Łösungen zur Übung zur Lehrveranstaltung

Grundlagen Semantic Web

Seminar für Computerlinguistik, Universität Heidelberg

Sebastian Rudolph

Wintersemester 2009/10

http://semantic-web-grundlagen.de

Übung 4: SPARQL

Aufgabe 4.1 Consider the following RDF document with information about celestial bodies.

```
@prefix  ex: <http://example.org/> .
ex:Sun   ex:radius "1.392e6"^^xsd:double ;
         ex:satellite ex:Mercury, ex:Venus, ex:Earth, ex:Mars .
ex:Mercury  ex:radius     "2439.7"^^xsd:double .
ex:Venus    ex:radius     "6051.9"^^xsd:double .
ex:Earth    ex:radius     "6372.8"^^xsd:double ;
            ex:satellite  ex:Moon .
ex:Mars     ex:radius     "3402.5"^^xsd:double ;
            ex:satellite  ex:Phobos, ex:Deimos .
ex:Moon     ex:name       "Mond@de", "Moon@en" ;
            ex:radius     "1737.1"^^xsd:double .
ex:Phobos   ex:name       "Phobos" .
ex:Deimos   ex:name       "Deimos" .
```

Specify SPARQL queries which yield the following results in the form of a table.

- Objects which orbit around the sun or around a satellite of the sun.

```
PREFIX ex: <http://example.org/> SELECT ?object WHERE
  {{ ex:Sun      ex:satellite  ?object . } UNION
   { ex:Sun      ex:satellite  ?satellite .
     ?satellite  ex:satellite  ?object .}}
```

- Objects with a volume greater than $2 \cdot 10^{10}$ (km^3) together with the object – if it exists – of which they are a satellite. Assume for this that all celestial bodies are spherical.

```
PREFIX ex: <http://example.org/> SELECT ?object ?center WHERE
  {{ ?object   ex:radius   ?radius }
   OPTIONAL { ?center   ex:satellite   ?object . }
   FILTER (4 / 3 * 3.1416 * ?radius * ?radius * ?radius > 20000000000)
  }
```

- Objects with a satellite for which an English name is given, and which furthermore are satellites of an object with diameter greater than 3000 (km).

```
PREFIX ex: <http://example.org/> SELECT ?object WHERE
  {  ?object    ex:satellite  ?satellite .
     ?satellite ex:name       ?name     .
     ?center    ex:satellite  ?object   .
     ?center    ex:radius     ?radius   .
   FILTER ( langMATCHES( LANG(?name), "en") )
   FILTER ( 2 * ?radius > 3000 )
  }
```

- Objects with two or more satellites. Assume for this that different URIs denote different objects.

```
PREFIX ex: <http://example.org/> SELECT DISTINCT ?object WHERE
  { ?object      ex:satellite  ?satellite1 .
    ?object      ex:satellite  ?satellite2 .
    FILTER ( !sameTERM(?satellite1,?satellite2) )
  }
```

Aufgabe 4.2 Translate the queries from Exercise 4.1 into expressions in SPARQL algebra. You can simplify Join expressions with the empty graph Z as parameter.

```
Union(BGP(<http://example.org/Sun>
                        <http://example.org/satellite> ?object.),
      Join(BGP(<http://example.org/Sun>
                        <http://example.org/satellite> ?satellite.),
           BGP(?satellite <http://example.org/satellite> ?object.)
          )
     )

Filter((4/3 * 3.1416 * ?radius * ?radius * ?radius > 20000000000),
       LeftJoin(BGP(?object <http://example.org/radius> ?radius.),
                BGP(?center <http://example.org/satellite> ?object.),
                true
               )
      )

Filter(((langMATCHES( LANG(?name), "en")) && (2 * ?radius > 3000)),
       Join(Join(Join(BGP(?object
                        <http://example.org/satellite> ?satellite.),
```

```
                    BGP(?satellite
                        <http://example.org/name> ?name.)
                   ),
                BGP(?center <http://example.org/satellite> ?object.)
               ),
            BGP(?center <http://example.org/radius ?radius>)
           )
      )

Filter((!sameTERM(?satellite1,?satellite2)),
       Join(BGP(?object <http://example.org/satellite> ?satellite1.),
            BGP(?object <http://example.org/satellite> ?satellite2.),
           )
      )
```

Aufgabe 4.3 Compute the solutions to the expressions from Exercise 4.2 with respect to the knowledge base from Exercise 4.1 step by step. First query:

```
BGP(<http://example.org/Sun>
                  <http://example.org/satellite> ?satellite.)
```

satellite
ex:Mercury
ex:Venus
ex:Earth
ex:Mars

```
BGP(?satellite <http://example.org/satellite> ?object.)
```

satellite	object
ex:Sun	ex:Mercury
ex:Sun	ex:Venus
ex:Sun	ex:Earth
ex:Sun	ex:Mars
ex:Earth	ex:Moon
ex:Mars	ex:Phobos
ex:Mars	ex:Deimos

```
Join(BGP(<http://example.org/Sun>
                          <http://example.org/satellite> ?satellite.),
     BGP(?satellite <http://example.org/satellite> ?object.)
    )
```

satellite	object
ex:Earth	ex:Moon
ex:Mars	ex:Phobos
ex:Mars	ex:Deimos

```
BGP(<http://example.org/Sun>
                  <http://example.org/satellite> ?object.)
```

object
ex:Mercury
ex:Venus
ex:Earth
ex:Mars

```
Union(BGP(<http://example.org/Sun>
                          <http://example.org/satellite> ?object.),
      Join(BGP(<http://example.org/Sun>
                          <http://example.org/satellite> ?satellite.),
           BGP(?satellite <http://example.org/satellite> ?object.)
          )
     )
```

object
ex:Mercury
ex:Venus
ex:Earth
ex:Mars

satellite	object
ex:Earth	ex:Moon
ex:Mars	ex:Phobos
ex:Mars	ex:Deimos

Second query:

```
BGP(?object <http://example.org/radius> ?radius.)
```

object	radius
ex:Sun	"1.392e6"^^xsd:double
ex:Mercury	"2439.7"^^xsd:double
ex:Venus	"6051.9"^^xsd:double
ex:Earth	"6372.8"^^xsd:double
ex:Mars	"3402.5"^^xsd:double
ex:Moon	"1737.1"^^xsd:double

```
BGP(?center <http://example.org/satellite> ?object.)
```

center	object
ex:Sun	ex:Mercury
ex:Sun	ex:Venus
ex:Sun	ex:Earth
ex:Sun	ex:Mars
ex:Earth	ex:Moon
ex:Mars	ex:Phobos
ex:Mars	ex:Deimos

```
LeftJoin(BGP(?object <http://example.org/radius> ?radius.),
         BGP(?center <http://example.org/satellite> ?object.),
         true
        )
```

object	radius	center
ex:Sun	"1.392e6"^^xsd:double	
ex:Mercury	"2439.7"^^xsd:double	ex:Sun
ex:Venus	"6051.9"^^xsd:double	ex:Sun
ex:Earth	"6372.8"^^xsd:double	ex:Sun
ex:Mars	"3402.5"^^xsd:double	ex:Sun
ex:Moon	"1737.1"^^xsd:double	ex:Earth

```
Filter((4/3 * 3.1416 * ?radius * ?radius * ?radius > 20000000000),
       LeftJoin(BGP(?object <http://example.org/radius> ?radius.),
                BGP(?center <http://example.org/satellite> ?object.),
                true
               )
      )
```

object	radius	center
ex:Sun	"1.392e6"^^xsd:double	
ex:Mercury	"2439.7"^^xsd:double	ex:Sun
ex:Venus	"6051.9"^^xsd:double	ex:Sun
ex:Earth	"6372.8"^^xsd:double	ex:Sun
ex:Mars	"3402.5"^^xsd:double	ex:Sun
ex:Moon	"1737.1"^^xsd:double	ex:Earth

Third query – we omit the tables for the BGP expressions:

```
Join(BGP(?object <http://example.org/satellite> ?satellite.),
     BGP(?satellite <http://example.org/name> ?name.)
    )
```

object	satellite	name
ex:Earth	ex:Moon	"Moon@en"
ex:Mars	ex:Phobos	"Phobos"
ex:Mars	ex:Deimos	"Deimos"

```
Join(Join(BGP(?object <http://example.org/satellite> ?satellite.),
          BGP(?satellite <http://example.org/name> ?name.)
         ),
     BGP(?center <http://example.org/satellite> ?object.)
    )
```

center	object	satellite	name
ex:Sun	ex:Earth	ex:Moon	"Moon@en"
ex:Sun	ex:Mars	ex:Phobos	"Phobos"
ex:Sun	ex:Mars	ex:Deimos	"Deimos"

```
Join(Join(Join(BGP(?object <http://example.org/satellite>
?satellite.),
               BGP(?satellite <http://example.org/name> ?name.)
              ),
          BGP(?center <http://example.org/satellite> ?object.)
         ),
     BGP(?center <http://example.org/radius ?radius>)
    )
```

center	radius	object	satellite	name
ex:Sun	"1.392e6"^^xsd:double	ex:Earth	ex:Moon	"Moon@en"
ex:Sun	"1.392e6"^^xsd:double	ex:Mars	ex:Phobos	"Phobos"
ex:Sun	"1.392e6"^^xsd:double	ex:Mars	ex:Deimos	"Deimos"

```
Filter(((langMATCHES( LANG(?name), "en")) && (2 * ?radius > 3000)),
       Join(Join(Join(BGP(?object
                          <http://example.org/satellite> ?satellite.),
                      BGP(?satellite
                          <http://example.org/name> ?name.)
                     ),
                 BGP(?center <http://example.org/satellite> ?object.)
                ),
            BGP(?center <http://example.org/radius ?radius>)
           )
      )
```

object	satellite1	satellite2
ex:Sun	ex:Mercury	ex:Mercury
ex:Sun	ex:Mercury	ex:Venus
ex:Sun	ex:Mercury	ex:Earth
ex:Sun	ex:Mercury	ex:Mars
ex:Sun	ex:Venus	ex:Mercury
ex:Sun	ex:Venus	ex:Venus
ex:Sun	ex:Venus	ex:Earth
ex:Sun	ex:Venus	ex:Mars
ex:Sun	ex:Earth	ex:Mercury
ex:Sun	ex:Earth	ex:Venus
ex:Sun	ex:Earth	ex:Earth
ex:Sun	ex:Earth	ex:Mars
ex:Sun	ex:Mars	ex:Mercury
ex:Sun	ex:Mars	ex:Venus
ex:Sun	ex:Mars	ex:Earth
ex:Sun	ex:Mars	ex:Mars
ex:Earth	ex:Moon	ex:Moon
ex:Mars	ex:Phobos	ex:Phobos
ex:Mars	ex:Phobos	ex:Deimos
ex:Mars	ex:Deimos	ex:Phobos
ex:Mars	ex:Deimos	ex:Deimos

Abbildung 1: Exercise 4.3 fourth query: Table for `Join(BGP(
?object <http://example.org/satellite> ?satellite1.), BGP(?object
<http://example.org/satellite> ?satellite2.))`

center	radius	object	satellite	name
ex:Sun	"1.392e6"^^xsd:double	ex:Earth	ex:Moon	"Moon@en"

Fourth query – we omit the tables for the BGP expressions:

```
Join(BGP(?object <http://example.org/satellite> ?satellite1.),
     BGP(?object <http://example.org/satellite> ?satellite2.)
    )
```

See Fig. 1.

```
Filter((!sameTERM(?satellite1,?satellite2)),
       Join(BGP(?object <http://example.org/satellite> ?satellite1.),
            BGP(?object <http://example.org/satellite> ?satellite2.)
```

object	satellite1	satellite2
ex:Sun	ex:Mercury	ex:Venus
ex:Sun	ex:Mercury	ex:Earth
ex:Sun	ex:Mercury	ex:Mars
ex:Sun	ex:Venus	ex:Mercury
ex:Sun	ex:Venus	ex:Earth
ex:Sun	ex:Venus	ex:Mars
ex:Sun	ex:Earth	ex:Mercury
ex:Sun	ex:Earth	ex:Venus
ex:Sun	ex:Earth	ex:Mars
ex:Sun	ex:Mars	ex:Mercury
ex:Sun	ex:Mars	ex:Venus
ex:Sun	ex:Mars	ex:Earth
ex:Mars	ex:Phobos	ex:Deimos
ex:Mars	ex:Deimos	ex:Phobos

Abbildung 2: Exercise 4.3 fourth query: Table for
`Filter((!sameTERM(?satellite1,?satellite2)), Join(BGP(?object <http://example.org/satellite> ?satellite1.), BGP(?object <http://example.org/satellite> ?satellite2.) ) )`

```
            )
      )
```

See Fig. 2.

Aufgabe 4.4 It is possible to use SPARQL for searching for elements for which certain information is *not* given. This is done by combining filters with optional graph patterns.

Formulate a query which asks for all celestial bodies which do not have a satellite. Assume for this that the knowledge base from Exercise 4.1 has been completed with triples which assign to every celestial body the `rdf:type CelestialBody`.

```
PREFIX  ex:  <http://example.org/>
 SELECT ?object
 WHERE
   {         { ?object  rdf:type      ex:CelestialBody }
    OPTIONAL { ?object  ex:satellite  ?satellite       }
    FILTER ( !BOUND(?satellite) )
   }
```

Aufgabe 4.5 The game *Sudoku* is about completing incomplete tables with numbers while respecting certain rules. We consider the following simple 4×4 Sudoku:

			3
			4
2			
3			

You have to fill in numbers with values 1 to 4 in the empty slots in the table so that no number occurs twice in any row or any column, and so that no number is duplicated within any of the marked 2×2 squares.

We now want to use SPARQL for solving this Sudoku, i.e. we want to obtain all possible solutions by means of answers to a SPARQL query. In order to do this, set up a suitable RDF document and SPARQL query.

We need an RDF document because SPARQL variables can only obtain values which occur in the queried RDF document. So let's simply use the following.

```
<http://example.org/square>   <http://example.org/allowed>
            "1"^^xsd:int, "2"^^xsd:int, "3"^^xsd:int, "4"^^xsd:int .
```

We now assign to each of the squares a variable name, say from `?F11` to `?F44`. Finally, we need to formulate all conditions which constrain possible solutions:

- Every variable gets assigned one of the allowed numbers.
- Variables standing for table slots which are already filled in must get the corresponding value assigned.
- No two variables within the same row get assigned the same value.
- No two variables within the same column get assigned the same value.
- No two variables within the same marked 2×2 square get assigned the same value.

A possible solution now looks as follows.

```
PREFIX  ex:  <http://example.org/> SELECT  ?F11 ?F12 ?F13 ?F14
        ?F21 ?F22 ?F23 ?F24
        ?F31 ?F32 ?F33 ?F34
        ?F41 ?F42 ?F43 ?F44
WHERE {  ex:square ex:allowed ?F11, ?F12, ?F13, ?F14,
                        ?F21, ?F22, ?F23, ?F24,
                        ?F31, ?F32, ?F33, ?F34,
                        ?F41, ?F42, ?F43, ?F44.
```

```
FILTER ( ?F14 = "3"^^xsd:int )
FILTER ( ?F24 = "4"^^xsd:int )
FILTER ( ?F31 = "2"^^xsd:int )
FILTER ( ?F41 = "3"^^xsd:int )


FILTER ( ?F11 != ?F12 )  FILTER ( ?F11 != ?F13 )
FILTER ( ?F11 != ?F14 )  FILTER ( ?F12 != ?F13 )
FILTER ( ?F12 != ?F14 )  FILTER ( ?F13 != ?F14 )

FILTER ( ?F21 != ?F22 )  FILTER ( ?F21 != ?F23 )
FILTER ( ?F21 != ?F24 )  FILTER ( ?F22 != ?F23 )
FILTER ( ?F22 != ?F24 )  FILTER ( ?F23 != ?F24 )

FILTER ( ?F31 != ?F32 )  FILTER ( ?F31 != ?F33 )
FILTER ( ?F31 != ?F34 )  FILTER ( ?F32 != ?F33 )
FILTER ( ?F32 != ?F34 )  FILTER ( ?F33 != ?F34 )

FILTER ( ?F41 != ?F42 )  FILTER ( ?F41 != ?F43 )
FILTER ( ?F41 != ?F44 )  FILTER ( ?F42 != ?F43 )
FILTER ( ?F42 != ?F44 )  FILTER ( ?F43 != ?F44 )


FILTER ( ?F11 != ?F21 )  FILTER ( ?F11 != ?F31 )
FILTER ( ?F11 != ?F41 )  FILTER ( ?F21 != ?F31 )
FILTER ( ?F21 != ?F41 )  FILTER ( ?F31 != ?F41 )

FILTER ( ?F12 != ?F22 )  FILTER ( ?F12 != ?F32 )
FILTER ( ?F12 != ?F42 )  FILTER ( ?F22 != ?F32 )
FILTER ( ?F22 != ?F42 )  FILTER ( ?F32 != ?F42 )

FILTER ( ?F13 != ?F23 )  FILTER ( ?F13 != ?F33 )
FILTER ( ?F13 != ?F43 )  FILTER ( ?F23 != ?F33 )
FILTER ( ?F23 != ?F43 )  FILTER ( ?F33 != ?F43 )

FILTER ( ?F14 != ?F24 )  FILTER ( ?F14 != ?F34 )
FILTER ( ?F14 != ?F44 )  FILTER ( ?F24 != ?F34 )
FILTER ( ?F24 != ?F44 )  FILTER ( ?F34 != ?F44 )


FILTER ( ?F11 != ?F22 )  FILTER ( ?F12 != ?F21 )
FILTER ( ?F13 != ?F24 )  FILTER ( ?F14 != ?F23 )
FILTER ( ?F31 != ?F42 )  FILTER ( ?F32 != ?F41 )
```

```
  FILTER ( ?F33 != ?F44 )  FILTER ( ?F34 != ?F43 )
}
```

It is obviously possible to solve larger Sudokus along the same lines. This actually shows that SPARQL is at least as hard as Sudoku, which is known to be *NP*-complete.