The Perfect Match: RPL and RDF Rule Languages

François Bry, Tim Furche, Benedikt Linse

Institute for Informatics, University of Munich, Germany

October 26, 2009

30.00

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

- 同 ト - ヨ ト - - ヨ ト

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

- 同 ト - ヨ ト - - ヨ ト

Overview

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

Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

< ロ > < 同 > < 回 > < 回 >

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) , \ldots , (n_{k-2}, n_{k-1}, n_k) are in G.

Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

э

An example RDF graph



François Bry, Tim Furche, <u>Benedikt Linse</u> The Perfect Match: RPL and RDF Rule Languages

Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

< ロ > < 同 > < 回 > < 回 >

An example RDF graph (continued)

• Omission of namespace prefixes.



Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

Concatenations and Flavored RPL expressions

edge-flavored expressions:

EDGES $e_1 \ldots e_k$

• node-flavored expressions:

NODES $n_1 \ldots n_k$

• edge-flavored expressions:

NODES $n_1e_1\ldots e_{k-1}n_k$

Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

(日) (同) (三) (三)

Direct class membership

• Find all pairs of nodes connected over a type edge.



Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

イロト イポト イラト イラト

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

Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

イロト イポト イヨト イヨト

RPL Example Queries (2)

• Find all direct subproperties of transport.



Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

RPL Example Queries (2)

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



Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

- 4 回 ト 4 ヨト 4 ヨト

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]]

Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

RPL Example Queries (4)

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



Concatenations and Flavored RPL expressions Directed and Adorned Expressions, Wildcards, Kleene Closure Tree Queries, Predicate Negation and Disjunction

・ 戸 ト ・ ヨ ト ・ ヨ

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



- 4 同 6 4 日 6 4 日 6

RPL Compositional Semantics

 \llbracket

∃ ▶

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 $(n_1, n_2) \in \llbracket e \rrbracket$ 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.

・ 同 ト ・ ヨ ト ・ ヨ ト

Efficient RPL Evaluation

• Idea:

- Let E(φ) 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:

- $\phi := PATH a (< b[PATH c d+ e] _)* (_ e[EDGES f]?)$
- $E(\phi) = \{a, \langle b[PATH \ c \ d \ e], _, e[EDGES \ f]\}$
- set $\tau := \mathsf{<b}\,[\mathsf{PATH}\ \mathsf{c}\ \mathsf{d}\ \mathsf{e}]$ and $\sigma := \mathsf{e}\,[\mathsf{EDGES}\ \mathsf{f}]$
- then $\phi = PATH a (\tau _) * (_ \sigma)?$

Efficient RPL Evaluation: Labeling Algorithm

- **(**) For each unnested RDF subexpression ψ of ϕ do:
- **2** Convert ψ into an NFA $NFA(\psi) = (Q, \Sigma, \delta, q_0, F)$.
- Solution Compute (a variant of) the product automaton $P(\psi, G)$ of $NFA(\psi)$ and the RDF graph G.
 - For details see [BFL09] and [PAG08].
- 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(|ψ|),
- Step 3: $O(|\psi|\cdot|G|)$,
- Step 4: $O(|\psi| \cdot |G|)$ (depth first search)

伺 ト イ ヨ ト イ ヨ ト

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

RPE Query Evaluator

RDF Format	 Turtle 	O RDF/XML	○ N-Triples	○ N3
RDF Data	@prefix <u>rdf</u> : @prefix <u>rdfs</u> @prefix p: <	<http: www.w3.<br="">: <http: www.w3<br="">http://example.c</http:></http:>	org/1999/82/22 Lorg/2000/01/r rg/> .	2-ndf-syntax-nsa⊳ . rdf-schamaer .
	p:Paris p:TG p:Paris p:TG p:Calais p:S p:Dover p:NE p:Dover p:NE	Y p:Calais . Y p:Dijon . eafrance p:Dover xpress p:Hasting xpress p:London	s .	
	p:Paris p:co p:TGV rdfs:s p:Stafrance p:NExpress r	untry p:France ubPropertyOf p:t rdfs:subProperty dfs:subPropertyO	rain . Of p:ferry . If p:bus .	
	p:train_rdfs p:ferry_rdfs p:bus_rdfs:s p:ferry_rdfs p:ferry_rdfs	subPropertyOf p subPropertyOf p ubPropertyOf p:t range p:coastal domain p:coasta	transport . transport . ransport . _city . L_city .	
RPE Query	PATH (p:Par	is >[PATH ((_	>rdfs:subPro	opertyOf)* p:transport)] _)
				Submit Query

Examples

Transport F0.4F
Load quary Which nodes can be directly reached from Paris via transport means? <u>path</u> (p:Paris >[PATH ((_>rdfs:subPropertyof)* p:transport)]_]
Load query Which nodes can be reached (over an arbitrary number of intermediate nodes) from Paris via transport means? PATH (p:Paris (>[PATH ((_ >rdfs:subPropertyOf)* p:transport)] _)+)
Which nodes, that have an in their name, can be reached (over an arbitrary number of intermediate nodes) from Paris Load genery Part (piparis (={Paris (
Load query Which nodes don't have any outgoing edges?
Load query Which nodes don't have any incoming and outgoing edges? (The result will always be empty:)
Which nodes are coastal clies? Load query PATH ([IRTH(srdfitpe pricosstal_city)] _[PATH ((srdfisisk@ropertyOf)* [PATH (srdfisidemain pricosstal_city)))) _]

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

François Bry, Tim Furche, Benedikt Linse The Perfect Match: RPL and RDF Rule Languages

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.

< ロ > < 同 > < 回 > < 回 >