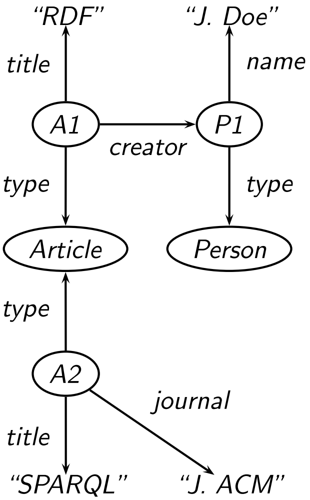
Foundations of SPARQL Query Optimization

Michael Schmidt

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

Disputation Talk, 22 December 2009

The RDF Data Format



The RDF Data Format

- W3C Recommendation
- **R**esource **D**escription **F**ramework
- Graph-structured data model
- Developed to encode information 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: Triple Patterns

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: Triple Patterns

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

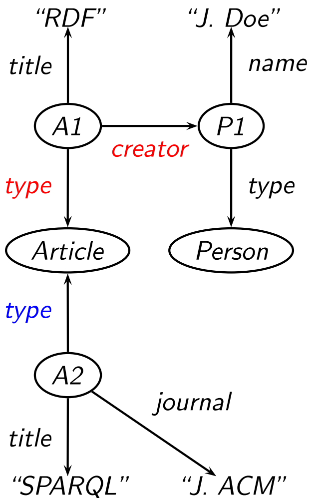
SPARQL Evaluation

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

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

SELECT, AND, UNION, FILTER, OPTIONAL

SPARQL: A Query Language for RDF



SPARQL Query

Select all articles and, if present in the database, the persons that authored the articles:

Q :

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

Solution

$$[\![Q]\!]_D = \{\{?article \mapsto A1, ?person \mapsto P1\}, \{?article \mapsto A2\}\}$$

Outline

1. **Novel techniques for SPARQL query optimization**
   - **Algebraic query optimization**
   - **Semantic query optimization**
2. Study of SPARQL evaluation complexity
3. SP^2Bench: a performance benchmark for 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

- 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

- 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

- 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 $\sigma_R([\![A]\!]_D)$
- ...

Algebraic Rewritings for SPARQL

Central Results: Various Algebraic Equivalences over SPARQL Algebra

- In total about 30 new equivalences over SPARQL Algebra
- Cover different aspects of SPARQL Algebra
  - Comprehensive investigation of algebraic laws
  - Study of rewriting rules known from Relational Algebra
  - SPARQL-specific rewritings
- Complicated by the issue of unbound variables in result mappings

SPARQL Algebra Rewritings: 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

SPARQL Algebra Rewritings: 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

Approach

- Approximate variables that are definitely bound/unbound
- Useful tool to state equivalences in a compact and precise way

Semantic Query Optimization in SPARQL

Motivation: Semantic Query Optimization

- Databases typically satisfy a variety of integrity constraints
  - Keys and functional dependencies
  - Foreign keys
  - User-defined constraints (e.g., subclass relationships)
- Semantic query optimization
  - Goal: find queries that are equivalent on each database instance satisfying the constraints
  - Rich theory of constraint-based optimization in the Relational context has been developed

Semantic Query Optimization in SPARQL

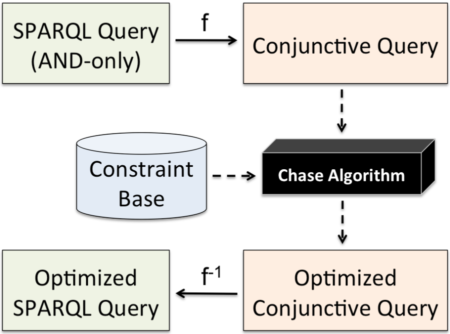
Exploiting RDF Constraints for Query Optimization

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

Semantic Query Optimization in SPARQL

Exploiting RDF Constraints for Query Optimization

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



Optimization Scheme

- Provably sound optimization approach for SPARQL AND-only queries
- Also works for AND-connected subqueries
- Complemented by rule-based optimization of remaining SPARQL operators

Outline

1. Novel techniques for SPARQL query optimization
   - Algebraic query optimization
   - Semantic query optimization
2. **Study of SPARQL evaluation complexity**
3. SP^{2}Bench: a performance benchmark for SPARQL

The SPARQL Evaluation Problem

EVALUATION Problem

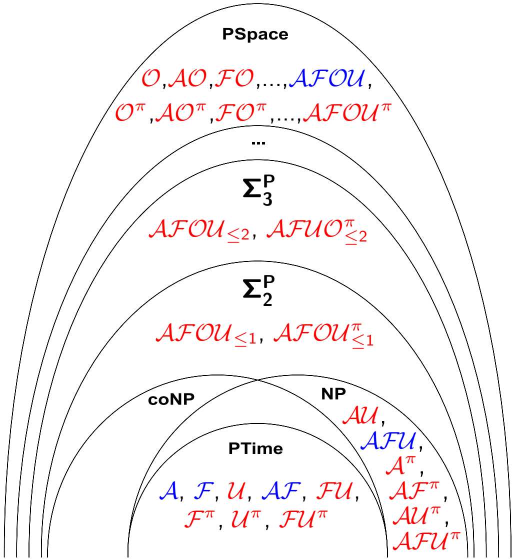
Input: SPARQL query Q, candidate mapping μ, and RDF document D
Question: Is $\mu \in [\![Q]\!]_D$?

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)

Complexity of SPARQL Evaluation



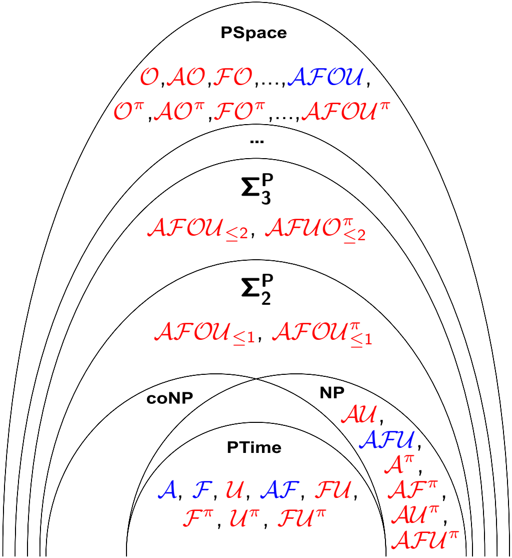
Fragment Shortcuts

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

Further Explanation

- blue color: previous work
- red color: new results
- Fragments in NP, Σ_i^P, and PSPACE are complete for the respective classes

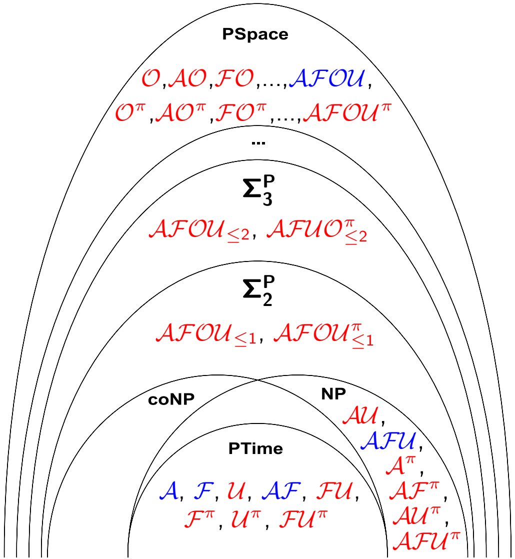
Complexity of SPARQL Evaluation



Central Results

- **Complete** classification of all operator fragments

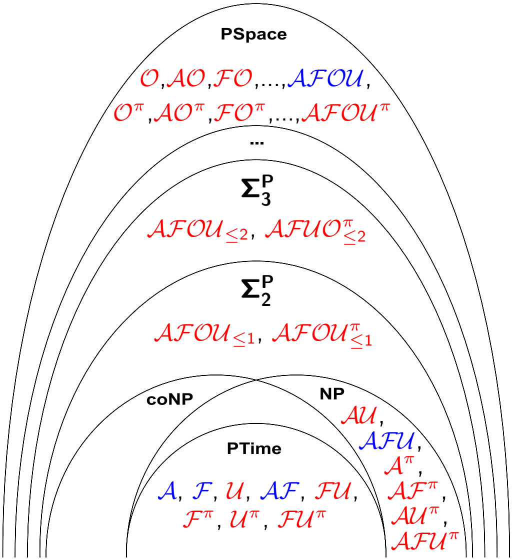
Complexity of SPARQL Evaluation



Central Results

- **Complete** classification of all operator fragments
- Already OPTIONAL alone leads to PSPACE-hardness

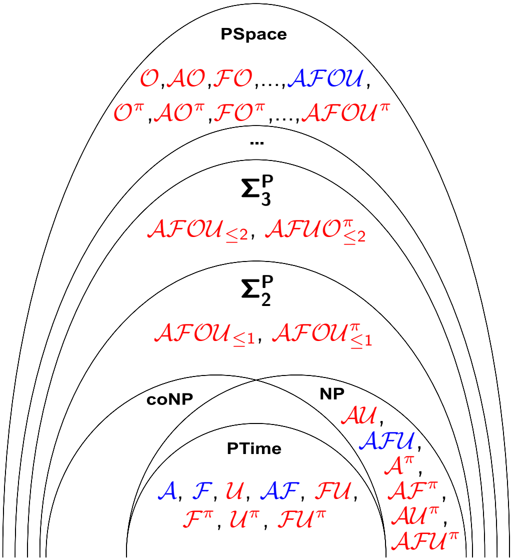
Complexity of SPARQL Evaluation



Central Results

- **Complete** classification of all operator fragments
- Already OPTIONAL alone leads to PSPACE-hardness
- Nesting depth of OPTIONAL expressions fixes class in the polynomial hierarchy

Complexity of SPARQL Evaluation



Central Results

- **Complete** classification of all operator fragments
- Already OPTIONAL alone leads to PSPACE-hardness
- Nesting depth of OPTIONAL expressions fixes class in the polynomial hierarchy
- Combinations AND/UNION and AND/SELECT cause NP-hardness

Outline

1. Novel techniques for SPARQL query optimization
   - Algebraic query optimization
   - Semantic query optimization
2. Study of SPARQL evaluation complexity
3. **SP^{2}Bench: a performance benchmark for SPARQL**

SP^2Bench: A SPARQL Performance Benchmark

SP^2Bench

Language-specific benchmark to test SPARQL engines

SP^{2}Bench: A SPARQL Performance Benchmark

SP^{2}Bench

Language-specific benchmark to test SPARQL engines

Data Generator

- **Scenario:** DBLP library
- Creates arbitrarily large RDF documents
- Builds upon in-depth study of DBLP distributions
- Approximated by natural function families
  - Powerlaw distributions
  - Logistic curves

SP²Bench: A SPARQL Performance Benchmark

SP²Bench

Language-specific benchmark to test SPARQL engines

Data Generator

- **Scenario:** DBLP library
- Creates arbitrarily large RDF documents
- Builds upon in-depth study of DBLP distributions
- Approximated by natural function families
  - Powerlaw distributions
  - Logistic curves

Real-world Queries

- In total 14 SELECT queries and 3 boolean queries
- Vary in broad range of characteristics
  - Operator constellations
  - Solution modifiers (like DISTINCT, ORDER BY)
  - RDF access patterns
  - Optimization approaches
  - ...

SP²Bench: A SPARQL Performance Benchmark

SP²Bench

Language-specific benchmark to test SPARQL engines

Data Generator

- **Scenario:** DBLP library
- Creates arbitrarily large RDF documents
- Builds upon in-depth study of DBLP distributions
- Approximated by natural function families
  - Powerlaw distributions
  - Logistic curves

Real-world Queries

- In total 14 SELECT queries and 3 boolean queries
- Vary in broad range of characteristics
  - Operator constellations
  - Solution modifiers (like DISTINCT, ORDER BY)
  - RDF access patterns
  - Optimization approaches
  - ...

Performance Metrics

Standardized way to evaluate and compare SP²Bench benchmark results

Benchmark Queries and Optimization

Example: SP2Bench Queries *Q5a* and *Q5b*

Informally: *Return the names of all persons that occur as author of at least one inproceeding and at least one article document.*

Q5a:

```
SELECT DISTINCT ?person ?name
WHERE { ?article type    Article.
        ?article creator ?person.
        ?inproc  type    Inproceedings.
        ?inproc  creator ?person2.
        ?person  name    ?name.
        ?person2 name    ?name2
        FILTER (?name=?name2) }
```

Q5b:

```
SELECT DISTINCT ?person ?name
WHERE { ?article type    Article.
        ?article creator ?person.
        ?inproc  type    Inproceedings.
        ?inproc  creator ?person.
        ?person  name    ?name }
```

- *Q5a*: implicit join through filter *?name* = *?name2*
- *Q5b*: explicit join on the variable *?person*
- **Observation:** *Q5b* several orders of magnitude faster than *Q5a* when executed with state-of-the-art engines
- Can be transformed into each other using algebraic and semantic optimization

Conclusion

Closing Remarks

- Research on SPARQL evaluation still in its infancy
- Results in the thesis improve on critical research subjects
  - **Algebraic and semantic optimization approaches**
    $\Rightarrow$ lay the theoretical foundations for building efficient, scalable SPARQL engines
  - **Complexity results**
    $\Rightarrow$ deepen understanding of the SPARQL query language, its fragments, and interrelations to established query languages
  - **SP^{2}Bench performance benchmark:**
    $\Rightarrow$ eases comprehensive testing of SPARQL engines and may improve future research in this area
- An important step towards efficient Semantic Web data processing

Thank you for your attention!