Stream Reasoning for the Internet of Things: Challenges and Gap Analysis

Xiang Su, Ekaterina Gilman*, Peter Wetz†,*
Jukka Riekki, Yifei Zuo* & Teemu Leppänen**

**University of Oulu, Finland*
†TU Wien, Austria

UNIVERSITY of OULU
OULUN YLIOPISTO

Introduction

- This research focuses on extracting actionable knowledge and providing value-added services by means of reasoning techniques for IoT application and services
- Deduce timely, sufficiently accurate, and reliable knowledge.
- Stream Reasoning
  - Stream reasoning combines reasoning and stream processing techniques
  - processing several streams on-the-fly, and implementing realtime services
  - designed for processing dynamic, heterogeneous, and scalable data for IoT

IoT Challenges for Stream Reasoning

- Variety, Velocity, and Volume of Data Streams (C1)
  - Big amount of information collected from IoT
  - information is continuously generated at high sampling rates
  - Requires integration of multiple sources
  - different formats and various models
  - integration of data coming from IoT data streams with background knowledge
  - Recent efforts in the Semantic Web field, such as Semantic Sensor Network Ontology, have been introduced to enable semantic interoperability.

IoT Challenges for Stream Reasoning

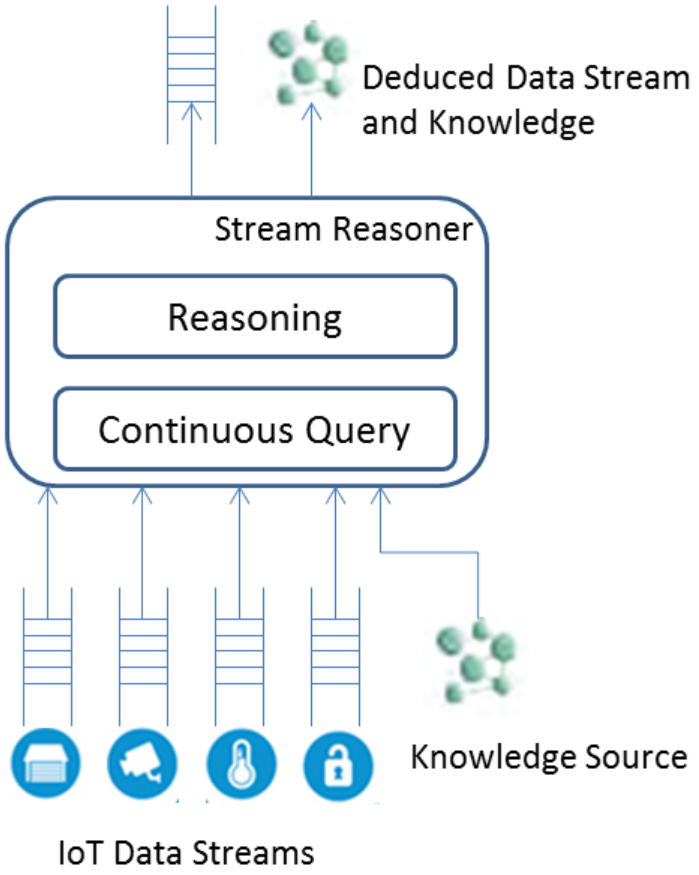
- Efficiency (C2)
  - low latency reasoning, i.e. timely generation of new knowledge before it becomes outdated
  - reasoning on resource-constrained devices and systems, i.e. delivery of reliable solutions considering computing, storage, communication, and energy limitations of IoT devices and protocols
  - Standard semantic technologies require expressive representations and complex reasoning techniques, which require a considerable amount of resources unavailable in IoT systems

IoT Challenges for Stream Reasoning

- Semantic Expressive Power (C3)
  - Suitable knowledge models and processing approaches
  - RDF needs to be extended to consider data that changes frequently over time
- Robustness (C4)
  - IoT data are often unreliable, incomplete, arrive out-of-order or even get lost
  - How to guarantee complete reasoning results?
  - Because a stream is observed only through a window of finite size

Stream Reasoner

- Applications and IoT devices generate IoT data streams, which are unbounded sequences of time-varying data elements
- Simultaneous queries can be passed to a reasoner as an input.
- A timebased data model where data items can be annotated with timestamps,
- Background knowledge, constitute knowledge sources essential to deduce new knowledge.
- Networked stream reasoners.



C-SPARQL

- A language for continuous queries over streams of RDF data
- One of the first contributions in stream reasoning
- C-SPARQL extends SPARQL for querying RDF streams
- Support (1) **timestamped RDF triples**, (2**) continuous queries** over RDF streams, and (3) **integration and processing both background data and streams**
- An RDF stream is defined as an ordered sequence of pairs, where each pair is constituted by an RDF triple and its timestamp t:

  (<Subject$_i$, Predicate$_i$, Object$_i$ > t$_i$)

IMaRS (Incremental Materialization for RDF Streams***)***

- Developed on top of C-SPARQL and it focuses on **materialization**
  - Materialization is making all possible conclusions from a given input explicit, meaning no further reasoning is needed when the user queries the knowledge base
- an incremental reasoning approach and the specific data and processing models of C-SPARQL to compute the expiration time of streaming RDF triples
- IMaRS relies on the strong assumption that the expiration time of each triple can be pre-computed, which limits its applicability

TrOWL

- A notable reasoner for **incremental reasoning**.
- TrOWL supports more complex ontology languages, covering the expressive power of **OWL2 DL**
- TrOWL utilizes **syntactic approximation** to reduce reasoning complexity
  - Syntactic approximation ensures that all derived knowledge is correct, i.e., it ensures soundness, but not completeness of reasoning
- TrOWL is designed for more **dynamic knowledge**
  - The design of TrOWL considers updates from 20% to 80% of the ontology and the authors measure a processing time that varies from 20% to 70% with respect to the naive approach of recomputing all derivations

Deductive and inductive stream reasoning

- An extension to the C-SPARQL model
- Propose to combine inductive with deductive reasoning to **increase result accuracy**
- Proposed approach exhibits **low processing delays** which remarkably outperforms SPARQL query engines
- Inductive stream reasoning deals with mining of large portions of data and applying statistical and machine learning techniques to extract new knowledge
- Inductive stream reasoning supports heterogeneous data integration and possible materialization

C-SPARQL on S4

- Implements partial RDFS reasoning and part of the C-SPARQL query language on the **S4 streaming platform**
- C-SPARQL on S4 enables splitting the processing load over multiple machines
- An important effort to improve **scalability and efficiency**, especially useful in IoT related scenarios

CQELS

- Another comparable system with C-SPARQL
- Designed for **combining static and streaming linked data**.
- CQELS offers a processing model in which query evaluation is not periodic, but triggered by the arrival of new triples
- CQELS **strictly integrates the evaluation of background and streaming data**, without delegating them to external components
- Possible to apply query rewriting techniques and optimizations well studied in the field of relational databases

Sparkwave

- A system designed for **high performance** and on-the-fly reasoning over RDF data streams
- It trades complexity for performance
- Sparkwave implements a variant of the RETE algorithm

- Sparkwave poses several **limitations to the size of the background knowledge**, which has to fit into the main memory of a single machine
- Moreover, it operates with a pre-loaded RDF schema and provides limited reasoning functionality

EP-SPARQL

- EP-SPARQL is a solution that inherits the **language constructs and processing model of Complex Event Processing (CEP) systems**
- EP-SPARQL also provides windowing operators for isolating portions of the streams
- EP-SPARQL focuses more on detection of RDF triples in a specific temporal order
- Another important difference of EP-SPARQL with respect to other languages is in the data model and consists in the way time is associated to RDF triples

STARQL

- An expressive and flexible stream query **framework** that offers the possibility to embed different temporal description logics as filter query languages over ontologies
- STARQL can **embed various query languages** that can be combined with more expressive DL ontology languages
- Support orthogonal approaches such as ABox modularization, a technique that supports accessing big data over very expressive ontologies
- This framework utilizes a First Order Logic (FOL) fragment for temporal reasoning over ABox sequences constructed within the query

Others

- ***Streaming SPARQL***
- Streaming SPARQL is another extension of SPARQL for processing RDF streams
- The main contributions of this work are theoretical

- ***Streaming Knowledge Bases***
- built on top of the TelegraphCQ DSMS and provides reasoning using a subset of RDFS and OWL over streaming RDF triples

Time-aware reasoning approaches

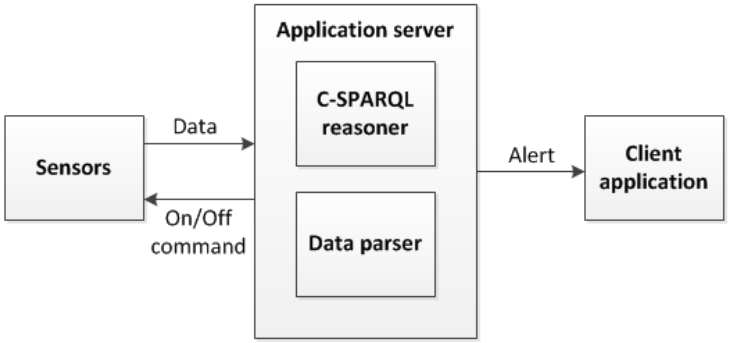
- Time Annotated RDF(TA-RDF)
  - a semantic content management **middleware** to provide transparent integration among heterogeneous data streams and their metadata
  - TA-RDF provides basic functionality for the representation and querying of time-related RDF data
- Temporal RDF
  - an extension to RDF to capture the notion of time, enabling reasoning over time enriched RDF data. (similar to TA-RDF)
- stRDF
  - a constraint data model that extends RDF with the ability to represent spatial and temporal data
  - An extension of SPARQL is developed for querying stRDF data

Analysis of Stream Reasoners for IoT

Table 1: Analysis of Stream Reasoners for IoT (Note: ST stands for Single timestamp, SO for Squencing operation, QA for Query answering, and M for Materialization)

Reasoners	C-SPARQL	IMaRS	TrOWL	Ded. Ind. Stream Reasoning	C-SPARQL on S4	CQELS	Streaming-SPARQL	Streaming Knowledge Bases	Sparkwave	EP-SPARQL	STARQL	TA-RDF	Temporal RDF	stRDF
Heterogeneous data	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Efficiency	Low	Low	Low	Low	High	Low	Low	Low	High	Low	Low	Low	Low	Low
Scalability	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Stream combination	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Background knowledge	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Reasoning Capability	RDFS	Transitive Property	OWL2 DL/EL + Approximation	✗	RDFS Subset	✗	✗	RDFS/ OWL2 Subset	RDFS Subset	RDFS	First Order Logic	✗	✗	✗
Continuous query	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Time model	ST	ST	ST	ST	ST	ST	ST	ST	ST	Interval	SO	ST	ST	ST
Typical Reasoning Tasks	QA	M	M	M	QA	QA	QA	QA	M	QA	QA	QA	QA	QA
Open source	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗

C-SPARQL Experiments




```
REGISTER QUERY WasteEnergyLight AS
   PREFIX ee1: <http://myexample1.org/>
   PREFIX ee2: <http://myexample2.org/>
   SELECT ?sensor1 ?sensor2
   FROM STREAM <http://ee.oulu.fi/V1
         #/lightSensorStream> [RANGE 120s STEP 10s]
   FROM STREAM <http://ee.oulu.fi/V1
         #/pirSensorStream> [RANGE 120s STEP 10s]
   WHERE {
   ?sensor1 ee1:status ?st1 .
   ?sensor2 ee2:status ?st2 .
   FILTER NOT EXISTS { ?sensor1 ee1:status
         ee1:off }
   FILTER NOT EXISTS { ?sensor2 ee2:status
         ee2:moving }
   };
```

This system owns sensing capabilities based on events and delivers alerts to client applications on mobile devices. The behavior of IoT devices could also be changed according to the reasoning results.

First Rule: When light is on and PIR sensor detects no movement, we deduce wasting energy

Second Rule:When the door is closed and no Wi-Fi or no movement for two minutes, we turn on the Wi-Fi sensor, PIR sensor and light sensor.

```
REGISTER QUERY WasteEnergyRoom AS
   PREFIX ee2: <http://myexample2.org/>
   PREFIX ee3: <http://myexample3.org/>
   PREFIX ee4: <http://myexample4.org/>
   SELECT ?sensor3 ?sensor2 ?sensor4
   FROM STREAM <http://ee.oulu.fi/V1
         #/pirSensorStream> [RANGE 120s STEP 10s]
   FROM STREAM <http://ee.oulu.fi/V1
         #/tiltSensorStream> [RANGE 120s STEP 10s]
   FROM STREAM <http://ee.oulu.fi/V1
         #/wifiSensorStream> [RANGE 120s STEP 10s]
   WHERE {
   ?sensor2 ee2:status ?st2 .
   ?sensor3 ee3:status ?st3 .
   ?sensor4 ee4:status ?st4 .
   FILTER NOT EXISTS { ?sensor2 ee2:status
         ee2:moving }
   FILTER NOT EXISTS { ?sensor3 ee3:status
         ee3:opem }
   FILTER NOT EXISTS { ?sensor4 ee4:status
         ee4:on }
   };
```

A Gap Analysis

Table 2: A Gap Analysis Summary

	Current Status	Expectations	Gaps	Recommendations
Heterogeneous and scalable Data Streams	- Limited stream reasoners support scalable data processing. - Stream reasoners accept limited selection of data model and formats.	- Handling a large amount of heterogeneous, ubiquitous, and dynamic IoT data. - Resource-constrained IoT nodes need to be considered.	- Support of resource constrained IoT nodes. - Ease of data integration with different models.	- Developing data models. - Applying parallel and distributed reasoning approaches.
Reasoning	- Focusing on retrieval and search. - Trading expressive power for efficiency.	- Support application scenarios with user defined rules. - Lightweight reasoning mechanisms.	- Balance between expressive power and efficiency	- Incremental algorithms. - Limiting materialized knowledge. - Reasoning with user defined rules.
Managing Uncertainty	- limited support.	- Processing imprecise, incomplete, inconsistent, and incorrect IoT data.	- Suitable models are required, considering different levels of uncertainty.	- Introduce uncertainty modelling approaches.
Time model	- Simple time models.	- Flexible time models for diverse IoT applications	- Expressive time models for representing various timing and ordering relations.	- Developing a simple, but flexible time model.
Reasoning Tasks	- Support query answering and materialization	- Perform RDFS and OWL 2 reasoning to compute full materialization of dynamic IoT data.	- Flexible software architecture for different IoT applications.	- Stream Reasoning oriented software architecture
Benchmarking	- Existing stream systems are evaluated on small-scale scenarios.	- Stream systems should process heterogeneous large volumes of IoT data.	- Large scale evaluations.	- Development of benchmarks for concrete measurements of existing solutions.

Recommendations

1. Developing proper **data models** and applying **distributed reasoning approaches** for handling heterogeneous and scalable data streams
2. Applying **incremental reasoning** algorithms, limiting materialized knowledge, and reasoning with **user defined rules** for IoT applications
3. Introducing **uncertainty modelling** approaches, such as probability theory, fuzzy logic, etc.
4. Developing **flexible time models**, which should be simple and expressive, not introducing too much computation
5. Developing stream reasoning oriented software **architecture**.
6. Developing **benchmarks** for concrete measurements of existing solutions

Conclusions

- This paper:
  - examined key challenges that stream reasoners should address, studied off-the-shelf stream reasoners
  - Highlighted their strengths and limitations
  - examined their capabilities to meet the specific challenges of IoT
  - presented a preliminary experiment
  - performed a gap analysis to evaluate stream reasoners
  - proposed recommendations and suggested future research for development of stream reasoners in IoT

Conclusions

- Future work:
  - study new approaches to deal with streaming data expressed in a format compatible with Semantic Web languages
  - identify all pieces of IoT data and knowledge
  - define a suitable model for time
  - take into account incremental techniques operating with data that is influenced by changes

Thanks!

Comments and Questions?