Documentation For Logic Factory

Contents

1 Operations

1.1 Comparison Operations

Comparison Operations allow the user to compare two of the same variable type. The result of these comparisons is a boolean value.

Syntax	**Operation**	**Supported Types**
X = Y	Equals	Integer, String, Boolean and Object
X != Y	Not Equals	Integer, String, Boolean and Object
X < Y	Less Than	Integer
X > Y	Greater Than	Integer
X <= Y	Less Than or Equal To	Integer
X >= Y	Less Than or Equal To	Integer

1.2 Boolean Operations

Boolean Operations allow comparison of two boolean values.

Syntax	**Operation**
$X \wedge Y$	logical AND
$X \vee Y$	Logical OR
$\neg X$	Negation
$X \veebar Y$	Exclusive OR
$X \leftrightarrow Y$	If and Only If
$X \rightarrow Y$	Implies

1.3 Existential Operations

Existential Operations allow the user to write statements about all, or some of the object in the world. All Existential Operations take the following format.

$$\exists[Variable], ... \cdot [Expression]$$

All Existential Operations start with their respective character, and have one or more variables. Each variable is comma(,) separated and then terminated with a dot $(\cdot)$. After the dot is the expression to be evaluated against. Here all named variables are evaluated as objects.

2 Variables

2.1 Natural Numbers

Natural numbers are non-negative integer. These numbers are represented by a string of numerical characters.

Example:
1 <= 35

2.2 Strings

String are series of characters. The notation of strings in Logic Factory are contained by both double and single quotation marks.

Example:
"Hello World"
'Hello World'

2.3 Objects

Objects are containers of one or more variables. They can contains any non-object variable, and have a corresponding field key used to access the variable in the object.

[*Object Name*].[*Field Key*]

Example:
x.name - Retrieves the String value of the objects name.
x.colour - Retrieves the objects Integer Colour value.

Colour

Colour is a special object, that exists in the world. It is used to compare colours in the world in a text format. The Colour values are stored as integers.

Colour.[*colour name*]
Currently the following colours are supported and colour names must be in lower-case. The following are the names of supported colours:

- Red
- Green
- Blue
- Orange
- Yellow
- Purple
- Brown
- Black
- White

3 Libraries

Libraries are a way to define a set of objects that can used to define a world for evaluation in Logic Factory.

Libraries are written in the JSON format.

All libraries have the following fields:

Field Key	**Value Type**	**Required**
library_name	String	✓
version	String	
background	String	
library	Object Array	✓
grid_width and grid_height	Integer	

Library Name

This field defines the name of the library, which will be user to refer to the library throughout the application. It is the name that will be used in the list of libraries available in the Library drop-down list in the application.

Version

The version number of the library.

Background

This defines the background of the application. It supports hexadecimal, decimal and string values for the background. If the colour is a string of characters, it will use one of the supported colours, outlined in 2.3 Objects.

Grid Width and Height

These values define the number of rectangles high and wide, to draw a grid.

Library

The array of objects that can exist in the world.

3.1 Library Items

Field Key	**Value Type**	**Required**
type	String	✓
image_path	String	Images Only
poly	Integer	Polygons Only
def_col	Colour	Polygons Only
radius	Integer	Circles only
size	Integer	Non-rectangle, Non-circle Polygons
Width and Height	Integer	Images and Rectangles

Type

This is the title name give to the object.

Images

If the object is an image, it has an image_path field, which defines where the image is stored.

It also requires a width and height to define how large to draw the image.

Polygons

This defines the number of edges a polygon has. All polygons require a size, which defines the diameter of the polygon. It has two special cases, circles and rectangles.

The first special case is the circle, which is defined by a poly value of '0'. It requires a radius field, instead of the size field.

The second special case is the rectangle, which is defined by the poly value of '4'. It requires a width and height field instead of the size field to define it.

4 User Interface

4.1 World

The world view is located on the left side of the screen. It serves as a platform in which objects of the world can be compared.

Objects in the world can be selected and moved around. A user can select an object by clicking on it. The user will know when an object has been selected because it appears a shade lighter in colour than the other objects in the world. Objects can be moved around the world environment by holding the mouse down on them and dragging them around the world. When the user releases the mouse at a position the object is placed there. Certain libraries have a grid environment in which objects are automatically snapped to a square on the grid when the user releases an object near that position on the grid.

4.2 Adding Objects

Objects can be added to the world by the users, through a panel on the top right of the interface. A variety of objects are available based on the current library. For example in the geometry library a user can add various shaped objects to the world.

The panel has two drop-down boxes and a button title 'Add Object'. The top drop-down allows the user select and object from the current library. The lower drop-down allows the user to select a colour for objects that get added to the world, this only works for geometric shapes.

The 'Add Object' will add a new object to the world based on the current selection of the two drop-down boxes.

4.3 Modifying Objects

Users can modify objects in the world if they want to via the centre panel. A selected objects can be made bigger, smaller, copied or removed, using the relevant buttons in the panel. You can also add or change the name of an object, via the 'Object Name' text box. This serves as an identifier. However multiple objects can have the same name.

4.4 Adding Expressions

Expressions can be entered via the right panel, in the text box. Various symbol buttons line the bottom edge, to aid input, not accessible on a keyboard. These inputs can also be access via keyboard shortcuts.
The Expression can be added to the Expression Log by Clicking the 'Add Expression', which is required to evaluate an expression.
The expression text box can also be emptied by clicking the 'Clear Expression'

4.4.1 Keyboard Shortcuts

Symbol/Action	**Shortcut**
Add Expression	Enter Key
$\neg$	˜
$\wedge$	ˆ
$\vee$	—
$\veebar$	\x
$\rightarrow$	$->$
$\leftrightarrow$	$<->$
$\forall$	\a
$\exists$	\e
$\cdot$	:

Logic statements (expressions) can be added to the evaluator which is on the left side of the screen. After being typed into the text box they can be added to the evaluator by pressing the add expression button. Suppose if the user types in an expression that is syntactically wrong, an error message appears below it to indicate where exactly the expression is incorrect.

4.5 Expression Log

When a user adds an expression to the evaluator, it gets populated in the expression log. When the 'Go' button below the log is pressed all of the expressions in the log will be evaluated to either true or false, against the current world. An expression on the log can be copied back on to the text box when they are clicked on. There is a small 'X' on the right side of expression on the log. This can be clicked to remove the expression from the log.

4.6 Utility Bar

Various user functions can be accessed via the utility bar in the top of the application.

4.6.1 Library Selection

The library can be changed via the drop-down box titled 'Current Library', and will display the name of the current library being used. Libraries are independent and changing libraries will remove all object from the current world.

4.6.2 Saving and Loading

The current list of expression in the Expresssion Log as well as objects contained in the world can be saved to a local file, by pressing the 'Save File' button. Clicking the button will prompt the user for a file name, if none is give the file will be called 'save.json'.

Files can also be loaded from a local directory with the 'Load File' button. Which will load all objects and expressions in the file into the application, and change the current library to match the files.

4.6.3 Cheat Sheet

If an user is having trouble they can access a cheat sheet that gives a run-down of basic functionality of the application.