Chapter 1

Problem Settings

In this chapter we present firstly the question that leads our research and the motivations which sustain it. We extend the issues our work involves and the methods we follow to face them. In the end we state the requirements we must satisfy together with their the technical formalization.

1.1 Overview

Stream Reasoning research field 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. It can exploit reasoning techniques upon rapid changing information [**?**, **?**, **?**] managing the information stream at semantic level. Consequently an RSP Engine can answer the complex queries typical of Semantic Web. The Stream Reasoning community is working on the standardization of a protocol to talk with RSP Engines, because the number of different RSP techniques they implement 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, but the critics on these works have showed their limitations and further developments goes only partially beyond [**?**]. Moreover we noted that the community still lacks the formalization of a comparative approach for the research on RSP Engines and also an infrastructure that allows the its rigorous application.

In this thesis we propose $\mathcal{H}$eaven, an open source framework that tries

to solve the Stream Reasoning need of a Systematic Comparative Research Approach for 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. Single case studies allow a deep understanding of that case, but is difficult to engage any form of generalisation. Cross-case studies is necessary and their multi-faceted nature represents an first research problem. We need a strategy which allow comparison without loose the case complexity. A solution is considering complex cases as configurations, a combination of known conditions, upon which is possible to identify parallelism or state contrasts. RSP Engines are complex system too and may be difficult to analyse them even in case of simple and well defined architectures (Chapter 6 show this in details). This emphasises the importance to be able to conduct comparative research based on controlled experimental conditions and, thus, the need for a open source framework like $\mathcal{H}$eaven .

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 as a configuration. $\mathcal{H}$eaven must be parametric from its configuration, whose specification is up to the user and poses the following requirements:

R.1 *test data independence*, relevant RDF data streams and ontologies can be choose from user domain of interest.

R.2 *query independence*, relevant queries can be registered from user domain of interest.

R.3 *engine independence*, user must be able to put an RSP engine on the test by the means of easy to implement software interfaces.

Repeatability regards the variation in repeated measurements made on the same subject under identical conditions. The last statement, maintain identical experimental conditions, is the main challenge. $\mathcal{H}$eaven must not affect the RSP engine evaluation to win it. This, from a practical point of view, poses two requirements to the test stand:

R.4 it must not be running when the RSP engine is under execution.

R.5 it must have reduced (and possibly constant) memory footprint.

Comparability is related to the nature of the performance measurements and the relation between some experimental conditions. A Systematic Comparative Research Approach (SCRA) is case-oriented and it requires both the definition of *Metrics that allow comparison* and the standardization of *Evaluation conditions*. $\mathcal{H}$eaven must satisfies three requirements about this:

R.6 include *basic set of performance measurements* [**?**].

R.7 enable users to extend the test stand adding their own software sensors, in order to collect custom metrics results.

R.8 support the collection of the performance measurements, to allow further analysis.

Previous works about Stream reasoning benchmarking shows that the minimal performance measure set includes **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. The next step consists in providing *Tools for qualitative analysis*, which allow visualisation of the results [R.9], pivoting on different experiment which can be seen as a formalization for the execution setting.

In terms of software engineering the list of requirements above demands:

R.10 *Extendible Design*, i.e., the possibility to replace theoretically each module with one with the same interfaces, but different behaviour, without affecting architecture stability.

R.11 *Event-base architecture* to properly communicate with RSP Engines, which normally exploit it.

R.12 *Easy-to-Parse RDF Serialisation* for the event presented to the RSP Engine in exam by the test stand.

SCRA states the need of *Successful analysis and evaluations examples*, which can be followed as guideline, and *Initial terms of comparison*, a.k.a. baselines. We call baseline 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 start our research.

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

R.13 a Solution: it solves the problem entirely, without concerning about performance.

R.14 Elementary: it requires the minimum effort in terms of design. A baseline does not represent a innovative solution w.r.t. the state of the art.

R.15 Valid: the performance measurements must be comparable with smarter solution and their comparison must have sense in a realistic context.

R.16 Relevant: it must cover one of the most relevant variations in theoretical solution, in agreement with the case-oriented nature of CSRA.

R.17 Simple: the design of the solution 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.13] which is very simple w.r.t the commercial ones presented in Chapter 2 [R.14]. Chapter 2 also 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 four Baselines [R.16], which cover all possible combinations those design choices.