Linked Data Fragments World

Data dumps | Triple patterns | ... | SPARQL endpoints
Linked-data documents | Bindings-restricted triple patterns | Basic graph patterns

Generic requests
High server availability
High client effort

Specific requests
Low server availability
High server effort
Is this really a line?

“Give me all the subjects and objects of triples whose predicate is rdf:type”
Is this really a line?

“Give me all persons reachable from Peter following two foaf:knows links”
A more fundamental understanding of LDF interfaces

Server  Can we analyze an interface before actually implementing it?

Client  What is the best way to use an interface given a specific budget?
A Formal Framework to Compare Linked Data Fragments

Olaf Hartig  Ian Letter  Jorge Pérez
Main contributions

Formal machine model for LDF settings

*based on Turing Machines*

Complete expressiveness lattice

*considering several combinations of interfaces*

Fine grained complexity analysis

*classical complexity, # requests, data transferred*
Linked Data Fragment Machine (LDFM)
Client Oracle

output expression

query

answer

LDFM

response1

response2

... (response containers)

responseK

Turing Machine

Server Oracle

server request

G

(regular tapes)
(\(L_C L_S\))-LDFM

Client Oracle

\(L_C\)

output expression

Server Oracle

\(L_S\)

server request

Turing Machine

(response containers)

(regular tapes)
Client Oracle

Server Oracle

r1 \Join r2

\{\Join\}

\{(\Join), TPF\}-LDFM

r1 = \{ u1, u2, \ldots \}

r2 = \{ v1, v2, \ldots \}

(?X, a, ?Y) \land (?X, b, ?Y)

Turing Machine

(?X, a, ?Y) AND (?X, b, ?Y)
What are the queries computed by \((L_C L_S)\) LDFMs?

\[(L_C L_S) \equiv (R_C R_S)\]

\[(L_C L_S) < (R_C R_S)\]
Expressiveness Lattice
Client Languages

response combinations

subsets of
\{ \cup, \Join, \exists, \pi \}

Server Languages

LDF interfaces

TPF
brTPF
BGP
SPARQL

LDF interfaces
\((\{U, \cap, \cup, \pi\}, \text{TPF})\)

\((\{U\}, \text{TPF})\)

\((\xi\ (\{\cap\}, \text{TPF})\ , \ 'F')\)

\((\emptyset, \text{TPF})\)
\[ (\{U, \exists, \exists, \pi\}, \text{brTPF}) \equiv (\{U, \exists\}, \text{brTPF}) \]

\[ (\{U, \exists, \exists, \pi\}, \text{SPARQL}) \]
\((\{U, \bowtie, \exists, \pi\}, \text{brTPF}) \equiv (\{U, \bowtie\}, \text{brTPF})\)
\((\{U, \bowtie, \exists, \pi\}, \text{SPARQL})\)
Fine-Grained Complexity Analysis
\[(L_C L_S) \prec_T (R_C R_S)\]

in terms of
\[|\text{resp1}| + |\text{resp2}| + \ldots + |\text{respK}|\]

\[(L_C L_S) \prec_R (R_C R_S)\]

in terms of K
A theory for comparing different access protocols for SemWeb data

More fundamental understanding of combinations of LDF interfaces

Machine model + first results on expressiveness and complexity
A theory for comparing different access protocols for SemWeb data

Include new LDF interfaces and client languages

Improve the machine model and consider new metrics

Formal study of SemWeb query planning
A Formal Framework to Compare Linked Data Fragments

Olaf Hartig      Ian Letter      Jorge Pérez
@olafhartig      @ianletter      @perez

Linköping University

Chilean Center for Semantic Web Research
Client Oracle

\{U\}

r2 ∪ r3 ∪ ... ∪ rK

Server Oracle

\(\{U\}, \text{brTPF}\)-LDFM

r1 = \{ u1, u2, ..., uK \}

r2 = \{ v1, v2, ... \}

rK = \{ w1, w2, ... \}

\( (?X,a,?Y) \AND (?X,b,?Y) \)

Turing Machine

G

\text{answer}

r2 ∪ r3 ∪ ... ∪ rK

\text{brTPF}
(\{U, \bowtie, \exists, \pi\}, \text{brTPF}) \equiv (\{U, \bowtie\}, \text{brTPF})
(\{U, \bowtie, \exists, \pi\}, \text{SPARQL})

(\{U, \bowtie, \exists, \pi\}, \text{TPF})
(\{U\}, \text{brTPF})
(\{\bowtie\}, \text{brTPF})

(\{U\}, \text{TPF})
(\{\bowtie\}, \text{TPF})
(\{\bowtie\}, \text{TPF})

(\emptyset, \text{brTPF})
(\emptyset, \text{TPF})

(\emptyset, \text{TPF})