Recursion in SPARQL – online appendix

Juan L. Reutter, Adrián Soto, and Domagoj Vrgoč

PUC Chile and Center for Semantic Web Research

Appendix

Proof Sketches

Proof of Proposition 1. It was proved in [1] that the problem CONSTRUCTQUERYANSWERING is NP-complete for non recursive c-queries, and Pérez et al. show in [3] that the problem SELECTQUERYANSWERING if PSPACE-complete for non-recursive SPARQL queries, and Π_2^p for non-recursive SPARQL queries given by unions of well-designed graph patterns.

This immediately gives us hardness for all three problems when recursion is allowed. To see that the upper bound is maintained, note that for each nested query, the temporal graph can have at most $|D|^3$ triples. Since we are computing the least fixed point, this means that in every iteration we add at least one triple, and thus the number of iterations is polynomial. This in turn implies that the answer can be found by composing a polynomial number of NP problems, to realize the database, followed with a PSPACE problem for SELECTQUERYANSWERING, a Π_2^p problem for SELECTQUERYANSWERING assuming queries given by unions of well designed patterns and NP problems for CONSTRUCTQUERYANSWERING. First two classes are closed under composition with NP, and the last NP bound can be obtained by just guessing all meaningful queries and computed triples at the same time.

Proof of Proposition 2. Both SELECT and CONSTRUCT have NLOGSPACE data complexity (see again [3] and [1]). Following the same idea as in the proof of Proposition 1, upper bound then follows by composing a polynomial number of NLOGSPACE algorithms.

For the lower bound one can easily do a reduction from the problem of computing the answers of a single rule datalog problem (see Georg Gottlob, Christos Papadimitriou, On the complexity of single-rule datalog queries, Information and Computation, Volume 183, Issue 1, 25 May 2003, Pages 104-122), using the transformation from recursive queries to datalog introduced in [1]. The reduction is straightforward and can be carried out both for SELECT queries and for CONSTRUCT queries.

Proof of Theorem 1. Both SELECT and CONSTRUCT can be evaluated in NLOGSPACE in data complexity (see again [3] and [1]). If the recursive query is in addition linear, then the complete derivation of the answer resembles a tree, and one can propose the same algorithm as for showing the NLOGSPACE upper bound for evaluating TriAL expressions in [2].

Queries in Section 4.1.

Query from Subsection 4.1 stated without nesting:

```
PREFIX prov: <http://www.w3.org/ns/prov#>
WITH RECURSIVE http://db.ing.puc.cl/temp AS {
    CONSTRUCT {?x ?u ?y}
    FROM NAMED <http://db.ing.puc.cl/temp>
    WHERE{{
        ?x prov:wasRevisionOf ?z .
        ?x prov:wasGeneratedBy ?w .
        ?w prov:used ?z .
        ?w prov:wasAssociatedWith ?u}
    UNION{
        ?x prov:wasRevisionOf ?z .
        ?x prov:wasGeneratedBy ?w .
        ?w prov:used ?z .
        ?w prov:wasAssociatedWith ?u .
        GRAPH <http://db.ing.puc.cl/temp> {?z ?u ?y}}}
}
SELECT ?x ?y
FROM <http://db.ing.puc.cl/temp>
WHERE ?x ?u ?y
```

Queries from Subsection 5.1.

The query Q1 is expressed using the following recursive query:

```
WITH RECURSIVE http://db.ing.puc.cl/temp AS{
    CONSTRUCT {<http://data.linkedmdb.org/resource/actor/29539>
    <http://relationship.com/collab> ?act}
    FROM NAMED <http://db.ing.puc.cl/temp>
FROM <Quad.defaultGraphIRI>
WHERE {
{?mov <http://data.linkedmdb.org/resource/movie/actor>
<http://data.linkedmdb.org/resource/actor/29539> .
?mov <http://data.linkedmdb.org/resource/movie/actor> ?act}
UNION {
{?mov <http://data.linkedmdb.org/resource/movie/actor> ?act1} .
{?mov <http://data.linkedmdb.org/resource/movie/actor> ?act} .
GRAPH <http://db.ing.puc.cl/temp>
{<http://data.linkedmdb.org/resource/actor/29539>
<http://relationship.com/collab> ?act1}}
}
}
SELECT ?z FROM NAMED <http://db.ing.puc.cl/temp>
WHERE {GRAPH <http://db.ing.puc.cl/temp>
{<http://data.linkedmdb.org/resource/actor/29539>
<http://relationship.com/collab> ?z}}
```

The following is the formulation of the query Q2:

```
WITH RECURSIVE http://db.ing.puc.cl/temp AS
{
CONSTRUCT {<http://data.linkedmdb.org/resource/actor/29539> ?dir ?act}
FROM NAMED <http://db.ing.puc.cl/temp>
FROM <Quad.defaultGraphIRI>
WHERE
{
{?mov <http://data.linkedmdb.org/resource/movie/actor>
<http://data.linkedmdb.org/resource/actor/29539> .
?mov <http://data.linkedmdb.org/resource/movie/actor> ?act .
?mov <http://data.linkedmdb.org/resource/movie/director> ?dir}
UNION
{{?mov <http://data.linkedmdb.org/resource/movie/director> ?dir} .
{?mov <http://data.linkedmdb.org/resource/movie/actor> ?act1} .
{?mov <http://data.linkedmdb.org/resource/movie/actor> ?act} .
GRAPH <http://db.ing.puc.cl/temp>
{<http://data.linkedmdb.org/resource/actor/29539> ?dir ?act1}}
}
}
SELECT ?y ?z FROM NAMED <http://db.ing.puc.cl/temp>
WHERE {GRAPH <http://db.ing.puc.cl/temp>
{<http://data.linkedmdb.org/resource/actor/29539> ?y ?z}}
```

Below is the recursive formulation of Q3:

```
WITH RECURSIVE http://db.ing.puc.cl/temp AS
{
CONSTRUCT {<http://data.linkedmdb.org/resource/actor/29539>
<http://relationship.com/collab> ?act}
FROM NAMED <http://db.ing.puc.cl/temp>
FROM <Quad.defaultGraphIRI>
WHERE
{
{?mov <http://data.linkedmdb.org/resource/movie/actor>
<http://data.linkedmdb.org/resource/actor/29539> .
?mov <http://data.linkedmdb.org/resource/movie/actor> ?act .
?mov <http://data.linkedmdb.org/resource/movie/director> ?dir .
?dir <http://data.linkedmdb.org/resource/movie/director_name> ?x .
?y <http://data.linkedmdb.org/resource/movie/actor_name> ?x}
UNION
{{?mov <http://data.linkedmdb.org/resource/movie/director> ?dir} .
{?dir <http://data.linkedmdb.org/resource/movie/director_name> ?x} .
{?y <http://data.linkedmdb.org/resource/movie/actor_name> ?x} .
{?mov <http://data.linkedmdb.org/resource/movie/actor> ?act1} .
{?mov <http://data.linkedmdb.org/resource/movie/actor> ?act} .
GRAPH <http://db.ing.puc.cl/temp>
{<http://data.linkedmdb.org/resource/actor/29539>
<http://relationship.com/collab> ?act1}}
}
}
SELECT ?z FROM NAMED <http://db.ing.puc.cl/temp>
WHERE {GRAPH <http://db.ing.puc.cl/temp>
{<http://data.linkedmdb.org/resource/actor/29539>
<http://relationship.com/collab> ?z}}
```

Queries from Subsection 5.2.

The following are the queries Q_A and Q_B expressed using recursion:

```
//The query Q_A
WITH RECURSIVE http://db.ing.puc.cl/temp AS
{
CONSTRUCT {?x <http://relationship.com/wasRevisionOf> ?z}
FROM NAMED <http://db.ing.puc.cl/temp>
WHERE {
{?x <http://relationship.com/wasRevisionOf> ?z}
UNION
{{?y <http://relationship.com/wasRevisionOf> ?z} .
GRAPH <http://db.ing.puc.cl/temp>
{?x <http://relationship.com/wasRevisionOf> ?y}}
}
}
SELECT ?x ?y ?z FROM NAMED <http://db.ing.puc.cl/temp>
WHERE {GRAPH <http://db.ing.puc.cl/temp> {?x ?y ?z}}


//The query Q_B
WITH RECURSIVE http://db.ing.puc.cl/temp AS
{
CONSTRUCT {?x <http://relationship.com/WU> ?z}
FROM NAMED <http://db.ing.puc.cl/temp>
WHERE {
    {?x <http://relationship.com/wasGeneratedBy> ?y .
    ?y <http://relationship.com/used> ?z}
UNION
{{?y <http://relationship.com/wasGeneratedBy> ?r .
?r <http://relationship.com/used> ?z} .
GRAPH <http://db.ing.puc.cl/temp> {?x <http://relationship.com/WU> ?y}}
}
}
SELECT ?x ?y ?z FROM NAMED <http://db.ing.puc.cl/temp>
WHERE {GRAPH <http://db.ing.puc.cl/temp> {?x ?y ?z}}
```

Below are the queries Q_A and Q_B as used by Virtuoso. Note that we have to specify a starting point for a property path in Virtuoso, so we use `?x <http://relationship.com/wasRevisionOf> ?z` to do so. Since for every node in the graph we have only one such triple, this does not increase the evaluation time in a significant way. In Jena the queries are the same, but without the part extracting $?x$.

```
//The query Q_A in Virtuoso:
SELECT ?x ?y
WHERE
{{?x <http://relationship.com/wasRevisionOf> ?z} .
{?x <http://relationship.com/wasRevisionOf>* ?y}}


//The query Q_B in Virtuoso:
SELECT ?x ?y
WHERE
{{?x <http://relationship.com/wasRevisionOf> ?z}.
{?x (<http://relationship.com/wasGeneratedBy>/<http://relationship.com/used>)* ?y}}
```

Queries from Subsection 5.3.

The following is SPARQL rewriting of the query Q1 computing Bacon number of length at most 5:

```
SELECT ?act WHERE{{?mov
<http://data.linkedmdb.org/resource/movie/actor>
<http://data.linkedmdb.org/resource/actor/29539> .
?mov <http://data.linkedmdb.org/resource/movie/actor> ?act}

UNION { ?mov <http://data.linkedmdb.org/resource/movie/actor>
<http://data.linkedmdb.org/resource/actor/29539> .
?mov <http://data.linkedmdb.org/resource/movie/actor> ?act2  .
?mov2 <http://data.linkedmdb.org/resource/movie/actor> ?act2 .
?mov2 <http://data.linkedmdb.org/resource/movie/actor> ?act}

UNION {?mov <http://data.linkedmdb.org/resource/movie/actor>
<http://data.linkedmdb.org/resource/actor/29539> .
?mov <http://data.linkedmdb.org/resource/movie/actor> ?act3  .
?mov2 <http://data.linkedmdb.org/resource/movie/actor> ?act3 .
?mov2 <http://data.linkedmdb.org/resource/movie/actor> ?act2 .
?mov3 <http://data.linkedmdb.org/resource/movie/actor> ?act2 .
?mov3 <http://data.linkedmdb.org/resource/movie/actor> ?act  }

UNION {?mov <http://data.linkedmdb.org/resource/movie/actor>
<http://data.linkedmdb.org/resource/actor/29539> .
?mov <http://data.linkedmdb.org/resource/movie/actor> ?act4  .
?mov2 <http://data.linkedmdb.org/resource/movie/actor> ?act4 .
?mov2 <http://data.linkedmdb.org/resource/movie/actor> ?act3 .
?mov3 <http://data.linkedmdb.org/resource/movie/actor> ?act3 .
?mov3 <http://data.linkedmdb.org/resource/movie/actor> ?act2 .
?mov4 <http://data.linkedmdb.org/resource/movie/actor> ?act2 .
?mov4 <http://data.linkedmdb.org/resource/movie/actor> ?act  }}
```

Other rewritings are similar and can be found at http://web.ing.puc.cl/~jreutter/Recsparql.html.

References

1. E. V. Kostylev, J. L. Reutter, and M. Ugarte. CONSTRUCT queries in SPARQL. In *ICDT*, pages 212–229, 2015.
2. L. Libkin, J. Reutter, and D. Vrgoč. Trial for rdf: adapting graph query languages for rdf data. In *PODS*, pages 201–212. ACM, 2013.
3. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3), 2009.