Logical and Arithmetic Expressions

Syntax Specification

Version	Date	Autor	Changes
0.1	2013-02-19	Peter Pavek	Initial version.
0.2	2013-03-01	Peter Pavek	Values and variables clarified.

Introduction

Background

Teraim develops software for mobile data collection. The key aspect of this software is its configurability and embedded data value validation.

Scope

This document describes the syntax for logical and arithmetic expressions used in the software.

Definitions

Page	Used in this document to mean the contents of the screen – both the visible parts and the parts that may be hidden, but can be scrolled to.
Variable	Data structure for storing a number of data values. A variable has an associated entry field. The graphical layout of the field depends on the type of the variable. Each variable has an “Is Available” flag. By default, it is TRUE.

Syntax

Arithmetic Expressions

Arithmetic expressions are those that evaluate to a number. In this context, they may contain variables. Once these have numeric values, the expressions evaluate to a number.

Syntax

The BNF syntax below defines arithmetic expressions (<*a e*>). Their semantics is also provided.

<*a e*> ::= <*number*> | <*variable*>

where

<*number*> is any floating point number. It may be written with or without the decimal part.

<*variable*> is a name of a variable existing in the system. It may contain only alphanumeric characters a to z, 0 to 9, and _ (underscore) in arbitrary order. The exception being that a variable name may not start with a digit. Upper and lower case characters are allowed, but case is not relevant. Thus, upper and lower case characters are treated as equivalent.

<*a e*> ::= <*a e*> <*arithmetic operator*> <*a e*>
<*arithmetic operator*> ::= + | – | * | / | ^
<*a e*> ::= – <*a e*>
<*a e*> ::= (<*a e*>)
<*a e*> ::= <*unary arithmetic function*> (<*a e*>)
<*unary arithmetic function*> ::=
abs | acos | asin | atan | ceil | cos | exp | floor | log | round | sin | sqrt | tan
<*a e*> ::= <*binary arithmetic function*> (<*a e*> , <*a e*>)
<*binary arithmetic function*> ::= atan2 | max | min

Semantics of Arithmetic Functions

The meaning of arithmetic functions is explained in the table below.

Arithmetic Function	Explanation
abs(<*a e*>)	Absolute value of a number.
acos(<*a e*>)	Trigonometric function arccosine.
asin(<*a e*>)	Trigonometric function arcsine.
atan(<*a e*>)	Trigonometric function arctangent.
atan2(<*a e*>$_1$, <*a e*>$_2$)	Trigonometric function similar to arctangent. Evaluates to the angle, in

	radians, between the positive x-axis and the point (<*a e*>$_2$, <*a e*>$_1$). The angle is positive for counter-clockwise angles and negative for clockwise angles.
ceil(<*a e*>)	Up-rounded number.
cos(<*a e*>)	Trigonometric function cosine.
exp(<*a e*>)	Exponential function.
floor(<*a e*>)	Truncated number.
log(<*a e*>)	Logarithmic function.
max(<*a e*>, <*a e*>)	The bigger of the two arguments.
min(<*a e*>, <*a e*>)	The smaller of the two arguments.
round(<*a e*>)	Rounded value.
sin(<*a e*>)	Trigonometric function sine.
sqrt(<*a e*>)	Square root function.
tan(<*a e*>)	Trigonometric function tangent.

Logical Expressions

Logical expressions are those that evaluate to true or false. In this context, *true* is implemented as 1.0 and *false* is implemented as 0.0. Logical-valued variables are allowed in the expressions.

Syntax

<*logical expression*> ::= 0.0 | 1.0 | <*variable*>
<*logical expression*> ::= (<*logical expression*>)
<*logical expression*> ::= <*logical expression*> OR <*logical expression*>
<*logical expression*> ::= <*logical expression*> AND <*logical expression*>
<*logical expression*> ::= <*logical expression*> EQ | NE <*logical expression*>
<*logical expression*> ::= <*a e*> EQ | NE | LT | LE | GT | GE <*a e*>

Semantics of Logical Operators

All of the implemented operators take two arguments, written with one before and one after the operator (infix notation.)

Logical Operator	Explanation
OR	Combines the two arguments and is true if one or both arguments are true. Otherwise it is false.

AND	combines the two arguments and is true if both arguments are true. Otherwise it is false.
EQ	Combines the the two arguments and is true if they have the same value. Otherwise it is false.
NE	Combines the two arguments and is true if the arguments have different values. Otherwise it is false.
LT	Combines the two arguments and is true if the first is less than the second one. Otherwise it is false.
LE	Combines the two arguments and is true if the first is less than or equal to the second argument. Otherwise it is false.
GT	Combines the two arguments and is true if the first is greater than the second one. Otherwise it is false.
GE	Combines the two arguments and is true if the first is greater or equal to the second one. Otherwise it is false.

References

[WFE] Generic Workflow Engine, 2012, Pavek, P.

[Rules] Rules for Validation, 2012, Pavek, P.