How to Build a Stream Reasoning Application
D. Dell’Aglio, E. Della Valle,
T. Le-Pham, A. Mileo, and R. Tommasini
http://streamreasoning.org/events/streamapp2017

Introduction and RDF streams
Daniele Dell’Aglio

mail: dellaglio@ifi.uzh.ch  
url: http://dellaglio.org  
twitter: @dandellaglio
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 a credits slide stating
  
    These slides are partially based on “How to Build a Stream Reasoning Application 2017” by D. Dell'Aglio, E. Della Valle, T. Le-Pham, A. Mileo, and R. Tommasini available online at http://streamreasoning.org/events/streamapp2017

- To view a copy of this license, visit
  
  http://creativecommons.org/licenses/by/3.0/
Structure of the tutorial

Introduction and RDF streams
Introduction to the hands-on platform
Continuous queries
Hands-on on querying RDF streams

Coffee break

DL-based stream reasoning
Hands-on on DL reasoning on RDF streams

ASP-based stream reasoning
Hands-on on ASP reasoning on RDF streams

Conclusions and final remarks
Continuous extensions of RDF

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

  \[
  \langle \text{sub}_1 \ \text{pred}_1 \ \text{obj}_1 \rangle \\
  \langle \text{sub}_2 \ \text{pred}_2 \ \text{obj}_2 \rangle
  \]

- 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?
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!
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?
In this tutorial we consider the following setting:

- A RDF graph is an event
- Application time: point-based

```
{:alice :isWith :bob.} [1]
{:alice :isWith :carl.} [3]
{:bob :isWith :diana, :carl.} [6]
...
```
A serialization format for RDF streams

An RDF stream can be represented as an (infinite) ordered sequence of time-annotated **data items** (RDF graphs)...

... serialized in JSON-LD

```
[{
  "@graph": {
    "@id": "http://.../G1",
    {"@id": "http://.../a",
     "http://.../isIn": {"@id": "http://.../rRoom"}}
  },
  {"id": "http://.../G1",
   "generatedAt": "2016-16-12T00:01:00"
  }
},
{
  "@graph": {
    "@id": "http://.../G2",
    {"@id": "http://.../b",
     "http://.../isIn": {"@id": "http://.../rRoom"}}
  },
  {"id": "http://.../G2",
   "generatedAt": "2016-16-12T00:03:00"
  }
},
{
  "@id": "http://.../G3",
  "generatedAt": "2016-16-12T00:03:00"
},...
```
RDF Stream Descriptor

- In addition to the stream itself, we need a way to provide information *about* the stream
  - e.g. who’s the owner of the stream; how to access it

- We introduce the notion of RDF stream descriptor
  - Also known as *sGraph*
  - It contains:
    - The identifier of the stream
    - Data item samples (see next slide)
    - A description of the schema
    - The location of the stream endpoint (e.g. WebSocket URL)
RDF Stream Descriptor

```json
{
  "@graph": {
    "@id": "http://knoesis.wright.edu/ssw/Observation_RelativeHumidity_C1390_2004_08_08_06_05_00",
    "@type": "http://knoesis.wright.edu/ssw/ont/weather.owl#RelativeHumidityObservation",
    "observedProperty": "http://knoesis.wright.edu/ssw/ont/weather.owl#_RelativeHumidity",
    "procedure": "http://knoesis.wright.edu/ssw/System_C1390",
    "result": "http://knoesis.wright.edu/ssw/MeasureData_RelativeHumidity_C1390_2004_08_08_06_05_00",
    "samplingTime": "http://knoesis.wright.edu/ssw/Instant_2004_08_08_06_05_00",
    "@context": {
      "procedure": {
        "@id": "http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#procedure",
        "@type": "@id"
      },
      "samplingTime": {
        "@id": "http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#samplingTime",
        "@type": "@id"
      },
      "observedProperty": {
        "@id": "http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#observedProperty",
        "@type": "@id"
      },
      "@id": "http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#result",
      "@type": "@id"
    }
  }
}
```
TripleWave is an open-source framework for creating and publishing RDF streams over the Web.

Feeding TripleWave

- TripleWave supports a variety of data sources.
  - RDF dumps with temporal information
  - RDF with temporal information exposed through SPARQL endpoints
  - Streams available on the Web
From RDF to RDF streams

- Converts RDF stored in files/SPARQL endpoints
  - Containing some time information

- ... into an RDF stream
  - **continuous flow** of RDF data
  - **ordered** according the original timestamps
  - the time between two items is preserved

- Use Cases
  - Evaluation, testing and benchmarking
  - Simulation systems
Building TripleWave

Time-annotated RDF datasets → RDF → Replay Loop → RDF Stream Descriptor → RDF Streams

Finite RDF substreams

TripleWave

(RDF | Web socket | HTTP-chunk | JSON-LD)

etc.
Consumes an existing **Web stream**...  
• through connectors  

... and converts it into an RDF Stream  
• Each data item is lifted to RDF  

**Use Cases**  
• Querying and reasoning  
• Data integration
Building TripleWave

- Time-annotated RDF datasets
- Finite RDF substreams
- Live Non-RDF Streams

Conversion to RDF Stream

RDF Stream Descriptor

RDF Streams

(Web socket | HTTP-chunk | etc.)

R2RML Mappings
Introduction
Daniele Dell’Aglio

Introduction
Daniele Dell’Aglio