Graph Database Systems – Two Categories

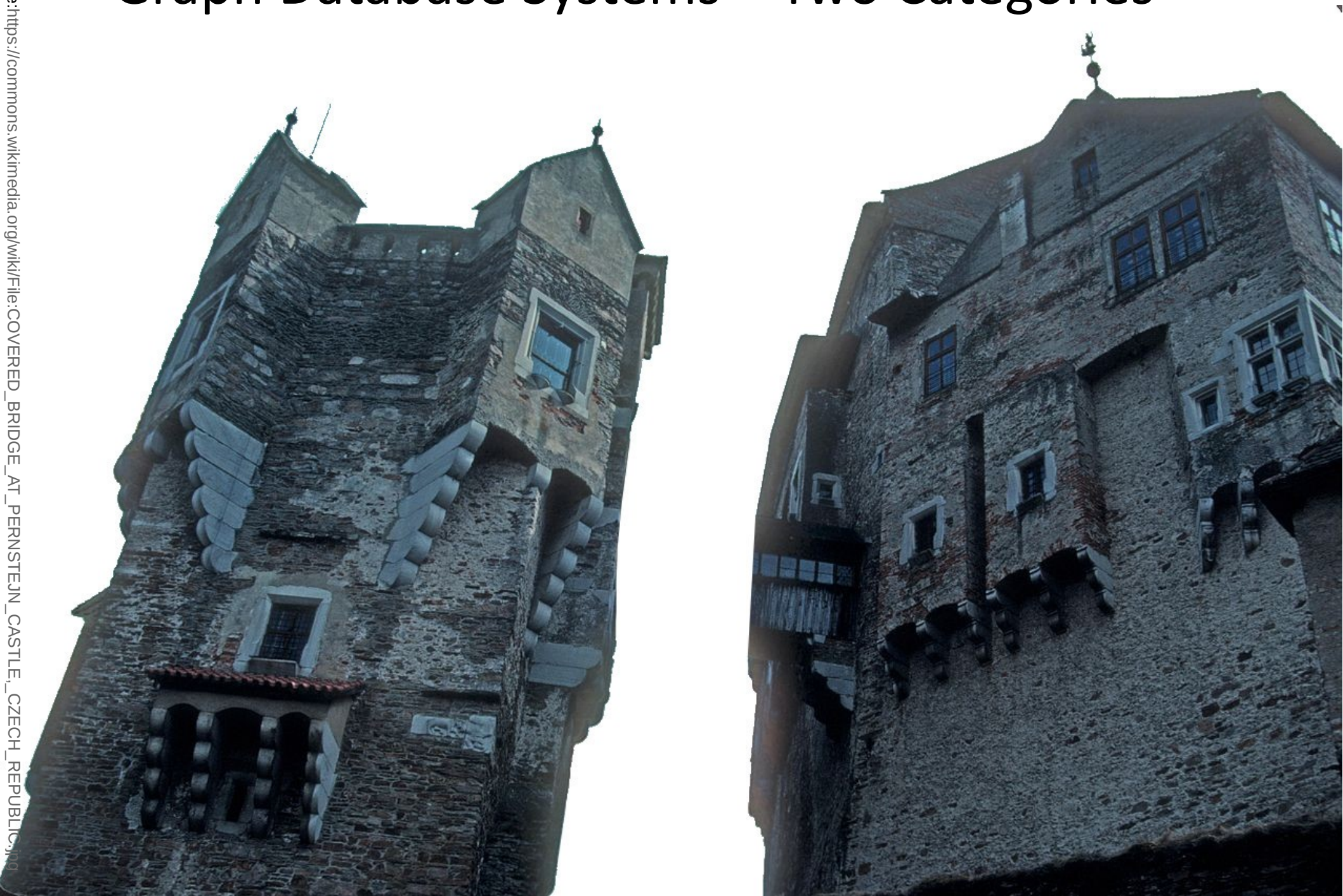


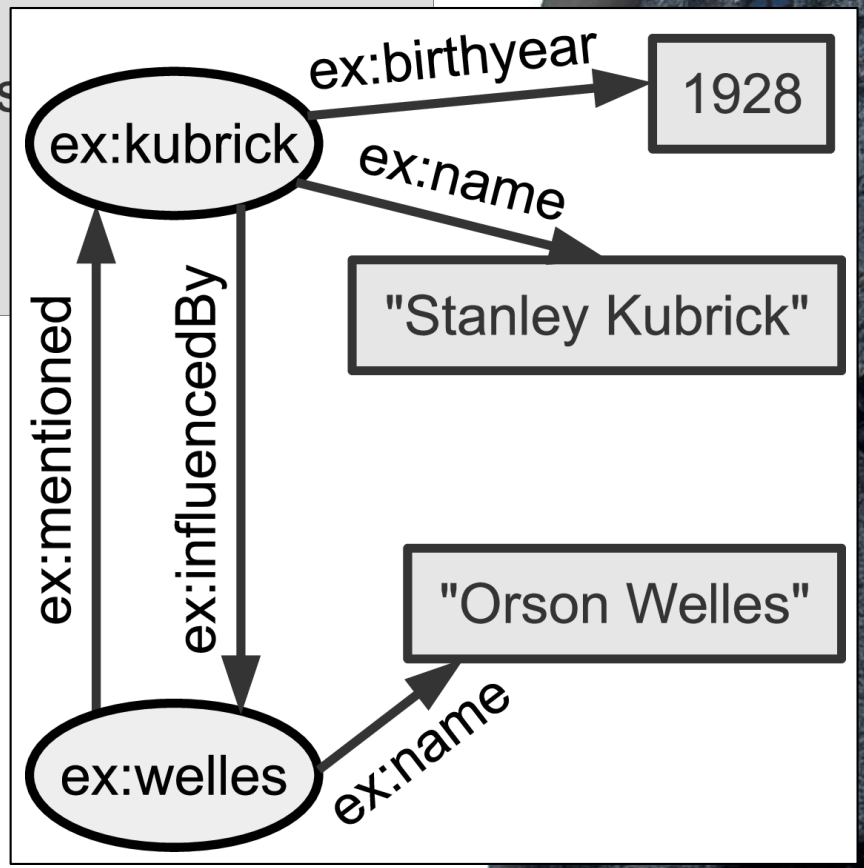
Image source:https://commons.wikimedia.org/wiki/File:COVERED_BRIDGE_AT_PERNSTEJN_CASTLE,_CZECH_REPUBLIC.jpg

- Triplestores support RDF and SPARQL

```
@prefix ex: <http://www.example.org/> .

ex:kubrick  ex:name  "Stanley Kubrick" .
ex:kubrick  ex:birthyear  1928 .
ex:kubrick  ex:influencedBy  ex:welles .

ex:welles  ex:name  "Ors
ex:welles  ex:mentioned
```

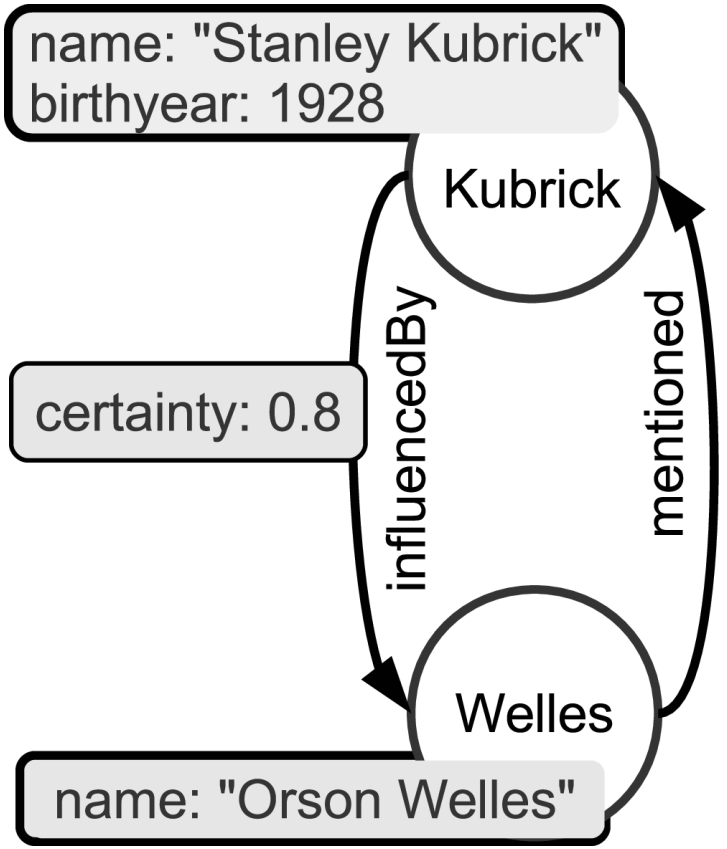


MarkLogic GraphDB AllegroGraph Virtuoso Universal Server blazegraph ANZO

Image source:https://commons.wikimedia.org/wiki/File:COVERED_BRIDGE_AT_PERNSTEJN_CASTLE,_CZECH_REPUBLIC.jpg

Image source:https://commons.wikimedia.org/wiki/File:COVERED_BRIDGE_AT_PERNSTEJN_CASTLE,_CZECH_REPUBLIC.jpg

- Support of so-called Property Graphs (PGs) or Labeled PGs (LPGs)



JanusGraph
INFINITEGRAPH
TigerGraph
neo4j
OrientDB
ArangoDB
Apache TinkerPop

Coexistence!

- Too much has been invested for any one of them to get abandoned now
- Both have merits and a user base

Image source:https://commons.wikimedia.org/wiki/File:COVERED_BRIDGE_AT_PERNSTEJN_CASTLE,_CZECH_REPUBLIC.jpg

Interoperability

Image source:https://commons.wikimedia.org/wiki/File:COVERED_BRIDGE_AT_PERNSTEJN_CASTLE,_CZECH_REPUBLIC.jpg

We need well-defined approaches!

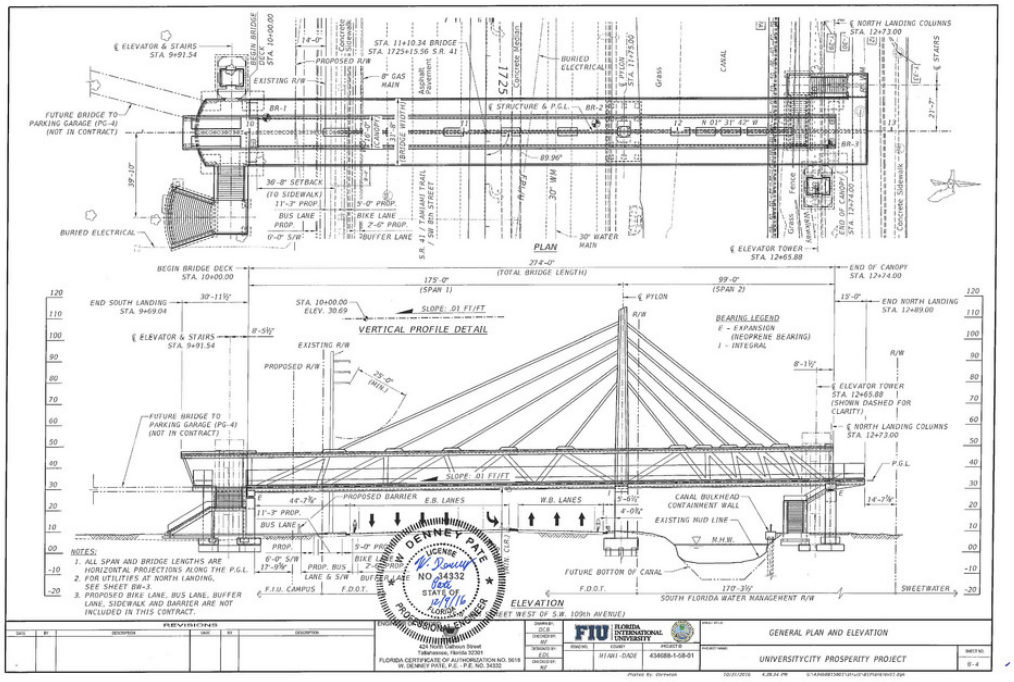


Image sources:
https://www.publicdomainpictures.net/en/view-image.php?image=253435 and https://www.flickr.com/photos/ntsb/40210711664

Foundations to Query Labeled Property Graphs using SPARQL*

Olaf Hartig

@olafhartig

LINKÖPING UNIVERSITY

Differences between LPGs and RDF Graphs

- Node and edge identifiers in an LPG are local to that LPG, whereas URIs are globally unique identifiers
  - important for data integration
- Edge labels cannot appear as nodes in an LPG, whereas in RDF we may have (s, p, o) and (p, p2, o2)
  - important for making the semantics of data explicit within the data itself
- No multivalued properties in an LPG, whereas in RDF we may have (s, p, literal1) and (s, p, literal2)
  - unless we use collection objects as values of properties in LPGs
- RDF graphs are not truly multigraphs, LPGs are (i.e., two edges between the same nodes may have the same label)
- No edge properties in RDF graphs!

Differences between LPGs and RDF Graphs

- Node and edge identifiers in an LPG are local to that LPG, whereas URIs are globally unique identifiers
  - important for data integration
- Edge labels cannot appear as nodes in an LPG, whereas in RDF we may have (s, p, o) and (p, p2, o2)
  - important for making the semantics of data explicit within the data itself
- No multivalued properties in an LPG, whereas in RDF we may have (s, p, literal1) and (s, p, literal2)
  - unless we use collection objects as values of properties in LPGs
- RDF graphs are not truly multigraphs, LPGs are (i.e., two edges between the same nodes may have the same label)
- No edge properties in RDF graphs!

Differences between LPGs and RDF Graphs

- Node and edge identifiers in an LPG are local to that LPG, whereas URIs are globally unique identifiers
  - important for data integration
- Edge labels cannot appear as nodes in an LPG, whereas in RDF we may have (s, p, o) and (p, p2, o2)
  - important for making the semantics of data explicit within the data itself
- No multivalued properties in an LPG, whereas in RDF we may have (s, p, literal1) and (s, p, literal2)
  - unless we use collection objects as values of properties in LPGs
- RDF graphs are not truly multigraphs, LPGs are (i.e., two edges between the same nodes may have the same label)
- No edge properties in RDF graphs!

Differences between LPGs and RDF Graphs

- Node and edge identifiers in an LPG are local to that LPG, whereas URIs are globally unique identifiers
  - important for data integration
- Edge labels cannot appear as nodes in an LPG, whereas in RDF we may have (s, p, o) and (p, p2, o2)
  - important for making the semantics of data explicit within the data itself
- No multivalued properties in an LPG, whereas in RDF we may have (s, p, literal1) and (s, p, literal2)
  - unless we use collection objects as values of properties in LPGs
- RDF graphs are not truly multigraphs, LPGs are (i.e., two edges between the same nodes may have the same label)
- No edge properties in RDF graphs!

Differences between LPGs and RDF Graphs

- Node and edge identifiers in an LPG are local to that LPG, whereas URIs are globally unique identifiers
  - important for data integration
- Edge labels cannot appear as nodes in an LPG, whereas in RDF we may have (s, p, o) and (p, p2, o2)
  - important for making the semantics of data explicit within the data itself
- No multivalued properties in an LPG, whereas in RDF we may have (s, p, literal1) and (s, p, literal2)
  - unless we use collection objects as values of properties in LPGs
- RDF graphs are not truly multigraphs, LPGs are (i.e., two edges between the same nodes may have the same label)
- No edge properties in RDF graphs!

RDF*/SPARQL* Approach [1,2]

- Annotations of individual triples (i.e., like edge properties in LPGs)
- Idea: nest triples into one another; similarly for query patterns

```
…
ex:kubrick  ex:name  "Stanley Kubrick" .
ex:kubrick  ex:birthyear  1928 .
<<ex:kubrick  ex:influencedBy  ex:welles>>  ex:certainty  0.8 .
…
```

```
SELECT ?i ?c WHERE {
   << ex:kubrick  ex:influencedBy  ?i >> ex:certainty  ?c  }
```

- Already supported by two commercial triplestores
- Other vendors are working on it (incl. Amazon Web Services)
- Collaboration to bring this approach to standardization by W3C

[1] Olaf Hartig and Bryan Thompson: "**Foundations of an Alternative Approach to Reification in RDF.**" CoRR abs/1406.3399, 2014.

[2] Olaf Hartig: "**Foundations of RDF* and SPARQL* - An Alternative Approach to Statement-Level Metadata in RDF.**" In Proc. of AMW 2017.

RDF*/SPARQL* Approach [1,2]

- Annotations of individual triples (i.e., like edge properties in LPGs)
- Idea: nest triples into one another; similarly for query patterns

```
…
ex:kubrick  ex:na   Stanle          " .
e           ex:h
<                                 y 0.8 .
```

```
         edBy           ex:certainty  ?c  }
```

Tomorrow! 12:15, Session 2.4 in Hall 4

Paper that adopts this idea for RDF Stream Processing [2]

- Alre   ppor   by two commercial triplestores
- Oth   vendors are working on it (incl. Amazon Web Services)
- Collaboration to bring this approach to standardization by W3C

ANZO blazegraph

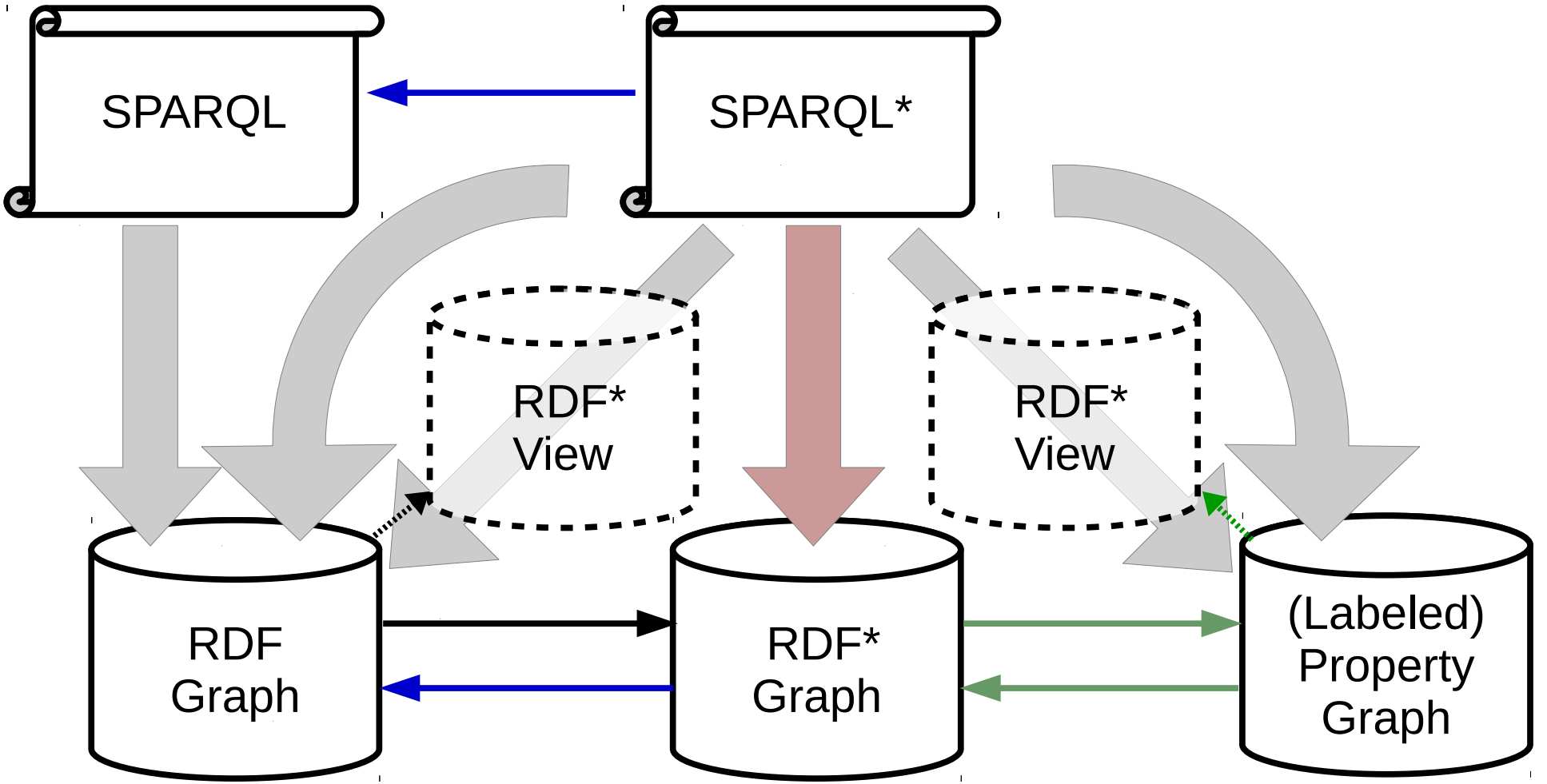
[1] Olaf Hartig and Bryan Thompson: "**Foundations of an Alternative Approach to Reification in RDF.**" CoRR abs/1406.3399, 2014.

[2] Olaf Hartig: "**Foundations of RDF* and SPARQL* - An Alternative Approach to Statement-Level Metadata in RDF.**" In Proc. of AMW 2017.

[3] Robin Keskisärkkä, Eva Blomqvist, Leili Lind, and Olaf Hartig: "**RSP-QL*: Enabling Statement-Level Annotations in RDF Streams.**" In Proc. of SEMANTiCS 2019.

LINKÖPING UNIVERSITY

An Approach to Interoperability

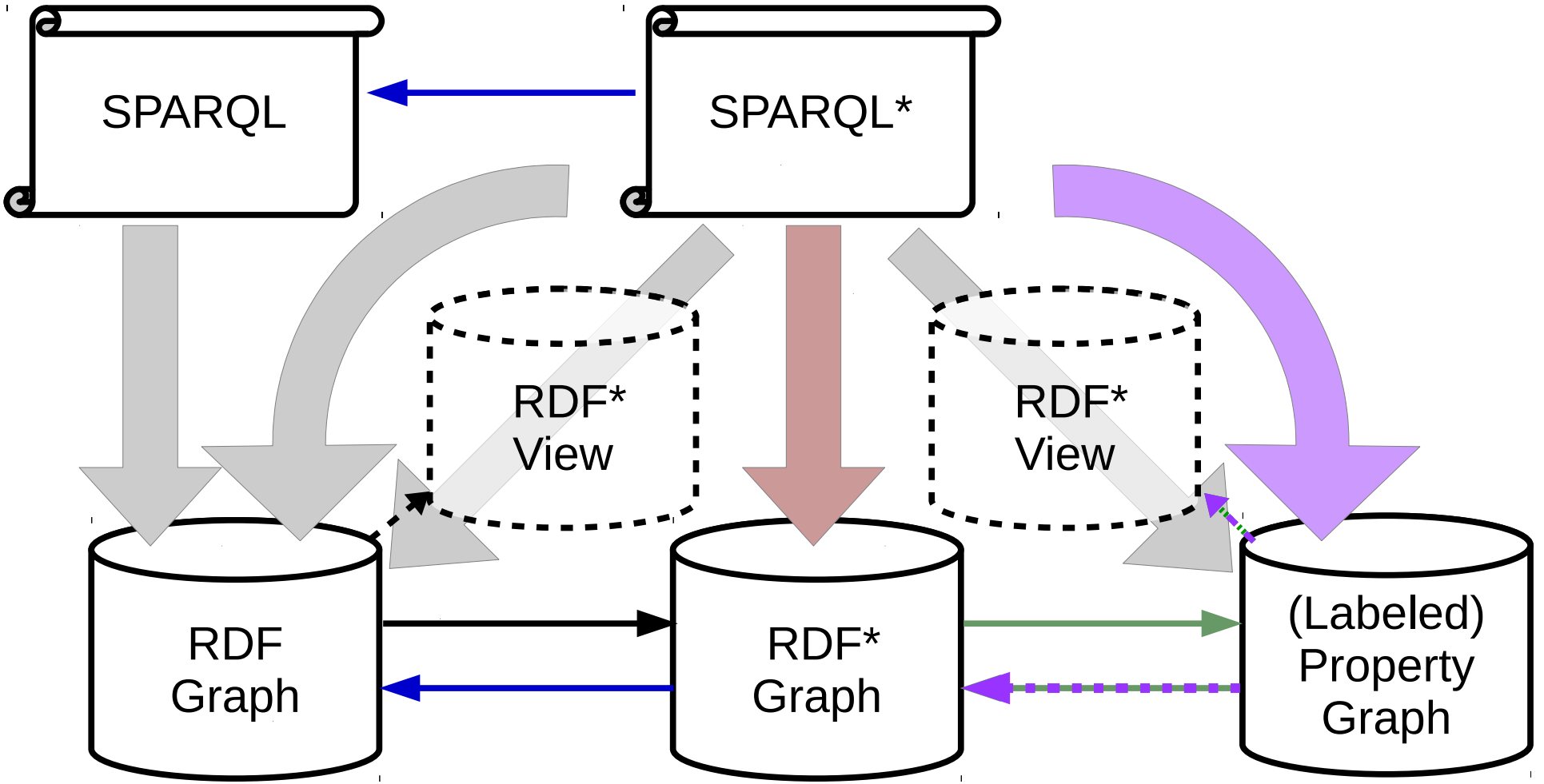


[1] Olaf Hartig and Bryan Thompson: "**Foundations of an Alternative Approach to Reification in RDF.**" CoRR abs/1406.3399, 2014.

[2] Olaf Hartig: "**Foundations of RDF* and SPARQL* - An Alternative Approach to Statement-Level Metadata in RDF.**" In Proc. of AMW 2017.

[4] Olaf Hartig: "**Reconciliation of RDF* and Property Graphs.**" In abs/1409.3288, 2014

This Paper

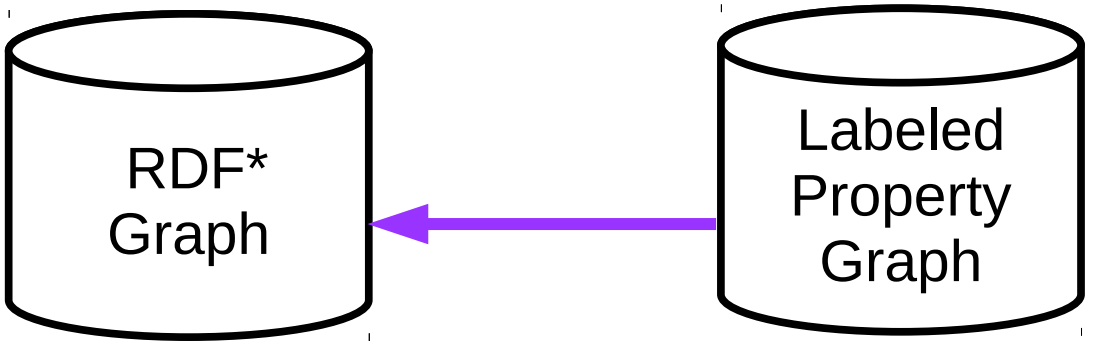


[1] Olaf Hartig and Bryan Thompson: "**Foundations of an Alternative Approach to Reification in RDF.**" CoRR abs/1406.3399, 2014.

[2] Olaf Hartig: "**Foundations of RDF* and SPARQL* - An Alternative Approach to Statement-Level Metadata in RDF.**" In Proc. of AMW 2017.

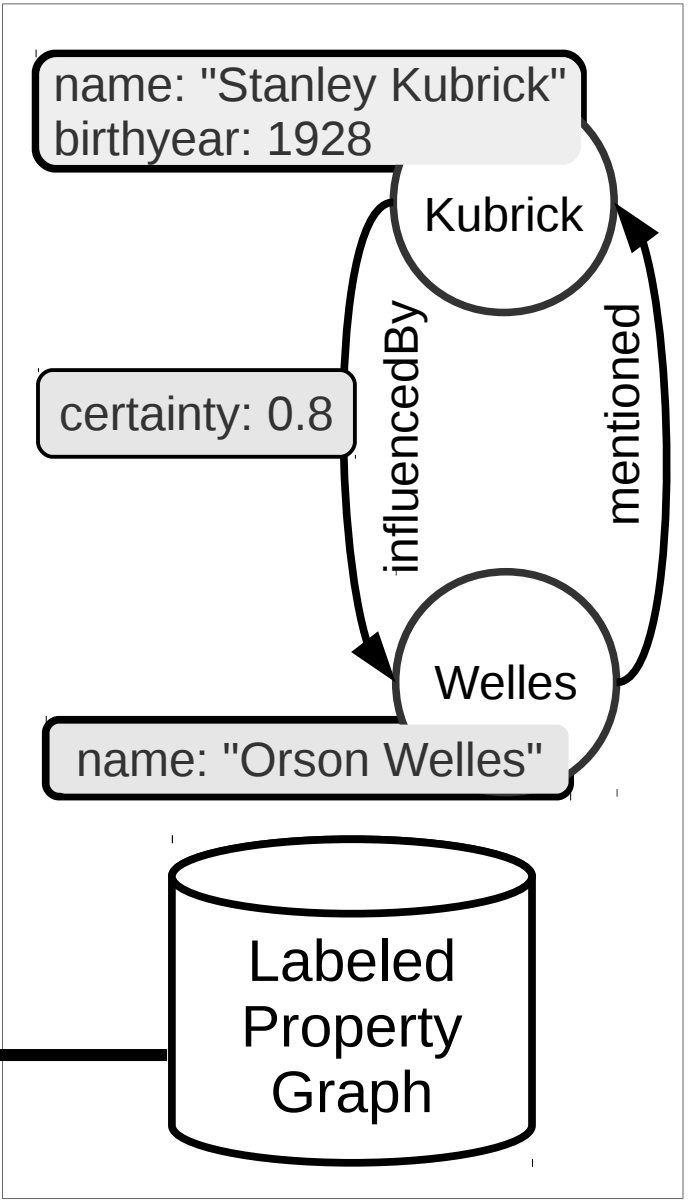
[4] Olaf Hartig: "**Reconciliation of RDF* and Property Graphs.**" In abs/1409.3288, 2014

Direct LPG-to-RDF* Mapping



Direct LPG-to-RDF* Mapping

- every node with its label → ordinary triple
- every node property → ordinary triple
- every edge (incl. its label) → ordinary triple
- every node property → nested triple

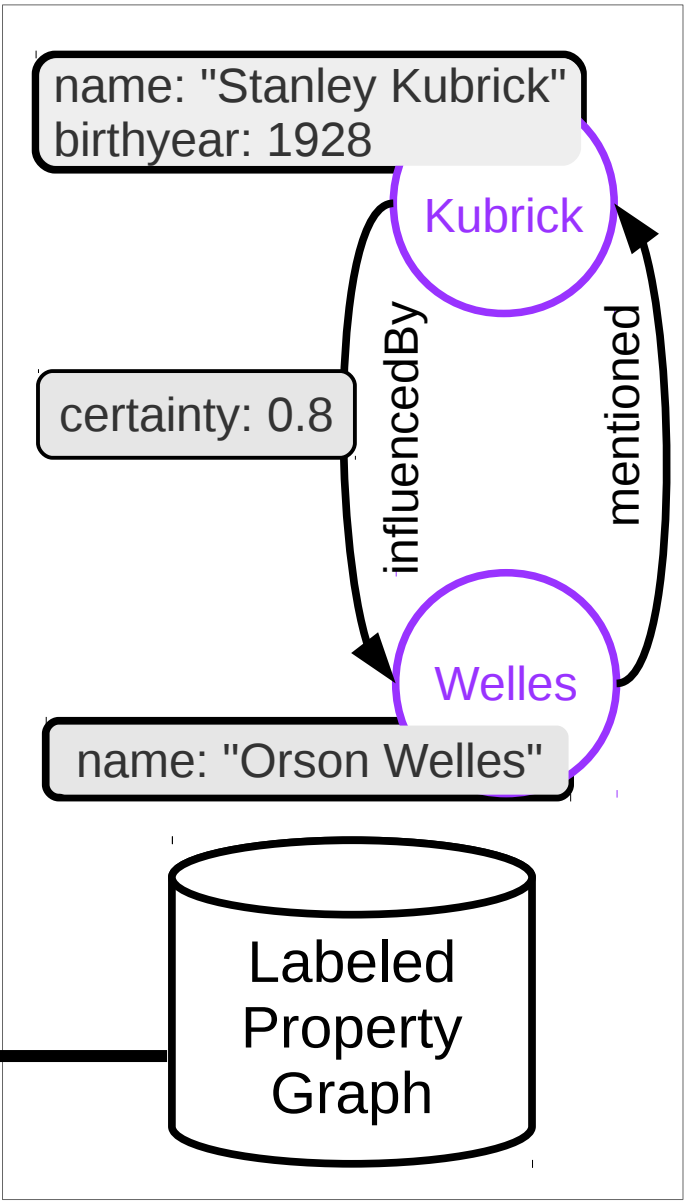



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Direct LPG-to-RDF* Mapping

- every node with its label → ordinary triple
- every node property → ordinary triple
- every edge (incl. its label) → ordinary triple
- every node property → nested triple

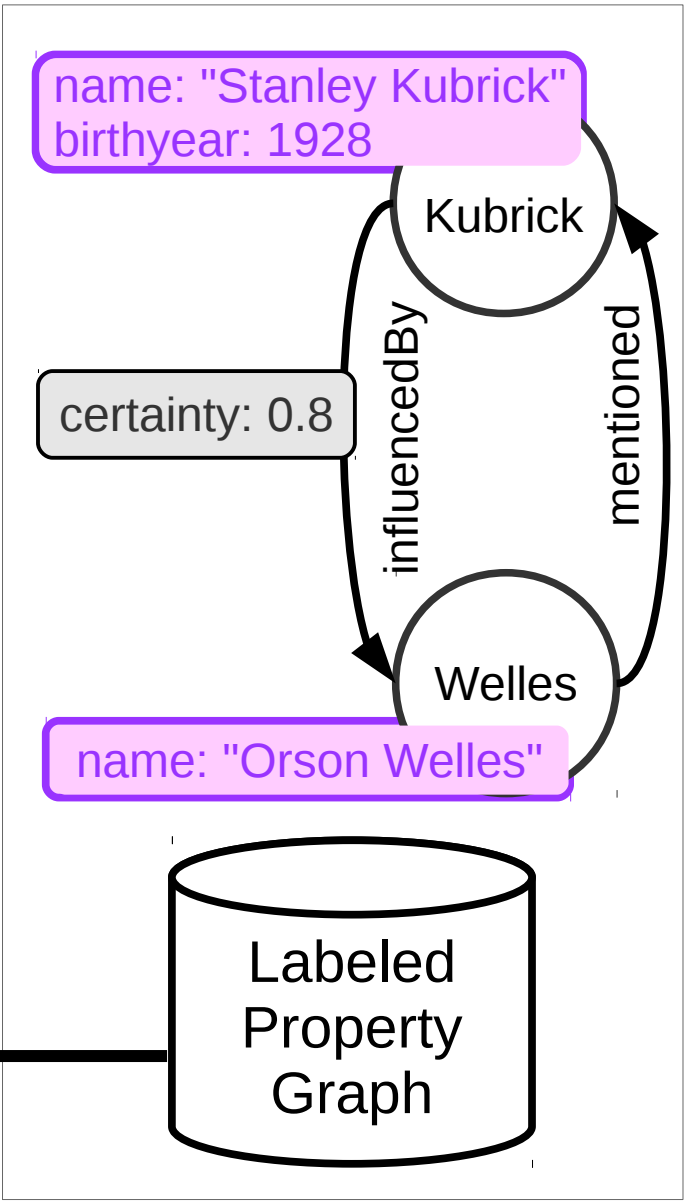



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Direct LPG-to-RDF* Mapping

- every node with its label → ordinary triple
- every node property → ordinary triple
- every edge (incl. its label) → ordinary triple
- every node property → nested triple

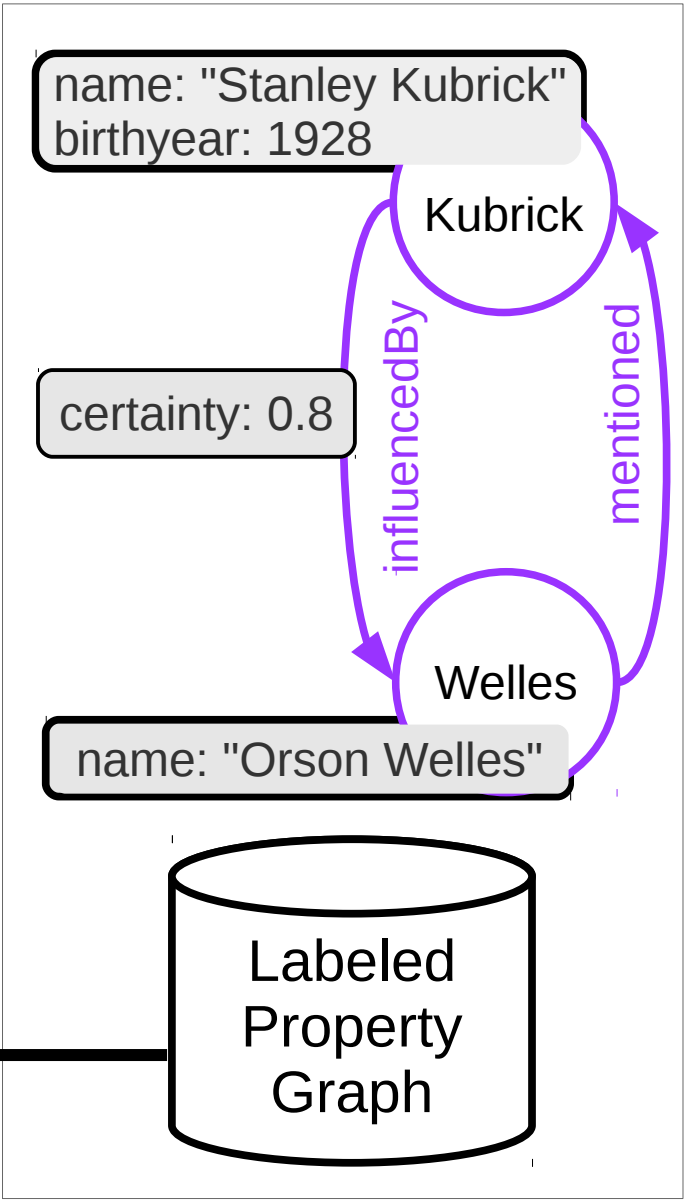



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Direct LPG-to-RDF* Mapping

- every node with its label → ordinary triple
- every node property → ordinary triple
- every edge (incl. its label) → ordinary triple
- every node property → nested triple

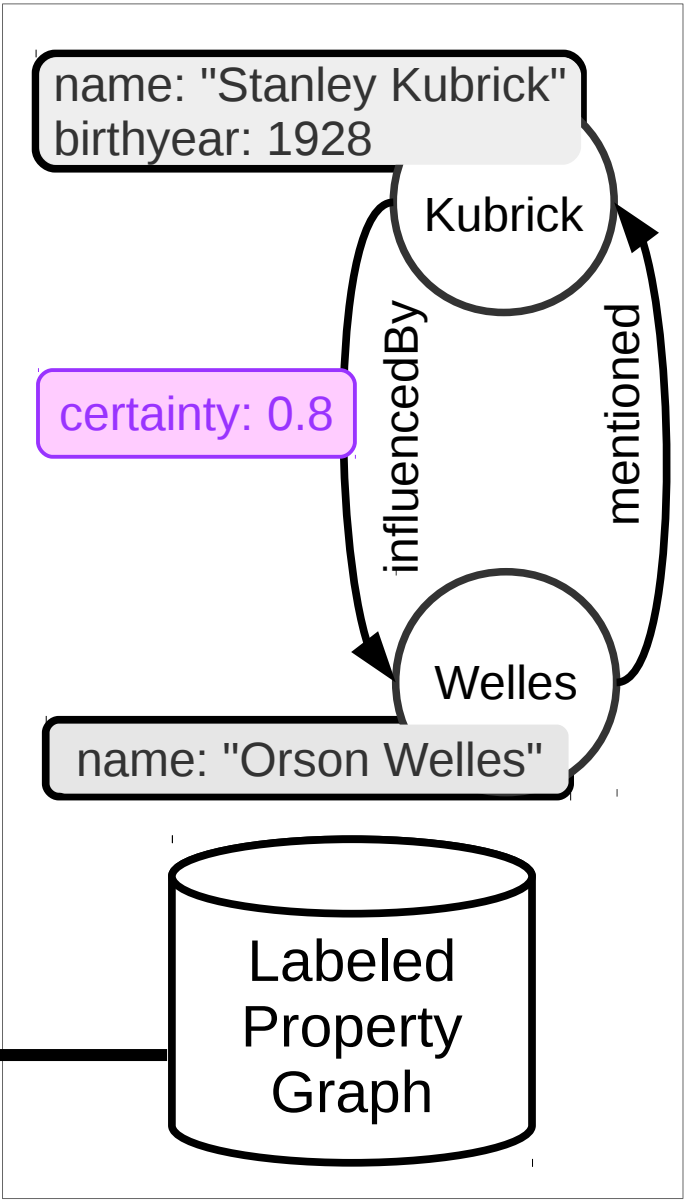



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Direct LPG-to-RDF* Mapping

- every node with its label → ordinary triple
- every node property → ordinary triple
- every edge (incl. its label) → ordinary triple
- every node property → nested triple

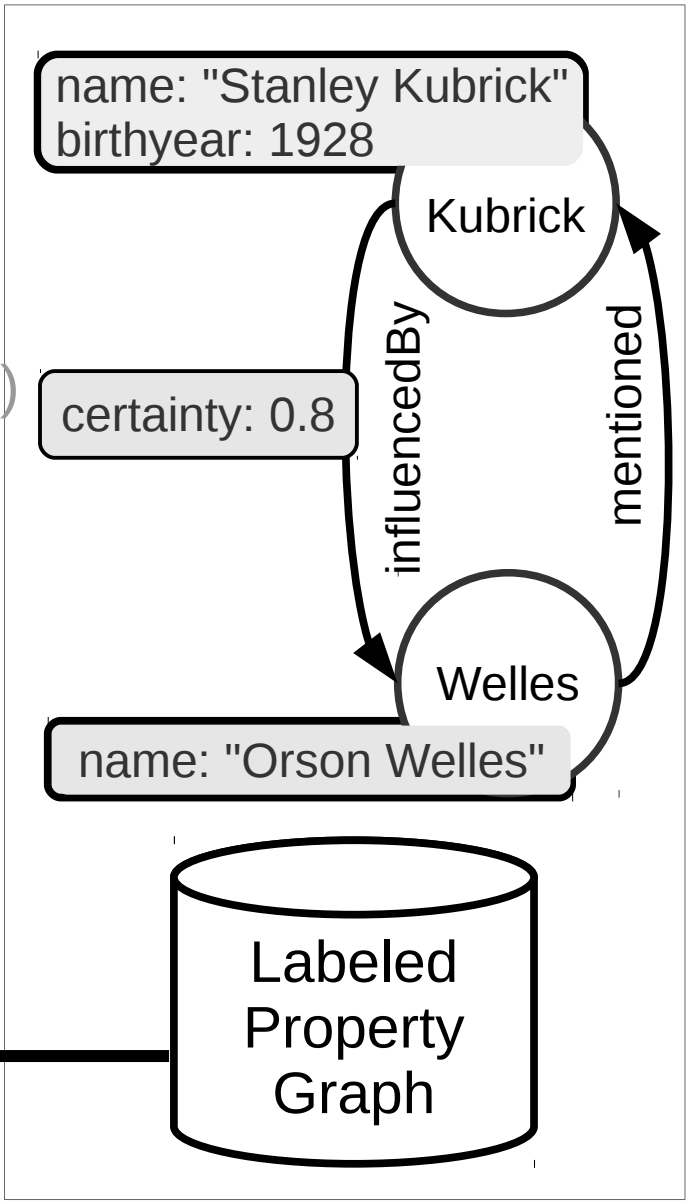



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Mapping is User-Configurable

- node mapping (injective, to bnodes/IRIs)
- node label predicate (IRI)
- node label mapping (injective, to IRIs/literals)
- edge label mapping (injective, to IRIs)
- property name mapping (injective, to IRIs)

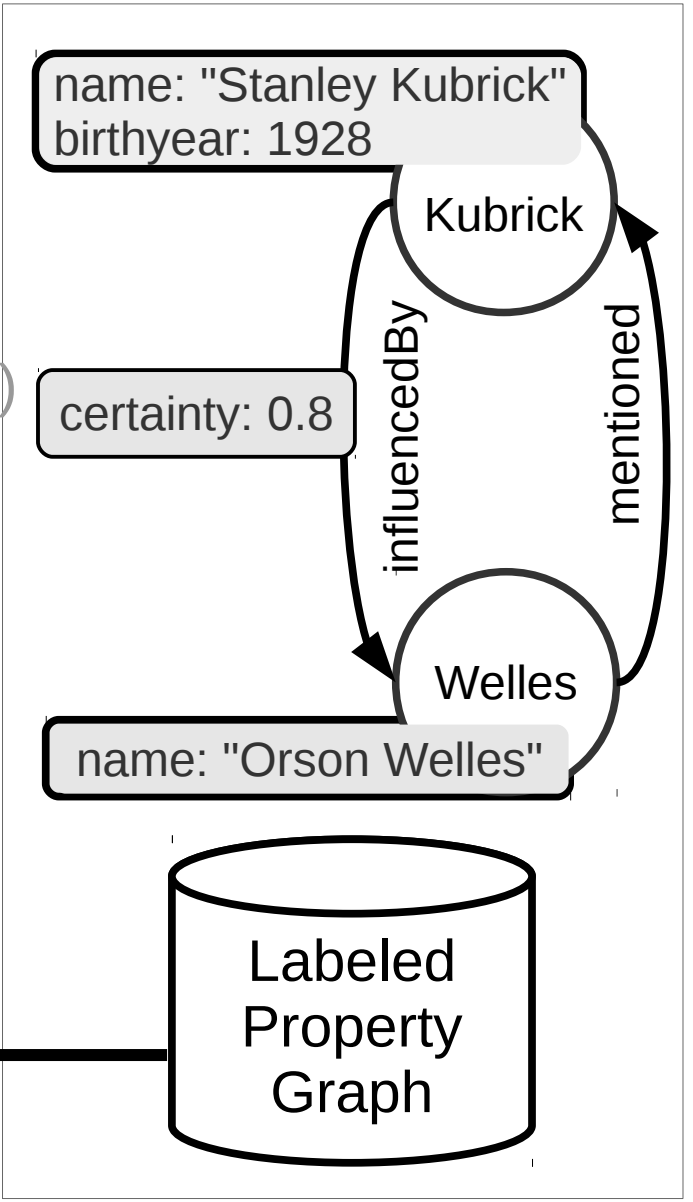



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Mapping is User-Configurable

- node mapping (injective, to bnodes/IRIs)
- node label predicate (IRI)
- node label mapping (injective, to IRIs/literals)
- edge label mapping (injective, to IRIs)
- property name mapping (injective, to IRIs)

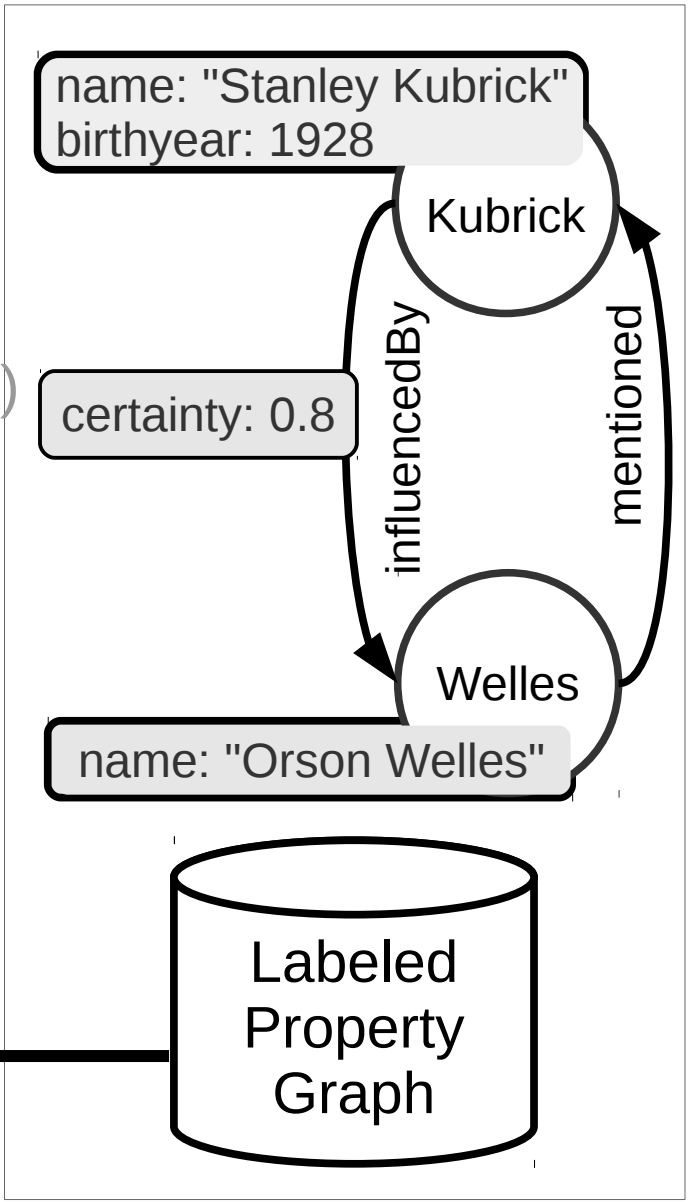



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Mapping is User-Configurable

- node mapping (injective, to bnodes/IRIs)
- node label predicate (IRI)
- node label mapping (injective, to IRIs/literals)
- edge label mapping (injective, to IRIs)
- property name mapping (injective, to IRIs)

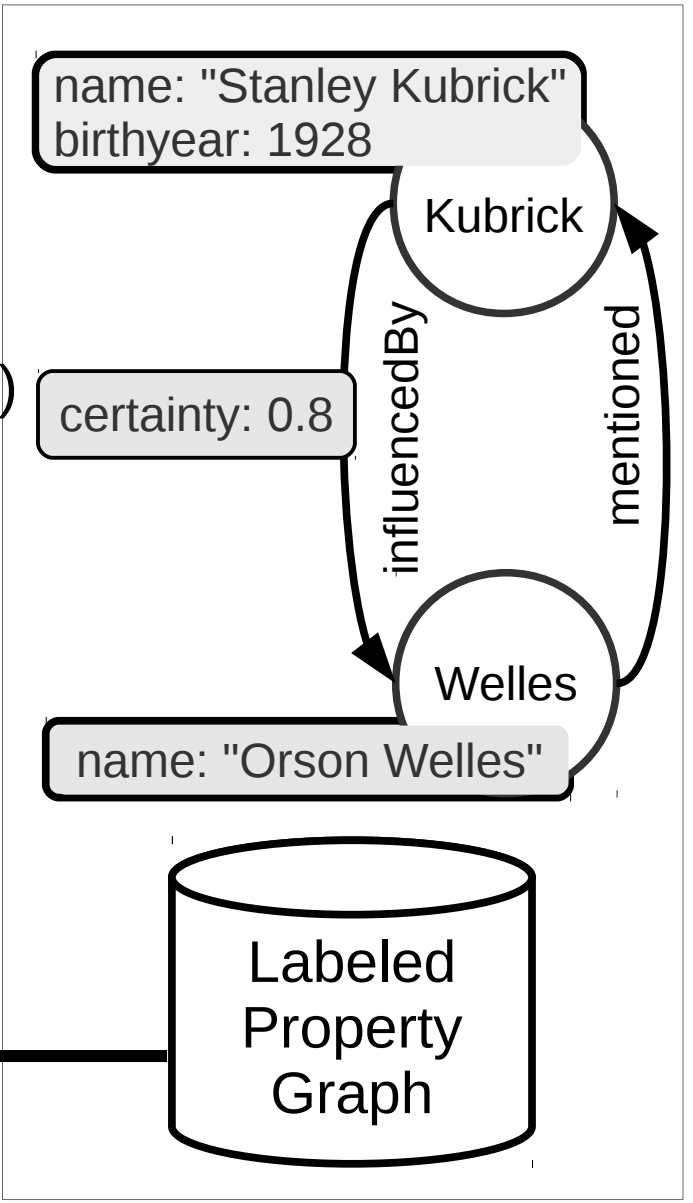



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Mapping is User-Configurable

- node mapping (injective, to bnodes/IRIs)
- node label predicate (IRI)
- node label mapping (injective, to IRIs/literals)
- edge label mapping (injective, to IRIs)
- property name mapping (injective, to IRIs)

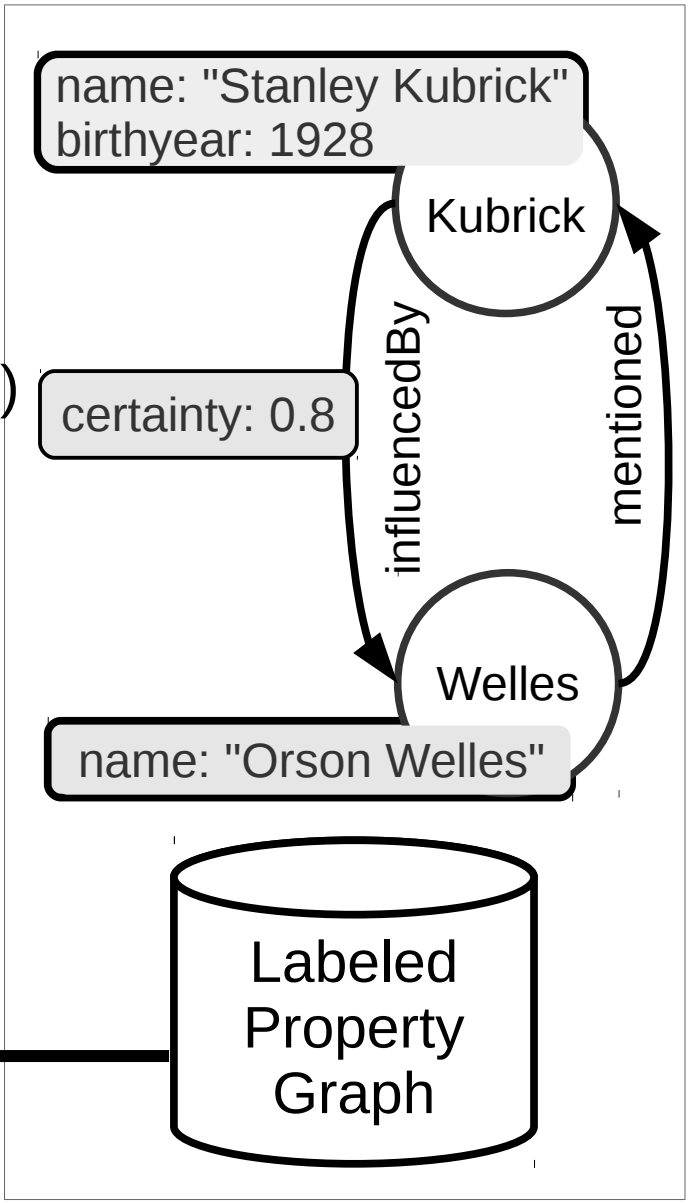



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Mapping is User-Configurable

- node mapping (injective, to bnodes/IRIs)
- node label predicate (IRI)
- node label mapping (injective, to IRIs/literals)
- edge label mapping (injective, to IRIs)
- property name mapping (injective, to IRIs)

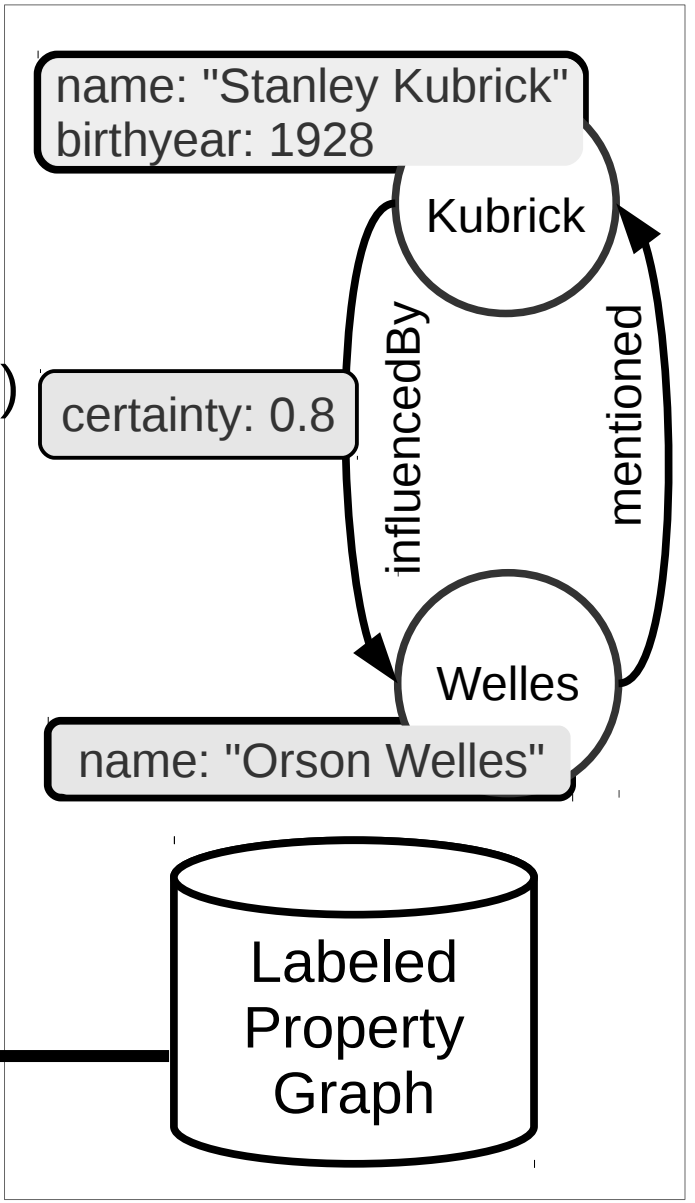



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Mapping is User-Configurable

- node mapping (injective, to bnodes/IRIs)
- node label predicate (IRI)
- node label mapping (injective, to IRIs/literals)
- edge label mapping (injective, to IRIs)
- property name mapping (injective, to IRIs)



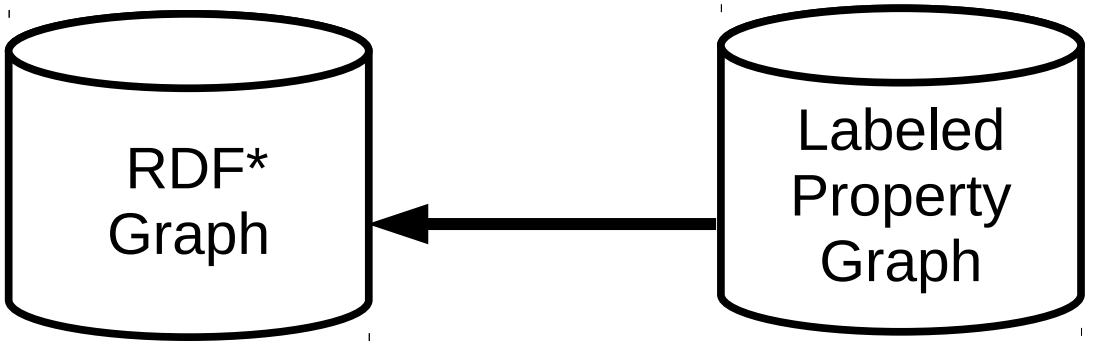

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1  rdfs:label  "Kubrick" .
_:b1  p:name  "Stanley Kubrick" .
_:b1  p:birthyear  1928 .
_:b2  rdfs:label  "Welles" .
_:b2  p:name  "Orson Welles" .
_:b2  r:mentioned  _:b1 .
<< _:b1 r:influencedBy _:b2 >>  p:certainty  0.8 .
```

Properties of the Mapping

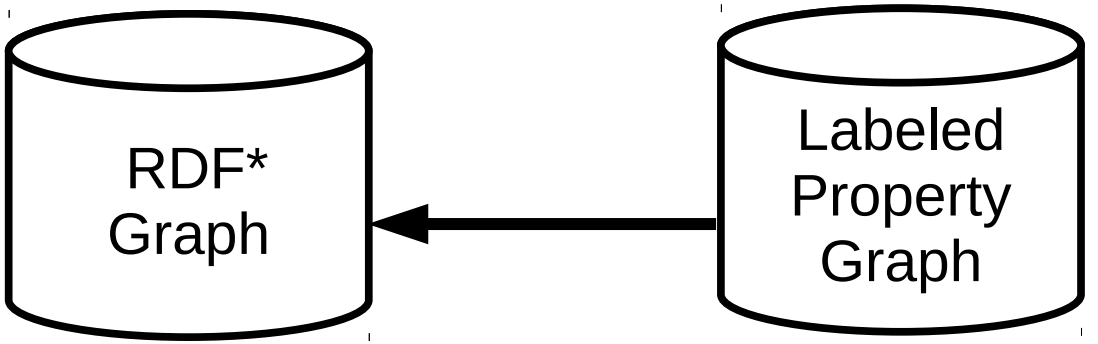
Note 1: the mapping is *linear* (for all possible configurations)

i.e., for every LPG *g*, the size of the resulting RDF* graph is linear in the size of *g*



Properties of the Mapping (cont'd)

Proposition 1:† for all *edge-unique* LPGs, the mapping is *information preserving* (for all possible configurations)

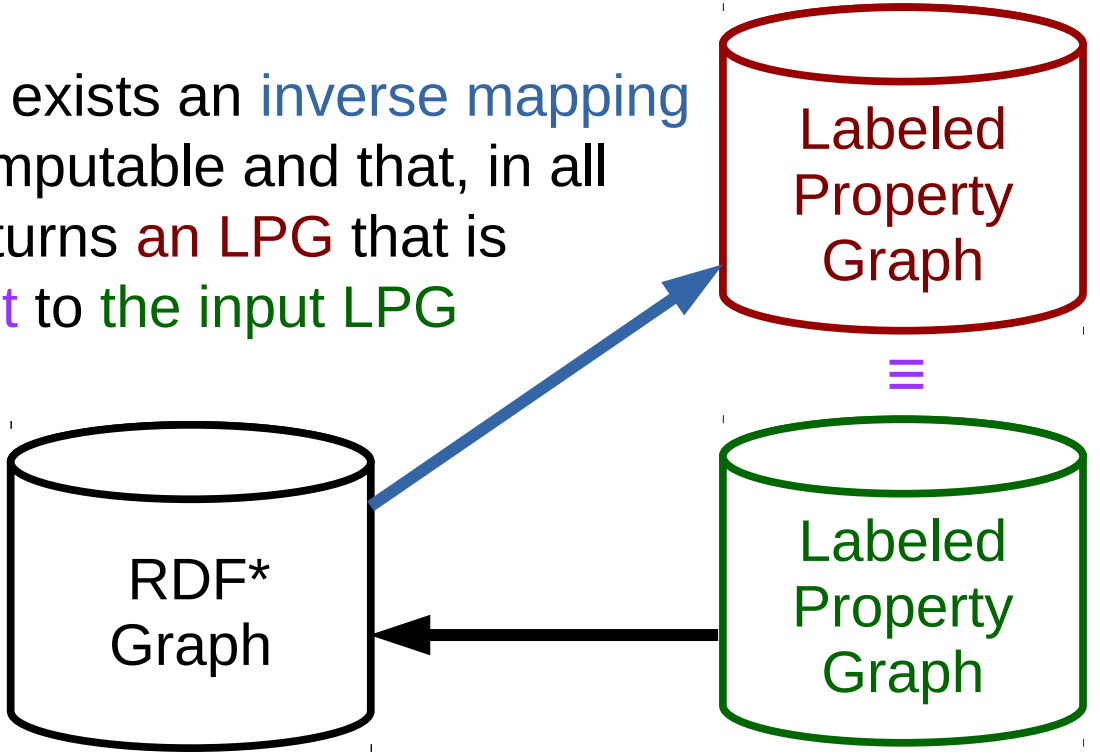


†for formal statement and proof sketch, see paper

Properties of the Mapping (cont'd)

Proposition 1:† for all *edge-unique* LPGs, the mapping is *information preserving* (for all possible configurations)

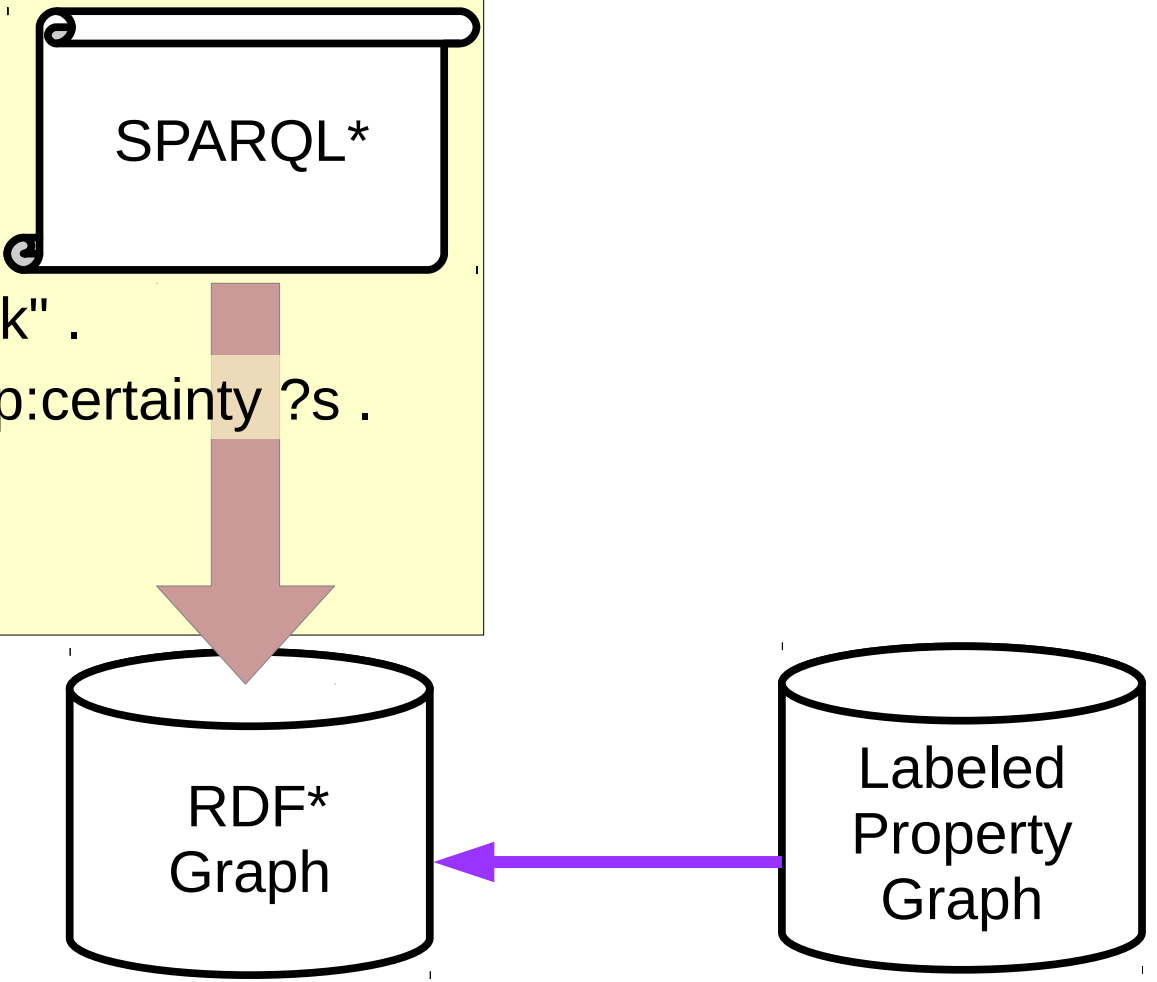
i.e., there exists an inverse mapping that is computable and that, in all cases, returns an LPG that is equivalent to the input LPG



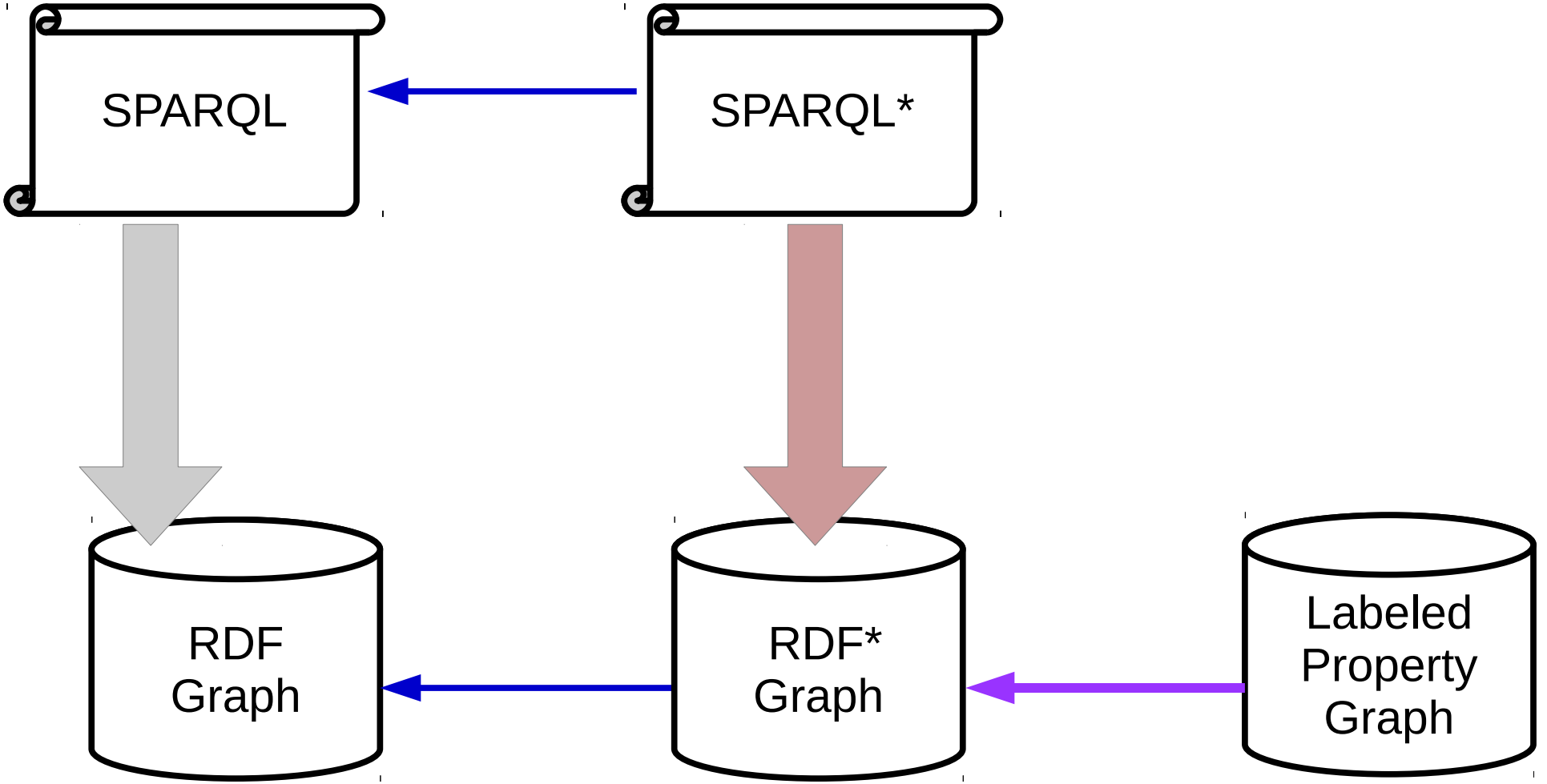
†for formal statement and proof sketch, see paper

Query the RDF* Representation of an LPG

```
SELECT ?n ?s WHERE {
    ?k p:name "Stanley Kubrick" .
    <<?k r:influencedBy ?x>> p:certainty ?s .
    ?x p:name ?n .
}
```

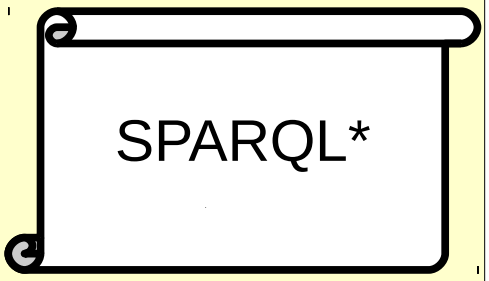


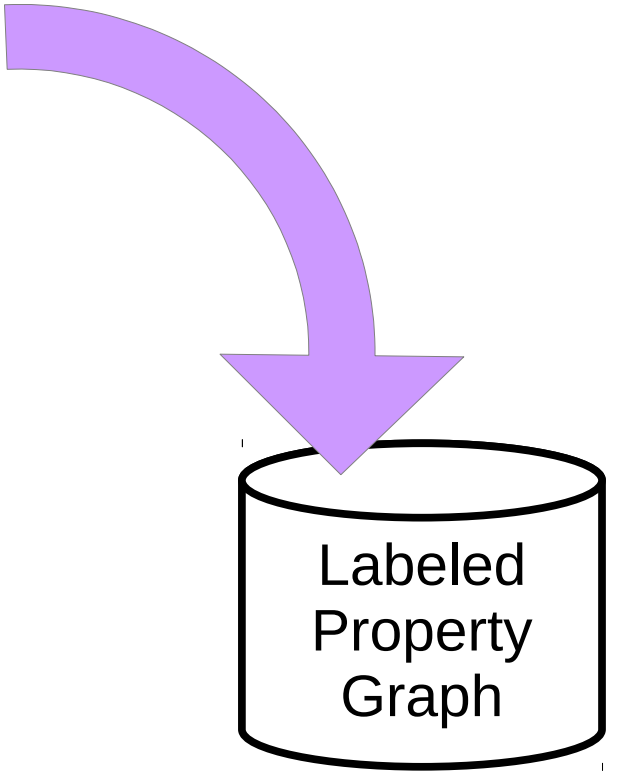
Query an RDF Representation of an LPG



LPG-based Query Semantics for SPARQL*

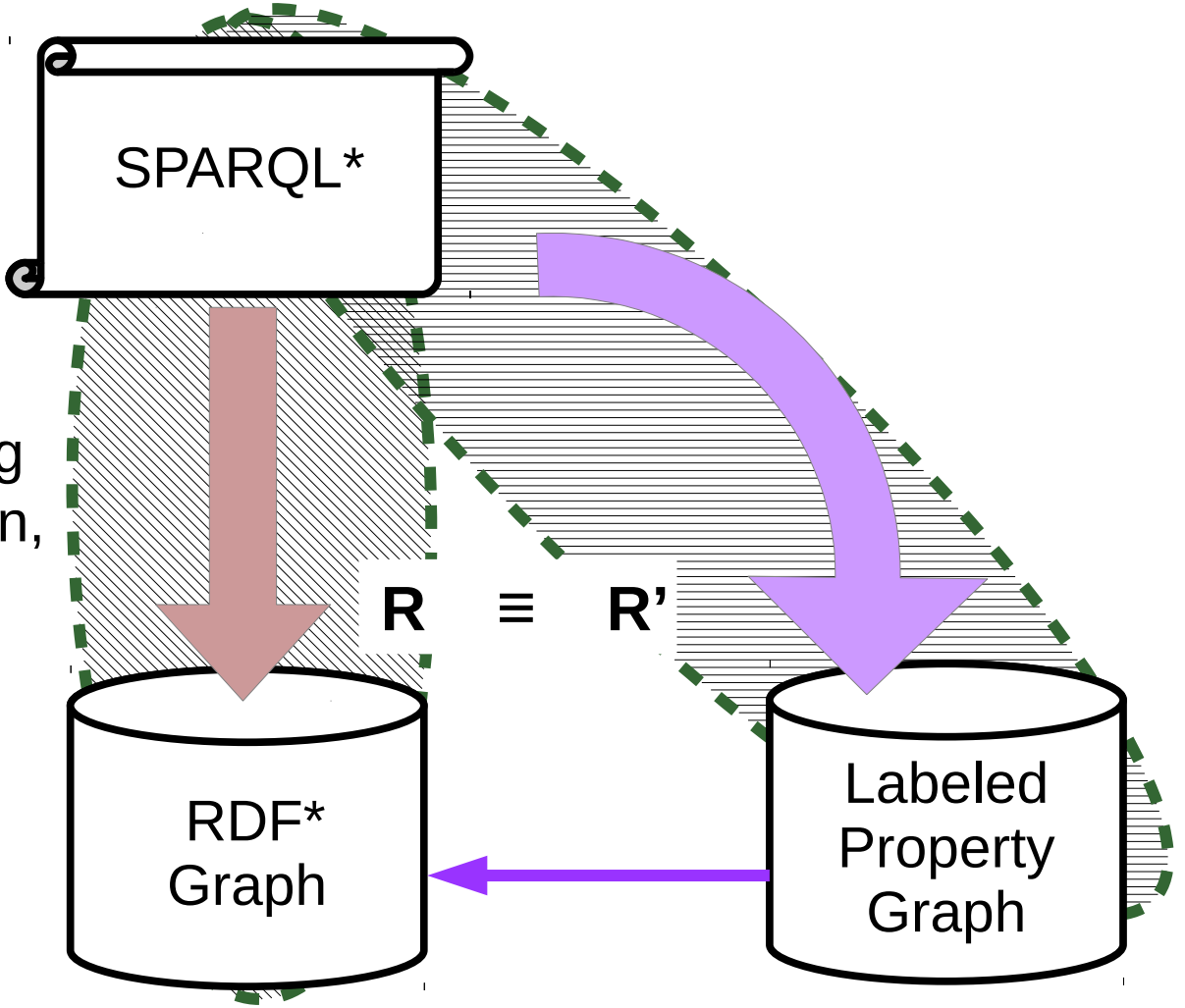
```
SELECT ?n ?s WHERE {
    ?k p:name "Stanley Kubrick" .
    <<?k r:influencedBy ?x>> p:certainty ?s .
    ?x p:name ?n .
}
```





Property of this Query Semantics

Proposition 2:† by using the same configuration, the query results are equivalent (including for LPGs that are *not* edge-unique)



†for formal statement and proof sketch, see paper

Take Away

- User-configurable direct LPG-to-RDF* mapping
  - information preserving (for edge-unique LPGs)
- User-configurable semantics for SPARQL* directly over LPGs
  - coincides with the RDF*-based semantics

www.liu.se

LINKÖPING UNIVERSITY