rdflog—Taming Existence

A Logic-based Query Language for RDF

Authors (A)
François Bry, Tim Furche
Clemens Ley, Benedikt Linse

Physical Home (I)
Institut für Informatik

Digital Home (U)
pms.ifi.lmu.de

Example: Filling Containers 1

```
FORALL City City' City'' Country
IF     City    locatedIn    Country,
       City'   locatedIn    Country,
       City    <_lex        City',
   NOT( City'' locatedIn    Country,
       City    <_lex        City'',
       City''  <_lex        City' )
THEN   City    predecessor  City'
```

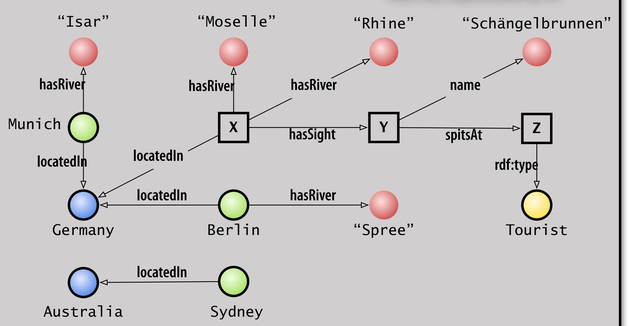
```
FORALL Bag City City'
IF     Bag     rdf:_n       City,
       City    predecessor  City'
THEN   Bag     rdf:_(n+1)   City'
```

```
FORALL Country EXISTS Bag FORALL City City'
IF     City    locatedIn    Country,
   NOT City'   predecessor  City
THEN   Bag     rdf:_1       City,
       Bag     rdf:type     rdf:Bag,
       Country cities       Bag
```

— **for each** country, creates a **container** of its cities
— note *similarity* between the natural language and the rdflog formulation in rule 2
— container resources are, as recommended by the RDF specification, **existential variables**
— **Bag** is one of the three container types of RDF

RDF Data Graph 0

— on cities and rivers of the World
— can you guess the name of the third city (represented by X)?



Legend (Data): X existential variable; country; city; class; literal

Example: Gathering Collections 1

```
FORALL City City' City'' Country
IF     City    locatedIn    Country,
       City'   locatedIn    Country,
       City    <_lex        City',
   NOT( City'' locatedIn    Country,
       City    <_lex        City'',
       City''  <_lex        City' )
THEN   City    predecessor  City'
```

```
FORALL Coll EXISTS Coll'
       FORALL City City'
IF     Coll    rdf:first    City,
       City    predecessor  City'
THEN   Coll    rdf:rest     Coll',
       Coll'   rdf:first    City
```

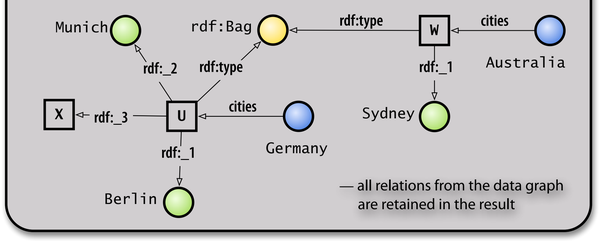
```
FORALL City EXISTS Coll
       FORALL City' Country
IF     City    locatedIn    Country,
   NOT( City'  locatedIn    Country,
       City'   predecessor  City )
THEN   Coll    rdf:first    City
```

```
FORALL Coll City City'
IF     Coll    rdf:first    City,
   NOT City    predecessor  City'
THEN   Coll    rdf:rest     rdf:nil
```

RDF Data Graph as Formula 3

$$\begin{aligned}&\textbf{locatedIn}(\text{Munich},\text{Germany})\wedge\textbf{hasRiver}(\text{Munich},\text{“Isar”})\wedge\\&\textbf{locatedIn}(\text{Berlin},\text{Germany})\wedge\textbf{hasRiver}(\text{Berlin},\text{“Spree”})\wedge\\&\textbf{locatedIn}(\text{Sydney},\text{Australia})\wedge\\&\exists x\big(\textbf{locatedIn}(x,\text{Germany})\wedge\\&\quad\textbf{hasRiver}(x,\text{“Rhine”})\wedge\textbf{hasRiver}(x,\text{“Moselle”})\wedge\\&\quad\exists y(\textbf{hasSight}(x,y)\wedge\textbf{name}(y,\text{“Schängelbrunnen”})\wedge\\&\qquad\exists z(\textbf{spitsAt}(y,z)\wedge\textbf{rdf:type}(z,\text{Tourist}))\big)\end{aligned}$$

Filled Containers 2



— all relations from the data graph are retained in the result

Model Theoretic Semantics 1

– Ψ an RDFLOG program.

$$[\![\Psi]\!]^M := \{\varphi \in \text{RDF} \mid \Psi \models \varphi\}$$

Fixed Point Semantics 2

– Ψ an RDFLOG program, φ an RDF graph,
– $(Q_1 x_1 \dots Q_n x_n . t_1 \wedge \dots \wedge t_n \to t) = r \in \Psi$ an RDFLOG rule,
– S skolemisation for $\Psi \cup \{\varphi\}$, σ substitution for the universal var.s in r.

$$\text{cons}_{r,\sigma}(\varphi) := \begin{cases}\varphi \cup (t^S\sigma)^{(S\sigma)^{-1}} & \text{if } \{t_1,\dots,t_n\}^S\sigma \subseteq \varphi^S \\ \varphi & \text{otherwise}\end{cases} \qquad T(\varphi) := \bigcup_{r\in\Psi}\bigcup_{\sigma\in\text{Subst}} \text{cons}_{r,\sigma}(\varphi)$$

$$[\![\Psi]\!]^{FP} := \text{lfp}(T)$$

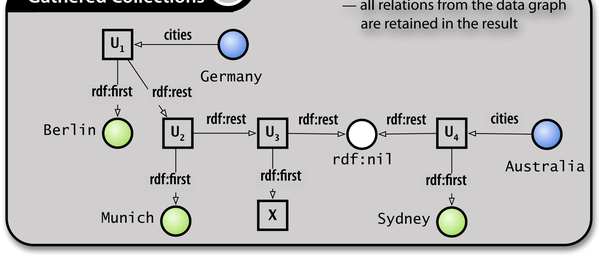
Operational Semantics 3

– Ψ an RDFLOG program,
– S a skolemisation for Ψ,
– $[\![P]\!]^L$ the operational semantics of a logic program P, and
– $\sigma(\varphi)$ the substitution used in the proof of φ from Ψ^S.

$$[\![\Psi]\!]^O := \{\varphi^{(S\sigma(\varphi))^{-1}} \mid \varphi \in [\![\Psi^S]\!]^L\}$$

Gathered Collections 2

— all relations from the data graph are retained in the result



Commutativity of leaner and cons 4

– r an RDFLOG rule,
– σ a substitution for the universal variables in r, and
– τ a substitution for the existential variables in φ.

$$\text{exvar}(r) \cap \text{dom}(\tau) \cap \text{ran}(\tau) = \emptyset \implies \text{leaner}_\tau \circ \text{cons}_{r,\sigma} = \text{cons}_{r,\sigma} \circ \text{leaner}_\tau$$

Soundness & Completeness 5

– Ψ an RDF program.

$$[\![\Psi]\!]^O \subseteq [\![\Psi]\!]^M \qquad \text{(soundness)}$$
$$\forall \varphi \in [\![\Psi]\!]^M\ \exists \psi \in [\![\Psi]\!]^O\ \exists \sigma \in \text{Subs}(\text{exvar}(\varphi))\ .\ \varphi\sigma \models\!\mid \psi \qquad \text{(completeness)}$$

Skolemisation & Entailment 6

– Ψ a set of formulas,
– φ a formula, and
– S a skolemisation.

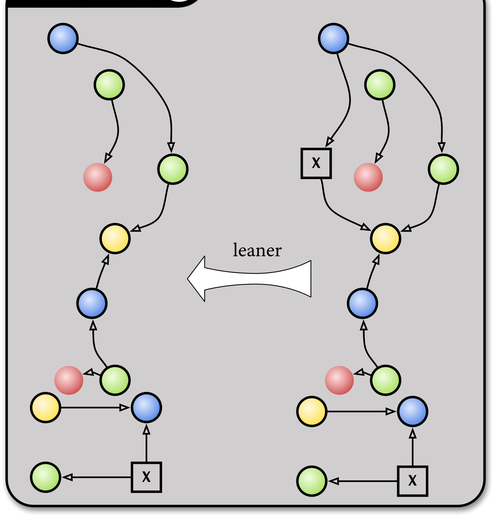
$$\Psi \models \varphi \Leftrightarrow \Psi^S \models \varphi^S$$

Leaner, Lean Operator 7

– φ an RDF graph and
– σ a substitution for the existential variables in φ.

$$\text{leaner}_\sigma(\varphi) := \begin{cases}\varphi\sigma & \text{if } \varphi\sigma \subseteq \varphi \\ \varphi & \text{otherwise}\end{cases} \qquad L(\varphi) := \bigcap_{\sigma\in\text{Subst}} \text{leaner}_\sigma(\varphi)$$

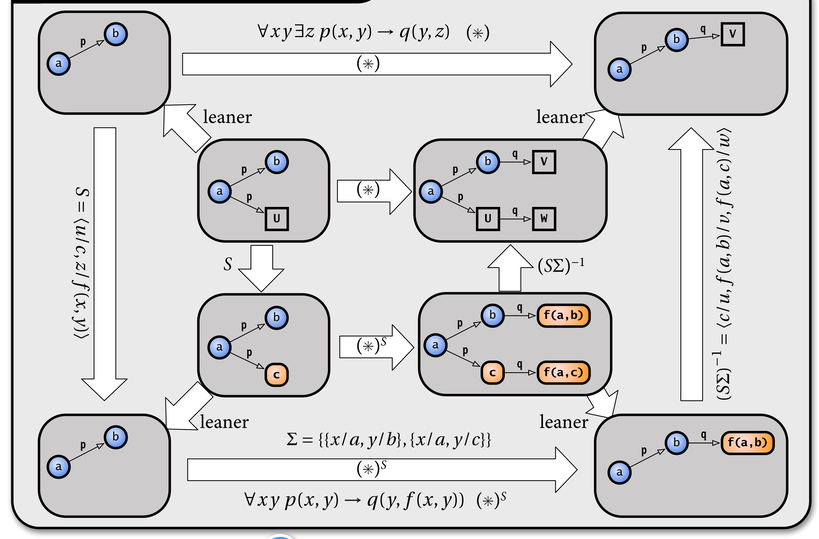
Pat & Patachon 4



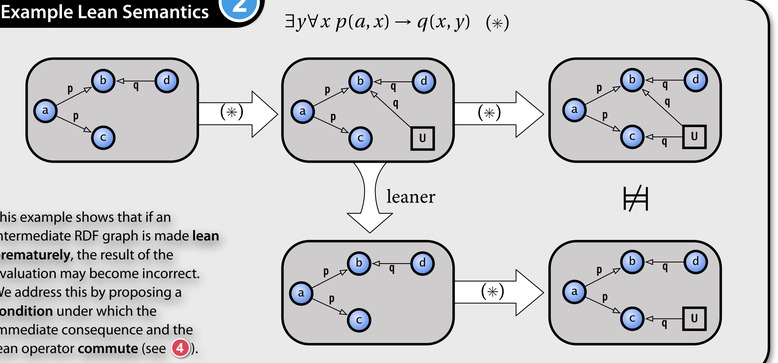
rdflog—Rules & Programs

—An **rdflog rule** is a Horn clause *extended with existentially* quantified variables in the head.
—An rdflog rule has to be *range restricted* wrt. its universally quantified variables.
—An **rdflog program** is a *stratifiable* set of rdflog rules.

Example of Operational Semantics 1



Example Lean Semantics 2



Example Lean Semantics 3