Towards Comparing RDF Stream Processing Semantics

Minh Dao-Tran    Harald Beck    Thomas Eiter

1st Workshop on High-Level Declarative Stream Processing
September 22, 2015
RDF STREAM PROCESSING COMMUNITY GROUP

The mission of the RDF Stream Processing Community Group (RSP) is to define a common model for producing, transmitting and continuously querying RDF Streams. This includes extensions to both RDF and SPARQL for representing streaming data, as well as their semantics. Moreover, this work envisions an ecosystem of streaming and static RDF data sources whose data can be combined through standard models, languages and protocols. Complementary to related work in the area of databases, this Community Group looks at the dynamic properties of graph-based data, i.e., graphs that are produced over time and which may change their shape and data over time.
Recent Developments in RSP

RSP Query Engines:
- C-SPARQL [Barbieri et al., 2010]
- CQELS [Phuoc et al., 2011]
- SPARQL\textsubscript{Stream} [Calbimonte et al., 2010]
- ...

Benchmarking systems:
- LSBench [Phuoc et al., 2012]
- SRBench [Zhang et al., 2012]
- CSRBench [Dell’Aglio et al., 2013]
- YABench [Kolchin and Wetz, 2015]
Recent Developments in RSP

RSP Query Engines:
- C-SPARQL [Barbieri et al., 2010]
- CQELS [Phuoc et al., 2011]
- SPARQL\textsubscript{Stream} [Calbimonte et al., 2010]
- ...

Benchmarking systems:
- LSBench [Phuoc et al., 2012]
- SRBench [Zhang et al., 2012]
- CSRBench [Dell’Aglio et al., 2013]
- YABench [Kolchin and Wetz, 2015]

All comparison at the operational level!
Comparison at the Semantics Level?

... EXCEPT LARS OF MARS!
Running scenario

RDF & RDF Stream Processing (RSP)

A Logic for Analyzing Reasoning over Streams (LARS [Beck et al., 2015])

Capturing RSP with LARS
Running Scenario
Running Scenario
Running Scenario

30%

25%
Running Scenario
Running Scenario

30%

25%
RDF

Subject
Predicate
Object

G = {

"mbt": g classify: 1,
"rayban": g classify: 0,

}

g1 = {

a: offers: c1,
"c1": on: "mbt",
"c1": reduce: 30,

}

g2 = {

b: offers: c2,
"c2": on: "rayban",
"c2": reduce: 25,

}

g3 = {

"claire": isNear: a,
"claire": isNear: b,

}

RDF

\[ G = \{ \text{"mbt" : g\_classify : 1. \ "rayban" : g\_classify : 0. ... } \} \]
\[ g_1 = \{ \text{a : offers : c}_1. \ \text{c}_1 : \text{on : "mbt". \ \text{c}_1 : \text{reduce : 30.} } \} \]
\[ g_2 = \{ \text{b : offers : c}_2. \ \text{c}_2 : \text{on : "rayban". \ \text{c}_2 : \text{reduce : 25.} } \} \]
\[ g_3 = \{ \text{"claire" : isNear : a. \ \text{"claire" : isNear : b.} } \} \]
SELECT ?shop ?product ?percent
WHERE {
?user :isNear ?shop.
FILTER (?percent >= 20 && ?gender != 1)
}
µ = {?shop ↦→ a, ?product ↦→ "rayban", ?percent ↦→ 25}
SPARQL

SELECT ?shop ?product ?percent

FROM

WHERE

\[\mu = \{ ?shop \mapsto a, ?product \mapsto "rayban", ?percent \mapsto 25\}\]
SPARQL

SELECT ?shop ?product ?percent
FROM <http://products>
<http://coupons_snapshot>
<http://locations_snapshot>
WHERE

  ?user :isNear ?shop.
  FILTER (?percent >= 20 && ?gender != 1) }

µ = { ?shop ↦→ a, ?product ↦→ "rayban", ?percent ↦→ 25 }
SELECT ?shop ?product ?percent
FROM <http://products>
    <http://coupons.snapshot>
    <http://locations.snapshot>
WHERE {?shop :offers ?coupon.
    ?user :isNear ?shop.
SPARQL

SELECT ?shop ?product ?percent
FROM <http://products>
<http://coupons_snapshot>
<http://locations_snapshot>
WHERE {?shop :offers ?coupon.
  ?user :isNear ?shop.
FILTER (?percent >= 20 && ?gender != 1)}
SPARQL

SELECT ?shop ?product ?percent
FROM <http://products>
  <http://coupons_snapshot>
  <http://locations_snapshot>
WHERE {?shop :offers ?coupon.
  ?user :isNear ?shop.
FILTER (?percent >= 20 && ?gender != 1)}

μ = {?shop ↦ a, ?product ↦ “rayban”, ?percent ↦ 25}
RDF Stream Processing Queries in C-SPARQL

```
SELECT ?shop ?product ?percent
FROM <http://products>
  <http://coupons_stream> [RANGE 30m]
  <http://locations_stream> [RANGE 5m]
WHERE {?shop :offers ?coupon.
  ?user :isNear ?shop.
FILTER (?percent >= 20 && ?gender != 1)}
```
SELECT ?shop ?product ?percent
FROM <http://products>
WHERE {

<http://coupons_stream> [RANGE 30m] {
}

<http://locations_stream> [RANGE 5m] {
    ?user :isNear ?shop.
}
FILTER (?percent >= 20 && ?gender != 1)\}
# Key Differences between C-SPARQL and CQELS

<table>
<thead>
<tr>
<th></th>
<th>C-SPARQL</th>
<th>CQELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>create snapshot</td>
<td>merge patterns on input streams into the default graph</td>
<td>apply patterns on input streams</td>
</tr>
<tr>
<td>execution mode</td>
<td>pull-based</td>
<td>push-based</td>
</tr>
</tbody>
</table>
Modeling RSP Queries $Q = (V, P, D, S)$
Modeling RSP Queries $Q = (V, P, \mathcal{D}, S)$

\[ V = \{\text{?shop, ?pname, ?percent}\} \]
Modeling RSP Queries $Q = (V, P, D, S)$

\[
V = \{ ?\text{shop}, ?\text{pname}, ?\text{percent} \} \\
P = (P_1 \cup P_2 \cup P_3) \text{ FILTER } R \\
\begin{align*}
P_1 &= \{ \text{?shop} : \text{offers} \ ?\text{coupon}. \} \\
P_2 &= \{ \text{?user} : \text{isNear} \ ?\text{shop}. \} \\
P_3 &= \{ \text{?product} : \text{g\_classify} \ ?\text{gender}. \} \\
\end{align*}
\]
Modeling RSP Queries $Q = (V, P, D, S)$

\[ V = \{ \text{?shop}, \text{?pname}, \text{?percent} \} \]

\[ P = (P_1 \cup P_2 \cup P_3) \text{ FILTER } R \]

\[ P_1 = \{ \begin{align*} &\text{?shop} : \text{offers} \text{ ?coupon.} \\ &\text{?coupon} : \text{on} \text{ ?product.} \\ &\text{?coupon} : \text{reduce} \text{ ?percent.} \end{align*} \} \]

\[ P_2 = \{ \text{?user} : \text{isNear} \text{ ?shop.} \} \]

\[ P_3 = \{ \text{?product} : \text{gclassify} \text{ ?gender.} \} \]

\[ R = (\text{?percent} \geq 20 \land \text{?gender} \neq 1) \]
Modeling RSP Queries $Q = (V, P, D, S)$

\[
V = \{ ?\text{shop}, ?\text{pname}, ?\text{percent} \}
\]

\[
P = (P_1 \cup P_2 \cup P_3) \text{ FILTER } R
\]

\[
P_1 = \begin{cases} 
?\text{shop} &: \text{offers} \ ?\text{coupon}. \\
?\text{coupon} &: \text{on} \ ?\text{product}. \\
?\text{coupon} &: \text{reduce} \ ?\text{percent}. 
\end{cases}
\]

\[
P_2 = \{ ?\text{user} &: \text{isNear} \ ?\text{shop}. \}
\]

\[
P_3 = \{ ?\text{product} &: \text{gclassify} \ ?\text{gender}. \}
\]

\[
R = (?\text{percent} \geq 20 \land ?\text{gender} \neq 1)
\]

\[
D = \{ \langle \text{products} \rangle \}
\]
Modeling RSP Queries \( Q = (V, P, D, S) \)

\[
V = \{ \text{?shop, ?pname, ?percent} \}
\]

\[
P = (P_1 \cup P_2 \cup P_3) \text{ FILTER } R
\]

\[
P_1 = \{ \text{?shop : offers ?coupon.} \}
\]

\[
P_1 = \{ \text{?coupon : on ?product.} \}
\]

\[
P_1 = \{ \text{?coupon : reduce ?percent.} \}
\]

\[
P_2 = \{ \text{?user : isNear ?shop.} \}
\]

\[
P_3 = \{ \text{?product : g_classify ?gender.} \}
\]

\[
R = (?\text{percent} \geq 20 \& \& ?\text{gender} \neq 1)
\]

\[
D = \{ \text{<products>} \}
\]

\[
S = \{ (\text{<http://coupons>}, [\text{RANGE 30m}], P_1),
\]

\[
(\text{<http://locations>}, [\text{RANGE 5m}], P_2) \}
\]
LARS in a Nutshell: Stream Representation

\[ S = (T, v) \]

\[ T = [0, 50] \]

\[ v = \begin{cases} 
40 \mapsto \{offer(a, mbt, 30)\}, & 45 \mapsto \{offer(b, rayban, 25)\} \\
48 \mapsto \{isNear(a), isNear(b)\} 
\end{cases} \]
LARS in a Nutshell: Window functions

\[ S' = w_t(S, t, \vec{x}) \]
LARS in a Nutshell: Window functions

\[ S' = w(S, 48, (5, 0, 1)) = ([43, 48], \left\{45 \mapsto \{\text{offer}(b, \text{rayban}, 25)\},
               48 \mapsto \{\text{isNear}(a), \text{isNear}(b)\}\right\}) \]
LARS in a Nutshell: LARS formulas

\[ \alpha ::= \]

\[ \text{offer}(a, mbt, 30) \quad \text{offer}(b, rayban, 25) \quad \text{isNear}(a) \quad \text{isNear}(b) \]
LARS in a Nutshell: LARS formulas

\[ \alpha ::= a \mid \neg \alpha \mid \alpha \land \alpha \mid \alpha \lor \alpha \mid \alpha \rightarrow \alpha \]
LARS in a Nutshell: LARS formulas

\[ \alpha ::= a \mid \neg \alpha \mid \alpha \land \alpha \mid \alpha \lor \alpha \mid \alpha \to \alpha \mid \Diamond \alpha \mid \Box \alpha \mid \Theta_t \alpha \]

- various ways for time references
LARS in a Nutshell: LARS formulas

\[ α ::= a \mid \neg α \mid α \land α \mid α \lor α \mid α → α \mid \Diamond α \mid \Box α \mid \Thetaₜ α \mid □ x α \]

- various ways for time references
- window operators with possibility to nest
LARS in a Nutshell: LARS formulas

\[ α ::= a | ¬α | α \land α | α \lor α | α → α | ♦α | □α | Θ_t α | \bigotimes^τ α \]

- various ways for time references
- window operators with possibility to nest
  - \[ \bigotimes^10 ♦ offer(Sh, Pr, Pe) \]
LARS in a Nutshell: LARS formulas

\[ \alpha ::= a \mid \neg \alpha \mid \alpha \land \alpha \mid \alpha \lor \alpha \mid \alpha \rightarrow \alpha \mid \Diamond \alpha \mid \Box \alpha \mid \Theta_t \alpha \mid \Box_t^x \alpha \]

- various ways for time references
- window operators with possibility to nest
- \( \Box^{10} \Diamond offer(Sh, Pr, Pe) \)
- \( \Box^5 \Box isNear(a) \)
LARS in a Nutshell: LARS rules/programs

\[ \alpha \leftarrow \beta_1, \ldots, \beta_j, \text{not } \beta_{j+1}, \ldots, \text{not } \beta_n. \]
LARS in a Nutshell: LARS rules/programs

\[ \alpha \leftarrow \beta_1, \ldots, \beta_j, \text{not } \beta_{j+1}, \ldots, \text{not } \beta_n. \]

\[ \text{ans}(Sh, Pr, Pe) \leftarrow \boxplus^{30} \cdot \text{offer}(Sh, Pr, Pe), \boxplus^{5} \cdot \text{isNear}(Sh), \]

\[ g\_\text{classify}(Pr, Ge), Pe \geq 20, Ge \neq 1. \]
LARS in a Nutshell: LARS rules/programs

\[ \alpha \leftarrow \beta_1, \ldots, \beta_j, \text{not} \ \beta_{j+1}, \ldots, \text{not} \ \beta_n. \]

\[ \text{ans}(Sh, Pr, Pe) \leftarrow \bigoplus^{30} \diamond \text{offer}(Sh, Pr, Pe), \bigoplus^5 \diamond \text{isNear}(Sh), \]
\[ g\_\text{classify}(Pr, Ge), \ Pe \geq 20, \ Ge \neq 1. \]

\[ \ominus_{T+5} \text{near}(b) \leftarrow \bigoplus^5 \ominus_T \text{isNear}(a). \]
LARS in a Nutshell: LARS rules/programs

\[ \alpha \leftarrow \beta_1, \ldots, \beta_j, \text{not } \beta_{j+1}, \ldots, \text{not } \beta_n. \]

\[ \text{ans}(Sh, Pr, Pe) \leftarrow \triangleleft^{30} \diamond offer(Sh, Pr, Pe), \triangleleft^{5} \diamond isNear(Sh), \]
\[ \ g \_classify(Pr, Ge), Pe \geq 20, Ge \neq 1. \]

\[ @_{T+5} near(b) \leftarrow \triangleleft^{5} \@_T isNear(a). \]

\[ \triangleleft^{+5} \diamond near(b) \leftarrow \triangleleft^{5} \diamond isNear(a). \]
LARS in a Nutshell: Semantics

- extend Answer Set Programming semantics
- answer streams: input stream + intentional facts + satisfaction + minimality
LARS in a Nutshell: Semantics

- extend Answer Set Programming semantics
- answer streams: input stream + intentional facts + satisfaction + minimality

\[ I = ([0, 50], \nu_I) \text{ where}\]

\[ \nu_I = \begin{cases} 
40 &\mapsto \{offer(a, rayban, 30)\} \\
45 &\mapsto \{offer(b, rayban, 25)\} \\
48 &\mapsto \{isNear(a), isNear(b), ans(b, rayban, 25)\} 
\end{cases} \]
LARS to Analyze RSP Queries

Translations from RSP queries to LARS programs:

- offer the correspondence between query answers and answer streams
- capture differences in RSP queries:
  - execution modes
  - creating snapshots
Capture push-, pull-based execution modes:

\[
\text{trigger}(r) = H(r) \leftarrow B(r), \text{trig.}
\]

\[
\text{trigger}(P) = \{ \text{trigger}(r) \mid r \in P \land B(r) \neq \emptyset \}
\]

\[
\triangledown(P) = \text{trigger}(P) \cup \{ \text{trig} \leftarrow \bigbox@NOW p(\bar{X}). \mid p \in A^I \}
\]

\[
\triangleleft(P, U) = \text{trigger}(P) \cup \{ \text{trig} \leftarrow \bigbox@NOW@T \text{true}, T \% U = 0. \}
\]
Translating at a Glance

Capture push-, pull-based execution modes:

\[
\text{trigger}(r) = H(r) \leftarrow B(r), \text{trig.}
\]

\[
\text{trigger}(P) = \{\text{trigger}(r) \mid r \in P \land B(r) \neq \emptyset\}
\]

\[
\triangledown(P) = \text{trigger}(P) \cup \{\text{trig} \leftarrow \boxplus_{\text{NOW}} p(\vec{X}) \mid p \in A^I\}
\]

\[
\triangleleft(P, U) = \text{trigger}(P) \cup \{\text{trig} \leftarrow \boxplus_{\text{NOW} \circ_T \text{true}, T \% U = 0}\}
\]

Capture SPARQL operators: extend a translation from SPARQL to Datalog [Polleres, 2007]
Conclusions and Future Work

- A unified query model for RSP
- Translations from RSP queries to LARS programs
- Next:
  - Comparing RSP semantics by comparing translated LARS programs
  - Equivalence of LARS programs

Jean-Paul Calbimonte, Óscar Corcho, and Alasdair J. G. Gray.
Enabling ontology-based access to streaming data sources.

Daniele Dell’Aglio, Jean-Paul Calbimonte, Marco Balduini, Óscar Corcho, and Emanuele Della Valle.
On Correctness in RDF Stream Processor Benchmarking.

Maxim Kolchin and Peter Wetz.
Demo: Yabench - yet another rdf stream processing benchmark.
In *RDF Stream Processing Workshop*, 2015.
Danh Le Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth.
A native and adaptive approach for unified processing of linked streams and linked data.
In *ISWC (1)*, pages 370–388, 2011.

Danh Le Phuoc, Minh Dao-Tran, Minh-Duc Pham, Peter Boncz, Thomas Eiter, and Michael Fink.
Linked stream data processing engines: Facts and figures.

Axel Polleres.
From SPARQL to rules (and back).
Ying Zhang, P. Minh Duc, O. Corcho, and J. P. Calbimonte.
SRBench: A Streaming RDF/SPARQL Benchmark.