IoT-Intelligence and Web Stream Reasoning

Towards Complex Reasoning over Big Data Streams with Answer Set Programming

Alessandra Mileo
Senior Research Fellow
Insight Centre for Data Analytics, NUIG
alessandra.mileo@insight-centre.org

Ali Intizar
Research Fellow
Insight Centre for Data Analytics, NUIG
ali.intizar@insight-centre.org
**IoT- Intelligence at Unit for Reasoning & Querying (URQ)**

1. Representation and linking
2. Finding what we need
3. Dynamic Problem Solving (Stream Reasoning)
IoT- Intelligence at *Unit for Reasoning & Querying (URQ)*

1. Representation and linking
2. Finding what we need
3. Dynamic Problem Solving (Stream Reasoning)

Smart Cities

Enterprise Communication
What streams do I need and how “good” are they?
(Stream Discovery and Federation)
Quality and context-aware stream discovery

• What information do I need?
  – Data interoperability: Semantic descriptions
  – Interface interoperability: streams as event services

• How good is it?
  – ADAPT to quality requirements and preferences for data source selection
  – Efficient processing of event logic
Quality and context-aware stream discovery

• What information do I need?
  – Data interoperability: Semantic descriptions
  – Interface interoperability: streams as event services

• How good is it?
  – ADAPT to quality requirements and preferences for data source selection
  – Efficient processing of event logic
Quality and context-aware stream discovery

• What information do I need?
  – Data interoperability: Semantic descriptions
  – Interface interoperability: streams as event services

• How good is it?
  – ADAPT to quality requirements and preferences for data source selection
  – Efficient processing of event logic
Quality and context-aware stream discovery

• What information do I need?
  – Data interoperability: Semantic descriptions
  – Interface interoperability: streams as event services

• How good is it?
  – ADAPT to quality requirements and preferences for data source selection
  – Efficient processing of event logic
Quality and context-aware stream discovery

- What information do I need?
  - Data interoperability: Semantic descriptions
  - Interface interoperability: streams as event services
- How good is it?
  - ADAPT to quality requirements and preferences for data source selection
  - Efficient processing of event logic

Semantic Web
Service Oriented Architectures
Continuous constraint checking
DSMS and CEP
Summary of the Approach

- How to describe complex event services?
  - Create an Event Service Ontology with Event Patterns.
- How to determine if two event patterns are functionally equivalent?
  - Create and compare canonical event patterns to find substitutes.
- How to create event compositions and choose the optimal?
  - Top-down traverse to find functionally-equivalent canonical patterns.
- How to derive event service compositions efficiently?
  - Construct and utilize an Event Reusability Hierarchy for event service composition.

Automated Complex Event Implementation System

Stream Reasoning Workshop, Vienna, 9th November 2015
IoT - Intelligence at Unit for Reasoning & Querying (URQ)

1. Representation and linking
2. Finding what we need
3. Dynamic Problem Solving (Stream Reasoning)

Smart Cities
Enterprise Communication
How to leverage the IoT and Semantic Web infrastructure for (efficient) Web Stream Reasoning?
The StreamRule idea

• 2-tier approach: not all data streams are relevant for complex reasoning
• Enrich the ability of complex reasoning over data streams
• Keep the solution scalable
• Leverage existing engines from both stream processing and non-monotonic reasoning research areas

Limitations

- The more expressive the inference task, the longer it takes to perform reasoning
- Bottleneck when results are returned not as fast as the next input arrives
Limitations

• The more expressive the inference task, the longer it takes to perform reasoning

• Bottleneck when results are returned not as fast as the next input arrives
Adaptation Heuristics: ongoing work

- More than an engineering problem
  - How to model interactions between RSP and ASP components, including different semantics, input split, window-size tuning,…

- Design and runtime features
  - E.g. operational semantics (design) and throughput (runtime)

- Streaming rate and window size: where’s the tradeoff?

- Reasoning Complexity: how far can we go? How can we parametrise the complexity to estimate the execution time
Given a fixed streaming size $S$ with fixed complexity $C$ and unit of time $U$, find a window size $W$ such that the time required to process $S$ events using windows of size $W$ is less than or equal to one unit of time $U$

$$T_\omega(S, W) \leq U$$

Experimental setup

• Dataset
  – Simulated randomly generated events of the type
    \texttt{event(type, name, value, latitude, longitude)}
    
    \textit{E.g. event(weather, strong-wind, 2014-11-26T13:00:00, 38.011736, 12.186724)}

• Reasoning tasks
  – Ranking event criticality
  – Contextualizing events based on user status
  – Default rule to detect changes in event criticality

• Run
  – Streaming size up to 30000
  – Reasoner triggered 20 times for each $S$
Empirical Results
Empirical Results

\[ T(20000) = 1232 \text{ ms} \]
Empirical Results

\[ T(5000) = 216 \text{ ms} \]

\[ T(20000) = 1232 \text{ ms} \]
Empirical Results

\[ T_\omega(20000, 5000) = \left[ \frac{20000}{5000} \right] \times T(5000) = 4 \times 216 \text{ ms} = 864 \text{ ms} \]

\[ T(5000) = 216 \text{ ms} \]

\[ T(20000) = 1232 \text{ ms} \]
Empirical Results

\[ T_\omega(20000, 5000) = \left[ \frac{20000}{2000} \right] \times T(2000) = 10 \times 72 \text{ ms} = 720 \text{ ms} \]

\[ T(5000) = 216 \text{ ms} \]

\[ T(20000) = 1232 \text{ ms} \]
Ongoing work

- Relaxing the independence assumption
  - Extended notion of dependency graph
  - Possibly use duplication
  - Smaller input sets given to the same ASP program
  - Demonstrate correctness of results

- Going parallel
  - Explore parallelism of SPARK for higher scalability
  - Requires to map an ASP program to SPARK jobs

- Correlation between reasoning complexity and execution time
Dealing with Uncertainty and learning relational structures
IoT data are messy: deal with uncertainty

• Expressive inference
  – non-monotonicity, noisy, partial and inconsistent data
• “ease of” declarative logic-based reasoning to model a problem/domain. Still we need to manage uncertainty and non-monotonicity
• Probabilistic rules for uncertain knowledge and learning by example
  – represent, use, infer and learn probabilistic knowledge (PrASP)

Can we (learn the) answer to questions about uncertain knowledge using qualitative (declarative) inference in dynamic environments?
What is Streaming PrASP

A framework that uses:

1. PrASP as an uncertainty reasoning server to reason over Streaming Web Data
2. Continuous Query Processing over Linked Data Streams for data filtering

What is Streaming PrASP

A framework that uses:

1. PrASP as an uncertainty reasoning server to reason over Streaming Web Data
2. Continuous Query Processing over Linked Data Streams for data filtering


What is PrASP then?
PrASP is...

... an experimental Statistical Relational Learning (SRL) reasoner based on Answer Set Programming (ASP)

PrASP can...

... represent, use infer and learn probabilistic knowledge

Matthias Nickles, Alessandra Mileo: A System for Probabilistic Inductive Answer Set Programming. SUM 2015: 99-105
Streaming PrASP

- Streaming **new beliefs** are added incrementally to a loaded PrASP program
- Streaming **new learning examples** are added to the set of learning examples $E$

- Assert/Retract, time decay and sliding windows supported

- Windows prefixes realized by a caching mechanism (no reactive ASP used) for faster processing

- Preprocessing based on RDF query processing over streams (SPARQL 1.1 + streaming operators)
Streaming PrASP framework

Query & learning results → PrASP CQELS client (RDF filtering client) → CQELS → RDF data stream

CQELS/SPARQL query → window → Incremental results

transformation pattern → Probabilistic ASP/FOL beliefs/examples stream → window

PrASP {PrASP program (knowledge base), Probabilistic queries} → Hypotheses

ASP grounder/solver
Streaming PrASP framework

- Query & learning results
- PrASP CQELS client (RDF filtering client)
- CQELS
- RDF data stream
- PrASP program (knowledge base)
- PrASP
- PrASP CQELS client
- Probabilistic ASP/FOL beliefs/examples stream
- CQELS/SPARQL query
- Incremental results
- Transformation pattern
- RDF data stream
- Probabilistic queries
- Static learning examples
- Hypotheses
- ASP grounder/solver
- PrASP
- RDP filetring engine
- PrASP CQELS client
- Probabilistic ASP/FOL beliefs/examples stream
- Window
Ongoing work

• More experiments on Web Stream Reasoning with PrASP
  – Looking for Use Case Scenarios and DATA to test PrASP
  – Building a set of modules that can be downloaded and used for feedback

• Continuously exploring options for optimization, especially in the learning task

• More ambitious goals
  – Structure Learning
  – Relation between Streaming ASP and PrASP (currently not using Streaming ASP)
Data Streams

Stream Reasoning

Semantic Complex Event Processing

Stream Query Processing

Applications

Expressivity

Scalability

relevant events

solution sets

complex events
IoT-Intelligence and Web Stream Reasoning

Towards Complex Reasoning over Big Data Streams with Answer Set Programming

Alessandra Mileo
Senior Research Fellow
Insight Centre for Data Analytics, NUIG
alessandra.mileo@insight-centre-org

Ali Intizar
Research Fellow
Insight Centre for Data Analytics, NUIG
ali.intizar@insight-centre-org