TECHNISCHE UNIVERSITÄT DRESDEN

KNOWLEDGE GRAPHS

Lecture 8: Limits of SPARQL

Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 2nd Dec 2019

Review

SPARQL is:

- PSpace-complete for combined and query complexity
- NL-complete for data complexity

$\leadsto$ scalable in the size of RDF graphs, not really in the size of query
$\leadsto$ similar situation to other query languages

Hardness is shown by reducing from known hard problems

- Truth of quantified boolean formulae (QBF)
- Reachability in a directed graph

Membership is shown by (sketching) appropriate algorithms

- Naive, iteration-based solution finding procedure runs in polynomial space
- For fixed queries, the complexity drop to nondeterministic logspace

Expressive power

Expressive power and the limits of SPARQL

The expressive power of a query language is described by the question:
"Which sets of RDF graphs can I distinguish using a query of that language?"

More formally:

- Every query defines a set of RDF graphs: the set of graph that it returns at least one result for
- However, not every set of RDF graphs corresponds to a query (exercise: why?)

Note: Whether a query has any results at all is not what we usually ask for, but it helps us here to create a simpler classification. One could also compare query results over a graph and obtain similar insights overall.

Definition 8.1: We say that a query language $\mathbf{Q_1}$ is more expressive than another query language $\mathbf{Q_2}$ if it can characterise strictly more sets of graphs.

Complexity limits expressivity

Intuition: The lower the complexity of query answering, the lower its expressivity.

Question: which complexity are we talking about here? — data complexity!

- Given a set of RDF graphs that we would like to classify,
- we ask if there is one (fixed) query that accomplishes this.

If classifying the set of graphs encodes a computationally difficult problem, then the query evaluation has to be at least as hard as this problem with respect to data complexity.

Example 8.2: We have argued that SPARQL queries can evaluate QBF, and we could encode QBF in RDF graphs (in many reasonable ways). However, there cannot be a SPARQL query that recognises all RDF graphs that encode a true QBF, since this problem is PSpace-complete, which is known to be not in NL.

Complexity is not the same as expressivity

Complexity-based arguments are often quite limited:

- They only apply to significantly harder problems
- Additional assumptions are often needed (e.g., it is assumed that NL ≠ NP, but it was not proven yet)
- Typically, query languages cannot even solve all problems in their own complexity class (i.e., they do not "capture" this class)

$\leadsto$ Direct arguments for non-expressivity need to be sought.

Example: Complexity ≠ expressivity

The problem of parallel reachability is defined as follows:
Given: An RDF graph G; two vertices s and t; and two RDF properties p and q
Question: Is there a directed path from s to t, where each two neighbouring nodes on the path are connected by both a p-edge and a q-edge?

Proposition 8.3: Parallel reachability is in NL.

Proof: The check can be done using a similar algorithm as for s-t-reachability, merely checking for two edges in each step. □

Proposition 8.4: SPARQL cannot express parallel reachability.

Proof: The only SPARQL feature that can check for paths are property path patterns, but:

- a match to a property path pattern is always possible using only vertices of degree 2 on the path; higher degrees can only be enforced for a limited number of nodes that are matched to query variables
- the query requires an arbitrary number of nodes of degree 4 on the path □

Other structural limits to SPARQL reachability expressivity

SPARQL's regular recursions is also limited in many other cases:

- Non-regular path languages cannot be expressed
- "Wide" paths consisting of repeated graph patterns cannot be expressed
- Tree-like patterns and other non-linear patterns cannot be expressed
- "Nested regular expressions" with tests cannot be expressed

(See board)

Limits by design

Besides mere expressivity, SPARQL also has some fundamental limits since it simply has no support for some query or analysis tasks:

- SPARQL is lacking some datatypes and matching filter conditions, most notably geographic coordinates (major RDF databases add this)
- SPARQL cannot talk about path lengths, e.g., one cannot retrieve the length of the shortest connecting path between two elements
- SPARQL cannot return paths (of a priori unknown length) in results
- SPARQL has no support for recursive/iterative computation, e.g., for page rank or other graph algorithms

Potential reasons: performance concerns (e.g., page rank computation would mostly take too long; longest path detection is NP-complete [in data complexity!]), historic coincidence (geo coordinates not in XML Schema datatypes); design issues (handling paths in query results would require many different constructs)

Outlook

SPARQL: Outlook

A number of SPARQL features have not been discussed:

- Graphs: SPARQL supports querying RDF datasets with multiple graphs, and queries can retrieve graph names as variable bindings
- Updates: SPARQL has a set of features for inserting and deleting data

Example 8.5: The following query replaces all uses of the `hasSister` property with a different encoding of the same information:

```
DELETE { ?person eg:hasSister ?sister }
INSERT {
 ?person eg:hasSibling ?sister .
 ?sister eg:sex eg:female .
}
WHERE { ?person eg:hasSister ?sister }
```

- Result formats: SPARQL has several encodings for sending results, and it can also encode results as RDF graphs (**CONSTRUCT**).
- Federated queries: SPARQL can get sub-query results from other SPARQL services

Datalog

A rule-based query language

Datalog has been introduced as a rule-based query language in (relational) databases

Example 8.6: The following set of rules describes a query for all ancestors of the individual Alice from the given binary relations father and mother, where we assume that the predicate Result denotes the output relation:

$$\begin{aligned}\text{Parent}(x, y) &:- \text{father}(x, y)\\ \text{Parent}(x, y) &:- \text{mother}(x, y)\\ \text{Ancestor}(x, y) &:- \text{Parent}(x, y)\\ \text{Ancestor}(x, z) &:- \text{Parent}(x, y), \text{Ancestor}(y, z)\\ \text{Result}(y) &:- \text{Ancestor}(\text{alice}, y)\end{aligned}$$

Rules have their consequence on the left and preconditions on the right, so we can read :– as "if" and , as "and".

It is not hard to apply this approach to graphs.

Datalog Syntax

To define Datalog, we recall some basic definitions of (predicate) logic:

- We use mutually disjoint (infinite) sets of constants, variables, and predicate symbols
- A term is a constant or a variable.
- An atom is a formula of the form $R(t_1, \ldots, t_n)$ with R a predicate symbol of arity n, and $t_1, \ldots, t_n$ terms.

Definition 8.7: A Datalog rule is an expression of the form:

$$H :- B_1, \ldots, B_m$$

where H and $B_1, \ldots, B_m$ are atoms. H is called the head or conclusion; $B_1, \ldots > B_m$ is called the body or premise. A rule with empty body ($m = 0$) is called a fact. A ground rule is one without variables (i.e., all terms are constants).

A set of Datalog rules is a Datalog program. A Datalog query is a Datalog program together with a distinguished query predicate.

Datalog Semantics

A Datalog query is evaluated on a given database, which we can view as a set of facts.

Example 8.8: If the database table for mother is given by

alice	barbara
barbara	christine
dave	emmy

then this data is represented by the facts mother(alice, barbara), mother(barbara, christine), and mother(dave, emmy).

We can then define Datalog query results based on the usual semantics of first-order logic:

Definition 8.9: The result of a Datalog query $\langle P, \text{Result}\rangle$ over a database D is the set of all facts over Result that are logically entailed by $D \cup P$ when reading Datalog rules as first-order logic implications (from right to left).

Note: Datalog semantics is set-based (no multiplicity of results).

Summary

SPARQL expressivity is still limited, partly by design

Datalog is a rule-based query language that can express more powerful recursive queries

What's next?

- More about Datalog
- Property graph
- The Cypher query language