International Journal of Semantic Computing

World Scientific
www.worldscientific.com

C-SWRL: A UNIQUE SEMANTIC WEB FRAMEWORK FOR REASONING OVER STREAM DATA

EDMOND JAJAGA

Department of Computer Science, South East European University, Ilindenska n. 335, Tetovë, 1200, Macedonia
e.jajaga@seeu.edu.mk

LULE AHMEDI

Department of Computer Engineering, University of Prishtina, Kodra e diellit pn, Prishtinë, 10000, Kosova
lule.ahmedi@uni-pr.edu

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The synergy of Data Stream Management Systems and Semantic Web applications has steered towards a new paradigm known as Stream Reasoning. The Semantic Web standards for knowledge base modeling and querying, namely RDF, OWL and SPARQL, has extensively been used by the Stream Reasoning community. However, the Semantic Web rule languages, such as SWRL and RIF, have never been used in stream data applications. Instead, different non-Semantic Web rule systems have been approached. Since RIF is primarily intended for exchanging rules among systems, we focused on SWRL applications with stream data. This proves difficult following the SWRL's open world semantics. To overcome SWRL's expressivity issues we propose an infrastructure extension, which will enable SWRL reasoning with stream data. Namely, a query processing system, such as C-SPARQL, was layered under SWRL to support closed-world and time-aware reasoning. Moreover, OWLAPI constructs were utilized to enable non-monotonicity, while SPARQL constructs were used to enable negation as failure. Water quality monitoring was used as a validation domain of the proposed system.

Keywords: stream data; Semantic Web; reasoning; SWRL; rules.

1. 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 [5]. 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 satisfied with the classic solutions for data stream and complex event processing and with reasoning engines for static data [44]. The W3C RDF Stream Processing Community Group 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 [50], StreamRule [13] etc.), rule-based reasoning over RDF streams still remains vastly unexplored.

This paper proposes a unified Semantic Web approach for rule-based reasoning over stream data complementing state of the art query processing engine C-SPARQL [10] with the W3C recommended Semantic Web rule language SWRL.

Semantic technologies have proved evidence of efficient implementations on stream data domains [1]. Firstly, OWL ontologies have been widely used for modeling stream data domains, e.g., the SSN ontology [40]. Secondly, querying these knowledge bases has been merely done by SPARQL extensions e.g. C-SPARQL [10], EP-SPARQL [11], etc. However, the windows opened over streams can determine changes in the static information sources [16]. 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 [3], using Semantic Web recommended rule languages, SWRL [4] and RIF [25], over stream data has to the best of our knowledge not been considered to date. Thus, as described in our previous works [1-3], there is an inherent need for a Semantic Web unified rule system capable of reasoning with stream data. In line with this vision, we have previously developed the INWATERSENSE (hereinafter referred to as INWS) ontology [2] and an expert system [3] demonstrating its usage. In this paper, we de-scribe Continuous SWRL (or simply C-SWRL), 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. The system is publicly available on http://streamreasoning.uni-pr.edu/. The link contains the source code, installation instructions and getting started tutorial.

The system was validated with simulated data in the water quality monitoring (WQM) domain, but it is developed for use within the InWaterSense project with real data. InWaterSense is an EU funded re-search project aimed to apply recent advanced practices stemming from ICT in WQM for healthy environment, and strengthen Kosovo's capacity in research in national priority sectors of environment and ICT. An intelligent wireless sensor network (WSN) for monitoring surface water quality has been deployed in a river in Kosova [1-3], and is further being enriched with more intelligent behavior like is the contribution presented in this paper.

The paper is organized as follows. Section 2 describes the motivation of building C-SWRL. The main contribution is presented in Section 3 which exhibits our proto-typical design and implementation. System validation is presented in Section 4 through examples in the domain of WQM. Section 5 describes the challenges encountered while building C-SWRL. Related works take place on Section 6. Finally, the paper closes with conclusion and future plans.

2. Motivation

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 [6]. 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 [6]. 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, a SR system should support reasoning over both streaming information and background data [7]. Moreover, some specific requirements about this property already mentioned in state of the art systems e. g. StreamRule [13], should be also 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.

2.1. *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 [8]. For example, aggregate operations are present in almost every rule for classifying water bodies into corresponding statuses [14] 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 [39] al-lows this through the use of `sqwrl:average` [2]. However, that approach is not supported, since using SQWRL constructs in SWRL rules for asserting new knowledge is not allowed [15].

Additionally, a number of example rules need to infer new knowledge in absence of a fact or incomplete knowledge, the concept known as negation as failure (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" [14] cannot be expressed in SWRL.

2.2. *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 [17]. In SR applications, change to the facts occurs "regularly". A technique for computing ontological entailments on SR is presented in [18]. It uses Logic Programming (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 [17], changes to the ontology will typically require changes in the rules. Authors of [17] 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.

2.3. *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 [7]. INWS ontology implements the time model through OWL Time ontology [19]. 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 [9]. 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 [13]. 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, we define the concept of continuous rules 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.

3. System design and implementation

The conceptual architecture of our prototype system is depicted in Figure 1. 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 rule layer consists of rules (pink track left) and continuous rules (pink track right). Grey arrows describe data flow direction.

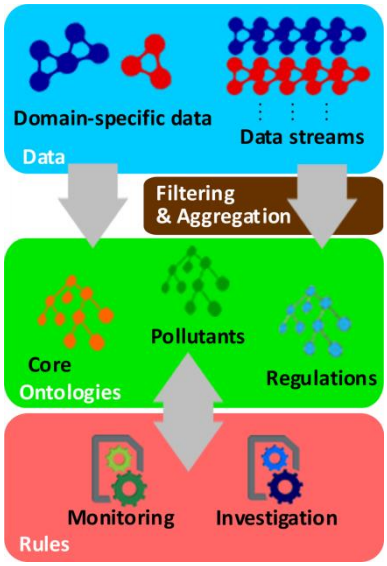

Fig. 1. C-SWRL conceptual architecture.

3.1. *Data layer*

Two kinds of data are present in SR applications: (i) domain specific ABox data which do not change or change "slowly" that are formulated in the form of RDF data e.g. river names, and (ii) stream data. Stream processing systems (e.g., C-SPARQL and EP-SPARQL) model stream data in the form of RDF streams. RDF streams are defined as a sequence of RDF triples that are continuously produced and annotated with a timestamp [10]. For example, a single RDF stream will usually hold information of a measured water quality value coupled with timestamp and location information. In C-SWRL's conceptual framework, RDF data populate the corresponding ontology classes, while RDF streams are firstly processed by C-SPARQL and SWRL rules and then archived as historical data in the ontology as depicted in Figure 2.

3.2. *Ontology layer*

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.

Ontologies are defined as formal specification of a shared conceptualization [12]. They have been extensively used for modeling stream data domains. In our previous

work [2], we have built the INWS ontology, an ontology framework for modeling water quality monitoring systems. INWS ontology consists of three ontology modules: core, regulations and pollutants. The core ontology is a SSN-based ontology [40] which models WSN infrastructure entities, observations and water quality parameters. The regulations ontology models classification of water bodies based on different regulation authorities such as Water Framework Directive (WFD) [28]. Finally, the pollutants ontology models the entities for investigating sources of pollution.

To manipulate with ontology modules the OWL API [27] version 4.0.2 was used in the development of C-SWRL. All ontology modules are imported and loaded at application startup.

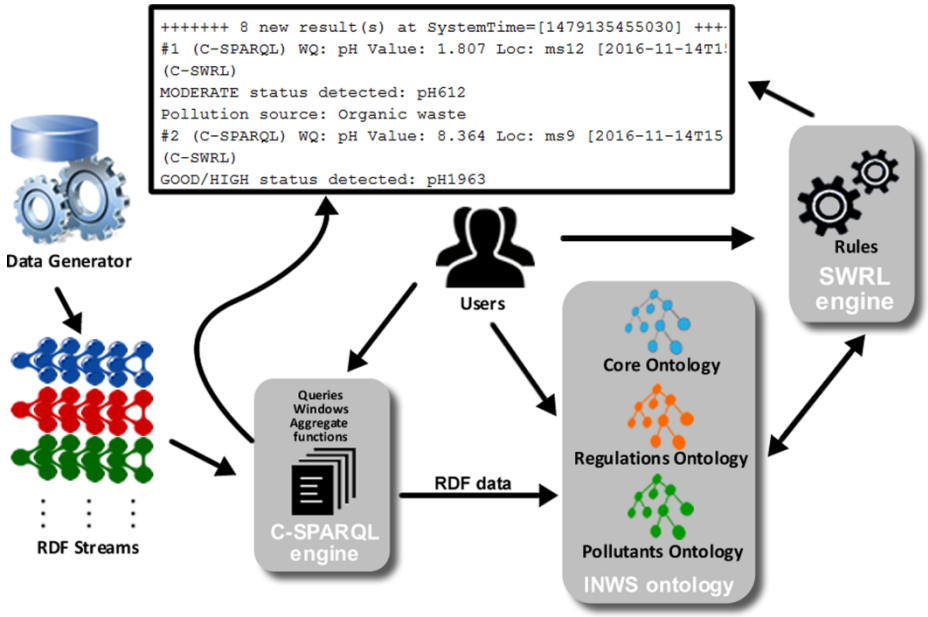

Fig. 2. C-SWRL system workflow.

3.3. *Rules layer*

In the previous section, the concept of continuous rules was introduced as rules that should continually infer new implicit knowledge from RDF streams. In C-SWRL, the inferred knowledge forms a set of RDF data which are stored back into the ontology to maintain record of historical data and for doing further reasoning over them.

In order to enable SWRL to reason over stream data 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 the previous section are supported. State-of-the-art SWRL extensions may support one, but fail on another feature. For example, JNOMO [20] is a SWRL extension for enabling non-monotonic reasoning, but it does not support time-aware reasoning. JNOMO [20] is also an example of translating SWRL into Jena [21]. Moreover, an intelligent tutoring system framework introduced in [22] and SweetJess [23] represent further examples of rules interoperability systems with translating SWRL and RuleML respectively into Jess rules [24]. 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. As depicted in Figure 2, C-SWRL uses C-SPARQL output RDF streams as input for SWRL 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 [27] 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. These rules mainly fall into two broad categories:

(i) *monitoring rules*, rules for continuous classification of water bodies based on in situ observations, and
(ii) *investigation rules*, 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.

In another domain, say medicine, the monitoring rules may continually classify the human‘s health status, while the investigation ones may try to identify the potential sources of the disease in cases of critical status detection. SWRLAPI [29] 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.

4. The water quality monitoring case study

C-SWRL is validated in a typical water quality monitoring scenario based on WSN. We assume that sensors are deployed in different measurement sites at different times. They continually emit water quality measured values. C-SWRL will (1) classify the water body into the appropriate status according to WFD regulations [41, 14] and (2) identify the potential sources of pollution if the values are out of the allowed standard. For brevity, we will demonstrate the cases of Biochemical Oxygen Demand (BOD5) and pH observations. Like most of water quality parameter observations, BOD5 observations are classified based on the average value of measurements within a time interval while pH ones are considered one by one [14]. Both examples run at the same time over the same RDF streams which are filtered out by two different C-SPARQL queries. RDF streams generator runs in background simulating sensor measurements on, arbitrarily set, three measurement sites: ms10, ms11 and ms12. pH measurements appear on each site while BOD5 ones are appearing on ms10 and ms11. The streaming rate is arbitrarily set to one stream per second. A single RDF stream holds information of the measured value, water quality name, observation time and location and the device providing the observation. Figure 3 illustrates a screenshot of the C-SWRL console output of the running examples 1 and 2.

4.1. *Example 1: 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 [14]. 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 [9].

After processing the first window of RDF streams a new BOD_5 average value is calculated by the following C-SPARQL query:

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

The query runs against the input RDF streams in the time frame of 20 seconds, sliding the window by 20 seconds. The chosen time frame is arbitrary and the user can change its values as desired. It produces triples of values: the water quality name (`?qo`), the location of measurements (`?loc`) and the calculated average value (`?avg`). The triples are filtered out to exclude pH ones and are firstly grouped by the water quality name and then by the measurement site. At every 20 seconds new RDF streams enter into the window and old ones exit. An output of a window processing of this query is depicted in the lower part of Figure 3, namely on the lines marked with '#' symbol followed by an order number and (C-SPARQL) label. Namely, C-SPARQL has outputted two results.

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 `ssn: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 3 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)
```

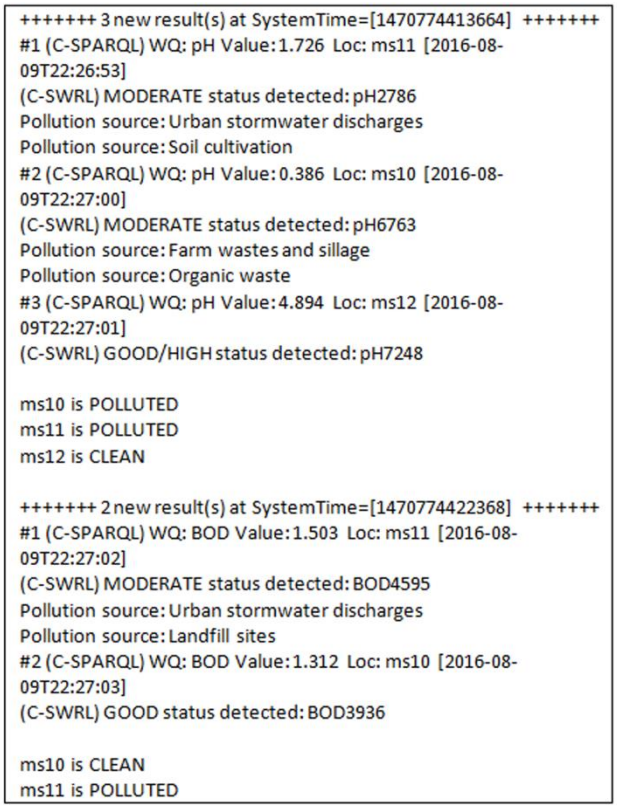

Fig. 3. An output excerpt of the running examples 1 and 2 on C-SWRL.

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 3. From the Figure we can observe 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 3, we can observe that the last statuses for measurement sites `ms10` and `ms11` are "clean" and "polluted", respectively.

4.2. *Example 2: pH observations*

A WFD rule for classifying pH observations looks like follows: The pH as individual value should be between 4.5 and 9.0 [14]. 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 [9].

The query to filter individual pH measurements is much simpler than the BOD_5 one. 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 3. 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.

5. Challenges

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

5.1. *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 we used the technique "remove and assert". 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.

5.2. *Aggregates*

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.

5.3. *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.

5.4. *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[a], but the aggregate functions are hardly implementable in SWRL following its OWA.

6. Related works

State-of-the-art rule-based systems for reasoning over stream data mainly fall into two broad categories: hybrid and homogeneous approaches [1]. 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 [51].

6.1. *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 [51]. In our previous work [1], we 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 subclass statements consisting of disjunction of classes or expressed with existential quantification are not possible into Plausible Logic [30].
- *Reasoner issues*: Since the ontology and the rules are treated separately then a rule engine and a DL reasoner will run concurrently [52]. As argued in [52], 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 [6]. This makes it harder for domain experts to write rules without IT support. In some cases (as shown in [6]), development layers are conflate to each other making rules maintenance more laborious.

A similar approach to C-SWRL is followed by StreamRule [13], 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 [31], while OClingo [32] is used to enable non-monotonic reasoning. In C-SWRL we use C-SPARQL for filtering and aggregation purposes, and OWLAPI for non-monotonic

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

reasoning. Even though that CQELS outperforms C-SPARQL [38], we preferred C-SPARQL following its advantage to use nested aggregations and negation [37, 38]. Moreover, we plan to support temporal operators, which lack any support in CQELS [37]. Another feature difference between StreamRule and C-SWRL is the historical data management, which is one of the key requirements of SR tools [7]. 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, [46] 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 negation as failure 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 [36] 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 [50] together with EP-SPARQL [11] 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 [38]. Moreover, there is no support for triple-based windows too.

6.2. *Semantic Web approaches*

In the literature this approach is also referred to as interaction of ontologies and rules with tight semantic integration [51]. Even though using SWRL with OWL has distinct advantages, these approaches mainly suffer from limited expressiveness or undecidability [51]. 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 [47], [48] and [49] prove that decidability can be retained by the so-called DL-safe rules. For example, retaining decidability in [49] 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 [33, 34], 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 [43] 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 [16, 35] describe a Rete-based [42] 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 [20] 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.

7. Conclusion and future works

Until recently most of the SR research has been dedicated on ontology and query processing developments. Dealing with stream reasoning issues through query processing is not enough. Our work goes beyond the query processing achievements and thus focusing on rule level implications of stream data. SWRL, on its own, lacks the required expressivity level to reason over stream data. The main contribution of this paper is in establishing a unique Semantic Web rule system capable for expressive reasoning over stream data. In this vision, we developed C-SWRL which layers SWRL on top of C-SPARQL to enable time-aware, closed-world and non-monotonic reasoning on stream data applications. We believe 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. 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.

Our future work includes enabling temporal operators (serial, sequence, etc.) on C-SWRL. We plan 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.

Furthermore, we are currently evaluating the examples presented here in Drools, for which we shall conduct a thorough performance evaluation and thus analyze the scalability issues. 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. We also plan to compare C-SWRL performance against our previously developed Jess system [3].

References

[1] E. Jajaga, L. Ahmedi and L. Abazi-Bexheti, Semantic Web trends on reasoning over sensor data, in: 8th South East European Doctoral Student Conference, Greece, 2013.
[2] L. Ahmedi, E.Jajaga and F. Ahmedi, An ontology framework for water quality management, in: Ó. Corcho, C. A. Henson, and P. M. Barnaghi, ed., SSN@ISWC, Sydney, 2013, pp. 35-50.
[3] E. Jajaga, L. Ahmedi and F.Ahmedi, StreamJess: a stream reasoning framework for water quality monitoring, International Journal of Metadata, Semantics and Ontologies (In press).
[4] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, SWRL: A Semantic Web rule language combining OWL and RuleML, 2004.
[5] E. Della Valle, S. Ceri, D. F. Barbieri, Daniel Braga, and A. Campi, A first step towards stream reasoning, in: Proc. Future Internet Symposium (FIS 08), Springer, 2008, pp. 72–81.
[6] I. MacLarty, L. Langevine, M. V. Bossche and P. Ross, Using SWRL for rule driven applications, Technical report, 2009.
[7] A. Margara, J. Urbani, F. van Harmelen and H. Bal, Streaming the web: reasoning over dynamic data, Web Semantics: Science, Services and Agents on the World Wide Web, 25(0), (2014), 24 – 44.
[8] C. Zaniolo, Logical foundations of continuous query languages for data streams, in: Proc. Datalog, 2012, pp. 177–189.
[9] Sources of Pollution, Foundation for Water Research, Information Note FWR-WFD16, 2005.
[10] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle and M. Grossniklaus, C-SPARQL: a continuous query language for RDF data streams, International Journal of Semantic Computing 04-01 (2010), 3–25.
[11] D. Anicic, P. Fodor, S. Rudolph and N. Stojanovic, EP-SPARQL: a unified language for event processing and stream reasoning, in: WWW 2011, 2011, pp. 635–644.
[12] T. R. Gruber, A translation approach to portable ontologies, Knowledge Acquisition 5(2) (1993), 199–220.
[13] A. Mileo, A. Abdelrahman, S. Policarpio and M. Hauswirth, StreamRule: a nonmonotonic stream reasoning system for the semantic web, in: W. Faber, D. Lembo, ed., RR 2013, LNCS, Springer, Heidelberg, 2013, vol. 7994, pp. 247–252.
[14] European Communities Environmental Objectives (Surface Waters) Regulations, 2009.
[15] Protégé mailing list archive, https://mailman.stanford.edu/pipermail/protege-user/2014-February/000082.html
[16] S. Tallevi-Diotallevi, S. Kotoulas, L. Foschini, F. Lecue and A. Corradi, Real-time urban monitoring in Dublin using semantic and stream technologies, in: The Semantic Web ISWC 2013, H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. Parreira, L. Aroyo, N. Noy, C. Welty and K. Janowicz, ed., vol. 8219 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 178–194.
[17] R. Volz, S. Staab and B. Motik, Incrementally maintaining materializations of ontologies stored in logic databases, J. Data Semantics II (2005), 3360, 1–34.
[18] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle and M. Grossniklaus, Incremental reasoning on streams and rich background knowledge, in: Proc. of the Extended Semantic Web Conf. (ESWC 2010), 2010, Heraklion, Crete, Greece, pp.1-15.
[19] OWL time ontology, http://www.w3.org/TR/owl-time/
[20] J. M. A. Calero, A. M. Ortega, G. M. Perez, J. A. B. Blaya and A. F. G. Skarmeta, A non-monotonic expressiveness extension on the semantic web rule language, J. Web Eng., 11(2), pp. 93–118, 2012.
[21] B. McBride, Jena: implementing the RDF model and syntax specification, in: Proc. at Semantic Web Workshop (WWW), 2004.
[22] E. Wang and Y. S. Kim, A teaching strategies engine using translation from SWRL to Jess, in: 8th International Conference on Intelligent Tutoring Systems, ITS 2006, June 26-30, 2006, LNCS vol. 4053, pp. 51-60.

[23] B. N. Grosof, M. D. Gandhe and T. W. Finin, SweetJess: Translating DamlRuleML to Jess, in: Proc. Intl. Wksh. on Rule Markup Languages for Business Rules on the Semantic Web, held at 1st Intl. Semantic Web Conf., 2002.
[24] E. F. Hill, Jess in action: Java rule-based systems, Manning Publications Co., Greenwich, CT, 2003.
[25] H. Boley, M.Kifer, P.-L. Patranjan, and A. Polleres, Rule interchange on the web, in: G. Antoniou, U. Aßmann, C. Baroglio, S. Decker, N. Henze, P.-L. Patranjan, R. Tolksdorf, ed., Reasoning Web, LNCS, Springer, Heidelberg, 2007, vol. 4636, pp. 269–309.
[26] D. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus: 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.
[27] M. Horridge and S. Bechhofer, The OWL API: A Java API for working with OWL 2 ontologies, in: OWLED 2009, 6th OWL Experienced and Directions Workshop, Chantilly, Virginia, 2009.
[28] 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.
[29] M. J. O'Connor, H. Knublauch, S. W. Tu, B. Grossof, M. Dean, W. E. Grosso, and M. A. Musen, Supporting rule system interoperability on the Semantic Web with SWRL, in: 4th International Semantic Web Conference (ISWC), Galway, Ireland, Springer Verlag, LNCS 3729, 2005, pp. 974-986.
[30] A. Groza and I. A. Letia, Plausible Description Logic Programs for Stream Reasoning, in: Future Internet, 2001, vol. 4, pp. 865-881.
[31] D. Le-Phuoc, M. Dao-Tran, J. Xavier Parreira and M. Hauswirth, A native and adaptive approach for unified processing of linked streams and linked data, in: The Semantic Web–ISWC 2011, 2011, pp. 370–388.
[32] M. Gebser, T. Grote, R. Kaminski, P. Obermeier, O. Sabuncu, and T. Schaub, Answer set programming for stream reasoning, in: CoRR, 2013.
[33] W. Wei and P. Barnaghi, Semantic annotation and reasoning for sensor data, in: Smart Sensing and Context, 2009, pp.66-76.
[34] C. Keßler, M. Raubal and C, Wosniok, Semantic rules for context-aware geographical information retrieval, in: P. Barnaghi, ed., European Conference on Smart Sensing and Context, EuroSSC 2009, LNCS, Springer, 2009, vol. 5741, pp. 77–92.
[35] R. Albeladi, K. Martinez and N. Gibbins, Incremental rule-based reasoning over RDF streams: An expression of interest, in: RDF Stream Processing Workshop at the 12th Extended Semantic Web Conference, Portoroz, Slovenia, 2015.
[36] T. Liebig and M. Opitz, Reasoning over dynamic data in expressive knowledge bases with Rscale, in: The 10th Int. Semantic Web Conference, Bonn, Germany, 2011.
[37] N. Lanzanasto, S. Komazec and I. Toma, Deliverable D4.8: Reasoning over real time data streams, ENVISION Consortium 2009-2012.
[38] D. Le-Phuoc, M. Dao-Tran, M.-D. Pham, P. Boncz, T. Eiter, and M. Fink, Linked stream data processing engines: facts and figures, in: The Semantic Web–ISWC 2012, Springer, 2012, pp. 300–312.
[39] M. J. O'Connor and A. K. Das, SQWRL: a query language for OWL, OWL: Experiences and Directions (OWLED), 6th Int. Workshop, Chantilly, VA, 2009.
[40] M. Compton, P. Barnaghi, L Bermudez, R. GarcíaCastro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. A. Henson, A. Herzog, V. A. Huang, K. Janowicz, W. D. Kelsey, D. L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. R. Page, A. Passant, A. P. Sheth and K. Taylor, The SSN ontology of the W3C semantic sensor network incubator group, Journal of Web Semantics 17 (2012), 25–32.

[41] Method statement for the classification of surface water bodies, v2.0 (external release), Monitoring Strategy v2.0, July 2011.
[42] C. L. Forgy, Rete: A fast algorithm for the many pattern/many object pattern match problem, Artificial Intelligence 19 (1) (1982), 17 – 37.
[43] J. Anderson, T. Athan and A. Paschke: 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, July 6-9, 2016.
[44] 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.
[45] D. Le-Phuoc, M. Dao-Tran, M.-D. Pham, P. Boncz, T. Eiter and M. Fink: Linked stream data processing engines: facts and figures, in: The Semantic Web – ISWC 2012. Springer, pp. 300–312, 2012.
[46] M. I. Ali, N. Ono, M. Kaysar, Z. U. Shamszaman, T.-L. Pham, F. Gao, K. Griffin and A. Mileo: Real-time Data Analytics and Event Detection for IoT-enabled Communication Systems, J. of Web Semantics: Science, Services and Agents on the World Wide Web, July, 2016.
[47] 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.
[48] 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.
[49] B. Motik, U. Sattler and R. Studer: Query Answering for OWL-DL with rules, Journal of Web Semantics, 3(1), pp. 41–60, 2005.
[50] D. Anicic, P. Fodor, S. Rudolph, R. Stuhmer, N. Stojanovic, R. Studer: 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.
[51] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer and H. Tompits: Reasoning with rules and ontologies, in: P. Barahona, F. Bry, E. Franconi, N. Henze, U. Sattler (Eds.), Reasoning Web, Second International Summer School 2006, Tutorial Lectures, LNCS, vol. 4126, Springer, pp. 93–127, 2006.
[52] 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.