Model-Driven Software Development

Group project External DSL

Quick Math

Authors:
Mohebullah Toofan motoo14@student.sdu.dk
Nasib Sarvari nasar14@student.sdu.dk
Štefan Töltési uttol17@student.sdu.dk

Date:
15th of May 2018

Problem Domain

It can be difficult to visualize functions in Java. You have to import libraries and write complex code to make the charts. In order to save time and reduce the complexity of making charts, many elements of the generic purpose language (GSL) Java can be excluded.

Therefore we made our own domain specific language (DSL), where the programmer is able to write functions with parameters to compute them and visualize them. The code structure was simplified, so the code was specific to our domain - generating charts. Our external DSL is called Quick Math and it uses XChart, which is an open source library for Java[1], for generating charts.

Metamodel

Figure 1 shows a metamodel of our external DSL Quick Math. It shows how Quick Math is structured internally.

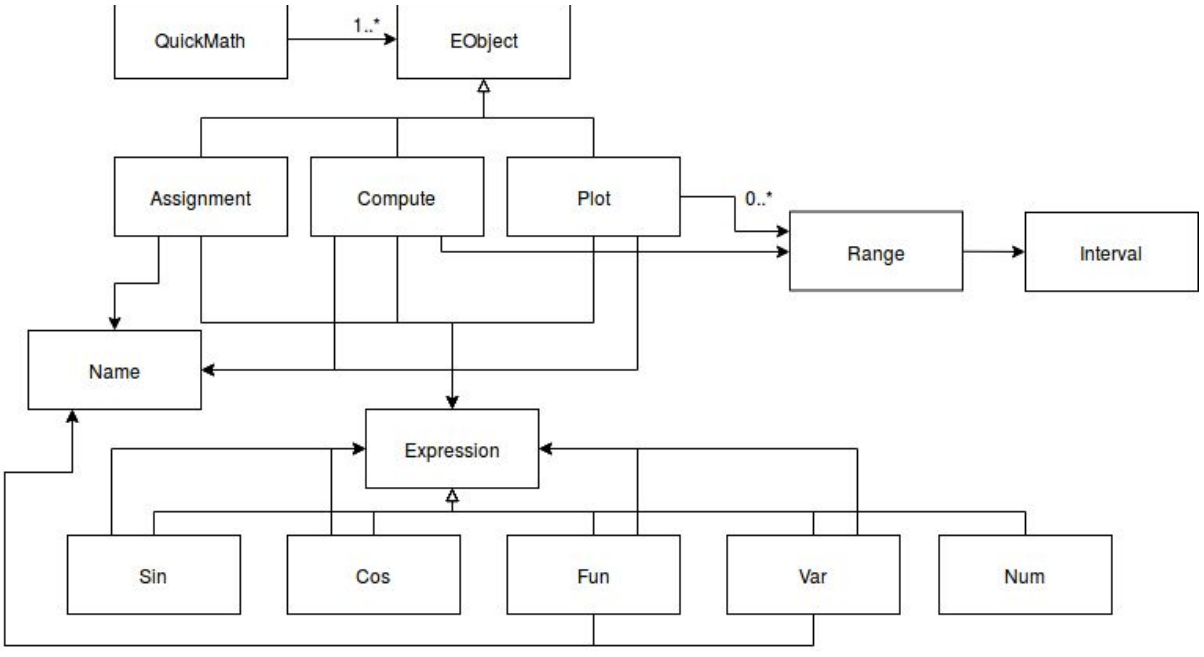

Figure 1: Metamodel of Quick Math

Not all the expressions are shown. Parts used in mathematical expression assignment are excluded. Such as Mult, Div, Plus, Minus. Adding more predefined mathematical functions is easy in our metamodel.

[1] https://knowm.org/open-source/xchart/

Example of Program

```
sheet Example1 //output generated into Example1.png
pow(x) = x*x
plot pow(par) with par in [0,100] step 1 // x axis is called par in result

func(a,b) = a*pow(b) - (sin a / b)
compute func(x,z) with x in [~2,10] step 0.2 and z in 3
func2(x) = x - sin x
plot func2(x) with x in [0,6] step 0.02
```

Code Snippet 1: Example of the external DSL Quick Math

User can assign mathematical expressions to variables. Those can be used later since they are saved in global scope.

There are two main commands present in our DSL.

- Compute
- Plot

Compute command computes a value a function and prints the result with input parameters to standard output.

Plot command is using XChart library and generates a graph that is shown to user and saved to local directory.

Both of commands support interval expressions that generates input values for computations of functions.

Example of Generated Output

The plot below is generated based on the function (*func2* in Code Snippet 1)

We support visualization of functions that has one input variable since we are generating 2D graphs. For functions that have more input variables user is still able to compute result value with implemented compute command in Quick Math.

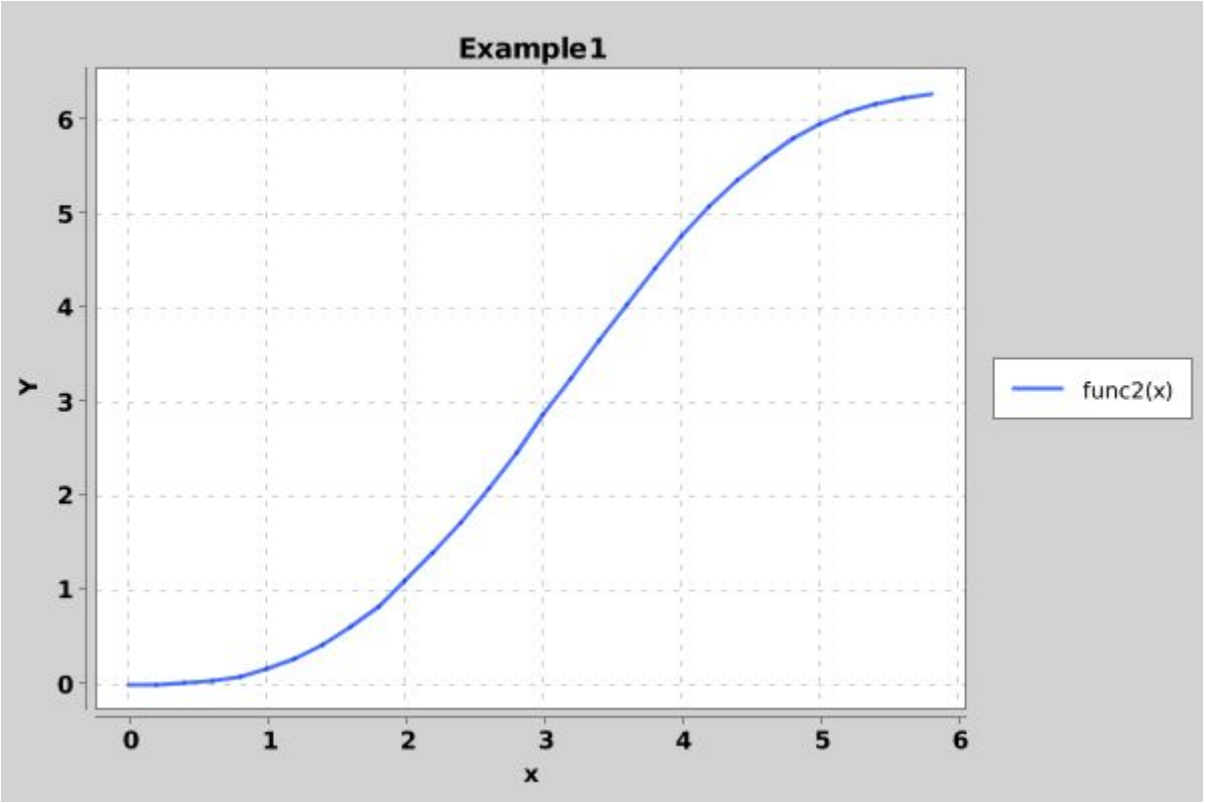

Figure 2: Plot based on func2 in Code Snippet 1

Computed output for compute *func* is:

x:	z:	result:
-3.0	3.0	-26.95295999731338
-2.9	3.0	-26.020250223595337
-2.8	3.0	-25.088337283281366
-2.7	3.0	-24.157540039922058
-2.6	3.0	-23.228166209392846

Future Work

Begin with validation so user enters valid Quick Math programme. Most important validation would be in intervals where we want to check that number of steps for each interval is equal.

Another thing to check is if the user is using defined variables in Expressions.

Those are things that a grammar can not handle.

Another improvement could be making more use of XChart library and adding an option to draw several functions into one file and customize the visuals of the generated file.

Quick Math Setup

1. Create an Xtext project
2. Add the Xtext grammar file
3. Add generator file
4. Add the FunctionScope.java class inside the generator folder.
5. In zip file there is a XChart library that needs to be added to manifest file of Eclipse project. Runtime tab and Add Classpath to XChart jar file.