UNIVERSITETI I EJL
ЈИЕ УНИВЕРЗИТЕТ
SEE UNIVERSITY

FAKULTETI SHKENCAVE DHE TEKNOLOGJIVE BASHKËKOHORE
ФАКУЛТЕТ ЗА СОВРЕМЕНИ НАУКИ И ТЕХНОЛОГИИ
FACULTY OF CONTEMPORARY SCIENCES AND TECHNOLOGIES

PHD STUDIES

THESIS:

RULE-BASED REASONING OVER STREAM DATA ON SEMANTIC WEB

CANDIDATE:

MSc. Edmond Jajaga

MENTOR:

Prof. Dr. Lule Ahmedi

Tetovë, 2017

Abstract

Semantic technologies have been extensively used for integrating stream data applications. However, using SWRL, which has become the de facto standard rule language in Semantic Web, has never been used in stream data applications. Its open world assumption and monotonic nature makes SWRL powerless for doing continuous inference over stream data. For example, using aggregate functions on a particular window of streams cannot be expressed in SWRL.

Semantic Web standard query language, SPARQL, has been extensively used in stream data applications. A number of its extensions have been developed to enable powerful stream processing capabilities including data filtering and aggregation functions. One of them, C-SPARQL, is a framework which supports continuous querying over data streams combined with "static" knowledge bases. However, stream processing systems cannot modify the knowledge base.

State of the art stream reasoning systems have achieved the desired expressivity and scalability level. However, as hybrid approach they suffer from translation, reasoner and side-effects issues.

The purpose of this thesis, therefore, is to provide a unified Semantic Web stream reasoning framework that further supports continuous inference over stream data. It was developed C-SWRL, a system that uses SWRL rules in conjunction with C-SPARQL filtering and aggregation of RDF streams to enable closed-world and time-aware reasoning over stream data. Moreover, the non-monotonic behavior is supported with the use of OWLAPI constructs. In particular, it is shown how negation as failure (NAF) can be enabled in this system. C-SWRL is presented by means of examples in water quality monitoring.

Moreover, the contribution of this thesis also includes the development of an ontology for water quality management called INWS ontology. Namely, it is an SSN-based ontology to support water quality classification based on different regulation authorities such as Water Framework Directive. Furthermore, to demonstrate its usage, StreamJess was developed, which is an expert system which uses INWS ontology for water quality monitoring and investigation of potential sources of pollution.

Table of Contents

List of Figures

List of Tables

Chapter I Introduction

Sensor measurements, social networks, health monitoring, smart cities and other massive data sources are continuously producing massive amount of data called stream data. Stream data are defined as unbounded sequences of time-varying data elements [99]. Reasoning with these kinds of data with Semantic Web techniques has eventually contributed in a new research area called Stream Reasoning (SR). The aim to derive high level knowledge from low level data streams is one of the challenging requirements which cannot be easily achieved with the classic solutions for data stream and complex event processing and with reasoning engines for static data [40]. The World Wide Web Consortium (W3C) RDF Stream Processing Community Group[1] has set their mission to define common model for producing, transmitting and continuously querying RDF Streams. However, even though different works exist (e. g. ETALIS [14], StreamRule [34] etc.), rule-based reasoning over RDF streams still remains vastly unexplored.

In this thesis is proposed a unified Semantic Web approach for rule-based reasoning over stream data, thus complementing state of the art query processing engine e.g. C-SPARQL [93] with the W3C recommended Semantic Web rule language SWRL.

Semantic technologies have proved evidence of efficient implementations on stream data domains [3]. Firstly, OWL ontologies have been widely used for modeling stream data domains, e.g., the SSN ontology [4]. Secondly, querying these knowledge bases has been merely done by SPARQL extensions e.g. C-SPARQL [93], EP-SPARQL [15], etc. However, the windows opened over streams can determine changes in the static information sources. Managing the knowledge bases and reasoning with background and streaming data is merely done by rule systems. Although layering different rule systems over ontologies has already been suggested, using Semantic Web recommended rule languages, SWRL [150] and RIF

[1] Cf. https://www.w3.org/community/rsp/.

[171], over stream data has to the best of our knowledge not been considered to date. Thus, as described in our works [3, 148, 5], there is an inherent need for a Semantic Web unified rule system capable of reasoning with stream data.

In line with this vision was developed the INWATERSENSE (hereinafter referred to as INWS) ontology [5] and an expert system [100] demonstrating its usage. Moreover, StreamJess [6] was developed to enable stream reasoning with production rules. More importantly, Continuous SWRL (or simply C-SWRL) [cswrl] was developed to represent a SWRL system for reasoning with stream data. It utilizes C-SPARQL definition of RDF streams and windows that further supports non-monotonic and time-aware reasoning on stream data.

Publications from this thesis. The initial works on this thesis were published in [3] (cf. Section 2.3), describing Semantic Web trends on reasoning over stream data including latest developments on ontology, query and rule layer. The INWS ontology model is described in [5] (cf. Chapter 3). A Jess expert system demonstrating the INWS ontology usage is described in [100] (cf. Section 3.3), whereas its stream data version appeared in [6] (cf. Chapter 4). The main contribution of this thesis is presented in [cswrl] (cf. Chapter 5).

Outline. The material provided in this thesis is organized as follows: Chapter 2 describe current trends on representing stream data with semantic technologies and their pros and cons. The next subsection states the hypothesis and presents the motivation that inspired the works on this thesis.

Chapter 3 describes the INWS ontology, an ontology framework for modeling the domain of the water quality monitoring (WQM) systems, which is used as a case study throughout this thesis. INWS ontology consists of three ontology modules: core, regulations and pollutants. All these modules are integrated into a single ontology to serve as a single access point for the rules to enable WQM and investigation of potential sources of pollution. The model was validated with Jess rules described in Section 3.4.1.

An expert system, StreamJess, capable of expressive reasoning over stream data is elaborated in Chapter 4. It layers Jess on top of C-SPARQL to enable time-aware, closed-world and non-monotonic reasoning on stream data domains. Namely, with a couple of examples in the WQM domain it demonstrates the usage of the INWS ontology coupled with production rules i.e. Jess rules.

Chapter 5 unfolds the main contribution of the work providing an explanation of the overall architecture of the C-SWRL. It describes how SWRL can be enabled to run efficiently on

stream data domains. Moreover, it explains how the required reasoning features missing in SWRL are fulfilled by a stream processing system such as C-SPARQL.

The relation of the INWS ontology and C-SWRL compared to their counterparts is explained in Chapter 6, while the challenges behind building them and related discussion take part on Chapter 7.

Finally, Chapter 8 outlines the perspectives and concludes the thesis.

Chapter II Preliminaries

This chapter provides a general summary of the Semantic Web and Stream Reasoning. The definitions and formalisms of both paradigms are given in short borrowed and cited from the related literature. Moreover, the Semantic Web trends on reasoning over stream data are described and moreover the motivations inspiring the works on this thesis are highlighted. They set the stage for the presentation of the work which commences in the following chapter. Namely, the description of Semantic Web standards for data modeling, querying and reasoning takes place on the next section. Section 2.2 describes the notions of reasoning, stream data, stream reasoning, windows and continuous processing. Literature review on ontology models and rule-based approaches for stream data applications are presented in Section 2.3. The chapter concludes with hypothesis and problem statement of the study of this thesis.

II. 1 Semantic Web

The importance of the World Wide Web (WWW) and its impact on our everyday life is huge. At every single moment people are sharing information worldwide. They share them on social networks, web sites, web-based applications etc. This has led in a situation where there are too much data and less semantics. The inventor of WWWs, Sir Tim Berners-Lee, came with another brilliant idea. He proposed to represent Web content in a form that is more easily machine-processable and to use intelligent techniques to take advantage of these representations. This paradigm opened up the way for new developments on the Web and was formalized as a new discipline called Semantic Web. However, this does not mean to eliminate all current developments on the Web, instead the Semantic Web technologies are built to equip the human-directed information with machine-processable affinities.

The development of the Semantic Web has a lot of industry momentum, and governments are investing heavily. The U.S. government has established the DARPA Agent Markup Language (DAML) Project, and the Semantic Web is among the key action lines of the European Union's Sixth Framework Programme [110].

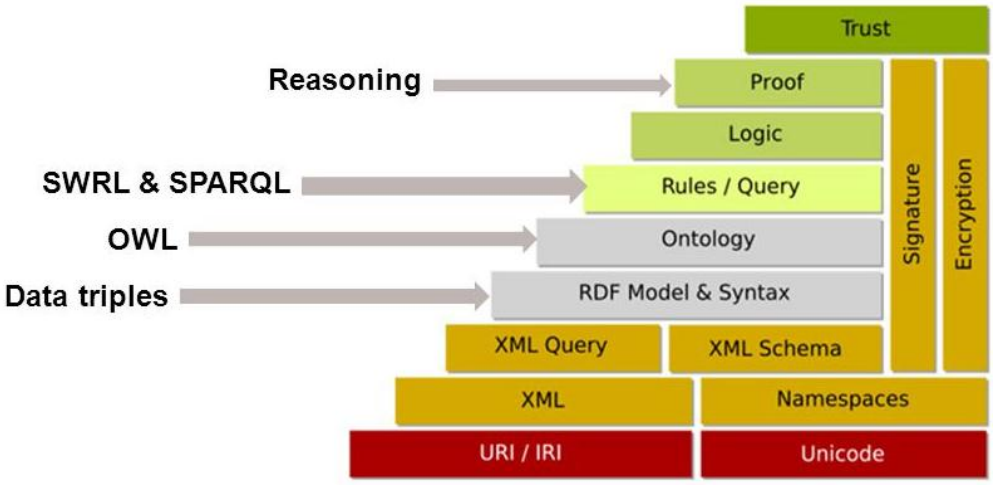


Figure 1. The Semantic Web cake

II. 1. 1 Semantic Web technologies

This section recalls current recommended standards for building Semantic Web applications. For a concise illustration of the relation between the layers of Semantic Web applications will be consulted the popular Semantic Web cake depicted in Figure 1.

Only definitions of the main layers will be presented below necessary for understanding the description of this thesis.

RDF data. The first important thing is the data model. The W3C has set RDF (Resource Description Framework) as a de facto standard for building Semantic Web models. As the name may suggest, it describes the Web objects, called resources, and the relations between them. Uniform Resource Identifiers (URIs) are used to uniquely identify each resource. They can be URLs or any other unique identifiers.

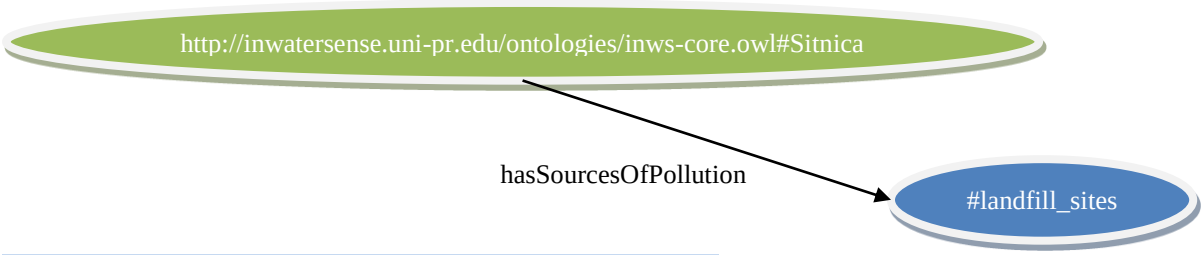
The main idea behind RDF is the constitution of the so-called triples (subject, predicate, object) between resources. For example, the statement "Landfill sites are discharging on the body of the river Sitnica", represented as RDF triple (in three forms of representation) looks as depicted in Figure 2.

RDF Schema. RDF Schema (RDFS) makes semantic information machine accessible, in accordance with the Semantic Web vision. RDFS is an extension of RDF, which allows users to describe resources using its vocabularies. For the readers familiar with SQL databases, RDFS is to RDF, what is SQL schema to SQL data. RDFS defines classes and properties of specific application domains.

TRIPLE

```
(http://inwatersense.uni-pr.edu/ontologies/inws-
core.owl#Sitnica, hasSourcesOfPollution, landfill_sites)
```

SEMANTIC NET



XML-based

```
<body:River rdf:about="http://inwatersense.uni-
pr.edu/ontologies/inws-core.owl#Sitnica">
  <hasSourcesOfPollution rdf:resource="#landfill_sites"/>
</body:River>
```

Figure 2. A RDF triple example from the domain of WQM

OWL ontologies. The expressivity of RDF and RDF Schema that was described previously is deliberately very limited. RDF is (roughly) limited to binary ground predicates, and RDF Schema is (roughly) limited to a subclass hierarchy and a property hierarchy, with domain and range definitions of these properties. However, the Web Ontology Working Group of W3C identified a number of characteristic use cases for the Semantic Web that would require much more expressiveness than RDF and RDF Schema offer. A number of research groups in both the United States and Europe had already identified the need for a more powerful ontology modeling language. This led to a joint initiative to define a richer language, called DAML+OIL (the name is a join of the names of the U.S. proposal DAML-ONT and the European language OIL). DAML+OIL in turn was taken as the starting point for the W3C Web Ontology Working Group in defining OWL, the language that is aimed to be the standardized and broadly accepted ontology language of the Semantic Web [110].

OWL builds upon RDF and RDFS and has the same kind of syntax. For ontological knowledge, one may reason about: class membership, equivalence of classes, consistency and classification. Ontologies are defined as explicit and formal specification of a conceptualization [19]. Typically, an ontology consists of a finite list of terms and the relationships between these terms. The terms denote important concepts (classes of objects) of the domain. For example, in the case of WQM data model, as shown in Figure 3, the class of measurement sites is a sub class of the class of bodies of water; the fact that potential sources of pollution may discharge in a particular measurement site is captured by the relation `hasSourcesOfPollution` linking instances of `MeasurementSites` with `PollutionSources` ones.

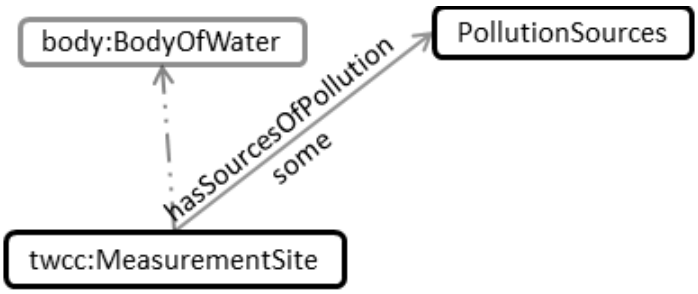


Figure 3. An ontology excerpt

SPARQL queries. The default query language for RDF is SPARQL. It is a W3C standard query language since January 15th, 2008. The main principle behind SPARQL is to match RDF triples based on the values provided for the subject, predicate and object positions. It supports variables instantiation on each of these positions to output the matching graph of triples. The body of the query follows the SQL-like approach in the form of SELECT-FROM-WHERE, where SELECT clause specifies the projection of the retrieved results, FROM specifies the source being queried and WHERE clause is used to apply constraints over the matched triples. An example query to return all the individuals of the `ssn:Observation` class might be like follows:

```
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
SELECT ?x
WHERE {
    ?x rdf:type ssn:Observation .
}
```

The PREFIX command is used to register namespaces for omitting them on the body of the query.

SWRL rules. The recommended rule language of the Semantic Web is Semantic Web Rule Language (SWRL). It is a W3C member submission since May 21st, 2004. As an inference system, SWRL extends OWL axioms with Horn-like rules. The rules are composed in the form of implication between an antecedent (body) and consequent (head). Both rule's body and head consist of zero or more atoms. Atoms on these rules are OWL concepts: `C(x)` and `P(x, y)`, where `C` is an OWL class, `P` is an OWL property and `x, y` are variables, OWL individuals or OWL data values. Moreover, predicates `sameAs(x, y)`, `differentFrom(x, y)` and `builtIn(r, x, ...)` are added to the language to include the semantics of interpreting same and different objects and SWRL built-in libraries, respectively. For example, a SWRL rule to populate the class `PHPollutedSite` with individuals of class `MeasurementSite`, which sources of pollution include pH ones can be defined as follows:

```
MeasurementSite(?x) ∧ hasSourcesOfPollution(?x, ?y) ∧
potentialPollutant(?y, pH) → PHPollutedSite(?x)
```

Namely, this rule binds the measurement sites with variable `?x` and matches them with the sources of pollution `?y` present on each particular measurement site. Furthermore, each potential pollutant `?y` is checked whether it is equal to `pH`, if so the matched `?x` measurement sites will also become individuals of class `PHPollutedSite`.

Another Semantic Web rule language is the Rule Interchange Format (RIF). Since its primer purpose is for exchanging rules between different rule paradigms, it was decided to omit the study of its application on SR domains.

II. 2 Stream Reasoning

Reasoning is defined as the ability to generate non-trivial conclusions from premises or assumptions [111]. The known asserted facts are termed as *explicit knowledge*, while the inferred ones as *implicit knowledge*. The choice of RDF as data model, in combination with ontological languages (e.g., OWL), enables the implementation of algorithms that can "reason" on existing data to infer new knowledge [80]. However, this approach considers data change occurs in low frequency rate. In social networks, smart cities, WSNs etc. data is

super dynamic. On the time of writing this thesis[2] each second are produced 7 426 tweets, 754 photos are uploaded on Instagram, 2 365 Skype calls are made, etc. On the other side, sensors, logging systems, etc. are producing huge amounts. For example, it is estimated that 312 million gigabytes of data are produced by car sensors worldwide[3]. In this situations, the reasoning tool need to process gigantic "on the fly" data, e.g. sensor data, by combining them with data that changes slowly and with background domain-specific knowledge. These new settings have opened up new challenges known as *Stream Reasoning (SR)*. It utilizes Semantic Web techniques for reasoning with stream data [99]. *Stream data* are defined as unbounded sequences of time-varying data elements [99]. In the literature SR is defined as *"logical reasoning in real time on gigantic and inevitably noisy data streams in order to support the decision process of extremely large numbers of concurrent users"* [29].

In fact, SR is a combination of *Data Stream Management Systems (DSMSs)*, which have been developed by the database community and *Complex Event Processing (CEP) systems*, which have been developed by the community working on event based systems [80]. Namely, SR utilizes the DSMS's notion of windows and continuous processing and CEP's representation of events (data streams) with timestamp. SR can further effectively handle background knowledge and perform reasoning.

The following statements define the notions of windows and continuous processing as described in [80]:

Window. Traditional reasoning problems are based on the idea that all the information available should be taken into account when solving the problem. In SR, this principle is eliminated and the reasoning is restricted to a certain window of concern, which consists of a subset of statement recently observed on the stream while previous information is ignored. This is necessary for different reasons. First of all, ignoring older statements allow us to save computing resources in terms of memory and processing time to react to important events in real time. Further, in many real-time applications there is a silent assumption that older information becomes irrelevant at some point.

Continuous Processing. Traditional reasoning approaches are based on the idea that the reasoning process has a well-defined beginning (when a request is posed to the reasoner) and end (when the result is delivered by the system). In SR, one will move from this traditional model to a continuous processing model, where requests in terms of reasoning goals are

[2] Metrics taken from http://www.internetlivestats.com/one-second/, on 30.11.2016
[3] Metrics taken from http://www.ibmbigdatahub.com/blog/big-data-wheels, on 30.11.2016

registered at the reasoner and are continuously evaluated against a knowledge base that is constantly changing.

II. 3 Semantic Web trends on reasoning over stream data

Semantic Web applications are growing day to day. Meanwhile Semantic Web standards are also maturing. Sensor rapid development and deployment in different disciplines including weather forecasting, water quality management, civic planning for traffic management etc. requires efficient machine communication. Many organizations and institutions have taken initiatives to take advantage from the synthesis of both "worlds" to provide semantics on different application domains. In 2008, Kno.e.sis initiated a project for building Semantic Sensor Web assembling sensor metadata from all over the world. The initiative is aligned-well with standardization efforts of W3C and Open Geospatial Consortium (OGC), in particular with Semantic Web and Semantic Web Enablement (SWE) activities, respectively. In fact, Semantic Sensor Web represents a synergy of both initiatives by semantic annotating of simple sensor data i.e. time, spatial and thematic data. In line with Semantic Sensor Web the W3C Semantic Sensor Network Incubator group (the SSN-XG) recently produced an OWL 2 [11] ontology named SSN [4], which enhances OGC SWE simple spatial and temporal concepts with semantic annotation for analyzing and Linked Data publishing. The SSN ontology models sensor data in four main perspectives: sensor, observation, system and feature and property perspectives.

Sensor data are an example of stream data which are rapidly changing data. These huge amounts of data need to be quickly consumed and reasoned over. For example, if a particular water quality parameter drops from its allowed threshold then this information needs to be consumed quickly and an appropriate decision should follow. Sensors continually produce water quality parameter values. Historical and real-time data produced by sensors require a flexible knowledge management system. An area which deals with continues execution of queries over stream data is Data Stream Management Systems (DSMS). As indicated in [22] it lacks the ability to reason about complex tasks and lacks a protocol for wide publication. The Semantic Web fulfills these gaps but caching all the knowledge for rapidly changing information is inappropriate. Similar to DSMS is Complex Event Processing (CEP) which provides on-the-fly analysis of event streams, but cannot perform reasoning tasks [15]. Following the pros and cons of DSMS and CEP a new research area has been investigated by

the community, namely Stream Reasoning [22]. Stream Reasoning integrates data streams, the Semantic Web and reasoning techniques into a unique platform. Unlike in a traditional reasoning environment, where all the information is taken into account, in stream reasoning there are two concepts which indicate the distinguished approach. The window concept restricts the reasoning to a certain subset of statements recently observed on the stream while previous information is ignored, furthermore continuous processing means continuous evaluation of streams against the knowledge base which is constantly changing.

In general, querying RDF triples of stream data has been leveraged with different SPARQL extensions like: Streaming SPARQL [24], Continuous SPARQL (C-SPARQL) [23] and Time-Annotated SPARQL [25].

This thesis is mainly focused on the Semantic Web rule layer. State-of-the-art rule-based systems for dealing with sensor data reasoning are mainly:

- *Hybrid systems* e.g. CEP with Semantic Web in [14] and [15], production rules with Semantic Web in [13] and [27].
- *Pure Semantic Web rule systems* as given in [37], [109] and [21], but which do not deal with the streaming nature of sensor data.

The following subsections describe state-of-the-art ontology, query and rule systems for reasoning over stream data e.g., sensor data.

II. 3. 1 Ontologies and queries

Ontologies are defined as formal specification of a shared conceptualization [19]. Because of its knowledge reuse and sharing, the ontological knowledge model has been widely leveraged for representing WSNs. One of the first WSNs which has benefited from including the ontological model into its knowledge base is OntoWEDDS [9], a decision-support system for wastewater management, which extends its previous version's case-based and rule-based reasoning with WaWO [10] ontology. The evaluation results have yielded an improvement of 70-100% successful diagnosis and no impasse situations including WaWO reasoning, against 60-70% and 10 out of 57 impasse situations without using it.

Interoperability between sensors and sensing systems was enabled with the development of the SSN ontology. Its foundation is based on DOLCE-UltraLight[4] (DUL) ontology. To model

[4] http://www.loa-cnr.it/ontologies/DUL.owl

a knowledge base of sensor networks one would include SSN interested features extending it with units, location, feature and time ontologies [4]. Additional classes and properties can be defined and added to model specific domain knowledge.

There are also initiatives dealing with sensor streaming data on query level. Shahriar et al. (2011) proposes smart query system considering both streaming data and historical data from marine sensor networks. ES3N [18] and C-SPARQL [23] are also dealing with sensor stream data. C-SPARQL is an extension of SPARQL for supporting stream data querying. Query processing is an important issue on the Semantic Sensor Web [26], but it is out of the scope of this thesis. Instead, the focus is on the rule layer reasoning.

II. 3. 2 Rules

As claimed in the previous section almost every sensor network knowledge base is modeled through OWL ontologies. The Semantic Sensor Web foundation has enabled semantic enrichment of simple sensor data through these ontologies. However, inferring new and implicit knowledge from known facts represented in ontological terms is enabled through a powerful mechanism known as rule-based reasoning. In general, the limited expressivity of SWRL [26], which currently has the status of W3C submission, has forced the community to consider hybrid systems while keeping the knowledge base modeled in the form of ontologies. Specifically, for the domain of sensor data an obstacle appears from the continuous flow of data. These data need to be consumed quickly and efficiently infer new knowledge by combining them with the background knowledge. Because of this nature when trying to infer logical consequences from sensor data different rule systems are considered by the community. In the remaining of this section they will be described briefly.

Association Rules Mining

Ding et al. (2011) have proposed a framework for association rule mining and scoping in spatial datasets [8]. For example, they have used an association rule to infer dangerous arsenic levels with 100% confidence.

As envisioned by Bhatnaghar and Kochhar, association rules mining performing on stream data are increasingly in need. They are employed in the estimation of missing data streams of data generated by sensors and frequency estimation of internet packet streams [7].

Production Rules

Sottara et al. (2012) models a hybrid Environmental Decision Support System (EDSS) for Waste-Water Treatment Plants (WWTP). They argue that the WWTP domain should be modeled through ontologies, for modeling sensor data, in pair with decision-making rules, for processing incoming sensor data and recommending actions to be taken. As an example of production rule they infer invalid NO3 measurement values.

Another production rules implemented system has been designed by Chau (2007) in the domain of water quality modeling. Namely, the system simulates human expertise during the problem solving of coastal hydraulic and transport processes. Both forward-chaining and backward-chaining are used collectively during the inference process [13].

Event Processing

Another hybrid approach while dealing with sensor data reasoning is using OWL ontologies and CEP which is a similar area as Stream Reasoning. Taylor and Leidinger (2011) translate the whole OWL ontology, which models the event definition and optimization and extends an early version of SSN ontology, into CEP statements for processing in an event processing engine. Unlike this approach, Anicic et al. (2011) have taken the advantage of both "worlds" synthesizing the ability of CEP systems to process real-time complex events within multiple streams of atomic occurrences and the Semantic Web i.e. ontological ability to effectively handle background knowledge and perform reasoning. The later approach has resulted with a new rule-based language ETALIS [14] and EP-SPARQL [15], a query language extending the SPARQL language with event processing and stream reasoning capabilities. Both are implemented in Prolog, which has its foundations in Logic Programming (LP).

Semantic Web rules

As previously mentioned on Section 2.1.1, the recommended rule system for Semantic Web applications is SWRL. However, since 2005 the W3C has formed the Rule Interchange Format (RIF) Working Group for building a standard for exchanging rules among rule systems. As a result, the initiative has offered to the community a family of languages with well-founded syntax and semantics. Namely, for the logic-based rule systems, i.e. first-order and LP, the group has defined Basic Logic Dialect (RIF-BLD) and a subset the RIF Core Dialect. As for action rules system, i.e. production rules and reactive rules, the group has defined Production Rule Dialect (RIF-PRD). An extensibility framework, Framework for

Logic Dialects (RIF-FLD), has been also defined motivated by the diversity of the logical theories underlying the different logic-based rule systems.

RIF offers a unique standard for rule exchange between different rule systems and was not intended to bring a one-fits-all rule language. On the other side, SWRL's tight coupling with OWL provides no translation and reasoner issues. Because RIF is an exchange rule language and using SWRL with OWL has distinct advantages, studying SWRL applications over stream data domains was set as the main subject of this thesis.

Dealing with sensor data, a pure Semantic Web approach has been implemented by Keßler et al. (2009). They have utilized the SWRL's ability to express free variables and the use of its built-ins for modeling mathematical functions which has fulfilled the OWL's lack of mathematical processing capabilities. The approach is tested for geographical information retrieval (GIR) task for recommending personalized surf spots based on user location and preferences. A similar approach is taken by Wei and Barnaghi (2009) who demonstrate how rule-based reasoning can be performed over sensor observation and measurement data within the terms of Semantic Sensor Web. They emphasize the ability of rules not just to infer accurate but also approximate knowledge.

Henson et al. (2009) have used Jena Semantic Web Framework [12] as an engine for reasoning with rules implemented for Semantic Sensor Web on weather domain. Using Jena rules they infer new knowledge about sensor observation data and link the newly generated relations with original observation time and location data.

II. 4 Hypothesis and problem statement

The main hypothesis of this thesis is to prove that the Semantic Web rule layer technologies are capable for reasoning over stream data.

Layering SWRL rules over OWL ontologies is a recommended approach to be considered while building Semantic Web applications. SWRL supports declarative programming. Using a formal, declarative rules language that operates over a formal and declarative model, such as OWL, has distinct advantages. First of all SWRL rules are not bound to a particular execution algorithm when reasoning with a backward-chaining engine [64]. Unlike in production systems the rule expert should not be aware of any side-effects. No side-effects means no need to prioritize rules or have knowledge of the execution algorithm, simplifying rule design and maintenance [64]. Secondly, no translation or mapping system is required

between OWL DL model and SWRL rules. SWRL works directly with OWL classes and properties.

On the other side, using SWRL, which has become the de facto standard rule language in Semantic Web, has never been used in stream data applications. Its open world assumption and monotonic nature makes SWRL powerless for doing continuous inference over stream data. For example, using aggregate functions on a particular window of streams cannot be expressed in SWRL. A SR system should support reasoning over both streaming information and background data [153]. Moreover, some specific requirements about this property already mentioned in state of the art systems e. g. StreamRule [34], should also be considered. Namely, SR rule systems need to support a conjunction of reasoning features like: closed-world, non-monotonic, incremental and time-aware reasoning. The following subsections discuss these features in more detail.

II. 4. 1 Closed-world and non-monotonic reasoning

OWL and SWRL's open world assumption (OWA) and monotonic reasoning do not offer the desired expressivity level required in Stream Reasoning application domains. For example, modifying the river pollution status is not allowed through SWRL rules. Following the SWRL's monotonic nature a measurement site instance firstly asserted as "clean" cannot be later modified to "polluted".

Non-monotonic operators, aggregates and negation, are common requirements for processing data streams [80]. For example, aggregate operations are present in almost every rule for classifying water bodies into corresponding statuses [83] e.g. finding arsenic observations' average value. OWA's approach means one cannot "close" the world to calculate an average value. The SWRL's query language SQWRL [55] allows this through the use of `sqwrl:average` [148]. However, that approach is not supported, since using SQWRL constructs in SWRL rules for asserting new knowledge is not allowed [41].

Additionally, a number of example rules need to infer new knowledge in absence of a fact or incomplete knowledge, the concept known as NAF. For example, the rule "assign 'undetermined status' to those remaining bodies of water where the agency is not, by that date, in a position to assign a reliable interim classification due to a lack of data or other reason" [83] cannot be expressed in SWRL.

II. 4. 2 Incremental reasoning

Pre-computing and storing of implicit ontology entailments is a process known as materialization. Every time a change occurs, a new materialization need to be computed, which in Semantic Web is known as incremental maintenance of materialization [71]. In SR applications, change to the facts occurs "regularly". A technique for computing ontological entailments on SR is presented in [94]. It uses LP, respectively Datalog rules to compute incremental materialization for window-based changes of ontological entailments. This approach is concerned with computing complete and correct materialization enforced by changes to facts, i.e., facts are added or removed from the knowledge base.

According to [71], changes to the ontology will typically require changes in the rules. Authors of [71] describe a technique of this type of incremental materialization. The frequency of changes to the ontology in SR applications does not differ from the traditional Semantic Web ones. Therefore, the techniques developed for this type of incremental materialization intended for "static" knowledge bases would also be suitable for stream data knowledge bases.

II. 4. 3 Time-aware reasoning

SR systems should include time-annotated data i.e. the time model, and like Complex Event Processing (CEP) should offer explicit operators for capturing temporal patterns over streaming information [80]. INWS ontology implements the time model through OWL Time ontology [86]. Supporting temporal operators (serial, sequence, etc.) means the system can express the following example rule: Enhanced phosphorus levels in surface waters (that contain adequate nitrogen) can stimulate excessive algal growth [79]. If before excessive algal growth, enhanced phosphorus level has been observed then more probably the change of phosphorus levels has caused the algal growth. Thus, a sequence of these events needs to be tracked to detect the occurrence of this complex event.

Moreover, in order to enable reasoning in terms of time and quantity intervals of continuous and possibly infinite streams, the SR notion of windows needs to be adapted for rules [34]. In traditional settings, rules operate over all asserted facts in the ontology. This is not practical with stream data as data flow is massive and rules may not always consider all RDF streams. Thus, the concept of continuous rules is defined as follows:

Definition 1. Rules that are evaluated against a particular set of RDF streams selected by a time or tuple window are called *continuous rules*.

Rather than evaluating rules against all static and on-the-fly RDF streams as in traditional Semantic Web rule systems, continuous rules will run against a time or quantity constrained window. For example, a continuous rule to assert which sensors provided observation measurements that are above allowed average threshold the last 3 minutes, sliding the window every minute, will be easily expressible with the help of the time-based window.

Chapter III The INWS ontology

The purpose of this chapter is to describe the model of the WQM domain, which is used to validate our developed SR systems described in the following chapters. Namely, both Jess [72] and SWRL rules will be used to perform rule-based reasoning over the proposed model on this chapter on StreamJess and C-SWRL, respectively. StreamJess is a production rule system reasoning with stream data, while C-SWRL is a unique Semantic Web system for reasoning over stream data.

The chapter resumes with introductory notes followed by ontology requirements specifications on Section 3.2. The model design comes in Section 3.3 by describing its modules and two case studies for its usability testing. Finally, an expert system was developed to validate the approach.

III. 1 Introduction

The old-fashioned approach of monitoring water quality by collecting water samples manually and transporting them to a laboratory for analyses is expensive, time-consuming, prone to miss fluctuations of pollutant concentrations such as periodic release of toxins, may be limited by weather conditions, and does not allow for continuous data collection [32, 56].

On the other side, the technological improvements on the sensor and network capabilities for long range data distribution and storage provide a capable platform to utilize low cost, high performance and real-time monitoring Wireless Sensor Networks (WSN) for WQM.

Sensor data processing encapsulates processing historical data stored on permanent databases, as well as real-time stream data. Thus, a flexible knowledge management system is required to represent the water domain knowledge. The research community has integrated different representational schemes. Modern approaches are mainly ontology-based [4, 35, 42,

47, 49, 50]. The ontological capability of knowledge reuse and sharing is the main reason why the ontologies are best suited for modeling water quality monitoring domains.

The current state-of-the-art WSNs are employing diverse Semantic Web technologies to not just automate real-time monitoring of water health, but also enrich it with semantics. Different intelligent real-time WQM systems are established and currently in place, be it centrally managed (e.g. [35]) or distributed on sensor nodes (e.g. [50]). Query answering has been leveraged in [35, 51] over water domain ontologies, while in [37, 38, 6] ontologies in pair with rules are used for efficient WSN. Yet in terms of support for WQM of semantic technologies, according to [45], there is to date no WSN for WQM able to address all requirements on water quality standards set up by the Water Framework Directive (WFD) [46] which represents one of the main environmental challenges in EU water policy [60].

The recent emergence of Semantic Sensor Web (SSW) has enabled the interoperability of heterogeneous WSNs. The SSN ontology [4], an OWL2 [55] ontology, offers a unique knowledge management base for WSNs. It is used as a foundation for development of the INWS ontology, an ontology for WQM.

III. 2 Requirements Specification

Firstly, an ontology will be build to model a WSN for WQM system. In traditional settings, WSN architecture for WQM is composed of spatially distributed (1) *sensor nodes* (also called *motes*) for capturing water quality values through one or more sensor probes or automatic samplers, (2) *gateway nodes* (also called *sink nodes*), usually one per site, for data gathering and transferring to a (3) *remote monitoring center* which retrieves data, performs some validation rules, stores them in a database, and eventually raises an alarm event if any parameter value is out of its threshold or any other alarming event occurs.

Secondly, the ontology should model the observations made by sensing devices, e.g., by sensor probes or automatic samplers. Observation data must be recorded such as: location (latitude and longitude of the sensor node), time (the sampling and entry system time), and the water quality element (e.g., pH, temperature etc.). Additionally, the ontology needs to model devices. In particular, the ontology shall model data on where the devices are deployed (i.e., in which sensor nodes), what RFID they hold, and the type of devices.

Thirdly, the system should support classification of sensor observations based on different regulation authorities. It remains per future work to classify the observation with four regulation authorities: the WFD, UNECE standards [57] (statistical classification of surface freshwater quality for the maintenance of aquatic life), etc.

Finally, the ontology should model pollution sources. Pollutant is any facility or entity discharging to the water body.

A typical scenario for WQM in a WSN platform is as below:

Scenario 1. Water quality sensor probes are deployed in different measurement sites of a river. A sensor probe emits water quality values. One may want to (1) classify the water body into the appropriate status according to WFD regulations and (2) identify the possible polluter if the values are below the allowed standard.

III. 3 The Ontology Model

This section describes the INWS ontology, which will fulfill the requirements specified in the previous section. According to [35], three types of water quality monitoring knowledge need to be modeled: *observational data items* (e.g., the amount of ammonia in water) collected by sensing devices, *regulations* (e.g., safe drinking water acts) published by authorities, and *water domain knowledge* maintained by scientists (e.g., water-relevant contaminants, bodies of water, etc.). This model will be extended to capture the knowledge of sources of pollution. Namely, it consists of four ontology modules:

- *The core ontology*, consisting of classes and relationships for deploying real-time observational water quality data coming from data sources i.e. sensors and lab measurements
- *The regulations ontology*, a module which deals with permitted water parameter thresholds regulated by different authorities
- *The pollutants ontology*, a module representing pollutants entities and their attributes
- *Water expert rules*, a module representing if-then water expert rules

In order to reason with ontology modules in general, or to express *Scenario 1* in particular, all of these modules will be integrated into a single ontology. As depicted in Figure 4, sensor observation data are consumed in the core ontology. Water expert rules will classify water bodies to appropriate status following the regulations ontology model and core ontology

observation data. Additionally, expert rules based on polluting semantics modeled in the pollutants ontology identify the pollution causes.

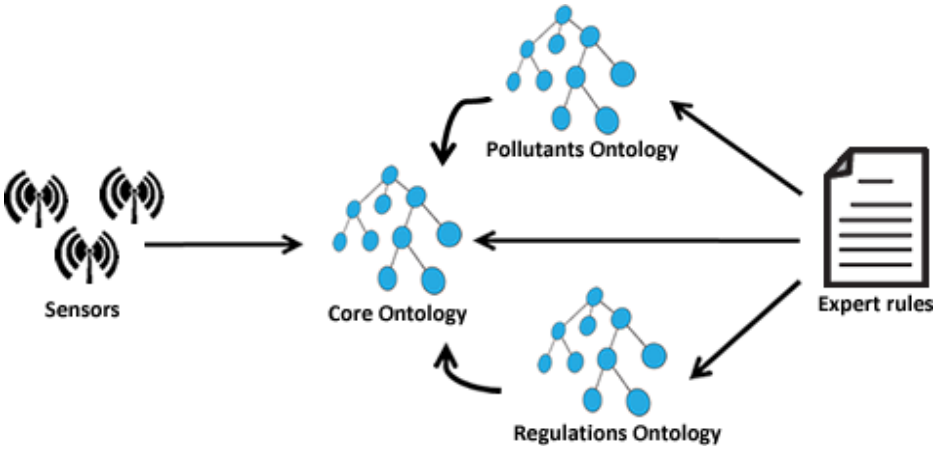


Figure 4. Ontology framework modules

III. 3. 1 The Core Ontology

Following the ontology design pattern used in [35], the core ontology will represent observational water quality data together with the corresponding descriptive metadata, including the type and unit of the data item as well as the provenance metadata such as the locations of sensor nodes, the time when the data item was observed and optionally the test methods and devices used to generate the observation. The SSN ontology has recently emerged as main upper ontology for modeling WSN knowledge bases. It can describe sensors in terms of capabilities, measurement processes, observations and deployments. Thus, this ontology is best suited to be used for our core ontology. It will eventually be extended with additional classes and relationships as needed by the system requirements. For example, for representing time related features OWL Time[5] ontology was used, while asserting geo location attributes longitude and latitude was realized through the basic geo location vocabulary[6]. The complete list of ontology namespaces used by the ontology modules is described in Table 1.

Because SWRL [27] rules are going to be employed in our framework, Protégé 3.5 was used as the main ontology development environment. Version 3 was chosen over 4 because of version 3's SWRL built-ins support. But, the SSN ontology is an OWL2 ontology which

[5] http://www.w3.org/TR/owl-time/
[6] http://www.w3.org/2003/01/geo/

cannot be directly imported in version 3. Hence, the desired SSN features were imported extending them with other ontologies.

Table 1. INWS ontology namespaces

Prefix	Namespace	Description
	http:// inwatersense.uni-pr.edu/ontologies/inws-core.owl#	INWS core ontology
ssn	http://purl.oclc.org/NET/ssnx/ssn#	The SSN ontology
body	http://sweet.jpl.nasa.gov/2.1/realmHydroBody.owl#	Describes water bodies like river, basin etc.
chem	http://sweet.jpl.nasa.gov/2.1/matr.owl#	Chemical substances ontology
elem	http://sweet.jpl.nasa.gov/2.1/matrElement.owl#	Chemical elements ontology
dul	http://www.loa-cnr.it/ontologies/DUL.owl#	Descriptive Ontology for Linguistic and Cognitive Engineering
event	http://www.csiro.au/EventOntology#	CSIRO event ontology
geo	http://www.w3.org/2003/01/geo/wgs84_pos#	Geographical location ontology
qu	http://www.purl.oclc.org/NET/ssnx/qu/qu#	Library for Quantity Kinds and Units
qurec	http://www.purl.oclc.org/NET/ssnx/qu/qu-rec20#	Ontology for Quantity Kinds and Units: units and quantities definitions
time	http://www.w3.org/2006/time#	OWL Time Ontology
twcc	http://tw2.tw.rpi.edu/zhengj3/owl/epa.owl#	TWC-SWQP core ontology
twcp	http://escience.rpi.edu/ontology/semanteco/2/0/pollution.owl#	TWC-SWQP pollution ontology

In Table 2, a summary of the concept additions following the imported SSN features is presented, whereas in Table 3 are listed the added properties. For brevity, OP is used as for "object property" notation and DP for "datatype property". For example, `CentralMonitoringNode` is modeled as both `geo:Point` and `ssn:Platform`, since it is an entity to which gateway nodes can be attached and it has geo location attributes.

A property `hasDevice` was added to indicate anything that is related with a particular device e.g. a sensor node consisting of a set of devices.

Table 2. Ontology class specifications of INWS core ontology

Class	Description and Axioms
event:Alert	The class of all alerts. Has subclass `event:EmailAlert` and `event:SMSAlert`.
DeviceType	The class of device types which include: sensors, auto samplers, RFIDs etc.
AutoSampler	The class of auto samplers. Subclass of `ssn:Device`.
WaterFeature	Subclass of `ssn:FeatureOfInterest`. `ssn:hasProperty some WaterQuality`
twcc:WaterMeasurement	Represents measurements as water feature. Subclass of `WaterFeature`.
time:Instant	The class of time individuals. `hasObservationTime some xsd:datetime`
time:Interval	The class of time intervals. `dul:hasIntervalDate some xsd:datetime`

body:Basin	Each basin passes through only one municipality. Each basin involves one or more rivers.
	`hasAnnualFlowing only xsd:float`
	`hasFlowingDirection some string`
	`hasFlowingQuantity some float`
	`hasMunicipality some Municipality.`

Municipality	In a community passes only one basin and some rivers.
	`hasCatchment only body:Basin`
	`hasCatchment exactly 1`
	`hasRiver some body:River.`

body:River	The class of all rivers. `hasBasin some body:Basin`
CentralMonitoringNode	Subclass of `geo:Point` and `ssn:Platform.`

GatewayNode	Subclass of `geo:Point` and `ssn:Platform.`
	`hasCentralMonitoringNode some CentralMonitoringNode`
	`hasMeasurementSite only twcc:MeasurementSite`

	hasSensingNode only SensingNode
SensingNode	Subclass of geo:Point and ssn:Platform. dul:hasLocation only SensingNodeLocation hasDevice some ssn:Device hasDevice min 1 ssn:Sensor hasDevice max 1 AutoSampler hasRFID min 1.
SensingNodeLocation	Subclass of twcc:MeasurementSite.
RiversWaterQuality	Subclass of WaterQuality. Super class of categories of quality elements: Biological, Hydromorphological and Physico-chemical.

Regarding the SSN features the following modifications/extensions were applied in order to fulfill the needs for the WQM domain:

- In ssn:Sensor class were added subclasses based on the water quality element measured by the sensor. For example, DissolvedOxygenSensor will hold sensor devices measuring dissolved oxygen. A sensor measuring more than one element can be instance of more than one ssn:Sensor subclasses.
- In the ssn:Observation class to describe the observation location the following two axioms observationResultLocation only geo:Point and observationResultLocation min 0 were added.
- In the class ssn:Platform the following axiom has been added dul:attachedSystem owl:hasValue InWaterSense indicating that all ssn:Platform instances will be attached to our system instance named InWaterSense.
- A ssn:FeatureOfInterest subclass WaterFeature was added, having a subclass twcc:WaterMeasurement, which in turn will eventually hold instance RiversWaterMeasurement for our first case study and DrinkingWaterMeasurement for the second one.

Table 3. Ontology properties specifications of INWS ontology

Property	Description and Axioms	Type
hasDevice	Anything that has as its range ssn:Device.	OP

hasMeasurementSite	Anything that has as its range `twcc:MeasurementSite`. Inverse property of `isMeasurementSiteOf`.	OP
isMeasurementSiteOf	Anything that has as its domain `MeasurementSite`	
isDeviceType	Is used to represent device types. Domain: `Device`. Range: `DeviceType`.	OP
observationResultLocation	Domain: `ssn:Observation`. Range: `geo:Point.`	OP
twcc:hasMeasurement	Is sub property of `ssn:hasProperty`. Has sub properties: `ssn:hasMeasurementCapability` and `ssn:hasMeasurementProperty`	OP
hasObservationStartTime	Used for describing start time of intervals for observations. Range: `xsd:datetime`	DP
hasObservationEndTime	Used for describing end time of intervals for observations. Range: `xsd:datetime`	DP
hasObservationTime	Used for describing an instant time of observations. Range: `xsd:datetime`	DP
geo:lat and geo:long	For expressing geo location parameters.	DP

III. 3. 2 Regulations Ontology

According to [35], regulations concerning water quality have not been modeled as part of any existing ontology so far. Their attempt has resulted with a basic regulations ontology following different authoritative water quality regulations. Based on our system requirements the regulation ontologies will be modeled following the WFD regulations [46]. The system is open to include other regulation authorities.

A class `Standard` holds all the regulations authorities. In the next subsection the WFD regulations ontology will be described, while the others are suggested as future works.

The WFD regulations ontology

The WFD regulations classify water quality parameters into three broad categories: biological, hydromorphological and physico-chemical [46]. This categorization is illustrated in ontological class-hierarchy representation in Figure 5.

In WFD, instead of classifying water bodies as polluted or clean as was used in [35], water bodies are classified through five statuses and corresponding color: high/blue, good/green, moderate/yellow, poor/orange and bad/red. In WFD a general rule, called one-out-all-out, applies: The quality element with the lowest (worst) status for a water body determines the overall ecological status [46].

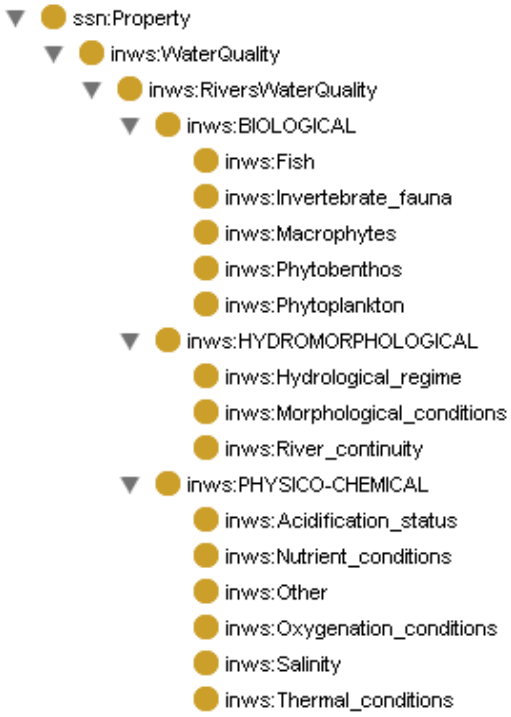


Figure 5. WFD categorization of water quality elements in Protégé class/hierarchy terms

A class named `WFDSurfaceWaterStatus` was used to capture the status hierarchy. The semantics of status/color pairs are captured through `owl:equivalentClass`. To express the WFD ecological status provided for different quality category a class `EcologicalStatus` was created. Since the latest class is about WFD regulations there is an `owl:Restriction` restricting the `hasStandard` property to have values only from `WFD` class. Class `twcc:WaterMeasurement` was reused as a super class of all water quality statuses e.g. `HighAmmoniaMeasurement`.

In [35] regulation status is expressed through OWL property restrictions. Based on SSN ontology design pattern it was impossible to do this in ontology level. This is a consequence of involvement of more individuals representing a single sensor data stream. SWRL's support of free variables is a suitable solution for expressing this rationale. For example, the following WFD rule "*If total ammonia is less than 0.04 (mean), than river belongs to the high status of nutrient conditions*" assuming that observations are queried after 2013-02-13 on 09:11, can be expressed through the following SQWRL query:

```
ssn:Observation(?x) ∧ ssn:observedProperty(?x, Ammonia) ∧
ssn:observationResultTime(?x, ?y) ∧ hasObservationTime(?y,
?z) ∧temporal:after(?z, "2013-02-13T09:11:00") ∧
ssn:observationResult(?x, ?r) ∧ ssn:hasValue(?r, ?v) ∧
dul:hasDataValue(?v, ?val) ∧ sqwrl:makeSet(?sv, ?val) ∧
sqwrl:avg(?avg, ?sv) ∧ swrlb:greaterThan(?avg, 0.04) →
HighAmmoniaMeasurement(?o)
```

III. 3. 3 Pollutants ontology

The pollutants ontology will model facilities and other entities discharging wastes in water bodies. The semantics modeled in this ontology in cooperation with other ontology modules will help to identify the possible cause of the pollution.

The INWS pollutants ontology was designed based on examples of sources of pollution and the potential pollutant discharges which could arise described in [79]. As depicted in Figure 6, two classes were added: `PollutionSources`, describing the sources of pollution e.g. urban storm water discharges, and `Pollutants`, representing contaminants present in the environment or which might enter the environment which, due to its properties or amount or concentration, causes harm e.g., heavy metals. A property `potentialPollutant` links individuals of `PollutionSources` and `Pollutants` (based on the Table on page 3 in [79]). `PollutionSources` class was also linked with a string through two properties: `pollutionSourceName`, representing the name of the pollution source, and `pollutionType`, representing the type of the pollution source which can be point, diffuse or both of them. Moreover, a property `hasSourcesOfPollution` was added to relate the rivers with the sources of pollution.

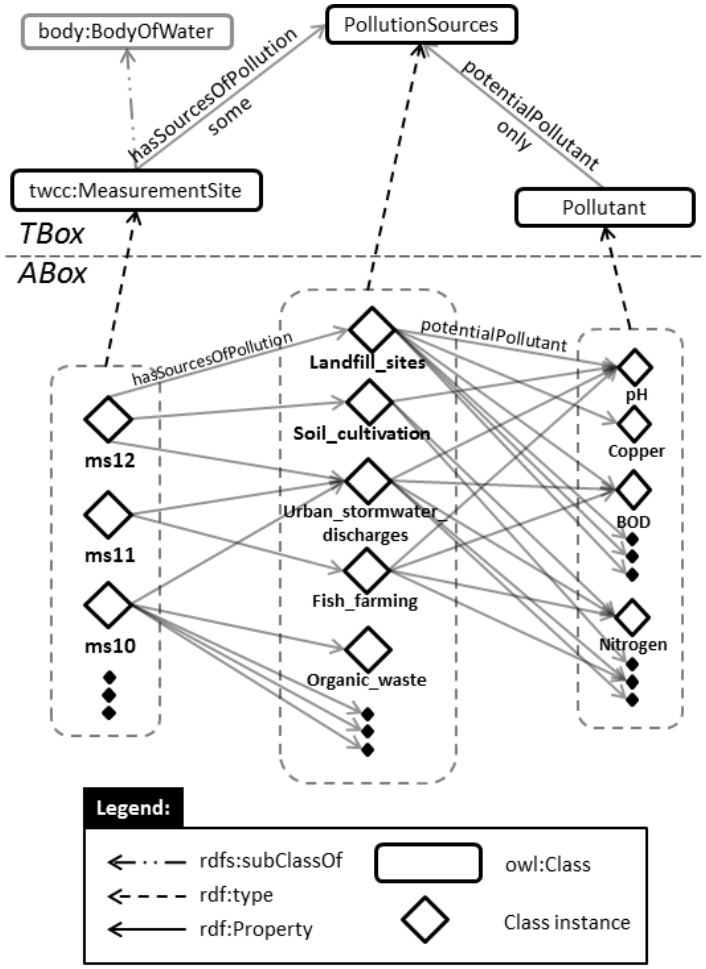


Figure 6. TBox and ABox statements for the INWS pollutants ontology module

III. 3. 4 Use Cases

Considering the domain of water quality management two use cases were approached to illustrate the usability of INWS ontologies. In particular, a stream data scenario from the domain of surface water quality management and static data scenario from the domain of drinking water quality management.

Use Case 1: Surface Water Quality Management

In absence of real sensor observation data the INWS ontology was investigated in the domain of surface waters with simulated SQL data. An SQL stream data generator was employed to produce simulated water quality data. For detailed description of the dataset the reader is advised to refer Appendix B Section 10.1. The generated data are then converted to RDF data

through D2RQ[7] mapping tool. Populating the INWS ontology with the D2RQ generated data in Protégé implied difficulties on rendering object property instances. Namely, instead of `rdf:Description` statements, Protégé 3.5[8] expects abbreviated syntax for object property instances. The following D2RQ generated code snippet describes an object relation linking the sensor node instance `sn3` with a sensor node location instance `sl3`:

```
<rdf:Description rdf:about="sn3">
   <dul:hasLocation rdf:resource="sl3"/>
   <rdf:type rdf:resource="&ont;SensingNode"/>
</rdf:Description>
```

The same assertion in terms of abbreviated RDF/XML syntax (expected in Protégé) is:

```
<SensingNode rdf:about="sn3">
   <dul:hasLocation rdf:resource="sl3"/>
</ont:SensingNode>
```

In order to enable this translation SWOOP [107] was used to load the D2RQ generated RDF data and produce the abbreviated syntax description of object property instances. SWOOP derived ontology is then imported in Protégé 3.5 by populating corresponding class and property assertions of the core ontology.

For this case study the following assumptions were asserted into the core ontology:

- Axiom `ssn:featureOfInterest owl:hasValue RiversWaterMeasurement` was added to indicate that all observation's feature of interest is river water quality
- `ssn:sensingMethodUsed owl:hasValue SimulatedData`
- `ssn:includesEvent owl:hasValue ScheduledObservation`

For example, Figure 7 illustrates a fragment of an observation instance of stream data, namely `oo11724`. As can be observed from the figure: the observation instance is a water temperature measurement, which is a river feature (`oo11724 ssn:observedProperty Temperature, Temperature ssn:isPropertyOf RiversWaterMeasurement`); it was produced by a device named d1 (`oo11724 ssn:observedBy d1`); it was sampled on 2013-02-13 at 09:32:22 and it has the same entry system time since there is no latency i.e. data are already in machine (`oo11724 ssn:observationSamplingTime v11724, oo11724 hasObservationTime`

[7] D2RQ Accessing Relational Databases as Virtual RDF Graphs, http://d2rq.org/
[8] Protégé ontology editor, http://protege.stanford.edu/

"2013-02-13T09:32:22.133"^^<xsd: date>); it's measured value is 15.58 (oo11724 ssn:observationResult so11724, so11724 ssn:hasValue ov11724, ov11724 dul:hasDataValue "15.58"^^<xsd:double>); it was measured from s2 sensing node (hasSensingNode sn2) and the sample position is 21.0E0 for longitude, and 42.0E0 latitude (oo11724 observationResultLocation l11724, l11724 geo:lat "42.0E0"^^<xsd:double>, l11724 geo:long "21.0E0"^^<xsd:double>).

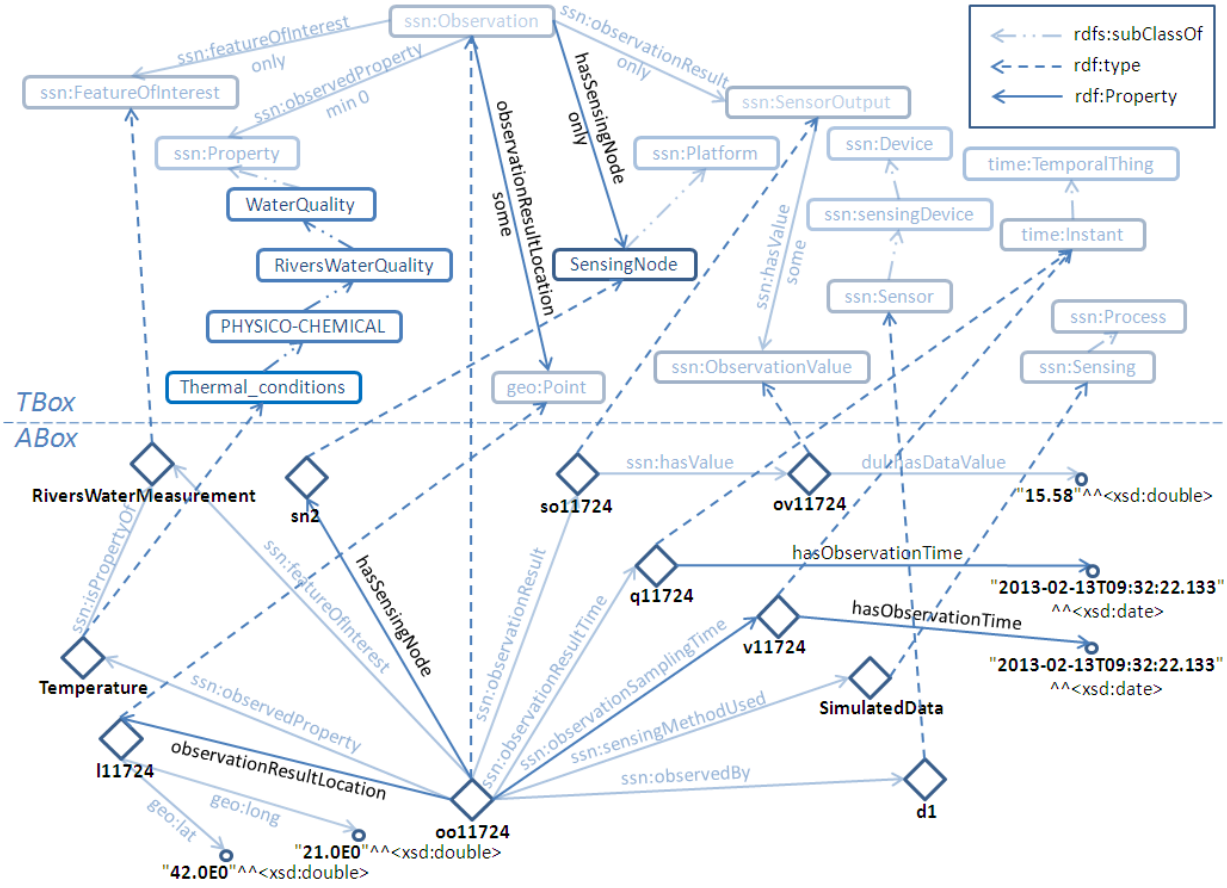


Figure 7. TBox and ABox statements for the surface waters case study

To query about each sensor node for which water quality elements is observing, the following SQWRL [109] query can be applied:

```
ssn:Observation(?x) ∧ inws:hasSensingNode(?x, ?y) ∧
ssn:observedProperty(?x, ?z) ∧ sqwrl:makeSet(?sx, ?x) ∧
sqwrl:groupBy(?sx, ?z) → sqwrl:select(?y, ?z)
```

For our simulated data this query produced the output depicted in Figure 7.

Use Case 2: Drinking Water Quality Management

Drinking waters represent another water quality management domain. INWS ontology supports population with data from this domain. CSV data available from [99] were converted to RDF to populate the ontology. In Appendix A are given the conversion details and mapping mechanisms. Data were taken from measurements made in 15 measurement sites in the city of Tetova (Macedonia) during three summer months of 2012: June, July and August.

?y	?z
inws:sn2	inws:Conductivity
inws:sn2	inws:Temperature
inws:sn2	inws:Turbidity
inws:sn3	inws:Ammonia
inws:sn3	inws:Sulphate
inws:sn3	inws:Temperature
inws:sn4	inws:TotalNitrogen
inws:sn5	inws:Temperature
inws:sn5	inws:TotalPhosphorus
inws:sn6	inws:Temperature

Figure 8. A sample rule output

Axiom `ssn:featureOfInterest owl:hasValue DrinkingWaterMeasurement` was added to indicate that all observation's feature of interest is drinking water quality. Figure 9 illustrates an observation instance `AugObserveChloridesT9` representing measured values of Chlorides (`AugObserveChloridesT9 ssn:observedProperty DrinkingWaterChlorides, DrinkingWaterChlorides ssn:isPropertyOf DrinkingWaterMeasurement`) during August 2012 (`AugObserveChloridesT9 ssn:observationResultTime August2012, August2012 ssn:startTime ObservationAugustStart, ObservationAugustStart time:inXSDDateTime "2012-08-01"^^<xsd:date>, August2012 ssn:endTime ObservationAugustEnd, ObservationAugustStart time:inXSDDateTime "2012-08-31"^^<xsd:date>`) on measurement point T9 (`AugObserveChloridesT9 ssn:observationResultLocation T9`) with measured Chloride value 8.3 (`AugObserveChloridesT9 ssn:observationResult AugOutputChloridesT9, AugOutputChloridesT9 ssn:hasValue AugValueChloridesT9, AugValueChloridesT9 dul:hasDataValue "8.3"^^<xsd:decimal>`).

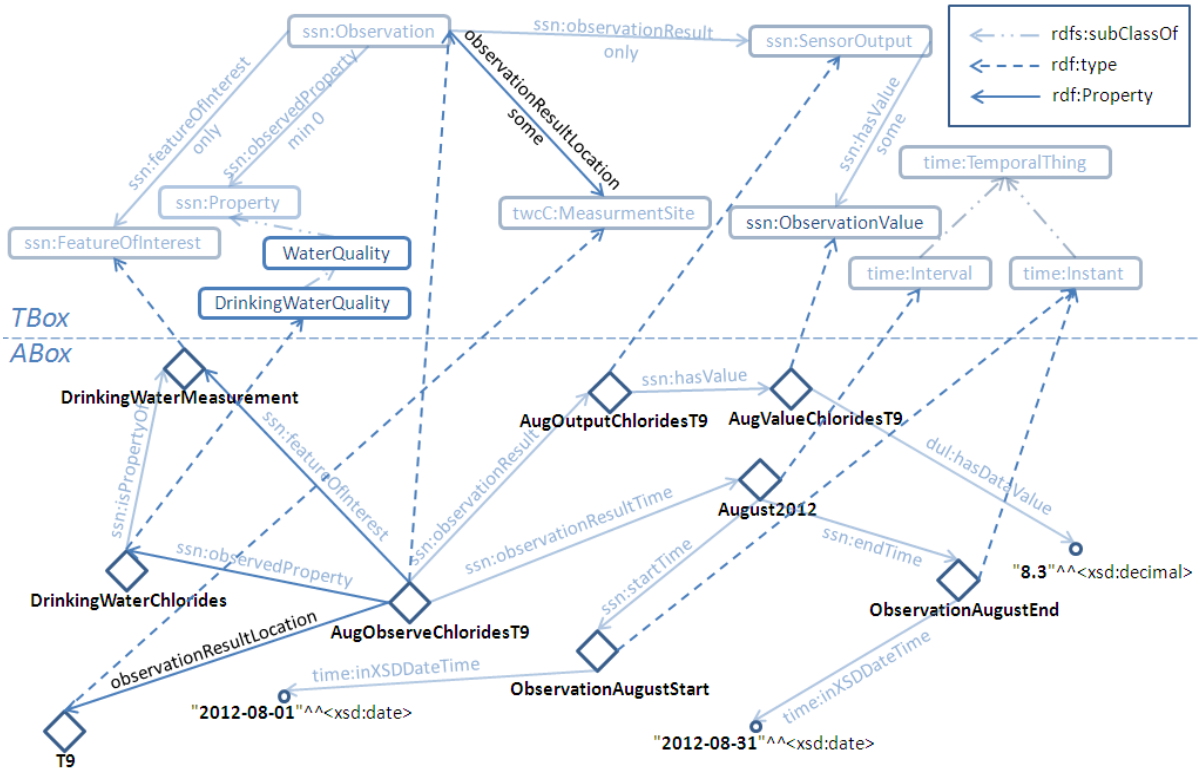


Figure 9. TBox and ABox statements for the drinking waters case study

If one would like to calculate the median of June temperature observations, the following SQWRL rule produced the same result obtained in [99] through Excel formulas:

```
ssn:Observation(?x) ∧ ssn:observedProperty(?x,
DrinkingWaterTemperature) ∧ ssn:observationResult(?x, ?r) ∧
ssn:hasValue(?r, ?v) ∧ dul:hasDataValue(?v, ?val)  ∧
sqwrl:makeSet(?sv, ?val) ∧ sqwrl:median(?m, ?sv) →
sqwrl:select(?m)
```

III. 4 An expert system for validating the INWS ontology

The INWS ontology was validated with an expert system [100]. It was developed using the Java Expert System Shell (Jess) [72]. Jess is a rule engine and scripting environment written in Java. The main characteristics of the Jess system for WQM system are described in this section. Namely, it classifies water bodies based on observed water quality values and investigates eventual sources of water quality degradation. However, the proposed approach does not support stream data which is subject to be included in another proposed approach StreamJess described in the next chapter.

As depicted in Figure 10, our system's architecture consists of three layers: *data*, *INWS ontology* and *rules layer*. The RDF data (up left) and RDF streams (up right) constitute the *data layer* (grey track). Arrows describe data flow direction. Domain specific ABox knowledge which does not change or changes "slowly" is formulated in the form of RDF data e.g. river names. RDF streams are defined as a sequence of RDF triples that are continuously produced and annotated with a timestamp [9]. Water quality measured values, annotated as RDF streams, will continuously populate the core ontology. In particular, a single RDF stream will hold information of observed water quality value, timestamp and location. The middle part of Figure 10 represents the *INWS ontology* (green track) described in the previous section. The *rule layer* (yellow track) consists of common rules (bottom left) and continuous rules (bottom right).

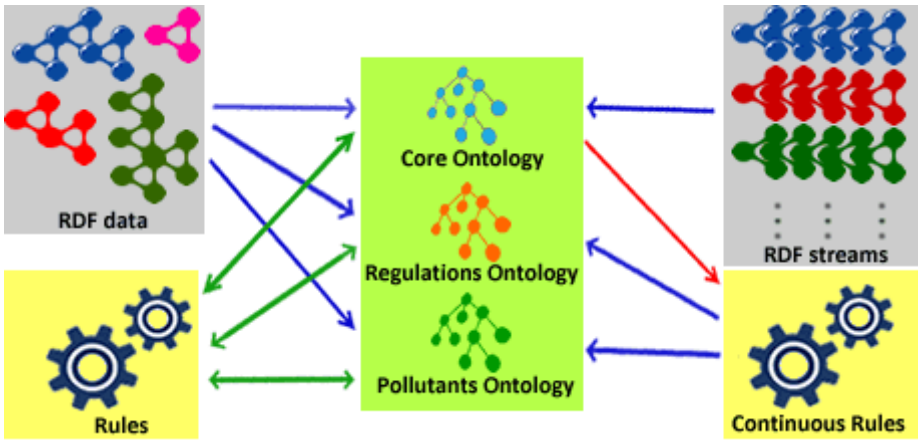


Figure 10. INWS conceptual framework: data layer (grey track), ontology layer (green track) and rules layer (yellow track)

Jess rules were decided to be used as a platform for implementing our system of reasoning over the INWS ontology framework. As a production rule system, Jess supports closed-world and non-monotonic reasoning. Moreover, it has a tight integration with Java through Jess's Java API and Protégé through JessTab[9] plugin. JessTab is a plug-in for the Protégé[10] ontology editor and knowledge-engineering framework that allows one to use Jess and Protégé together. The system was validated with simulated data, but it was developed for use within the InWaterSense project with real data.

The Jess implemented architecture of our system and its related components for reasoning over the INWS ontology are presented in Figure 11. Namely, input data in their available

[9] http://www.jessrules.com/jesswiki/view?JessTab
[10] Protégé ontology editor, http://protege.stanford.edu/

format, say SQL, are transformed into RDF streams using D2RQ[11] tool. SWOOP [72] is used to load the D2RQ generated RDF data and produce the abbreviated RDF/XML syntax for object property instances to be readable by Protégé [62]. RDF data streams are next imported into the core ontology. The set of rules for water quality classification based on WFD regulations are defined and may run against the knowledge base. Moreover, a set of rules for investigating sources of pollution by observing if eventual critical events appear are defined and may be activated. A simple user interface was developed using Java Swing[12], which offers a user to monitor water quality based on the WFD regulations and to eventually find the possible sources of pollution.

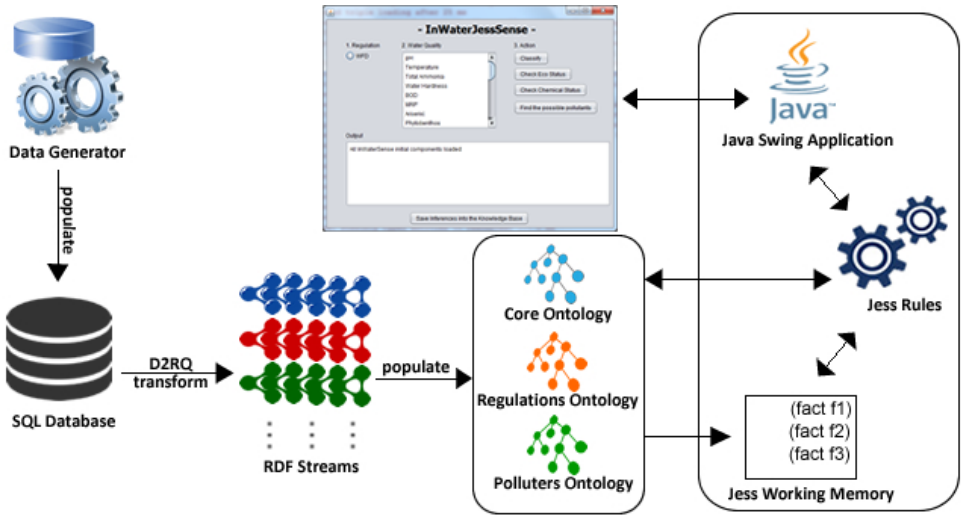


Figure 11. Jess implemented architecture for WQM

III. 4. 1 Implementation of a water quality monitoring scenario

To implement the *Scenario 1* using our system interface, as depicted in Figure 12, one should select the regulation authority i.e. WFD, select the water quality parameters which are to be monitored and press the button "Classify". The `JTextArea` below the "Output" label serves for printing rules messages.

The system offers multiple selections of water quality parameters. A simple rule is fired at application startup to set up the observations interval beginning time from the earliest time of observations streams and end time from the latest one. For brevity and clarity, Biochemical Oxygen Demand (BOD_5) observations will be demonstrated based on WFD classification.

[11] D2RQ Accessing Relational Databases as Virtual RDF Graphs, http://d2rq.org/
[12] http://openjdk.java.net/groups/swing/

According to WFD regulations: *if BOD_5 observations' average value is between 1.3 and 1.5 mg O_2/l then river belongs to "Good" status of oxygen condition, if the average is below 1.3 then river belongs to "High" status, else the river belongs to "Moderate" status.* Expressing this rationale with Jess rules was done through a number of rules. Namely, a rule of primer priority creates auxiliary Jess facts holding BOD_5 measurement values coming from the RDF streams. It is natural to use observation values directly from the ontology mappings, but the Jess rule which calculates the average value constrains the usage of Jess facts.

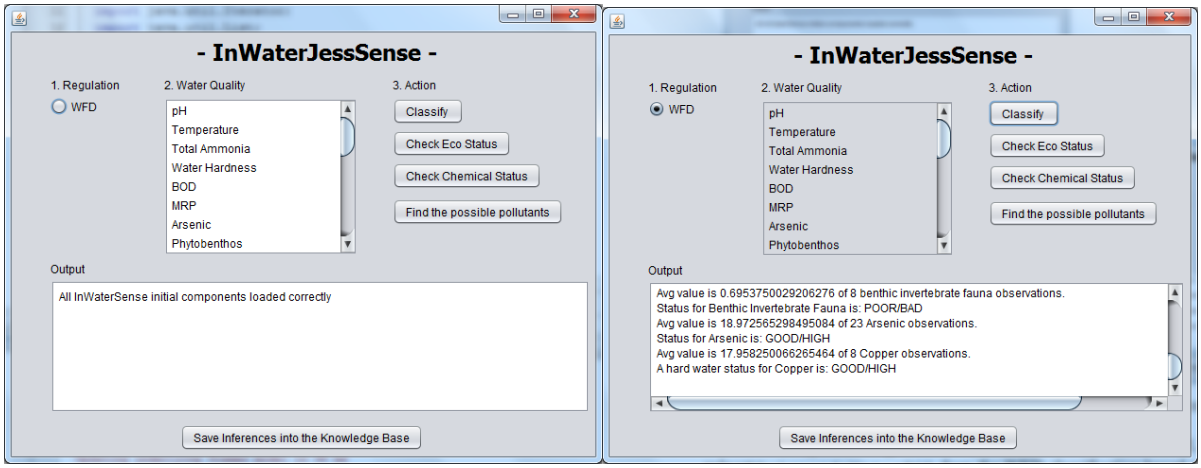


Figure 12. The Jess system interface: initial view (left) and after WFD classification view (right)

The calculated average value is asserted as a fact into the WM. Finally, another rule `WFDclassifyWaterBOD` does the WFD classification based on the previously asserted average value. This rule is illustrates below:

```
1  (defrule WFDclassifyWaterBOD
2  (BODaverage (v ?x)) (CurrentInterval (v ?i)) =>
3  (if (and (< ?x 1.5) (> ?x 1.3)) then (and
4  (printout t "Status for BOD is: GOOD" crlf)
5  (make-instance (str-cat "GoodBODStatus" ?*r*) of http://.../inws-
regulations.owl#GoodBODMeasurement map)
6  (make-instance (str-cat "ObservationInstantBOD" ?*r*) of
http://.../inws-regulations.owl#ObservationInstant map)
7  (slot-insert$ (str-cat "ObservationInstantBOD" ?*r*)
8  http://www.w3.org/2006/time#inXSDDateTime 1 ((new Date) toString))
9  (slot-insert$ (str-cat "ObservationInstantBOD" ?*r*)
10  http://.../inws-regulations.owl#hasStatus 1
11   (str-cat "http://.../inws-core.owl#GoodBODStatus" ?*r*))
12 (slot-insert$ (str-cat "http://.../inws-core.owl#" ?i)
13   http://.../inws-regulations.owl#hasStatus 1
14   (str-cat "http://.../inws-core.owl#GoodBODStatus" ?*r*))))
15 (if (< ?x 1.3) then <HIGH status classification code here>)
16 (if (> ?x 1.5) then <MODERATE status classification code here>))
```

Code in Line 1 serves for declaring a rule definition and its name. Line 2 represent the left hand side of the rule while lines 3-16 the right hand side of the rule. The previously calculated average value is assigned to variable `?x` while the current interval of observations present in the WM is assigned to `?i` (Line 2). If `?x` is between 1.5 and 1.3 begin assertions for good status (Line 4-14). Namely, a message is printed out (Line 4); a new instance of regulations ontology class `GoodBODMeasurement` is created (Line 5) (`?*r*` is a global variable holding random integer numbers); a new instance of `ObservationInstant` class is created (Line 6) associated with current date and time through `inXSDateTime` property (Line 7-8). This instance is also related with the instance created in Line 5 through `hasStatus` property (Line 9-11). Current interval instance (Line 12) is associated with the newly asserted status instance (Line 13-14). The same steps presented in line 4-14 are performed for the high and moderate status, which are omitted for brevity (Line 15-16).

The second part of *Scenario 1* is encoded through a couple of rules. The first one detects newly asserted instances of moderate status i.e. instances of `ModerateBODMeasurement` class. If there is at least one instance the second rule will fire and find BOD_5 sources of pollution discharging in the river body. An example of BOD_5 observations status is illustrated in Figure 13. BOD_5 sources of pollution are also listed after the user has clicked the "Find possible pollutants" button.

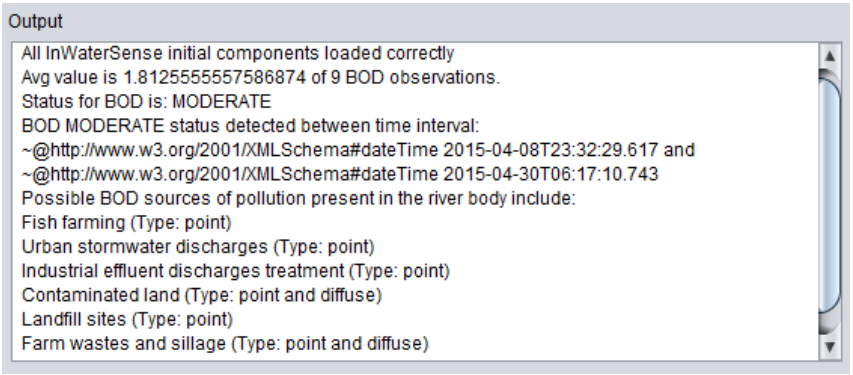


Figure 13. Scenario 1 example output for BOD5 observations WFD classification and sources of pollution

Chapter IV StreamJess

This chapter describes StreamJess, our expert system for WQM. Similarly to the system described in Section 3.4, it uses Jess rules to enable closed-world, time-aware and non-monotonic reasoning. However, the main difference between them is that StreamJess enables stream data support as contrary to bringing the input data manually, which was the case of the previous one. To offer the stream data feature, StreamJess utilizes C-SPARQL [93] abilities to filter and aggregate RDF streams on windows. The next section describes the proposed system design and its implementation. The chapter closes with Section 4.2, which gives evidence of system validation.

IV. 1 System design and implementation

This section describes the conceptual architecture of the proposed approach and its implementation. The domain of WQM is used as an illustrating case of stream data applications.

The conceptual architecture of StreamJess is depicted in Figure 14. It consists of three layers: data, ontology and rules layer. The RDF data (blue track left) and RDF streams (blue track right) constitute the data layer. The green track of the figure represents the ontology model. The concept of continuous rules described earlier in Section 2.4.3 is depicted by the pink track of Figure 14. They will continually infer new facts by reasoning over running RDF streams. These rules in StreamJess mainly fall into two broad categories:

- *monitoring rules* (pink track left), rules for continuous classification of water bodies based on in situ observations, and

- *investigation rules* (pink track right), which fire after monitoring rules detect any critical status. The information of sources of pollution stored into the pollutants ontology is used to prejudge the causer of the pollution.

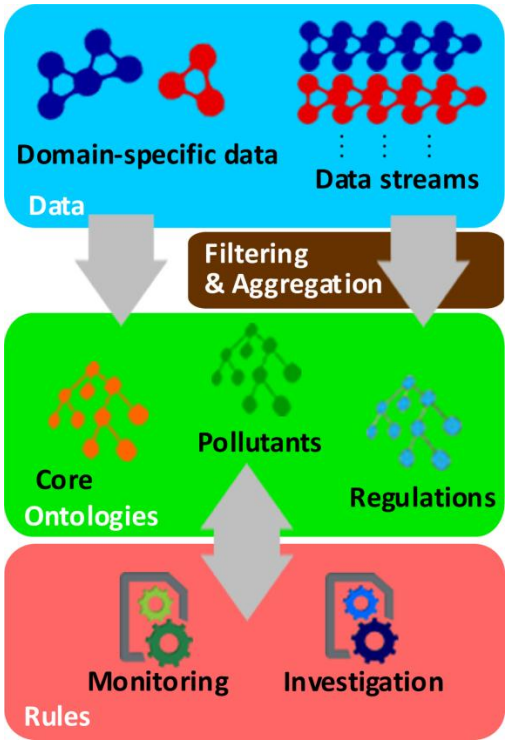


Figure 14. StreamJess conceptual architecture

In another domain, say medicine, the monitoring rules will continually classify the human's health status, while the investigation ones will try to identify the potential sources of the disease in cases of critical status detection. In StreamJess, both kinds of rules are loaded at system start up together with other Jess Tab commands described in Appendix C. Grey arrows describe data flow direction. As illustrated in Figure 15, our system acts as a pipeline. Sensor produced or simulated RDF streams are firstly filtered and aggregated by C-SPARQL queries. C-SPARQL results are published as observation data in the working memory and on the ontology. The running Jess Rete engine indicates the facts change and infers new knowledge according to the loaded rules.

Before implementing StreamJess, in order to enable Jess rules to reason over stream data, three approaches were considered:

- Extending Jess with stream data reasoning features,
- Translating Jess to another rule system which supports stream data reasoning and

- Layering Jess on top of another system to fill the gaps of Jess in support of stream data reasoning.

Extending Jess with stream data reasoning features is very expensive. Event stream processing with Jess is a fragile system, the code is complex and a lot of interferences have to be taken into account [102]. As the author of [102] argues, code could not be optimized even for simple temporal operations over event-streams. Another approach would be to translate Jess constructs into any CEP system. To the best of our knowledge there is not any evidence of such an approach. Albeit of the translation overhead we do not have confidence of how the system would perform.

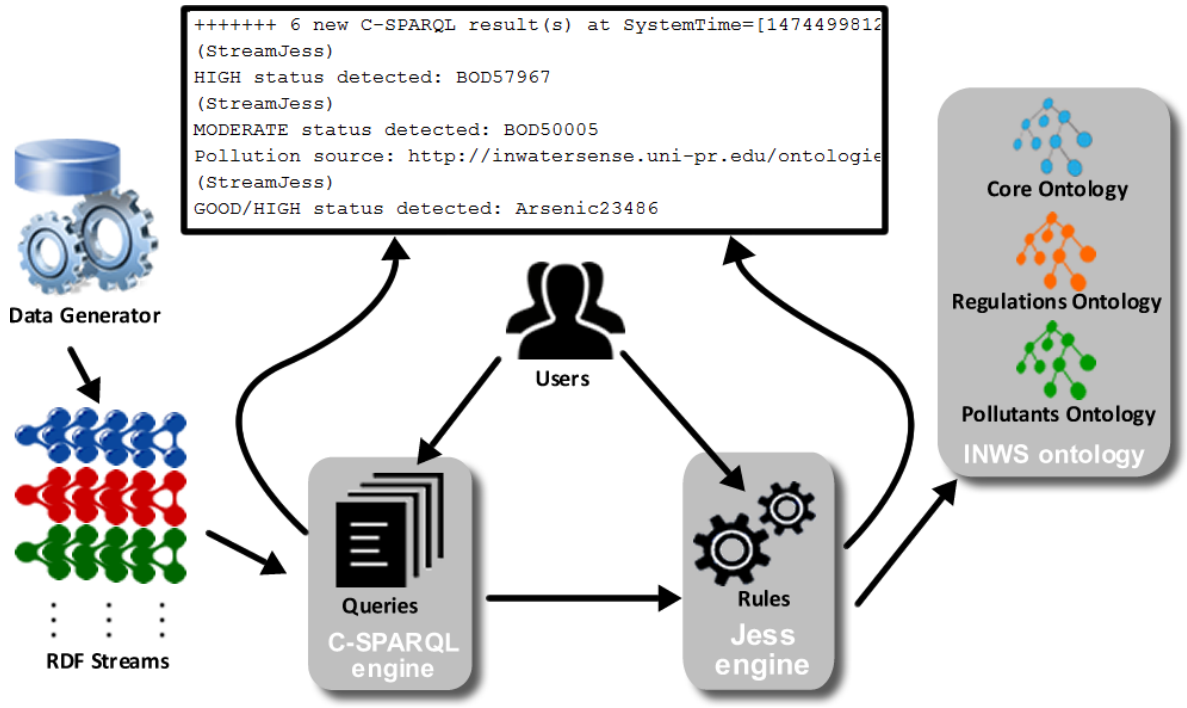


Figure 15. StreamJess system workflow

Given the drawbacks if approaching any of the previous two options, it was decided to layer Jess over an existing SR system such as C-SPARQL. C-SPARQL supports time-aware reasoning on stream data. However, as a query language, it is not intended to have any effect on the underlying ontology. In StreamJess, Jess rules are used for populating the knowledge base. Moreover, they enable data modifications i.e. non-monotonic reasoning and the tools for archiving data.

Each C-SPARQL query in StreamJess eventually outputs triples of values: the water quality name, the location of measurements and the calculated value. Every output triple is mapped into a temporary observation class. Furthermore, for each new incoming triple a new call to

the Rete method `run()` is invoked for doing rule-based reasoning. As illustrated in Figure 15, the Jess engine runs the rules against the temporary observation facts and it eventually activates the rule's RHS actions. The inferred knowledge forms another set of RDF data which is stored back into the ontology for further reasoning. Namely, monitoring rules do the water quality classifications based on the WFD regulation rules. In case a critical status is detected, investigation rules act to identify the pollution source.

StreamJess is implemented as a Java console application. The application uses an instance of `jess.Rete` which is created at system start up. It provides the central access point of the application as it loads the ontology, builds the working memory, holds the list of rules and offers the methods for doing CRUD operations over facts i.e. ontology individuals [19]. Multiple C-SPARQL queries and Jess rules can be defined to run over running observations.

StreamJess is open for loading other SR domain ontologies and write appropriate C-SPARQL queries and Jess rules. It is open source software and its installation details can be found on Appendix D.

IV. 2 Examples of StreamJess

As a proof of concept, StreamJess was implemented in a typical WQM scenario i. e. Scenario 1, based on WSN. In general, each water quality is monitored and investigated with a monitoring rule and an investigation one. A couple of examples are used to validate the system performance. Both examples run at the same time over the same RDF streams which are filtered out by two different C-SPARQL queries: one for finding the average values of water quality observations and another one for considering observation values one by one. The simulator was set up to randomly generate observation data for an arbitrary number of 70 measurement sites and 11 water quality parameters. For details about the dataset format one should refer Appendix B Section 10.2. A single sensor observation was arbitrarily set to be produced every second and includes 6 RDF streams representing time, location, device and quality of observation information. For example, in a 20 seconds window size 120 tuples will be produced. Moreover, the system supports registering multiple streamers to run concurrently.

IV. 2. 1 Example 1: pH observations

A WFD rule for classifying pH observations looks as follows: The pH as individual value should be between 4.5 and 9.0 [83]. Potential sources of pollution from which pH discharges could arise include: agricultural fertilizers, farm wastes and silage, effluent discharges from sewage treatment works, fish farming, organic waste recycling to land, soil cultivation and urban storm water discharges [79].

A simple C-SPARQL query to filter out incoming pH observations, i.e. pH RDF streams, is described below:

```
1  REGISTER STREAM IndObservations AS
2  PREFIX inws: <http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#>
3  PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
4  PREFIX dul: <http://www.loa-cnr.it/ontologies/DUL.owl#>
5  SELECT ?qo ?loc ?dv
6  FROM STREAM <http://inwatersense.uni-pr.edu/stream> [RANGE 10s STEP
   10s]
7  WHERE {
8  ?o ssn:qualityOfObservation ?qo .
9  ?o ssn:observationResult ?r .
10 ?r ssn:hasValue ?v .
11 ?v dul:hasDataValue ?dv .
12 ?o inws:observationResultLocation ?loc
13 FILTER (?qo = inws:pH)
14 }
```

The query name is registered on line 1 and prefixes used in the query are declared on lines 2, 3 and 4. The query runs against the input RDF streams in the time frame of 10 seconds, sliding the window by 10 seconds (line 6). The chosen time frame is arbitrary and can be changed as desired. It produces triples of values (line 5): the water quality name (`?qo`), the location of measurements (`?loc`) and the observation value (`?dv`). Based on the INWS metadata descriptions the incoming observation's (`?o`) water quality name is saved on variable `?qo` (line 8). To get the observation's value, `?o` individuals are bound with individuals `?r` through `ssn:observationResult` property (line 9). These ones in turn are related with individuals of class `ssn:ObservationValue` (line 10), which are finally related with the data value `?dv` through `dul:hasDataValue` property (line 11). The location of observations `?loc` is get through `inws:observationResultLocation` property. Finally, the list of observations is filtered out to include only pH observations (line 13).

Output query results, i.e. (?qo, ?loc, ?dv) triples, are consumed by Jess Tab functions for asserting new facts into the knowledge base. make-instance and slot-insert$ functions are used for creating new class individuals and inserting property values respectively. Namely, for every outputted triple, a new observation instance of the temporary class tmpObservation (a subclass of the ssn:Observation class) is created. tmpObservation holds the most current observation data which are retracted after StreamJess rules process them. Moreover, after retraction they are archived in the ssn:Observation class in the form of historical data. The newly created observation individual is further related with ?qo, ?loc and ?dv values based on the structure of the SSN and INWS metadata descriptions. The water quality name ?qo i.e. pH, becomes related with the new observation instance through the ssn:qualityOfObservation data property. The new observation instance also becomes related with ?loc through observationResultLocation object property. The location instance is of type Point of the basic geo location vocabulary, which means that it possesses longitude and latitude properties. A new ssn:SensorOutput individual is also created for holding the observed value ?dv. It is linked with the observation instance through ssn:observationResult property. Meanwhile, a new instance of class ssn:ObservationValue is created to be related with the previously created ssn:SensorOutput individual through ssn:hasValue property. The ?dv value is assigned to it through dul:hasDataValue data property.

To implement the scenario of this example a monitoring rule was designated for deciding the pH status and another one for identifying the eventual sources of pollution. The monitoring rule looks like follows (ontologies' full IRI are omitted for brevity):

```
1  (defrule classifyPHObsValues
2  (declare (salience 54))
3  (object (is-a ssn#ObservationValue)
4  (OBJECT ?ov)(DUL.owl#hasDataValue ?x))
5  (object (is-a ssn#SensorOutput)
6  (OBJECT ?so)(ssn#hasValue ?ov))
7  (object (is-a time#Instant)
8  (OBJECT ?ot)(time#inXSDDateTime ?time))
9  (object (is-a inws-core.owl#tmpObservation)
10 (OBJECT ?o)(ssn#observationResult ?so)
11 (inws-core.owl#observationResultLocation
   ?loc)(ssn#observationResultTime ?ot)
12 (ssn#qualityOfObservation ?qo&:(eq (instance-name ?qo) inws-
   core.owl#pH)))
```

```
13 =>
14 (bind ?*r* (random))
15 (printout t "(StreamJess)")
16 (if (and (> ?x 4.5) (< ?x 9))then (and
17 (printout t "(" ?*r* ") pH status is GOOD/HIGH" crlf "On: " ?time crlf
   "In: " (instance-name ?loc) crlf)
18 (make-instance (str-cat "GoodHighPHStatus" ?*r*) of inws-
   regulations.owl#GoodHighPHMeasurement map)
19 (slot-insert$ (str-cat "GoodHighPHStatus" ?*r*)
   inws-core.owl#observationResultLocation 1 ?loc)
20 (slot-insert$ (str-cat "GoodHighPHStatus" ?*r*)
   ssn#observationResultTime 1 ot)
21 (slot-set ?loc                                  inws-
   regulations.owl#isPolluted FALSE))
22 else (and
23 (printout t "(" ?*r* ") pH status is MODERATE" crlf "On: " ?time crlf
   "In: " (instance-name ?loc) crlf)
24 (make-instance (str-cat "ModeratePHStatus" ?*r*) of inws-
   regulations.owl#tmpModeratePHMeasurement map)
25 (slot-insert$ (str-cat "ModeratePHStatus" ?*r*) inws-
   core.owl#observationResultLocation  1 ?loc)
26 (slot-insert$ (str-cat "ModeratePHStatus" ?*r)
   ssn#observationResultTime 1 ?ot)
27 (slot-set ?loc                                  inws-
   regulations.owl#isPolluted TRUE)))
28 (make-instance (str-cat (instance-name ?o) ?*r*) of ssn#Observation
   map)
29 (slot-insert$ (str-cat (instance-name ?o) ?*r*) inws-
   core.owl#observationResultLocation 1 ?loc)
30 (slot-insert$ (str-cat (instance-name ?o) ?*r*) ssn#observationResult 1
   ?so)
31 (slot-insert$ (str-cat (instance-name ?o) ?*r*)
   ssn#observationResultTime 1 ?ot)
32 (slot-insert$ (str-cat (instance-name ?o) ?*r*)
   ssn#qualityOfObservation 1 inws-core.owl#pH)
33 (unmake-instance ?o))
```

The first line serves for declaring rule's definition and asserting its name. The second one is for declaring the rule priority. The left hand side (LHS) of the rule (lines 3-12) matches all pH observation individuals (`?o`) present in the `tmpObservation` class. The right hand side (RHS) of the rule (lines 14-31) asks if the matched observation value (`?x`) falls between the interval of values 4.5 and 9. If so, the observation is classified in "good/high" status (lines 16-21), otherwise it becomes "moderate" (lines 22-27). After the classification takes place the

observation individual is stored in the `ssn:Observation` class (lines 28-32) and the temporary observation individual (`?o`) gets retracted from the knowledge base (line 33).

Concretely, for each matched individual from temporary observation class `?o`, on the RHS a new random value is generated to be used for new individual names (line 14). An information string is printed out in the console to indicate that the upcoming outputs are processed by StreamJess rules (line 15). The code in line 16 asks whether the observation value `?x` falls between the allowed values for "good/high" status. If so, the user gets informed about the status detected at measurement site `?loc` on time `?time`. Next, a new individual of `GoodHighPHMeasurement` class gets created (line 18) and related with the location (line 19) and time (line 20) of measurement. Moreover, the pollution status of the measurement site `?loc` is modified to "clean" by changing its `isPolluted` value to "false" (line 21). `str-cat` command is used to concatenate strings. If the if condition specified on line 16 fails then the actions for specifying "moderate" status are activated. The steps to do this are analogical to the ones used for specifying "good/high" status. Namely, before setting the status of the measurement site as "polluted" (line 27) the new status instance is created to be of type `tmpModeratePHMeasurement` (line 24). These instances are temporary because the investigation rule to find potential pH sources of pollution will make use of them and after that will delete them. Prior to deletion the status instance is stored as a new instance of `ModeratePHMeasurement` class as historical data (lines 28-32) copying all `?o` properties. The retraction is performed for preventing investigations to be activated only once. The pH investigation rule is described below:

```
1  (defrule findPHsourcesOfPollution
2  (declare (salience 553))
3  (object (is-a epa.owl#MeasurementSite) (OBJECT ?loc) (inws-
   pollutants.owl#hasSourcesOfPollution $?sitepoll))
4  (object (is-a inws-regulations.owl#tmpModeratePHMeasurement) (OBJECT
   ?mob) (inws-core.owl#observationResultLocation ?loc)
   (ssn#observationResultTime ?ot))
5  =>
6  (bind ?*r* (random))
7  (make-instance (str-cat (instance-name ?mob) ?*r*) of inws-
   regulations.owl#ModeratePHMeasurement map)
8  (slot-insert$ (str-cat (instance-name ?mob) ?*r*) inws-
   core.owl#observationResultLocation 1 ?loc)
9  (slot-insert$ (str-cat (instance-name ?mob) ?*r*)
   ssn#observationResultTime 1 ?ot)
10 (foreach ?poll ?sitepoll
```

```
11  (foreach ?pollLsItem (slot-get ?poll inws-
   pollutants.owl#potentialPollutant)
12   (if(eq (instance-name ?pollLsItem) inws-core.owl#pH) then
13    (printout t "pH pollution source: " (instance-name ?poll) " crlf)
14    (slot-insert$ (str-cat (instance-name ?mob) ?*r*) inws-
   regulations.owl#foundPollutionSources 1 (instance-name ?poll)))))
15 (unmake-instance ?mob))
```

The rule binds the temporary "moderate" status pH observations into `?mob` variable and gets its location `?loc` and time `?ot` (line 4). The code in line 3 relates the list of sources of pollution present on the measurement site `?loc` into the list variable `$?sitepoll`. The RHS of the rule starts with archiving the temporary status instance `?mob` (lines 6-9). Namely, in absence of a Jess or Jess Tab mechanism to change the instance class assignment, the temporary status instance is copied in a new instance of class `ModeratePHMeasurement`. Afterwards, the list members of `?sitepoll` is iterated (line 10) to match only those sources of pollution which could increase pH discharges (lines 11-12). Namely, for each source of pollution in `?sitepoll` i.e. present on the measurement site, its potential pollutants list `?pollLsItem` is checked if it includes pH. The matching one's name will be printed out (line 13). As per saving historical data the archived status instance gets related with the list of pollution sources through `foundPollutionSources` property (line 14). Finally, the temporary status instance `?mob` gets discarded from the knowledge base. An example output of *Example 1* is illustrated in Figure 16. As can be observed, C-SPARQL query `IndObservations` has produced three output results. Two of these results (#1 and #3) have been classified with "good/high" status by rule `classifyPHObsValues`, while the remaining one (#2) with "moderate" status. Since the result #2 has been classified as a critical status the investigation rule `findPHsourcesOfPollution` has fired and identified that potential source of the pollution is "urban storm water discharges" on site `ms11`.

```
+++++++ 3 new C-SPARQL result(s) at SystemTime=[1463693318872] +
#1 (C-SPARQL) Observed WQ: pH Value: 4.62954251429808
(StreamJess)(39844) pH status is GOOD/HIGH
On: Thu May 19 23:28:38 CEST 2016
In: http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#ms12
#2 (C-SPARQL) Observed WQ: pH Value: 14.484745669988683
(StreamJess)(3501) pH status is MODERATE
On: Thu May 19 23:28:39 CEST 2016
In: http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#ms11
pH pollution source: Urban stormwater discharges
#3 (C-SPARQL) Observed WQ: pH Value: 7.0182989018469275
(StreamJess)(42061) pH status is GOOD/HIGH
On: Thu May 19 23:28:39 CEST 2016
In: http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#ms12
```

Figure 16. An output excerpt of the running Example 1

IV. 2. 2 Example 2: Biochemical Oxygen Demand (BOD_5) observations

A WFD rule for classifying - BOD_5 observations is as follows: If BOD_5 measurements in mg O_2/l is less than 1.3 (mean), then river belongs to "high" status of oxygen condition; if it is less than 1.5 then river belongs to "good" status of oxygen condition; otherwise the river belongs to "moderate" status of oxygen condition [83]. Potential sources of pollution from which BOD_5 discharges could arise include: contaminated land, farm wastes and silage, fish farming, effluent discharges from sewage treatment works, landfill sites and urban storm water discharges [79].

The C-SPARQL query to calculate the water quality observed average value on each window is given below:

```
1  REGISTER STREAM AvgObservations AS
2  PREFIX inws: <http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#>
3  PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
4  PREFIX dul: <http://www.loa-cnr.it/ontologies/DUL.owl#>
5  SELECT ?qo ?loc (AVG(?dv) AS ?avg)
6  FROM STREAM <http://inwatersense.uni-pr.edu/stream> [RANGE 20s STEP
   20s]
7  WHERE {
8  ?o ssn:qualityOfObservation ?qo .
9  ?o ssn:observationResult ?r .
10 ?r ssn:hasValue ?v .
11 ?v dul:hasDataValue ?dv .
12 ?o inws:observationResultLocation ?loc
13 FILTER (?qo != inws:pH)
14 }
15 GROUP BY ?qo ?loc
```

This query is similar to `IndObservations` query described previously. The lines 2-4 and 6-12 are the same. As opposed to it, this query filters the RDF streams to include all but those of pH observations (line 13). Moreover, it uses aggregate functions such as AVG (line 5) to calculate the average value of observations which are firstly grouped by water quality name and then by location of measurement (line 15). It is arbitrary set to run every 20 seconds sliding the window by 20 seconds (line 6).

Similar to *Example 1*, the output query results, i.e. (`?qo, ?loc, ?avg`) triples, are consumed by Jess Tab functions to create new `tmpObservation` instances. Afterwards, the following monitoring rule, similar to `classifyPHObsValues`, classifies BOD_5 observations:

```
1 (defrule classifyBOD5ObsValues
...
12(ssn#qualityOfObservation ?qo&:(eq   (instance-name ?qo)(instance-name
  inws-core.owl#BOD))))
=>
...
16(if (and (< ?x 1.5) (> ?x 1.3)) then (and
17(printout t "(" ?*r* ") BOD status is GOOD" crlf "On: " ?time crlf "In:
  " (instance-name ?loc) crlf)
18(make-instance (str-cat "GoodBODStatus" ?*r*) of inws-
  regulations.owl#GoodBODMeasurement map)
19(slot-insert$ (str-cat "GoodBODStatus" ?*r*) inws-
  core.owl#observationResultLocation 1 ?loc)
 ...
20(slot-insert$ (str-cat "GoodBODStatus" ?*r*) ssn#observationResultTime
  1 ?ot)
...
22(if (< ?x 1.3) then <HIGH status classification code here>)
23(if (> ?x 1.5) then <MODERATE status classification code here>))
```

Similarly to the rule `classifyPHObsValues` lines 3-12 bind BOD_5 observation data present in the temporary class `tmpObservation` with their corresponding variables. Lines 16-20 encode the semantics of the expression "if it is less than 1.5 then river belongs to "good" status of oxygen condition" from the example statement. Classification of water bodies into "high" (line 22) and "moderate" (line 23) status is omitted because it's analogical with lines 16-20 with the appropriate change on the name of the status, the corresponding class name and the setting of the pollution status of the site.

The streams processed by the C-SPARQL query `AvgObservations` will result in zero or many BOD5 observations. The number of BOD_5 observations will depend on the number of measurement sites. For example, as illustrated in Figure 17, C-SPARQL processing of RDF streams has resulted with 3 new observations on 3 measurement sites: `ms10`, `ms11` and `ms12`. Two observations have been classified as of "moderate" status (Figure 17 line #1 and #2) and one of "high" status (Figure 17 #3).

Whenever a critical i.e. "moderate" BOD_5 measurement is detected the following investigation rule to detect BOD_5 potential sources of pollution is activated:

```
1 (defrule findBOD5SourcesOfPollution
...
=>
...
```

```
7  (make-instance (str-cat (instance-name ?mob) ?*r*) of inws-
   regulations.owl#ModerateBODMeasurement map)
...
12 (if(eq (instance-name ?pollLsItem) inws-core.owl#BOD) then
13 (printout t "BOD pollution source: " (instance-name ?poll) crlf)
...
```

Similar to `findPHsourcesOfPollution` rule, this one will cause the Rete engine to detect newly asserted individuals of `tmpModerateBODMeasurement` on a specified measurement site. In fact, the LHS of the rules is the same. It will get the sources of pollution on that site which in turn are filtered out to include only BOD_5 potential pollutants (lines 9-13). Each of the matched sources of pollution will be printed out in the console as shown in observation #1 and #2 in Figure 17. Namely, a "moderate" BOD_5 status is detected on sites `ms11` and `ms10`. Potential sources of pollution include urban storm water discharges and fish farming on site `ms11` while urban storm water is potential source of BOD_5 discharges on site `ms10`.

```
++++ 3 new C-SPARQL result(s) at SystemTime=[1462808878402] ++++
#1 (C-SPARQL) Observed WQ: BiochemicalOxygenDemand Value: 1.6622
(StreamJess)(42693) BOD status is MODERATE
On: Mon May 09 17:47:58 CEST 2016
In: http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#ms11
BOD5 pollution source: Urban stormwater discharges
BOD5 pollution source: Fish farming
#2 (C-SPARQL) Observed WQ: BiochemicalOxygenDemand Value: 1.9215
(StreamJess)(42181) BOD status is MODERATE
On: Mon May 09 17:47:58 CEST 2016
In: http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#ms10
BOD5 pollution source: Urban stormwater discharges
#3 (C-SPARQL) Observed WQ: BiochemicalOxygenDemand Value: 0.6389
(StreamJess)(50374) BOD status is HIGH
On: Mon May 09 17:47:58 CEST 2016
In: http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#ms12
```

Figure 17. An output excerpt of the running Example 2

IV. 2. 3 Example 3: The 'undetermined status'

In StreamJess, the feature of NAF is enabled in the stream processing level. Recall the Section 2.4.1 example of assigning the 'undetermined status' to measurement sites to which the data are missing. The SPARQL support for NAF was utilized as described in the following query.

```
REGISTER STREAM undefinedMeasurmentSites  AS
PREFIX inwsp: <http://inwatersense.uni-pr.edu/ontologies/inws-
pollutants.owl#>
```

```
PREFIX rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX twcc: <http://tw2.tw.rpi.edu/zhengj3/owl/epa.owl#>
SELECT ?ms
FROM STREAM <http://inwatersense.uni-pr.edu/stream> [RANGE 60s STEP 60s]
FROM <http://inwatersense.uni-pr.edu/ontologies/data.rdf>
WHERE {
?ms rdf:type twcc:MeasurementSite
OPTIONAL { ?ms inwsp:isPolluted ?tf } .
FILTER (!BOUND(?tf))
}
```

After (C-SPARQL) processing and (Jess) reasoning on each observation instance a measurement site will be related with a "true" or "false" value through `isPolluted` property. This query will match the remaining measurement sites, present in the background knowledge base (`data.rdf` file), for which no pollution status is recorded. The query is arbitrarily set to run every minute. On each query output result the matching measurement sites' `isPolluted` status is set to 'undefined' through Jess Tab `make-instance` construct.

Chapter V C-SWRL

The previous chapter described how production rules can be enabled to reason over stream data. Here will be described C-SWRL, a unified Semantic Web approach for rule-based reasoning over stream data. Similarly to StreamJess, it complements state of the art query processing engine C-SPARQL [93] with the W3C recommended Semantic Web rule language SWRL. The chapter is organized as follows. Section 5.1 describes the prototypical design and implementation. System validation is presented in Section 5.2 through examples in the domain of WQM. Section 5.3 describes the challenges encountered while building C-SWRL.

V. 1 System design and implementation

C-SWRL conceptual architecture is the same as StreamJess's one depicted in Figure 13. However, it uses SWRL rules to reason over stream data. At the system design phase three approaches were considered:

- Extending SWRL with stream data reasoning features,
- Translating SWRL to another rule system which supports stream data reasoning and
- Layering SWRL on top of another system to fill the gaps of SWRL in support of stream data reasoning.

Extending SWRL with stream data reasoning features is very expensive since none of the required reasoning features described in Section 1. 2. State-of-the-art SWRL extensions may support one, but fail on another feature. For example, JNOMO [85] is a SWRL extension for enabling non-monotonic reasoning, but it does not support time-aware reasoning. JNOMO [85] is also an example of translating SWRL into Jena [12]. Moreover, an intelligent tutoring system framework introduced in [73] and SweetJess [107] represent further examples of rules interoperability systems with translating SWRL and RuleML respectively into Jess rules

[107]. These implementations do not deal with the different nature of stream data and they also have the potential of losing information while translating the constructs. Given the drawbacks if approaching any of the previous two options, it was decided to layer SWRL over an existing SR system such as C-SPARQL. C-SPARQL is specifically designed for stream data applications. It supports closed-world and time-aware reasoning on stream data. However, as a query language, it is not intended to have any effect on the underlying ontology. Analogical to StreamJess, C-SWRL uses C-SPARQL output RDF streams as input for SWRL, instead of Jess, to infer and assert new knowledge to ontologies. Firstly, input RDF streams filtering and aggregation is done by C-SPARQL. Secondly, based on C-SPARQL output RDF streams, OWLAPI [173] constructs are invoked for asserting new OWL individuals in a temporary class holding all observation's information. Finally, these individuals are processed by SWRL continuous rules loaded at application startup which include monitoring and investigation rules, defined in the previous chapter. Recall that, the information of sources of pollution stored into the pollutants ontology is used to prejudge the causer of the pollution.

SWRLAPI [89] methods are called for doing SWRL rule-based reasoning. SWRL inference occurs at each window processing. Namely, monitoring rules detect the temporary observation data and classify the observation into appropriate status based on WFD standards e.g. good, high or moderate. Whenever a moderate status is detected the investigation rules fire to assert the polluted site and potential sources of pollution. Since this process is continuous and iterative, to avoid reasserting of individuals into appropriate classes, the temporary observation class needs to be cleared at each window processing. This was done by using the OWLAPI's `removeAxiom` construct. The same construct was used to enable system's non-monotonic behavior. Namely, SWRL's ability to assert new information in conjunction with OWLAPI's one to remove information enables the modification of the measurement site's pollution status. At each window processing, which processes an observation on a particular measurement site, the last known pollution status gets removed from the knowledge base (by OWLAPI constructs) and a new status is inferred based on the SWRL rules. In particular, this was managed through the object property `isPolluted` relating measurement sites with one of the instances true or false. Thus, one can query for measurement sites' state at any time of C-SWRL running application. Moreover, every time a measurement site gets polluted a new instance of the class `PollutedSite` is asserted related with time and pollutants information.

C-SWRL is implemented in Java following the availability of Java codes of C-SPARQL, OWLAPI and SWRLAPI. The system is open for loading different SSN-based domain ontologies, write appropriate C-SPARQL queries and SWRL rules. Moreover, instead of C-SPARQL and SWRL, with less effort different SPARQL-like query processing engines coupled with different rule languages can be integrated, respectively. C-SWRL is open source software and its installation details can be found on Appendix D.

V. 2 System validation

As with validation of StreamJess, BOD_5 and pH observations will be used to validate the prototype of C-SWRL. The same validation settings configured for StreamJess, described in Section 3.2 were also applied for validation of C-SWRL. Figure 18 illustrates a screenshot of the C-SWRL console output of the running examples 1 and 2.

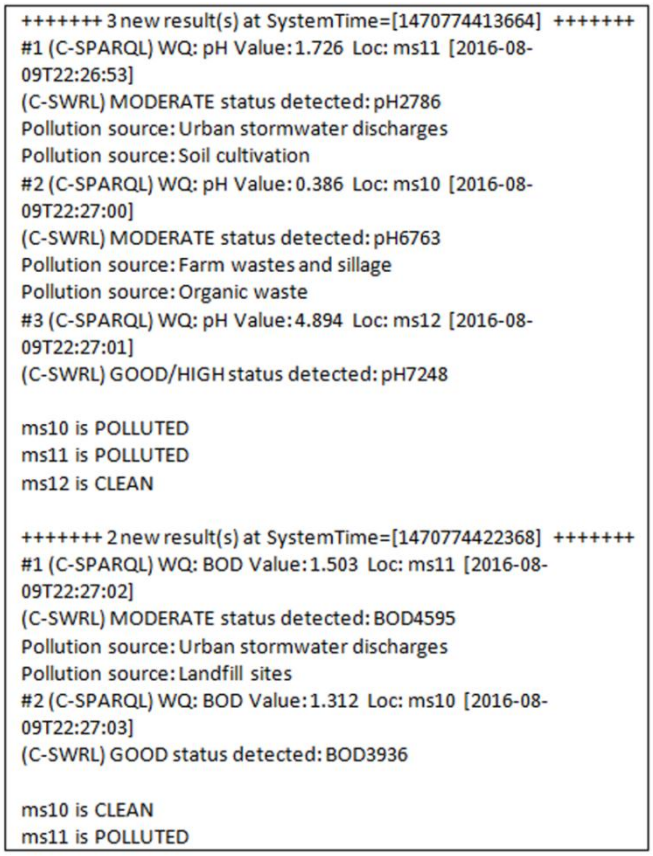


Figure 18. An output excerpt of the running examples 1 and 2 on C-SWRL

V. 2. 1 Example 1: BOD_5 classification

Recall Section 3.2.2 which describes the WFD rule for classification of BOD_5 observations. The same query, `AvgObservations`, output triples are used to populate corresponding ontology classes. Namely, at every query execution, for each new triple `(?qo, ?loc, ?avg)`, a new individual of a temporary INWS class `tmpObservation` is asserted into the ontology using OWLAPI constructs. This individual indicates a new input observation has arrived. Following the INWS ontology design this individual is associated through:

- `ssn:qualityOfObservation` with the water quality parameter name i.e. `?qo`,
- `observationResultLocation` property with `?loc`,
- `ssn:observationResult` with new `ssn:SensorOutput` instance, which in turn is related with a new `ssn:ObservationValue` instance through `ssn:hasValue` property, which finally is associated with the observation's average value `?avg` through `dul:hasDataValue`.
- `ssn:observationResultTime` with the system's timestamp

Next, the SWRL rule engine is executed firing the registered SWRL monitoring rules. These rules include the following ones for BOD_5 WFD classification (user-defined prefixes are omitted for brevity):

```
1. tmpObservation (?x) ^ qualityOfObservation(?x,BiochemicalOxygenDemand)
^ observationResult(?x, ?y) ^ hasValue(?y, ?e) ^ hasDataValue(?e,?z) ^
swrlb:greaterThan(?z, 1.3) ^ swrlb:lessThan(?z, 1.5) ->
GoodBODMeasurement(?x) ^ tmpGoodBODMeasurement(?x) ^ isPolluted(?ms,
false) ^ Observation(?x)
2. tmpObservation (?x) ^ qualityOfObservation(?x,BiochemicalOxygenDemand)
^ observationResult(?x, ?y) ^ hasValue(?y, ?e) ^ hasDataValue(?e,?z) ^
swrlb:lessThan(?z, 1.3) ->  HighBODMeasurement(?x) ^
tmpHighBODMeasurement(?x) ^ isPolluted(?ms, false) ^ Observation(?x)
3. tmpObservation (?x) ^ qualityOfObservation(?x, BiochemicalOxygenDemand)
^ observationResult(?x, ?y) ^ hasValue(?y, ?e) ^ hasDataValue(?e,?z) ^
swrlb:greaterThan(?z, 1.5) -> ModerateBODMeasurement(?x)
^tmpModerateBODMeasurement(?x) ^ isPolluted(?ms, true) ^ Observation(?x)
```

The first rule matches the individuals (`?x`) of the temporary class related to BOD_5 measurements and checks its average value. If it is between 1.3 and 1.5 then the status is "good" i.e. the individual is asserted as of type `GoodBODMeasurement`. The same

matching is done with the second and third rule respectively. For the second one the average value is checked to be lower than 1.3 for its classification. If so, the status is "high" i.e. the individual is asserted as of type `HighBODMeasurement`. In the third rule the average value is checked to be greater than 1.5 for classifying in "moderate" status i.e. class `ModerateBODMeasurement`. A temporary class `tmpModerateBODMeasurement` is used for investigation of sources of pollution. In the first and second rule the respective temporary classes are used for displaying the calculated status to the user interface. In each RHS of the rules the temporary observation individual gets stored in the class `Observation` as per historical data records. Moreover, the `isPolluted` object property is used to maintain the current state of the measurement site. It is set to 'false' in the cases of "good" and "high" statuses while it is set to 'true' when detecting "moderate" status. In the running example the firing of rules has produced one "moderate" and one "good" status, as illustrated in the lower part of Figure 17 i.e. the lines starting with (C-SWRL) label followed by the detected status information. Since, the first C-SPARQL calculated average value is 1.503 which is greater than 1.5 the third rule has fired asserting new individuals in `ModerateBODMeasurement` and `tmpModerateBODMeasurement`.

New individual in the class `tmpModerateBODMeasurement` will cause to fire the following investigation rule, which is also registered at application startup:

```
4.tmpModerateBODMeasurement(?x) ^ observationResultTime(?x, ?t) ^
observationResultLocation(?x, ?ms) ^ hasSourcesOfPollution(?ms,
?pollsrc) ^ potentialPollutant(?pollsrc, BiochemicalOxygenDemand) -
> foundPollutionSources(?x, ?pollsrc)
```

This rule binds the "moderate" status observations (`?x`) with measurement site's (`?ms`) nearby BOD_5 sources of pollution (`?pollsrc`) extracted from the knowledge base. The observations (`?x`) satisfying the LHS clauses will become related with the matching pollution sources. These results will be displayed to the user interface right after the "moderate" status detection like is shown on the first C-SPARQL result in the lower part of Figure 17. It can be observed from the figure that the potential sources of pollution caused on `ms11` are "urban storm water discharges" and "landfill sites".

At the end of each window processing and reasoning, the current status of the sites are queried and printed out. On Figure 17, the last statuses for measurement sites `ms10` and `ms11` are "clean" and "polluted", respectively

V. 2. 2 Example 2: pH classifications

The query to filter individual pH measurements is much simpler than the BOD_5 one (`IndObservations` query from Section 3.2.1). No aggregation function is used in the SELECT statement and thus no grouping is needed. The FILTER clause uses the equal symbol rather than the unequal one. pH observations monitoring rules are similar to the ones (1-3) for BOD_5. The main difference is the need for expressing disjunction in the body of the rules for classification of "moderate" statuses. Namely, two rules are used to encapsulate each of the interval values $(-\infty, 4.5)$ and $(9, +\infty)$. Similarly to the investigation rule for identifying BOD_5 sources of pollution, pH investigation rule uses `tmpModeratePHMeasurement` previously asserted individuals to identify the potential sources of pollution in the polluted site.

An Example 2 output excerpt is depicted in the upper part of Figure 18. Two "moderate" statuses have been detected on `ms10` and `ms11` and the corresponding potential sources of pollution have been identified, while a "good/high" status is detected on `ms12`. A summary of the latest status of each observed measurement site is printed out at the end of the processed window.

V. 3 Discussion and challenges

Following are described the challenges that appeared while building C-SWRL.

V. 3. 1 Fact modification and retraction

SWRL's inability to modify or retract the facts from the knowledge base in C-SWRL was resolved with the help of OWLAPI construct `removeAxiom`. In absence of a dedicated OWLAPI construct for modifying ABox the technique "remove and assert" was used. Thus, the issue of modifying the measurement site's pollution status was managed by firstly removing its previously asserted status and then asserting the new one through firing of SWRL rules.

V. 3. 2 Aggregates

Following the SWRL's OWA, in C-SWRL, aggregate operations over stream data are done by C-SPARQL. Query results are next deployed into the ontology through OWLAPI add axiom construct, which will further trigger the firing of the matching rules.

V. 3. 3 Negation as Failure

NAF feature is enabled in the stream processing level. Recall Section 2 example of assigning the 'undetermined status' to measurement sites to which the data are missing. The SPARQL support for NAF was utilized as described in the following query.

```
REGISTER STREAM undefinedMeasurmentSites  AS
PREFIX inwsp: <http://inwatersense.uni-pr.edu/ontologies/inws-
pollutants.owl#>
PREFIX rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX twcc: <http://tw2.tw.rpi.edu/zhengj3/owl/epa.owl#>
SELECT ?ms
FROM STREAM <http://inwatersense.uni-pr.edu/stream> [RANGE 60s STEP 60s]
FROM <http://inwatersense.uni-pr.edu/ontologies/data.rdf>
WHERE {
?ms rdf:type twcc:MeasurementSite
OPTIONAL { ?ms inwsp:isPolluted ?tf } .
FILTER (!BOUND(?tf))
}
```

A measurement site will be related with a "true" or "false" value through `isPolluted` property after (C-SPARQL) processing and (SWRL) perform reasoning on each observation instance. This query will match the remaining measurement sites, present in the background knowledge base (`data.rdf` file), for which no pollution status is recorded. The query is arbitrarily set to run every minute. On each query output result the matching measurement sites' `isPolluted` status is set to 'undefined' through OWLAPI add axiom construct.

V. 3. 4 Continuous rule feature

Continuous rule feature in C-SWRL was implemented with the help of C-SPARQL's time or tuple-based windows. The rule engine gets activated after each arrival of new query results. Similarly to C-SPARQL, the ideal solution would be to adapt the window feature on SWRL.

This approach encapsulates stream data processing and reasoning at the same time. Filtering data streams may be easily realized through temporal built-ins such as SWRLTemporalBuiltInLibrary[13], but the aggregate functions are hardly implementable in SWRL following its OWA.

[13] https://github.com/protegeproject/swrlapi/wiki/SWRLTemporalBuiltInLibrary

Chapter VI Related Works

This chapter elaborates the related works of the main contributions of this thesis, namely the INWS ontology, StreamJess and C-SWRL. The following subsections will explicitly describe the current state-of-the-art developments as compared to our approaches.

VI. 1 State of the art Ontologies for WQM

A large number of WQM systems have been developed in the last decades. One of the first WQM systems that has benefited from the ontological knowledge representation is OntoWEDDS [9]. The inclusion of ontological reasoning alongside case-based and rule-based reasoning has resulted with significant improvement. In the rest of this section identify some of the current WQM systems as compared to our approach.

In order to provide a portal for WQM, Tetherless World Constellation (TWC)[14] has developed Semantic Water Quality Portal[15] (TWC-SWQP) described in [90]. They are pioneers for including regulations ontology. However, their approach is very basic since it only finds the excessive threshold measurements and classifies the polluted data sources. This ontology was reused and eventually extended for supporting regulations standards we are interested in. WFD regulations for example are more specific by specifying different quality statuses (high, good, moderate, poor or bad) based on the category of the water quality element (biological, physic-chemical, hydro morphological). Another issue is the core ontology. TWC-SWQP core ontology is not completely suitable for our purpose. For example, it does not model sensors. However, some of TWC-SWQP core ontology concepts was reused, e.g. `MeasurementSite` and `WaterMeasurement` while from the regulations ontology the concepts like `PollutedFacility` and `PollutedSite`.

[14] TWC, http://tw.rpi.edu/web/TWC
[15] TWC-SWQP, http://aquarius.tw.rpi.edu/projects/semantaqua/

Another distinction from our approach is the OWL2 classification inference used in TWC-SWQP. Instead, SWRL rules will be used in conjunction with OWL restrictions to support regulations features.

An ontology which models sensors is the SSN ontology. This ontology is the main building block of our core ontology. It was extended with some other ontologies to fulfill the system's requirements. An earlier version of this ontology has been used by Taylor and Ledinger in [16] for designing an ontology-based complex event processing system in the field of heterogeneous sensor networks. Complex Event Processing (CEP) represents an area dealing with timely detection of events inferred from complex correlations of stream values. In [16] authors translate the event ontology into CEP statements for processing of events. Another CEP approach has been taken by Anicic et al. [14] who combine the reasoning power of Semantic Web with real-time detection of events affinity of CEP. Opposed to CEP approaches our tendency is to build a pure Semantic Web approach by relying on Semantic Web standards and recommendations such as OWL and SWRL. In our previous work [3] it is stated the aware of the challenges appearing from the likes of open world and monotonicity semantics. CEP systems described in [14, 16] are implemented in Prolog, which is a Logic Programming language. This implies that CEP adopts the closed world assumption and non-monotonic reasoning. But the question is, are we confident on preferring one over the other i.e. open over closed world assumption or monotonic over non-monotonic reasoning or one should support both opposite "worlds". For example, if none of the observed quality elements has passed a threshold in closed world one may end up with a conclusion that the water body is healthy but in terms of open world one cannot infer this. There may still be any other condition which will probably classify the water body as polluted.

VI. 2 StreamJess related works

Two main strategies exist for systems combining ontologies with rules: hybrid and homogeneous approaches [3, 49]. In the former one, also called loosely-coupled approach, the reasoning is done by interfacing existing rule reasoner with existing ontology reasoner, while in the latter one, also called tightly-coupled approach, both ontologies and rules are embedded into the same logical language without making a priori distinction between the rule predicates and the ontology predicates [49].

VI. 2. 1 Hybrid approaches

Hybrid approaches layer different non-DL rule systems on top of DL ontologies like: production rules, CEP, LP, answer set programming (ASP), etc. In the literature this approach is also referred to as, integration of ontologies and rules with strict semantic separation [49] In our previous work [3], was described in more detail about each one of these approaches and their pros and cons. In general, hybrid solutions have achieved the desired system behavior while main drawbacks include: translation and reasoner issues and side-effects occurrence.

The first approaches combining ontologies with production rules are described in [49, 13]. Sottara et al. (2012) model a hybrid Environmental Decision Support System (EDSS) for Waste-Water Treatment Plants (WWTP). As an application of production rules they infer invalid NO_3 measurement values. They argue that the WWTP domain should be modeled through ontologies, for modeling sensor data, paired with decision-making rules, for processing incoming sensor data and recommending actions to be taken. Another system implemented in terms of production rules has been designed by Chau (2007) in the domain of water quality modeling. Namely, the system simulates human expertise during the problem solving of coastal hydraulic and transport processes. Both forward-chaining and backward-chaining are used collectively during the inference process Chau (2007). Even though that these approaches, together with our previous work [3] argue that pairing ontologies with production rules provides a fruitful solution, they do not make any distinction between stream and static data. As such, they do not implement the window feature.

StreamRule [34] represents the pioneer of coupling stream processing systems with ASP non-monotonic reasoning. Even though the approach is still much more prototypical it demonstrates how non-monotonic and time-aware reasoning can be integrated into a unique platform for stream data reasoning. Similarly to our approach, the continuous rule feature is implemented through separate steps. Namely, stream filtering and aggregation is done through a stream query processor such as CQELS [87], while OClingo [88] is used to enable non-monotonic reasoning. In StreamJess C-SPARQL was used for filtering and aggregation purposes, while non-monotonic reasoning is achieved through Jess rules and Jess Tab functions. Even though that CQELS outperforms C-SPARQL [92], C-SPARQL was preferred to follow its advantage to use nested aggregations and negation [97, 92]. Moreover, temporal operators are planned to be supported, which lack any support in CQELS [97]. The

main distinction of the stream reasoning component between StreamRule and StreamJess fall on the strategy of the inference process. Namely, OClingo, as LP-based approach, follows the backward chaining approach. It means to start from the conclusion of the rule and try to match the facts of the rule's condition part. Jess uses the Rete algorithm [70] to do fast pattern matching, which is natively forward chaining strategy. Even though the algorithm is ideally suited for complex event detection, it does not support temporal reasoning [13]. Moreover, it saves the states between cycles, which is not preferred in situations when most of the data change. However, its extensions are in place to support stream reasoning e.g. [13], [81] and [43]. Jess also supports backward chaining, which is effectively simulated in terms of forward chaining rules [109]. The forward chaining technique starts from the rule's condition part and finds the facts satisfying the rule's conclusion. Both approaches have their pros and cons: backward chaining is more memory efficient while forward chaining is faster but consumes more memory [62]. Jess was decided to be used because of the ability to use both strategies. Regarding the implemented features StreamRule lacks the historical data management component, which is one of the key requirements of SR tools [80]. StreamJess keeps evidence of every previous environment state. For example, one can query the INWS ontology for a particular measurement site's pollution status of the past. OClingo feeds back the reasoning results into Java runtime for further processing or display, while in StreamJess, the results are also deployed back into the knowledge base and thus the memory gets released and the data are available for query and retrieval. This was implemented through the Jess Tab's save-project function, which is called after processing each C-SPARQL window or alternatively be set to run periodically.

Recently, Ali et al. (2016) describe the descendant of StreamRule, which support C-SPARQL aside of CQELS. The system supports reasoning even in incomplete information cases through NAF, but like StreamRule it does not support historical data management. The SPARQL support of NAF was utilized in StreamJess to complement the difficulties for enabling NAF in Jess. Moreover, the reasoning results are returned as a JSON object to the corresponding web socket clients, while in StreamJess the reasoning results are returned as standard RDF data populating corresponding ontology classes. Their stream reasoning component is tested only with small amounts of input data. Our initial experimental results on StreamJess show better performance than the OClingo component implemented in StreamRule and [62] for small inputs, while system's performance evaluation for larger inputs is part of our future works. Jess is memory-intensive application, but recent Java

Virtual Machines include flexible and configurable garbage collection subsystem which is responsible for finding and deleting unused objects [62]. As argued by Hill (2003), the adjustment of two parameters: heap size and the object nursery size, has resulted with an improved 25% better performance.

Rscale [69] is another industrially-approved reasoning system which utilizes OWL 2 RL language profile to infer new knowledge. It enables incremental reasoning, non-monotonic and closed-world reasoning through translation of facts and rules into SQL tables and queries respectively. However, it does not support time-aware reasoning.

ETALIS [14] together with EP-SPARQL [15] enables CEP with stream reasoning. Even though ETALIS offers reasoning on time and location spaces it does not implement the windows feature. Time-based windows are supported through its wrapper EP-SPARQL, but complicated aggregations within windows are not supported [97]. Moreover, there is no support for triple-based windows too.

VI. 2. 2 Semantic Web approaches

In the literature this approach is also referred to as interaction of ontologies and rules with tight semantic integration [49]. Even though the tight coupling of the model the rule language has distinct advantages e.g. no mapping mechanism is required between them, these approaches mainly suffer from limited expressiveness or decidability [49]. Thus, to date, there is not a tight-coupled approach which supports all the stream reasoning requirements. Approaches described by Keßler et al. (2009) and Wei and Barnaghi (2009) do not make any distinction between stream and static data, while also lack implementation. They prove that SWRL can be used to infer new and approximate knowledge in stream data domains. However, their approach does not consider time-aware and non-monotonic reasoning. Recently, a SPARQL extension [42] that uses CONSTRUCT/WHERE clauses to express rules has been proposed. Yet again this approach does not consider non-monotonic reasoning. The works presented in [77] and [78] describe a Rete-based [70] approach of RDFS entailment rules for producing data in a continuous manner. Although supporting time-aware and incremental reasoning, the approach does not deal with non-monotonic and closed-world reasoning. JNOMO [85] shows how SWRL can be extended to embrace non-monotonicity, CWA and NAF. Namely, `NotExist` operator is defined to “close” the world and to enable

fact retraction. However, it does not deal with stream data, while inclusion of temporal reasoning is envisioned as per future works.

VI. 3 C-SWRL related works

As in the previous section, C-SWRL's related works will be divide into two broad categories: hybrid and homogeneous approaches.

VI. 3. 1 Hybrid approaches

Hybrid approaches layer different non-DL rule systems on top of ontologies like: production rules, CEP, LP, answer set programming (ASP), etc. In the literature this approach is also referred to as, integration of ontologies and rules with strict semantic separation [49]. In our previous work [3], is described in more detail about each one of these approaches and their pros and cons. In general, hybrid solutions have achieved the desired system behavior. However, some evident drawbacks are summarized as follows:

- *Translation issues*: In these approaches, the ontology is translated into the corresponding formalisms of the underlying rule system. A drawback of this translation is that a possible loss of information may occur. For example, translating complex sub-class statements consisting of disjunction of classes or expressed with existential quantification are not possible into Plausible Logic [66].
- *Reasoner issues*: Since the ontology and the rules are treated separately then a rule engine and a DL reasoner will run concurrently [14]. As argued in [14], some inferences would no longer be derived after separating OWL and rules.
- *Side-effects occurrence*: When adding a new rule, in some hybrid approaches a possible side-effect may occur. For example, in production rule systems adding a rule may require extra work because of the algorithm used for executing the rules [64]. This makes it harder for domain experts to write rules without IT support. In some cases (as shown in [64]), development layers are conflate to each other making rules maintenance more laborious.

A similar approach to C-SWRL is followed by StreamRule [34], the pioneer of coupling stream processing with ASP non-monotonic reasoning. Even though the approach is still much more prototypical it demonstrates how non-monotonic and time-aware reasoning can

be integrated into a unique platform for stream data reasoning. The continuous rule feature is implemented through separate steps. Namely, stream filtering and aggregation is done through a stream query processor such as CQELS [87], while OClingo [88] is used to enable non-monotonic reasoning. In C-SWRL C-SPARQL is used for filtering and aggregation purposes, and OWLAPI for non-monotonic reasoning. Even though that CQELS outperforms C-SPARQL [92], C-SPARQL was preferred following its advantage to use nested aggregations and negation [97, 92]. Moreover, it is a plan to support temporal operators, which lack any support in CQELS [97]. Another feature difference between StreamRule and C-SWRL is the historical data management, which is one of the key requirements of SR tools [80]. C-SWRL keeps evidence of every previous environment state. For example, one can query the ontology for a particular measurement site's pollution status of the past. OClingo feeds back the reasoning results into Java runtime for further processing or display, while in C-SWRL, the results are deployed back into the knowledge base and thus the memory gets released and the data are available for query and retrieval. This was implemented through the OWLAPI's `saveOntology` function, which is called after processing each C-SPARQL window or can be set periodically.

Recently, Ali et al. (2016) proposed another non-monotonic ASP-based SR system, which provides support for C-SPARQL query engine. The system supports reasoning even in incomplete information cases through NAF feature, but like StreamRule it does not support historical data management. Moreover, the reasoning results are returned as a JSON object to the corresponding web socket clients, while in C-SWRL the reasoning results are returned as standard RDF data populating corresponding ontology classes.

Rscale [69] is another industrially-approved reasoning system which leverages OWL 2 RL language profile to infer new knowledge. It enables incremental reasoning, non-monotonic and closed-world reasoning through translation of facts and rules into SQL tables and queries respectively. However, it does not support time-aware reasoning, and as a non-Semantic Web approach follows the hybrid approach disadvantages.

ETALIS [14] together with EP-SPARQL [15] enables CEP with stream reasoning. Even though ETALIS offers reasoning on time and location spaces it does not implement the windows feature. Time-based windows are supported through its wrapper EP-SPARQL, but complicated aggregations within windows are not supported [97]. Moreover, there is no support for triple-based windows too.

VI. 3. 2 Semantic Web approaches

In the literature this approach is also referred to as interaction of ontologies and rules with tight semantic integration [49]. Even though using SWRL with OWL has distinct advantages, these approaches mainly suffer from limited expressiveness or undecidability [49]. In C-SWRL, the required expressivity is extended by C-SPARQL and OWLAPI functions. Namely, CWA reasoning has been accomplished by the former one while non-monotonic reasoning by the collaboration of SWRL with OWLAPI functions. Additionally, works described in [38], [39] and [84] prove that decidability can be retained by the so-called *DL-safe* rules. For example, retaining decidability in [84] is done through restricting the interface between OWL and rules i.e. rules apply only to individuals explicitly introduced in the ABox.

State of the art homogeny approaches, like the ones described in [6, 37], do not make any distinction between stream and static data, while also lack implementation. They prove that SWRL can be used to infer new and approximate knowledge in stream data domains. However, their approach does not consider time-aware and non-monotonic reasoning. Recently, a SPARQL extension [42] that uses CONSTRUCT/WHERE clauses to express rules has been proposed. Yet again this approach does not consider non-monotonic reasoning. The works presented in [77] and [78] describe a Rete-based [70] approach of RDFS entailment rules for producing data in a continuous manner. Although supporting time-aware and incremental reasoning, the approach does not deal with non-monotonic and closed-world reasoning. JNOMO [85] shows how SWRL can be extended to embrace non-monotonicity, CWA and NAF. Namely, `NotExist` operator is defined to “close” the world and to enable fact retraction. However, it does not deal with stream data, while inclusion of temporal reasoning is envisioned as per future works.

Chapter VII Conclusion and Future Works

VII. 1 Conclusions

Until recently most of the SR research has been dedicated on ontology and query processing developments. Dealing with Big Data issues through query processing is not enough. In fact, their use is intended for answering the user queries by not having any effect on the underlying knowledge base. The works on this thesis go beyond the query processing achievements and thus focus on rule level implications of stream data. Following their expressivity limitations, the Semantic Web rules have been neglected or somehow omitted when doing inference on stream data domains. Thus, the main contribution of this paper is in establishing a unique Semantic Web rule system, so called C-SWRL, capable for expressive reasoning over stream data. The INWS ontology was developed following this vision, an SSN-based ontology framework for WSNs in WQM. Moreover, a production rules system, StreamJess, was developed to show how this model can be used to reason over stream data. The rest of this chapter describes the specific contributions of this thesis.

INWS ontology. Chapter 3 of this thesis describes the ontology that was built for modeling WQM systems based on WSNs. It is a SSN ontology extension that further captures the semantics of the specifics of this particular domain of discourse. It was shown how our approach differs from other ontological knowledge representations. Namely, the SSN ontology is designed for sensors and it does not deal with water body's classification. As such, it was used as a basis for building the INWS core ontology. Moreover, INWS supports different status classifications as opposed to TWC-SWQP ontology, which classifies water bodies into status "polluted" or not.

StreamJess. SWRL lacks the required expressivity level to reason over stream data. As an alternative, it was built StreamJess, a production rule system capable of expressive reasoning over stream data. It layers Jess on top of C-SPARQL to enable time-aware, closed-world and non-monotonic reasoning on stream data domains. Jess and Jess Tab functions were used to enable non-monotonic reasoning. Chapter 4 also described how StreamJess differs from other state-of-the-art approaches. Specifically, StreamJess supports both forward and backward-chaining, which offers us the opportunity to observe the trade-off between speed and memory

consumption. The system was validated in a WQM case study by running multiple C-SPARQL queries and Jess rules at the same time over the same RDF streams. *Example 1* demonstrated how C-SPARQL queries can be used to filter RDF streams in time windows. The outputted results were processed by Jess rules to classify individual pH observations into appropriate WFD statuses. Furthermore, an investigation rule fired in case of critical status detection and identified the potential sources of pollution. *Example 2* illustrated how WFD classification can be realized based on the average value of the observations. Except filtering the RDF streams were aggregated and then grouped by measurement site to classify and investigate BOD_5 observations.

C-SWRL. OWL and SWRL's OWA and monotonic reasoning provide the main challenging issues while building stream data applications. As a result, current state-of-the-art SR approaches have avoided Semantic Web rule standards and relied on CWA and non-monotonic rule-based systems. A SR system was built based on SWRL and thus proved the main hypothesis of this thesis. SWRL rules were layered on top of a state-of-the-art stream processing system, such as C-SPARQL, to enable time-aware, closed-world and non-monotonic reasoning on stream data applications. It was shown that for non-monotonic reasoning purposes, C-SWRL uses SWRL together with OWLAPI constructs to modify the knowledge base. Moreover, NAF was implemented in the stream processing level. Furthermore, decidability issues induced by the combination of OWL and SWRL have been tackled by a number of works [38, 39, 84].

VII. 2 Future Works

Our main activity as per future work remains the evaluation of the developed systems i.e. StreamJess and C-SWRL. Our initial findings show that evaluating C-SWRL proves difficult due to the nature of our system, code availability of related systems and published evaluation results. Regarding the stream processing level it has been discovered that C-SPARQL yields considerably lower throughput compared to JTALIS and CQELS [45]. Thus, our main evaluation concern remains the stream reasoning component. We agree with Barbieri et al. [26] urgency for development of specialized reasoners for stream data applications. It is also planned to compare C-SWRL performance against StreamJess [6]. Moreover, following the Jess's ability to support both forward and backward chaining, it is also planned to evaluate

StreamJess on both strategies and find the optimal trade-off between memory consumption and execution time.

As per incremental reasoning, it is believed that maintaining materializations on rules following ontology changes do not differ for stream data domains. However, a deeper research in this direction remains per future work.

Our future work also includes enabling temporal operators (serial, sequence, etc.) on C-SWRL. It is planned to build the application layer that will offer the user to pose queries over historical data, offer the possibility to select which measurement sites and/or water quality to monitor, etc.

Chapter VIII Appendix A – Drinking water observations dataset

In this appendix will be demonstrated the conversion of CSV data into RDF. The data set consists of drinking waters quality measurements made in summer 2012, namely June, July and August on 15 measurements sites in the region of the city of Tetova. According to [45] during the lab analysis the following parameters were investigated: THMs, water temperature, turbidity, residual chlorine, pH, electrical conductivity (EC), the total residue after of evaporation (TRAE), total dissolved solids (TDS), chemical oxygen demand (COD), total organic carbon (TOC), dissolved organic carbon (DOC), ultra-violet absorbance in 254 nm (UV254), specific ultra-violet absorbance (SUVA), nitrates and chlorides. Data were available in CSV format for three months. A CSV to RDF converter tool named QUIDICRC - QUIck 'n DIrty Csv to RDF Converter, a product of MIND-SWAP Project, was used to model data appropriately for importing in SSN ontology. Figure 18 represents the measured values in each measurement point (the first column), for each water quality parameter (the rest columns) during August 2012.

Sample point	Temperature	Turbidity	Residual chlorine	pH	EC	TRAE	TDS	COD	TOC	DOC	UV254	SUVA	Nitrates	Chlorides	THMs
T1	10.9	1.4	0.1182	7.36	254	162	189	4.59	4.41	4.28	0.145	0.03388	1.2	1.8	17.945
T2	11.2	0.3	0.1502	7.51	281	166	196	3.06	2.96	2.82	0.095	0.03369	1.6	3.3	30.184
T3	11.5	0.4	0.1502	7.52	282	167	197	3.08	2.97	2.83	0.096	0.03392	1.8	3.5	31.645
T4	11.2	0.4	0.1607	7.64	325	172	195	3.12	3.03	2.85	0.097	0.03403	2.4	4.6	33.544
T5	11.3	0.3	0.2501	7.12	258	158	184	2.67	2.54	2.46	0.083	0.03374	1.3	1.1	16.632
T6	11.6	0.7	0.2596	7.43	273	184	198	3.34	3.28	3.17	0.107	0.03375	2.3	3.7	19.404
T7	11.4	0.8	0.2807	8.05	291	185	201	3.42	3.38	3.26	0.11	0.03374	3.8	3.9	30.184
T8	11.5	0.9	0.3207	8.14	294	184	205	3.45	3.39	3.27	0.12	0.03697	2.9	4.2	28.887
T9	11.3	1.2	0.3113	8.53	302	197	208	3.52	4.44	4.38	0.148	0.03379	3.5	8.3	39.088
T10	11.4	1.5	0.24	8.81	385	203	243	3.87	3.67	3.55	0.12	0.0338	3.6	16.4	45.08
T11	11.3	1.6	0.2501	8.76	420	231	265	3.64	3.54	3.47	0.118	0.03401	3.7	18.3	43.596
T12	11.6	1.1	0.2045	7.46	288	164	187	3.37	3.29	3.16	0.107	0.03086	1.7	3.2	27.72
T13	11.3	1.2	0.22	8.52	296	177	227	3.82	3.66	3.54	0.119	0.03362	2.7	19.4	36.041
T14	11.4	1.3	0	7.37	691	384	522	3.67	3.53	3.43	0.116	0.03382	28.3	32.4	0
T15	11.4	1.3	0	7.38	693	386	526	3.68	3.54	3.42	0.116	0.03392	28.5	32.6	0

Figure 18. Input CSV file: August2012.csv

A fragment of mapping file `inwd.map.August.txt` is given below indicating the chlorides values during August on each measurement site.

```
<rdf:Description rdf:about="http://www.co-
ode.org/ontologies/ont.owl#AugObserveChlorides{{Sample Point}}">
  <rdf:type
rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#Observation"/>
  <ssn:observationResult rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#AugOutputChlorides{{Sample Point}}"/>
  <ssn:observationResultTime rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#August2012"/>
  <inws:observationResultLocation rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#{{Sample Point}}"/>
  <ssn:featureOfInterest rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#DrinkingWaterFeature"/>
  <ssn:hasQualityOfObservation rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#DrinkingWaterChlorides"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.co-
ode.org/ontologies/ont.owl#AugOutputChlorides{{Sample Point}}">
  <rdf:type
rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#SensorOutput"/>
  <ssn:hasValue rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#AugValueChlorides{{Sample Point}}"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.co-
ode.org/ontologies/ont.owl#AugValueChlorides{{Sample Point}}">
  <rdf:type
rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#ObservationValue"/>
  <dul:hasDataValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">{{Chlorides}
}</dul:hasDataValue>
</rdf:Description>
```

All parameter values for each month are coded in the mapping file similarly to the code above. The following QUIDICRC command was used to obtain the RDF data format:

```
quidicrc.pl map="inwd.map.August.txt" in="August2012.csv"
out="CSV\output_August2012.rdf"
```

A fragment of the output RDF file `output_August2012.rdf` is provided below illustrating a conversion result as from the mapping fragment-code given above for measurement point T9 of chlorides values.

```
<ssn:Observation rdf:about="http://www.co-
ode.org/ontologies/ont.owl#AugObserveChloridesT9">
   <ssn:observationResult>
      <ssn:SensorOutput rdf:about="http://www.co-
ode.org/ontologies/ont.owl#AugOutputChloridesT9">
         <ssn:hasValue>
            <ssn:ObservationValue rdf:about="http://www.co-
ode.org/ontologies/ont.owl#AugValueChloridesT9"><dul:hasDataValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">8.3</dul:hasDataVal
ue>
            </ssn:ObservationValue>
         </ssn:hasValue>
      </ssn:SensorOutput>
   </ssn:observationResult>
   <ssn:observationResultTime rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#August2012"/>
   <ssn:hasQualityOfObservation rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#DrinkingWaterChlorides"/>
   <ssn:featureOfInterest rdf:resource="http://www.co-
ode.org/ontologies/ont.owl#DrinkingWaterFeature"/>
</ssn:Observation>
```

Generated RDF files for each month were imported into the INWS ontology. Because the exact time of observations was not available, it was used the month of the observations encoded as OWL time ontology `Interval` individuals. An observation is related through property `ssn:observationResultTime` with a time interval instance, which in turn has `ssn:startTime` and `ssn:endTime` individuals of class `Instant` (a time ontology class). Individuals of class `Instant` are related with XML Schema date time data type through property `inXSDDateTime`.

Chapter IX Appendix B – Rivers quality observations dataset

During our experiments for the case of river's quality observations dataset two kinds of data were taken:

- Offline SQL data generator
- RDF streams generator

IX. 1 Offline SQL data generator

This kind of dataset was used for validating the INWS ontology with the expert system described in Section 3.3. As previously described in this section, simulated SQL data were transformed into RDF format with D2RQ data converter and then loaded on application startup. The mapping file is large, thus for brevity, will be given some examples of it here.

At the beginning, namespaces are set up like the following:

```
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
```

Then, the database connection is configured like follows:

```
map:database a d2rq:Database;
d2rq:jdbcDriver
"com.microsoft.sqlserver.jdbc.SQLServerDriver";
d2rq:jdbcDSN"jdbc:sqlserver://EDI-
PC:1433;instanceName=./SQLEXPRESS;user=sa;password=****;Databa
seName=WaterQuality"; .
```

And then follow the mapping to *class instances* and property relations. Namely, a class instance mapping example is encoded as follows:

```
map:Observation a d2rq:ClassMap;
d2rq:dataStorage map:database;
d2rq:uriPattern "http://inwatersense.uni-
pr.edu/ontologies/inws-
core.owl#oo@@dbo.WorkingDataStreams.Id|urlify@@";
d2rq:class ssn:Observation; .
```

Each tuple of the SQL table `WorkingDataStreams` is converted into RDF instance of the class `ssn:Observation`. Namely, new instance names are formed by appending to the string `http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#oo` the ID value of the tuple.

To illustrate the building of *object property* relations within instances it was used the following D2RQ commands:

```
map:Observation_SensorOutput a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:Observation;
d2rq:property ssn:observationResult;
d2rq:refersToClassMap map:SensorOutput;
d2rq:join "dbo.WorkingDataStreams.Id =>
dbo.WorkingDataStreams.Id"; .
```

This code relates `ssn:Observation` individuals with `ssn:SensorOutput` ones through `ssn:observationResult` object property. A *data property* assertion is made in the following form:

```
map:ObservationEntryTimeInstance_TimeInstance a
d2rq:PropertyBridge;
d2rq:belongsToClassMap map:ObservationEntryTimeInstance;
d2rq:property time:inXSDDateTime;
d2rq:column "dbo.WorkingDataStreams.EntryDate";
d2rq:datatype xsd:dateTime; .
```

This code excerpt relates all time instances i.e. of class `time:Instant` with the DateTime values taken from the database, which will become related through `time:inXSDDateTime` data property.

IX. 2 RDF streams generator

This dataset was used for validating StreamJess and C-SWRL. The data generator simulates sensor data in the following RDF streams general format. A single sensor observation includes information about the water quality parameter name, the measured value and measurement location.

```
(http://inwatersense.uni-pr.edu/stream#obs_60
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://inwatersense.uni-pr.edu/ontologies/inws-
core.owl#tmpObservation . (1481034750573))
(http://inwatersense.uni-pr.edu/stream#obs_60
http://purl.oclc.org/NET/ssnx/ssn#qualityOfObservation
http://inwatersense.uni-pr.edu/ontologies/inws-core.owl#pH .
(1481034750573))
(http://inwatersense.uni-pr.edu/stream#obs_60
http://purl.oclc.org/NET/ssnx/ssn#observationResult
http://inwatersense.uni-pr.edu/stream#so_60 . (1481034750573))
(http://inwatersense.uni-pr.edu/stream#so_60
http://purl.oclc.org/NET/ssnx/ssn#hasValue
http://inwatersense.uni-pr.edu/stream#ov_60 . (1481034750573))
(http://inwatersense.uni-pr.edu/stream#ov_60 http://www.loa-
cnr.it/ontologies/DUL.owl#hasDataValue
10.493^^http://www.w3.org/2001/XMLSchema#double .
(1481034750573))
```

Namely, a new `tmpObservation` individual `obs_60` is created. It becomes related with the quality of observation, in this case `pH` through `ssn:qualityOfObservation`. Moreover, it is related with `so_60` through `ssn:observationResult`, which in turn is related with `ov_60` through `hasValue` property. The latest instance is related through `dul:hasDataValue` with the measured value, namely 10.493. The quality name, measurement site and the measured value are randomly generated.

Chapter X Appendix C – Mapping Jess initial facts

The following Jess Tab commands are executed at each startup of StreamJess application:

```
(mapclass http://purl.oclc.org/NET/ssnx/ssn#Observation)
(mapclass http://purl.oclc.org/NET/ssnx/ssn#SensorOutput)
(mapclass http://purl.oclc.org/NET/ssnx/ssn#ObservationValue)
(mapclass http://inwatersense.uni-pr.edu/ontologies/inws-
regulations.owl#WFDstatus)
(mapclass http://inwatersense.uni-pr.edu/ontologies/inws-
core.owl#WaterQuality)
(mapclass http://inwatersense.uni-pr.edu/ontologies/inws-
core.owl#tmpObservation)
(mapclass http://www.w3.org/2006/time#Instant)
(mapclass
http://sweet.jpl.nasa.gov/2.1/realmHydroBody.owl#BodyOfWater)
(mapclass http://inwatersense.uni-pr.edu/ontologies/inws-
pollutants.owl#Pollutant)
(mapclass http://inwatersense.uni-pr.edu/ontologies/inws-
pollutants.owl#PollutionSources)
(mapclass
http://tw2.tw.rpi.edu/zhengj3/owl/epa.owl#WaterMeasurement)
(mapclass http://tw2.tw.rpi.edu/zhengj3/owl/epa.owl#MeasurementSite)
(reset)

; import Java classes
(import java.util.Random)
(import java.util.Date)

; A global random variable for building unique OWL resources
(bind ?*r* (random))

(printout t "All StreamJess initial components loaded..." crlf)
```

Chapter XI Appendix D – Stream reasoning systems source codes

The Stream Reasoning systems developed within this thesis, C-SWRL and StreamJess, extend C-SPARQL with non-monotonic capabilities. Namely, C-SWRL is a unique Semantic Web system for reasoning over stream data, while StreamJess is a Jess system capable of expressive reasoning over stream data.

The systems are written in Java 1.8. The "ready to go packs" are NetBeans projects. They are open source applications and are published on: http://streamreasoning.uni-pr.edu.

To install and start using C-SWRL on your machine you should download application's source distribution from the Download section of http://streamreasoning.uni-pr.edu. Unzip the zip file into your local folder. Import the project into your NetBeans. Download and import all the jar libraries into your project including: C-SPARQL[17] v0.9.6, OWL API[18] v4.0.2, SWRLTab[19] v1.0, SWRL API Drools Engine[20] v1.0 and JUnit[21] v4.10. A *Getting Started* tutorial is also available from the web page.

Similarly, to install and start using StreamJess on your machine you should download application's source distribution from the Download section of http://streamreasoning.uni-pr.edu. Unzip the zip file into your local folder. Import the project into your NetBeans. Download a copy of the InWaterSense ontology Protege project including all ontology modules from the Download section. Download and import all the jar libraries into your

[17] http://streamreasoning.org/resources/c-sparql
[18] http://owlapi.sourceforge.net/
[19] https://github.com/protegeproject/swrltab
[20] https://github.com/protegeproject/swrlapi-drools-engine
[21] http://junit.org/junit4/

project including: C-SPARQL v0.9.6, Jess[22] v7.1p2, Jess Tab[23] v1.7 and Protégé[24] v3.5. A *Getting Started* tutorial is also available from the web page.

[22] http://www.jessrules.com/
[23] http://www.jessrules.com/jesswiki/view?JessTab
[24] http://protege.stanford.edu/

References

[1] Ahmedi, L., Jajaga, E.: Normalization of relations and ontologies, In: The 10th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Databases, pp. 419-425, Cambrige, UK (2011)

[2] Ahmedi, L., Jajaga, E.: A database normalization tool using Semantic Web technologies, In: International Journal of Systems Applications, Engineering and Development, vol. 5 (2011)

[3] Jajaga, E., Ahmedi, L., Abazi-Bexheti, L.: Semantic Web Trends on Reasoning Over Sensor Data. In: 8th South East European Doctoral Student Conference, Thessaloniki, Greece (2013)

[4] Compton, M., Barnaghi, P., Bermudez, L., Garcıa-Castro, R., Corcho, O., Cox, S., Graybeal, J., M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. Le Phuoc, Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The SSN Ontology of the W3C Semantic Sensor Network, Incubator Group, Journal of Web Semantics (2012)

[5] Ahmedi, L., Jajaga, E., Ahmedi, F.: An Ontology Framework for Water Quality Management. In Corcho, Ó., Henson, C. A., Barnaghi, P. M. (eds.) SSN@ISWC. pp. 35-50. Sydney (2013)

[6] E. Jajaga, L. Ahmedi and F. Ahmedi: StreamJess: a stream reasoning framework for water quality monitoring. J. of Metadata, Semantics and Ontologies, in press.

[7] Bhatnagar, V., Kochhar, S.: Association Rule Mining, Encyclopedia of Artificial Intelligence, pp. 172-178, IGI Global (2009)

[8] Ding, W., Eick, C. F., Yuan, X., Wang, J., Nicot, J.: A Framework for Regional Association Rule Mining and Scoping in Spatial Datasets, Geoinformatica, vol. 15, issue 1, pp. 1-28 (2011)

[9] Ceccaroni, L., Cortes, U., Sanchez-Marre, M.: OntoWEDSS: an ontology-underpinned decision-support system for wastewater management (2001)

[10] Ceccaroni, L., Cortes, U., Sanchez-Marre, M.: WaWO-An ontology embedded into an environmental decision-support system for wastewater treatment plant management, In: Proceedings of ECAI2000 - Wo9: Applications of ontologies and problem-solving methods, pp. 2.1-2.9, Berlin, Germany (2000)

[11] Hitzler, P., Krotzsch, M., Parsia, B., Patel-Schneider, P., Rudolph, S., OWL 2 Web Ontology Language Primer, http://www.w3.org/TR/owl2-primer/ (2009)

[12] Jena Semantic Web Framework, http://jena.sourceforge.net/, last accessed July, 2nd 2013

[13] Chau, K. W.: An ontology-based knowledge management system for flow and water quality modeling, In: Advances in Engineering Software, vol. 38, no. 3, pp. 172-181 (2007)

[14] Anicic, D., Fodor, P., Rudolph, S., Stuhmer, R., Stojanovic, N., Studer, R.: A Rule-Based Language for Complex Event Processing Reasoning, In: Proceedings of the Fourth International Conference on Web reasoning and rule systems, pp. 42-57, Springer-Verlag Berlin, Heidelberg (2010)

[15] Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning, In: WWW 2011, pp. 635-644 (2011)

[16] Taylor, K., Leidinger, L.: Ontology-Driven Complex Event Processing, In: ESWC, LNCS, pp. 285-299, Springer, Greece (2011)

[17] Amato, G., Caruso, A., Chessa, S., Massi, V., Urpi, A.: State of the Art and Future Directions in Wireless Sensor Network's Data Management, In: Istituto di Scienza e Tecnologie dell'Informazione del CNR, (2004)

[18] Lewis, M., Cameron, D., Xie, S., Arpinar, I. B.: ES3N: A Semantic Approach to Data Management in Sensor Networks, In: A workshop of the 5th International Semantic Web Conference ISWC (2006)

[19] Gruber, T. R.: A translation approach to portable ontologies, In: Knowledge Acquisition, vol. 5(2), pp. 199-220, (1993)

[20] Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Huang, Y., Tresp, V., Rettinger, A., Wermser, H.: Deductive and Inductive Stream Reasoning for Semantic Social Media Analytics. In: Proceedings of the Seventh Extended Semantic Web Conference (ESWC'10), L. Aroyo, G. Antoniou, E. Hyvonen, A. Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, Eds. LNCS, vol. 6088, pp. 1–15. Springer-Verlag (2010)

[21] Henson, C. A., Pschorr, J. K., Sheth, A. P., Thirunarayan, K.: SemSOS: Semantic Sensor Observation Service, In: Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems (CTS 2009), Baltimore, MD (2009)

[22] Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a Streaming World! Reasoning upon Rapidly Changing Information, In: IEEE Intelligent Systems, vol. 24(6), pp. 83–89 (2009)

[23] Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: SPARQL for Continuous Querying. In: Proceedings of the Eighteenth International Conference on World Wide Web (WWW'09), Quemada, J., Leon, G., Maarek, Y., Nejdl, W. (eds.), pp. 1061–1062, ACM Press (2009)

[24] Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - extending SPARQL to process data streams. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 448–462. Springer, Heidelberg (2008)

[25] Rodriguez, A., McGrath, R. E., Liu, Y., Myers, J.: Semantic Management of Streaming Data. In: Proceedings of International Workshop on Semantic Sensor Networks, SSN, pp.80-85 (2009)

[26] Shahriar, M. S., De Souza, P., Timms, G.: Smart Query Answering for Marine Sensor Data, Sensors 2011, vol. 11(3), pp. 2885-2897 (2011)

[27] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleM, W3C Member Submission, http://www.w3.org/Submission/SWRL/, last accessed April, 22nd 2013 (2004)

[28] Sottara, D., Bragaglia, S., Mello, P., Pulcini, D., Luccarini, L., Giunchi, D.: Ontologies, Rules, Workflow and Predictive Models: Knowledge Assets for an EDSS, In: Seppelt, R.,. Voinov, A. A., Lange, S., Bankamp, D. (eds.), International Environmental Modelling and Software Society (iEMSs), 2012 International Congress on Environmental Modelling and Software Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany (2012)

[29] Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Stream Reasoning: Where We Got So Far. In: Proceedings of the 4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS) (2010)

[30] Kifer, M., Boley, H.: RIF Overview (Second Edition), W3C Working Group Note 5 February 2013, http://www.w3.org/TR/rif-overview/, last accessed July, 2nd 2013

[31] Schaaf, M., Grivas, S. G., Ackermann, D., Diekmann, A., Koschel, A., Astrova, I.: Semantic Complex Event Processing, In: Proceedings of the 5th WSEAS congress on Applied Computing conference, and Proceedings of the 1st international conference on Biologically Inspired Computation, pp.38-43 (2012)

[32] Yang, X., Ong, K. G., Dreschel, W. R., Zeng, K., Mungle, C. S., Grimes, C. A.: Design of a Wireless Sensor Network for Long-term, In-Situ Monitoring of an Aqueous Environment. Sensors, 2, pp. 455-472 (2002)

[33] J. Mei, E. P. Bontas: Reasoning Paradigms for SWRL-Enabled Ontologies Protégé With Rules, in: Workshop held at the 8th International Protégé Conference, Madrid, Spain, 2005.

[34] Mileo, A., Abdelrahman, A., Policarpio, S., Hauswirth, M.: StreamRule: A nonmonotonic stream reasoning system for the semantic web. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 247–252. Springer, Heidelberg (2013)

[35] Wang, P., Zheng, J. G. , Fu, L. , Patton, E. W. , Lebo, T. , Ding, L., Liu, Q., Luciano, J. S., McGuinness, D. L.: TWC-SWQP: A Semantic Portal for Next Generation Environmental Monitoring. TWC RPI, Troy, NY, (2011)

[36] Bendadouche, R., Roussey, C., De Sousa, G., Chanet, J., Hou, K. M.: Extension of the Semantic Sensor Network Ontology for Wireless Sensor Networks: The Stimulus-WSNnode-Communication Pattern. In: 5th International Workshop on Semantic Sensor Networks in conjunction with the 11th International Semantic Web Conference (ISWC), Boston (2012)

[37] Keßler, C., Raubal, M., Wosniok, C.: Semantic Rules for Context-Aware Geographical Information Retrieval, In: Barnaghi, P. (eds.) European Conference on Smart Sensing and Context, (EuroSSC 2009), LNCS, vol. 5741, pp. 77–92, Springer (2009)

[38] F. M. Donini, M. Lenzerini, D. Nardi and A. Schaerf: AL-log: Integrating Datalog and Description Logics, J. of Intelligent Information Systems, 10(3), pp. 227–252, 1998.

[39] S. Heymans, D. V. Nieuwenborgh and D. Vermeir: Nonmonotonic Ontological and Rule-Based Reasoning with Extended Conceptual Logic Programs, in: Proc. Second European Semantic Web Conference (ESWC 2005), vol. 3532, pp. 392–407, Springer Verlag, 2005.

[40] E. Della Valle, D. Dell'Aglio, A. Margara: Tutorial: Taming Velocity and Variety Simultaneously in Big Data with Stream Reasoning, in: The 10th ACM International Conference on Distributed and Event-Based Systems, Irvine, USA, June 20-24, 2016.

[41] Protégé mailing list archive, https://mailman.stanford.edu/pipermail/protege-user/2014-February/000082.html

[42] Anderson, J., Athan T., and Paschke, A. (2016), 'Rules and RDF Streams' - A Position Paper, in Proceedings of the RuleML 2016 Challenge, Doctoral Consortium and Industry Track hosted by the 10th International Web Rule Symposium (RuleML 2016), New York, USA.

[43] Komazec, S., and Cerri, D. (2011), 'Towards Efficient Schema-Enhanced Pattern Matching over RDF Data Streams' in 10th ISWC, Springer, Bonn, Germany.

[44] Ali, M. I., Ono, N., Kaysar, M., Shamszaman, Z. U., Pham, T.-L., Gao, F., Griffin, K., and Mileo, A. (2016) 'Real-time Data Analytics and Event Detection for IoT-enabled Communication Systems', Journal of Web Semantics: Science, Services and Agents on the World Wide Web.

[45] Durmishi, B. H., Vezi, D., Ismaili, M., Shabani, A., Abduli, S.: Trihalomethanes in Tetova's Drinking Water. Journal of Chemical, Biological and Physical Sciences, vol. 3, no. 1, pp. 140-149 (2012)

[46] O'Flynn, B., Regan, F., Lawlor, A., Wallace, J., Torres, J., O'Mathuna, C.: Experiences and recommendations in deploying a real time, water quality monitoring system. Measurement Science and Technology, vol. 21, n. 12 (2010)

[47] Directive 2000/60/EC of the European Parliament and of the Council of Europe of 23 October 2000 establishing a framework for Community action in the Field of water quality O.J. L327/1 (2000)

[48] Schmidt, K., Stuhmer, R., and Stojanovic, L. (2008), 'Blending complex event processing with the rete algorithm' in 1st International workshop on Complex Event Processing for the Future Internet colocated with the Future Internet Symposium, CEUR Workshop Proceedings, Vol. 412.

[49] Eiter, T., Ianni, G., Polleres, A., Schindlauer, R. and Tompits, H. (2006), 'Reasoning with Rules and Ontologies', in Reasoning Web, Second International Summer School 2006, Tutorial Lectures, LNCS, vol. 4126, Springer, pp. 93–127.

[50] Raskin, R. G., Pan, M. J.: Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). Computers & Geosciences, vol. 31, n. 9, pp. 1119-1125 (2005)

[51] Kasi, M. K., Hinze, A., Legg C., Jones S.: SEPSen: Semantic event processing at the sensor nodes for energy efficient wireless sensor networks. In: Proceedings of the 6th

ACM International Conference on Distributed Event-Based Systems, New York, pp. 119-122 (2012)

[52] Zaniolo, C. (2012), 'Logical foundations of continuous query languages for data streams' in Proceedings of Datalog, pp. 177–189.

[53] Advancing Discovery in Science and Engineering. Computing Community Consortium. Spring 2011.

[54] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B. C., Hendler, J.: Swoop: A 'Web' Ontology Editing Browser. Journal of Web Semantics (2005)

[55] O'Connor, M.J., Das, A.K.: SQWRL: a query language for OWL. In: OWL: Experiences and Directions (OWLED), Fifth International Workshop, Chantilly, VA (2009)

[56] W3C OWL Working Group.: OWL 2 Web Ontology Language New Features and Rationale. W3C Recommendation (2012)

[57] Wang Q., Li Y., Obreza T., Munoz-Carpena R.: Monitoring Stations for Surface Water Quality. University of Florida, IFAS Extension, Fact Sheet SL 218 (2004)

[58] UNECE, Standard Statistical Classification of Surface Freshwater Quality for the Maintenance of Aquatic Life. In: Readings in International Environment Statistics, United Nations Economic Commission for Europe, United Nations, New York and Geneva (1994)

[59] Kosovo Environmental Protection Agency (KEPA), The State of Water in Kosovo, Prishtine, MESP (2010)

[60] Babac, P., Milanovic, M., Babac, D., Pavlovic, Z., Babac, A.: Prerada Komunalnih Otpadnih Voda. Beograd, MZZSRS (1999)

[61] CIRCA, Monitoring under the Water Framework Directive Policy Summary to Guidance Document 7, Produced by Working Group 2.7—Monitoring, Common Implementation Strategy for the Water Framework Directive (2000/60/EC), Published Guidance Documents CIRCA Library (2003)

[62] Hardy, C. E. (2013), Stream Reasoning on Resource-Limited Devices. University of Dublin, Dublin, United Kingdom.

[63] Luckham, D. and Roy Schulte, W. (2011), Event Processing Glossary -Version 2.0. Event Processing Technical Society, 2nd edition.

[64] MacLarty, I., Langevine, L., Bossche, M. V., Ross. P.: Using SWRL for Rule Driven Applications. Technical report (2009)

[65] Unel, G. Roman, D.: Stream reasoning: A survey and further research directions. In FQAS, pp. 653–662 (2009)

[66] Groza, A., Letia, I.A.: Plausible Description Logic Programs for Stream Reasoning. In: Future Internet, vol. 4, pp. 865-881 (2001)

[67] de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In: Fourteenth International World Wide Web Conference, pp. 623–632. ACM, Chiba, Japan (2005)

[68] Report Highlight for Survey Analysis: Big Data Investment Grows but Deployments Remain Scarce in 2014, http://www.gartner.com/newsroom/id/2848718

[69] Liebig, T., Opitz, M.: Reasoning over Dynamic Data in Expressive Knowledge Bases with Rscale. In: The 10th International Semantic Web Conference, Bonn, Germany (2011)

[70] Forgy, C. L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19 (1), 17 – 37 (1982)

[71] Volz, R., Staab, S. and Motik, B. (2005), 'Incrementally maintaining materializations of ontologies stored in logic databases', Journal of Data Semantics II, Vol. 3360, pp. 1–34.

[72] Hill, E. F.: Jess in Action: Java Rule-Based Systems, Manning Publications Co., Greenwich, CT (2003)

[73] Wang E. and Kim, Y. S. (2006), 'A teaching strategies engine using translation from SWRL to Jess', in 8th International Conference on Intelligent Tutoring Systems, June 26-30 2006, LNCS Vol. 4053, pp. 51-60.

[74] Stuckenschmidt, H., Ceri, S., Della Valle, E., van Harmelen. F.: Towards expressive stream reasoning. In: Proceedings of the Dagstuhl Seminar on Semantic Aspects of Sensor Networks (2010)

[75] Della Valle, E., Schlobach, S., Krötzsch, M., Bozzon, A., Ceri, S., Horrocks, I.: Order matters! Harnessing a world of orderings for reasoning over massive data. Semantic Web Journal 4(2), 219–231 (2013)

[76] Sheth, A., Henson, C., Sahoo, S.S.: Semantic Sensor Web. IEEE Internet Comput-ing 12(4), 78–83 (2008)

[77] Albeladi, R. Martinez, K., Gibbins, N.: Incremental Rule-based Reasoning over RDF Streams: An Expression of Interest, RDF Stream Processing Workshop at the 12th Extended Semantic Web Conference, Portoroz, Slovenia (2015)

[78] Tallevi-Diotallevi, S., Kotoulas, S., Foschini, L., Lecue F. and Corradi (2013), 'A Real-time urban monitoring in Dublin using semantic and stream technologies', in The

Semantic Web ISWC 2013, Vol. 8219 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 178–194.

[79] Sources of Pollution, Foundation for Water Research, Information Note FWR-WFD16, (2005)

[80] Margara, A., Urbani, J., van Harmelen, F., Bal, H.: Streaming the web: Reasoning over dynamic data. Web Semantics: Science, Services and Agents on the World Wide Web, 25(0): 24 – 44 (2014)

[81] Walzer, K., Groch, M., Breddin, T.: Time to the Rescue - Supporting Temporal Reasoning in the Rete Algorithm for Complex Event Processing. In: Proceedings of the 19th international conference on Database and Expert Systems Applications, Turin, Italy (2008)

[82] Whitehouse, K., Zhao, F., Liu, J.: Semantic streams: A framework for Composable Semantic Interpretation of Sensor Data. In: Proceedings of the 3rd European Conference on Wireless Sensor Networks, Zurich, Switzerland (2006)

[83] Statutory Instruments. (2009), European Communities Environmental Objectives (Surface Waters) Regulations 2015 [online], S.I. No. 386 of 2015. http://www.irishstatutebook.ie/eli/2015/si/386/made/en/pdf. (Accessed 7 June 2016)

[84] B. Motik, U. Sattler and R. Studer: Query Answering for OWL-DL with rules, Journal of Web Semantics, 3(1), pp. 41–60, 2005.

[85] Calero, J.M.A., Ortega, A.M., Perez, G.M., Blaya, J.A.B., Skarmeta, A.F.G.: A non-monotonic expressiveness extension on the semantic web rule language. J. Web Eng. 11(2), 93–118 (2012)

[86] OWL time ontology. [online] http://www.w3.org/TR/owl-time/ (Accessed 7 June 2016)

[87] Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth. M.: A native and adaptive approach for unified processing of linked streams and linked data. In: The Semantic Web–ISWC 2011, pp. 370–388 (2011)

[88] Gebser, M., Grote, T., Kaminski, R., Obermeier, P., Sabuncu, O., Schaub. T.: Answer set programming for stream reasoning. In: CoRR (2013)

[89] O'Connor, M. J., Knublauch, H., Tu, S. W., Grossof, B., Dean, M., Grosso, W. E. and Musen, M. A. (2005) 'Supporting rule system interoperability on the Semantic Web with SWRL', in 4th International Semantic Web Conference, Galway, Ireland, Springer Verlag, LNCS Vol. 3729, pp. 974-986.

[90] Ahmedi, L., Sejdiu, B., Bytyçi, E. and Ahmedi, F. (2015), 'An Integrated Web Portal for Water Quality Monitoring through Wireless Sensor Networks', International Journal of Web Portals (IJWP), Vol. 7 No. 1, pp. 28-46.

[91] McBride, B. (2004), 'Jena: implementing the RDF model and syntax specification', in Proceedings at Semantic Web Workshop (WWW).

[92] Le-Phuoc, D., Dao-Tran, M., Pham, M.-D., Boncz, P., Eiter, T. and Fink, M. (2012), 'Linked stream data processing engines: facts and figures', in The Semantic Web–ISWC 2012, Springer, pp. 300–312.

[93] Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E. and Grossniklaus, M. (2010), 'C-SPARQL: a continuous query language for RDF data streams', International Journal of Semantic Computing, Vol. 04 No. 01, pp. 3–25.

[94] Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E. and Grossniklaus, M. (2010) 'Incremental reasoning on streams and rich background knowledge', in Proceedings of the Extended Semantic Web Conf. (ESWC 2010), Heraklion, Crete, Greece, pp.1-15.

[95] Basic Geo (WGS84 lat/long) Vocabulary. [online] http://www.w3.org/2003/01/geo/. (Accessed 7 June 2016).

[96] Boley, H., Kifer, M., Pătrânjan, P.-L. and Polleres, A. (2007), 'Rule interchange on the web', in Reasoning Web, LNCS, Springer, Heidelberg, Vol. 4636, pp. 269–309.

[97] Lanzanasto, N., Komazec, S. and Toma, I. (2009), Deliverable D4.8: Reasoning over real time data streams, ENVISION Consortium.

[98] Jess Wiki: Jess Tab. [online] http://www.jessrules.com/jesswiki/view?JessTab (Accessed 7 June 2016).

[99] Della Valle, E., Ceri, S., Barbieri, D. F., Braga, D. and Campi, A. (2008), 'A first step towards stream reasoning' in Proceedings of Future Internet Symposium (FIS 08), Springer, pp. 72–81.

[100] Jajaga, E., Ahmedi, L. and Ahmedi, F. (2015), 'An Expert System for Water Quality Monitoring Based on Ontology', in MTSR 2015: Proceedings of the 9th Metadata and Semantics Research Conference, Manchester, UK, Vol. 544 pp. 89-100.

[101] Environment Agency. (2011), Method statement for the classification of surface water bodies, v2.0 (external re-lease) [online], Monitoring Strategy v2.0, July 2011. (Accessed 7 June 2016).

[102] Ermert, L. (2009), Comparing Jess and Esper for Event Stream Processing. Bachelor Thesis, Faculty IV - depart-ment computer science, Fachhochschule Hannover, Germany.

[103] INWS core ontology. [online] http://inwatersense.uni-pr.edu/ontologies/inws-core.owl (Accessed 7 June 2016).

[104] INWS pollutants ontology. [online] http://inwatersense.uni-pr.edu/ontologies/inws-pollutants.owl (Accessed 7 June 2016).

[105] INWS regulations ontology. [online] http://inwatersense.uni-pr.edu/ontologies/inws-regulations.owl (Accessed 7 June 2016).

[106] InWaterSense project. [online] http://inwatersense.uni-pr.edu/ (Accessed 7 June 2016).

[107] Grosof, B. N., Gandhe, M. D. and Finin, T. W. (2002), 'SweetJess: Translating DamlRuleML to Jess', in: Proceed-ings of International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, held at 1st International Semantic Web Conference.

[108] Horridge M. and Bechhofer, S. (2009), 'The OWL API: A Java API for working with OWL 2 ontologies', in 6th OWL Experienced and Directions Workshop, Chantilly, Virginia.

[109] Wei, W., Barnaghi, P.: Semantic Annotation and Reasoning for Sensor Data, In: Smart Sensing and Context, pp.66-76 (2009)

[110] Antoniou, G., and van Harmelen, F. *The Semantic Web Primer*, MIT Press Cambridge, MA, USA, 2004.

[111] Anicic, D. Event Processing and Stream Reasoning with ETALIS, PhD thesis, 2011.