Formula Meta Description

Author	Piotr Malczak
Reviewers	
Date	2013-10-08
Version	1

Validation rules - types

All business rules have been categorized into a few basic types. All of them are described and explained in the table below.

Basic type	Corresponding XBRL Formula	Description (with examples)
calculation	consistency assertion	The FMD calculation is a formula intended to calculate a value and put the result into a given fact. The calculation is in the form: A = B + C + … As far as the XBRL validator can only check the consistency of a report, the FMD calculation expressed in an XBRL formula can only be transformed into an assertion. The XBRL validator can only check the consistency of a report; therefore, the FMD calculation can be represented in an XBRL Formula only as an assertion. Common mathematical operators, parentheses and functions are allowed. There can be only one argument to the left side and an expression to the right side.
simple logical relationship (simple value assertion)	value assertion	The FMD logical relationship is for examining left and right statement with a logical operator. The logical relationship is in the form: A + B + C + … = X + Y + Z + …
conditional relationship	value assertion + precondition	The FMD conditional relationship is for examining a logical condition but only in a case the first condition (precondition) is true. The conditional relationship is in the form: if (A > 0) then (B + C + .. > = X + Y + ..)
complex relationship (complex value assertion)	value assertion	The FMD complex relationship is for any logical statement composed of all types of validations. The complex relationship is in the form: isNotNull(A) or (B >= C) and (D > 0)
existence assertion	existence assertion	The FMD existence assertion is for examining if a given fact occurs in an instance. For symmetry, the existence of a given fact in an instance file can be permitted. The existence assertion is in the form: isNotNull(A) if the existence of fact A is required, and isNull(A) if the existence of fact A is prohibited.

Validation rules – addressing schemas

All these defined formula types can be used in a few variants depending on the data addressing method. All addressing variants are described below.

Addressing variant	Description
simple	A formula operates just on single facts. Arguments point to just one data point.
dimensional	The dimensional variant means that a relationship described by a expression relates to dimensions. A dimensional rule is applied to a set of concepts – a concept filter is applied to a rule.
‘over-dimensional-mode’	The ‘over-dimensional-mode’ variant of addressing allows the user to express the relationship applied to a concepts. To some extent it is similar to calculation link-base applied to a dimensional model. An XBRL formula is valid for any combination of dimensions. A filter over dimensions can be used, which is an advantage of the ‘over-dimensional-mode’ addressing variant. Therefore it is much more precise than a standard XBRL calculation link-base.
tuple	An FMD is powerful enough to describe assertions within a single tuple. The tuple variant will not be discussed in detail here because of the limited use of tuples in Eurofiling taxonomies.

FMD notation

FMD provides following forms of validation rules

Basic type	**BNF**

simple logical relationship (simple value assertion)	`<value_assertion-simple> ::= <expression><operator><expression>`
	
	`<operator>   ::= "=" | "<" | "<=" | ">" | ">="`
	
	`<expression> ::= <argument><math_operator><argument>`
	`       | <function_name> ″(″ <expression> [″,″ <expression> ] * ″)″`
	
	`<argument> ::= <number>`
	`       | <data_point_address>`
	`       | <expression>`
	`       | "(" <expression> ")"`
	`       | ″′″<literal>″′″`
	
	`<math_operator> ::= "+" | "-" | "*" | "/"`
	
	`<number> ::= decimal number with a precision of two digits`
	
	`<function_name> ::= function ID`
	
	`<literal> ::= string`
	
	`<data_point_address> ::= depending on an addressing scheme:`
	
	`               – explicite address of a single data point,`
	
	`               - a range of data points with a measure filter`
	
	`               - a range of data points with a dimension filter`

Calculation	`<calculation> ::= <data_point_address-single_data_point> ″=″ <expression>`
	
	`<expression> ::= <argument><math_operator><argument>`
	`       | <function_name> ″(″ <expression> [″,″ <expression> ] * ″)″`
	
	`<argument> ::= <number>`
	`       | <data_point_address>`
	`       | <expression>`
	`       | "(" <expression> ")"`
	`       | ″′″<literal>″′″`
	
	`<math_operator> ::= "+" | "-" | "*" | "/"`
	
	`<number> ::= decimal number with a precision of two digits`
	
	`<function_name> ::= function ID`
	
	`<literal> ::= string`

existence assertion	`<existance_assertion> ::= ″isNotNull (″ <data_point_address> ″)″`
	`                         | ″isNull    (″ <data_point_address> ″)″`

Basic type	BNF

complex relationship (complex value assertion)	`<value_assertion-complex> ::= <complex_expression>`
	
	`<complex_expression> ::=`
	`<complex_expression><complex_expression_oper> <complex_expression>`
	`| ”not (” <complex_expression> ”)”`
	`| “true” | “false”`
	`| <value_assertion-simple>`
	`| <existance_assertion>`
	
	`<complex_expression_oper> ::= ”and” | ”or”`

conditional relationship	`<conditional assertion> ::= ”if (” <complex_expression> ”) then (” <complex_expression> ”)”`

data point address	`taxonomy_alias;concept_ID;set_of_dimensions;period_type`
	
	`where`
	
	• taxonomy_alias – an optional alias of a taxonomy namespace, e.g. C-COREP, F-FINREP, etc; an empty field is used for the current taxonomy
	
	• concept_ID – the ID of the concept defined in the taxonomy, e.g. p_ca_SolvencyRatio; it might be replaced by '*' in the 'over-dimensional-mode' addressing variant for each listed concept
	
	• set_of_dimensions – a set of dimension values (if applicable), e.g. d_Counterpts_ByCounterpartiesDimension:d_Counterpts_Retail; a set might be replaced by '*' in the dimensional addressing variant
	
	• period_type – an attribute derived from the taxonomy which covers period aspect using one letter shortcuts: S-start, E-end,D-duration and F-forever. The 'S' and the 'E' refer to instance periods for the beginning and the end of abstractly defined period for XBRL instance. Duration means interval from start to end values. Forever do not have to be additionally parameterized. If start, end or duration period types are used appropriate formulas requires parameters to represent start and end date
	
	The ***concept_ID*** and the ***set_of_dimensions*** are defined by XBRL ID, however other ID's can also be used. In the following examples, next to XBRL ID's, there are also examples with Polish technical labels. Another option is a qname of a concept. This option ensures consistency with any taxonomy.

Examples

<u>Example A simple non-dimensional address</u>

				C for COREP
				XBRL ID
				no dimensions
				end value of instant concepts
C;	p_ca_SolvencyRatio;	;	E	

<u>Example An address with a dimension value</u>

				F for FINREP
				XBRL ID
				dimension
				end value of instant concepts
F;	ifrs_gp_AvailableForSaleFinancialAssetsLoanAndAdvances;	d_Counterpts_ByCounterpartiesDimension: d_Counterpts_Retail;	E	

Simple addressing

Simple addressing doesn't require any special syntax. It is just an arithmetic expression with fields (also known as 'data-points') as arguments.

<u>Example. Simple value assertion – simple addressing</u>

```
"C;p-mi_MultiplicationFactorXAveragePrevious60workingDaysVaR;d-mr_MRiskIMDimension:d-
       mr_MRiskIMTotalPosition;E"

= "C;p-mi_MultiplicationFactorXAveragePrevious60workingDaysVaR; d-mr_MRiskIMDimension:d-
       mr_MRiskIMTradedDebtInstruments;E"

+ "C;p-mi_MultiplicationFactorXAveragePrevious60workingDaysVaR; d-mr_MRiskIMDimension:d-
       mr_MRiskIMEquities;E" + …
```

A more complex addressing scheme requires some syntactical glue. A 'let' clause is used for defining a filter over dimensions and an 'in' clause for an expression.

<u>Example Simple value assertion - 'Over-dimensional-mode' addressing</u>

```
let( dimension = "d-cr_CreditRiskDimension:*,*" )
in( "C;p-cm-ca_CreditRiskCapitalRequirements;*;*" = "C;p-cm-cr_RiskWeightedExposureAmount;*;*" * 0.08)
```

Dimensional addressing

Another complex addressing scheme – dimensional addressing – uses the same syntax as 'over-dimensional-mode' addressing. The only difference is that the 'let' clause contains a measure filter.

Example Simple value assertion – dimensional addressing

```
let ( measure = { 'p-cm-cr_RiskWeightedExposureAmounts', 'p-cm-cr_CreditRiskCapitalRequirements' } )

in ( "C;*;d-ec_ExposureClassDimension:d-ec_IRBECRetail,*;*" =
        "C;*;d-ec_ExposureClassDimension:d-ec_IRBECRetailOfWhichSME,*;*" +
        "C;*;d-ec_ExposureClassDimension:d-ec_IRBECOtherRetail,*;*" +
        "C;*;d-ec_ExposureClassDimension:d-ec_IRBECQualifyingRevolving,*;*" +
        "C;*;d-ec_ExposureClassDimension:d-ec_IRBECSecuredByRealEstate,*;*" )
```

The measure filter assigns a list of concepts to which a calculation will be applied.