Capitolo 1

Problem Settings

In this chapter we present firstly the question that leads our research and the motivations which sustain it. Next we extend in Section x the requirements posed by the research question and the issues it involves. In the end we provide the technical formalization of those requirements and why our work must cover them, which issue they solves and which limitation we have to face to.

1.1 Overview

Stream Reasoning research filed wonders to achieve a tight integration of known reasoning systems and DSMS. An RDF Stream Processing Engine, shortly RSP Engine, is the abstraction that realizes this integration and can exploit reasoning techniques upon rapid changing information [**?**, **?**, **?**]. Consequently it is able to manage rapidly changing worlds at the semantic level and answer typical Semantic Web complex queries. 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. Their 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 we noted that the community still lacks the formalization of a comparative approach for the research on RSP Engines and not least an infrastructure that allows the its rigorous application.

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.

1.2 Why Comparative?

The comparative research approach is typical of the Social Science, where the complexity of the subject go beyond the possible observable models. RSPEngine are complex system too, as can be seenin Chapter 6, is difficult to analyse them even in case of simple, well defined architectures. This emphasises the importance to be able to conduct comparative research based on controlled experimental conditions and, thus, the need for a open source16 framework like Heaven.

1.3 Requirements

We developed $\mathcal{H}$eaven in order to simplify and support Systematic Comparative Research Approach 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?

A proper answer to Question Q.1 can be stated exploiting the traditional definition of *experiment*: a test under controlled conditions that is made to demonstrate a known truth or examine the validity of an hypothesis. Going deeply, we answer Q.2 bringing about the notions of *reproducibility*, *repeatability*, and *comparability* of experiments. The concepts we identified make easy to answer Q.3 formalising the technical requirements for $\mathcal{H}$eaven.

Reproducibility is related to the variation in measurements made on a subject under changing conditions. The concept of experiment gather this conditions. $\mathcal{H}$eaven must allow its users to define it in details:

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.

Repeatability of measurements regards the variation in repeat measurements made on the same subject under identical conditions. $\mathcal{H}$eaven must not affect the RSP engine evaluation to grant it. 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.

The comparative research is case-oriented. It allows the systematic analysis of complex cases, exploiting comparable metrics. The complex cases are seen as configurations, a combination of known properties, upon which is possible to identify parallelism or state contrasts. A Systematic Comparative Research Approach (SCRA) requires both the definition of *Metrics that allow comparison* and the standardization of *Evaluation conditions*. $\mathcal{H}$eaven must support the collection of the performance measurements as custom metrics [R.9]; again the concept of experiment is required as a formalization for the execution setting. The next step consists in providing *Tools for qualitative analysis*, which allow visualisation of the results [R.10].

Moreover *Initial terms of comparison*, a.k.a. baselines, are needed to guide the SCRA. A baseline is an elementary solution for the research problem, which maintains experimental validity and consequently is relevant as a term of comparison. Baselines can be exploited as a simple case-studies to support the need of *Successful analysis and evaluations examples*, which can be seen as guidelines.

$\mathcal{H}$eaven contains specific modules to fulfil this request, which of course presents their own requirements. As follow, a Baseline must be..

R.11 a solution: it solves the problem entirely,without concerning about performance

R.12 elementary: it does not represent a innovative solution w.r.t. the state of the art. Baselines requires the minimum effort in terms of design

R.13 valid: the performance measurements must be comparable with smarter solution and their comparison must have sense in a realistic context

R.14 relevant: it must cover one of the most relevant variations in theoretical solution, in agreement with the case-oriented nature of CSRA

R.15 simple: the solution design it implements must be easy to understand. In this way is possible to formulate hypothesis from a theoretical viewpoint. Obviously there are no guarantees about the experimental verification

Some of those requirements assume a problem-specific meaning. As advocated in the early works on Stream Reasoning [**?**, **?**] the most simple approach to create a stream reasoning system is arrange into a pipeline DSMS with a reasoner. Baselines design must follow this statement to develop a full solution [R.11] which is very simple w.r.t the commercial ones presented in Chapter 2 [R.12]. The background section related to this states how two main design decisions can distinguish the baselines: the RDF stream model and the architecture. From this point of view $\mathcal{H}$eaven must provide at least 4 Baselines [R.14], which cover all possible combinations of the two alternative RDF stream models and the possible architectures variants, which are again two.