SPAR²QL: From SPARQL to Rules

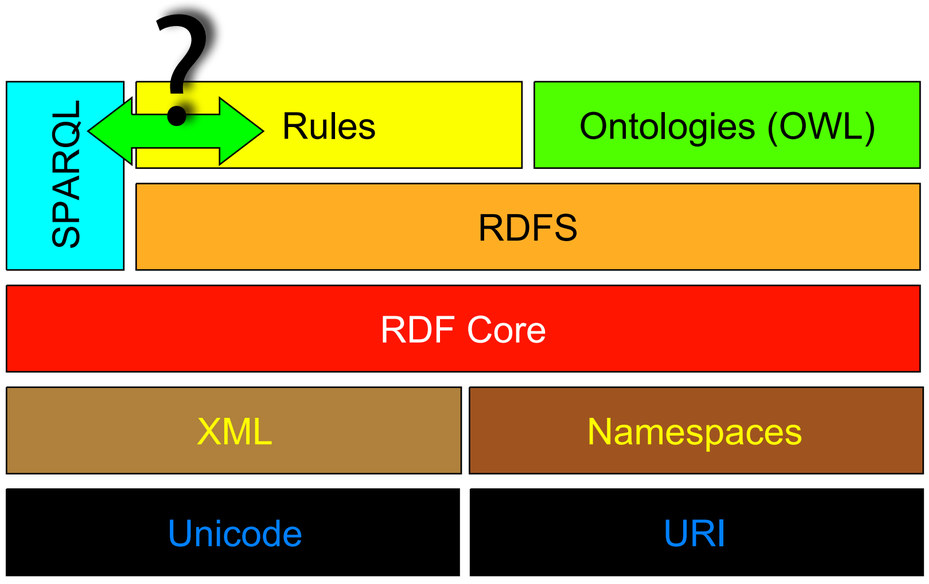
Axel Polleres (Universidad Rey Juan Carlos, Móstoles, Madrid, Spain) **http://www.polleres.net**
Joint work with: **Roman Schindlauer**, TU Vienna,
supported by the Spanish MEC, by the EC funded projects TripCom (FP6-027324)
and REWERSE (IST-2003-506779), and by the Austrian Science Funds.

SPARQL:

- The upcoming standard for an RDF query language.
- Currently candidate recommendation in W3C DAWG WG
- Somewhat reminds of N3 and SQL in its syntax

Current Obstacles:

- Few implementations available
- Implementations have problems with peculiarities (see [Perez et al., 2006])
- Combination with the Rules Layer (W3C RIF WG) unclear




```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage
FROM <file:examples/spec5.3.ttl>
WHERE { ?x foaf:name ?name .
        OPTIONAL { ?x foaf:mbox ?mbox } .
        OPTIONAL { ?x foaf:homepage ?hpage }
      }
```

Datalog with Stratified Negation

- Well-known to capture big part of SQL
- Efficient engines handle aggregates, external predicates, such as import of RDF data, arithmetics, etc.

Our Approach:

Use **translation** similar to the one for SQL in order to serve to implement a **SPARQL engine!**

Benefits:

- Clarify the **relation to Rules**
- Translated ruleset is **embeddable/combinable with rules** (SPARQL queries in Rule bodies, additional features such as **nesting, aggregates** doable)

Core translation function $\tau(\overline{vars},P,D,i)$

case $P = (s\ p\ o.)$ return $\{\texttt{answer}_i(\overline{vars})\ \texttt{:-}\ \texttt{triple}(s,p,o,D).\}$

case $P = (P'\ \text{AND}\ P'')$ return

$\{\texttt{answer}_i(\overline{vars})\ \texttt{:-}\ \texttt{answer}_{2*i}(\overline{(vars \cap vars(P')) \cup (vars(P') \cap vars(P''))}),$
$\texttt{answer}_{2*i+1}(\overline{(vars \cap vars(P'')) \cup (vars(P') \cap vars(P''))}).\}$
$\cup\tau(\overline{(vars \cap vars(P')) \cup (vars(P') \cap vars(P''))}, P', D, 2*i)$
$\cup\tau(\overline{(vars \cap vars(P'')) \cup (vars(P') \cap vars(P''))}, P'', D, 2*i+1)$

case $P = (P'\ \text{OPTIONAL}\ P'')$ return

$\tau(\overline{vars}, P'\ \text{AND}\ P'', D, i) \cup$
$\{\texttt{answer}_i(\overline{vars[vars \backslash vars(P') \rightarrow \texttt{null}]})\ \texttt{:-}\ \texttt{answer}_{2*i}(\overline{(vars \cap vars(P')) \cup (vars(P') \cap vars(P''))}),$
$\texttt{not answer}'_{2*i}(\overline{vars(P') \cap vars(P'')}).$
$\texttt{answer}'_{2*i}(\overline{vars(P') \cap vars(P'')})\ \texttt{:-}\ \texttt{answer}_{2*i+1}(\overline{(vars \cap vars(P'')) \cup (vars(P') \cap vars(P''))}).$

case $P = (P'\ \text{UNION}\ P'')$ return

$\{\texttt{answer}_i(\overline{vars[vars \backslash vars(P') \rightarrow \texttt{null}]})\ \texttt{:-}\ \texttt{answer}_{2*i}(\overline{vars \cap vars(P')}).$
$\{\texttt{answer}_i(\overline{vars[vars \backslash vars(P'') \rightarrow \texttt{null}]})\ \texttt{:-}\ \texttt{answer}_{2*i+1}(\overline{vars \cap vars(P'')}).\}$
$\cup\tau(\overline{vars \cap vars(P')}, P', D, 2*i)$
$\cup\tau(\overline{vars \cap vars(P'')}, P'', D, 2*i+1)$

dlvhex
HEX-program

SPARQL Semantics

```
#namespace("foaf","http://xmlns.com/foaf/0.1/")

triple(S,P,O,default) :- &rdf["file:examples/spec5.3.ttl"](S,P,O).

answer_1(X_name,X_x)  :- triple(X_x,"foaf:name",X_name,default).
answer_2(X_mbox,X_x)  :- triple(X_x,"foaf:mbox",X_mbox,default).
answer_3(X_hpage,X_x) :- triple(X_x,"foaf:homepage",X_hpage,default).

answer_opt_1(X_mbox,X_name,X_x) :- answer_1(X_name,X_x), answer_2(X_mbox,X_x).
answer_opt_1(null  ,X_name,X_x) :- answer_1(X_name,X_x), not answer_2'(X_x).
answer_2'(X_x) :- answer_2(X_mbox,X_x).

answer_opt_2(X_hpage,X_mbox,X_name,X_x) :- answer_opt_1(X_mbox,X_name,X_x), answer_3(X_hpage,X_x).
answer_opt_2(null   ,X_mbox,X_name,X_x) :- answer_opt_1(X_mbox,X_name,X_x), not answer_3'(X_x).
answer_3'(X_x) :- answer_3(X_hpage,X_x).

answer(X_hpage,X_mbox,X_name) :- answer_opt_2(X_hpage,X_mbox,X_name,X_x).
```

Prototype available at:
http://con.fusion.at/dlvhex/sparql-query-evaluation.php

Current Prototype Features:

- SELECT queries with simple graph patterns, UNION and OPTIONAL
- N3, RDF/XML, Turtle syntax for input
- Full N3 syntax for triple patterns, including
  - blank nodes
  - arbitrary nesting of patterns
- Simple conjunctive FILTER expressions:
  - conjunction (&&)
  - isBound
  - isBlank
  - binary comparison operators (=, =<, >=, !=, <, >)

Limitations:

- CONSTRUCT, ASK and DESCRIBE result forms not yet supported.
- Solution modifiers not supported (due to their non-declarative aspects)
- FILTERs support still preliminary
- lack of language-tag awareness or (typed) literals
- Output not yet SPARQL protocol conformant.

Future work:

- Integration with Rule bases (e.g. N3 Rules)
- Adding useful features to SPARQL (Recursion, aggregates, nested queries)