Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho,
D. Dell'Aglio, E. Della Valle, and J.Z. Pan

http://streamreasoning.org/sr4ld2013

ISWC 2013
Sydney, Australia

RDF stream processing models

Daniele Dell’Aglio, daniele.dellaglio@polimi.it

Jean-Paul Cabilmonte, jp.calbimonte@upm.es

Share, Remix, Reuse — Legally

- This work is licensed under the Creative Commons Attribution 3.0 Unported License.
- **Your are free:**
  - **to Share** — to copy, distribute and transmit the work
  - **to Remix** — to adapt the work
- **Under the following conditions**
  - **Attribution** — You must attribute the work by inserting
    - "[source http://streamreasoning.org/sr4ld2013]" at the end of each reused slide
    - a credits slide stating
      - These slides are partially based on "Streaming Reasoning for Linked Data 2013" by M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan http://streamreasoning.org/sr4ld2013
- To view a copy of this license, visit http://creativecommons.org/licenses/by/3.0/

Outline

- Continuous RDF model extensions
  - RDF Streams, timestamps
- Continuous extensions of SPARQL
  - Continuous evaluation
  - Additional operators
- Overview of existing systems
  - Implemented operators
  - Different evaluation approaches

Continuous extensions of RDF

- As you know, "*RDF is a standard model for data interchange on the Web*" (http://www.w3.org/RDF/)

$$\langle sub_1\ pred_1\ obj_1\rangle$$

$$\langle sub_2\ pred_2\ obj_2\rangle$$

- We want to extend RDF to model data streams
- A data stream is an (infinite) ordered sequence of **data items**
- A data item is a self-consumable informative unit

Data items

- With **data item** we can refer to:
  1. A **triple**

     ```
     <:alice :isWith :bob>
     ```

  2. A **graph**

     ```
     <:alice :posts :p>
     <:p :who :bob>
     <:p :where :redRoom>
     ```

     :graph1

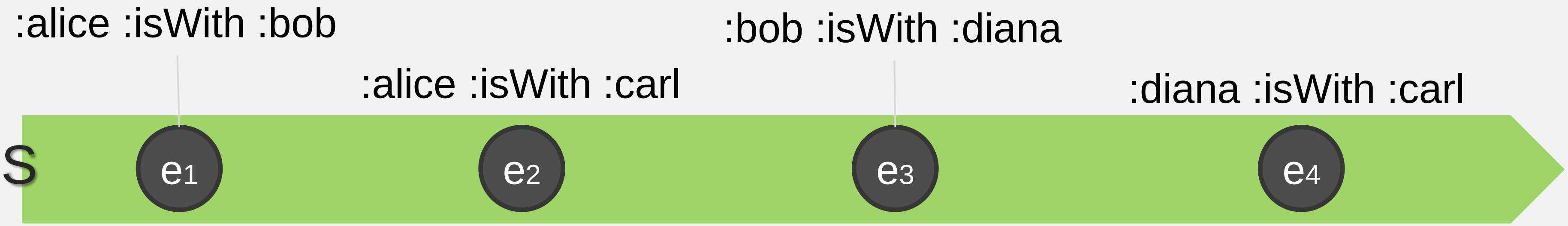
Data items and time

- Do we need to associate the time to data items?
  - It depends on what we want to achieve (see next!)
- If yes, how to take into account the time?
  - Time should not (but could) be part of the schema
  - Time should not be accessible through the query language
  - Time as object would require a lot of reification
- How to extend the RDF model to take into account the time?

Application time

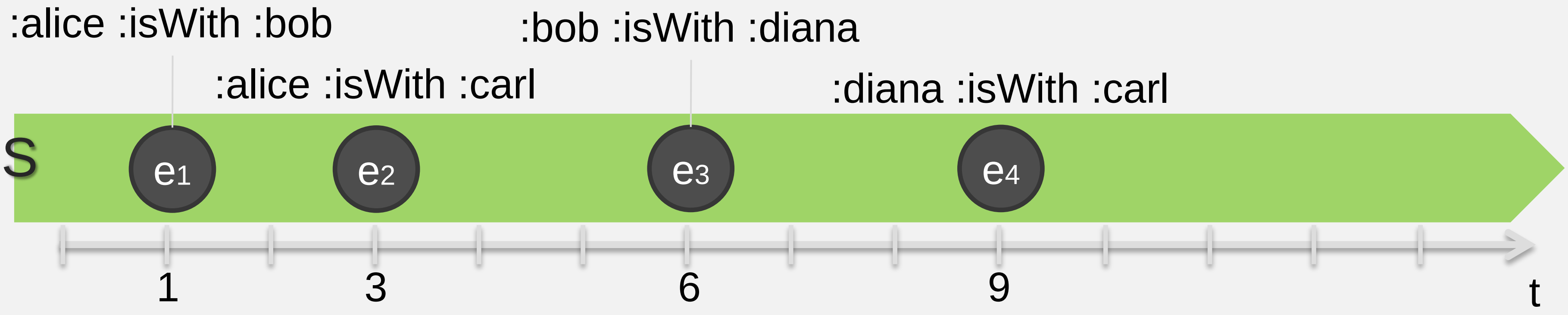
- A timestamp is a temporal identifier associated to a data item
- The **application time** is a set of one or more timestamps associated to the data item
- Two data items can have the same application time
  - Contemporaneity
- Who does assign the application time to an event?
  - The one that generates the data stream!

Missing application time



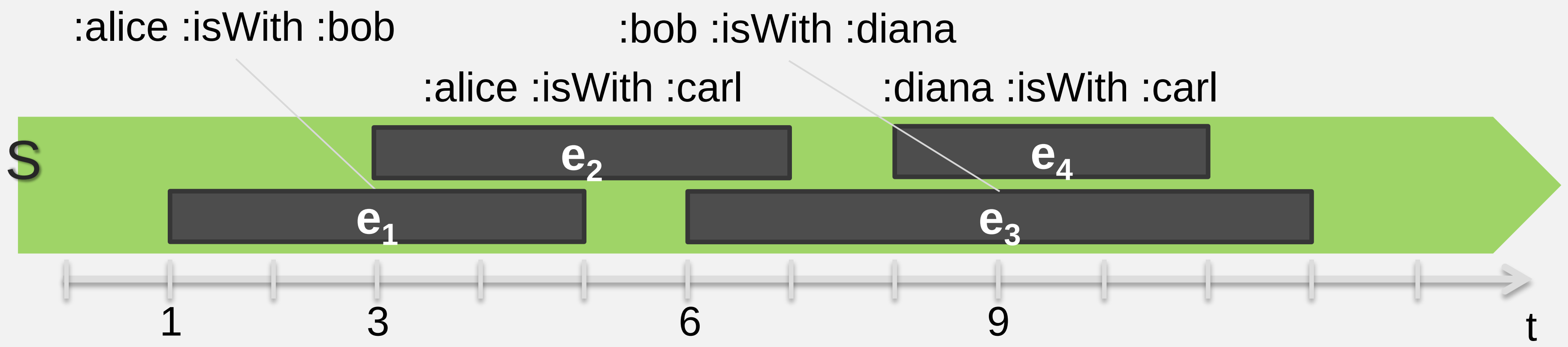
- A RDF stream without timestamp is an ordered sequence of data items
- The order can be exploited to perform queries
  - Does Alice meet Bob before Carl?
  - Who does Carl meet first?

Application time: one timestamp



- One timestamp: the time on which the data item occurs
- We can start to compose queries taking into account the time
  - How many people has Alice met in the last 5m?
  - Does Diana meet Bob and then Carl within 5m?

Application time: two timestamps



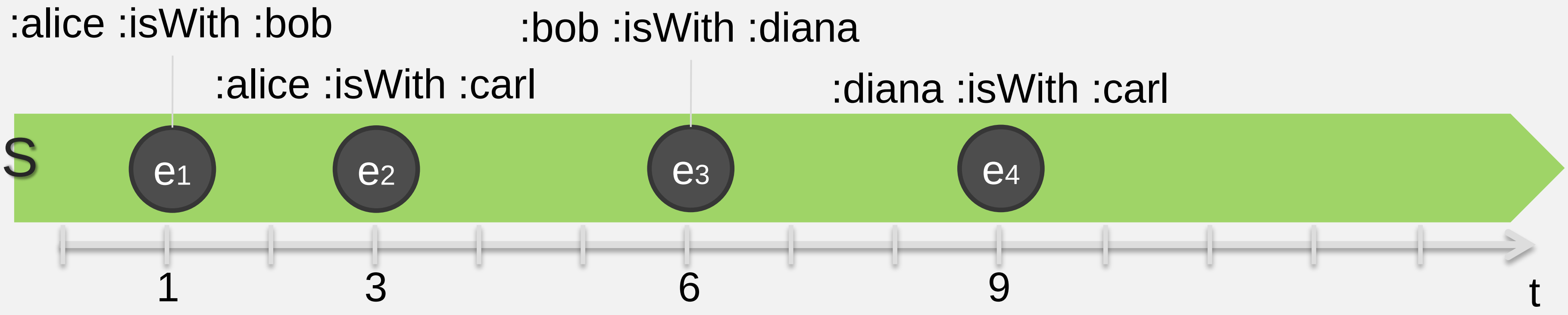
- Two timestamps: the time range on which the data item is valid (from, to]

- It is possible to write even more complex constraints:
  - Which are the meetings the last less than 5m?
  - Which are the meetings with conflicts?

Classification of existing systems

	Triple	Graph
No timestamp	Instans	
One timestamp	C-SPARQL CQELS SPARQLstream	SLD
Two timestamps	EP-SPARQL/Etalis	

Our assumptions



- In the following we will consider the following setting
  - A RDF triple is an event
  - Application time: single timestamp
  - System time = application time

```
<:alice :isWith:bob>:[1]

<:alice :isWith:carl>:[3]

<:bob :isWith :diana>:[6]

...
```

Let’s process the RDF streams!

- DSMS and CEP worlds suggest different techniques and approaches to process data streams
- We focus on the CQL/STREAM model

System time

- Stream processors can elaborate data streams exploiting the timestamps associated to the events
- When a system receives an event, it could have the need of associating a timestamp
  - This is the **system time**
- The system time is an **internal** value, it does not exit from the system!
- The system time must be **unique**
- Can application and system time coincide?
  - It depends
  - Approximation

RDF stream

- An RDF stream is an infinite sequence of timestamped events (triples or graphs)

…

$\langle event_{i}, t_{i} \rangle$

$\langle event_{i+1}, t_{i+1} \rangle$

$\langle event_{i+2}, t_{i+2} \rangle$

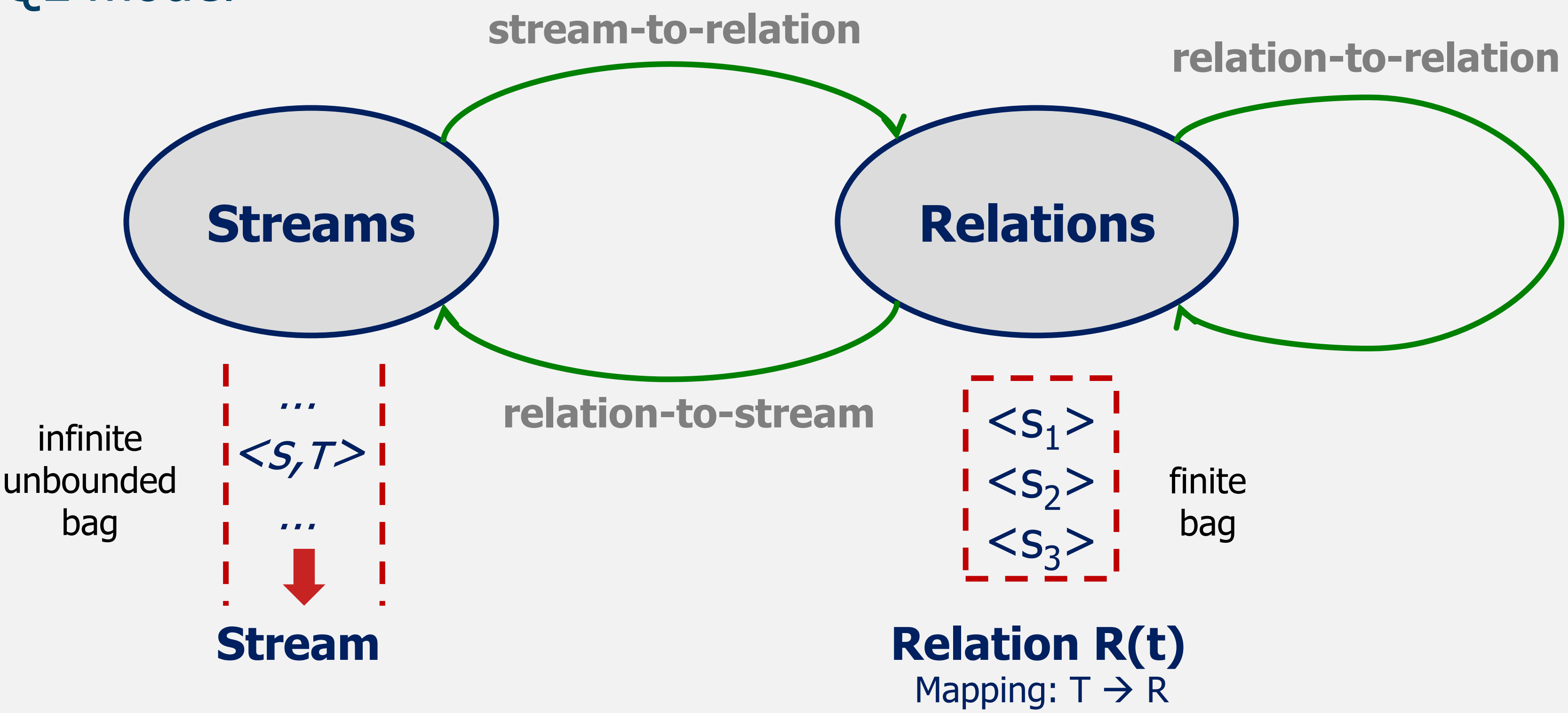
…

- The (application) timestamps must be non-decreasing

$t_i <= t_{i+1}$

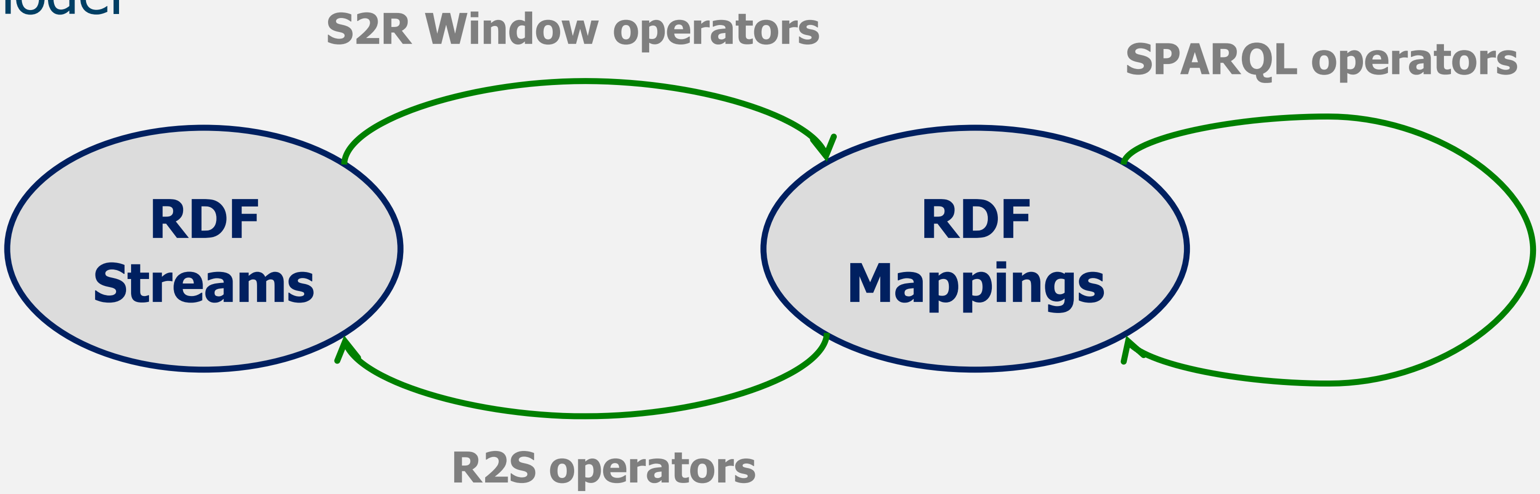
Querying data streams

- CQL model



Querying RDF data streams

- CQL model


S2R Window operators

SPARQL operators

RDF Streams

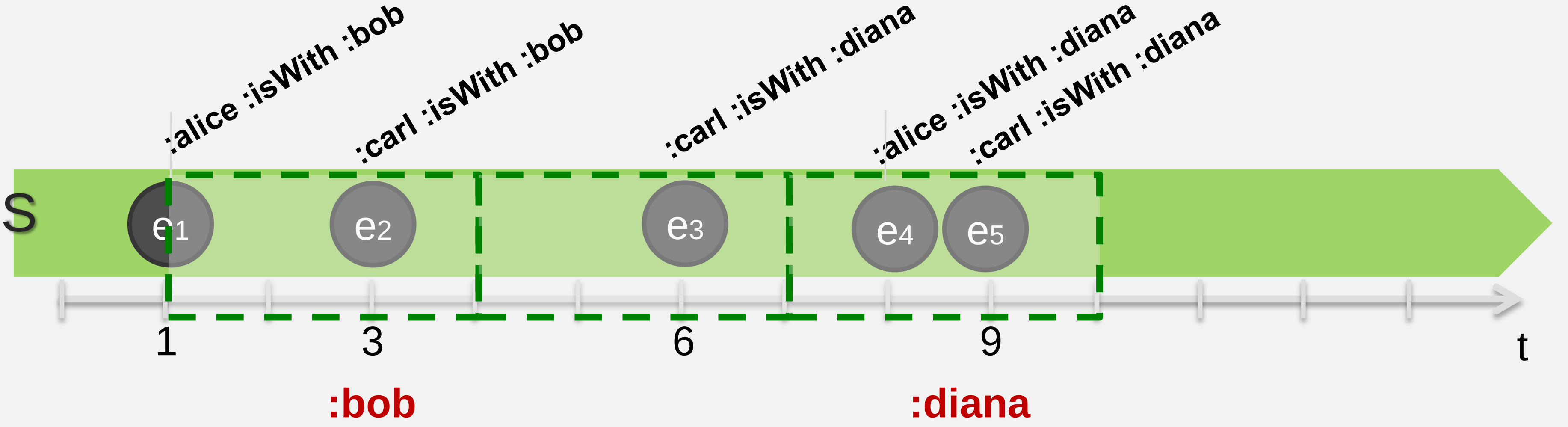
RDF Mappings

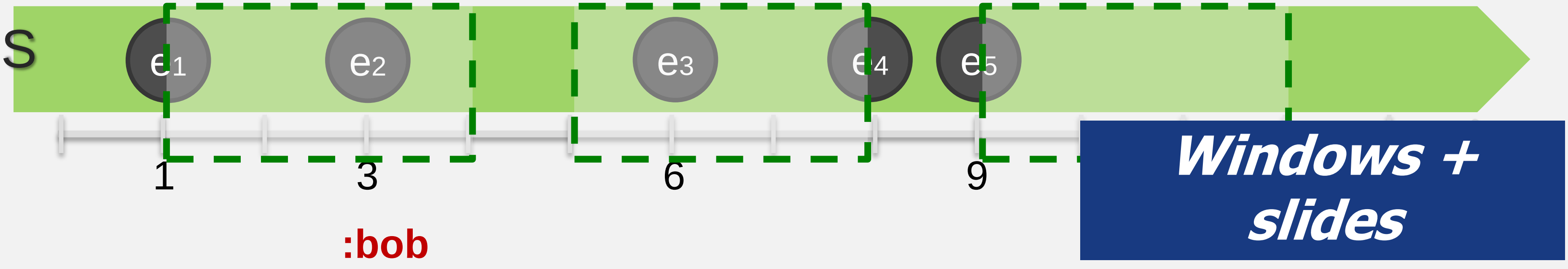
R2S operators

***Abstract* query processing model**

Time-based Windows

- Who are both alice and carl meeting?

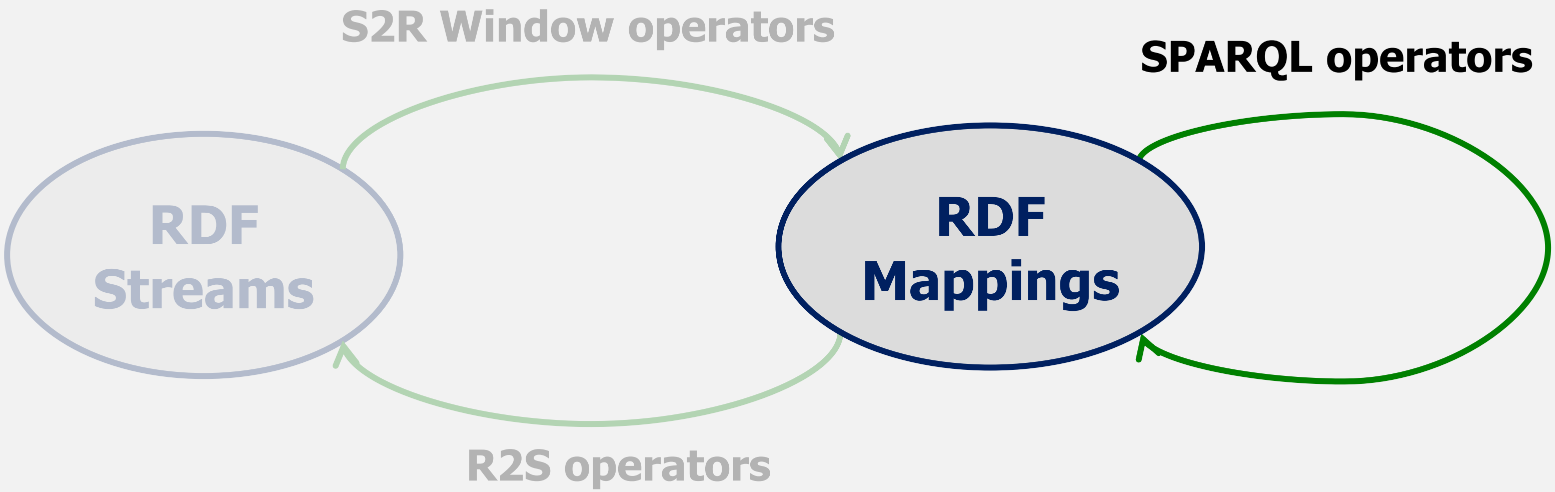




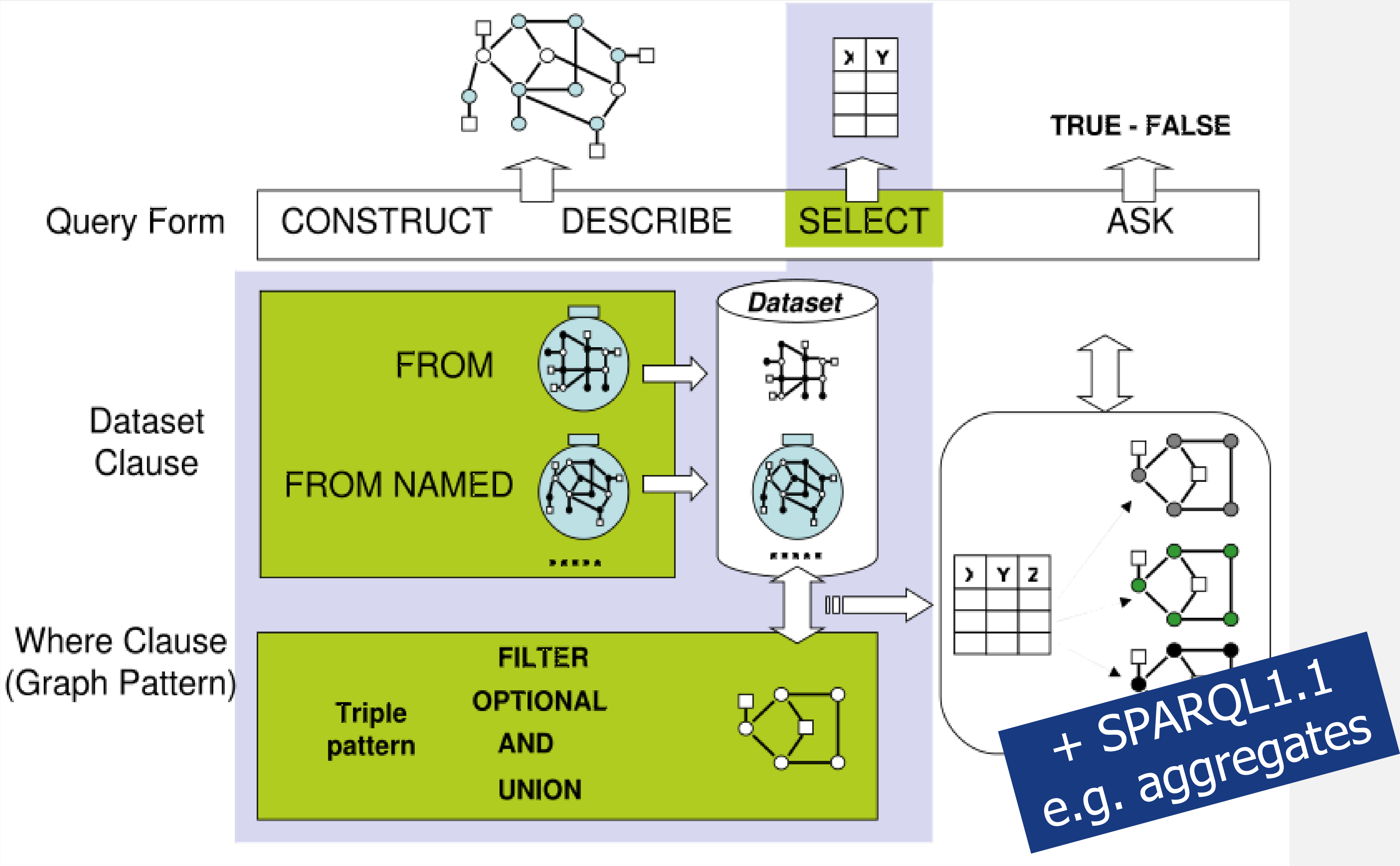
Windows + slides

R2R operators

- SPARQL operators
  - Graph pattern matching
  - JOIN
  - OPTIONAL JOIN
  - SELECTION
  - UNION



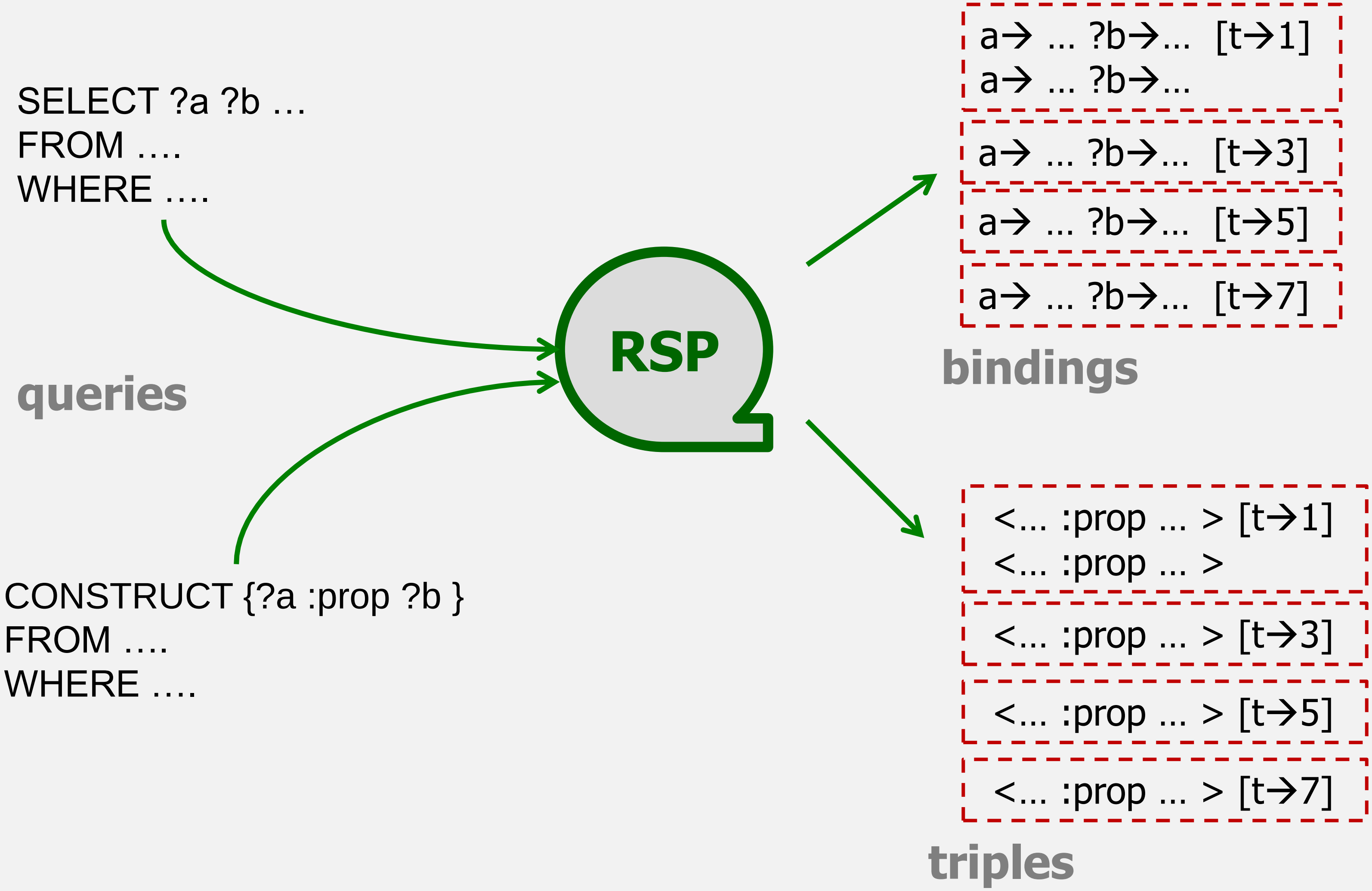
SPARQL: a quick recap



+ SPARQL1.1
e.g. aggregates

Output: relation

- Case 1: the output is a set of timestamped mappings



SELECT ?a ?b …
FROM ….
WHERE ….

CONSTRUCT {?a :prop ?b }
FROM ….
WHERE ….

queries

a→ … ?b→… [t→1]
a→ … ?b→…

a→ … ?b→… [t→3]

a→ … ?b→… [t→5]

a→ … ?b→… [t→7]

bindings

<… :prop … > [t→1]
<… :prop … >

<… :prop … > [t→3]

<… :prop … > [t→5]

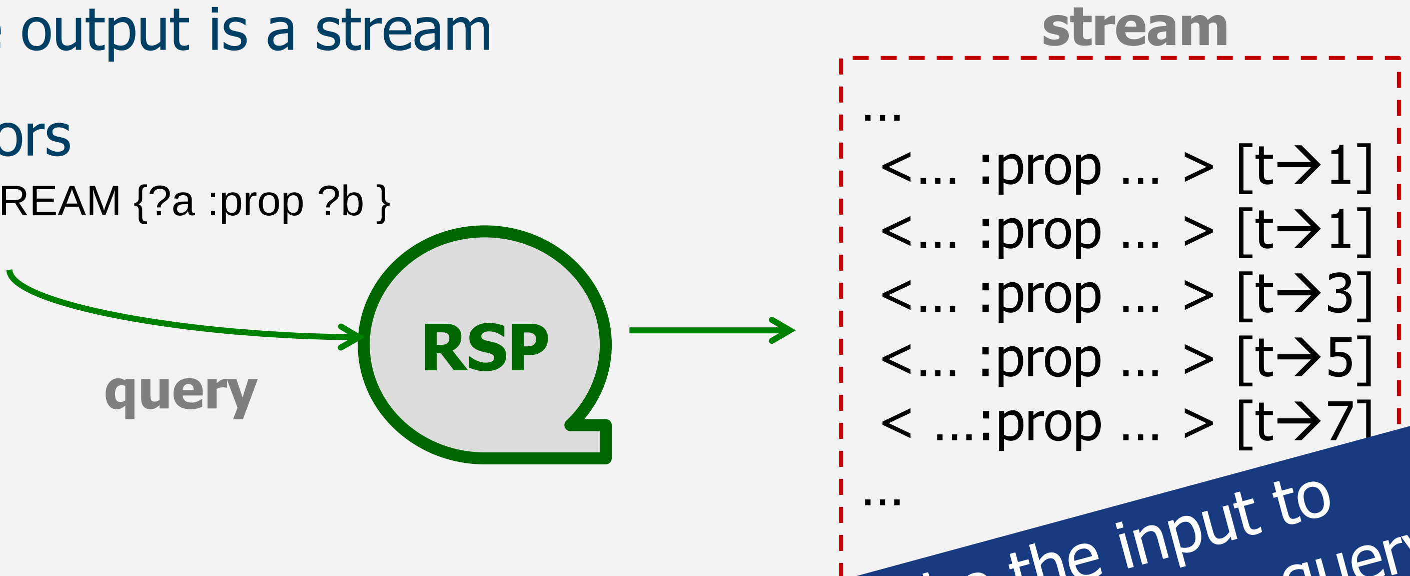
<… :prop … > [t→7]

triples

Output: stream

- Case 2: the output is a stream
- R2S operators

```
CONSTRUCT RSTREAM {?a :prop ?b }
FROM ….
WHERE ….
```



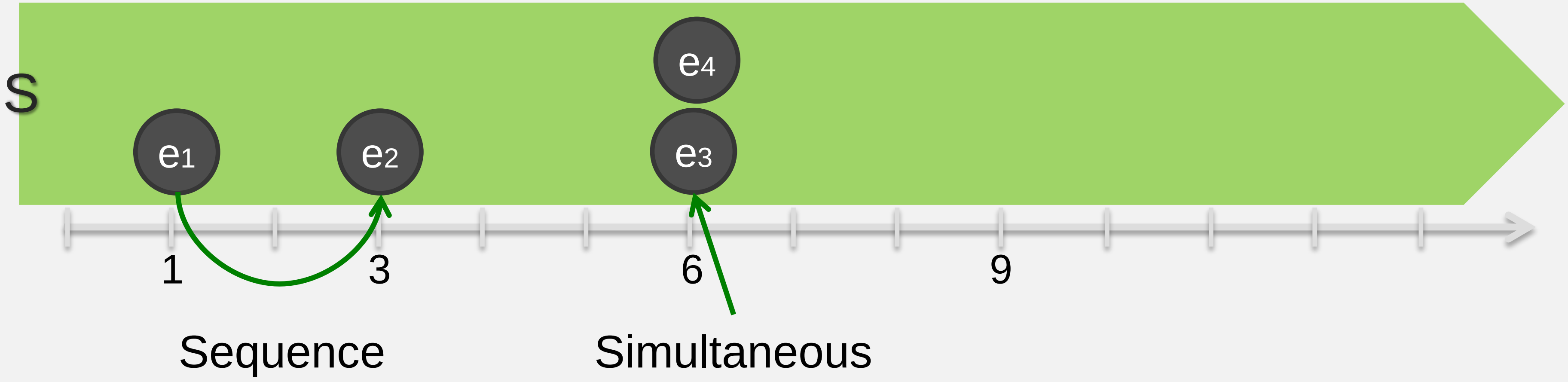
…
<… :prop … > [t→1]
<… :prop … > [t→1]
<… :prop … > [t→3]
<… :prop … > [t→5]
< …:prop … > [t→7]
…

Can be the input to another continuous query

- R2S operators:
  - ISTREAM: stream out data in the last step wasn’t on the previous step
  - DSTREAM: stream out data in the previous step that isn’t in the last step
  - RSTREAM: stream out all data in the last step

Other operators

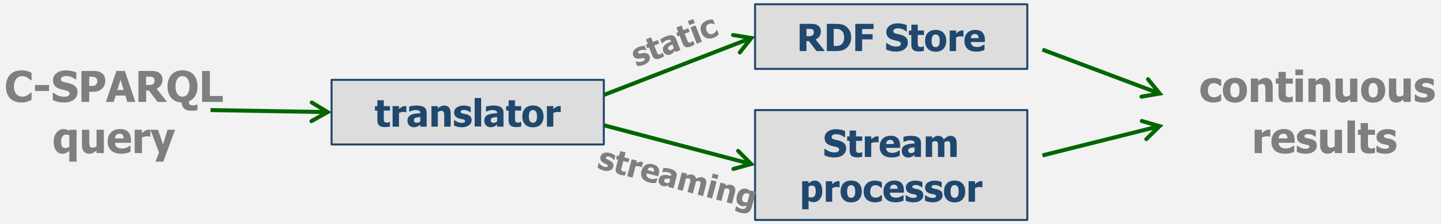
- Sequence operators and CEP world



- SEQ: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if e' occurs after e
- EQUALS: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if they occur simultaneously
- OPTIONALSEQ, OPTIONALEQUALS: Optional join variants

Existing RSP systems

- C-SPARQL: RDF Store + Stream processor
  - Combined architecture



C-SPARQL query → translator → static → RDF Store → continuous results

translator → streaming → Stream processor → continuous results

- CQELS: Implemented from scratch. Focus on performance
  - Native + adaptive joins for static-data and streaming d

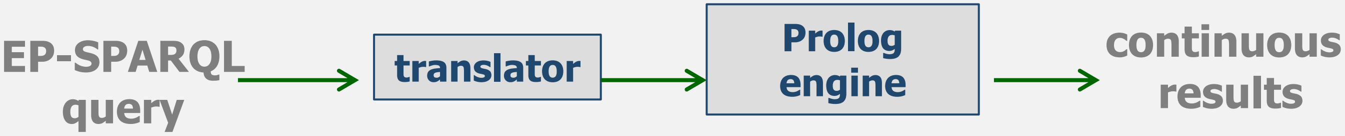
More details later on!

CQELS query → Native RSP → continuous results

Disclaimer: **oversimplified descriptions**

Existing RSP systems

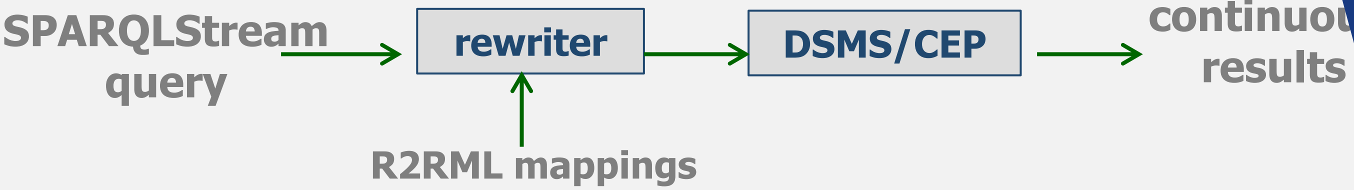
- EP-SPARQL: Complex-event detection
  - SEQ, EQUALS operators



EP-SPARQL query → translator → Prolog engine → continuous results

- SPARQLStream: Ontology-based stream query answering
  - Virtual RDF views, using R2RML mappings
  - SPARQL stream queries over the original data streams.

More details later on!



SPARQLStream query → rewriter → DSMS/CEP → continuous results

R2RML mappings → rewriter

- Instans: RETE-based evaluation

Disclaimer: **oversimplified descriptions**

Query languages syntax

```
SELECT ?sensor
FROM NAMED STREAM <http://www.cwi.nl/SRBench/observations> [NOW-3 HOURS SLIDE 10
MINUTES]
WHERE {
  ?observation om-owl:procedure ?sensor ;
               om-owl:observedProperty weather:WindSpeed ;
               om-owl:result [ om-owl:floatValue ?value ] . }
GROUP BY ?sensor HAVING ( AVG(?value) >= "74"^^xsd:float )
```

SPARQLStream

```
SELECT ?sensor
FROM STREAM <http://www.cwi.nl/SRBench/observations> [RANGE 1h STEP 10m]
WHERE {
  ?observation om-owl:procedure ?sensor ;
               om-owl:observedProperty weather:WindSpeed ;
               om-owl:result [ om-owl:floatValue ?value ] . }
GROUP BY ?sensor HAVING ( AVG(?value) >= "74"^^xsd:float )
```

C-SPARQL

We'll see more later

```
SELECT ?sensor
WHERE {
  STREAM <http://www.cwi.nl/SRBench/observations> [RANGE 10800s SLIDE 600s] {
    ?observation om-owl:procedure ?sensor ;
                 om-owl:observedProperty weather:WindSpeed ;
                 om-owl:result [ om-owl:floatValue ?value ] .} }
GROUP BY ?sensor HAVING ( AVG(?value) >= "74"^^xsd:float )
```

CQELS

Classification of existing systems

	Model	Continuous execution	Union, Join, Optional, Filter	Aggregates	Time window	Triple window	R2S operator	Sequence, Co-ocurrence	Time function
TA-SPARQL	TA-RDF	✗	✔	Limited	✗	✗	✗	✗	✗
tSPARQL	tRDF	✗	✔	✗	✗	✗	✗	✗	✗
Streaming SPARQL	RDF Stream	✔	✔	✗	✔	✔	✗	✗	✗
C-SPARQL	RDF Stream	✔	✔	✔	✔	✔	✗	✗	✔
CQELS	RDF Stream	✔	✔	✔	✔	✔	✗	✗	✗
SPARQLStream	(Virtual) RDF Stream	✔	✔	✔	✔	✗	✔	✗	✗
EP-SPARQL	RDF Stream	✔	✔	✔	✗	✗	✗	✔	✗
Instans	RDF	✔	✔	✔	✗	✗	✗	✗	✗

Disclaimer: other features may be missing

RDF Stream Processors

Can we compare these RSPs?

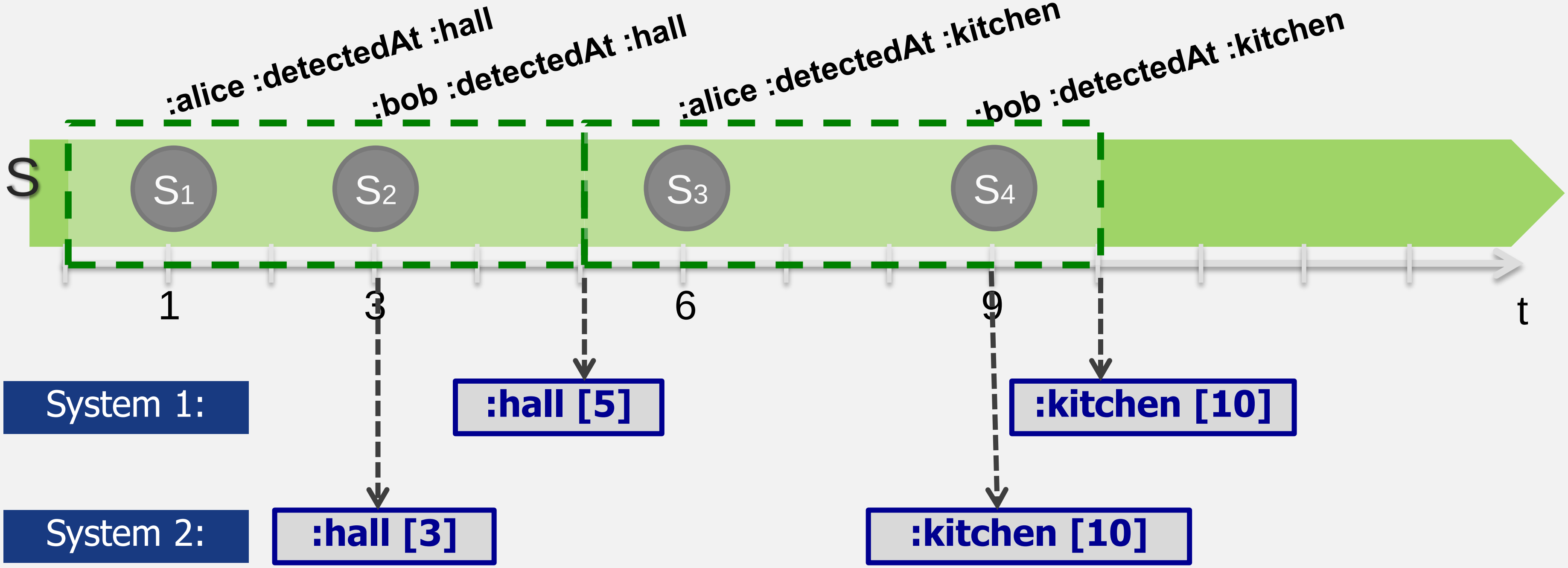
Do RSPs behave the same?

Do we get the same results form RSPs?

Check **operational semantics**

Operational Semantics

Where are both alice and bob in the last 5s?



Both correct?

Find out more later this week on the ISWC **Evaluation Track!** Thursday at noon!

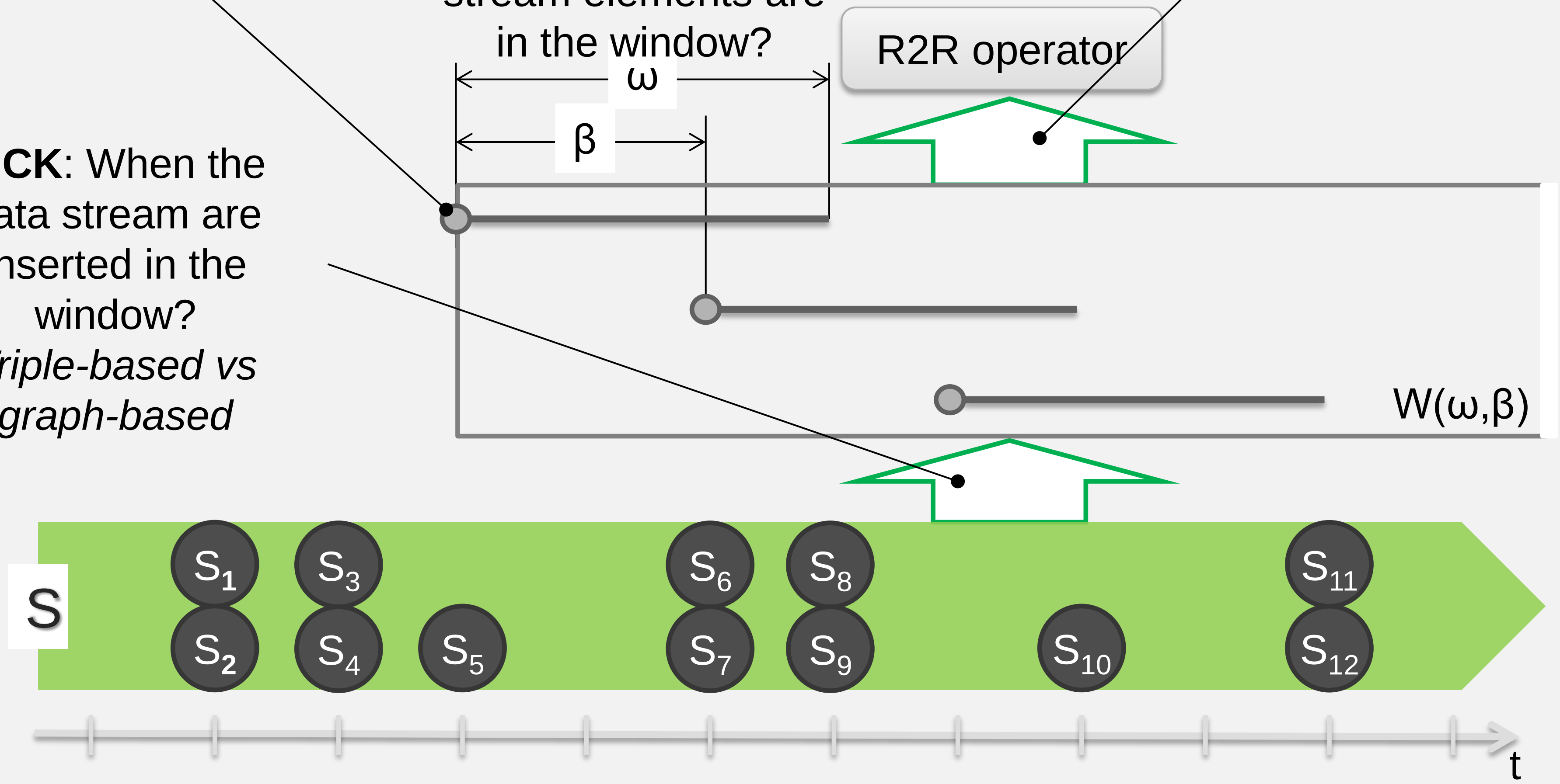
SECRET Model: understand operational semantics

t_0: When does the windowing start? (internal window param)

WINDOW CONTENT: Which stream elements are in the window?

REPORT: When is the window content made available to the R2R operator? *Non-empty content, Content-change, Window-close, Periodic*

TICK: When the data stream are inserted in the window? *Triple-based vs graph-based*



SECRET model classification

	CQELS	C-SPARQL	SPARQL$_{\text{stream}}$
Report	Content-change	Window-close Non-empty content	Window-close Non-empty content
Tick	Tuple-driven	Tuple-driven	Tuple-driven
Empty relation notification	No	Yes	No

- Characterize non-window-based RSPs?
- Multiple streams?, reasoning?, linking with static data?

Benchmarks and comparing

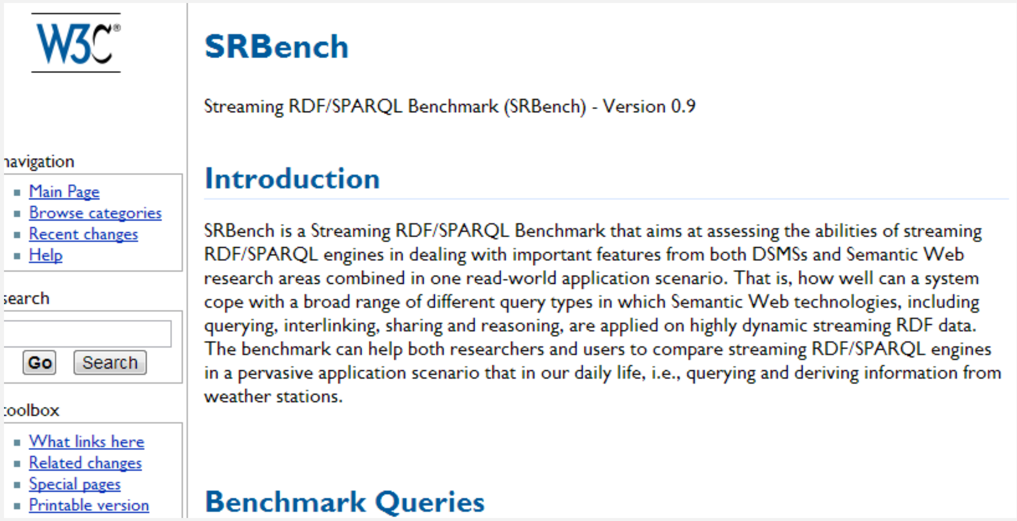
ISWC 2013
Sydney, Australia

http://www.w3.org/wiki/SRBench

C-SPARQL

SPARQLStream

CQELS



SRBench

Streaming RDF/SPARQL Benchmark (SRBench) - Version 0.9

Introduction

SRBench is a Streaming RDF/SPARQL Benchmark that aims at assessing the abilities of streaming RDF/SPARQL engines in dealing with important features from both DSMSs and Semantic Web research areas combined in one read-world application scenario. That is, how well can a system cope with a broad range of different query types in which Semantic Web technologies, including querying, interlinking, sharing and reasoning, are applied on highly dynamic streaming RDF data. The benchmark can help both researchers and users to compare streaming RDF/SPARQL engines in a pervasive application scenario that in our daily life, i.e., querying and deriving information from weather stations.

Benchmark Queries

Not exhaustive!

Functional Evaluation

System	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17
SPARQLStream	●	PP	A	G	G	●	●	G	G,IF	SD	SD	PP,SD	PP,SD	PP,SD	PP,SD	PP,SD	PP,SD
CQELS	●	PP	A	●	●	●	D/N	●	IF	●	●	PP	PP	PP	PP	PP	PP
C-SPARQL	●	PP	A	●	●	●	D	●	IF	●	●	PP	PP	PP	PP	PP	PP

Ask
Dstream
Group by and aggregations
IF expression
Negation
Property Path
Static **D**ataset

A lot to do…

- Agree on an RDF model?
  - Metamodel?
  - Timestamps in graphs?
  - Timestamp intervals
  - Compatibility with normal (static) RDF
- Additional operators for SPARQL?
  - Windows (not only time based?)
  - CEP operators
  - Semantics
- Go Web
  - Volatile URIs
  - Serialization: terse, compact
  - Protocols: HTTP, Websockets?

References

- Arasu, A., Babu, S., Widom, J.: The CQL continuous query language : semantic foundations. The VLDB Journal 15(2) (2006) 121–142
- Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: A continuous query language for RDF data streams. IJSC **4**(1) (2010) 3–25
- Botan, I., Derakhshan, R., Dindar, N., Haas, L., Miller, R.J., Tatbul, N.: Secret:A model for analysis of the execution semantics of stream processing systems. PVLDB 3(1) (2010) 232–243
- Calbimonte, J.P., Jeung, H., Corcho, O., Aberer, K.: Enabling Query Technologies for the Semantic Sensor Web. IJSWIS **8**(1) (2012) 43–63
- Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: ISWC. (2011) 370–388
- Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW. (2011) 635–644

Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan

http://streamreasoning.org/sr4ld2013

ISWC 2013
Sydney, Australia

RDF stream processing models

Daniele Dell’Aglio, daniele.dellaglio@polimi.it

Jean-Paul Cabilmonte, jp.calbimonte@upm.es