Structural Coverage Demo with Model Instrumentation

The path for the demo folder is as follows:

```
\demos\Structural_Coverage\With_Model_Instrumentation
```

It contains one Simulink model named as:

```
demo_file.mdl
```

Model Instrumentation

In this demo, the simulink model is instrumented automatically and the following files are created:

1-The Simulink model with corresponding output ports:

```
demo_file_outports_added.slx
```

2-The Black Box model with multiple Hybrid Automatons:

```
BlackBox_demo_file.m
```

Black Box model contains Hybrid Automatons of four switch blocks (with 2 states each) and one saturation block (with 3 states). Each Hybrid Automaton has its corresponding guards and adjacent nodes

Testing with S-Taliro

To run demo the following command should be used:

```
>>staliro_Instr_StrCoverage.m
Running S-TaLiRo
STRUCTURAL_COVERAGE
MODEL_INSTRUMENTATION
```

After model instrumentation the structural coverage is initiated.

The original specification is

```
phi = '(!(<>_[0,0.1](predi1/\predi2/\predi3)))';
```

In first iteration, s-taliro runs the ‘SA_Taliro’ algorithm for 100 numbers of tests:

```
Run number 1 / 1
 Initial robustness value ==> <0,93.2128>
Best ==> <0,92.5001>
...
Best ==> <0,90>
100 Acceptance Ratio=0.68 beta=-33.75,0.99
  Running time of run 1: 6.8762 sec
```

s-taliro saves the location traces in the history.

For this demo, structural coverage will run 10 iterations and in each iteration it updates the location history and it chooses the next location based on the unobserved location combinations.

The first iteration of structural coverage and the corresponding location history is as follows:

```
...

Iteration number is
     1
Location History
     2     1     1     2     3
     2     1     1     2     2
     2     2     1     2     2
     2     2     1     2     3
...
```

Predicate location is created based on the random location combination that does not exist in the location history:

```
next location =
     1     2     1     1     1
Location Predicate:
    str: 'staliroStrCovPred1'
      A: []
      b: []
    loc: {[1]  [2]  [1]  [1]  [1]}
```

then it updates the specification as follows:

...

```
((!(<>_[0,0.1](predi1/\predi2/\predi3))))\/!<>_[0,inf)staliroStrCovPred1
```

And the new specification is used for the next iteration of s-taliro:

```
Run number 1 / 1
 Initial robustness value ==> <0,111.8507>
Best ==> <0,107.2888>
...
Best ==> <0,90.0001>
100 Acceptance Ratio=0.6 beta=-33.75,0.99
  Running time of run 1: 16.04 sec
```

And next iteration of structural coverage will continue and at each time it updates the location history and chooses the location based on the unobserved location combination similar to previous iteration:

```
Iteration number is
     2
Location History
     2     1     1     2     3
...
     1     1     1     2     1
next location =
     1     1     2     2     2
Location Predicate:
    str: 'staliroStrCovPred2'
      A: []
      b: []
    loc: {[1]  [1]  [2]  [2]  [2]}
((!(<>_[0,0.1](predi1/\predi2/\predi3))))\/!<>_[0,inf)staliroStrCovPred2
```

At the end of the structural coverage flow, this demo creates the four following lists:

1- List of all visited location combinations which is the history of all location combinations that are observed so far:

```
List of all visited locations:
     2     1     1     2     3
     2     1     1     2     2
     2     2     1     2     2
 ....
```

2- List of all the location combinations that structural coverage randomly chose and tried as the new location predicate:

```
List of location predicates that are tested:
     1     2     1     1     1
     1     1     2     2     2
     2     2     1     1     1
....
```

3- When structural coverage tests new location predicates, it monitors the executions to see whether the new location predicate forces the system to go to that location or not. Therefore, it creates the list of location predicates that are observed by the s-taliro. This list is sub set of the previous list:

```
List of location predicates that are visited:
      2     2     2     1     3
```

4- The list of unobserved location predicate is the list of location predicate that are not seen in the location history which means the system is not forced to go to those location. This list is also a sub-set of the second list:

```
List of location predicates that are not visited:
     1     2     1     1     1
     1     1     2     2     2
     2     2     1     1     1
...
```