Exercise Sheet 2: Graphs and RDF

Maximilian Marx, Markus Krötzsch

Knowledge Graphs, 2025-11-05, Winter Term 2025/2026

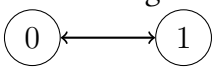
Exercise 2.1. Which of the following literals describe the same value? Explain your answer.

1. `"2"^^xsd:integer` vs. `"2.0"^^xsd:decimal`
2. `"2"^^xsd:decimal` vs. `"2"^^xsd:float`
3. `"2018-11-06T15:40:00+01:00"^^xsd:dateTime` vs. `"2018-11-06T14:40:00Z"^^xsd:dateTime`
4. `"2018-11-06T15:40:00+01:00"^^xsd:dateTime` vs. `"2018-11-06T14:40:00"^^xsd:dateTime`

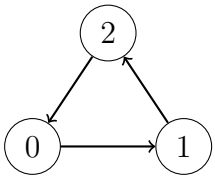
A detailed description of each of the various XML Schema datatypes is given in the online specification: see https://www.w3.org/TR/xmlschema11-2/.

Exercise 2.2. A *graph homomorphism* from a directed graph $G = \langle V, E \rangle$ to a directed graph $G' = \langle V', E' \rangle$ is a map $h : V \to V'$ such that for every edge $\langle a, b \rangle \in E$, we also have $\langle h(a), h(b) \rangle \in E'$, i. e., such that adjacent vertices are mapped to adjacent vertices. A *strong homomorphism* furthermore requires that if $\langle h(a), h(b) \rangle \in E'$, then we already have $\langle a, b \rangle \in E$, i. e., only adjacent vertices are mapped to adjacent vertices.

For each of the following graphs, determine if there is a homomorphism from the graph to the graph $G' = \langle V', E' \rangle$ with $V' = \{ 0, 1 \}$ and $E' = \{ \langle 0, 1 \rangle, \langle 1, 0 \rangle \}$. Which graphs also admit a strong homomorphism to G'?



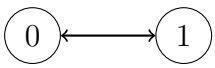
1. $G_1 = \langle V_1, E_1 \rangle$ with $V_1 = \{ 0, 1, 2 \}$ and $E_1 = \{ \langle 0, 1 \rangle, \langle 1, 2 \rangle, \langle 2, 0 \rangle \}$



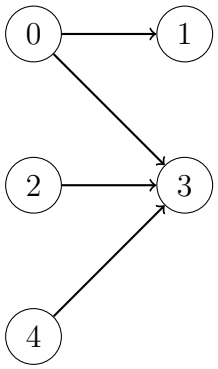
2. $G_2 = \langle V_2, E_2 \rangle$ with $V_2 = \{ 0, 1 \}$ and $E_2 = \varnothing$

3. $G_3 = \langle V_3, E_3 \rangle$ with $V_3 = \{ 0 \}$ and $E_3 = \{ \langle 0, 0 \rangle \}$

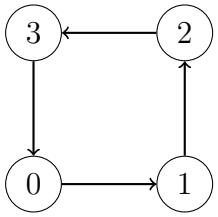
4. $G_4 = \langle V_4, E_4 \rangle$ with $V_4 = \{ 0, 1 \}$ and $E_4 = \{ \langle 0, 1 \rangle, \langle 1, 0 \rangle \}$



5. $G_5 = \langle V_5, E_5 \rangle$ with $V_5 = \{ 0, 1, 2, 3, 4 \}$ and $E_5 = \{ \langle 0, 1 \rangle, \langle 0, 3 \rangle, \langle 2, 3 \rangle, \langle 4, 3 \rangle \}$.



6. $G_6 = \langle V_6, E_6 \rangle$ with $V_4 = \{ 0, 1, 2, 3 \}$ and $E_6 = \{ \langle 0, 1 \rangle, \langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 4 \rangle \}$



Exercise 2.3. Recall that blank nodes act as placeholders for arbitrary resources in RDF: they assert that there is something without saying what it is. Such an assertion might logically follow from other, stronger assertions, so that some triples in a graph might be redundant. For example, the second triple in the following dataset can be omitted without loss of information:

```
eg:s   eg:p   eg:o .
_:1    eg:p   _:2  .
```

More generally, an *instance* of an RDF graph G is a graph $\sigma(G)$ obtained by applying a function σ that maps blank nodes to arbitrary RDF terms. A graph is *lean* if it does not have any instance $\sigma(G) \subset G$ that is strictly contained in G. In the example, $\sigma = \{$`_:1` $\mapsto$ `<s>`, `_:2` $\mapsto$ `<o>`$\}$ shows that this graph is not lean.

Determine if the following graphs are lean:

(a)
```
eg:s   eg:p   eg:o .
_:1    eg:p   _:1  .
```

(b)
```
eg:s   eg:p   _:2  .
_:1    eg:p   eg:o .
```

(c)
```
eg:s   eg:p   eg:o .
_:1    eg:p   [ eg:p   [] ] .
```

(d)
```
eg:s   eg:p   eg:s .
_:1    eg:p   [ eg:p   [] ] .
```

∗ **Exercise 2.4.** Show that it is NP-complete to decide if an RDF graph is not lean.

Hint:
For hardness, find a reduction from 3-colorability. Making an RDF graph non-lean if a graph is colorable is not hard. Making it lean if it is not colorable requires some trick to prevent the encoded graph from embedding into itself.

Exercise 2.5. Write a program that reads a graph in N-Triples format and checks whether the graph is bipartite. Use it to decide whether `authorship.nt.gz`[1] and `coauthors.nt.gz`[1] are bipartite.

Hint: each of the uncompressed graphs is roughly 4 GiB in size. In Python, you can use `gzip.GzipFile`[2] to process the compressed file without decompressing it first. There is also `authorship-snippet.nt.gz`[1], a small part of the graph that you can use during development.

Please note: In order to get the correct data files, please install git-lfs[3] on your system, and then activate it in your local repository (git lfs install).

Exercise 2.6. The bibliographic database DBLP[4] offers individual data records as RDF in N-Triples format. This data can be downloaded from the URL obtained by appending `.nt` to the URI. Use this interface to find all publications that have https://dblp.org/pid/s/RudiStuder.html as their only author.

- Download some RDF files in your browser to find out how this information is encoded.
- Write a program that crawls a small part of the data to answer the query.

Note: If your program sends too many requests in a short time, the server will deny the request and cancel the connection. Dirty trick: use `time.sleep(1)` before executing a request.

Hint: `requests`[5] provides a high-level API for making HTTP requests in Python, but you may need to install it, e.g., using `pip`.[6] A built-in alternative that provides a lower-level interface is `urllib.requests`.[7]

[1] https://github.com/knowsys/Course-Knowledge-Graphs/tree/main/data/dblp
[2] https://docs.python.org/3/library/gzip.html
[3] https://git-lfs.github.com/
[4] https://dblp.org
[5] https://requests.readthedocs.io/en/latest/
[6] https://pypi.org/project/pip/
[7] https://docs.python.org/3/library/urllib.request.html