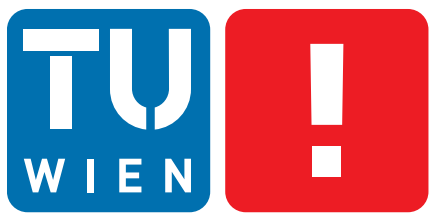
Towards Comparing RDF Stream Processing Semantics

Minh Dao-Tran Harald Beck Thomas Eiter

1st Workshop on High-Level Declarative Stream Processing

September 22, 2015



FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

COMMUNITY & BUSINESS GROUPS

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_{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_{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?

Outline

Running scenario

RDF & RDF Stream Processing (RSP)

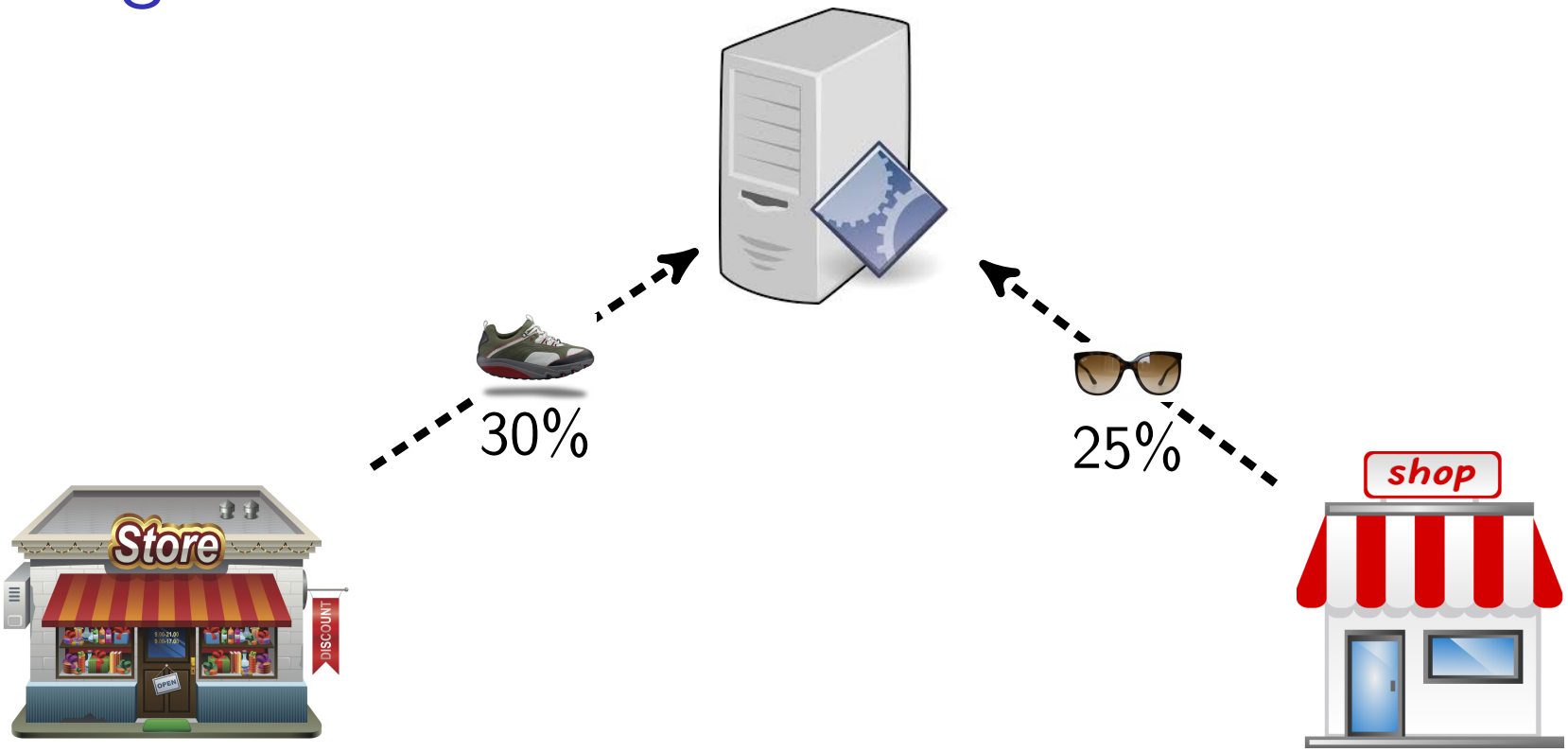
A **L**ogic for **A**nalyzing **R**easoning over **S**treams (LARS [Beck *et al.*, 2015])

Capturing RSP with LARS

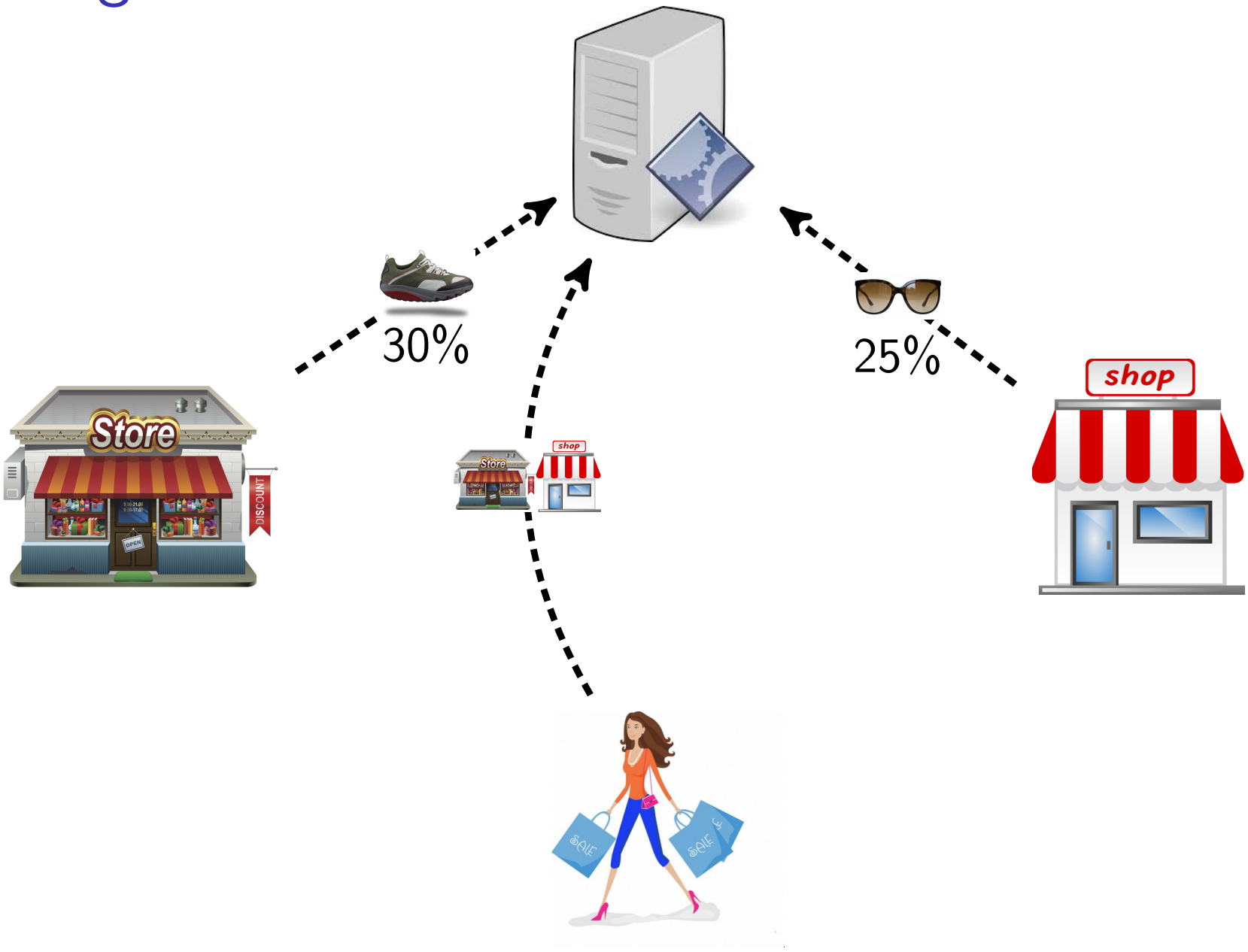
Running Scenario

Running Scenario

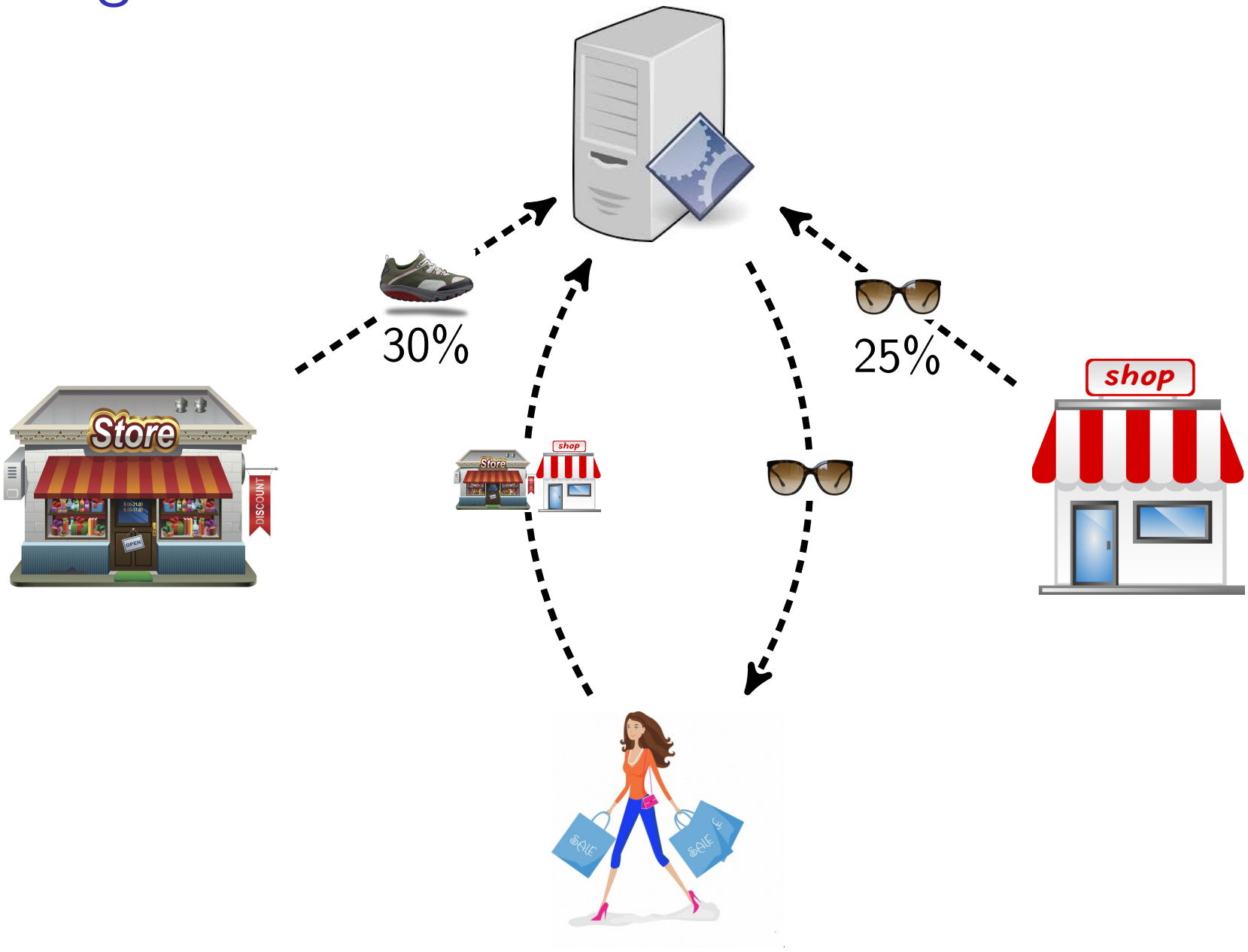
Running Scenario



Running Scenario

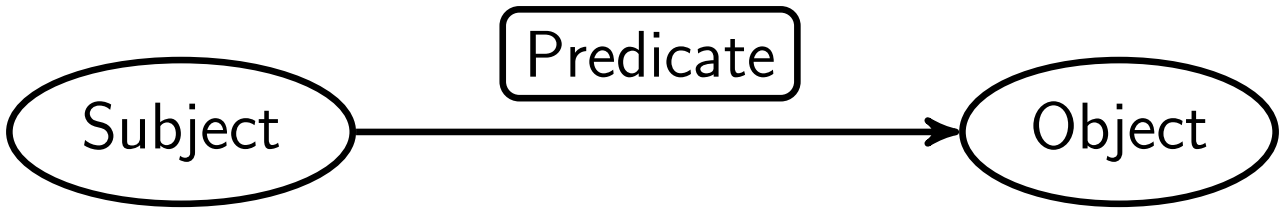


Running Scenario

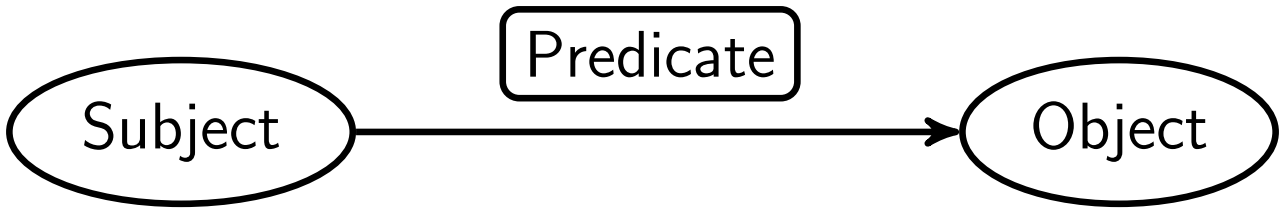


RDF

RDF



RDF



$$G = \{ :\text{“mbt”} \ :\texttt{g\_classify} \ :1. \quad :\text{“rayban”} \ :\texttt{g\_classify} \ :0. \ \ldots \}$$

$$g_1 = \{:\texttt{a} \ :\texttt{offers} \ :\texttt{c}_1. \quad :\texttt{c}_1 \ :\texttt{on} \ :\text{“mbt”}. \quad :\texttt{c}_1 \ :\texttt{reduce} \ :30.\}$$

$$g_2 = \{:\texttt{b} \ :\texttt{offers} \ :\texttt{c}_2. \quad :\texttt{c}_2 \ :\texttt{on} \ :\text{“rayban”}. \quad :\texttt{c}_2 \ :\texttt{reduce} \ :25.\}$$

$$g_3 = \{:\text{“claire”} \ :\texttt{isNear} \ :\texttt{a}. \quad :\text{“claire”} \ :\texttt{isNear} \ :\texttt{b}.\}$$

SPARQL

```
SELECT

FROM



WHERE
```

SPARQL

```
SELECT ?shop ?product ?percent
FROM



WHERE
```

SPARQL

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

SPARQL

```
SELECT ?shop ?product ?percent
FROM   <http://products>
       <http://coupons_snapshot>
       <http://locations_snapshot>
WHERE  {?shop    :offers     ?coupon.
        ?coupon  :reduce     ?percent.
        ?coupon  :on         ?product.
        ?user    :isNear      ?shop.
        ?product :g_classify ?gender.
```

SPARQL

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

SPARQL

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

$\mu = \{?\text{shop} \mapsto \text{a}, ?\text{product} \mapsto \text{“rayban”}, ?\text{percent} \mapsto 25\}$

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.
        ?coupon  :reduce     ?percent.
        ?coupon  :on         ?product.
        ?user    :isNear      ?shop.
        ?product :g_classify ?gender.
FILTER (?percent >= 20 && ?gender != 1)}
```

RDF Stream Processing Queries in CQELS

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

Key Differences between C-SPARQL and CQELS

	C-SPARQL	CQELS
create snapshot	merge patterns on input streams into the default graph	apply patterns on input streams
execution mode	pull-based	push-based

Modeling RSP Queries $Q = (V, P, \mathcal{D}, \mathcal{S})$

Modeling RSP Queries $Q = (V, P, \mathcal{D}, \mathcal{S})$

$$V \quad = \quad \{?\texttt{shop}, ?\texttt{pname}, ?\texttt{percent}\}$$

Modeling RSP Queries $Q = (V, P, \mathcal{D}, \mathcal{S})$

$$
\begin{aligned}
V &= \{\texttt{?shop}, \texttt{?pname}, \texttt{?percent}\} \\
P &= (P_1 \cup P_2 \cup P_3) \text{ FILTER } R \\
P_1 &= \left\{ \begin{array}{lll} \texttt{?shop} & \texttt{:offers} & \texttt{?coupon.} \\ \texttt{?coupon} & \texttt{:on} & \texttt{?product.} \\ \texttt{?coupon} & \texttt{:reduce} & \texttt{?percent.} \end{array} \right\} \\
P_2 &= \{\ \texttt{?user} \quad \texttt{:isNear} \quad \texttt{?shop.}\ \} \\
P_3 &= \{\ \texttt{?product} \quad \texttt{:g\_classify} \quad \texttt{?gender.}\ \}
\end{aligned}
$$

Modeling RSP Queries $Q = (V, P, \mathcal{D}, \mathcal{S})$

$$
\begin{aligned}
V &= \{\texttt{?shop}, \texttt{?pname}, \texttt{?percent}\} \\
P &= (P_1 \cup P_2 \cup P_3) \text{ FILTER } R \\
P_1 &= \left\{ \begin{array}{lll} \texttt{?shop} & \texttt{:offers} & \texttt{?coupon.} \\ \texttt{?coupon} & \texttt{:on} & \texttt{?product.} \\ \texttt{?coupon} & \texttt{:reduce} & \texttt{?percent.} \end{array} \right\} \\
P_2 &= \{ \texttt{?user} \quad \texttt{:isNear} \quad \texttt{?shop.} \} \\
P_3 &= \{ \texttt{?product} \quad \texttt{:g\_classify} \quad \texttt{?gender.} \} \\
R &= (\texttt{?percent} \geq 20 \;\&\&\; \texttt{?gender} \neq 1)
\end{aligned}
$$

Modeling RSP Queries $Q = (V, P, \mathcal{D}, \mathcal{S})$

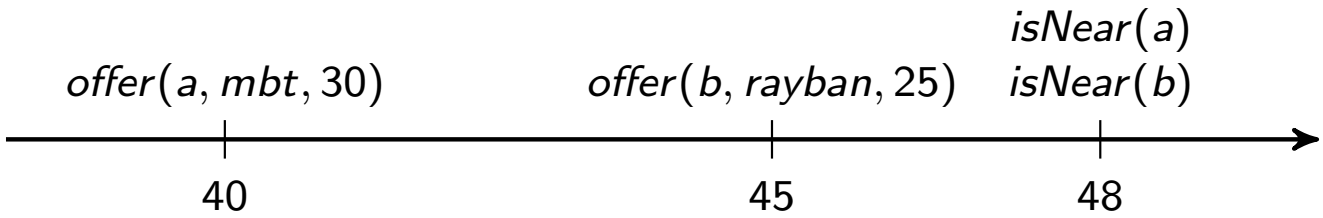
$$
\begin{aligned}
V &= \{\texttt{?shop}, \texttt{?pname}, \texttt{?percent}\} \\
P &= (P_1 \cup P_2 \cup P_3) \text{ FILTER } R \\
P_1 &= \left\{ \begin{array}{lll} \texttt{?shop} & \texttt{:offers} & \texttt{?coupon.} \\ \texttt{?coupon} & \texttt{:on} & \texttt{?product.} \\ \texttt{?coupon} & \texttt{:reduce} & \texttt{?percent.} \end{array} \right\} \\
P_2 &= \{\ \texttt{?user} \quad \texttt{:isNear} \quad \texttt{?shop.}\ \} \\
P_3 &= \{\ \texttt{?product} \quad \texttt{:g\_classify} \quad \texttt{?gender.}\ \} \\
R &= (\texttt{?percent} \geq 20\ \texttt{\&\&}\ \texttt{?gender} \neq 1) \\
\mathcal{D} &= \{\texttt{<products>}\}
\end{aligned}
$$

Modeling RSP Queries $Q = (V, P, \mathcal{D}, \mathcal{S})$

$$
\begin{aligned}
V &= \{\texttt{?shop}, \texttt{?pname}, \texttt{?percent}\} \\
P &= (P_1 \cup P_2 \cup P_3) \text{ FILTER } R \\
P_1 &= \left\{ \begin{array}{lll} \texttt{?shop} & \texttt{:offers} & \texttt{?coupon.} \\ \texttt{?coupon} & \texttt{:on} & \texttt{?product.} \\ \texttt{?coupon} & \texttt{:reduce} & \texttt{?percent.} \end{array} \right\} \\
P_2 &= \{\ \texttt{?user} \quad \texttt{:isNear} \quad \texttt{?shop.}\ \} \\
P_3 &= \{\ \texttt{?product} \quad \texttt{:g\_classify} \quad \texttt{?gender.}\ \} \\
R &= (\texttt{?percent} \geq 20\ \texttt{\&\&}\ \texttt{?gender} \neq 1) \\
\mathcal{D} &= \{\texttt{<products>}\} \\
\mathcal{S} &= \left\{ \begin{array}{l} (\texttt{<http://coupons>}, \texttt{[RANGE 30m]}, P_1), \\ (\texttt{<http://locations>}, \texttt{[RANGE 5m]}, P_2) \end{array} \right\}.
\end{aligned}
$$

LARS in a Nutshell: Stream Representation

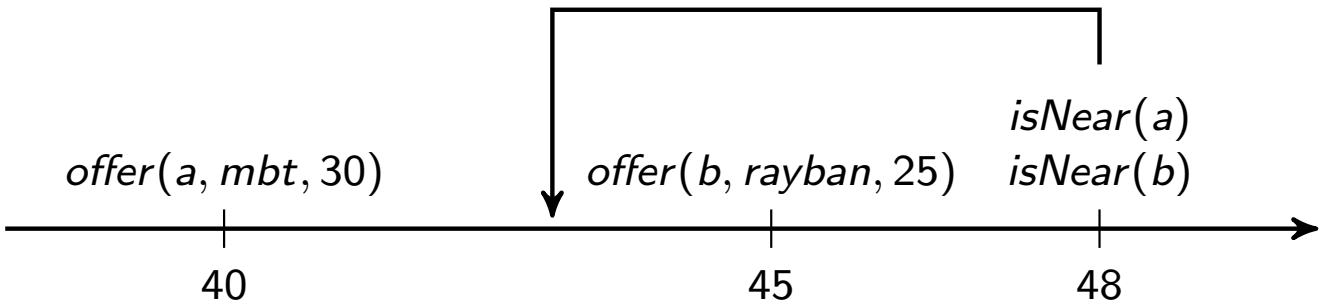
$S = (T, \upsilon)$



$T = [0, 50]$

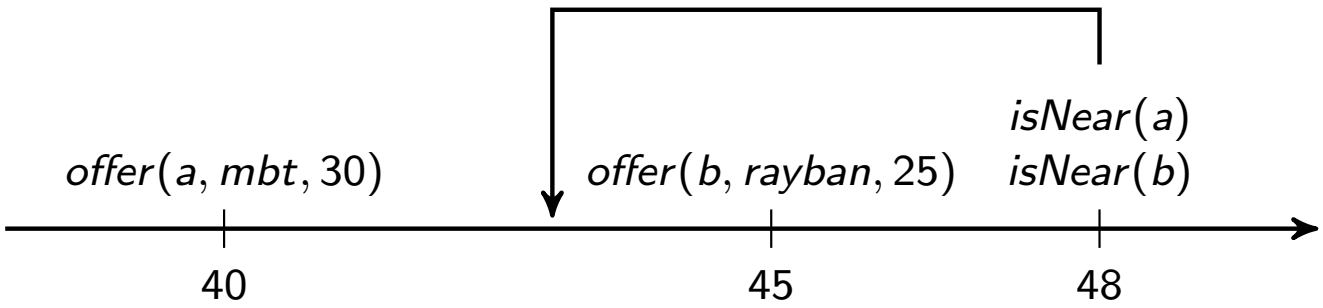
$$\upsilon = \left\{ \begin{array}{l} 40 \mapsto \{offer(a, mbt, 30)\}, 45 \mapsto \{offer(b, rayban, 25)\} \\ 48 \mapsto \{isNear(a), isNear(b)\} \end{array} \right\}$$

LARS in a Nutshell: Window functions



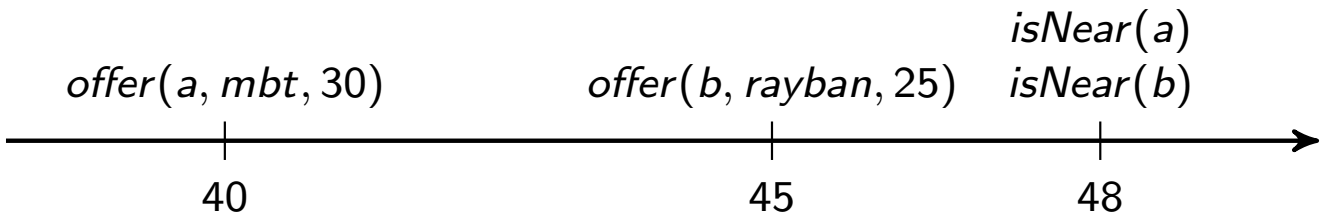
$$S' = w_\iota(S, t, \vec{x})$$

LARS in a Nutshell: Window functions



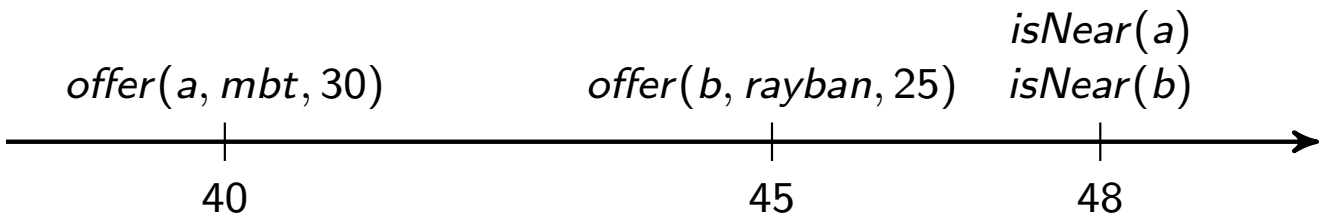
$$S' = w(S, 48, (5, 0, 1)) = ([43, 48], \left\{ \begin{array}{l} 45 \mapsto \{offer(b, rayban, 25)\} \\ 48 \mapsto \{isNear(a), isNear(b)\} \end{array} \right\})$$

LARS in a Nutshell: LARS formulas



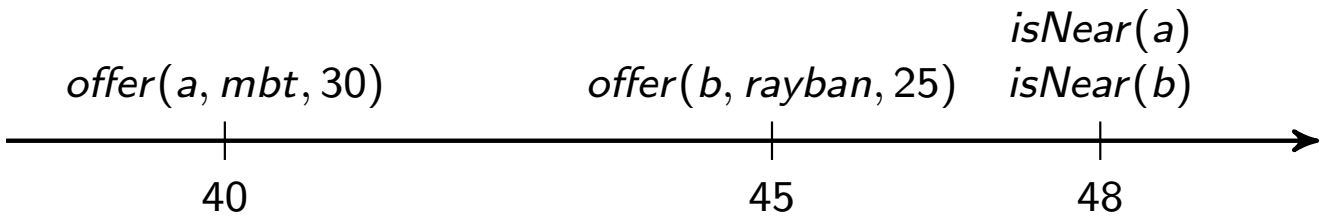
$\alpha ::=$

LARS in a Nutshell: LARS formulas



$$\alpha ::= a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha$$

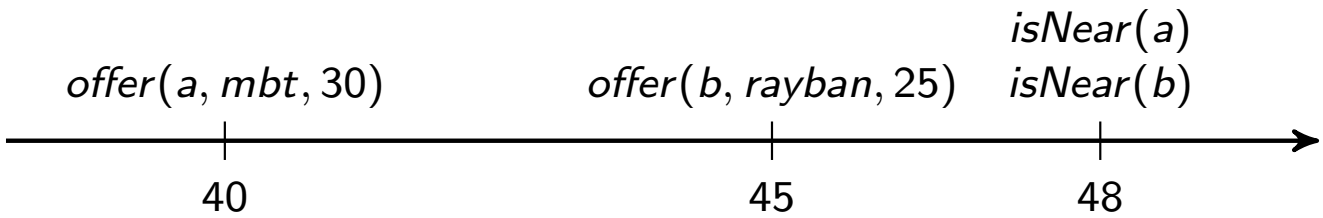
LARS in a Nutshell: LARS formulas



$$\alpha ::= a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha$$

- various ways for time references

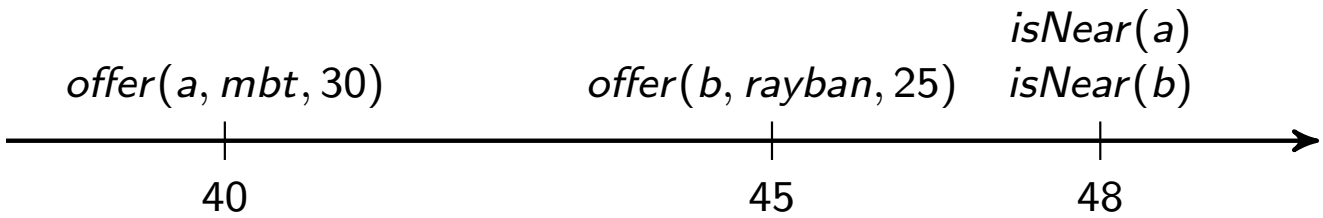
LARS in a Nutshell: LARS formulas



$$\alpha ::= a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \mid \boxplus_{\iota}^{\vec{x}}\alpha$$

- various ways for time references
- window operators with possibility to nest

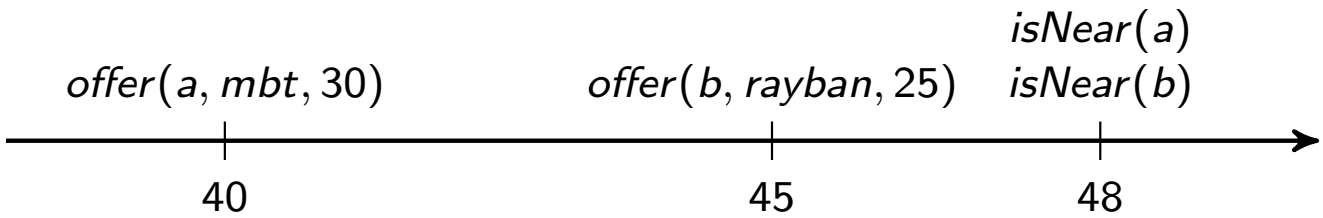
LARS in a Nutshell: LARS formulas



$$\alpha ::= a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \mid \boxplus_{\iota}^{\vec{x}}\alpha$$

- various ways for time references
- window operators with possibility to nest
- $\boxplus^{10}\Diamond offer(Sh, Pr, Pe)$

LARS in a Nutshell: LARS formulas



$$\alpha ::= a \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \Diamond\alpha \mid \Box\alpha \mid @_t\alpha \mid \boxplus_{\iota}^{\vec{x}}\alpha$$

- various ways for time references
- window operators with possibility to nest
- $\boxplus^{10}\Diamond offer(Sh, Pr, Pe)$
- $\boxplus^{5}\Box isNear(a)$

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

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

$$\begin{aligned} ans(Sh, Pr, Pe) \leftarrow \; & \boxplus^{30} \Diamond offer(Sh, Pr, Pe), \boxplus^{5} \Diamond isNear(Sh), \\ & g\_classify(Pr, Ge), Pe \geq 20, Ge \neq 1. \end{aligned}$$

LARS in a Nutshell: LARS rules/programs

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

$$\begin{aligned} ans(Sh, Pr, Pe) \leftarrow\ & \boxplus^{30} \Diamond offer(Sh, Pr, Pe), \boxplus^{5} \Diamond isNear(Sh), \\ & g\_classify(Pr, Ge), Pe \geq 20, Ge \neq 1. \end{aligned}$$

$$@_{T+5} near(b) \leftarrow \boxplus^{5} @_T isNear(a).$$

LARS in a Nutshell: LARS rules/programs

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

$$\begin{aligned} ans(Sh, Pr, Pe) \leftarrow\ & \boxplus^{30} \Diamond offer(Sh, Pr, Pe), \boxplus^{5} \Diamond isNear(Sh), \\ & g\_classify(Pr, Ge), Pe \geq 20, Ge \neq 1. \end{aligned}$$

$$@_{T+5} near(b) \leftarrow \boxplus^{5} @_T isNear(a).$$

$$\boxplus^{+5} \Diamond near(b) \leftarrow \boxplus^{5} \Diamond isNear(a).$$

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

$I = ([0, 50], \upsilon_I)$ where

$$\upsilon_I = \left\{ \begin{array}{l} 40 \mapsto \{\mathit{offer}(a, \mathit{rayban}, 30)\} \\ 45 \mapsto \{\mathit{offer}(b, \mathit{rayban}, 25)\} \\ 48 \mapsto \{\mathit{isNear}(a), \mathit{isNear}(b), \mathit{ans}(b, \mathit{rayban}, 25)\} \end{array} \right\}$$

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

Translations at a Glance

Capture push-, pull-based execution modes:

$$
\begin{aligned}
\mathit{trigger}(r) &= H(r) \leftarrow B(r), \texttt{trig}. \\
\mathit{trigger}(P) &= \{\mathit{trigger}(r) \mid r \in P \wedge B(r) \neq \emptyset\} \\
\rhd(P) &= \mathit{trigger}(P) \cup \{\texttt{trig} \leftarrow \boxplus_{\mathit{NOW}}\, p(\vec{X}). \mid p \in \mathcal{A}^{\mathcal{I}}\} \\
\lhd(P, U) &= \mathit{trigger}(P) \cup \{\texttt{trig} \leftarrow \boxplus_{\mathit{NOW}} @_T \mathit{true}, T \,\%\, U = 0.\}
\end{aligned}
$$

Translations at a Glance

Capture push-, pull-based execution modes:

$$
\begin{aligned}
\mathit{trigger}(r) &= H(r) \leftarrow B(r), \mathtt{trig}. \\
\mathit{trigger}(P) &= \{\mathit{trigger}(r) \mid r \in P \wedge B(r) \neq \emptyset\} \\
\rhd(P) &= \mathit{trigger}(P) \cup \{\mathtt{trig} \leftarrow \boxplus_{\mathit{NOW}}\, p(\vec{X}). \mid p \in \mathcal{A}^{\mathcal{I}}\} \\
\lhd(P, U) &= \mathit{trigger}(P) \cup \{\mathtt{trig} \leftarrow \boxplus_{\mathit{NOW}} @_T \mathit{true}, T \,\%\, U = 0.\}
\end{aligned}
$$

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
  - Equivalance of LARS programs

References I

Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus.
C-SPARQL: a continuous query language for rdf data streams.
Int. J. Semantic Computing, 4(1):3–25, 2010.

Harald Beck, Minh Dao-Tran, Thomas Eiter, and Michael Fink.
LARS: A Logic-based Framework for Analyzing Reasoning over Streams.
In *AAAI*, 2015.

References II

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

In *ISWC (1)*, pages 96–111, 2010.

Daniele Dell'Aglio, Jean-Paul Calbimonte, Marco Balduini, Óscar Corcho, and Emanuele Della Valle.
On Correctness in RDF Stream Processor Benchmarking.
In *ISWC 2013*, pages 326–342, 2013.

Maxim Kolchin and Peter Wetz.
Demo: Yabench - yet another rdf stream processing benchmark.
In *RDF Stream Processing Workshop*, 2015.

References III


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.
In *ISWC - ET*, pages 300–312, 2012.


Axel Polleres.
From SPARQL to rules (and back).
In *WWW 2007*, pages 787–796, 2007.

References IV


Ying Zhang, P. Minh Duc, O. Corcho, and J. P. Calbimonte.
SRBench: A Streaming RDF/SPARQL Benchmark.
In *ISWC*, pages 641–657, 2012.