
  - Filter Pushing
  - Projection Pushing
- SPARQL-specific rewritings
  - Rules concerning unbound variables
  - Simulated negation
- Difference between bag and set semantics

$\Rightarrow$ in total about 30 new equivalences over SPARQL Algebra

SPARQL Algebra vs. Relational Algebra

Question

Given the close connection between Relational and SPARQL Algebra, do the rewritings trivially follow from Relational Algebra rewritings?

SPARQL Algebra vs. Relational Algebra

Question

Given the close connection between Relational and SPARQL Algebra, do the rewritings trivially follow from Relational Algebra rewritings?

Answer

There are important discrepancies between both algebras, most notably:

Relational Algebra

- Defined over relations with fixed schema
- Joins over NULL values rejecting

SPARQL Algebra

- Defined over mapping sets with loose schema
- Joins over unbound variables accepting

$\Rightarrow$ profound implications that complicate SPARQL Algebra rewritings

SPARQL Algebra vs. Relational Algebra

Rejecting vs. Accepting Joins

Relational Algebra (SQL)

R_{RA}	A	B
	1	2

S_{RA}	B	C
	NULL	3

$R_{RA} \bowtie S_{RA} = \emptyset$

SPARQL Algebra

$R_{SA} := \{\{?A \mapsto 1, ?B \mapsto 2\}\}$

$S_{SA} := \{\{?C \mapsto 3\}\}$

$R_{SA} \bowtie S_{SA} =$
$\{\{?A \mapsto 1, ?B \mapsto 2, ?C \mapsto 3\}\}$

Example

Candidate SPARQL Algebra Equivalence: Filter Pushing

$$\text{(FP)} \quad \sigma_{?y=d}(A \bowtie B) \stackrel{?}{\equiv} \sigma_{?y=d}(A) \bowtie B$$

Problem

- Assume the evaluation result of A contains $\mu := \{?x \mapsto c\}$
- μ never contributes to the result of the right-side expression
- **But:** μ may contribute to the result of the left-side expression, e.g. when joined with a mapping $\nu := \{?y \mapsto d\}$ from B

Approximating Bound Variables

Approach

- Classify variables appearing in SPARQL Algebra expressions
  - *Possible variables*: overestimation for variables that might be bound in every result mapping
  - *Certain variables*: underestimation for variables that are definitely bound in every result mappings
- Classification independent from input document

Approximating Bound Variables

Approach

- Classify variables appearing in SPARQL Algebra expressions
  - *Possible variables*: overestimation for variables that might be bound in every result mapping
  - *Certain variables*: underestimation for variables that are definitely bound in every result mappings
- Classification independent from input document

Filter Pushing Revisited

If $?y$ is a *certain* variable of A or $?y$ is not a *possible* variable of B, then

$$\text{(FP)} \quad \sigma_{?y=d}(A \bowtie B) \equiv \sigma_{?y=d}(A) \bowtie B$$

Semantic Query Optimization in SPARQL

Motivation: Semantic Query Optimization

- Constraints are vital to assert data integrity and avoid anomalies
  - Keys and functional dependencies
  - Foreign keys
  - User-defined constraints (e.g., subclass relationships)
- Semantic query optimization
  - Find queries that are equivalent on each database instance satisfying the constraints
  - Queries often exhibit non-trivial rewritings in the presence of constraints
  - Rich theory of constraint-based optimization in the Relational context has been developed

RDF Constraints in SPARQL

Representing RDF Constraints

Idea: encode RDF constraints as First-order Logic sentences over a ternary predicate $D(subject, predicate, object)$.

Upcoming Question

Is SPARQL a candidate for a constraint language, i.e. is it expressive enough to ...

- ... test if a (first-order) constraint holds on some database?
- ... express user-defined constraints?

RDF Constraints in SPARQL

Representing RDF Constraints

Idea: encode RDF constraints as First-order Logic sentences over a ternary predicate $D(subject, predicate, object)$.

Upcoming Question

Is SPARQL a candidate for a constraint language, i.e. is it expressive enough to ...

- ... test if a (first-order) constraint holds on some database?
- ... express user-defined constraints?

For a slight extension of SPARQL, called SPARQL^C, the following holds:

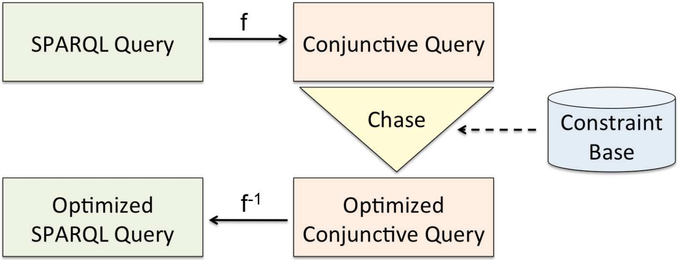
Theorem

Let φ be a (first-order) RDF constraint and D be an RDF document. There is a SPARQL^C query Q s.t. $[\![Q]\!]_D = \emptyset \Leftrightarrow D \models \varphi$.

Semantic Query Optimization in SPARQL

Exploiting RDF Constraints for Query Optimization

- Exploit close connection between SPARQL AND-only queries and Conjunctive Queries, an important subclass of Relational Algebra
- Fall back on established semantic optimization techniques developed in the Relational context, namely the classical *Chase* algorithm



Semantic Query Optimization in SPARQL

Central Results and Properties

- Provably **sound** approach to semantic query optimization for AND-only queries
- Also works for AND-connected blocks inside more complex queries
- **Completeness** depends mainly on the termination of the Chase algorithm (like in the Relation context)
- Can easily be coupled with a cost estimation function
- Complemented by a rule-based optimization approach for the remaining SPARQL operators (e.g., OPTIONAL)

Conclusion

Closing Remarks

- Research on SPARQL still in its infancy
- Results in the paper improve on two critical research subjects
  - **Complexity results**
    $\Rightarrow$ deepen understanding of the SPARQL query language, its fragments, and interrelations to established query languages
  - **Algebraic and semantic optimization approaches**
    $\Rightarrow$ lay the theoretical foundations for building efficient, scalable SPARQL engines
- An important step towards efficient Semantic Web data processing

Thank you for your attention!