XBRL FORMULA LANGUAGE, PROGRESS, COMPARISONS WITH FORMULA LINKBASE

Herm Fischer

Frankfurt,

FORMULA PURPOSE AND DEPLOYMENT

Validation of business reports

- Submissions to regulatory agencies

Introduced when Xlink Linkbases were vogue

- Intended to exploit the XML syntax

Difficult and low level nature

- perceived as a deterrent to wider use of XBRL Formula.

Benefits of first-order predicate language ideally suited to report validation

- Similar origins as SQL
  - . SQL has survived for decades, but with a programming language syntax.
- Ideas originally in the Prolog language
- Maps well into validation of large sets of data with complex accounting and submission rules

MANIFEST LIMITATIONS

Standard had been based on a linkbase syntax

- Formula files almost unreadable
- Modularity hidden by Xlink syntax
- Requires specialized tools (which are few)

Assertion processing is unmanaged

- Control of which ones run based on report and outcome of other formulas
- Assertions run even when not relevant (to data, to control flow)
- Desire workflow of collections of assertion

WHY XLINK AND XML

2002-2007 was “heyday” for linkbases

Formula intended to use linkbases for extensibility of

- Formula labeling and messaging
- Override and customization of filters
- Partitioning modularity of files for development

Retrospective

- No known use of override or customization by formula linkbases

LINKBASE CREATION TOOLS

Authoring

- A few tools create formula linkbases at syntax element level
- DPM Architect creates formula linkbases for DPM-structured projects

LANGUAGE BASED SYNTAX

Style of language, tradeoffs

- SQL is a predicate-based language (but specific to databases)
- Java and C-based languages incompatible with XPath expression styles

Goal is to compile syntax language into formula linkbase

- Preserve investment in formula processors
- Preserve investment of existing formula linkbases of projects

XPATH EXPRESSION SYNTAX SOLUTION

The XQuery-style syntax wraps XPath expressions

- Grammar for formula syntax is based on XQuery style
- Other styles were not compatible with a simple-to-parse grammar

Prototype grammar construction guided completion of the syntax solution

- It was necessary to build working translators to/from syntax, to complete the grammar

A CDP FORMULA LINKBASE 1/2

```
<gen:link xlink:type="extended" xlink:role="http://www.xbrl.org/2008/role/link">
  <validation:assertionSet xlink:type="resource" xlink:label="assertionSet" xlink:title="assertionSet" id="assertionSet"/>
  <va:valueAssertion xlink:type="resource" xlink:label="valueAssertion" xlink:title="valueAssertion" id="valueAssertion"
    aspectModel="dimensional" implicitFiltering="true"
   ,,test="fn:number(fn:string($OperationsYearEnding))  &gt;= 2010 and fn:number(fn:string($OperationsYearEnding))  &lt;= 2016"/>
  <gen:arc xlink:type="arc" xlink:arcrole="http://xbrl.org/arcrole/2008/assertion-set" xlink:from="assertionSet" xlink:to="valueAssertion”
   ,,xlink:title="user-defined: assertionSet to valueAssertion" priority="0" order="1.0"/>
  <variable:factVariable xlink:type="resource" xlink:label="factVariable" xlink:title="factVariable" id="factVariable" bindAsSequence="false”
    fallbackValue="0" nils="true"/>
  <variable:variableArc xlink:type="arc" xlink:arcrole="http://xbrl.org/arcrole/2008/variable-set" xlink:from="valueAssertion"
    xlink:to="factVariable" xlink:title="user-defined: valueAssertion to factVariable" priority="0" order="1.0"
    name="OperationsYearEnding"/>
  <cf:conceptName xlink:type="resource" xlink:label="conceptName" xlink:title="conceptName" id="conceptName">
   <cf:concept>
    <cf:qname>cdp-og:EmissionsIntensitiesScope1Scope2ProductionOperationsYearEnding</cf:qname>
   </cf:concept>
  </cf:conceptName>
<variable:variableFilterArc xlink:type="arc" xlink:arcrole="http://xbrl.org/arcrole/2008/variable-filter" xlink:from="factVariable”
   xlink:to="conceptName" xlink:title="user-defined: factVariable to conceptName" priority="0" order="1.0" complement="false"
   cover="true"/>
```

A CDP FORMULA LINKBASE 2/2

```
<df:typedDimension xlink:type="resource" xlink:label="typedDimension" xlink:title="typedDimension"
    id="typedDimension">
  <df:dimension>
   <df:qname>cdp-og:EmissionsIntensitiesScope1Scope2ProductionOperationsAxis</df:qname>
  </df:dimension>
</df:typedDimension>
<variable:variableFilterArc xlink:type="arc" xlink:arcrole="http://xbrl.org/arcrole/2008/variable-filter"
  xlink:from="factVariable" xlink:to="typedDimension" xlink:title="user-defined: factVariable to typedDimension”
   priority="0" order="2.0" complement="false" cover="true"/>
<msg:message xlink:type="resource" xlink:label="message" xlink:role="http://www.xbrl.org/2010/role/message"
    xlink:title="message" xml:lang="en" id="label">Emissions intensities (Scope1 + Scope 2) associated with current
    production and operations, Year ending must be between 2010 and 2016.
</msg:message>
<gen:arc xlink:type="arc" xlink:arcrole="http://xbrl.org/arcrole/2010/assertion-unsatisfied-message"
     xlink:from="valueAssertion" xlink:to="message" xlink:title="user-defined: valueAssertion to message”
     priority="0" order="1.0"/>
 </gen:link>
</link:linkbase>
```

SAME CDP EXAMPLE IN XF (LANGUAGE)

```
namespace cdp-og = "http://www.cdp.net/xbrl/cdp/og/2016-08-30/";
assertion-set assertionSet {
  assertion valueAssertion {
    unsatisfied-message (en) "Emissions intensities (Scope1 + Scope 2) associated with current production
                              and operations, Year ending must be between 2010 and 2016. ";
    variable $OperationsYearEnding {
      nils
        fallback {0}
        concept-name cdp-og:EmissionsIntensitiesScope1Scope2ProductionOperationsYearEnding;
      typed-dimension cdp-og:EmissionsIntensitiesScope1Scope2ProductionOperationsAxis;
    };
    test {fn:number(fn:string($OperationsYearEnding))  >= 2010 and
          fn:number(fn:string($OperationsYearEnding))  <= 2016};
  };
};
```

XF GRAMMAR, MODULE LEVEL

```
module ::=
  (namespace-declaration)*
  defaults*
  parameter*
  (filter-declararion | fact-variable | general-variable | function-declaration)*
  (assertion-set | assertion)*

comment ::=
  "(:" comment-contents ":)"

comment-contents ::=
  (Char+ - (Char* ('(:' | ':)') Char*))
```

comments are removed from XPath contents when generating formula linkbase

separator = ";"

XF SYNTAX, ASSERTION

```
enclosed-expression ::=
  "{" XPath-expression "}"
XPath-expression ::=
  regex "[\s]*[\S]+[\s\S]*"
assertion ::=
  "assertion" name "{"
  (label ("(" lang ")")? quoted-string separator) |
   ("satisfied-message" | "unsatisfied-message") ("(" lang ")")? quoted-string separator) |
   ("unsatisfied-severity" message-severity separator) |
   ("aspect-model-non-dimensional" separator) |
   ("no-implicit-filtering" separator) )*
  filter-declararion*
  group-filter*
  (variable | referenced-parameter)*
  precondition*
  (value-expression | existence-expression)
  "}" separator
value-expression ::=
  "test" enclosed-expression separator
```

XF SYNTAX, ASPECT RULES & VARIABLE

```
aspect-rules ::=
  "aspect-rules" ("source" qname)? "{"
    (
     "concept" (qname | enclosed-expression)?
          separator |
     "entity-identifier"
        ("scheme" enclosed-expression)?
        ("identifier" enclosed-expression)?
         separator |
     "period"
        ("forever" |
         "instant" enclosed-expression? |
         "duration"
            ("start" enclosed-expression)?
            ("end" enclosed-expression)? )
          separator |
     "unit" "augment"? "{"
           (("multiply-by"|"divide-by")
              ("source" qname)?
              ("measure" enclosed-expression)?
             separator
            )*
       "}" separator |
     "explicit-dimension"  qname "{"
           (("member" (qname | enclosed-expression) |
            "omit") separator)*
       "}" separator |
       "typed-dimension" qname "{"
           (("xpath" enclosed-expression |
             "value" quoted-xml-string |
            "omit") separator)*
       "}" separator
        )*
     "}" separator
```

```
fact-variable ::=
  "variable" "$" name "{"
  "bind-as-sequence"?
  "nils"?
  "matches"?
  ("fallback" enclosed-expression)?
  filter*
  "}" separator
```

HMRC CONSISTENCY CHECK (SUM)

```
undefined-severity ERROR

assertion-set intangiblesConsistencyChecks{

 assertion IntangibleAssets {
    variable $theSum {
       concept-name IntangibleAssets;
    }
    variable $theAddends {
       bind-as-sequence
       concept-name IntangibleAssetsGrossCost
                AccumulatedAmortisationImpairmentIntangibleAssets;
    }
    test {
      $theSum eq sum($theAddends)
    };
 };
```

HMRC … (SUM MULTIPLE ADDENDS)

```
assertion IncreaseDecreaseInIntangibleAssets {
    variable $theSum {
       concept-name IncreaseDecreaseInIntangibleAssets;
    }
    variable $theAddends {
       bind-as-sequence
       concept-name
         TotalAdditionsIncludingFromBusinessCombinationsIntangibleAssets
         DisposalsIntangibleAssets
         DecreaseThroughDiscontinuedOperationsIntangibleAssets
         TotalIncreaseDecreaseFromRevaluationsIntangibleAssets
         IncreaseDecreaseFromForeignExchangeDifferencesIntangibleAsset
         IncreaseDecreaseDueToTransfersIntoOrOutIntangibleAssets
         IncreaseDecreaseDueToTransfersBetweenClassesIntangibleAssets
         FurtherItemIncreaseDecreaseInIntangibleAssetsComponentTotalChangeInIntangibleAssets
         IncreaseDecreaseThroughOtherChangesIntangibleAssets
         ;
    };
    test {
      $theSum eq sum($theAddends)
    };
};
```

HMRC … (MOVEMENT PATTERN)

```
assertion IntangibleAssetsGrossCost {
    variable $theEndingBalance {
        concept-name IntangibleAssetsGrossCost;
        instant-duration end $theFlowAddends;
    }
    variable $theChanges {
        concept-name IncreaseDecreaseInIntangibleAssets;
    }
    variable $theStartingBalances {
        concept-name IntangibleAssetsGrossCost;
        instant-duration start $theFlowAddends;
    }
    test {
        $theEndingBalance eq $theChanges + $theStartingBalances
    };
};
```

HMRC (DIMENSIONAL AGGREGATION)

```
value-assertion TotalIntangibleAssetsIncludingGoodwillDefault {
    variable $TotalIntangibleAssetsIncludingGoodwillDefault {
      dimension IntangibleAssetClassesDimension;
    };
    variable $theAggregands {
      bind-as-sequence
      dimension IntangibleAssetClassesDimension
        member $theAggregationSum axis child;
    };
    test {$theAggregationSum eq sum($theAggregands)};
 };
```

HMRC (UNSATISFIED MESSAGE)

```
assertion PropertyPlantEquipmentClassesDimension {

    unsatisfied-message "Consistency checks - Property, plant and equipment - update 07-09-15.
                         aggregation MotorVehicles not equal to sum of members.";

     variable $aggregation {
       explicit-dimension PropertyPlantEquipmentClassesDimension
                member MotorVehicles;
     };
     variable $aggregands {
       bind-as-sequence
       explicit-dimension PropertyPlantEquipmentClassesDimension
              member CommercialMotorVehicles member MotorCars;
     };
     test { $aggregation eq sum($aggregands) };
  };
```

XF CONVERTERS

Arelle open source on GitHub

- Linkbase to xf: plugin formulaSaver.py
  - GUI: load, Tools->Save Xbrl Formula File
  - Cmd line: arelleCmdLine -f {dts} --plugins formulaSaver.py --save-xbrl-formula {file-name.xf}
- Xf to linkbase: plugin formulaLoader.py
  - Stand-alone or integrated xf-to-linkbase converter
    - python3.5 formulaLoader.py [--debug] {files}
      - Linkbase outputs saved as {file}-formula.xml
  - Interactive
    - load in arelle GUI; Tools->Save Xbrl Formula File
    - run in production with xf files instead of formula linkbase files