Exercise Sheet 10: Cypher

Maximilian Marx, Markus Krötzsch

Knowledge Graphs, 2021-01-26, Winter Term 2020/2021

Exercise 10.1. Consider the following cypher query from example 11.17:

```
MATCH (prof {occupation: "Professor" })-[:SPOUSE]-()
MATCH (prof)-[:HAS_CHILD]->(child)
RETURN prof, count(child)
```

Do the query results change if count(DISTINCT child) is used instead?

Exercise 10.2. Which of the following graph patterns are expressible in Cypher? Explain your answer by either giving a Cypher query or by arguing why there is none.

1. Find nodes that are connected by an `:EDGE` path of length ≥ 100
2. Find nodes that are connected by an `:EDGE` path of length ≤ 100
3. Find nodes that are connected by an `:EDGE` path of length $\neq 100$
4. Find nodes that are not connected by an `:EDGE` path of length 100
5. In a graph with a `:PARENT` relationship type, find nodes with a common ancestor
6. In a graph with a `:PARENT` relationship type, find nodes that are cousins (of any degree)
7. Find nodes that are connected by `:PROP_A` but not by `:PROP_B`
8. Find nodes that are connected by a `:PROP_A` path, but not by a `:PROP_B` path
9. Find nodes that are connected by a path of nodes as in 7.
10. Find nodes connected by an arbitrary path
11. Find nodes connected by an arbitrary path of even length
12. Check if the graph contains an even number of nodes

Exercise 10.3. Neo4j provides numerous extension over the openCypher language, including the list predicate functions `all`[1] and `any`[2], that check whether a condition is true for all elements (or any element, respectively) of a list.

Show that these two functions are sufficient to encode **TrueQBF** in a Cypher query. What can you say about the complexity of answering Cypher queries?

[1]https://neo4j.com/docs/cypher-manual/current/functions/predicate/#functions-all
[2]https://neo4j.com/docs/cypher-manual/current/functions/predicate/#functions-any

Exercise 10.4. Wikidata Property Constraints[3] are a mechanism to specify how properties should be used on Wikidata. As an example, an Inverse Constraint[4] specifies that every statement for a given property must have a matching statement in the reverse direction using some other property (e.g., every "mother" statement must have a matching "child" statement).

Use the Rulewerk client[3] and the Wikidata Query Service[5] to find statements violating an Inverse Constraint:

- write a SPARQL query to find all Inverse Constraints and the related properties,
- write a SPARQL query that finds violating statements for a given pair of forward and inverse properties,
- write a rules program that combines these two SPARQL data sources to obtain all statements violating Inverse Constraints.

Hint: Finding all violations for all inverse constraints might take a long time. For testing, limit your queries to, e.g., 10 pairs of properties. To achieve that for Rulewerk data sources, note that you can nest a subquery inside a graph pattern.

[3]https://www.wikidata.org/wiki/Help:Property_constraints_portal
[4]https://www.wikidata.org/wiki/Help:Property_constraints_portal/Inverse
[3]https://github.com/knowsys/rulewerk/wiki/Standalone-client
[5]https://query.wikidata.org