Faculty of Computer Science

CS6795 Semantic Web Techniques

October 29th, 2012

Project proposal

"Testing, inverting and round tripping RON Normalizer for RuleML 1.0 in XSLT 2.0"

Instructor: Dr. Harold Boley

Advisor: Dr. Tara Athan

Team: Simranjit Singh

Bijiteshwar R Aayush

Pratik Shah

1. Introduction

RULEML is the canonical Web language for rules using XML markup, formal semantics, and efficient implementations. **RULEML** ***is a unifying family of XML-serialized rule languages spanning across all industrially relevant kinds of Web rules.***

RULEML ***1.0 rules and queries are marked up (serialized) in an 'object-oriented' flavor of XML, where XML trees alternate elements that act as Classes/Types (Nodes) or methods/roles (edges)***. There are various equivalent ways of such (Node-edge-… -Node-edge-Node-) 'striped' XML serialization, even apart from logical equivalences. For example, the <Implies> node (the main element for rules) normally contains an <if> and a <then> edge (sub elements for the rule premise and conclusion, respectively). But, when written in this natural 'if-then' sequence, the left-to-right order of ***XML sub elements can be relied upon, and the stripe of <if> and <then> edges may be omitted.***

NORMALIZER: As it is easily understood from the name itself the Normalizer is a tool which is used to normalize the format of the Rules in XML. By normalization we mean to give the exact shape to the XML trees where each element is a rule so that it can be easily understood and implemented. The normalizer makes the elements of XML both syntactically and canonically correct by placing all the stripes in proper canonical order which is the fully expanded form. The current normalizer is the XSLT 2.0 tool developed by the team of previous year. Our normalizer will also work on the current version of rule ML i.e. 1.0.

2. Objective

There are three major objectives which we will obtain with the course of this project.

A. In our first part our task is to test the existing normalizer against some defined test cases.

B. This is the major part of our project where we will develop a Compactifier for RuleML

which will invert the existing normalizer. ***Due to its functioning we have named our tool as ROC: RuleML official Compactifier***. The ROC will first check the missing stripes in the XML and will place them wherever needed. Then it will check the canonical order of the stripes in the element. After the canonical check is performed it will remove all the stripes from the element to convert it into a fully skipped form which will help in obtaining our objective to make the XML element fully compact as is explicit from the name. We will be implementing our tool in the XSLT 2.0.

C. The inverse form obtained from the second step will be composed for round tripping which will lead to improvement of both the suites obtained from a and b part.

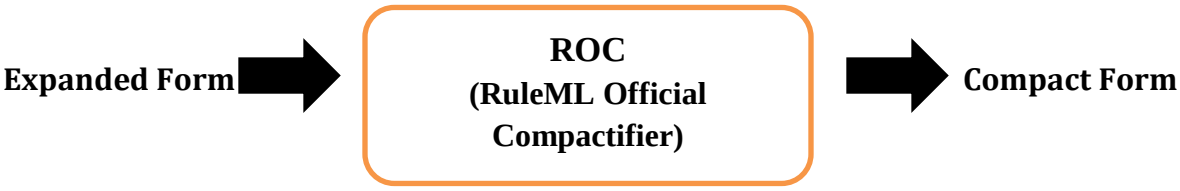
3. Project Life-Cycle

3.1 Working Model

First of all we will be testing **RON**. The input here will be in the form, known as **Compact Form** is now passed through RON. RON will convert the input of Compact Form into a different form known as **Expanded Form**. This form now has no missing stripes and it is in a proper canonical order. This is also called as Fully Striped Form. We will also use some partially skipped inputs also.

The major part of our project will include the **Expanded Form** of data is passed through the **ROC** (RuleML official Compactifier). ROC will convert the same **Expanded Form** back to the **Compact Form** i.e. it will remove all the stripes and present the data in a stripe-skipped form.

But before doing this we have to work on ordering of the elements. The elements and their sub-elements have to be in the canonical order, which is already proposed in the normalizer schema. So, here it will build canonically ordered content of each element



The last part of our project is to compose the round tripping of the ROC by considering all the cases which can be taken as input and the output will be checked for that.

3.2 Working Details

3.2.1 RON

We will input data of Compact Form and the RON will convert the input into Expanded Form. In between it will also check for missing stripes and the canonical ordering. The output will be Fully Striped form of the input.

Input: Compact Form

```
<Assert mapClosure="universal">
                <Implies>
                        <Atom>...</Atom>
                                <Atom>
                                        <then>...</then>
                                        <if>...</if>
                                </Atom>

                </Implies>
                                <Atom>...</Atom>
</Assert>
```

Output: Expanded Form

```
<Assert mapClosure="universal">
                <Implies>
                        <if>
                            <Atom>
                                        <And>...</And>
```

```
                    </Atom>
                </if>
                    <then>
                        <Atom>…</Atom>
                    </then>
            </Implies>
</Assert>
```

3.2.2 ROC

We will implement this new kind of normalizer which we now call as Compactifier, whose job will be to transform the Expanded Form of data into the Compact Form. In between it will also check for any missing stripes and the canonical ordering. In this Compactifier we will input a Fully Striped Form of data and ROC will convert it to the Stripe Skipped Form.

Input: Expanded Form

```
<Assert mapClosure="universal">
            <Implies>
                <if>
                    <Atom>
                        <And>...</And>
                    </Atom>
                </if>
                    <then>
                        <Atom>…</Atom>
                    </then>
            </Implies>
</Assert>
```

Output: Compact Form

```
<Assert mapClosure="universal">
            <Implies>
                <Atom>...</Atom>
                <Atom>...</Atom>
            </Implies>
</Assert>
```

4. Project tools

1. To test the output of our transformations we will use a validator for XML Schema at http://www.w3.org/2001/03/webdata/xsv

2. **XSLT** (Extensible Style sheet Language) is a style sheet language for XML documents. XPath and XSLT are both parts of the XSL. For the preparation of this project, we need to know the basic knowledge XSLT, which includes the content of XSLT element like "apply-templates", "for-each" and "value-of".

3. **RuleML (**http://www.ruleml.org/) is a markup language for sharing rules in XML markup. Originally, it was specified by DTDs, but is now specified by XSD schemas.

4. **RON** (http://ruleml.org/1.0/xslt/normalizer/) is a RuleML official Normalizer.

5. **Oxygen XML editor.**

5. References

XSLT 2.0

http://www.xfront.com/rescuing-xslt.html
http://www.cafeconleche.org/books/bible3/chapters/ch15.html
http://www.w3.org/2005/08/online_xslt

RuleML
http://ruleml.org/

RuleML 1.0 Normalization
http://ruleml.org/1.0/xslt/normalizer/100_normalizer.xslt