SBMLToolbox

Version 3.1

January 2010

[Note: Changes from V 3.0 are marked in red]

Testing

Sarah M Keating

http://www.sbml.org
mailto:sbml-team@caltech.edu

SBMLToolbox-3 provides extensive testing of the functions supplied using an easily extensible approach that facilitates the testing of user developed functions.

1. Top-level Test directory

The top-level toolbox directory contains a subdirectory Test. The contents of this directory are listed in Table 1.

Table 1: Contents of the top-level Test directory

Name	Purpose
RunTest.m	Function that runs all the tests within the toolbox
TestFunction.m	Generic function to apply a given function to a given set of arguments and compare the output to the given expected output
TestOutput.m	Functions that tests the OutputSBML function
CompareFiles.m	Function that compares the content of two given text files
/test-data	Subdirectory containing a number of SBML files used by tests within the toolbox

1.1 RunTest

RunTest assumes a particular directory structure and as such should be called from its home directory. This function calls a test function within each Test subdirectory of the toolbox followed by the TestOutput function; thus testing the entire toolbox.

1.2 TestFunction

Format	y = TestFunction(func_name, no_input, no_output, varargin)	
Argument(s)	func_name	name of function to test
	no_input	number of input arguments
	no_output	number of returned variables
		list of input arguments
		list of expected output variables
Returns	1	if the output from the function does NOT equal the expected output supplied
	0	otherwise (actual output = expected output)

NOTE: Cannot deal with functions requiring more than 3 input arguments or functions expecting more than 3 output arguments.

EXAMPLE:

Consider the function GetGlobalParameters

GetGlobalParameters
takes a SBMLModel
and returns

1) an array of character names representing all global parameters within the model
2) an array of the values of each parameter

A model m has two parameters, ‘p1 = 4’ and ‘p2 = 3’.

y = TestFunction(‘GetGlobalParameters’, 1, 2, m, {‘p1’, ‘p2’}, {4, 3})

returns y = 0 (actual output EQUAL TO expected output)

y = TestFunction(‘GetGlobalParameters’, 1, 2, m, {‘k1’, ‘p2’}, {4, 3})

returns y = 1 (actual output NOT EQUAL TO expected output)

1.3 TestOutput

The TestOutput function:

1) loads each of the SBML files in the test-data directory
2) writes the file out into a created directory Out-test using the OutputSBML function
3) compares the output file with the original file and reports any errors.

1.4 CompareFiles

CompareFiles performs a line by line comparison between two text files.

Format	y = CompareFiles(file1, file2)	
Argument(s)	file1	filename of first file
	file2	filename of second file
Returns	1	if the first file does NOT match the second file
	0	otherwise (file1 = file2)

1.5 test-data directory

The test-data subdirectory contains a number of SBML files. Each of these contains a valid SBML model designed to provide different components and aspects of the SBML language.

Filename	Description
algebraicRules.xml	Contains algebraicRules; assignmentRule and reactions.
csymbolDelay.xml	Uses the csymbol delay
csymbolTime.xml	Uses the csymbol time
funcDefsWithInitialAssignments.xml	Contains initialAssignments that use functionDefinitions
functionDefinition.xml	Contains functionDefinition used within rules and reactions
initialAssignments.xml	Contains initialAssignments for parameter, species and compartment
l1v1.xml	Contains all components present in SBML Level 1 Version 1
l1v2-all.xml	Contains all components present in SBML Level 1 Version 2
l2v1-all.xml	Contains all components present in SBML Level 2 Version 1
l2v2-all.xml	Contains all components present in SBML Level 2 Version 2
l2v2-newComponents.xml	Contains all components introduced in SBML Level 2 Version 2: compartmentType; speciesType; initialAssignment & constraint
l2v3-all.xml	Contains all components present in SBML Level 2 Version 3
l2v4-all.xml	Contains all components present in SBML Level 2 Version 4
nestedPiecewise.xml	Contains a reaction that uses a nested piecewise operatior in the MathML
piecewise.xml	Contains a reaction that uses a piecewise operatior in the MathML
rateRules.xml	Contains a rateRule for a species
sparseStoichiometry.xml	Contains a model for which the stoichiometry matrix is sparse
species.xml	Contains a number of species

The directory also contains the file SBML_Models.mat. This is a MATLAB data file containing the MATLAB_SBML model structures for each of the test-data files. These can be loaded directly to the MATLAB workspace or accessed via the SBMLToolbox functions in the StoreModels directory.

These files are used by tests in most directories of the toolbox.

2. Other Test directories

Each directory of the toolbox has a Test subdirectory.

Each Test subdirectory contains a set of files named TestSomeFunction.m; where SomeFunction is the name of the function that is tested by that particular set of tests. These test functions use the 'TestFunction' utility described in section 1.2 above.

EXAMPLE:

Consider the function GetGlobalParameters in the AccessModel directory.

In the directory AccessModel/Test there is a TestGetGlobalParameters function containing a number of tests constructed as shown:

```
m = TranslateSBML('../../Test/test-data/initialAssignments.xml');

names = {'k', 'k1', 's1', 's2', 's3', 'c', 'c1'};
values = [6, 2, 3, 4, 1, 6, 2];

fail = TestFunction('GetGlobalParameters', 1, 2, m, names, values);
```

Each Test subdirectory also contains a test_xyz.m function which runs all the tests in that particular subdirectory.

The RunTest function in the top-level Test directory calls the test function from each of the toolbox subdirectories.