Datalog Relaunched: Simulation Unification and RDFLog

François Bry (University of Munich)

joint work with
Tim Furche (University of Munich)
Clemens Ley (Oxford University)
Bruno Marnette (Oxford University)
Benedikt Linse (Thomson Reuters)
Sebastian Schaffert (Salzburg Research)

Content

1. Motivation
2. Simulation Unification: Unification for Semi-Structured Data
3. RDFLog: Datalog with Value Invention
4. Related Work
5. Perspective: Rich Unification
6. References

1. Motivation – Reasoning on the Web

Datalog is convenient for reasoning on the Web, because

- it is simple and intuitive
- it has solid foundations

Datalog is however lacking

- **data extraction from semi-structured data** (needed for HTML and XML)
- **value invention** (needed e.g. for RDF)

Motivation – XML Data Extraction in Datalog

book elements at any depth containing at any depth and in any order:

- a *title* element containing "Datalog“
- an *author* element containing "Ann“

Standard solution (à la Lixto):

- enhanced Datalog with XPath axes and other constructs
- unbounded traversals (like descendant, following, etc.) expressed in terms of recursion

Drawbacks:

- awkward: much away from a natlang spec.
- error-prone: even termination is uncertain.
- hard to optimize since traversal-based
- mixes up data extraction and reasoning

Motivation – Value Invention on the Web

RDF

- Blank nodes amount to existentially quantified variables
- Blank nodes are needed in Datalog rule heads:
  - for constructing containers and collections
  - for reifying

Blank node in a Datalog rule head amounts to value invention

Value invention is useful in distributed reasoning

- for a local, incomplete construction of complex objects
- for glue-ing incomplete parts of complex objects

2. Simulation Unification – Queries

Incomplete query specifications:

- **bib[book[[]], book[[]]]**
  bib with exactly two books each with whatever content
- **bib[[book [[title["Datalog"]]]**
  bib with at least a book with title "Datalog"
- **bib{{ book[[title["Datalog"]], book[[author["Ann"]] }}**
  bib with a least two books in any order
- **book[[$Author is author[[first["Ann"], last["Abel"]]], $X]**
  possibly constrained variables

Additional constructs:

- **desc, position, optional,**
- **without (book[[without title["Datalog"]]])**
- **except ($Author is author[[first["Ann"], except last["Abel"]]])**

2. Simulation Unification – Declarative Semantics

Simulation:

A ground term t_1 simulates into a ground term t_2 iff the labels and the structure of t_1 can be mapped into t_2

Rooted simulation:

in addition, the root of t_1 is mapped onto the root of t_2

Simulation Unification (declarative semantics):

query t_1 "simulation unifies" into query t_2 iff every ground instances of t_1 "root simulate" in a ground instance of t_2

2. Simulation Unification – Algorithm

With $\mathbf{v_1}$ is mapped onto $\mathbf{v_2}$ in presence of:

- $\mathbf{v_1}$**[...]** and $\mathbf{v_2}$**[...]** the children mapping is bijective and index monotonic
- $\mathbf{v_1}$**{ ... }** and $\mathbf{v_2}$**[...]** the children mapping is bijective
- $\mathbf{v_1}$**{ ... }** and $\mathbf{v_2}$**{ ... }** the children mapping is bijective
- $\mathbf{v_1}$**[[...]]** and $\mathbf{v_2}$**[...]** the children mapping is index monotonic
- $\mathbf{v_1}$**[[...]]** and $\mathbf{v_2}$**[[...]]** the children mapping is index monotonic
- $\mathbf{v_1}$**{{ ... }}** the children mapping is injective

further conditions for variables and for **descendant**, **position**, **optional**, **without** and **except**

2. Simulation Unification – Properties

Simulation unification is decidable, sound, complete, has polynomial data and combined (time) complexities:

- without **[[…]]**, **{{…}}** and variables: **O(q × d)** (q query size, d data size)
- with variables: **NP hard** (PSPACE complete?)

Simulation subsumption (query containment under simulation unification) has time complexities:

- without **[[…]]**, **{{…}}**, **optional** and **except**: **O(n)**
- otherwise: **O(n!^{n})**

Simulation unification achieves a complexity for incomplete XML queries with variables similar to that of XPath 2.0.

3. RDFLog – Quantifier Alternation

Need for rules with arbitrary quantifier alternation

- ∃ ∀ : "Someone knows each professor“
- ∀ ∃ : "Each lecture must be practiced in a tutorial“
- ∀ ∃ ∀ : "Each lecture has a tutorial attended by all students attending the lecture“

RDFLog: an RDF Datalog with arbitrary quantifier alternation

blank nodes may occur in the scope of all, some, or none of the universal variables of a rule.

Datalog's decidability is lost by living the ∃*∀* fragment

or additional provisos needed such as Lewis‘ freeness of Cycles of pseudo-unifiability, Fagin et al. weak acyclicity, Marnette's super weak acyclicity, Calis-Gottlob's guards for "taintable" variables, etc.

3. RDFLog – Declarative Semantics

RDFLog's semantics is closed (every answer is an RDF graph), but lifts RDF's restrictions on blank nodes for intermediary data.

RDFLog's declarative semantics defined in terms of RDF entailment (a model theory might result in illegal RDF expressions)

Arbitrary quantifiers alternations over RDF graphs by expressing RDF graphs as formulas

(Possibly infinite) RDF graphs expressed as (possibly infinite) formulas

3. RDFLog – Operational Semantics

RDFLog's operational semantics:

- Skolemization
- Standard Datalog evaluation
- Un-Skolemization

RDFLog's operational semantics is sound and complete

RDFLog/Datalog with full quantifier alternation and

RDFLog/Datalog with $\forall * \exists *$ prefixes are equivalent

4. Related Work

Simulation Unification is closely related to **XPath Forwards** but applies to graph-shaped documents

RDFLog

a light-weight, ad hoc implementation outperforms both ARQ SPARQL and Sesame SPARQL (with main memory store) on ∀ ∀ and ∀ ∃ ∀ sirups and up to 14.000 triples

SPARQLog

- variant of RDFLog with SPARQL syntax
- covers collections of RDF graphs
- usefull for rewriting rules with full quantifier alternation to ∀ * ∃ * rules

5. Perspective: Rich Unifications

Generalizations of simulation unification for all kinds of complex objects:

- asymmetrical "embedding" relation
- injectivity
- decidability
- recursively defined on structured data

References

François Bry, Tim Furche, and Benedikt Linse: ***Simulation Subsumption or Déjà vu on the Web***. Proc, Int. Conf. on Web Reasoning and Rule Systems (RR2008), LCNS, 2008

François Bry, Tim Furche, Clemens Ley, Benedikt Linse, and Bruno Marnette: ***RDFLog: It's like Datalog for RDF.*** Proc. 22nd Workshop on (Constraint) Logic Programming (WLP 2008), 2008

Sebastian Schaffert and François Bry: ***Querying the Web Reconsidered: A Practical Introduction to Xcerpt.*** Proc. Extreme Markup Languages, 2004

François Bry and Sebastian Schaffert: ***Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification.*** Proc. Int. Conf. on Logic Programming (ICLP), LNCS 2401, Springer-Verlag, 2002