LIRIS

UMR 5205 CNRS

# PREC: Semantic Translation of Property Graphs

**Julian Bruyat**, Pierre-Antoine Champin, Lionel Médini, Frédérique Laforest

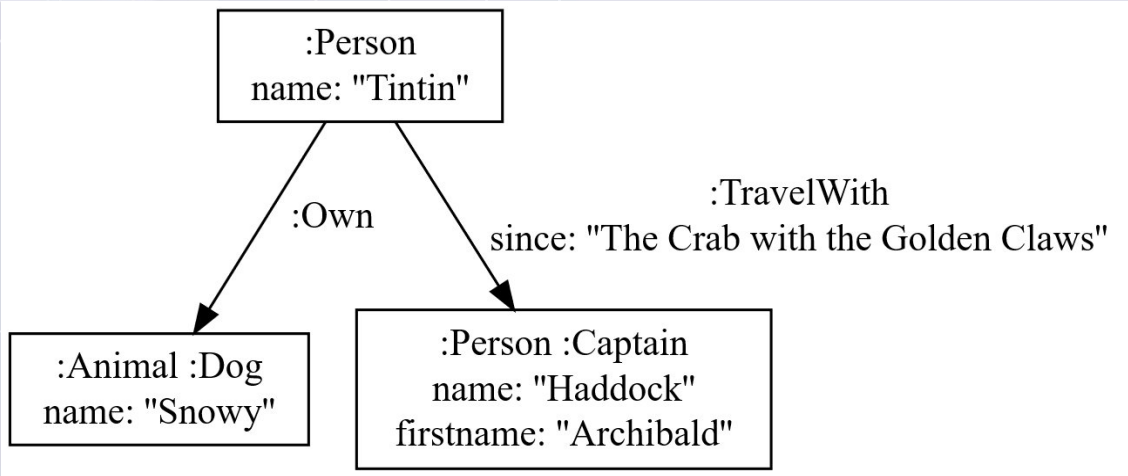
Laboratoire d'InfoRmatique en Image et Systèmes d'information

cnrs INSA Lyon 1 Université Lumière Lyon 2 École Centrale Lyon

# Structure of the talk

* **Property graphs and RDF graphs**
* **Existing bridges**
* **PREC**
  * Motivations
  * Workflow
  * Context
* **Future work**

# Property Graphs in this talk




*A Property Graph about the adventures of Tintin*

- No standard yet for Property Graphs
- Nodes and edges
- There are 0 to n labels on nodes, 1 label on edges
- Nodes and edges can hold properties. Properties themselves can hold properties

# RDF-star graphs

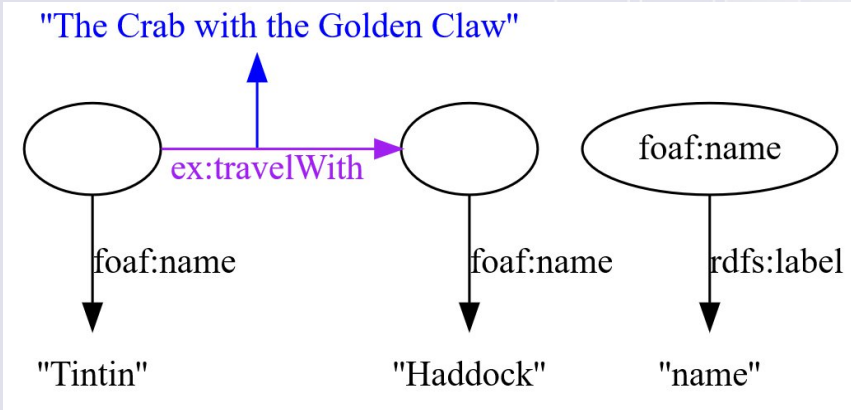
```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex: <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:tintin foaf:name "Tintin" .
_:haddock foaf:name "Haddock" .

foaf:name rdfs:label "name" .

_:tintin ex:travelWith _:haddock .

# An RDF-star triple
<< _:tintin ex:travelWith _:haddock >>
  ex:since "The Crab with the Golden Claws" .
```




*An RDF-star graph about the adventures of Tintin*
*with a Turtle representation on the left and a graphic representation on the right*

RDF is a W3C standard
- A set of RDF triples that can be represented as a graph
- RDF-star extension (in blue): we can use triples as the subject or the object of other triples, which enables to anotate triples

# Political differences between the two types of graphs

| | Property Graphs | RDF Graphs / Set of triples |
|---|---|---|
| **Selling point** | Easy to use | Standardized |
| **Some implementations** | Neo4j, TinkerPop, Amazon Neptune, Azure Cosmos DB | Blazegraph, Amazon Neptune, Jena, N3.js |
| **Query Paradigm** | Cypher, Gremlin, GQL: Graph browsing | SPARQL: Pattern Matching |
| **Closed/Open World Assumption** | Depends on the user (Closed-world assumption) | Open-world assumption |
| **Scoping** | Locally scoped | IRIs are globally scoped |

# State of the art - Existing bridges

## - Syntactic elements to translate PGs to RDF:

* RDF-star (First article from O. Hartig and B. Thompson in 2014, Community Group since 2020 lead by O. Hartig et PA. Champin)
* A Tales of Two Graphs (Das, S, et al, 2014): Study possible representation of PG edges in RDF
* graphConv/pgo Ontology (Tomaszuk et al, 2020): An ontology to describe PG in RDF

## - Converters:

* NeoSemantics (J. Barrassa): Neo4j <-> RDF conversion, inference on PG based on RDFs/OWL
* graphConv
* G2GML (S. Matsumoto et al, 2018): SPARQL pattern to produce a PG
* rdf2neo (M. Brandizi et al, 2016): Use case to produce a Neo4j database from an RDF graph. Authors are currently working on rdf2pg

## - Query rewriting:

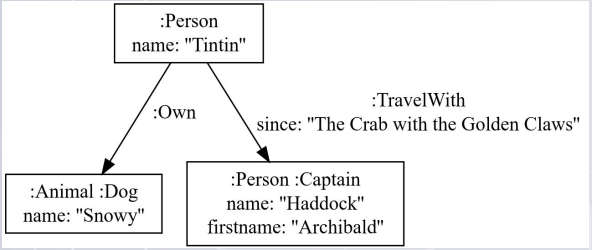
* Gremlinator (H. Thakkar et al, 2018): Query an RDF database with Gremlin

## - Mapping:

* R2RML: W3C standard to produce triples from an SQL table. User driven
* RML (A. Dimou et al, 2014): Extension of R2RML to other formats like XML or JSON
* JSON-LD: Specific JSON format (named context) to translate any JSON document to RDF

# PREC - Motivations

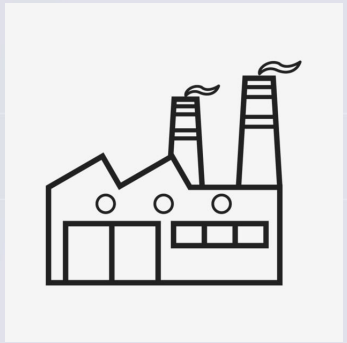
* **Current solutions are:**
  * Too simplistic - NeoSemantics: everything is true, information loss
  * Not enough - graphConv: Literal description of the PG
* **Target RDF schema should suit the data**
  * Let the user decide: use a context (à la JSON-LD) for translation



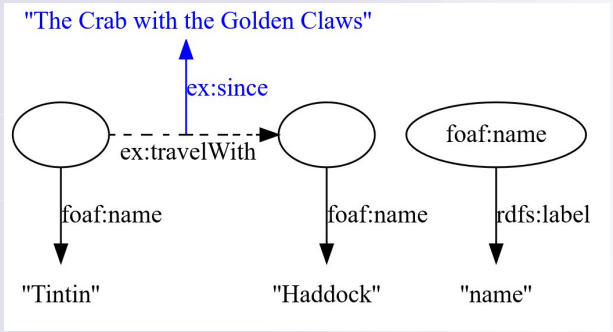

*Source Property Graph*

```
{ :TravelWith = http://example.org/travelWith + not true,
  name (of a :Person) = foaf:name ,
   (...)
}
```

*Context / Mapping*

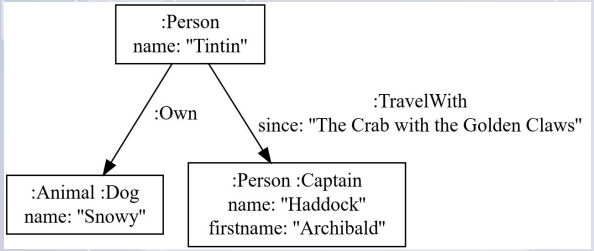


*PREC*



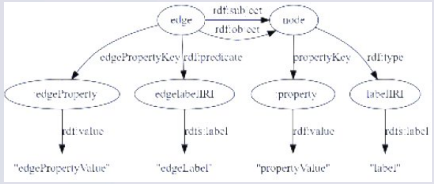

*(A part of) the produced RDF-star graph*

# PREC - Workflow




*Source Property Graph (from any engine)*

***PREC-0***
***Describe the PG in RDF***



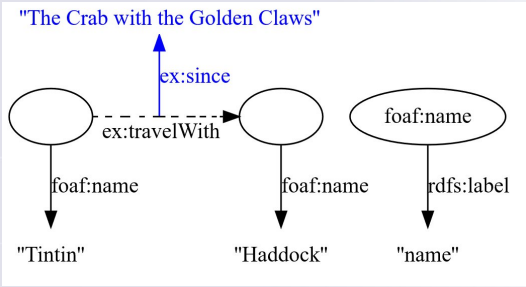

*Schema of the produced RDF graph*

```
[] a prec:PropertyRule ;
  prec:nodeLabel "Person" ;
  prec:propertyName "name" ;
  prec:propertyIRI foaf:name ;
  prec:templatedBy prec:DirectTriples .

[] a prec:EdgeRule ;
  prec:edgeLabel "TravelWith" ;
  prec:edgeIRI ex:travelWith ;
  prec:templatedBy :annotationOnly .
# ...
```

*Context*

***PREC-C***
***Apply the context on the RDF description of the PG***




*(A part of) the produced RDF-star graph*

# PREC Context

* **A PREC context**
  * Enbles the user to specify the semantic of the terms used in the PG
  * Declarative style in RDF
  * https://bruy.at/prec

```
[] a prec:PropertyRule ;
  prec:nodeLabel "Person" ;
  prec:propertyName "name" ;
  prec:propertyIRI foaf:name ;
  prec:templatedBy prec:DirectTriples .
```

```
[] a prec:EdgeRule ;
  prec:edgeLabel "TravelWith" ;
  prec:edgeIRI ex:travelWith ;
  prec:templatedBy :annotationOnly

:annotationOnly  a prec:EdgeTemplate ;
  prec:composedOf
    << << ?source ?edgeIRI ?destination >>
                          ?propertyPredicate ?propertyObject >>
.
```

# Conclusion and Future works

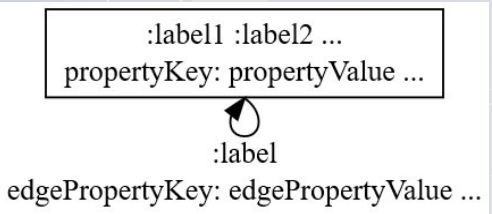
- **PREC: a two step conversion from PG to RDF:**
  - Convert the PG to a structural description of the PG in RDF
  - Main contribution: Translate the structure of the PG to an idiomatic RDF graph
    - let the user map the labels and property names of the PG
    - choose the representation of nodes, edges and properties
  - https://github.com/BruJu/PREC

- **Future works:**
  - **Revert back from RDF to PREC**
    - Detection of information loss
    - Ambiguity of a context (w.r.t. the produced data graph)
  - **Query rewriting**
    - Context is a description of how to expose the data: we do not need to convert the graphs

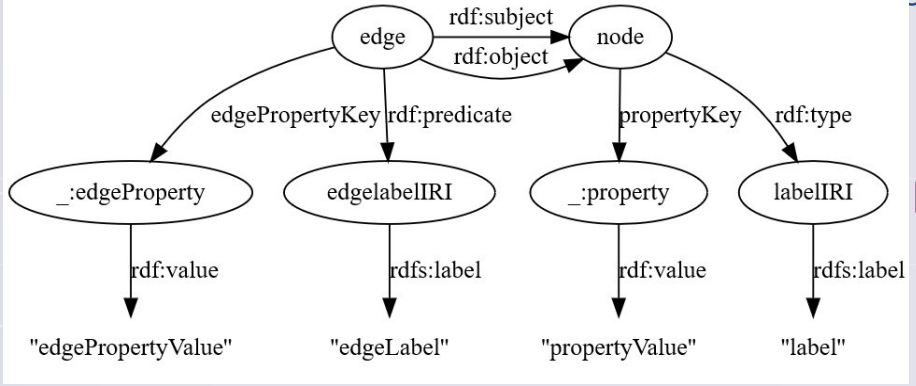
# PREC - Context application details

* **Tradeoff between expressivity and easy of use.**
* **Vocabulary mentions the original PG (*e.g.* the prec:edgeLabel for the label of the edge)**

* **On which graph is applied the context?**
  * Great diversity of PG implementations and API (Cypher, Gremlin, …)
  * Two step transformation:
    * First transform the PG into an RDF graph that describes the structure of the PG. Agnostic from the context
    * Then apply the context to transform this RDF graph. Agnostic from the original PG engine.




*General schema of Property graphs*




*Schema of RDF graphs produced by PREC-0 (description of the PGs)*

*After the context application, the schema depends on the context*