Faculty of Computer Science

CS6795 Semantic Web Techniques

Project Report

"A Test Case Suite for Hornlog+ RuleML 1.01"

Professor: Dr. Harold Boley
Advisor: Dr. Tara Athan

Team: Zhenzhi Cui
Radhika Yadav

ACKNOWLEDGEMENT

We hereby acknowledge and extend our heartiest thanks to our professor Dr. Harold Boley and advisor Dr. Tara Athan for their guidance and motivation throughout this project.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 RuleML

RuleML is knowledge representation language which is conceived for the interchange of the major kinds of Web rules in an XML format that is uniform across various rule logics and platforms. It is defined as an extensible family of sublanguages and has broad coverage. Its modular construction of schemas allows rule interchange with high accuracy [1].

1.2 Deliberation RuleML

Deliberation RuleML incorporates a family of sublanguages, commencing with Datalog, Hornlog, and other Derivation Rules, and advancing to FOL and beyond. The outstanding aspect of Deliberation RuleML 1.01 is the possibility to amalgamate one or more of the following Datalog extensions which jointly constitute Datalog+:

- **Existential Rule:** The "then" part of a rule has existentially quantified variables, as needed for description logics, F- logic and PSOA RuleML, Rule-Based Data Access, etc.
- **Equality Rule:** The "then" part of a rule is the "Equal" predicate, as needed for user-defined/'semantic' equality in logics with equality and functional logic programming (this was already allowed in RuleML 1.0).
- **Integrity Rule:** The "then" part of a rule is falsity, as an appropriate way to express negative integrity constraints [1].

In Deliberation RuleML 1.01, there are new options for obtaining a more fine-grained customization of sublanguages. It has the potential to combine one or more of the Datalog extensions jointly making up Datalog+.

In Deliberation RuleML 1.01, each of these Datalog+ features can now also be combined with a *conjunction* (e.g., an "And" of "Atoms", rather than just one "Atom") in the "then" part.

Deliberation RuleML 1.01 also allow mixing in the characteristic Disjunctive Datalog feature of *disjunction* (e.g., an "Or" of "Atoms") in the "then" part.

1.3 MYNG 1.01

MYNG 1.01 is a tool that is used to generate the Relax NG schemas. In MYNG 1.01 we have different radio buttons and checkboxes available that are used to select the particular specifications in accordance to sublanguage.

Key new features in MYNG 1.01 include:

- Integration of new Relax NG schema modules -- and the RuleML sublanguages they define -- into MYNG, e.g.
    - Datalog+, Hornlog$^+$ and their many extensions.
- Improved functionality of the MYNG GUI and REST interface, e.g.
    - GUI access to automatically-generated monolithic XSD schemas that are compatible with XML tools, e.g. JAXB.
    - Display of myng-code and myng-code-based schema URLs. [6]

So, if we need to generate a Relax NG schema for Hornlog+ we have to select the radio buttons enabling it to *Hornlogic, Equations, Conjunctive Heads, Existential Heads and Negative constraints*. Then we refresh the schema, and we get a URL with the backbone. When we click on that URL, we see that the related schema file is downloaded; this can be used by us in the further process of validating the ruleml file.

We have generated schemas for sublanguages like *Binhornlog+* (a binary Hornlog+), *Dishornlog+*, *Nafhornlog+*, *Neghornlog+ and Nafnegdishornlog+*, where we selected the appropriate radio buttons in the MYNG 1.01 tool and refreshed the schema so that the related sublanguage schema is generated by its URL accordingly.

Below we show one such URL, of the Hornlog+ schema, that is generated using the MYNG 1.01:
http://deliberation.ruleml.org/1.01/relaxng/schema_rnc.php?backbone=xf&default=x7&termseq=x7&lng=x1&propo=x3cf&implies=x5f&terms=xf1f&quant=x7&expr=x3&serial=xf

We have used the above schema to validate the Hornlog+ examples in *validator.nu*. [7]

2. PROJECT DESCRIPTION AND IMPLEMENTATION

2.1 Methodology

The following discussion is visualized in Fig.1.

Firstly, to create documents of smaller (less expressive) sublanguages, we will remove some features from a Hornlog+ document and consider Datalog+, Hornlog and Binhornlog+ documents. The corresponding schemas are created by using the MYNG 1.01 tool that is developed by Dr. Tara Athan. We do this by selecting the desired checkboxes according to the sublanguages used. The generated schema is then used to validate the documents using validator.nu.

Then, we will add features to develop the Hornlog+ document to make it Dishornlog+, Nafhornlog+, Neghornlog+ and Nafnegdishornlog+ documents. These are larger (more expressive) sublanguages which still fall under the parent category of Deliberation RuleML 1.01.

Finally, we will characterize the new documents with their validation results and screenshots as examples or counter-examples of, respectively, larger/equal or smaller sublanguages. This is visualized in Fig.2.

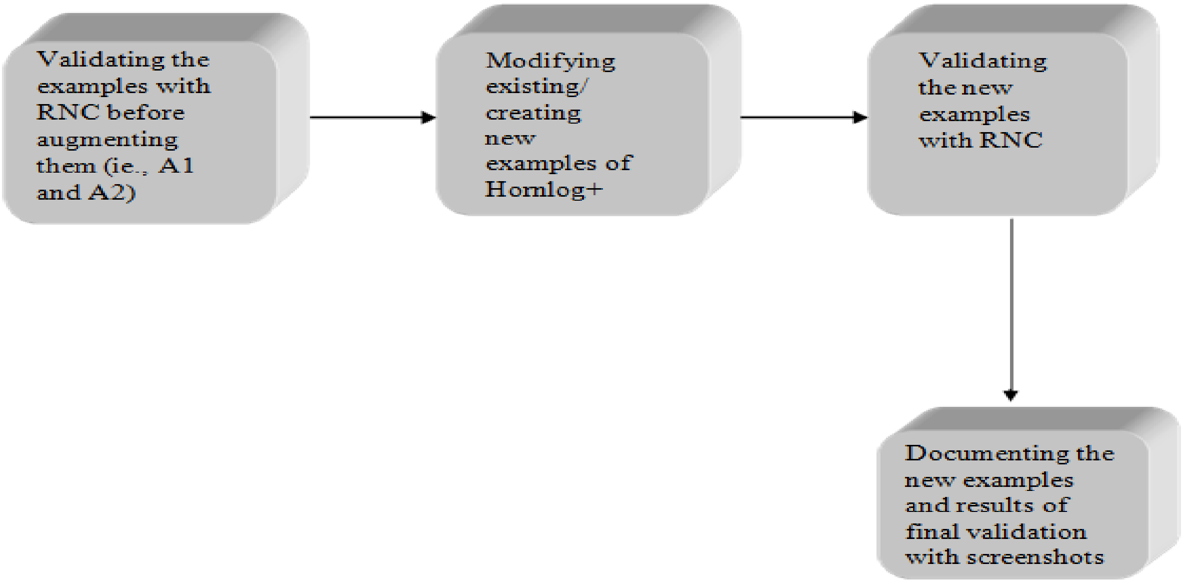

Validating the examples with RNC before augmenting them (ie., A1 and A2)
Modifying existing/ creating new examples of Hornlog+
Validating the new examples with RNC
Documenting the new examples and results of final validation with screenshots

Fig.1. Project Work Flow

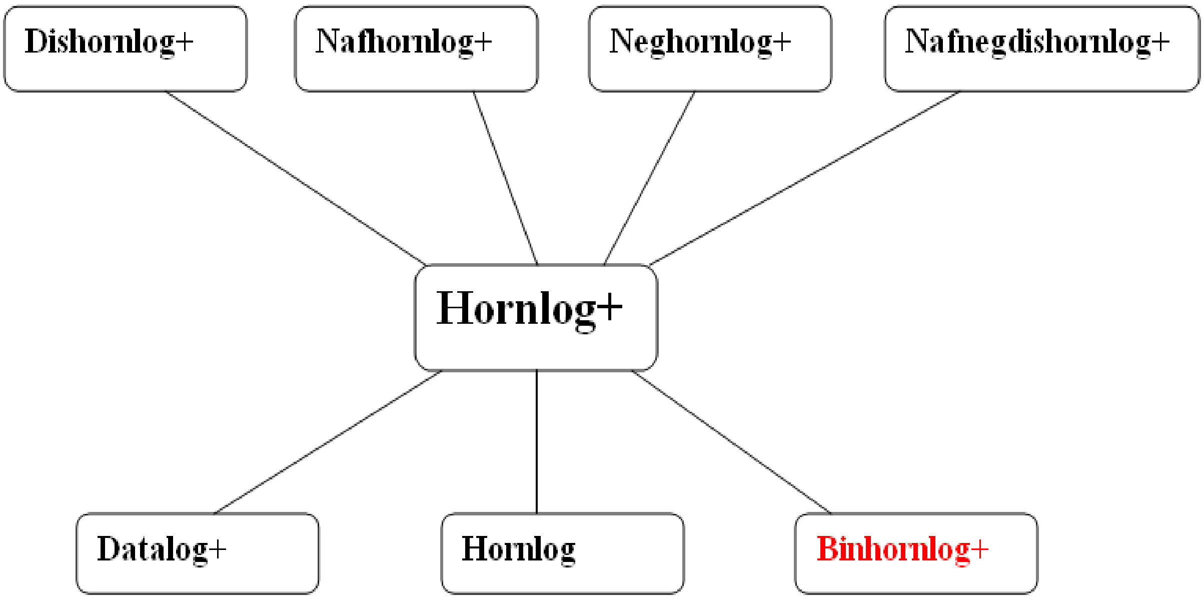


Fig.2. Hornlog+ with smaller and larger sublanguages
(Binhornlog+ is in red: not- (yet-)-anchor languages are highlighted as such)

2.2 Testing in Validator.nu

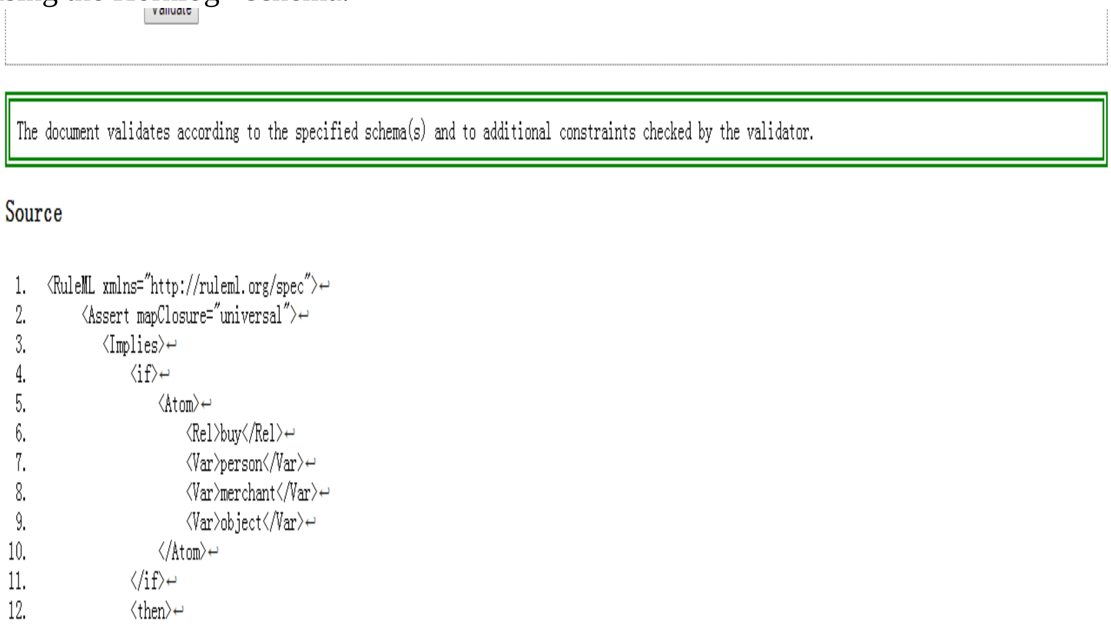
Here we have some documents that are Hornlog+ examples and some documents are counter-example for Hornlog+; meanwhile the Hornlog+ example are also counter-examples for the Binhornlog+, Datalog+ and Hornlog+.

We have used *Oxygen* editor for modification and creation of RuleML documents.

Consider this buy.ruleml document:

```
<RuleML xmlns="http://ruleml.org/spec">
  <Assert mapClosure="universal">
    <Implies>
      <if>
        <Atom>
          <Rel>buy</Rel>
          <Var>person</Var>
          <Var>merchant</Var>
          <Var>object</Var>
        </Atom>
      </if>
      <then>
        <Exists>
          <Var>retailer</Var>
          <And>
            <Atom>
              <Rel>buy</Rel>
              <Var>merchant</Var>
              <Var>retailer</Var>
              <Var>object</Var>
            </Atom>
            <Atom>
              <Rel>buy</Rel>
              <Var>retailer</Var>
              <Expr>
                <Fun>factoryOf</Fun>
                <Var>retailer</Var>
              </Expr>
              <Var>object</Var>
            </Atom>
          </And>
        </Exists>
      </then>
    </Implies>
  </Assert>
</RuleML>
```

The buy.ruleml document is a Hornlog+ example. The Fig.3 shows the validation result using the Hornlog+ schema.



The document validates according to the specified schema(s) and to additional constraints checked by the validator.

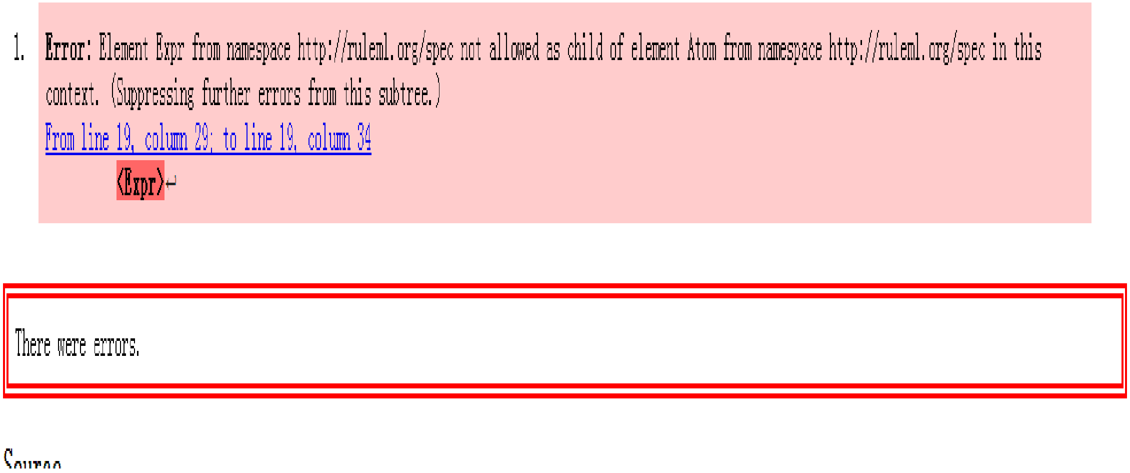
Source

```
1.  <RuleML xmlns="http://ruleml.org/spec">↵
2.      <Assert mapClosure="universal">↵
3.          <Implies>↵
4.              <if>↵
5.                  <Atom>↵
6.                      <Rel>buy</Rel>↵
7.                      <Var>person</Var>↵
8.                      <Var>merchant</Var>↵
9.                      <Var>object</Var>↵
10.                 </Atom>↵
11.             </if>↵
12.             <then>↵
```

Fig.3

The buy.ruleml document is also a counter-example for Datalog+, Hornlog and Binhornlog+.

The validation result for the buy.ruleml document when we tried to validate it with the Datalog+ schema is shown in Fig.4.



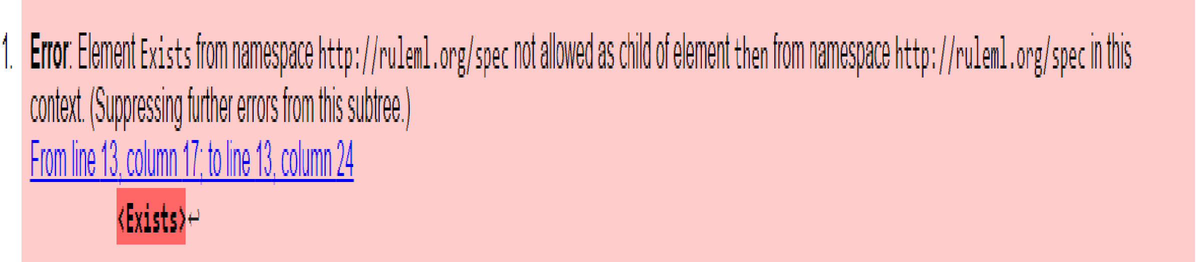
1. **Error:** Element Expr from namespace http://ruleml.org/spec not allowed as child of element Atom from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
From line 19, column 29; to line 19, column 34
        <Expr>↵

There were errors.

Fig.4

So, the buy.ruleml document is a counter-example for Datalog+, because of its function application (Expr) characteristic (from Hornlog+).

The validation result for the buy.ruleml document when we tried to validate it with the Hornlog schema is shown in Fig.5.

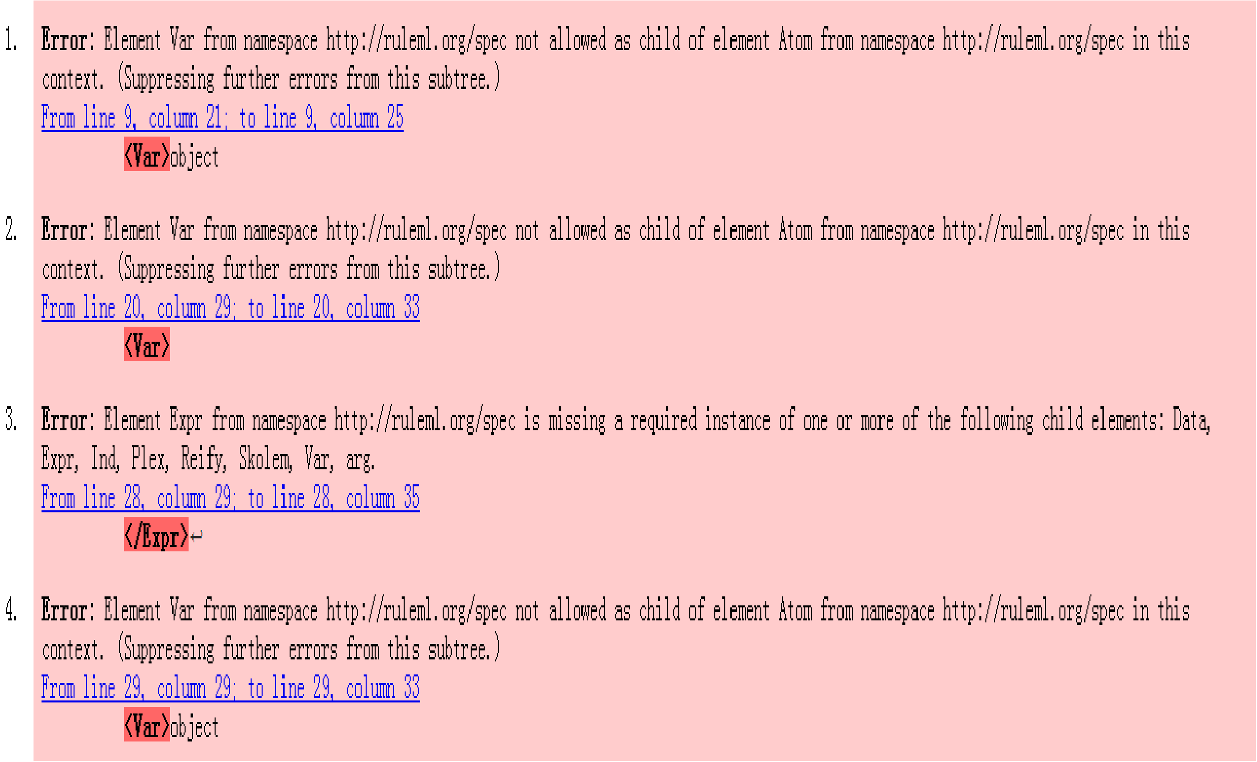


```
1. Error: Element Exists from namespace http://ruleml.org/spec not allowed as child of element then from namespace http://ruleml.org/spec in this
   context. (Suppressing further errors from this subtree.)
   From line 13, column 17; to line 13, column 24
           <Exists>↵
```

Fig.5

The buy.ruleml document is also a counter-example for Hornlog, because of its Existential characteristic (from Hornlog+).

The validation result for the buy.ruleml document when we tried to validate it with the Binhornlog+ schema is shown in Fig.6.



```
1. Error: Element Var from namespace http://ruleml.org/spec not allowed as child of element Atom from namespace http://ruleml.org/spec in this
   context. (Suppressing further errors from this subtree.)
   From line 9, column 21; to line 9, column 25
           <Var>object

2. Error: Element Var from namespace http://ruleml.org/spec not allowed as child of element Atom from namespace http://ruleml.org/spec in this
   context. (Suppressing further errors from this subtree.)
   From line 20, column 29; to line 20, column 33
           <Var>

3. Error: Element Expr from namespace http://ruleml.org/spec is missing a required instance of one or more of the following child elements: Data,
   Expr, Ind, Plex, Reify, Skolem, Var, arg.
   From line 28, column 29; to line 28, column 35
           </Expr>↵

4. Error: Element Var from namespace http://ruleml.org/spec not allowed as child of element Atom from namespace http://ruleml.org/spec in this
   context. (Suppressing further errors from this subtree.)
   From line 29, column 29; to line 29, column 33
           <Var>object
```

Fig.6

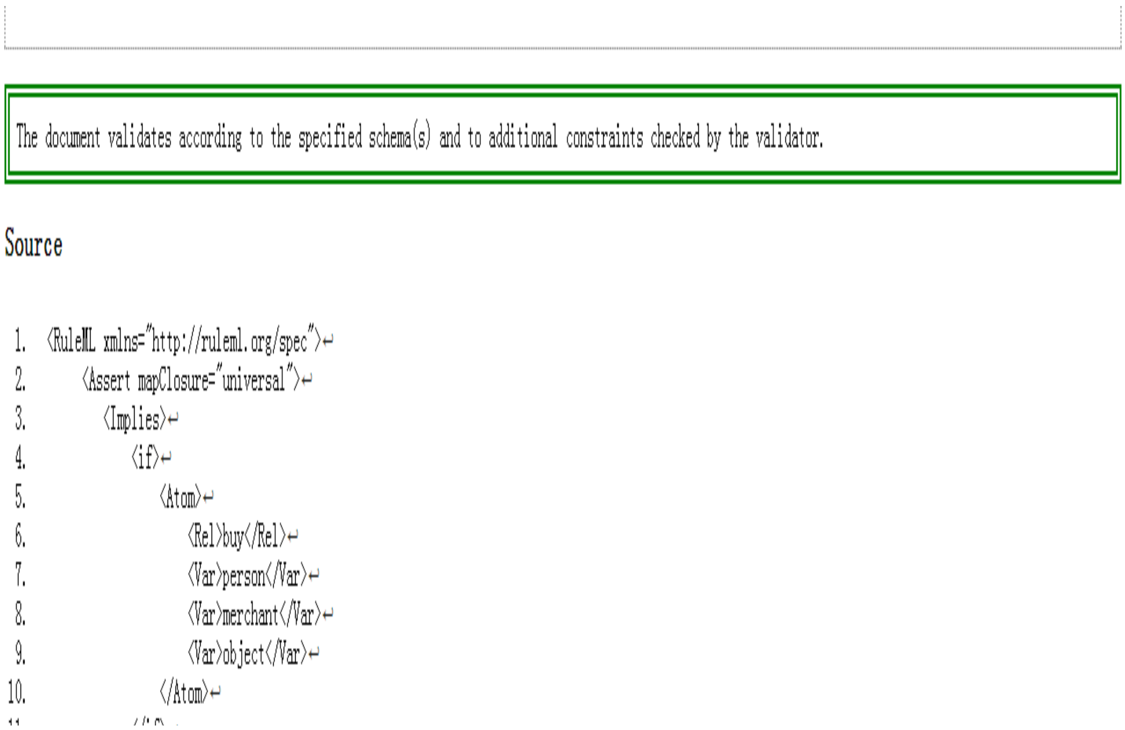
The buy.ruleml document is a counter-example for Binhornlog+, because of its zero or more characteristics. In Binhornlog+, there can be only zero or two arguments.

The buy.ruleml document is not only a counter-example for Datalog+, Hornlog and Binhornlog+ but also of all their smaller sublanguages such as Datalog.

Below is the buy1.ruleml document which is a Dishornlog+ example:

```
<RuleML xmlns="http://ruleml.org/spec">
  <Assert mapClosure="universal">
    <Implies>
      <if>
        <Atom>
          <Rel>buy</Rel>
          <Var>person</Var>
          <Var>merchant</Var>
          <Var>object</Var>
        </Atom>
      </if>
      <then>
        <Exists>
          <Var>retailer</Var>
          <Or>
            <Atom>
              <Rel>buy</Rel>
              <Var>merchant</Var>
              <Var>retailer</Var>
              <Var>object</Var>
            </Atom>
            <Atom>
              <Rel>buy</Rel>
              <Var>retailer</Var>
              <Expr>
                <Fun>factoryOf</Fun>
                <Var>retailer</Var>
              </Expr>
              <Var>object</Var>
            </Atom>
          </Or>
        </Exists>
      </then>
    </Implies>
  </Assert>
</RuleML>
```

The validation result for the buy1.ruleml document when we validated it with the Dishornlog+ schema is shown in Fig.7



The document validates according to the specified schema(s) and to additional constraints checked by the validator.

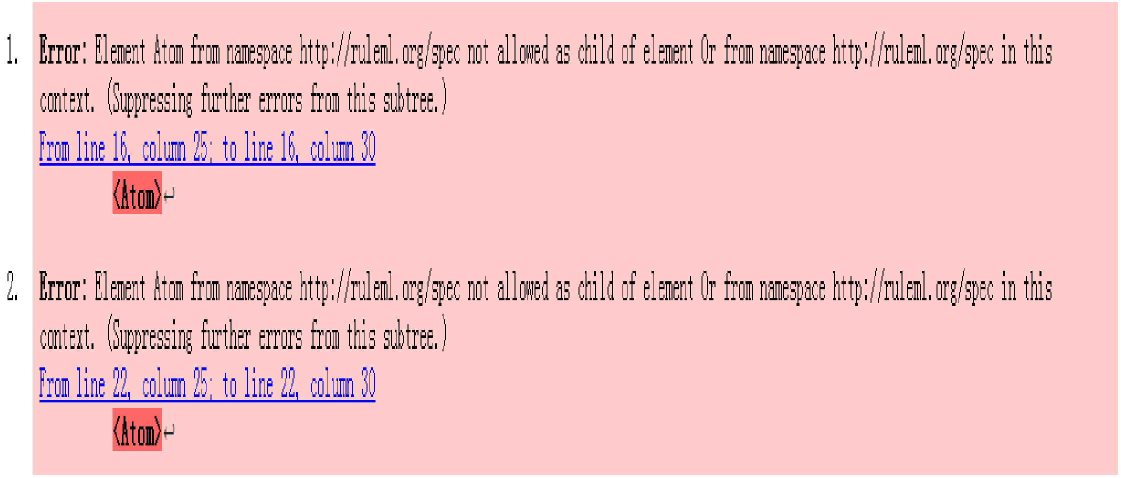
Source

```
<RuleML xmlns="http://ruleml.org/spec">
   <Assert mapClosure="universal">
      <Implies>
         <if>
            <Atom>
               <Rel>buy</Rel>
               <Var>person</Var>
               <Var>merchant</Var>
               <Var>object</Var>
            </Atom>
```

Fig.7

The buy1.ruleml is a counter-example for Hornlog+ because of its ‘disjunctive head’ characteristic (from Dishornlog+).

The validation result for the buy1.ruleml document when we tried to validate it with the Hornlog+ schema is shown in Fig.8.



1. **Error**: Element Atom from namespace http://ruleml.org/spec not allowed as child of element Or from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
   From line 16, column 25; to line 16, column 30
   `<Atom>`

2. **Error**: Element Atom from namespace http://ruleml.org/spec not allowed as child of element Or from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
   From line 22, column 25; to line 22, column 30
   `<Atom>`

Fig.8

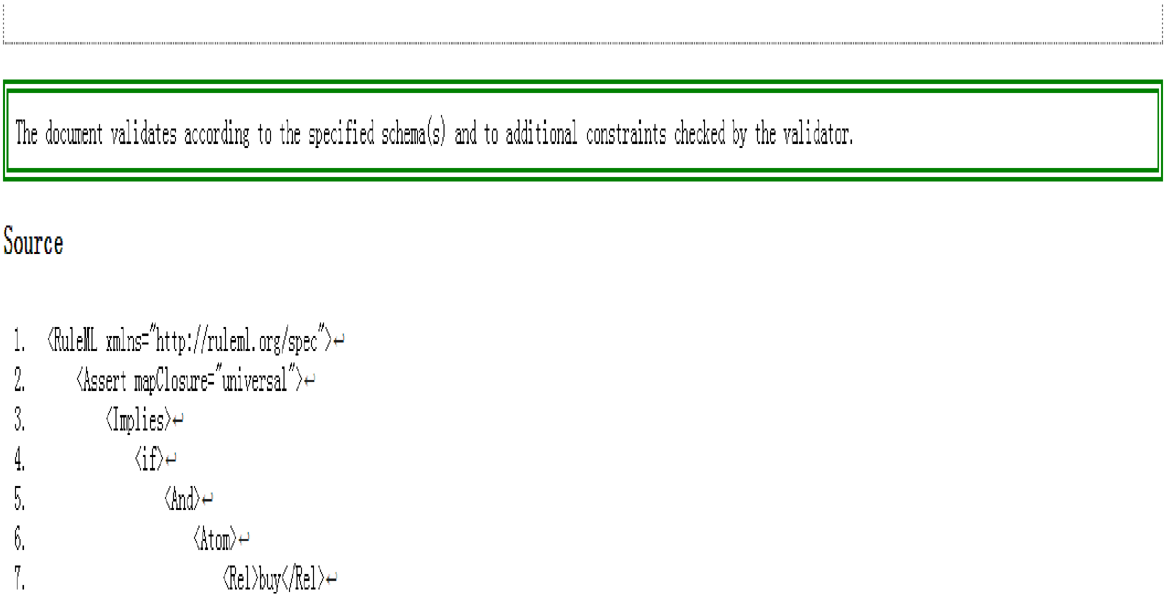
The buy1.ruleml document is an example for Dishornlog+ and it is a counter-example for Hornlog+, because of the non-empty disjunctive head <Or>…..</Or>.

The buy1.ruleml document is not only a counter-example for Hornlog+, but also for all the smaller sublanguages such as Datalog+; Hornlog and Binhornlog+ as well as *their* smaller sublanguages.

Consider the buy2.ruleml document which is a Nafhornlog+ example:

```
<RuleML xmlns="http://ruleml.org/spec">
   <Assert mapClosure="universal">
      <Implies>
         <if>
            <And>
               <Atom>
                  <Rel>buy</Rel>
                  <Var>person</Var>
                  <Var>merchant</Var>
                  <Var>object</Var>
               </Atom>
               <Naf>
                  <Atom>
                     <Rel>produce</Rel>
                     <Var>merchant</Var>
                     <Var>object</Var>
                  </Atom>
               </Naf>
            </And>
         </if>
         <then>
            <Exists>
               <Var>retailer</Var>
               <And>
                  <Atom>
                     <Rel>buy</Rel>
                     <Var>merchant</Var>
                     <Var>retailer</Var>
                     <Var>object</Var>
                  </Atom>
                  <Atom>
                     <Rel>buy</Rel>
                     <Var>retailer</Var>
                     <Expr>
                        <Fun>factoryOf</Fun>
                        <Var>retailer</Var>
                     </Expr>
                     <Var>object</Var>
                  </Atom>
               </And>
            </Exists>
         </then>
      </Implies>
    </Assert>
</RuleML>
```

The validation result for the buy2.ruleml document when we tried to validate it with the Nafhornlog+ schema is shown in Fig.9.



The document validates according to the specified schema(s) and to additional constraints checked by the validator.

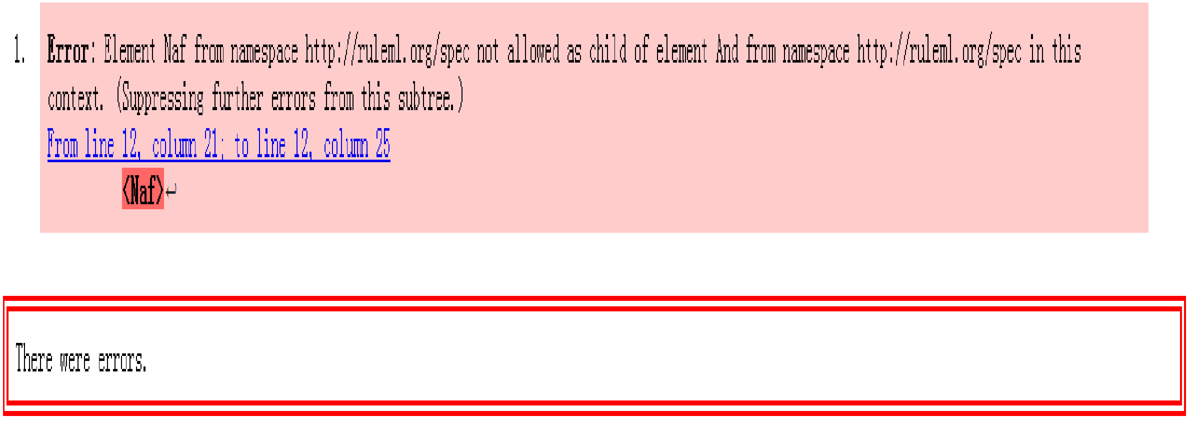
Source

```
<RuleML xmlns="http://ruleml.org/spec">↵
  <Assert mapClosure="universal">↵
    <Implies>↵
      <if>↵
        <And>↵
          <Atom>↵
            <Rel>buy</Rel>↵
```

Fig.9

The buy2.ruleml is a counter-example for Hornlog+.

The validation result for the buy2.ruleml document when we tried to validate it with the Hornlog+ schema is shown in Fig.10.



1. **Error**: Element Naf from namespace http://ruleml.org/spec not allowed as child of element And from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
From line 12, column 21; to line 12, column 25
`<Naf>↵`

There were errors.

Fig.10

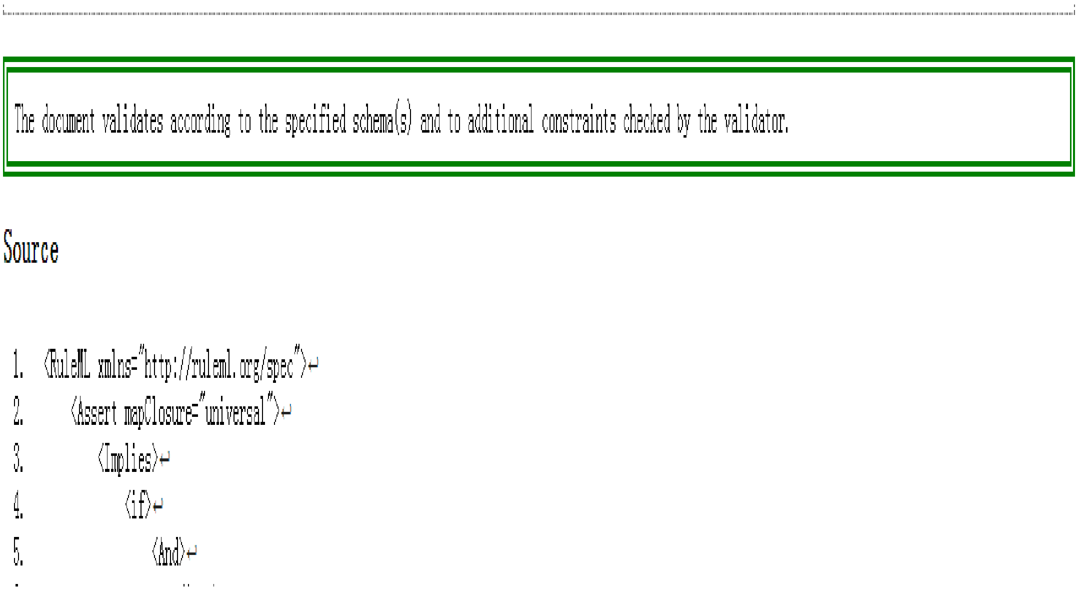
The buy2.ruleml document is an example for Nafhornlog+ and it is a counter-example for Hornlog+, because of the feature <Naf>.

The buy2.ruleml document is not only counter-example for Hornlog+, but also for its smaller sublanguages such as Datalog+ and Hornlog.

Below is the buy3.ruleml document which is a Neghornlog+ example:

```
<RuleML xmlns="http://ruleml.org/spec">
   <Assert mapClosure="universal">
      <Implies>
         <if>
            <And>
               <Atom>
                  <Rel>buy</Rel>
                  <Var>person</Var>
                  <Var>merchant</Var>
                  <Var>object</Var>
               </Atom>
               <Neg>
                  <Atom>
                     <Rel>produce</Rel>
                     <Var>merchant</Var>
                     <Var>object</Var>
                  </Atom>
               </Neg>
            </And>
         </if>
         <then>
            <Exists>
               <Var>retailer</Var>
               <And>
                  <Atom>
                     <Rel>buy</Rel>
                     <Var>merchant</Var>
                     <Var>retailer</Var>
                     <Var>object</Var>
                  </Atom>
                  <Atom>
                     <Rel>buy</Rel>
                     <Var>retailer</Var>
                     <Expr>
                        <Fun>factoryOf</Fun>
                        <Var>retailer</Var>
                     </Expr>
                     <Var>object</Var>
                  </Atom>
               </And>
            </Exists>
         </then>
      </Implies>
   </Assert>
</RuleML>
```

The validation result for the buy3.ruleml document when we tried to validate it with the Neghornlog+ schema is shown in Fig.11.



The document validates according to the specified schema(s) and to additional constraints checked by the validator.

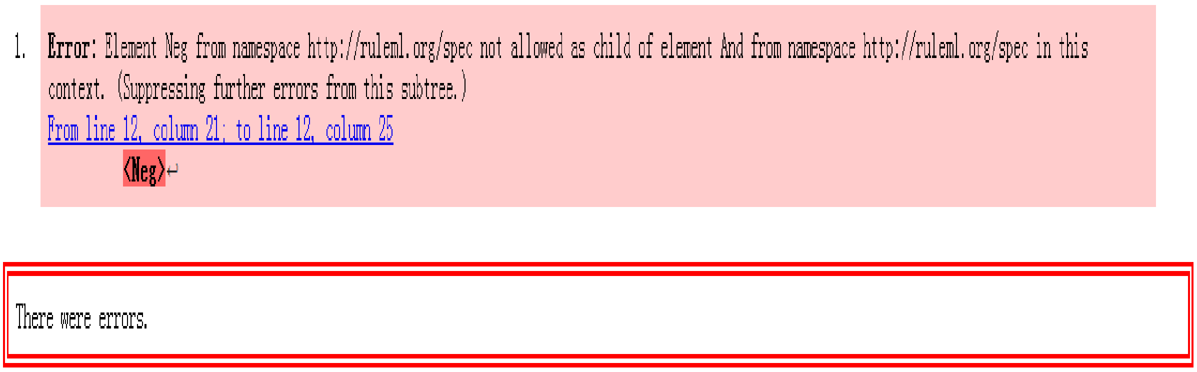
Source

```
<RuleML xmlns="http://ruleml.org/spec">↵
   <Assert mapClosure="universal">↵
      <Implies>↵
         <if>↵
            <And>↵
```

Fig.11

The buy3.ruleml is a counter-example for Hornlog+.

The validation result for the buy3.ruleml document when we tried to validate it with the Hornlog+ schema is shown in Fig.12.



1. **Error**: Element Neg from namespace http://ruleml.org/spec not allowed as child of element And from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
From line 12, column 21; to line 12, column 25
        <Neg>↵

There were errors.

Fig.12

The buy3.ruleml document is an example for Neghornlog+ and it is a counter-example for Hornlog+, because of the feature <Neg>.

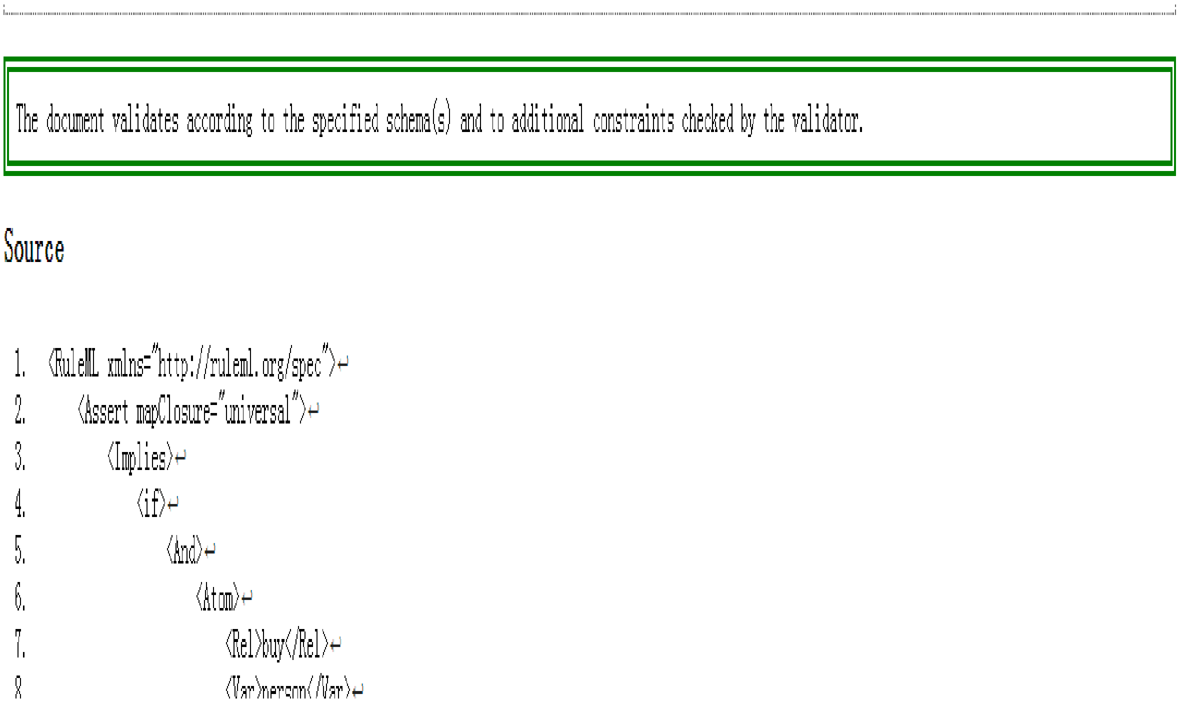
The buy3.ruleml document is not only a counter-example for Hornlog+ but also for its smaller sublanguages such as Datalog+ and Hornlog.

Below is the buy4.ruleml document which is a Nafnegdishornlog+ example:

```
<RuleML xmlns="http://ruleml.org/spec">
   <Assert mapClosure="universal">
      <Implies>
         <if>
```

```
            <And>
               <Atom>
                  <Rel>buy</Rel>
                  <Var>person</Var>
                  <Var>merchant</Var>
                  <Var>object</Var>
               </Atom>
               <Naf>
                  <Neg>
                     <Atom>
                        <Rel>produce</Rel>
                        <Var>merchant</Var>
                        <Var>object</Var>
                     </Atom>
                  </Neg>
               </Naf>
            </And>
         </if>
         <then>
            <Exists>
               <Var>retailer</Var>
               <Or>
                  <Atom>
                     <Rel>buy</Rel>
                     <Var>merchant</Var>
                     <Var>retailer</Var>
                     <Var>object</Var>
                  </Atom>
                  <Atom>
                     <Rel>buy</Rel>
                     <Var>retailer</Var>
                     <Expr>
                        <Fun>factoryOf</Fun>
                        <Var>retailer</Var>
                     </Expr>
                     <Var>object</Var>
                  </Atom>
               </Or>
            </Exists>
         </then>
      </Implies>
   </Assert>
</RuleML>
```

The validation result for the buy4.ruleml document when we tried to validate it with the Nafnegdishornlog+ schema is shown in Fig.13.



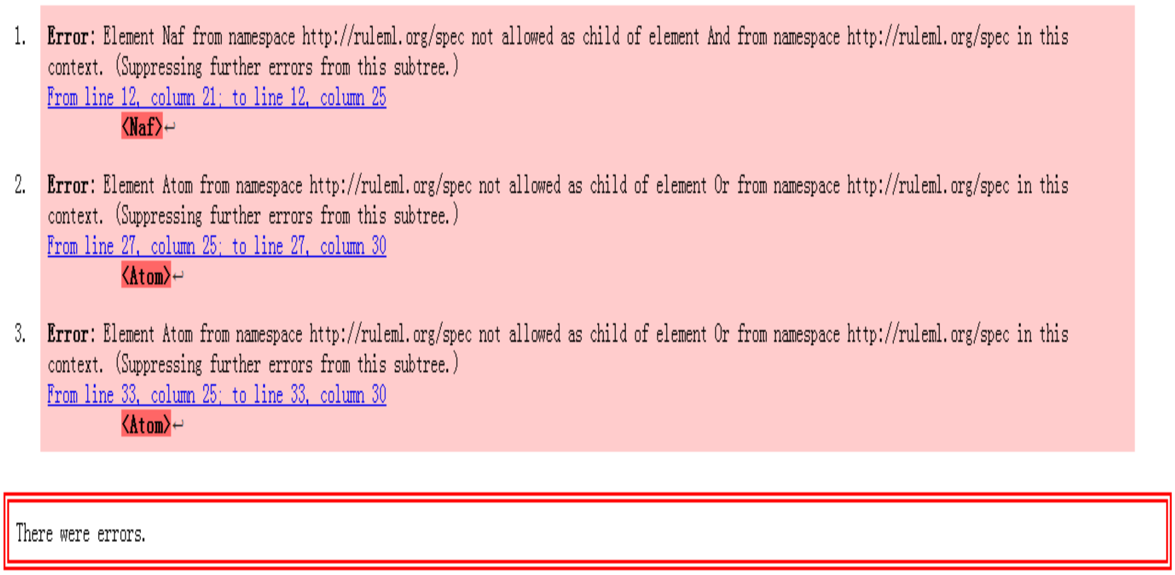
The document validates according to the specified schema(s) and to additional constraints checked by the validator.

Source

```
<RuleML xmlns="http://ruleml.org/spec">↵
  <Assert mapClosure="universal">↵
    <Implies>↵
      <if>↵
        <And>↵
          <Atom>↵
            <Rel>buy</Rel>↵
            <Var>person</Var>↵
```

Fig.13

The buy4.ruleml is a counter-example for Hornlog+.
The validation result for the buy4.ruleml document when we tried to validate it with the Hornlog+ schema is shown in Fig.14.



1. **Error**: Element Naf from namespace http://ruleml.org/spec not allowed as child of element And from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
   From line 12, column 21; to line 12, column 25
   `<Naf>↵`

2. **Error**: Element Atom from namespace http://ruleml.org/spec not allowed as child of element Or from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
   From line 27, column 25; to line 27, column 30
   `<Atom>↵`

3. **Error**: Element Atom from namespace http://ruleml.org/spec not allowed as child of element Or from namespace http://ruleml.org/spec in this context. (Suppressing further errors from this subtree.)
   From line 33, column 25; to line 33, column 30
   `<Atom>↵`

There were errors.

Fig.14

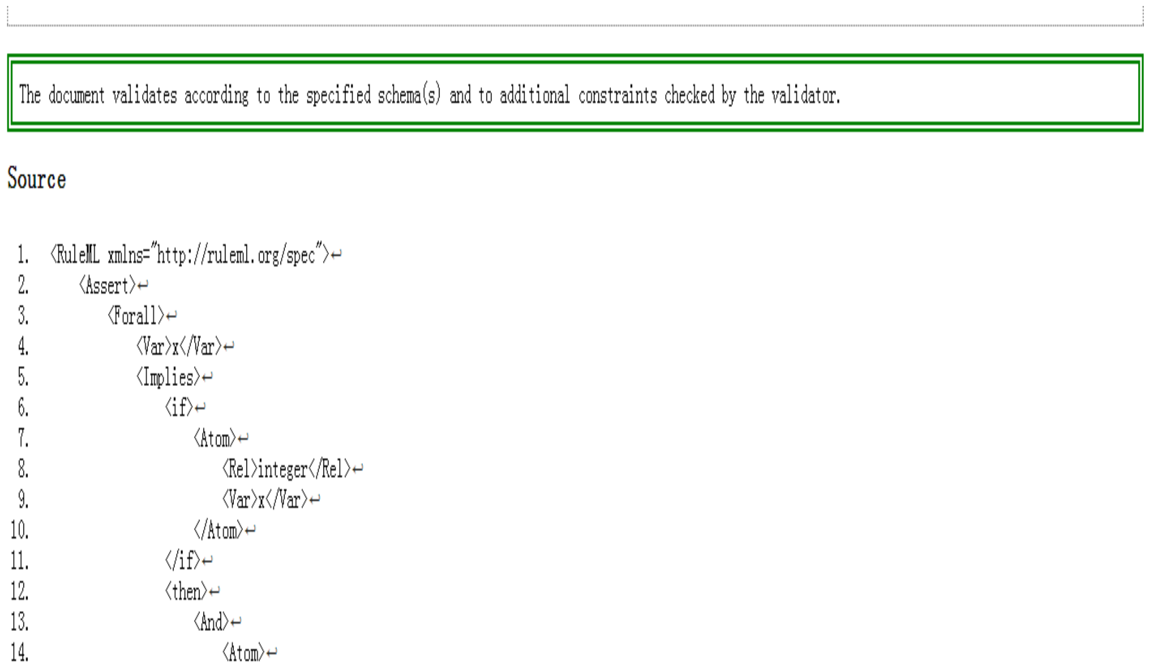
The buy4.ruleml document is an example for Nafnegdishornlog+ and it is a counter-example for Hornlog+, because of the features <Naf>, <Neg> and disjunctive head.

The buy4.ruleml document is not only counter-example for Hornlog+, but also for the larger sublanguages Nafhornlog+, Neghornlog+ and Dishornlog+ as well as the smaller sublanguages such as Datalog+ and Hornlog.

Below is the *co-own.ruleml* document which is is a new Hornlog+ example we created:

```
<RuleML xmlns="http://ruleml.org/spec">
   <Assert mapClosure="universal">
      <Implies>
         <if>
            <And>
               <Atom>
                  <Rel>owns</Rel>
                  <Ind>Criss</Ind>
                  <Ind>house1</Ind>
               </Atom>
               <Atom>
                  <Rel>owns</Rel>
                  <Ind>Kristen</Ind>
                  <Ind>house2</Ind>
                  <Expr>
                     <Fun>wifeOf</Fun>
                     <Ind>Criss</Ind>
                  </Expr>
               </Atom>
            </And>
         </if>
         <then>
            <Equal>
               <Ind>house1</Ind>
               <Ind>house2</Ind>
            </Equal>
         </then>
      </Implies>
   </Assert>
</RuleML>
```

We tried to validate it with the Hornlog+ schema. The Fig.15 shows the validation result.



The document validates according to the specified schema(s) and to additional constraints checked by the validator.

Source

```
<RuleML xmlns="http://ruleml.org/spec">↵
    <Assert>↵
        <Forall>↵
            <Var>x</Var>↵
            <Implies>↵
                <if>↵
                    <Atom>↵
                        <Rel>integer</Rel>↵
                        <Var>x</Var>↵
                    </Atom>↵
                </if>↵
                <then>↵
                    <And>↵
                        <Atom>↵
```

Fig.15

The co-own.ruleml document is not only an example for Hornlog+, but also for the larger sublanguages such as Dishornlog+ and Nafhornlog+.

And it is not only a counter-example for Datalog+, but also for the Hornlog and Binaryhornlog+.

3. LIBRARY STRUCTURE

As we have seen, different documents like buy.ruleml through buy4.ruleml can be example or couter-example of sublanguages like *Binhornlog+*, *Dishornlog+*, *Nafhornlog+*, *Neghornlog+ and Nafnegdishornlog+*. We provide all the examples online that we modified and created using a webpage. For this purpose, we created a new webpage that has all the project related files. The url of our webpage is given below and screenshot is provided in **Appendix B:**
http://testcasesuiteforhornlogruleml101.yolasite.com/

3.1 RelaxNG Compact Schema

We also made another file available on the webpage as RelaxNG compact schema. In this file, we have all the schemas we created using the MYNG 1.01 tool for this project.
Appendix A gives all the schema details.

4. CONCLUSION AND FUTURE WORK

Hornlog+ can be differentiated from other sublanguages on the basis of examples and counter-examples. The set of Deliberation RuleML 1.01 schemas include specific sublanguages which are Hornlog+, Dishornlog+, Nafhornlog+ and Neghornlog+.

Hornlog+ enlarges Datalog+ with functions, which is analogous to the role Hornlog plays for Datalog.

Possible future work on this project is that the library can be augmented by adding further RuleML documents as examples or counter-examples of existing and new (anchor) sublanguages.

5. REFERENCES

[1] Specification of Deliberation RuleML 1.01
http://wiki.ruleml.org/index.php/Specification_of_Deliberation_RuleML_1.01

[2] MYNG 1.01- the Deliberation RuleML Schema Selection Form
http://deliberation.ruleml.org/1.01/myng/

[3] The MYNG 1.01 Suite for Deliberation RuleML 1.01: Taming the Language Lattice
http://ceur-ws.org/Vol-1211/paper/.pdf

[4] RuleML 1.01 Examples (grouped according to sublanguages):
http://deliberation.ruleml.org/1.01/exa

[5] W3C RIF Test Cases:
http://www.w3.org/2005/rules/wiki/Category:Test_Case

[6] Demo of MYNG 1.01:
http://wiki.ruleml.org/index.php/Demo_of_MYNG_1.01

[7] RNC validation tool:
http://validator.nu/

Appendix A: Schemas Generated and Used in Our Project

Datalogplus_min schema:
http://deliberation.ruleml.org/1.01/relaxng/datalogplus_min_relaxed.rnc

hornlog schema:
http://deliberation.ruleml.org/1.01/relaxng/hornlog_relaxed.rnc

binhornlog+ schema:
http://deliberation.ruleml.org/1.01/relaxng/schema_rnc.php?backbone=xf&default=x7&termseq=x2&lng=x1&propo=x3cf&implies=x5f&terms=xf1f&quant=x3&expr=x7&serial=xf

hornlog+ schema:
http://deliberation.ruleml.org/1.01/relaxng/schema_rnc.php?backbone=xf&default=x7&termseq=x7&lng=x1&propo=x3cf&implies=x5f&terms=xf1f&quant=x7&expr=x3&serial=xf

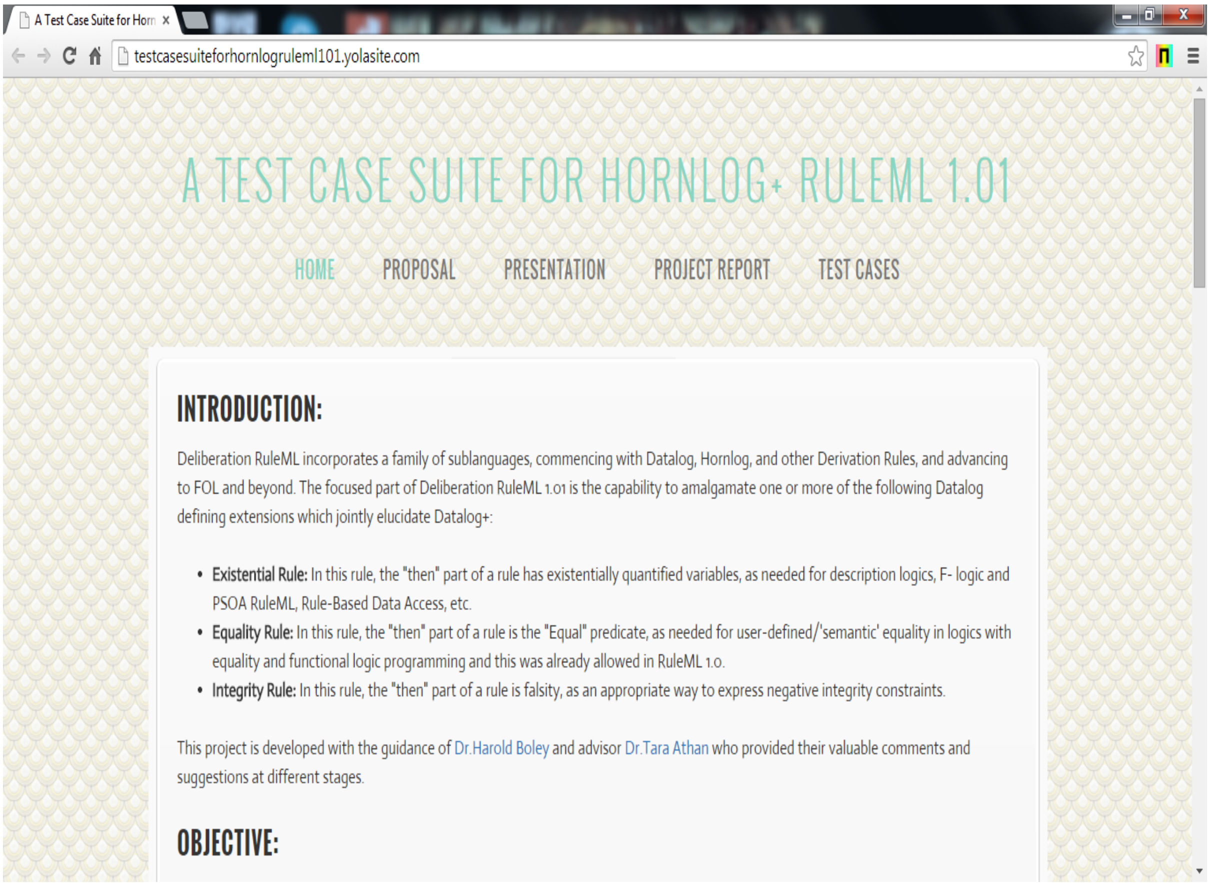
dishornlog+ schema:
http://deliberation.ruleml.org/1.01/relaxng/schema_rnc.php?backbone=x1f&default=x7&termseq=x7&lng=x1&propo=x3cf&implies=x7f&terms=xf1f&quant=x7&expr=x3&serial=xf

nafhornlog+ schema:
http://deliberation.ruleml.org/1.01/relaxng/schema_rnc.php?backbone=xf&default=x7&termseq=x7&lng=x1&propo=x3ef&implies=x5f&terms=xf1f&quant=x7&expr=x3&serial=xf

neghornlog+ schema:
http://deliberation.ruleml.org/1.01/relaxng/schema_rnc.php?backbone=xf&default=x7&termseq=x7&lng=x1&propo=x3df&implies=x5f&terms=xf1f&quant=x7&expr=xf&serial=xf

nafnegdishornlog+ schema:
http://deliberation.ruleml.org/1.01/relaxng/nafnegdishornlogplus_relaxed.rnc

Appendix B: Screenshot of Our Homepage



A Test Case Suite for Horn
testcasesuiteforhornlogruleml101.yolasite.com

A TEST CASE SUITE FOR HORNLOG+ RULEML 1.01

HOME
PROPOSAL
PRESENTATION
PROJECT REPORT
TEST CASES

INTRODUCTION:

Deliberation RuleML incorporates a family of sublanguages, commencing with Datalog, Hornlog, and other Derivation Rules, and advancing to FOL and beyond. The focused part of Deliberation RuleML 1.01 is the capability to amalgamate one or more of the following Datalog defining extensions which jointly elucidate Datalog+:

- **Existential Rule:** In this rule, the "then" part of a rule has existentially quantified variables, as needed for description logics, F- logic and PSOA RuleML, Rule-Based Data Access, etc.
- **Equality Rule:** In this rule, the "then" part of a rule is the "Equal" predicate, as needed for user-defined/'semantic' equality in logics with equality and functional logic programming and this was already allowed in RuleML 1.0.
- **Integrity Rule:** In this rule, the "then" part of a rule is falsity, as an appropriate way to express negative integrity constraints.

This project is developed with the guidance of Dr.Harold Boley and advisor Dr.Tara Athan who provided their valuable comments and suggestions at different stages.

OBJECTIVE: