Reasoning in Dynamical Systems on the Web: Classification and Challenges

Andreas Harth
Karlsruhe Institute of Technology (KIT)
Stream Reasoning Workshop
From Stream Processing...

PREFIX : <http://example.org/>
SELECT * WHERE {
  :Axel :isIn ?x .
  :Andreas :isIn ?x .
}
... to Stream Reasoning
Selected Projects

• ARVIDA: Applied Reference Architecture for Virtual Services and Applications
  • Break up monolithic industrial applications into distributed applications
  • Components based on RESTful interfaces
  • RDF, RDFS and OWL for knowledge representation (vocab.arvida.de)

• i-VISION: Immersive Semantics-based Virtual Environments for the Design and Validation of Human-centred Aircraft Cockpits
  • Human-Cockpit Operations Analysis
  • Semantic Virtual Cockpit
  • Virtual Cockpit Design Environment
Demo Setup
Requirements

• Connect different subsystems (tracking, simulation, rendering...)
• Process data in near-realtime (sub-second) for VR/AR
• Handle dynamics and behaviour (of both systems and humans)
Mismatches

• Data: different data formats (binary, XML, JSON, CSV...)
• Metadata: different annotations (data source, timestamp, valid time...)
• Protocol: different access protocols (sockets, HTTP, ROS, MQTT, UPC-UA...)
• Architecture: different access method (pull vs. push)
• Semantics: different vocabularies (ASDM, DC...)
• Aspect: different view on aspect (state vs. event)

• Goal: uniform interfaces to simplify integration
Technologies

- Web technologies for identification (URIs) and access and manipulation protocol (HTTP)
- RESTful systems (Richardson Maturity Model level 2/3)
- Semantic Web technologies for data modelling (RDF, RDFS, subset of OWL for data integration; SHACL for checking constraints)
- Wrappers ("administration shell") to unify access to components
Resolving Mismatches

• Data: everything in RDF (except binary files)
• Metadata: source (HTTP URI/request with HTTP header info)
• Protocol: everything in HTTP
• Architecture: everything pull (user input: push)
• Semantics: reasoning to handle different vocabularies
• Aspect: everything is modelled in state view (user input: event)
Uniform Network Interface

• Components A and B, data flows from A (source/producer) to B (sink, consumer)

  ![Diagram of components A and B with a line connecting them]

• REST assumes request/response communication pattern between components with client connector and server connector

  • Clients emit requests, receive response
  • Servers answer to incoming requests with a response
Network Interface: Push vs. Pull

**Push**
- A is client, B is server
- A emits PUT request
- At A: B.put(value)
- Loop at A

**Pull**
- A is server, B is client
- B emits GET request
- At B: A.get(value)
- Loop at B
Linked Data-Fu

• Linked Data processor
• Query and rule evaluation
• Operates on snapshots of current world state (as described in Linked Data)
• Repeated evaluation of rule-based programs
• Support HTTP CRUD (create-read-update-delete)

• http://linked-data-fu.github.io/
Conclusion

• Linked Data provides uniform interface to diverse set of data sources
• Web provides decentralised open platform (that can be used internally, too)
• Vocabularies and Ontologies allow for modelling data
• SPARQL allows for querying data

• With the right technology, we can process real-time heterogeneous data
• Both at rest (batch) and at motion (real-time), read and write

• Existing industrial prototypes for applications in Industrie 4.0 and Internet of Things
References


