Exercise Sheet 5: Datalog in Theory & Practice

Maximilian Marx, Markus Krötzsch

Knowledge Graphs, 2025-12-03, Winter Term 2025/2026

Exercise 5.1. Consider the Datalog program P

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

and the facts

$$\begin{aligned}
\mathsf{mother}(\mathsf{alice}, \mathsf{barbara}) \qquad & \qquad \mathsf{father}(\mathsf{alice}, \mathsf{bob}) \\
\mathsf{mother}(\mathsf{barbara}, \mathsf{christine}) \qquad & \qquad \mathsf{father}(\mathsf{barbara}, \mathsf{charles}) \\
\mathsf{mother}(\mathsf{dave}, \mathsf{emmy}) \qquad & \qquad \mathsf{father}(\mathsf{bob}, \mathsf{dave})
\end{aligned}$$

Compute all query results for $\langle P, \mathsf{Result} \rangle$.

Exercise 5.2. Show that D_P^∞ is the least fixed point of the T_P operator.

1. Show that it is a fixed point, i.e., that $T_P(D_P^\infty) = D_P^\infty$.
2. Show that every fixed point of T_P must contain every fact in D_P^∞.

Exercise 5.3. Consider databases that encode a labelled, directed graph by means of a ternary EDB predicate e (“edge”). The two parameters are the source and target nodes of the edge, while the third parameter is its label. For example, the edge $n_1 \xrightarrow{a} n_2$ would be represented by the fact $e(n_1, n_2, a)$. Moreover, assume that only constants a and b are used as labels.

Can you express the following queries using Datalog?

1. “Which nodes in the graph are reachable from the node n?”
2. “Are all nodes of the graph reachable from the node n?”
3. “Does the graph have a directed cycle?”
4. “Does the graph have a path that is labelled by a palindrome?”
   (a palindrome is a word that reads the same forwards and backwards)
5. “Is the connected component that contains the node n 2-colourable?”
6. “Is the graph 2-colourable?”
7. “Which pairs of nodes are connected by a path with an even number of a labels?”
8. “Which pairs of nodes are connected by a path with the same number of a and b labels?”
9. “Is there a pair of nodes that is connected by two distinct paths?”

Exercise 5.4. Solve Exercise 2.5 using Nemo[1]: Write a Datalog program extracting, from the `coauthors.nt.gz` graph[2], the *connected component* containing <http://dblp.uni-trier.de/pers/s/Studer:Rudi>, i.e, extract the induced subgraph that

- contains <http://dblp.uni-trier.de/pers/s/Studer:Rudi>,
- contains all nodes reachable from <http://dblp.uni-trier.de/pers/s/Studer:Rudi> by some path, and
- contains all edges that are present in the full graph between these nodes.

Note that, while an RDF graph is inherently directed, edges in `coauthors.nt.gz` are symmetric, i.e., the graph is essentially undirected.
Hint: `authorship-snippet.nt.gz`[2] contains <http://dblp.uni-trier.de/pers/s/Studer:Rudi> and can be used for testing.
Hint: You can use, e.g., https://media.githubusercontent.com/media/knowsys/Course-Knowledge-Graphs/refs/heads/main/data/dblp/authorship-snippet.nt.gz as the resource in the Nemo web UI to avoid downloading the graph onto your computer.

Exercise 5.5. DBpedia is a knowledge graph based on information extracted from Wikipedia. Use Nemo[1] and the Wikidata[??] and DBpedia[3] SPARQL endpoints to integrate and compare the *parent* relations from DBpedia and Wikidata: Use SPARQL queries to fetch both *parent* relationships (for Wikidata, you can restrict to items with articles on English Wikipedia). Make sure your queries include a common feature that can be used for integration, e.g., the URL of the related Wikipedia article.[4] Lastly, use rules to compute the total number of (unique) relations found in both graphs, in Wikidata only, and in DBpedia only.

[1] https://tools.iccl.inf.tu-dresden.de/nemo/next
[2] https://github.com/knowsys/Course-Knowledge-Graphs/tree/main/data/dblp
[3] https://dbpedia.org/sparql
[4] DBpedia still stores `http` URLs, whereas Wikidata uses `https`. You can use `SUBSTR` and `BIND` in your SPARQL queries to align such URLs.