# Tutorial on RDF Stream Processing

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

http://streamreasoning.org/rsp2014





# RDF Stream models
# Continuous query models

Daniele Dell'Aglio

daniele.dellaglio@polimi.it

# 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

1. Continuous RDF model extensions
   - RDF Streams, timestamps
2. Continuous extensions of SPARQL
   - Continuous evaluation
   - Additional operators
3. Overview of existing systems
   - Features
   - Comparison

# Outline

1. Continuous RDF model extensions
   - RDF Streams, timestamps
2. Continuous extensions of SPARQL
   - Continuous evaluation
   - Additional operators
3. Overview of existing systems
   - Features
   - Comparison

# 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

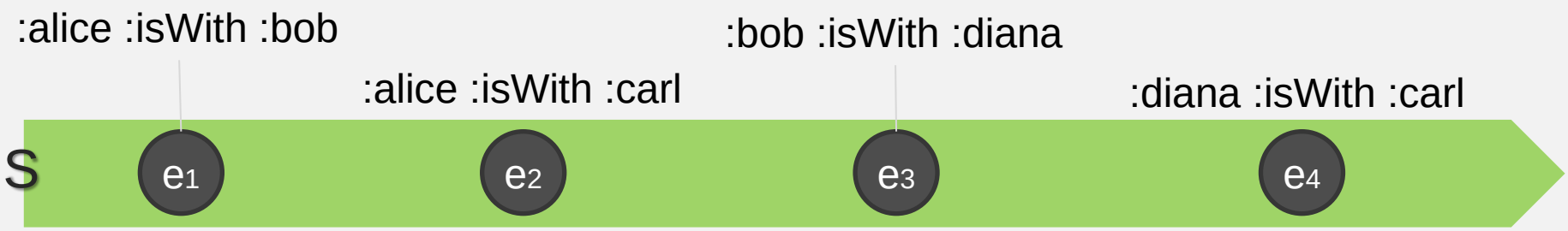
# Data items and time

- 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

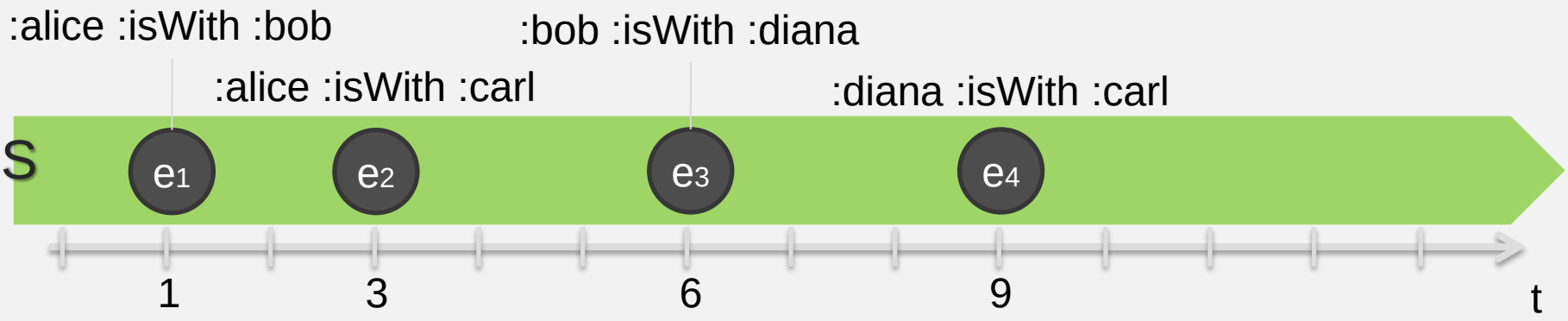
- A timestamp is a temporal identifier associated to a data item
- The **application time** is a set of one or more timestamps associated to the data item
- Two data items can have the same application time
  - Contemporaneity
- Who does assign the application time to an event?
  - The one that generates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

# Missing application time



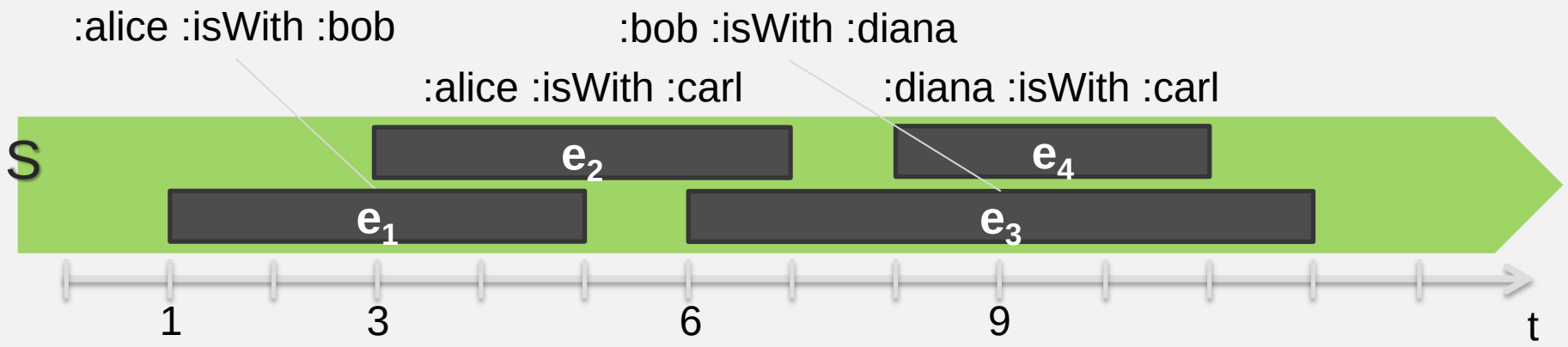

- A RDF stream without timestamp is an ordered sequence of data items
- The order can be exploited to perform queries
  - Does Alice meet Bob before Carl?
  - Who does Carl meet first?

# Application time: point-based extension




- One timestamp: the time instant 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: interval-based extension




- 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?

# Outline

1. Continuous RDF model extensions
   - RDF Streams, timestamps
2. Continuous extensions of SPARQL
   - Continuous evaluation
   - Additional operators
3. Overview of existing systems
   - Features
   - Comparison

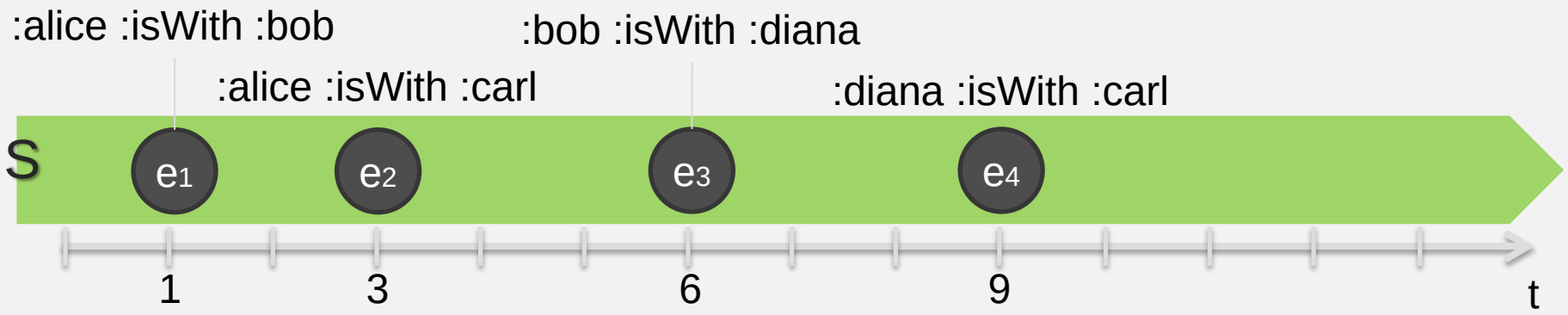
# Continuous query evaluation

- From SPARQL
  - One query, one answer
  - The query is sent after that the data is available
- To a continuous query language
  - One query, multiple answers
  - The query is **registered** in the query engine
  - The registration usually happens before that the data arrives
  - Real-time responsiveness is usually required

# Let's process the RDF streams!

- In literature there are two different main approaches to process streams
- Data Stream Management Systems (DSMSs)
  - Roots in DBMS research
  - Aggregations and filters
- Complex Event Processors (CEPs)
  - Roots in Discrete Event Simulation
  - Search of relevant patterns in the stream
  - Non-equi-join on the timestamps (after, before, etc.)
- Current systems implements feature of both of them
  - EPL (e.g. Esper, ORACLE CEP)
- Now we focus on the CQL/STREAM model
  - Developed in the DSMS research
  - C-SPARQL (and others) is inspired to this model

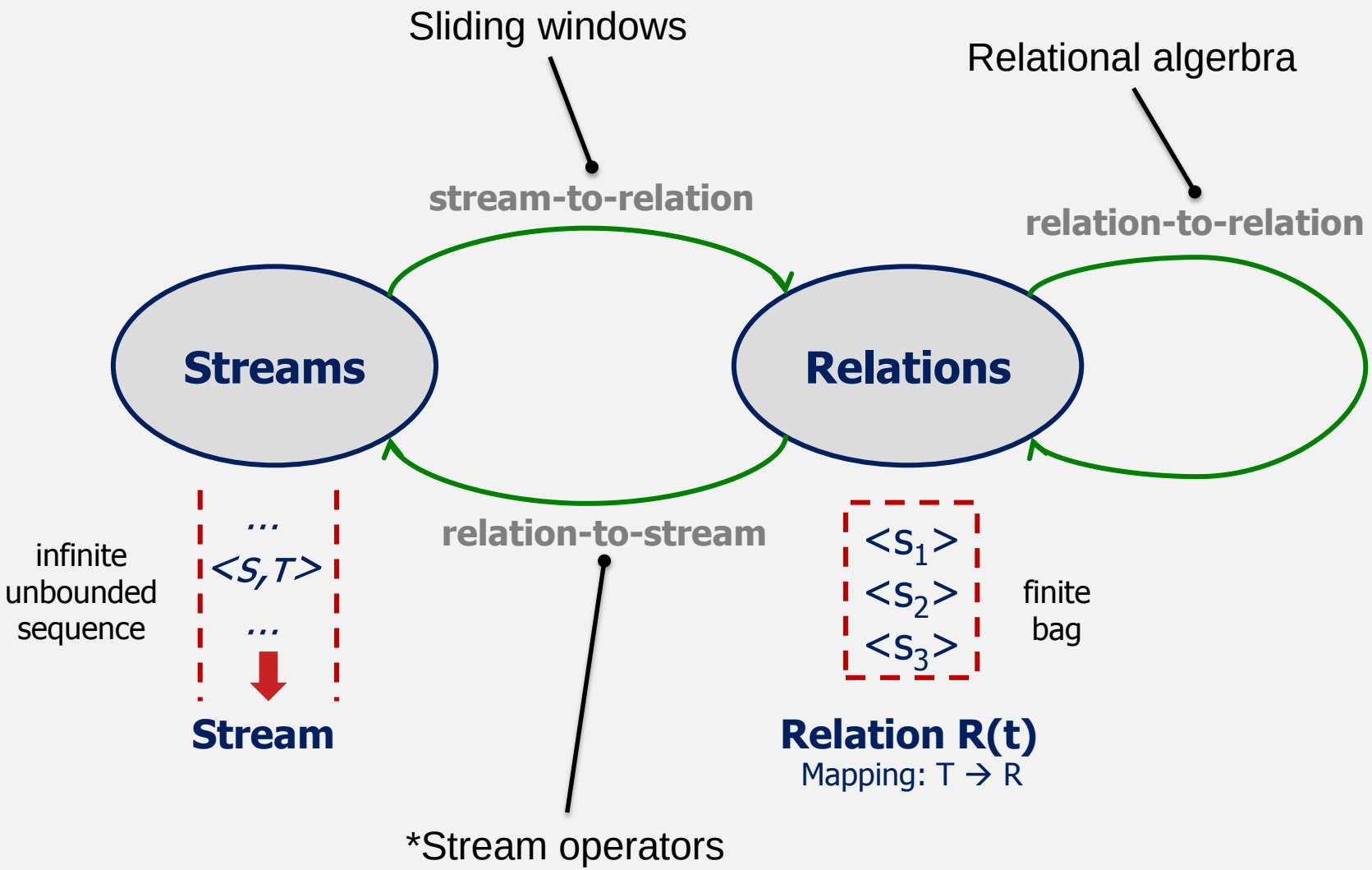
# Our assumptions



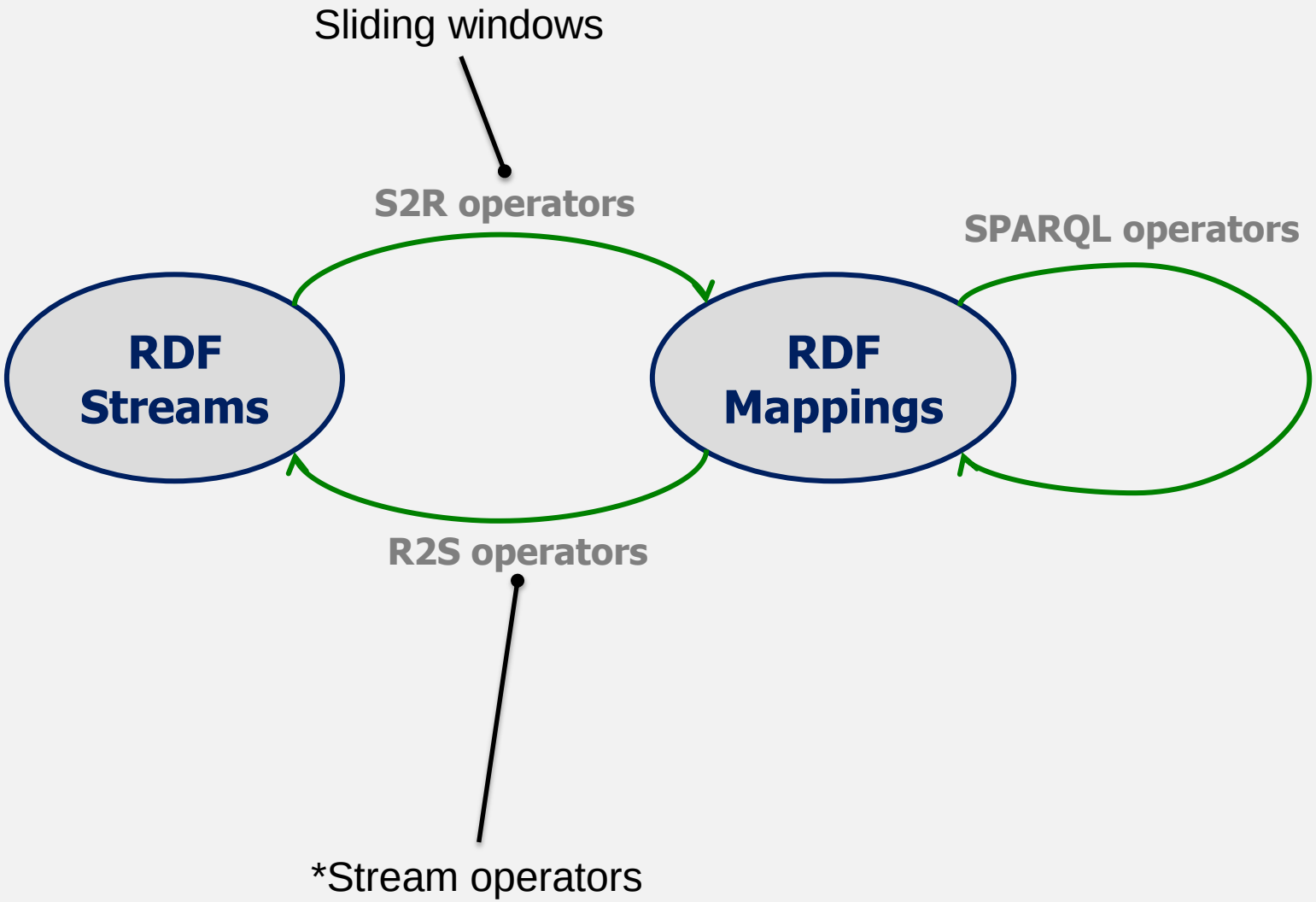

- In the following we will consider the following setting
  - A RDF triple is an event
  - Application time: point-based

```
<:alice :isWith:bob>:[1]
<:alice :isWith:carl>:[3]
<:bob :isWith :diana>:[6]
...
```

# Querying data streams – The CQL model



# CQL extension for querying RDF data streams



# Time-based sliding window


slide

width

ω

β

R2R operator

W(ω,β)

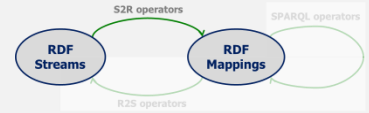
S

$S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$ $S_8$ $S_9$ $S_{10}$ $S_{11}$ $S_{12}$

t

# Time-based sliding window - tumbling

# Tuple-based sliding window




ω tuples in the window

Slide of β tuples

R2R operator

Contemporaneity implies a non-deterministic selection

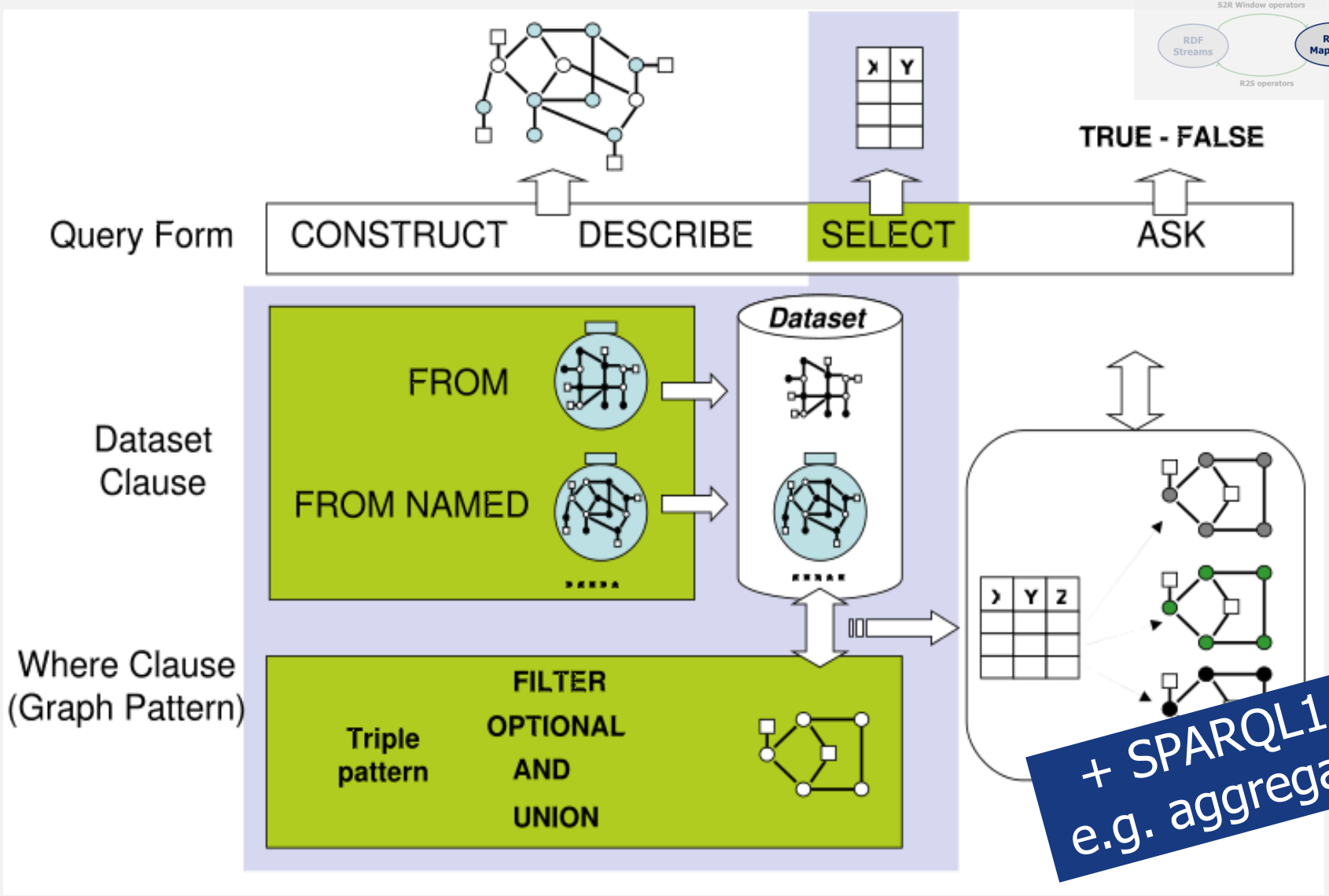
W(ω,β)

S

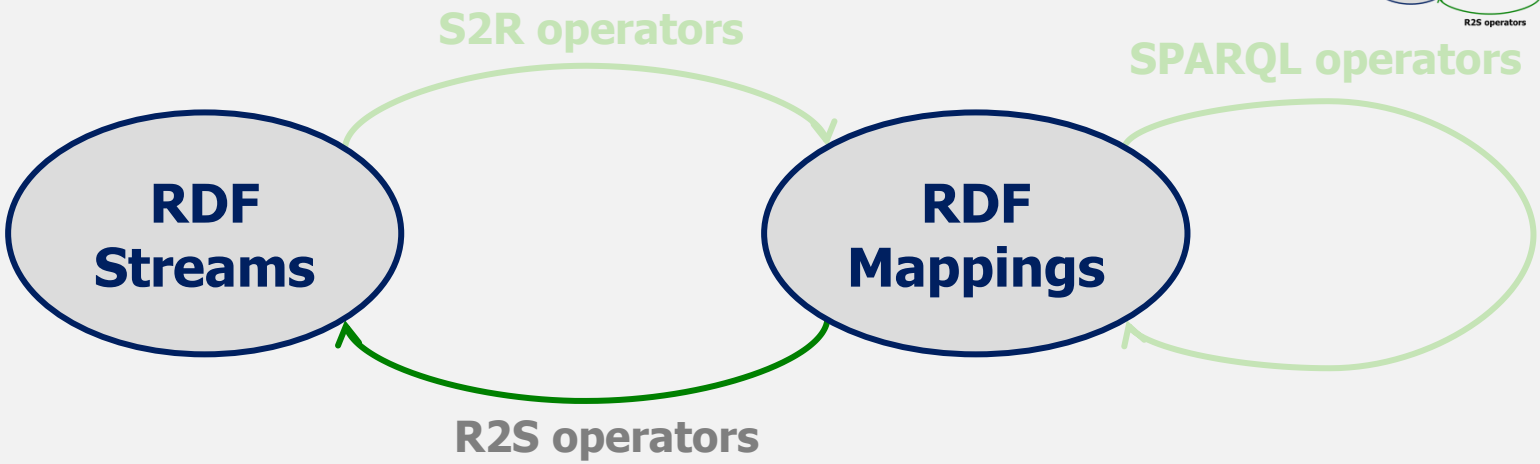
$S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$ $S_8$ $S_9$ $S_{11}$ $S_{12}$

t

# SPARQL: a quick recap



# The query output




- Which is the format of the answer?
- We can distinguish two cases
  1. No R2S operator: the output is a relation (that changes during the time)
  2. R2S operator: a stream.
     - An RDF stream? It depends by the Query Form

# No R2S operator: relation


SELECT ?a ?b …
FROM ….
WHERE ….

queries

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

RSP

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]

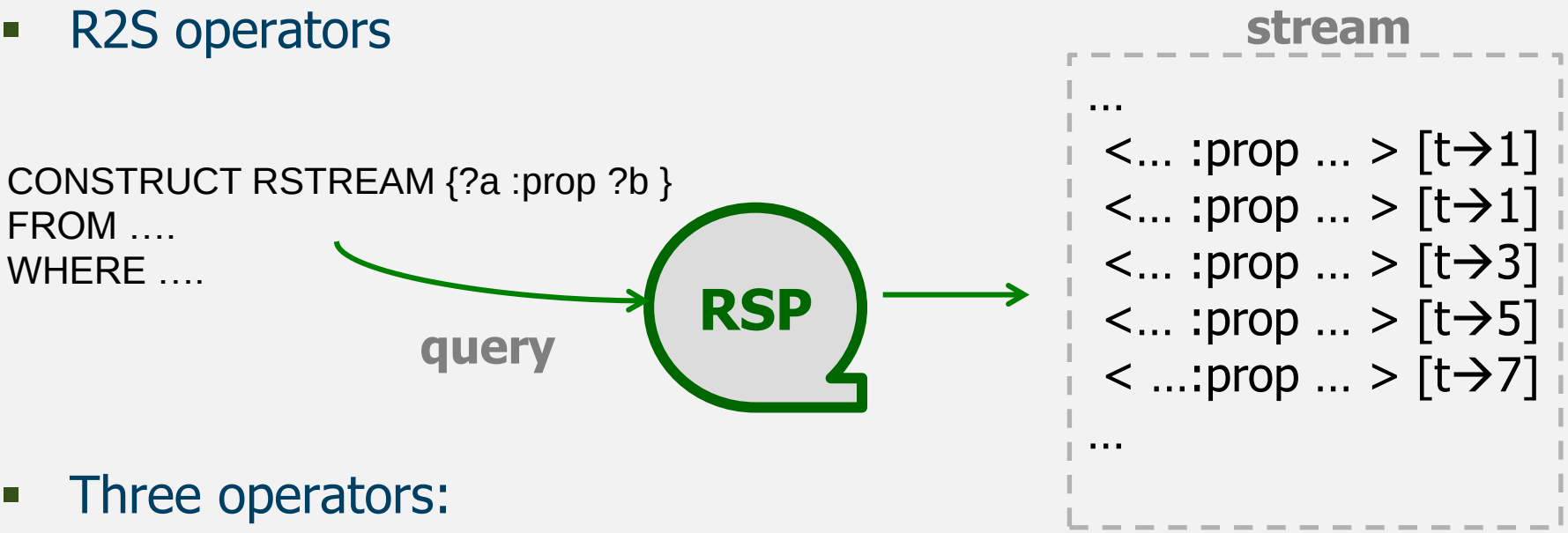
<… :prop … > [t→5]

<… :prop … > [t→7]

triples

# R2S operator: stream

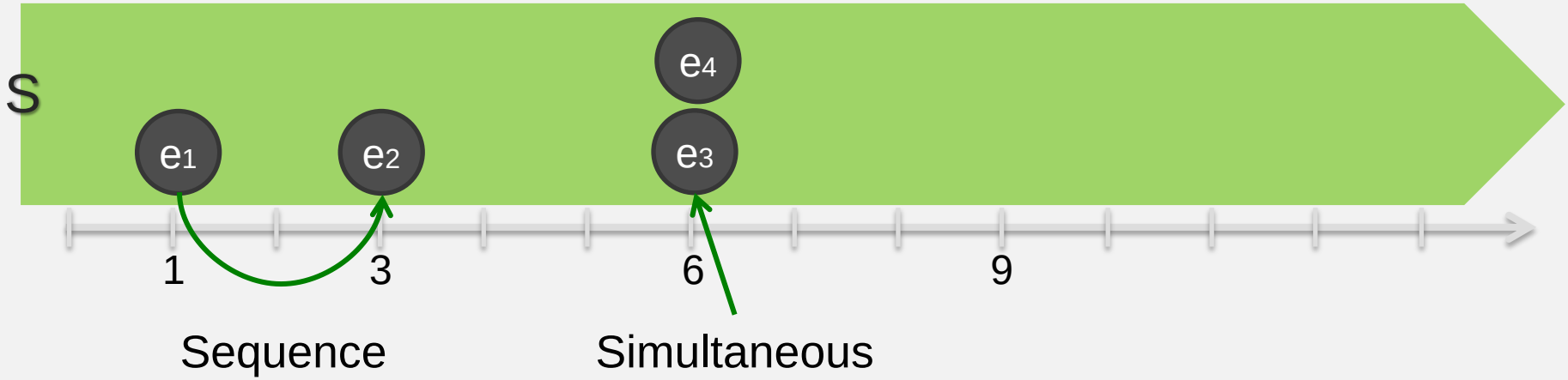
- R2S operators




- Three operators:
  - Rstream: streams out all data in the last step
  - Istream: streams out data in the last step that wasn't on the previous step, i.e. streams out what is **new**
  - Dstream: streams out data in the previous step that isn't in the last step, i.e. streams out what is **old**

# Brief overview on the CEP operators

- Sequence operators and CEP world



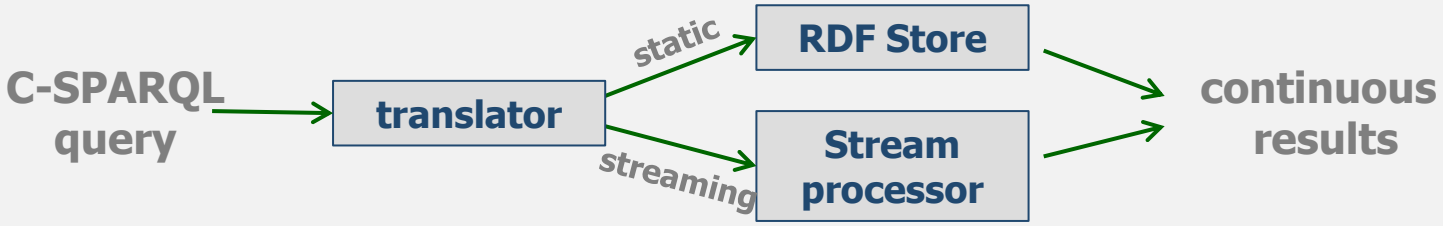

- SEQ: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if e' occurs after e
- EQUALS: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if they occur simultaneously
- OPTIONALSEQ, OPTIONALEQUALS: Optional join variants

# Outline

1. Continuous RDF model extensions
   - RDF Streams, timestamps
2. Continuous extensions of SPARQL
   - Continuous evaluation
   - Additional operators
3. Overview of existing systems
   - Features
   - Comparison

# Existing RSP systems (oversimplified!)

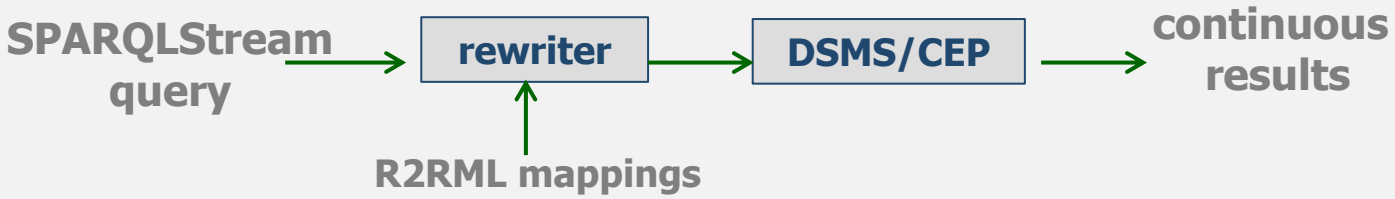
- C-SPARQL: RDF Store + Stream processor
  - Combined architecture



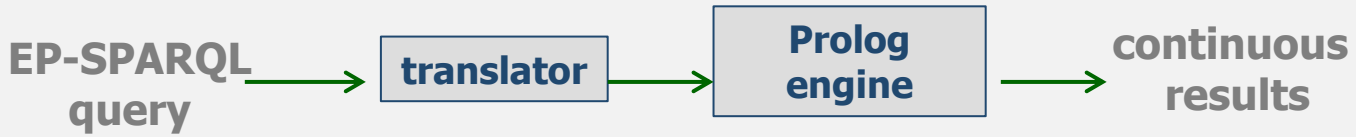

- CQELS: Implemented from scratch. Focus on performance
  - Native + adaptive joins for static-data and streaming data

# Existing RSP systems (oversimplified!)

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




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




- Instans: RETE-based evaluation

# Classification of existing systems

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

Disclaimer: other features may be missing

# Similar models, similar (not equals!) query languages

**SPARQLStream**

```
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 )
```

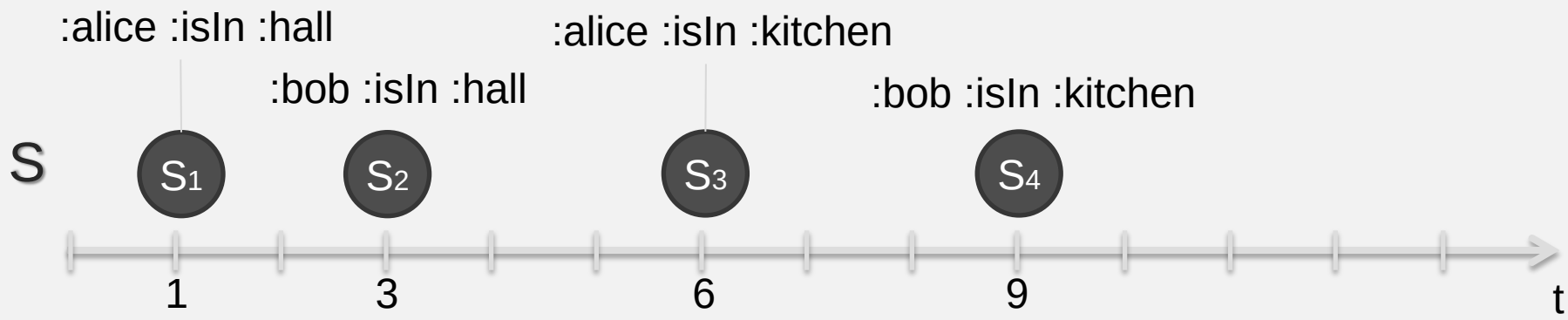
**C-SPARQL**

```
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 )
```

**CQELS**

```
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 )
```

# The correctness problem (1)

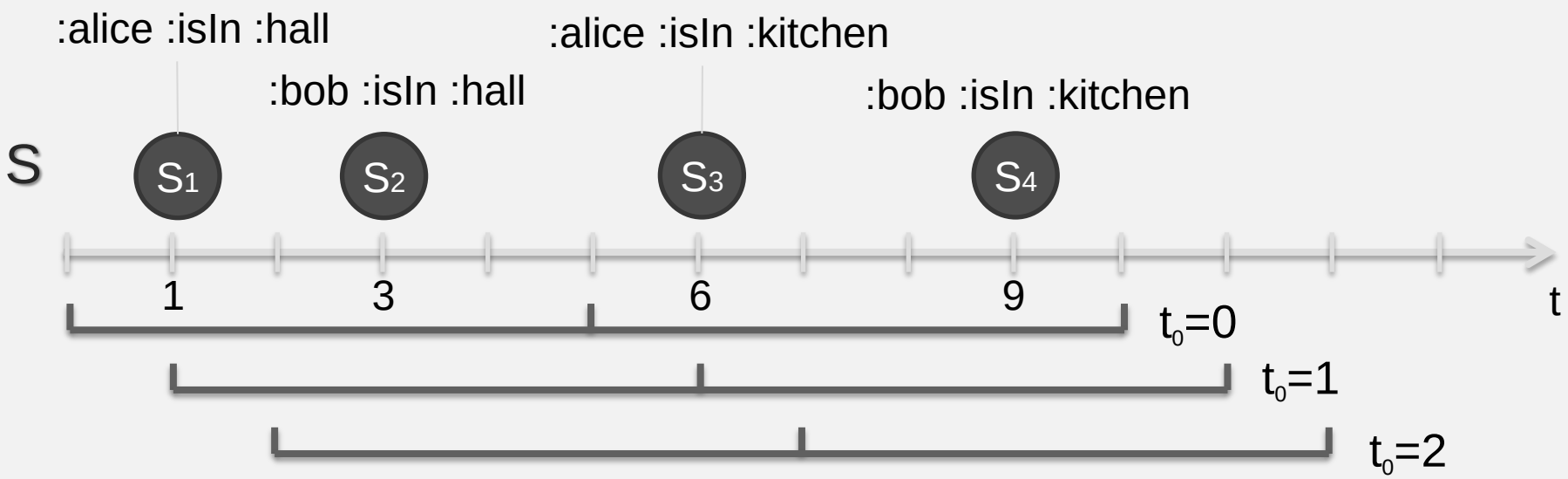



- Where are Alice and Bob, when they are together?
- Let's consider a tumbling window W(ω=β=5)
- Let's execute the experiment 4 times on C-SPARQL

| Execution | 1° answer | 2° answer |
|---|---|---|
| 1 | :hall [6] | :kitchen [11] |
| 2 | :hall [5] | :kitchen [10] |
| 3 | :hall [6] | :kitchen [11] |
| 4 | - [7] | - [12] |

**Which is the correct answer?**

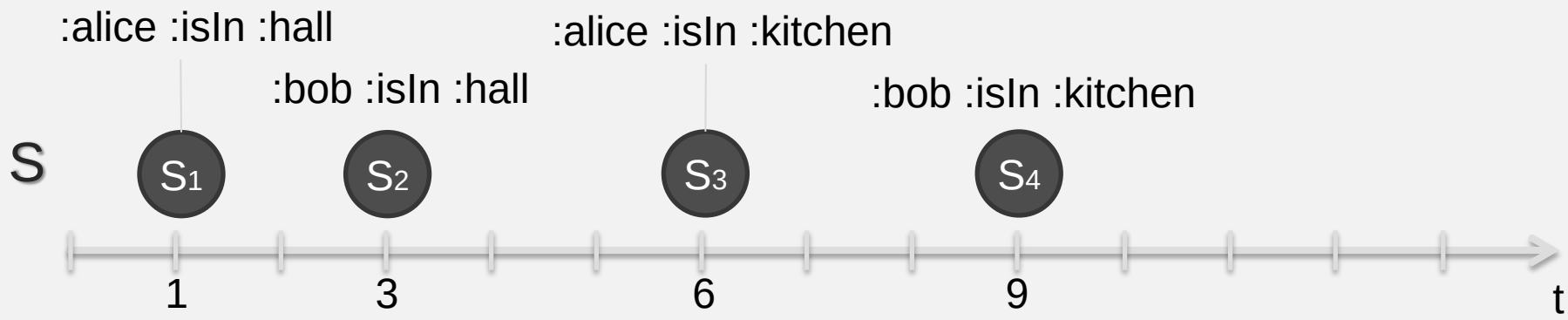
# RSP output correctness: the $t_0$ parameter




| Exec | 1° answer | 2° answer |
|---|---|---|
| 1 | :hall [6] | :kitchen [11] |
| 2 | :hall [5] | :kitchen [10] |
| 3 | :hall [6] | :kitchen [11] |
| 4 | - [7] | - [12] |

| Window | 1° answer | 2° answer |
|---|---|---|
| $t_0$=0 | :hall [5] | :kitchen [10] |
| $t_0$=1 | :hall [6] | :kitchen [11] |
| $t_0$=2 | - [7] | - [12] |

# The correctness problem (2)




C-SPARQL

| Execution | 1° answer | 2° answer |
|---|---|---|
| 1 | :hall [6] | :kitchen [11] |
| 2 | :hall [5] | :kitchen [10] |
| 3 | :hall [6] | :kitchen [11] |
| 4 | - [7] | - [12] |

CQELS

| Execution | 1° answer | 2° answer |
|---|---|---|
| 1 | :hall [3] | :kitchen [9] |
| 2 | No answers | |
| 3 | :hall [3] | :kitchen [9] |
| 4 | No answers | |

**Which system behaves in the correct way?**

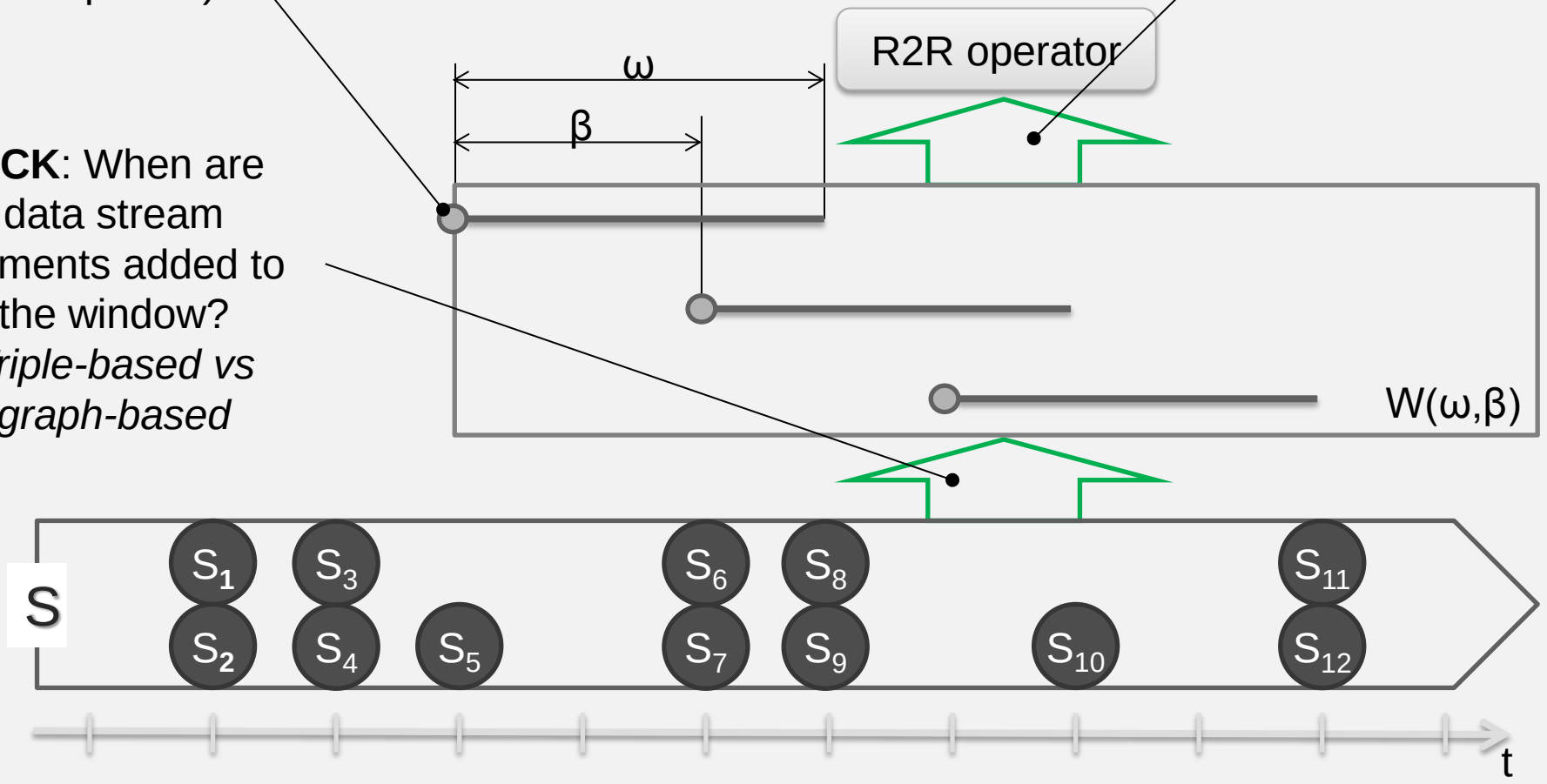
**Both!**

# The window operator (through SECRET)

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

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

**TICK**: When are data stream elements added to the window? *Triple-based vs graph-based*



# Understanding the RSPs

- They share similar models, but they behave in different ways
- The C-SPARQL, CQELS and SPARQLstream models does not allow to determine in a unique way which should be the answer given the inputs and the query
  - There are missing parameters (encoded in the implementations)
- Why is it important to understand those behaviours?
  - To assess the correct implementation of the systems
  - To improve the comprehension of the benchmarking

# References


- DSMSs and CEPs
  - Arasu, A., Babu, S., Widom, J.: The CQL continuous query language : semantic foundations. The VLDB Journal 15(2) (2006) 121–142
  - Gianpaolo Cugola, Alessandro Margara: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. 44(3): 15 (2012)
  - 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
- RDF Stream Processors
  - 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
  - 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
- Benchmarks and RSP comparison
  - Ying Zhang, Minh-Duc Pham, Óscar Corcho, Jean-Paul Calbimonte: SRBench: A Streaming RDF/SPARQL Benchmark. International Semantic Web Conference (1) 2012: 641-657
  - Danh Le Phuoc, Minh Dao-Tran, Minh-Duc Pham, Peter A. Boncz, Thomas Eiter, Michael Fink: Linked Stream Data Processing Engines: Facts and Figures. International Semantic Web Conference (2) 2012: 300-312
  - Daniele Dell'Aglio, Jean-Paul Calbimonte, Marco Balduini, Óscar Corcho, Emanuele Della Valle: On Correctness in RDF Stream Processor Benchmarking. International Semantic Web Conference (2) 2013: 326-342

# Tutorial on RDF Stream Processing

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

http://streamreasoning.org/rsp2014





# RDF Stream models
# Continuous query models

Daniele Dell'Aglio

daniele.dellaglio@polimi.it