The Perfect Match: RPL and RDF Rule Languages

François Bry, Tim Furche, Benedikt Linse

Institute for Informatics, University of Munich, Germany

October 26, 2009

RDF Principles and Usage

- Keep it simple, stupid.
- Information in the form of statements (subject, predicate, object).
- URIs for resolving ambiguity.
- A model theory as a precise foundation for entailment.

RDF Data on the Web:

- DBPedia: about 274 million triples (2008)
- LinkedData Initiative: estimated 2 Billion triples (last year)
- OpenStreetMaps: more than 3 Billion triples

The need for an RDF path query language

- What SPARQL (and other single-rule languages) cannot do:
  - arbitrary depth path traversal
- Path query languages for SSD:
  - XPath for XML
  - CSS for HTML
- No expressible, yet efficient RDF Path query language yet!!
- The need for *qualified descendant* pointed out in [Mar05]
- Embeddability in more general rule and query languages
  - NREs [PAG08] may evaluate to pairs of nodes and edges
  - path languages should *complement* rule languages

Overview

- RPL by example
    - Simple queries
    - RDFS queries
    - Nested queries with negation
- RPL Syntax and Formal Semantics
- RPL Evaluation
- RPL Implementation and Demo

Paths in RDF Graphs

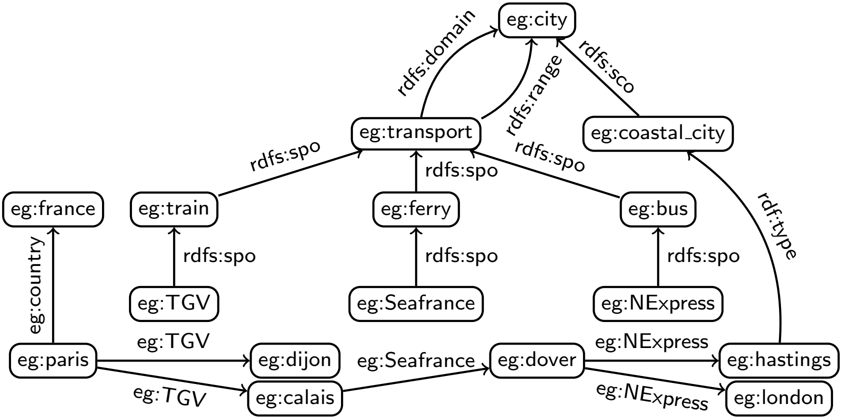
Definition (RDF triple, graph)

Let U, B, L be three disjoint sets of URIs, blank node identifiers and RDF literals. Then $t = (s, p, o) \in U \cup B \times U \times U \cup B \cup L$ is an *RDF triple*. A *RDF graph* is a set of RDF triples.

Definition (Path in an RDF graph)

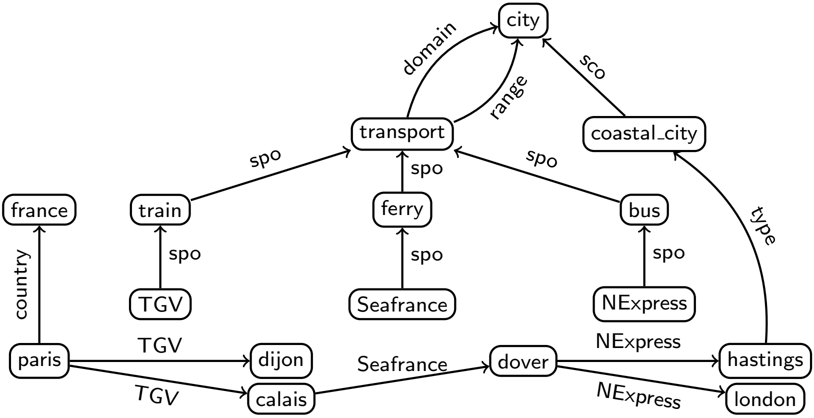
Let G be an RDF graph. The sequence $n_1, \ldots, n_k$ is a *path* in G, iff the triples (n_1, n_2, n_3), (n_3, n_4, n_5), ..., (n_{k-2}, n_{k-1}, n_k) are in G.

An example RDF graph



An example RDF graph (continued)

- Omission of namespace prefixes.



Concatenations and Flavored RPL expressions

- *edge-flavored expressions:*

$$\texttt{EDGES}\ \ e_1 \dots e_k$$

- *node-flavored expressions:*

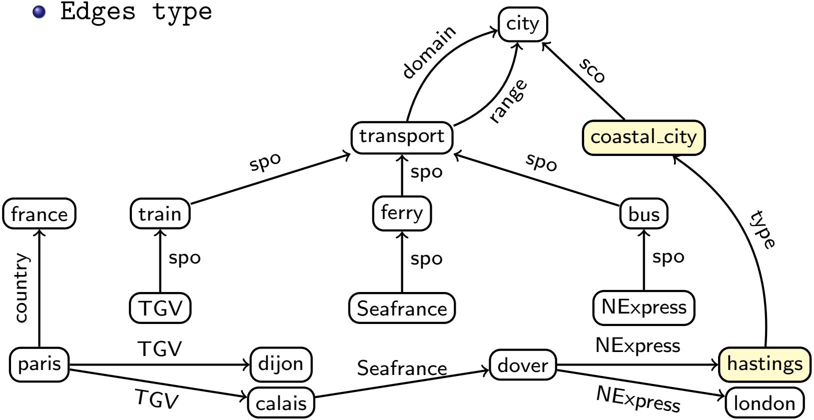
$$\texttt{NODES}\ \ n_1 \dots n_k$$

- *edge-flavored expressions:*

$$\texttt{NODES}\ \ n_1 e_1 \dots e_{k-1} n_k$$

Direct class membership

- Find all pairs of nodes connected over a `type` edge.
- `Edges type`

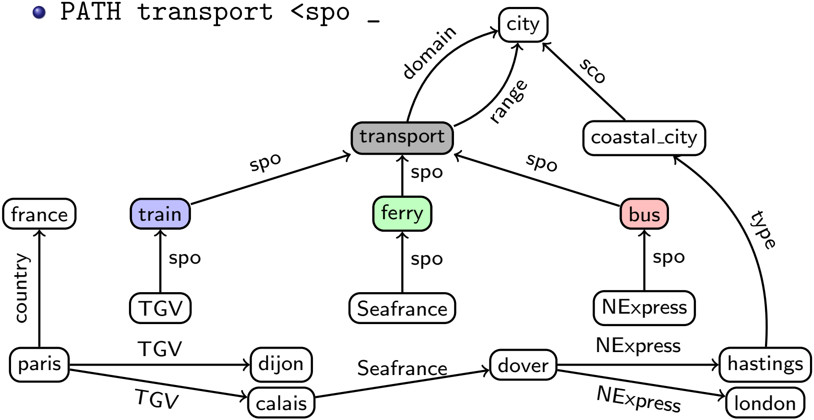


Directed and Adorned Expressions, Wildcards, RSEs, Kleene closure

- RPL expression evaluate to pairs of nodes
- path- and edge-flavored expressions may have *directed edges*
- '_' matches any node or edge
- regular string expressions (RSE) for
  - matching literals only: `/".*"/`
  - matching blank nodes only: `/_:/`
  - matching nodes/edges from a particular domain: `eg:/.*/`
- Kleene closure operators (?, *, +) for matching paths of arbitrary length, qualified descendant

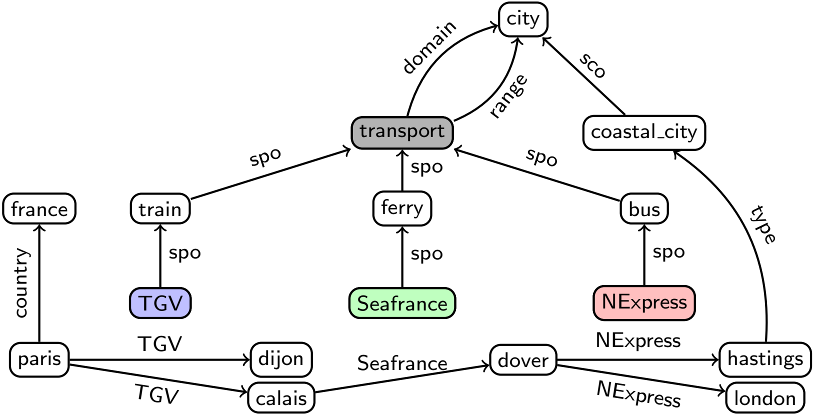
RPL Example Queries (2)

- Find all *direct* subproperties of transport.
- `PATH transport <spo _`



RPL Example Queries (2)

- Find all *indirect* subproperties of transport.
- `PATH transport <spo _ (<spo _)+`



Tree Queries, Predicate Negation and Disjunction

- necessity for tree queries
- nested predicates

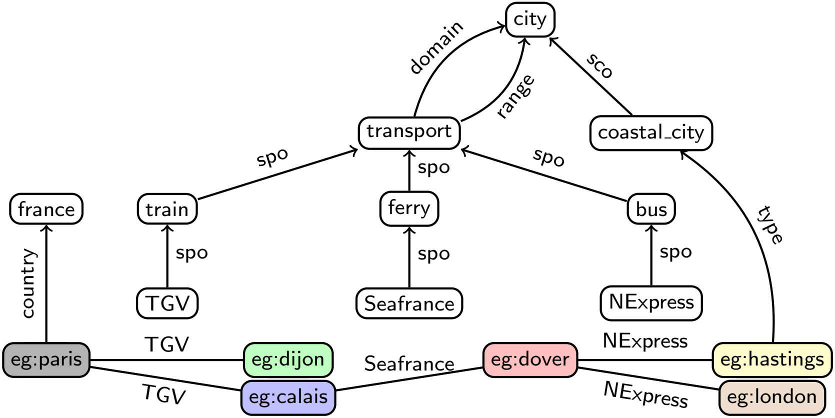
```
EDGES a _[Nodes b[PATH d e f]]
```

- predicates map binary relations on their first components. In Haskell: `predicate list = map fst list`
- predicate negation

```
EDGES a _[not Nodes _[not PATH d e f] ]
```

RPL Example Queries (4)

- Find all cities reachable from Paris.
- `PATH paris (_[PATH (_ spo)* transport] _)+ _`

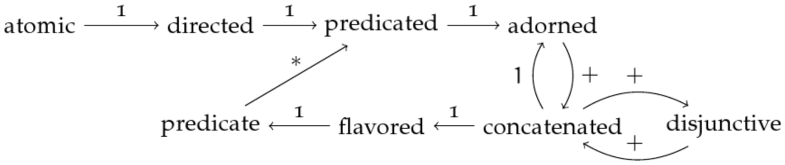


Integration of RPL into Rule Languages

$$\forall x\ p_1\ y\ p_2\ .\ (x\ p_2\ y) \leftarrow (x\ p_1\ y), (p_1\ \texttt{[EDGES sp*]}\ p_2)$$

- RDFLog: simple RDF rule language with node invention and explicit quantifier alternation.
- sp abbreviation for `rdfs:subPropertyOf`

RPL Abstract Syntax



RPL Compositional Semantics

$$
\begin{aligned}
[\![u]\!]^{E,P} &= \{(n_1,n_2) \mid (n_1,u,n_2) \in G\} \\
[\![\_]\!]^{E,P} &= \{(n_1,n_2) \mid \exists p \,.\, (n_1,p,n_2) \in G\} \\
[\![/re/]\!]^{E,P} &= \{(n_1,n_2) \mid \\
&\quad \exists p \in \mathcal{L}(re) \,.\, (n_1,p,n_2) \in G\} \\
[\![>pe]\!]^{X} &= [\![pe]\!]^{X} \textit{ for } X \in \{E,P\} \\
[\![<pe]\!]^{X} &= \{(n_2,n_1) \mid \\
&\quad (n_1,n_2) \in [\![>pe]\!]^{X}\} \textit{ for } X \in \{E,P\} \\
[\![e_1 \ \ldots \ e_k]\!]^{E} &= \{(n_1,n_{k-1}) \mid \exists n_2,\ldots n_k \, (\\
&\quad \forall 1 \leqslant i \leqslant k \, ((n_i,n_{i+1}) \in [\![e_i]\!]^{E}) \,) \, \} \\
[\![(e_1 \mid \ldots \mid e_k)]\!]^{X} &= [\![e_1]\!]^{X} \cup \ldots \cup [\![e_k]\!]^{X} \textit{ for } X \in \{P,E,N\} \\
[\![a[f_1]\ldots[f_k]]\!]^{E} &= \bigcup_{a' \in [\![a[f_1]\ldots[f_k]]\!]^{V}} [\![a']\!]^{E} \\
[\![\epsilon]\!]^{E,P} &= \{(n,n) \mid n \in N\} \\
[\![e+]\!]^{X} &= [\![e]\!]^{X} \cup [\![e\ e+]\!]^{X} \textit{ for } X \in \{P,E,N\} \\
[\![e*]\!]^{X} &= [\![\epsilon]\!] \cup [\![e+]\!]^{X} \textit{ for } X \in \{P,E,N\} \\
[\![e?]\!]^{X} &= [\![\epsilon]\!] \cup [\![e]\!]^{X} \textit{ for } X \in \{P,E,N\}
\end{aligned}
$$

Efficient RPL Evaluation

Definition (RPL evaluation problem)

Given an RDF graph G, a pair of nodes $(n_1, n_2) \in G$, and an RPL expression e, decide if $(n_1, n_2) \in [\![e]\!]$ over G.

Theorem (complexity of RPL)

The evaluation problem for RPL is in $O(n \cdot p)$ where n is the size of the RDF graph and p is the size of the path expression.

Efficient RPL Evaluation

- Idea:
  - Let $E(\phi)$ be the set of predicated and atomic subexpressions of an RPL expression ϕ.
  - Then ϕ can be written as a regular expression over the vocabulary $E(\phi)$.
  - Apply a labeling algorithm (following slide)
- Example:
  - $\phi :=$ `PATH a (<b[PATH c d+ e] _)* (_ e[EDGES f]?)`
  - $E(\phi) = \{$`a` , `<b[PATH c d e]`, _, `e[EDGES f]`$\}$
  - set $\tau :=$ `<b[PATH c d e]` and $\sigma :=$ `e[EDGES f]`
  - then $\phi =$ `PATH a (`τ `_)* (_` σ`)?`

Efficient RPL Evaluation: Labeling Algorithm

1. For each unnested RDF subexpression ψ of ϕ do:
2. Convert ψ into an NFA $NFA(\psi) = (Q, \Sigma, \delta, q_0, F)$.
3. Compute (a variant of) the product automaton $P(\psi, G)$ of $NFA(\psi)$ and the RDF graph G.
    - For details see [BFL09] and [PAG08].
4. If a state (u, q_0) reaches a state (v, q_f) in $P(\psi, G)$, then add the label ψ to u in G.

Complexity:

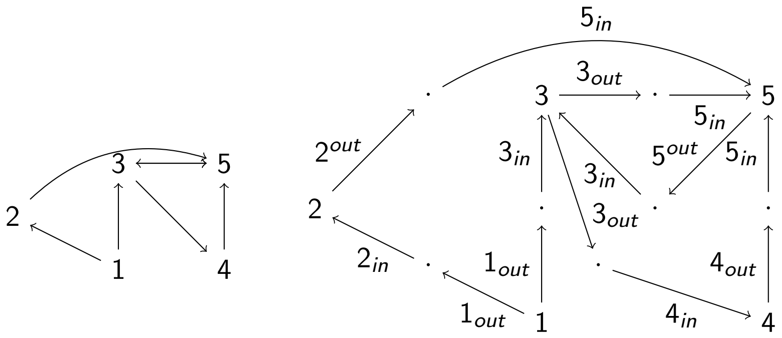
- Step 2: $O(|\psi|)$,
- Step 3: $O(|\psi| \cdot |G|)$,
- Step 4: $O(|\psi| \cdot |G|)$ (depth first search)

Extensions of RPL

- extension to *incomplete paths* is just syntactic sugar
- extension to *unordered paths* is NP-complete
  - for node-flavored RPL expressions
  - for edge-flavored RPL expressions
- extension for *consuming variables* is straightforward

RPL with Unordered Paths

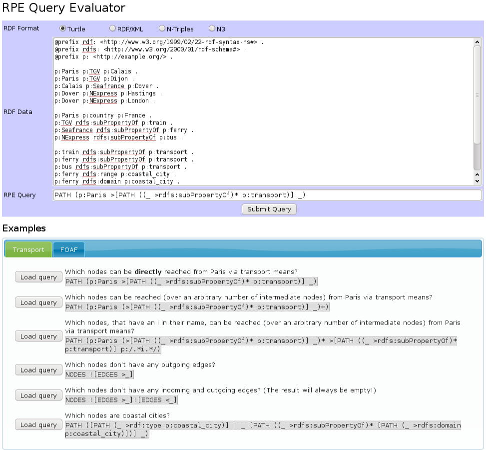
- Is there a path in a Hamiltonian Cycle in G?
- G contains a Hamiltonian cylce iff $\{$ `EDGES` $1_{in}, 1_{out}, \ldots, k_{in}, k_{out}$ $\}$ does not evaluate to $\emptyset$ over the *edge-expansion graph* of G.



Conclusion

- necessity for RDF path query languages
- Features of RPL
  - embedabble in RDF rule languages
  - regular string expressions
  - Kleene closure
  - nested expressions
  - negation
  - clear syntax and formal semantics
  - efficient evaluation

RPL online Demonstration



available at `http://vatulele.pms.ifi.lmu.de:8180/rpe/`

Francois Bry, Tim Furche, and Benedikt Linse.
The perfect match: RPL and RDF rule languages.
In *Proceedings of the third international conference on Web reasoning and rule Systems*. Springer, 2009.

M. Marx.
Conditional XPath.
ACM Transactions on Database Systems (TODS), 30(4):929–959, 2005.

Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez.
nSPARQL: A navigational language for RDF.
In Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan, editors, *International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 2008.