

5.3 RDF/XML Test cases for RDF Logic, Web Ontology and Maths content

Project name:

Semantic Web Advanced Development for Europe (SWAD-Europe)

Project Number:

IST-2001-34732

Workpackage name:

SWAD-Europe: Integration with XML Technology

Workpackage description:

└

<http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-5.html>

Deliverable title:

5.3 RDF/XML Test cases for RDF Logic, Web Ontology and Maths content

URI:

Authors:

└ [Dan Brickley](#), Max Froumentin, Libby Miller, Damian Steer

Abstract:

Demonstrator RDF Extractor System

Status:

Completed report, last updated 2004-07-15

Comments on this document should be sent to the public SWAD-Europe mailing list, [└ public-esw@w3.org](mailto:public-esw@w3.org),

Contents

Semantic Web application scenarios: Company Accounts

An experiment in W3C spec overlap: *using MathML, XML, RDF to represent company accounts in the Web.*

by [└ Dan Brickley](#) (W3C/ILRT), Max Froumentin, Libby Miller, Damian Steer

-
- [└ Introduction](#)
 - [└ Sample Rules](#)
 - [└ Applying the tests](#)
 - [└ Conclusions](#)
 - [└ References](#)
-

Introduction

Company Accounts provide an interesting Semantic Web application scenario, since there are a number of issues associated with representing accounting information in the Web that require the use of multiple W3C specifications.

We can use MathML to represent numeric information (for presentation and for content interchange), we can use XML Schemas and DTDs to represent document formats for interchange; or RDF Schemas and Web Ontologies to represent the structure and constraints associated with the information modeling that underpins such documents. SVG provides an excellent format, again written in XML, for the exchange of diagrammatic representation (eg. charts, figures) derived from company account information. RDF can be used to merge information about companies from multiple sources, and RDF rule and inference systems (perhaps combined with

MathML) may be useful for capturing the basic formulae that underpin company accounts. The challenge is to combine all these relevant components into a useful, understandable and maintainable whole.

This experiment begins with some sample data from the Biz/ed [\[BIZED\]](#) Web site.

The idea is to make some data available (see [\[reports.xml\]](#) and original [\[text format\]](#)), and then show how MathML, RDF and XSLT representations overlap.

This report illustrates three ways of illustrating the relationships between parts of a company report in a rule-like fashion: in text, as a perl function, as XSLT [\[XSLT\]](#) over MathML [\[MathML\]](#) and as N3 rules [\[N3\]](#) over an RDF data file [\[Reports\]](#).

Many thanks to Sean Palmer and Aaron Swartz from [#rdfig](#) for help with the N3 rules.

Sample Rules

The following definitions are used by the Biz/ed online company report profiler. They express relationships amongst the various measures used in company accounts.

The eleven simple equations can be expressed in many different ways with current XML and RDF tools. We have chosen two: N3 RDF rules and MathML. For reference we also include Perl5 functions. Below are the eleven rules expressed in prose and in each of the three languages.

Definitions -

SOURCE: Company Name

YEAR: Year of report

1. Return on Net Assets = Operating Profit / Net Assets * 100

Perl5:

```
sub RNA
{
  return("RNA") if($inHeader);
  return("") if ($elements[Index{'NA'}]==0); # NULL trap
  return(sprintf("%.2f", $elements[Index{'OPOA'}]/$elements[Index{'NA'}]*100));
}
```

N3:

```
@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :rna , :op , :na , :name , :company , :year .
{
  :company <http://xmlns.com/foaf/corp#op> :op;
  <http://xmlns.com/foaf/corp#source> :name;
  <http://xmlns.com/foaf/corp#na> :na ;
  <http://xmlns.com/foaf/corp#year> :year .
  :op log:notEqualTo "" .
  :na log:notEqualTo "" .
  ([is math:quotient of (:op :na)] "100") math:product :rna
}
log:implies
{
  :company <http://xmlns.com/foaf/corp#rna> :rna;
  <http://xmlns.com/foaf/corp#source> :name ;
  <http://xmlns.com/foaf/corp#year> :year .
} .
```

MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>rna</ci>
    <apply>
      <times/>
      <apply><divide/><ci>op</ci><ci>na</ci></apply>
      <cn>100</cn>
    </apply>
  </apply>
</math>
```

2. Profit Margin = Operating Profit / Sales Revenue

Perl5:

```
sub PM
{
  return("PM") if($inHeader);
  return("") if ($elements[$Index{'SR'}]==0); # NULL trap

  return(sprintf("%.2f", $elements[$Index{'OPOA'}]/$elements[$Index{'SR'}]*100));
}
```

N3:

```
@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :pm , :op , :sr , :name , :company , :year .

{
  :company <http://xmlns.com/foaf/corp#op> :op;
  <http://xmlns.com/foaf/corp#source> :name;
  <http://xmlns.com/foaf/corp#year> :year;
  <http://xmlns.com/foaf/corp#sr> :sr .
  :op log:notEqualTo "" .
  :sr log:notEqualTo "" .
  ([is math:quotient of (:op :sr)] "100") math:product :pm
}

log:implies

{
  :company <http://xmlns.com/foaf/corp#pm> :pm;
  <http://xmlns.com/foaf/corp#source> :name ;
  <http://xmlns.com/foaf/corp#year> :year .
} .
```

MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>pm</ci>
    <apply>
      <times/>
      <apply><divide/><ci>op</ci><ci>sr</ci></apply>
      <cn>100</cn>
    </apply>
  </apply>
</math>
```

3. Asset Turnover = Sales Revenue / Net Assets

Perl5:

```
sub AT
{
  return("AT") if($inHeader);
  return("") if ($elements[$Index{'NA'}]==0); # NULL trap
  return(sprintf("%.2f", $elements[$Index{'SR'}]/$elements[$Index{'NA'}]));
}
```

N3:

```

@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forAll :at , :na , :sr , :name , :company , :year .

{
    :company <http://xmlns.com/foaf/corp#na> :na;
    <http://xmlns.com/foaf/corp#source> :name;
    <http://xmlns.com/foaf/corp#year> :year;
    <http://xmlns.com/foaf/corp#sr> :sr .
    :na log:notEqualTo "" .
    :sr log:notEqualTo "" .
    (:sr :na) math:quotient :at
}

log:implies

{

    :company <http://xmlns.com/foaf/corp#at> :at;
    <http://xmlns.com/foaf/corp#source> :name ;
    <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>at</ci>
    <apply><divide/><ci>sr</ci><ci>na</ci></apply>
  </apply>
</math>

```

4. Number Of Employees = Sales Revenue / Sales Per Employee * 1000000

Perl5:

```

sub SPE
{
    return("SPE") if($inHeader);
    return("") if ($elements[$Index{'NOE'}]==0); # NULL trap

    return(sprintf("%.2f", $elements[$Index{'SR'}]/$elements[$Index{'NOE'}]*1000000));
}

```

N3:

```

prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forAll :spe , :sr , :noe , :company , :name , :year .

{
    :company <http://xmlns.com/foaf/corp#sr> :sr;
    <http://xmlns.com/foaf/corp#source> :name;
    <http://xmlns.com/foaf/corp#year> :year;
    <http://xmlns.com/foaf/corp#spe> :spe .
    :sr log:notEqualTo "" .
    :spe log:notEqualTo "" .
    ([is math:quotient of (:sr :spe)] "1000000") math:product :noe
}

log:implies

{

    :company <http://xmlns.com/foaf/corp#noe> :noe;
    <http://xmlns.com/foaf/corp#source> :name ;
    <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>noe</ci>
    <apply>
      <times/>
      <apply><divide/><ci>sr</ci><ci>spe</ci></apply>
    <cn>1000000</cn>
  </apply>
</math>

```

```
</apply>
</math>
```

5. Operating Profit per Employee = Operating Profit/Number of Employees * 1000000

Perl5:

```
sub OPPE
{
    return("OPPE") if($inHeader);
    return("") if ($elements[$Index{'NOE'}]==0); # NULL trap

    return(sprintf("%.2f", $elements[$Index{'OPOA'}]/$elements[$Index{'NOE'}]*1000000));
}
```

N3:

```
@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :oppe , :op , :noe , :name , :company , :year .

{
    :company <http://xmlns.com/foaf/corp#op> :op;
    <http://xmlns.com/foaf/corp#source> :name;
    <http://xmlns.com/foaf/corp#year> :year;
    <http://xmlns.com/foaf/corp#oppe> :oppe .
    :op log:notEqualTo "" .
    :oppe log:notEqualTo "" .
    ([is math:quotient of (:op :oppe)] "1000000") math:product :noe
}

log:implies

{
    :company <http://xmlns.com/foaf/corp#noe> :noe;
    <http://xmlns.com/foaf/corp#source> :name ;
    <http://xmlns.com/foaf/corp#year> :year .
} .
```

MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>noe</ci>
    <apply>
      <times/>
      <apply><divide/><ci>op</ci><ci>oppe</ci></apply>
      <cn>1000000</cn>
    </apply>
  </apply>
</math>
```

6. Non UK Sales Ratio = Non UK Sales Value /Sales Revenue

Perl5:

```
sub NUKSR
{
    return("NUKSR") if($inHeader);
    return("") if ($elements[$Index{'SR'}]==0); # NULL trap

    return(sprintf("%.2f", $elements[$Index{'NUKSV'}]/$elements[$Index{'SR'}]*100));
}
```

N3:

```
@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :nuksr , :nuks , :sr , :name , :company , :year .

{
    :company <http://xmlns.com/foaf/corp#nuks> :nuks;
```

```

    <http://xmlns.com/foaf/corp#source> :name;
    <http://xmlns.com/foaf/corp#year> :year;
    <http://xmlns.com/foaf/corp#sr> :sr .
    :nuks log:notEqualTo "" .
    :nuks log:notEqualTo "0" .
    :sr log:notEqualTo "" .
    :sr log:notEqualTo "0" .
    ([is math:quotient of (:nuks :sr)] "100") math:product :nuksr
  }
  log:implies
{
  :company <http://xmlns.com/foaf/corp#nuksr> :nuksr;
  <http://xmlns.com/foaf/corp#source> :name ;
  <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>nuksr</ci>
    <apply>
      <times/>
      <apply><divide/><ci>nuks</ci><ci>sr</ci></apply>
      <cn>100</cn>
    </apply>
  </apply>
</math>

```

7. Current Ratio = Current Assets/Current Liabilites

Perl5:

```

sub CR
{
  return("CR") if($inHeader);
  return("") if ($elements[$Index{'CL'}]==0); # NULL trap
  return(sprintf("%.2f",&CA()/&elements[$Index{'CL'}]));
}

```

N3:

```

@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :ca , :cr , :cl , :name , :company , :year .

{
  :company <http://xmlns.com/foaf/corp#cr> :cr;
  <http://xmlns.com/foaf/corp#source> :name;
  <http://xmlns.com/foaf/corp#year> :year;
  <http://xmlns.com/foaf/corp#cl> :cl .
  :cr log:notEqualTo "" .
  :cl log:notEqualTo "" .
  (:cr :cl) math:product :ca
}
log:implies
{
  :company <http://xmlns.com/foaf/corp#ca> :ca;
  <http://xmlns.com/foaf/corp#source> :name ;
  <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>ca</ci>
    <apply>
      <times/><ci>cr</ci><ci>cl</ci>
    </apply>
  </apply>
</math>

```

8. Acid Test Ratio = Current Assets Debtors + Current Assets Cash / Current

Liabilities

Perl5:

```

sub ATR
{
    return("ATR") if($inHeader);
    return("") if ($elements[$Index{'SR'}]==0); # NULL trap

return(sprintf("%.2f",($elements[$Index{'CAD'}]+$elements[$Index{'CAC'}])/elements[$Index{'CL'}
}

```

N3:

```

@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :cad , :cac , :cas , :cl , :at , :name , :company , :year .

{
    :company <http://xmlns.com/foaf/corp#cad> :cad;
    <http://xmlns.com/foaf/corp#source> :name;
    <http://xmlns.com/foaf/corp#year> :year;
    <http://xmlns.com/foaf/corp#cac> :cac;
    <http://xmlns.com/foaf/corp#cas> :cas;
    <http://xmlns.com/foaf/corp#cl> :cl .
    :cad log:notEqualTo "" .
    :cac log:notEqualTo "" .
    :cl log:notEqualTo "" .
    ([ is math:sum of (:cad :cac :cas)] :cl) math:quotient :at .
}

log:implies

{

    :company <http://xmlns.com/foaf/corp#at> :at;
    <http://xmlns.com/foaf/corp#source> :name ;
    <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>at</ci>
    <apply>
      <divide/>
      <apply>
        <plus/>
        <ci>cad</ci>
        <ci>cac</ci>
      </apply>
      <ci>cl</ci>
    </apply>
  </apply>
</math>

```

9. Current Assets == Current Assets Stocks + Current Assets Debtors + Current Assets Cash

Perl5:

```

sub CA
{
    return("CA") if($inHeader);

return(sprintf("%.2f", $elements[$Index{'CAS'}]+$elements[$Index{'CAD'}]+$elements[$Index{'CAC'}
}

```

N3:

```

@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :cad , :cac , :cas , :ca , :name , :company , :year .

```

```

{
  :company <http://xmlns.com/foaf/corp#cad> :cad;
  <http://xmlns.com/foaf/corp#source> :name;
  <http://xmlns.com/foaf/corp#year> :year;
  <http://xmlns.com/foaf/corp#cac> :cac;
  <http://xmlns.com/foaf/corp#cas> :cas .
  :cad log:notEqualTo "" .
  :cac log:notEqualTo "" .
  :cas log:notEqualTo "" .
  (:cad :cac :cas) math:sum :ca
}
log:implies
{
  :company <http://xmlns.com/foaf/corp#ca> :ca;
  <http://xmlns.com/foaf/corp#source> :name ;
  <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>ca</ci>
    <apply>
      <plus/>
      <ci>cas</ci>
      <ci>cad</ci>
      <ci>cac</ci>
    </apply>
  </apply>
</math>

```

10. Debtor Days =(Current Assets Debtors*100000/sales revenue) / 365

Perl5:

```

sub DD
{
  return("DD") if($inHeader);
  return("") if ($elements[$Index{'SR'}]==0); # NULL trap

  return(sprintf("%.2f", ($elements[$Index{'CAD'}]*100000/$elements[$Index{'SR'}])/365));
}

```

N3:

```

@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :cad , :sr , :dd , :name , :company , :year .

{
  :company <http://xmlns.com/foaf/corp#cad> :cad;
  <http://xmlns.com/foaf/corp#source> :name;
  <http://xmlns.com/foaf/corp#year> :year;
  <http://xmlns.com/foaf/corp#sr> :sr .
  :cad log:notEqualTo "" .
  :sr log:notEqualTo "" .
  ([ is math:quotient of (:cad :sr)] "365") math:quotient :dd .
}
log:implies
{
  :company <http://xmlns.com/foaf/corp#dd> :dd;
  <http://xmlns.com/foaf/corp#source> :name ;
  <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>dd</ci>
    <apply>
      <divide/>
      <apply>
        <divide/>
        <apply>
          <times/>

```

```

      <ci>cad</ci>
      <cn>100000</cn>
    </apply>
    <ci>sr</ci>
  </apply>
  <cn>365</cn>
</apply>
</math>

```

11. Stock Days = Current Assets Stocks * 10000 / cost of sales) /365

Perl5:

```

sub SD
{
  return("SD") if($inHeader);
  return("**") if ($elements[$Index{'SR'}]==0); # NULL trap

  return(sprintf("%.2f", ($elements[$Index{'CAS'}]*100000/$elements[$Index{'COS'}])/365));
}

```

N3:

```

@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .

this log:forall :cas , :cos , :sd , :name , :company , :year .

{
  :company <http://xmlns.com/foaf/corp#cas> :cas;
  <http://xmlns.com/foaf/corp#source> :name;
  <http://xmlns.com/foaf/corp#year> :year;
  <http://xmlns.com/foaf/corp#cos> :cos .
  :cas log:notEqualTo "" .
  :cos log:notEqualTo "" .
  ([is math:quotient of (:cas :cos)] "27.29726") math:product :sd
}

log:implies

{
  :company <http://xmlns.com/foaf/corp#dd> :sd;
  <http://xmlns.com/foaf/corp#source> :name ;
  <http://xmlns.com/foaf/corp#year> :year .
} .

```

MathML:

```

<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <ci>sd</ci>
    <apply>
      <divide/>
      <apply>
        <divide/>
        <apply>
          <times/>
          <ci>cas</ci>
          <cn>100000</cn>
        </apply>
        <ci>cos</ci>
      </apply>
    </apply>
    <cn>365</cn>
  </apply>
</math>

```

Applying the tests

These equations can be processed by XML and RDF tools. Here we use an example file [reports.xml](#), a simple RDF/XML file to describe the company reports for sixteen companies, for two years each. CWM [\[CWM\]](#), a forward-chaining rules engine written in Python, can be used to process the N3 RDF rules. XSLT can be used to process the MathML rules. In each case we get a set of results from the mathematical calculations.

The N3 and MathML testcases are available as separate [files](#) [\[Tests\]](#) and as a [zipped archive](#). Instructions are provided to run the tests [\[readme-xslt.txt\]](#), [\[readme-cwm.txt\]](#). The [XSLT source](#) for use with the MathML tests is available. The outputs (marked '_out') are also available in the directory.

Conclusions

This study has explored two standards-based approaches to the representation of simple mathematical functions. The N3-based approach builds on W3C's RDF data model, using the experimental N3 rule language to encode mathematical functions. The MathML-based approach uses W3C's MathML format, and is implemented using a prototype MathML interpreter written in W3C's XSLT language. A similar approach could also be implemented using XQuery [\[XQuery\]](#). The primary motivation for this piece of work within SWAD-Europe was to demonstrate to members of the Web standards community that the products (RDF, MathML, XSLT, N3) of different work areas could be applied to the same problem space; in this case, the problem of representing company account data and rules in a standards-based and machine-processable format. There are many possible elaborations on this work. In addition to the possibility of an XQuery implementation, future work (beyond the scope of SWAD-Europe) could explore different approaches to mechanical comparison of the output from interpreters (MathML, N3). In particular the use of RDF graph-diff tools could make it easier to automatically confirm that different interpreters for these rules are behaving identically when given the same inputs.

The primary conclusion of this work is that any future standardisation effort focussed on Semantic Web Rule languages should take care to consider the existing potential of languages already standardised within W3C, and to explore the possibility for mappings between these different approaches to representing data and rules on the Web. The test rules and data explored here serve to illustrate the possibility of using either MathML or N3 to represent company accounting ratios. This example scenario was chosen as illustrative of a wide variety of use cases for Web standards technology, ie. one in which there are multiple relevant technologies, each with different strengths, weaknesses and emphasis. By showing two concrete implementations, this report may help developers understand some of the practical implementation options available.

References

[Schema]

RDF schema for the properties in the company reports.

[\[http://www.w3.org/2001/sw/Europe/reports/xml_test_cases/schema-content.rdf\]](http://www.w3.org/2001/sw/Europe/reports/xml_test_cases/schema-content.rdf)

[IRC]

IRC discussion with Aaron and Sean

[\[http://ilrt.org/discovery/chatlogs/rdf/g/2002-04-16#T20-53-23\]](http://ilrt.org/discovery/chatlogs/rdf/g/2002-04-16#T20-53-23)

[BIZED]

Biz/Ed Website

[\[http://www.bized.ac.uk\]](http://www.bized.ac.uk)

[XSLT]

XSLT

[\[http://www.w3.org/TR/xslt\]](http://www.w3.org/TR/xslt)

[MathML]

MathML

[\[http://www.w3.org/Math/\]](http://www.w3.org/Math/)

[N3]

Notation 3: An RDF language for the Semantic Web

[\[http://www.w3.org/DesignIssues/Notation3.html\]](http://www.w3.org/DesignIssues/Notation3.html)

[Reports]

Company reports expressed as RDF

http://www.w3.org/2001/sw/Europe/reports/xml_test_cases/reports.xml

[Tests]

Company reports MathML and N3 tests directory

http://www.w3.org/2001/sw/Europe/reports/xml_test_cases/tests/

[CWM]

Cwm - a general purpose data processor for the Semantic Web

<http://www.w3.org/2000/10/swap/doc/cwm.html>

[XQuery]

XMLQuery (XQuery)

<http://www.w3.org/XML/Query>