COMS 4115: Programming Languages and Translators

SBML: Shen Bi Ma Liang

Chinese Character: 神筆馬良

English Translation: Magic Pen Boy

Language Reference Manual

Bin Liu (bl2329@columbia.edu)
Yiding Cheng (yc2434@columbia.edu)
Hao Li (hl2489@columbia.edu)
Wenhan Zhang (wz2174@columbia.edu)

Oct. 22, 2008

1. Introduction

SBML is a computer language to generate webpage with versatile objects only based on the program designed by web designer. The basic idea of this language is to firstly understand the web design from the user, and then implement the demanded work via generating corresponding Javascript code, which can be run directly by web browsers.

The name, SBML, inspired by a famous Chinese fairy tale, Shen Bi Ma Liang (Chinese characters: 神筆馬良), which is the story of a boy who has a magic pen, and whatever he drew with that pen, it going to turn alive. Unfortunately, the language can not effectively turn drawings alive, but it can make human thoughts alive on webpage.

SBML actually saved web designers from spending their time to learn various web development techniques, what they only need is the language designed by us which is easily understandable, then they don't have to anything else to design the web. By using SBML, user could create fantastic web pages with various complex shapes. Also, to make the drawings alive, users are allowed to attach events to every shape they create.

SBML also supports business application. By importing our standard library, designers may create lots of chart, such as Bar, Line which are Microsoft Excel compatible, which could be easily utilized to show demos and other fancy stuff to promote your business.

2. Lexical Conventions

2.1 Keywords

The following identifiers are reserved for use as keywords in SBML language:

int	*bool*	*String*
Color	*Start*	*Born*
Draw	*Mem*	*for*
if	*return*	*else*
Coordinate	*Line*	*Label*
Content	*Page*	*Image*
Circle	*Rect*	

2.2 Identifiers (Names)

Identifiers are sequences of letters and digits. Identifiers are case sensitive.

2.3 Constants

There is a kind of constant, as follows:

Integer Constants

An integer constant is a sequence of digits. Integers in SBML are all taken as decimal. The keyword to declare an integer is *int*.

2.4 String

In SBML, a string is a sequence of characters surrounded by double quotes “ " ”.

2.5 Others

In SBML, we use semicolon “ ; ” as a line break and support empty symbol which use space “ ” to express.

3. Data Types

```
type : int | string | build-in-shape;
```

Types	Description
Integer	A basic type that contains a 32 bit signed integer, with a range of -2147483648 to 2147483647
String	A basic type that contains a sequence of characters like
Line	A built-in type used to draw a line, which has 4 parameters: xstart, ystart, color, stroke
Point	A built-in type used to draw a point, which has a certain postion, containing 2 parameters: xstart, ystart
Circle	A built-in type used to draw a Circle, which has 4 parameters: radius, heart, fillcolor, color
Image	A built-in type used to load a picture on the page
Page	A built-in type used to add a page on the website
Coordinate	A built-in type which has two parameters:xstart,ystart
Rect	A built-in type which expresses a rectangle in the image with 6 parameters, position, color, content, stroke, width, height
Label	A built-in type which expresses a data type of a sequence of characters (or a class used to draw a string, with 5 parameters: content, font, size, color, coordinate)

3.1 Built-in Types

```
   build-in-shape : Page | Rect | Line | Image | Circle | Color |
Coordinate | Label;
```

- Color

Color is a type consisting of 4 parameters, the first one for a measurement of transparency, and the latter three parameters for each chromatic component of a pixel: red, green and blue. When a color object is created, its properties are all initialized to the value zero.

```
Color
{
      int A;
      int R;
      int G;
      int B;
}
```

- Label

Label is a data type of a sequence of characters (or a class used to draw a string, which has 5 parameters: content, font, size, color, coordinate.)

```
Label
{
      String content;
      String font;
      int size;
      Color color;
      Coordinate position;
}
```

Coordinate

Coordinate describes the position of a point in the image by its x and y coordinates. When a coordinate object is created, its properties are all initialized to the value zero.

```
Coordinate
{
      int x;
      int y;
}
```

Line

Line is a class used to draw a line, which has 4 parameters: xstart, ystart, color, stroke.

```
Line
{
      Coordinate position;
      Color color;
      int stroke;
}
```

Circle:

Circle is a class used to draw a Circle, which has 4 parameters: radius, heart, fillcolor, color.

```
Circle
{
      int radius;
      Color color;
      Coordinate heart;
      int stroke;
}
```

Rect:

Rect describes a rectangle in the image with 6 parameters, position, color, content, stroke, width, height.

```
Rect
{
      Coordinate position;
      Color color;
      Text content;
      int stroke;
      int width;
      int height;
}
```

4. Operators and special charcacters

4.1 Operators

4.1.1 Arithmetic operator

+
Usage example: `operation_code + operation_code` // add the two operation codes

-
Usage example: `operation_code - operation_code` // the fore operation code subtracts the latter

*
Usage example: `operation_code * operation_code` // the fore operation code multiplies the latter

/
Usage example: `operation_code / operation_code` // the fore operation code divides the latter

All arithmetic operators are binary operators, among them, multiplication and divide have the same precedence which is higher than plus and minus, as the same as in basic math.

4.1.2 Comparison operator

In SBML, “<”, “>”, “==”, “<=”, “>=”, “!=” are defined “less than”, “greater than”, “equal to”, “equal or lesser than”, “equal or greater than”, “unequal to” separately as comparison operators, which are used between two operation codes as the same as the computing operators.

4.1.3 Logic operator

In SBML, “||”, “&&” are defined “or”, “and” separately as logic operators, which are used between two operation codes as the same as the operators mentioned above.

4.2 Special Characters

In SBML, there are some special characters with functional significance:

Special Characters	Use	Example
()	Expression grouping; also used as a function parameter list delimiter	string a();
{}	Initializer list, function body	for a = 1 to 2 { ; }
" "	String literal	string str = "hello world";
,	Argument list separator	string getstring(string a, string b)
;	Line break	a=5;
>	Draw a graph onto a page	Draw rect > page;
.	Get the attribute belonged to an identifier	Rect.width = 5

5. Functions

5.1 User Defined Functions

SBML allows users to create their own functions to realize the specific operations desired by their on-going design projects. Also the user-defined functions could be stored in a shared library in order to be accessed and utilized by multiple users or for multiple projects.

5.2 Function definition syntax

```
function_declaration : type identifier '('
      (declaration (','declaration)* | empty )')'
      '{'
            statements (return_statement semicolon)?
      '}'
```

5.3 Built-in Functions

```
build-in-fun : fun_start | fun_draw | fun_mem | fun_born;
```

- **Start** (used to create a new page)

Usage example: Start page p; //starts a page p

```
fun_start : 'Start' 'Page' identifier;
```

- **Born** (used to create a new object)

Usage example: Born Circle c; //creates a circle c

```
fun_born : 'Born' '(' attribute_list ')';
```

- **Draw** (draw an object to a page)

Usage example: Draw c>p; //draws circle c to page p

```
fun_draw : 'Draw' identifier '>' identifier;
```

- **Mem** (save a page)

Usage example: Mem p; //saves a page p

```
fun_mem :'Mem' identifier;
```

6.Scope Rules

In SBML, the scope rules are very similar to those in C or C++. A variable or function is not available until it is declared. Functions cannot be nested and therefore cannot be overridden after they are declared. They are declared in the global scope and available until the program completes execution. A variable is available until the end of the block, in which it was declared, is reached. In the case of variable declared in the global scope, the variable only goes out of scope on program termination. Nested blocks can access variables defined in parent blocks. If a new variable is declared in a nested block with the same name as a variable from a parent block, said variable was overridden. The nested block will then only have access to the new variable from point of declaration. When the block is closed, the original variable will return to scope.

7.Expressions

7.1 Primary Expressions

Primary expressions in SBML involving

right_value (identifier, digits, string, binary_operation), declaration, function_call.

7.2 Digit(s)

The expression *digit* is defined to include *'0'-'9',* and *digits* is the closure of *digit*, which denoted as *digit digit **

7.3 Letter

The expression *letter* includes the lower case letters *'a'-'z'* and the upper case letters *'A'-'Z'* in English alphabet.

7.4 Identifier

The expression *Identifier* is the concatenation of the expression *letter* and the closure of *digit* or *letter*, which is denoted by *letter(digit | letter)*.*

7.5 Operation_code

The expression *operation_code* includes *digits* or *identifier*, denoted by *digits | identifier*

7.6 Binary_operation

The expression *binary_operation* has the form of the concatenation of a *digits* or *identifier* with a operator followed by another *digits* or *identifier*, denoted by

(digits | identifier)(arithmetic operator | comparison_operator | logical_operator)(digit | identifier)

7.7 Argument_list

Argument is separated by ","

```
argument_list : empty | identifier (',' identifier)*;
```

7.8 Attribute_list

```
attribute : type ':' right_value;
attribute_list : (attribute)* ;
```

7.9 Set and get value

```
set_value : identifier'.'type;
```

8. Statements

In SBML, statements are execute sequentially, the following are the types of statements:

```
statement : declaration | function_call | assignment  |
binary_operation | build_in_function | if_statement | for_statement;

statments : (statement semicolon)*;

program : ( statements | function_declaration )*;
```

- **If statement:**

```
if_statement : 'if' '(' binary_option ')'
        '{'
             statements
        '}'
             'else'
        '{'

             Statements
        '}'
```

- **Iteration statement:**

```
for_statement : 'for' identifer '=' digits 'to' digits
        '{'
             statements
        '}'
```

- **Declaration statement:**

```
type identifier;
```

- **Assignment statement:**

```
( identifier.attribute | type identifier | identifier ) = (string |
digits | identifier | binary_operation)
```

- **Return statement:**

```
return_statement : 'return' identifier | digits | string |
binary_operations;
```

- **Function call statement:**

```
function_call : identifier '(' argument_list ')';
```

9. Sample Code

This sample code is going to generate a rectangle with a label “hello world” in the page.

```
Start Page page;

string str = "hello world";

Born Color color = Color{R:0f;G:57;B:89;};

Born Label label = Label{Content:str;Color:color;};

Born Rect rect = Rect{Label:label;Color:color;Width:100;Height:100;};

Draw rect > page;

Mem page;
```