

An Algebraic Graph Transformation Approach for RDF and SPARQL

D. Duval and R. Echahed and F. Prost

LJK - LIG - CNRS - Université Grenoble Alpes

June 24, 2020

Introduction

- ▶ Graph databases are more and more used
 - W3C Workshop on Web Standardization for Graph Data .
- ▶ Today's de facto standard: RDF and SPARQL.
- ▶ Graph database queries are graph transformations.
 - ▶ DB querying from a graph transformation perspective?
 - ▶ DB querying implicitly involves universal quantification on matches.
- ▶ Motivations:
 - ▶ SPARQL is heterogeneous (CONSTRUCT and SELECT).
 - ▶ The SPARQL semantics, as given in W3C recommendations, is complex.

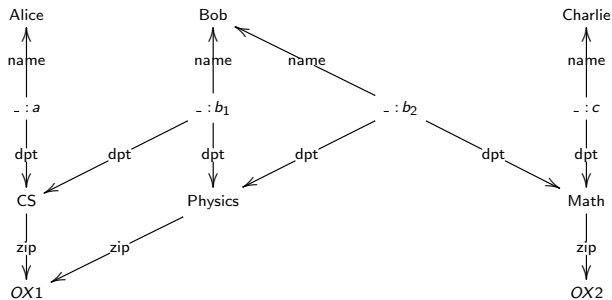
RDF triplestores=graphs

- ▶ Triple store: (s, p, o) viewed as labelled arrow $s \xrightarrow{p} o$.
- ▶ Three kinds of atoms :
 - ▶ IRI: internationalized Resource Identifier.
 - ▶ Litteral: strings of characters, integers.
 - ▶ Blank nodes.
- ▶ Not the usual graphs.
 - ▶ No isolated atoms.
 - ▶ Predicates can also be sources (subject) or destinations (object) of arrows.

RDF triplestore

RDF Triple store

- :a name Alice.	- :a dpt CS.	
- :b1 name Bob.	- :b1 dpt CS.	- :b1 dpt Physics.
- :b2 name Bob.	- :b2 dpt CS.	- :b2 dpt Math.
- :c name Charlie.	- :c dpt Math.	
CS zip OX1.	Physics zip OX1.	Math zip OX2.



SPARQL query language - CONSTRUCT

SPARQL Query

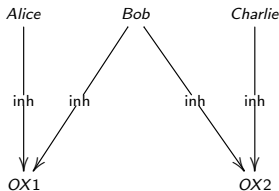
```
CONSTRUCT {?n inh ?z }  
WHERE {?x name ?n.  
       ?x dpt ?d.  
       ?d zip ?z }
```

RDF Triple store

```
- :a name Alice.      - :a dpt CS.  
- :b1 name Bob.      - :b1 dpt CS.        - :b1 dpt Physics.  
- :b2 name Bob.      - :b2 dpt CS.        - :b2 dpt Maths.  
- :c name Charlie.   - :c dpt Math.  
CS zip OX1.          Physics zip OX1.   Math zip OX2.
```

Query result

```
Charlie inh OX2.  
Bob inh OX2.  
Alice inh OX1.  
Bob inh OX1.
```



SPARQL semantics

- ▶ Officially defined by the W3C recommendations:

<https://www.w3.org/TR/sparql11-query/>

- ▶ There are other proposals of SPARQL formal semantics not based on graph transformation techniques, e.g. [KRU15].
- ▶ Specific points:
 - ▶ Done with SQL in mind (clear from imbalance between CONSTRUCT and SELECT in W3C recommendations).
 - ▶ There is no equivalent of the relational algebra for CONSTRUCT.

Algebraic point of view - Categories

Definition

Let A be a set, called the the of attributes.

- ▶ A *graph on A* is a subset of A^3 : subjects, predicates and objects.
- ▶ A *morphism $a : T \rightarrow T'$* is a partial map $A \rightarrow A$ that preserves triples.

\Rightarrow This yields the *category of graphs on A* , denoted $\mathcal{G}(A)$.

We say that a morphism $a : T \rightarrow T'$ of graphs on A *fixes* a subset C of A if $a(x) = x$ for each x in $T \cap C$.

The subcategory of $\mathcal{G}(A)$ made of the graphs on A with the morphisms fixing C is denoted $\mathcal{G}_C(A)$.

Data Graphs and Query Graphs Categories

Definition

Let I , B and V be the sets of *resource identifiers* (IRI and literals), *blanks* and *variables*. Let $IB = I \cup B$, $IV = I \cup V$ and $IBV = I \cup B \cup V$.

- ▶ The *category of data graphs* is $\mathcal{D} = \mathcal{G}(IB)$
- ▶ The *category of query graphs* is $\mathcal{Q} = \mathcal{G}(IBV)$

- ▶ Blanks and variables play a different role.

Definition

A *match* from a query graph L to a data graph G is a morphism of query graphs from L to G which fixes I . The set of matches from L to G is denoted $\text{Match}(L, G)$ and the set of all matches from L to any data graph is denoted $\text{Match}(L)$.

Query rules for basic CONSTRUCT Queries

The basic SPARQL query "CONSTRUCT {R} WHERE {L}", basic meaning that R, L are query graphs, is seen as follows:

Definition

A *basic construct query* is (L, R) in $\mathcal{Q}_{IV} \times \mathcal{Q}_{IV}$, such that blanks are different in L and R and every variable in R is in L . The *transformation rule* of (L, R) is the cospan

$P_{L,R} = (L \xrightarrow{l} K \xleftarrow{r} R)$ where $K = L \cup R$ and l and r are the inclusions.

$$P_{L,R} = L \xrightarrow[\subseteq]{l} K = L \cup R \xleftarrow[\supseteq]{r} R$$

The POIM transformation - single match

1. PO: pushout of l and m in \mathcal{Q}_I .

The *cobase change along l* is the map

$l_* : \text{Match}(L) \rightarrow \text{Match}(K)$ that maps each $m : L \rightarrow G$ to $l_*(m) : K \rightarrow D$ defined from the pushout of l and m in \mathcal{Q}_I ,

2. IM: image factorization.

The *image factorization along r* is the map

$r^+ : \text{Match}(K) \rightarrow \text{Match}(R)$ that maps each $n : K \rightarrow D$ to $r^+(n) : R \rightarrow H$ where H is the image of R in D and $r^+(n)$ is the restriction of n and $h : H \rightarrow D$ is the inclusion.

$$\begin{array}{ccccc}
 L & \xrightarrow{l} & K & \xleftarrow{r} & R \\
 m \downarrow & & \vdots \downarrow & & \vdots \downarrow \\
 & (PO) & l_*(m) = n & (IM) & r^+(n) = p \\
 & & \downarrow & & \downarrow \\
 G & \xrightarrow{g} & D & \xleftarrow{h} & H \\
 & & & & \text{Polm}_{L,R}(m)
 \end{array}$$

The POIM Transformation - Multiple Matches (1)

- ▶ Querying a DB involves all eligible matches.
- ▶ Two equivalent approaches:
 - ▶ Low-level: make as many copies of (L, R) as there are matches.
 - ▶ High-level: make one big match out of small matches.

Definition (Low-level)

Let (L, R) be a basic construct query and G a data graph. Let m_1, \dots, m_k be the matches from L to G . For each $i = 1, \dots, k$ let H_i be the result of applying the POIM transformation $POIM_{L,R}$ to the match $m_i : L \rightarrow G$. It is the datagraph obtained from R by replacing each variable x in R by $m_i(x)$ and each blank in R by a fresh blank.

The *query result* of applying the basic construct query (L, R) to the data graph G is the data graph $H = H_1 \cup \dots \cup H_k$.

The POIM Transformation - Multiple Matches (2)

Definition (High-level)

Let (L, R) be a basic construct query and G a data graph. Let m_1, \dots, m_k be the matches from L to G . Consider the basic construct query (kL, kR) . Let m be the match from kL to G that coincides with m_i on the i -th component of kL . The *high-level query result of (L, R) against G* is the result H_{high} of applying the POIM transformation $Polm_{kL, kR}$ to the match $m : kL \rightarrow G$.

The POIM Transformation - Multiple Matches (2)

Definition (High-level)

Let (L, R) be a basic construct query and G a data graph. Let m_1, \dots, m_k be the matches from L to G . Consider the basic construct query (kL, kR) . Let m be the match from kL to G that coincides with m_i on the i -th component of kL . The *high-level query result of (L, R) against G* is the result H_{high} of applying the POIM transformation $Polm_{kL, kR}$ to the match $m : kL \rightarrow G$.

Proposition

Let (L, R) be a basic construct query and G a data graph. The low-level query result of (L, R) against G is isomorphic, in the category \mathcal{D}_I , to the high-level query result of (L, R) against G .

Example - High-Level

- ▶ Consider the query :

```
CONSTRUCT {?x name _ :n} WHERE { ?x name ?name }
```

Example - High-Level

- ▶ Consider the query :

CONSTRUCT { ?x name _ :n } WHERE { ?x name ?name }

?x1 name ?name1.
?x2 name ?name2.



?x1 name ?name1.
?x1 np _ :n1.
?x2 name ?name2.
?x2 np _ :n1.

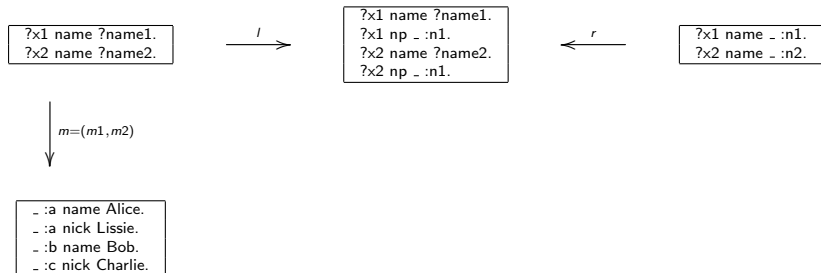


?x1 name _ :n1.
?x2 name _ :n2.

Example - High-Level

- ▶ Consider the query :

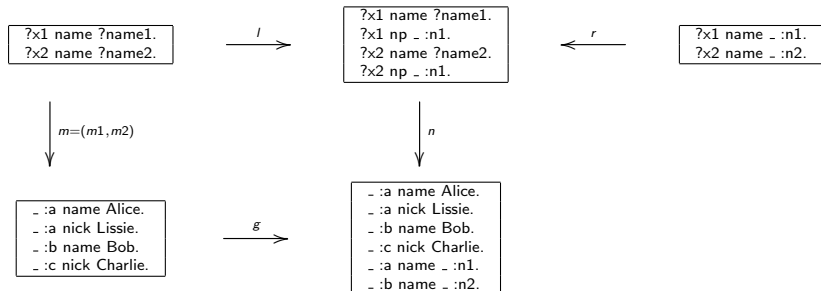
CONSTRUCT { ?x name _ :n } WHERE { ?x name ?name }



Example - High-Level

- ▶ Consider the query :

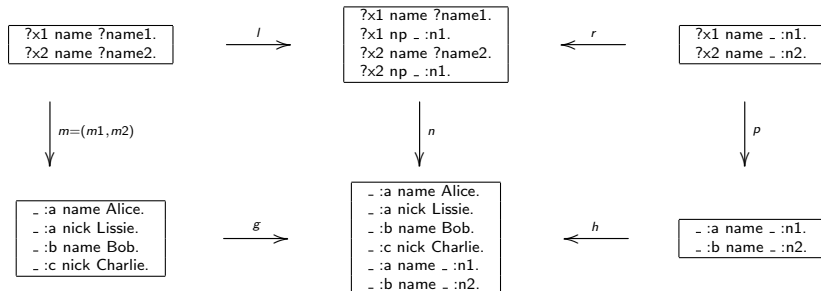
CONSTRUCT { ?x name _ :n } WHERE { ?x name ?name }



Example - High-Level

- ▶ Consider the query :

CONSTRUCT { ?x name _ :n } WHERE { ?x name ?name }



Conclusion

- ▶ SELECT queries can be considered as special cases of CONSTRUCT queries.





Conclusion

- ▶ SELECT queries can be considered as special cases of CONSTRUCT queries.
- ▶ Related works:
 - ▶ Formal semantics not based on graph transformations [KRU15].
 - ▶ Approach to RDF graph transformation MPOC-PO in [BB08].
 - ▶ Ontologies as categories [AJK15].
 - ▶ Non-local transformations [CDE⁺19]

Conclusion

- ▶ SELECT queries can be considered as special cases of CONSTRUCT queries.
- ▶ Related works:
 - ▶ Formal semantics not based on graph transformations [KRU15].
 - ▶ Approach to RDF graph transformation MPOC-PO in [BB08].
 - ▶ Ontologies as categories [AJK15].
 - ▶ Non-local transformations [CDE⁺19]
- ▶ Ongoing work:
 - ▶ Extend the kernel of SPARQL.
 - ▶ Extend subqueries.
 - ▶ Updates.

Bibliography

-  S. Aliyu, S.B. Junaidu, and A. F. Donfack Kana.
A category theoretic model of RDF ontology.
International Journal of Web & Semantic Technology (IJWesT), 2015.
-  Benjamin Braatz and Christoph Brandt.
Graph transformations for the resource description framework.
ECEASST, 10, 2008.
-  Andrea Corradini, Dominique Duval, Rachid Echahed, Frédéric Prost, and Leila Ribeiro.
The PBPO graph transformation approach.
J. Log. Algebraic Methods Program., 103:213–231, 2019.
-  Egor V. Kostylev, Juan L. Reutter, and Martín Ugarte.
CONSTRUCT queries in SPARQL.
In *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium*, pages 212–229, 2015.