Indice

Capitolo 1

Problem Settings

1.1 Overview

Stream Reasoning research filed wonders to achieve a tight integration of known reasoning systems and DSMS, to exploit reasoning techniques upon rapid changing information. An RDF Stream Processing Engine, shortly RSP Engine, is the system that realizes this integration: it can manage rapidly changing worlds at the semantic level and answer complex queries typical of Semantic Web. The Stream Reasoning community is working on the standardization of a protocol to talk with RSP Engines, which actually can be implemented applying a number of RSP techniques. This number is increasing together with the need of shared practices and tools to perform analyses and evaluations. As stated in Chapter 2, benchmarking RSP Engines is possible thanks to the proposed RDF streams, continuous queries, and performance measurements. the critics on these works have showed their limits and further developments partially went beyond. However, the the community still lacks the formalization of a comparative approach and an infrastructure that allows to apply it in rigorously. The comparative research is case-oriented. It allows the systematic analysis of complex cases, exploiting comparative methods. The complex cases are seen as configurations, a combination of known properties upon which is possible to read parallelism or contrasts.

In this thesis we propose $\mathcal{H}$eaven, a framework that tries to solve the Stream Reasoning need of a Systematic Comparative Approach on RSP Engine evaluation. A Systematic Comparative Approach requires firstly the definition of *Metrics that allow comparison* and standardization of *Evaluation conditions*. Because of this we include in $\mathcal{H}$eaven the possibility to define custom metrics collection and the concept of experiment as execution setting. The next step consists in providing *Tools for qualitative analysis* and *Simple Terms of comparison*, which have an experimental validity. The Analyser and the four Baselines fulfil this requests. Last but not least, the comparative method needs *Examples of successful analysis and evaluations*, which can be exploited as guidelines. The Chapter 5 of

this thesis present a set of experimental evaluation and show deeply the potential of $\mathcal{H}$eaven using the Baselines as subject of the analysis.

1.2 Requirements

1.3 Requirements

As stated in Section 1, $\mathcal{H}$eaven has the purpose to support comparative research on RSP engines. To this extent we need to answer the following questions:

Q.1 How can the behaviour of system be evaluated?

Q.2 What makes this evaluation rigorous?

Q.3 How can this rigorous evaluation be automated?

We answer Question Q.1 using the definition of *experiment*: a test under controlled conditions that is made to demonstrate a known truth or examine the validity of a hypothesis. Moreover, we answer Q.2 bringing about the notions of *reproducibility*, *repeatability*, and *comparability* of experiments. At this point we are able to answer Q.3 by eliciting the requirements for $\mathcal{H}$eaven .

To support reproducibility, $\mathcal{H}$eaven must allow its users to define an experiment. More specifically:

R.1 it must be *test data independent*, thus allowing users to chose relevant RDF data streams and ontologies from their domain of interest.

R.2 it must be *query independent*, thus allowing users to register relevant queries from their domains of interest.

R.3 it must be *engine independent*, thus allowing users to put an RSP engine on the test stand by the means of easy to implement software interfaces, e.g., it should adopt an event-base architecture as normally done by RSP engines and present events to RSP engine in a simple to parse RDF serialisation.

R.4 it must include a *basic set of performance measurements* [**?**] including **Latency** –defined as the delay between the injection of an event in the RSP engine and its response to it –, **Memory Usage** – defined as the difference between total system memory and free memory –, and **Completeness & Soundness** of query-answering results.

R.5 it should enable users to extend the test stand adding their own software sensors in order to other performance measurements

In terms of software engineering the list of requirements above demands an *Extendable Design* [R6], i.e., the possibility to replace theoretically each module with one with the same interfaces, but different behaviour, without affecting architecture stability.

To allow repeatability $\mathcal{H}$eaven must not affect the RSP engine evaluation. This from a practical point of view poses two requirements to the test stand:

R.7 it must not be running when the RSP engine is under execution.

R.8 it must have reduced (and possibly constant) memory footprint.

Last but not least, to enable comparability of experiments, $\mathcal{H}$eaven must support the collection of the performance measurements [R.9] and it should provide tools to analyse and visualise them [R.10].

Elenco delle figure

Elenco delle tabelle