Ember+ Formulas

Syntax Description

Author: pbo@l-s-b.de
Date: 2012-06-26

Contents

Introduction

This document describes the syntax of formulas used in the Glow schema, which is part of the Ember+ protocol standard. An Ember+ provider may define a formula for any parameter of numerical type. This formula can be used by consumers to translate the transmitted value of the parameter into the required unit.

A Glow formula is a tuple of two mathematical expressions, each of which applies a projection to one input value (referred to as $ in the syntax).

The two expressions must have the following mathematical relationship:
`expr2(expr1($)) = $ = expr1(expr2($))`

This means, expr2 must inverse the projection applied by expr1 and vice versa, like in these examples:

- expr1 = `$ / 2`
  expr2 = `$ * 2`
- expr1 = `exp($)`
  expr2 = `log($)`

In the case of read-only parameters, the second expression may be omitted.

Technically, Glow formulas support the following data types:

- 64 bit integer, referred to as INTEGER
- 64 bit floating-point, referred to as DOUBLE

When evaluating a formula expression, the data type of the result may be different from the input value's data type.

Examples

1. expr1 = `1 + log($, 2)`
   expr2 = `2^($ - 1)`
2. expr1 = `sin($ / 7.43)` *where $ is of type INTEGER*
   expr2 = `int(7.43 * asin($))`
3. expr1 = `$ - log($ / (e^(-$) * $^sin(1 / $)), 10)`
   expr2 = *exercise left to the reader*

Operators

<u>In ascending order of precedence</u>

- **Addition:**
  Operand1 **+** *Operand2*
  Example:
    - `100 + 12.1`
      = 112.1 (DOUBLE)

  Result:
    - DOUBLE if at least one of the two operands is of type DOUBLE
    - INTEGER if both operands are of type INTEGER
- **Subtraction:**
  Operand1 **-** *Operand2*
  Example:
    - `75 - 5`
      = 70 (INTEGER)

  Result:
    - DOUBLE if at least one of the two operands is of type DOUBLE
    - INTEGER if both operands are of type INTEGER

- **Multiplication:**
  Operand1 ***** *Operand2*
  Example:
    - `12.4 * 2`
      = 24.8 (DOUBLE)

  Result:
    - DOUBLE if at least one of the two operands is of type DOUBLE
    - INTEGER if both operands are of type INTEGER
- **Division:**
  Operand1 **/** *Operand2*
  Example:
    - `1 / 2`
      = 0.5 (DOUBLE)

  Result:
    - DOUBLE
- **Integer Division:**
  Operand1 \ *Operand2*
  Example:
    - `12.4 \ 5`
      = 12 \ 5 = 2 (INTEGER)

  Result:

  - INTEGER

- **Modulo:**

  Operand1 **%** *Operand2*

  Example:

  - `15 % 4.5`

    = 15 % 4 = 3 (INTEGER)

  Result:

  - INTEGER

- **Negation:**

  - *Operand*

  Example:

  - `-11.3`

    = -11.3 (DOUBLE)

  Result:

  - DOUBLE if operand is DOUBLE
  - INTEGER if operand is INTEGER

- **Power:**

  Operand1 **^** *Operand2*

  Example:

  - `2^10`

    = 1024 (INTEGER)

  Result:

  - INTEGER if both operands are of type INTEGER and Operand2 is positive or zero.
  - DOUBLE otherwise.

- **Parenthesis:**

  (*Expression***)**

  Example:

  - `2^(9 + 1)`

    = 2^10 = 1024 (INTEGER)

  Result:

  - DOUBLE if expression is of type DOUBLE
  - INTEGER if expression is of type INTEGER

Built-in Constants

- **$**
  The formula's input value.
  Type:
  - Either DOUBLE or INTEGER
- **PI**
  Mathematical constant π.
  Type:
  - DOUBLE
- **E**
  Mathematical constant e of type DOUBLE.
  Type:
  - DOUBLE

Built-in Functions

- **Sine**
  sin(*Expression***)**
  Expression is in radians
  Type:
    - DOUBLE
- **Cosine**
  cos(*Expression***)**
  Expression is in radians
  Result:
    - DOUBLE
- **Tangent**
  tan(*Expression***)**
  Expression is in radians
  Result:
    - DOUBLE
- **Hyperbolic Sine**
  sinh(*Expression***)**
  Expression is in radians
  Type:
    - DOUBLE
- **Hyperbolic Cosine**
  cosh(*Expression***)**
  Expression is in radians
  Result:
    - DOUBLE
- **Hyperbolic Tangent**
  tanh(*Expression***)**
  Expression is in radians
  Result:
    - DOUBLE
- **Arc Sine**
  asin(*Expression***)**
  Expression is in radians
  Type:
    - DOUBLE
- **Arc Cosine**
  acos(*Expression***)**
  Expression is in radians
  Result:

  - DOUBLE
- **Arc Tangent**
  atan(*Expression***)**
  atan(*Expression1***,** *Expression2***)** *“atan2” function in C*
  Expression is in radians
  Result:
  - DOUBLE
- **Square Root**
  sqrt(*Expression***)**
  Result:
  - DOUBLE
- **Natural Logarithm (base E)**
  log(*Expression***)**
  Type:
  - DOUBLE
- **Logarithm**
  log(*Expression***,** *BaseExpression***)**
  Result:
  - DOUBLE
- **Exponential**
  exp(*Expression***)** *equivalent to e^(Expression)*
  Result:
  - DOUBLE
- **Round to closest integer**
  round(*Expression***)**
  Type:
  - INTEGER
- **Round to next bigger integer**
  ceil(*Expression***)**
  Result:
  - INTEGER
- **Convert to integer, truncating fractional part**
  int(*Expression***)**
  Result:
  - INTEGER
- **Convert to floating-point**
  float(*Expression***)**
  Result:
  - DOUBLE