Offshore Holdings Analytics Using Datalog$^+$ RuleML Rules

Mohammad Sadnan Al Manir and Christopher J.O. Baker

Department of Computer Science and Applied Statistics
University of New Brunswick, Saint John, Canada
{sadnan.almanir,bakerc}[at]unb.ca

Abstract. In April 2013, the International Consortium of Investigative Journalists (ICIJ) exposed the details of 130,000 offshore accounts. Although there are legitimate businesses which use such accounts, there exist a number of accounts which are possibly linked to international tax fraud and money laundering. The ICIJ investigation was based on 2.5 million records of offshore holdings linked to 170 countries. All these records have been made available for further examination and analysis. Based on these records, a set of facts, rules and queries have been formulated in Datalog$^+$ RuleML 1.01/XML and these rules have been tested and validated against the Relax NG schema for Datalog$^+$ in RuleML 1.01/XML. The usefulness of such rules for offshore holdings analytics is demonstrated here via incremental step-by-step approach, through the discovery of relationships between persons hiding large sums of assets and the offshore companies where the assets are purported to be hidden.

1 Introduction

The International Consortium of Investigative Journalists (ICIJ) [1] has exposed the details of ownership of more than 100,000 offshore entities located in 'tax havens' in 2013. The publicly released Offshore Leaks Database [2] shows relationships and networks among people or companies and offshore entities, often hiding their true owners. The database contains data on the companies and people identified as a director, a shareholder, a beneficiary, or a trustee. Precisely, ten offshore jurisdictions have been investigated: British Virgin Islands, Cayman Islands, Cook Islands, Singapore, Hong Kong, Samoa, Seychelles, Mauritius, Labuan, and Malaysia.

In this article, we demonstrate the effectiveness of expressive Datalog$^+$ RuleML 1.01 [3] by showcasing a rulebase containing a set of rules and queries over a subset of the Offshore Leaks Database. This rulebase will be used for analytics to infer new knowledge by deriving relations among different entities which are not easy to find manually. Our main motivation comes from the article published in [4] on big data visualization based on the database. In our rulebase, the formulation of rules have been simplified to the extent possible to ensure maximum comprehensibility to a broader user base and unambiguous interpretation of derived relations. Our method is introduced with an illustrative example, where

the goal is to find out if any connection can be established between the top-level officials such as the president or the prime minister of a country and their possible offshore investments, which would usually be managed via immediate family members or distant relatives and the intermediary companies they deal with. Here, it is assumed that top-level officials are very careful about exposing their holdings abroad and are not willing to be directly involved in a list of offshore holdings owners. Hence, any enquiry regarding their direct involvements in owning offshore holdings will almost certainly be in vein, and a search for indirect involvement might be more fruitful.

The article is organized as follows: In section 2, the formalization of our rulebase is illustrated in detail by authoring facts, rules, and executing queries on them, often by checking the consistency of the rulebase. In section 3, the expressivity and the effectiveness of rules used here have been briefly discussed. Finally, we conclude by summarizing the tools and procedures used for rule authoring, schema design, and their validation in Appendix A.

2 Rulebase

Our Datalog$^+$ RuleML 1.01/XML Document incrementally asserts into, retracts from, and queries, a rulebase (within the <RuleML> element) for a total of 37 transactions: 14 Asserts, 3 retracts, and 20 distinct Queries. The rules formalized in this document not only cover the smallest Datalog sublanguage, but also cover the highlights of RuleML 1.01, the superlanguage of the decidable Datalog$^{+/-}$ [5] called Datalog$^+$ [6] as well as Disjunctive Datalog features. The rules include all three Datalog extensions which allow existentially quantified variables, "Equal" predicate, and falsity in rule heads.

The fact *"The president of Azerbaijan is Ilham"* is formalized with a binary `president` relation in RuleML. A query immediately follows, formalizing *"Who is the president of Azerbaijan?"*, and resulting in an `x` binding to `Ilham`.

```
<?xml-model
  href=="http://deliberation.ruleml.org/1.01/relaxng/naffologeq_relaxed.rnc"
?>
<RuleML xmlns="http://ruleml.org/spec">
  <Assert>
    <Atom>
      <Rel>president</Rel>
      <Ind>Ilham</Ind>
      <Ind>Azerbaijan</Ind>
    </Atom>
  </Assert>
  <Query>
    <Atom>
      <Rel>president</Rel>
      <Var>x</Var>
      <Ind>Azerbaijan</Ind>
    </Atom>
  </Query>
```

A variation of the above query can be formalized as *"Which country does Ilham preside over?"*, resulting in a `y` binding to `Azerbaijan`.

```
<Query>
  <Atom>
    <Rel>president</Rel>
    <Ind>Ilham</Ind>
    <Var>y</Var>
  </Atom>
</Query>
```

A generic formalization of the above query can be expressed as *"List all the individuals and countries where the individual is the president of a country"*. Given the only ground fact, `x` and `y` represent the individual and the country binding to `Ilham` and `Azerbaijan`, respectively.

```
<Query>
  <Atom>
    <Rel>president</Rel>
    <Var>x</Var>
    <Var>y</Var>
  </Atom>
</Query>
```

The rule *"No one is both the president and the prime minister of a country"* is formalized as an integrity rule with falsity in rule head.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Implies>
      <if>
        <And>
          <Atom>
            <Rel>president</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
          <Atom>
            <Rel>primeMinister</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
        </And>
      </if>
      <then>
        <Or/>
      </then>
    </Implies>
  </Forall>
</Assert>
```

The formalization of the fact *"Ilham is the prime minister of Azerbaijan"*, when asserted with the previous rule, creates an inconsistency because of the already existing assertion *"Ilham is the president of Azerbaijan"*.

```
<Assert>
  <Atom>
    <Rel>primeMinister</Rel>
    <Ind>Ilham</Ind>
    <Ind>Azerbaijan</Ind>
  </Atom>
</Assert>
```

The inconsistency can be detected by the query *"Is there any inconsistency?"*. It succeeds because, according to the rule with falsity, `Ilham` cannot hold both the post of a president and a prime minister of `Azerbaijan`.

```
<Query>
  <Or/>
</Query>
```

To remove inconsistency from the rulebase, the fact *"Ilham is the prime minister of Azerbaijan"* is retracted.

```
<Retract>
  <Atom>
    <Rel>primeMinister</Rel>
    <Ind>Ilham</Ind>
    <Ind>Azerbaijan</Ind>
  </Atom>
</Retract>
```

The consistency of the rulebase is checked again by the same query. This time, the query fails and the rulebase regains consistency due to the retraction.

```
<Query>
  <Or/>
</Query>
```

Universally quantified equations are useful for expressing more complex formulas. The following rule *"Two persons have family ties if they are either married to each other, or there is a parent-child relationship between them, or they are distant relatives*" is formalized using the binary relations `hasFamilyTies`, `marriedTo`, `hasChild`, and `hasDistantRelative`. The facts *"Ilham is married to Mehriban*", *"Ilham has a daughter Arzu*", and *"Ilham has a daughter Leyla*" are formalized using the relations.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Var>y</Var>
    <Implies>
      <if>
        <Or>
          <Atom>
            <Rel>marriedTo</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
          <Atom>
            <Rel>hasChild</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
          <Atom>
            <Rel>hasDistantRelative</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
        </Or>
      </if>
      <then>
        <Atom>
```

```
          <Rel>hasFamilyTies</Rel>
          <Var>x</Var>
          <Var>y</Var>
        </Atom>
      </then>
    </Implies>
  </Forall>
  <Atom>
    <Rel>marriedTo</Rel>
    <Ind>Ilham</Ind>
    <Ind>Mehriban</Ind>
  </Atom>
  <Atom>
    <Rel>hasChild</Rel>
    <Ind>Ilham</Ind>
    <Ind>Leyla</Ind>
  </Atom>
  <Atom>
    <Rel>hasChild</Rel>
    <Ind>Ilham</Ind>
    <Ind>Arzu</Ind>
  </Atom>
</Assert>
```

The query, formalizing *"List the individuals having family ties to Ilham?"*, results in a `y` binding to three individuals named `Mehriban`, `Leyla`, and `Arzu`.

```
<Query>
  <Atom>
    <Rel>hasFamilyTies</Rel>
    <Ind>Ilham</Ind>
    <Var>y</Var>
  </Atom>
</Query>
```

On the other hand, the query *"Which individuals are tied to Mehriban as family members?"* can be formalized to retrieve `Ilham` using the `marriedTo` relation. Similarly, the binding of `x` to `Ilham` establishes family ties for both `Leyla` and `Arzu` because of the `hasChild` relation.

```
<Query>
  <Atom>
    <Rel>hasFamilyTies</Rel>
    <Var>x</Var>
    <Ind>Mehriban</Ind>
  </Atom>
</Query>

<Query>
  <Atom>
    <Rel>hasFamilyTies</Rel>
    <Var>x</Var>
    <Ind>Leyla</Ind>
  </Atom>
</Query>

<Query>
  <Atom>
    <Rel>hasFamilyTies</Rel>
    <Var>x</Var>
    <Ind>Arzu</Ind>
  </Atom>
</Query>
```

The query *"Find all individuals having family ties to each other"*, results in multiple combinations of `x` and `y` where `Ilham` binds to an `x` and each of `Mehriban`, `Leyla`, and `Arzu` binds to a `y`, respectively.

```
<Query>
  <Atom>
    <Rel>hasFamilyTies</Rel>
    <Var>x</Var>
    <Var>y</Var>
  </Atom>
</Query>
```

The rule *"Everything which operates either as an intermediary company or as an offshore company is designated as a company"* is formalized as a universally quantified rule with disjunctions in the rule body and it asserts a subclass relationship. Three ground facts *"ArborInvestmentsLtd operates as an intermediary company"*, *"NaziqAndPartners operates as an intermediary company"*, and *"HarvardManagementLtd operates as an offshore company"* are also formalized and added below.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Implies>
      <if>
        <Or>
          <Atom>
            <Rel>intermediaryCompany</Rel>
            <Var>x</Var>
          </Atom>
          <Atom>
            <Rel>offshoreCompany</Rel>
            <Var>x</Var>
          </Atom>
        </Or>
      </if>
      <then>
        <Atom>
          <Rel>company</Rel>
          <Var>x</Var>
        </Atom>
      </then>
    </Implies>
  </Forall>
  <Atom>
    <Rel>intermediaryCompany</Rel>
    <Ind>ArborInvestmentsLtd</Ind>
  </Atom>
  <Atom>
    <Rel>intermediaryCompany</Rel>
    <Ind>NaziqAndPartners</Ind>
  </Atom>
  <Atom>
    <Rel>offshoreCompany</Rel>
    <Ind>HarvardManagementLtd</Ind>
  </Atom>
</Assert>
```

The above rule implies that `ArborInvestmentsLtd`, `NaziqAndPartners`, and `HarvardManagementLtd` are all designated as companies. The formalization of

the query *"Which entities are designated as companies?"* results in `ArborInvestmentsLtd`, `NaziqAndPartners`, and `HarvardManagementLtd` as companies binding to the variable `x`.

```
<Query>
  <Atom>
    <Rel>company</Rel>
    <Var>x</Var>
  </Atom>
</Query>
```

From the above assertions, we can also find out if a particular company operates as an intermediary company or as an offshore company. For instance, the query *"Does ArborInvestmentsLtd operate as an intermediary company?"* succeeds as the company asserts as an intermediary company.

```
<Query>
  <Atom>
    <Rel>intermediaryCompany</Rel>
    <Ind>ArborInvestmentsLtd</Ind>
  </Atom>
</Query>
```

On the other hand, the query *"Does ArborInvestmentsLtd operate as an offshore company?"* fails as its operation does not assert as an offshore company but as an intermediary company.

```
<Query>
  <Atom>
    <Rel>offshoreCompany</Rel>
    <Ind>ArborInvestmentsLtd</Ind>
  </Atom>
</Query>
```

Another integrity rule with falsity in rule head is formalized as *"Nothing is both an onshore company and an offshore company"*.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Implies>
      <if>
        <And>
          <Atom>
            <Rel>onshoreCompany</Rel>
            <Var>x</Var>
          </Atom>
          <Atom>
            <Rel>offshoreCompany</Rel>
            <Var>x</Var>
          </Atom>
        </And>
      </if>
      <then>
        <Or/>
      </then>
    </Implies>
  </Forall>
</Assert>
```

The fact *"HarvardManagementLtd is an onshore company"* creates an inconsistency.

```
<Assert>
  <Atom>
    <Rel>onshoreCompany</Rel>
    <Ind>HarvardManagementLtd</Ind>
  </Atom>
</Assert>
```

Inconsistencies can be detected by the query *"Is there any inconsistency?"*.

```
<Query>
  <Or/>
</Query>
```

It succeeds because, according to the rule above, `HarvardManagementLtd` cannot be both an onshore company and an offshore company.

To remove inconsistency from the rulebase, the fact *"HarvardManagementLtd is an onshore company"* is retracted.

```
<Retract>
  <Atom>
    <Rel>onshoreCompany</Rel>
    <Ind>HarvardManagementLtd</Ind>
  </Atom>
</Retract>
```

When consistency of the rulebase is checked by the same query, it fails as the rulebase became consistent with the retraction.

```
<Query>
  <Or/>
</Query>
```

The rule *"It is unlikely that an offshore company reveals their clients' investment information"* is formalized as an example of a negative integrity constraint.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Implies>
      <if>
        <And>
          <Atom>
            <Rel>offshoreCompany</Rel>
            <Var>x</Var>
          </Atom>
          <Atom>
            <Rel>revealInvestmentInfo</Rel>
            <Var>x</Var>
          </Atom>
        </And>
      </if>
      <then>
        <Or/>
      </then>
    </Implies>
  </Forall>
</Assert>
```

The fact *"HarvardManagementLtd reveals investment information"* creates inconsistency when added to the rulebase.

```
<Assert>
  <Atom>
    <Rel>revealInvestmentInfo</Rel>
    <Ind>HarvardManagementLtd</Ind>
  </Atom>
</Assert>
```

The query *"Is there any inconsistency?"* succeeds because `HarvardManagementLtd` does not reveal investment information while operating as an offshore company.

```
<Query>
  <Or/>
</Query>
```

The fact *"HarvardManagementLtd reveals investment information"* is retracted to remove inconsistency from the rulebase.

```
<Retract>
  <Atom>
    <Rel>revealInvestmentInfo</Rel>
    <Ind>HarvardManagementLtd</Ind>
  </Atom>
</Retract>
```

Now that the consistency is checked again by the same query, it fails.

```
<Query>
  <Or/>
</Query>
```

The following assertion contains a rule and a ground fact. The rule *"Anyone is a shareholder of a company if and only if there exists a certain level of stocks of that company such as Small-cap level or Medium-cap level or Large-cap level, which he owns"* is formalized as an equivalence formula. The ground fact *"Arzu owns Medium-cap level of stocks in ArborInvestmentsLtd"* is formalized and added below.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Var>y</Var>
    <Equivalent>
      <Atom>
        <Rel>shareHolderOf</Rel>
        <Var>x</Var>
        <Var>y</Var>
      </Atom>
      <Exists>
        <Var>z</Var>
        <Atom>
          <Rel>ownsStockin-atLevel</Rel>
          <Var>x</Var>
          <Var>y</Var>
          <Var>z</Var>
        </Atom>
      </Exists>
    </Equivalent>
  </Forall>
  <Atom>
    <Rel>ownsStockin-atLevel</Rel>
    <Ind>Arzu</Ind>
```

```
    <Ind>ArborInvestmentsLtd</Ind>
    <Ind>Medium-cap</Ind>
  </Atom>
</Assert>
```

The benefit of this equivalence formula is that the `ownsStockin-atLevel` relation can be used when the level of stock ownership is important, otherwise `shareHolder` relation is sufficient as it does not care about the level of stocks to become a shareholder.

The `ownsStockin-atLevel` relation is defined as a functional relation, hence together with its existential formalization in the equivalence formula above, it must have uniqueness. This is achieved by formalizing the rule *"Every stock owner owns at most one level of stocks of a company"* with equality in the head.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Var>y</Var>
    <Var>z1</Var>
    <Var>z2</Var>
    <Implies>
      <if>
        <And>
          <Atom>
            <Rel>ownsStockin-atLevel</Rel>
            <Var>x</Var>
            <Var>y</Var>
            <Var>z1</Var>
          </Atom>
          <Atom>
            <Rel>ownsStockin-atLevel</Rel>
            <Var>x</Var>
            <Var>y</Var>
            <Var>z2</Var>
          </Atom>
        </And>
      </if>
      <then>
        <Equal>
          <Var>z1</Var>
          <Var>z2</Var>
        </Equal>
      </then>
    </Implies>
  </Forall>
</Assert>
```

The query *"Is there any individual who owns Small-cap level of stocks in ArborInvestmentsLtd?"* fails because there are no individuals binding to an `x` who own `Small-cap` level of stocks. A similar query with a different quantity `Large-cap` in place of `Small-cap` also fails.

```
<Query>
  <Atom>
    <Rel>ownsStockin-atLevel</Rel>
    <Var>x</Var>
    <Var>ArborInvestmentsLtd</Var>
    <Var>Small-cap</Var>
  </Atom>
</Query>
```

On the other hand, the query *"Is there any individual who owns Medium-cap level of stocks in ArborInvestmentsLtd?"* succeeds by providing the answer `Arzu` as the owner binding to an `x`.

```
<Query>
  <Atom>
    <Rel>ownsStockin-atLevel</Rel>
    <Var>x</Var>
    <Var>ArborInvestmentsLtd</Var>
    <Var>Medium-cap</Var>
  </Atom>
</Query>
```

The query *"List all shareholders of ArborInvestmentsLtd"* provides the individual `Arzu` as the only shareholder binding to an `x`.

```
<Query>
  <Atom>
    <Rel>shareHolderOf</Rel>
    <Var>x</Var>
    <Var>ArborInvestmentsLtd</Var>
  </Atom>
</Query>
```

The rule *"There is a link between a person and a company if the person is a director or a shareholder of the company or he owns a level of stock in the company"* is formalized as a universally quantified rule with disjunctions in rule body.

```
<Forall>
    <Var>x</Var>
    <Var>y</Var>
    <Implies>
      <if>
        <Or>
          <Atom>
            <Rel>directorOf</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
          <Atom>
            <Rel>shareHolderOf</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
          <Exists>
            <Var>z</Var>
            <Atom>
              <Rel>ownsStockin-atLevel</Rel>
              <Var>x</Var>
              <Var>y</Var>
              <Var>z</Var>
            </Atom>
          </Exists>
        </Or>
      </if>
      <then>
        <Atom>
          <Rel>hasLinksTo</Rel>
          <Var>x</Var>
          <Var>y</Var>
        </Atom>
      </then>
```

```
    </Implies>
  </Forall>
</Assert>
<Query>
  <Atom>
    <Rel>hasLinksTo</Rel>
    <Var>x</Var>
    <Var>y</Var>
  </Atom>
</Query>
```

The query *"Which individuals and companies are linked to each other?"* results in the name `Arzu` binding to the variable `x` and the company `ArborInvestmentsLtd` to a `y`.

The rule *"If a company C_1 manages assets of another company C_2 and the latter i.e. C_2 manages assets of yet another company C_3, then company C_1 manages the assets of the company C_3"* is formalized as a universally quantified rule implying a transitive relation. With this rule, relations between two distant companies can be established even if there is a chain of companies sitting in the middle managing assets. The facts *"ArborInvestmentsLtd manages assets of NaziqAndPartners"* and *"NaziqAndPartners manages assets of HarvardManagementLtd"* are formalized using the relation `manageAssets` and are added at the end of the assertion.

```
<Assert>
  <Forall>
    <Var>x</Var>
    <Var>y</Var>
    <Var>z</Var>
    <Implies>
      <if>
        <And>
          <Atom>
            <Rel>manageAssets</Rel>
            <Var>x</Var>
            <Var>y</Var>
          </Atom>
          <Atom>
            <Rel>manageAssets</Rel>
            <Var>y</Var>
            <Var>z</Var>
          </Atom>
        </And>
      </if>
      <then>
        <Atom>
          <Rel>manageAssets</Rel>
          <Var>x</Var>
          <Var>z</Var>
        </Atom>
      </then>
    </Implies>
  </Forall>
  <Atom>
    <Rel>manageAssets</Rel>
    <Ind>ArborInvestmentsLtd</Ind>
    <Ind>NaziqAndPartners</Ind>
  </Atom>
  <Atom>
    <Rel>manageAssets</Rel>
    <Ind>NaziqAndPartners</Ind>
    <Ind>HarvardManagementLtd</Ind>
```

```
  </Atom>
</Assert>
```

From the transitive rule and two facts mentioned above, it can be deduced that the company `ArborInvestmentsLtd` is, in fact, managing assets of the distant company `HarvardManagementLtd`. The query *"Find all the companies whose assets are being managed and the companies managing them"* can be used to infer such relations among companies.

```
<Query>
  <Atom>
    <Rel>manageAssets</Rel>
    <Var>x</Var>
    <Var>z</Var>
  </Atom>
</Query>
```

Finally, by formalizing the rule *"The president of a country (possibly) has offshore investments in a company if his family members have links to companies managing assets in that offshore company"*, it can be discovered if a president has invested in an offshore company to hide his assets using someone in his family.

```
<Assert>
  <Forall>
    <Var>p</Var>
    <Var>c</Var>
    <Var>fm</Var>
    <Var>ic</Var>
    <Var>oc</Var>
    <Implies>
      <if>
        <And>
          <Atom>
            <Rel>president</Rel>
            <Var>p</Var>
            <Var>c</Var>
          </Atom>
          <Atom>
            <Rel>hasFamilyTies</Rel>
            <Var>p</Var>
            <Var>fm</Var>
          </Atom>
          <Atom>
            <Rel>hasLinksTo</Rel>
            <Var>fm</Var>
            <Var>ic</Var>
          </Atom>
          <Atom>
            <Rel>manageAssets</Rel>
            <Var>ic</Var>
            <Var>oc</Var>
          </Atom>
        </And>
      </if>
      <then>
        <Atom>
          <Rel>possiblyHasOffshoreInvestmentIn</Rel>
          <Var>p</Var>
          <Var>oc</Var>
        </Atom>
      </then>
    </Implies>
  </Forall>
</Assert>
```

Based on the facts and rules provided, the query *"Find the presidents and companies in which the presidents possibly have offshore investments"* is formalized to find the presidents who have invested in offshore companies to hide their assets from the concerned national and international authority.

```
  <Query>
    <Atom>
      <Rel>possiblyHasOffshoreInvestments</Rel>
      <Var>p</Var>
      <Var>oc</Var>
    </Atom>
  </Query>
</RuleML>
```

The answer to the above query is president `Ilham` of `Azerbaijan` whose daughter `Arzu` has links with the company `ArborInvestmentsLtd`, which indirectly manages the assets in the offshore company called `HarvardManagementLtd`.

3 Conclusion

A rulebase formulated using Datalog$^+$ RuleML 1.01/XML rules is presented in this article. The incremental formalization of rules demonstrates how interesting relationships can be discovered among different entities from financial records, which are difficult to deduce in plain sight. The discovery of relationships is used for offshore holdings analytics.

Our rulebase is comprised of expressive RuleML 1.01 rules which, in addition to the existing RuleML sublanguages, cover two newly introduced anchor sublanguages called *datalogplus_min* and *disdatalogplus_min*. Thus our rules not only include positive facts and universally quantified implications, but also allows existential variables, "Equal" predicates, and falsity in the heads of implications. Moreover, conjunctions within existetials and non-empty disjunctions, arbitrarily nested with existentials and conjunctions are permitted in the heads of implications. Some of the rules in our rulebase have disjunctions both in the body and head of implications.

With the addition of new features from the two sublanguages, expressive rules are now easier to formulate in RuleML 1.01. The ability to use existential variables in the heads of implications lets us make some progress in solving old problems in semantic technologies. For instance, an earlier attempt mentioned in [7] for combining negation-free rules and ontologies needed a workaround called *DL-safety* because those rules did not allow existentials in heads. As RuleML 1.01 permits existentials in heads of implications, our intuition is that DL-safety won't be required if RuleML 1.01 rules are used instead of negation-free rules. Further research in this direction would be interesting and is left for future work. Moreover, because of the modularity in RuleML sublanguages, fine-grained rules can be authored, which might fall into a customized sublanguage. The availability of these choices gives the user the advantage of identifying appropriate resources for the sublanguage such as the inference engine saving significant time.

A Rulebase Authoring, Schema Design, and Validation

For authoring our rulebase[1], we used XML Copy Editor[2] which is a free software released under the GNU General Public License. The RuleML community has provided Modular sYNtax confiGurator (MYNG 1.01)[3], a highly customizable GUI for generating RELAX NG Compact Syntax (RNC) schemas or approximating XSD schemas. We formalized the rulebase based on RNC schemas and validated using a validation webservice[4]. The webservice can validate Datalog$^+$ RuleML 1.01/XML instances against schemas, including Relax NG schemas and Namespace-Based Validation Dispatching Language NVDL[5]. We validated every assert and query separately using the webservice against the appropriate RNC schema and retracted any assertions causing inconsistency in the rulebase. Finally, the complete rulebase containing all asserts, retracts, and queries is validated against the RNC schema.

References

1. The International Consortium of Investigative Journalists. `http://www.icij.org/`.
2. ICIJ Offshore Leaks Database. `http://offshoreleaks.icij.org/about/caveats`.
3. Datalog+ RuleML 1.01. `http://deliberation.ruleml.org/1.01`.
4. Analysing the Offshore Leaks with graphs. `http://linkurio.us/analysing-the-offshore-leaks-with-graphs/`.
5. Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09, pages 77–86, New York, NY, USA, 2009. ACM.
6. Georg Gottlob, Giorgio Orsi, Andreas Pieris, and Mantas Simkus. Datalog and its extensions for semantic web databases. In Thomas Eiter and Thomas Krennwallner, editors, *Reasoning Web*, volume 7487 of *Lecture Notes in Computer Science*, pages 54–77. Springer, 2012.
7. Mohammad Sadnan Al Manir. Towards rif-owl combination: An effective reasoning technique in integrating owl and negation-free rules. In Frank Olken, Monica Palmirani, and Davide Sottara, editors, *RuleML America*, volume 7018 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2011.

[1] http://deliberation.ruleml.org/1.01/exa/RulebaseCompetition2014/OffshoreHoldingAnalytics.ruleml
[2] http://xml-copy-editor.sourceforge.net/
[3] http://deliberation.ruleml.org/1.01/myng/
[4] http://validator.nu
[5] http://nvdl.org