Chapter 1

Problem Settings

In this chapter we state the questions that leads our research and the issues it poses. We present a possible solution and the requirements it must satisfy.

1.1 Comparative Research

The Computer Science (CS) community classifies its activity in many works and shows that its research mostly follow an engineering epistemology [**?**, **?**]. A majority of publications belongs to *empirical work* and *design & modelling* classes of Tichy's taxonomy (See **??**) and, among them, proposals of new systems or models are more common then the evaluations of existing ones. This contrasts with other engineering areas, where the experimental research is almost dominant. CS needs to focus on collecting, analysing, and interpreting the observations of those works it usually only designs or implements. Now we demand, *What are the motivation under this CS research lack?* The main problem in evaluating software systems or models regards their complex and multifaceted nature. Other explanations concern the difficulties of conducting realistic evaluations, because of the number of the involved variables.

Social Sciences typically use a Systematic Comparative Research Approach (SCRA), because the complexity of its research subjects goes beyond the possible observable models. For this reason, they exploit techniques to deal with complex cases that can not be simplified in experimental setting. The analysis of a single case study allows to deeply understand it, but it makes difficult to engage any form of generalisation. On the other hand, cross-case studies are more relevant and allow general thinking, but their final complexity represents

a problem. We need a strategy that reduces the analysis complexity without lose the relevance of each involved system. In this regard, Russel Schutt discusses four stages to systematic qualitative comparative studies for history social phenomena:

S.1 Premise of the investigation: identification of possible causes.

S.2 Choose the cases to examine (location, language, gender).

S.3 Examine the similarities and the differences with shared methods.

S.4 Propose a causal explanation for the phenomena.

We can define an *experiment* as a test under controlled conditions that is made to demonstrate a known truth or examine the validity of an hypothesis (W.R Inge). Complex cases are seen as a combination of known properties upon which is possible to identify parallelism or state contrasts and are used to set up experiment configuration. Thanks to experiments, we can exploit the notions of *reproducibility*, appreciating variations on changing experiment conditions, *repeatability*, consolidating observations trough multiple identical executions, and *comparability*, contrasting the results to identify the differences.

Case-driven analysis and the notion of experiment are not totally new concepts in Computer Science. Database community firstly explores the idea of comparative research trough benchmarking techniques and actually the quality of empirical studies is rising. It is worth to note Jim Gray's work about transactional benchmarking (TCP, Section 2) and Domain Specific Benchmarks (DSB). He states that *any comparison on performances starts with the definition of a benchmark or a workload*, but we need know the relevance of the metrics. Measurement variations are very frequent from one application to another, because each system is thought to solve a small problems set. A DSB must response properly to system diversities, by specifying a synthetic workload to describe typical applications in the problem domain; moreover it must provide workload performances on various systems and an estimation of relative performance on the problem domain. Gray proposes also four criteria that a DSB must meet, which are:

G.1 Relevance, it must measure the performance peak of systems when performing domain typical operations.

G.2 Portability, it must be easy-to-implement on many different systems and architectures

G.3 Scalability, it must be meaningful for both small and large computer systems

G.4 Simplicity, it must be understandable to obtain credibility

We note a relation between Schutt work in social science and Gray's criteria: Gray, in G.1, identifies the relevant metrics for the evaluation and Schutt, in S.1, demands a pre-analysis phase of the phenomenon. Social Science does not care about problem scaling as DB, because those properties that define the case also determine the problem dimension (S.2). In DB context G.2 demands implementation-related conditions and G.3 requires to consider the dimension-related issues. Last but not least, G.4 demands simplicity to obtain credibility while S.3 suggests to exploit those evaluation methods that are commonly accepted (shared) by the research community.

The growing number of RDF Stream Processing techniques alarms the Stream Reasoning (SR) community, which identifies the need of evaluation practices to allow comparison. The systems to study, RSP Engines (See **??**), has an high resulting complexity: the execution mechanism they implement summed to their internal semantic and the environment where this execution happens are evaluation-related characteristics, which increase this complexity and demonstrate that a meaningful comparison between RSP Engines is non-trivial. Chapter 6 shows in details that may be difficult to analyse those system, even in case of simple and well defined architectures. The interest on cros-case analysis between complex subjects draw the need of a comparative research approach. Initial efforts in this direction try to define frameworks that resume DSMSs [**??**] and Reasoning [**??**] benchmarking studies. LS Bench and SR Bench propose a set of queries, data sets, and methods to test and compare existing RSP engines [**??**]. Both these works share a common background: the Linear Road Benchmark (LRB) is the only existing benchmark for relational data stream processing engines. Actually it only states which requirements a solution must satisfy, without proposing a concrete one. LS Bench and SR Bench implement and extend this work, but they do not re-contextualise LRB requirements in SR context. Following discussions identify

other lacks, i.e none of them completely face the problem of evaluating query result correctness, because they do not analyse engine semantics. LS Bench concentrates attention to the evaluation of engines throughput and it checks the correctness of the query result measuring the mismatch after comparing different RSP Engines. SR Bench entirely ignores this issue and analyses the coverage of SPARQL constructs for each commercial engine. More recent works describe deeply all the RSP Engine properties, identify the future challenges and provide a standardization benchmarking requirements: commandments for a meaningful testing on RSP Engines[**?**].

In the end, LRB provides also a simulator to validate the benchmarked DSMS systems. Stream Reasoning community still lacks an infrastructure that can control the execution environment and allow to systematic testing. Previously the community questioned itself *How to support SRCA on RSP Engines*? Now we have queries, dataset and methods, that partially answer it and the new research question is "*Does SR research needs a validator to enable the SCRA?*". From the aerospace engineering we borrow the idea of an *Engine Test Stand*. This solution allows experiments design, their systematic execution and automatic results comparison, three ingredients to properly answer the research question.

1.2 Requirements

In order to simplify and support Systematic Comparative Research Approach on RSP engines trough an Engine Test Stand 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?

To answer Question Q.1 we exploit traditional definition of *experiment* presented above. We answer Q.2 referring to *reproducibility*, *repeatability*, and *comparability* of experiments. Trough this concepts it is easy to answer Q.3 formalising the requirements for the solution.

Reproducibility refers to measurement variations on a subject under changing conditions. We gather this conditions into experiment configuration, whose specification is up to the user. For this reason the solution must be independent from:

R.1 *Test data*, relevant RDF data streams and ontologies chosen from user domain of interest.

R.2 *Query*, relevant queries registered from user domain of interest.

R.3 *Engine*, any RSP Engine tested by the means of easy-to-implement software interfaces.

Repeatability refers to variations on repeated measurements on a subject under identical conditions. The solution must not affect the RSP engine evaluation, which, from a practical point of view, poses two requirements:

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 refers to performance measurements nature and the relations between experimental conditions. The SCRA demands both the definition of *comparable metrics* and the standardization of *evaluation methods*, which means the solution must:

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

R.7 enable users extensions with new software sensors and specific measurements collection.

R.8 support performance measurements collection for further analysis.

R.9 allow *qualitative analysis* trough tools for result visualization

Previous works about Stream reasoning [] 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 Load** – defined as the difference between total system memory and the free one –, and **Completeness & Soundness** of query-answering results.

In terms of software engineering, any solution which satisfies the requirements above demands also some technical ones:

R.10 *Extendible Design*, 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 events presented to the RSP Engine in exam

SCRA is case-oriented, it needs *successful analysis and evaluations examples* to pose experimental guidelines and *initial terms of comparison*. We call baseline an elementary solution for the RSP problem, which is relevant from an experimental viewpoint. To fulfil this request the answer to our research question must consider to have specific baseline modules and satisfy their own requirements, which are:

R.13 It must be Elementary: requiring the minimum design effort, which means be a naive solution and do not care about performances.

R.14 It must be Eligible: being a fair term of comparison w.r.t. commercial solutions.

R.15 It must be Relevant: covering one of the theoretical solutions.

R.16 It must be Simple: allowing to identify easily those characteristics which support hypothesis formulation and comparison.