Structural Coverage Demo

The demos are available in the below path:

```
\demos\Structural_Coverage
```

There are five demos in this folder, the files for each demo are as follows:

Demo 1: (HA3-Model)

```
HA3_model.mdl
BlackBoxHA3.m
staliro_HA3_StrCoverage.m
```

Demo 2: (Heat Benchmark)

```
heat30.mat
BlackBoxHeat1.m
BBox_Heat_01.mdl
staliro_heat_bench_demo_BBox_StrCvrg.m
```

Demo 3: (Navigation Benchmark)

```
navbench_hautomaton.m
navbench_Red.m
Red.fig
```

Demo 4: (Navigation Benchmark)

```
navbench_hautomaton.m
navbench_Yellow.m
Yellow.fig
```

Demo 5: (Navigation Benchmark with Coverage Metrics)

```
navbench_hautomaton.m
staliro_navbench_coverageTest.m
```

The detailed explanation for each demo is as follows:

Demo 1: (HA3 Model)
In this demo, the manually instrumented model is saved as:

```
HA3_model.mdl
```

Prior to opening the file, the following command may need to be run for certain Matlab versions:

```
slCharacterEncoding('Shift_JIS')
```

HA3_model.mdl contains three switches where their HA's location values are considered as outputs 11~13 in the '.mdl' file. Therefore, we have three hybrid automatons.

The black box script of HA3 model is saved in the following file:

```
BlackBoxHA3.m
```

Where each of the three Hybrid Automatons have their corresponding guards.

Testing)

To run demo the following command should be used:

```
>> staliro_HA3_StrCoverage

Running S-TaLiRo
STRUCTURAL_COVERAGE

Run number 1 / 1
 Initial robustness value ==> <0,118.3078>
...
```

At first iteration, s-taliro runs the 'SA_Taliro' algorithm for 1000 tests, without hybrid distance and it fails to falsify the specification after 1000 runs, then structural coverage runs to check which location is observed:

```
...

1000 Acceptance Ratio=0.6 beta=-576.6504,0.99
  Running time of run 1: 70.4878 sec
Iteration number is
     1
Location History
     2     2     2
...
```

This shows that hybrid automatons only visited the location "2 2 2". Which means that location 1 was never visited in the previous execution. Therefore structural coverage chooses the location "1 1 1" for all three unvisited locations as the next location.

```
...
next location =
     1     1     1
...
```

And then it creates the new location predicate for next iteration of s-taliro

```
Location Predicate:
    str: 'staliroStrCovPred1'
      A: []
      b: []
    loc: {[1]  [1]  [1]}
```

and updates the specification:

...

```
((!(<>_[0,0.1](predi1/\predi2/\predi3))))\/!<>_[0,inf)staliroStrCovPre
d1
```

...

Running the s-taliro with new specification and coverage metric leads the s-taliro to falsify the specification:

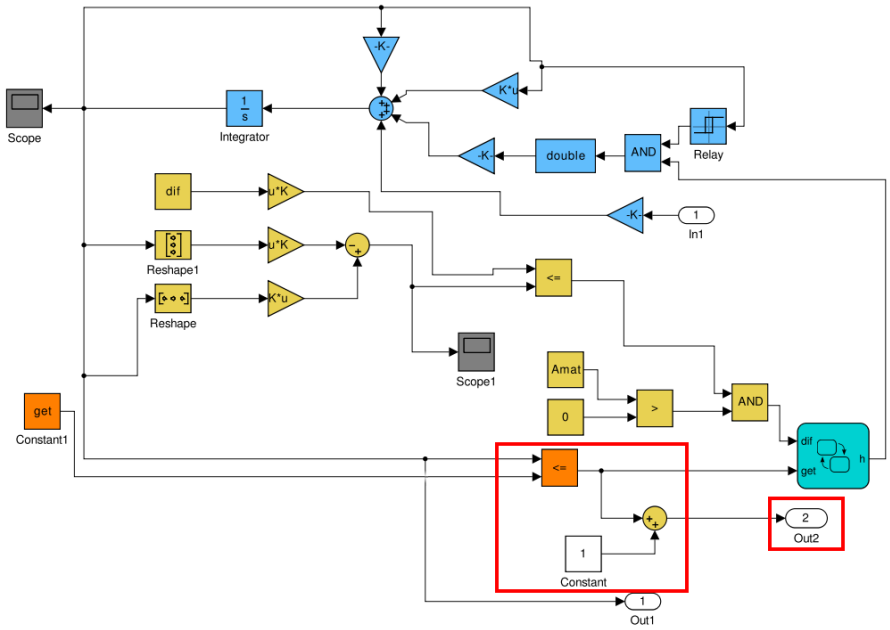
```
Run number 1 / 1
 Initial robustness value ==> <1,174.0308>
Best ==> <1,163.7137>
...
Best ==> <0,4.0213>
Best ==> <0,-1.8176>
FALSIFIED at sample ,150!
  Running time of run 1: 146.2973 sec
Stop
Elapsed time is 203.529367 seconds.
```

Demo 2: (Heat Benchmark)

In this benchmark, the manually instrumented model is saved in the following file:

```
BBox_Heat_01.mdl
```

In this model we considered the structural coverage of the temperature of 10 different rooms, and each room's temperature is compared with a constant value "get" using relational operator. The output of the relational operator for each room creates the mode of each hybrid automaton (which is send to out2 port) which corresponds to "get" control signal.



The script of 10 hybrid automatons are given in the following file:

```
BlackBoxHeat1.m
```

and the guards of each HA is created based on the constant value “get”.

Testing:
To run demo the following command should be used:

```
>> staliro_heat_bench_demo_BBox_StrCvrg
STRUCTURAL_COVERAGE

Run number 1 / 1
 Initial robustness value ==> 2
Best ==> 2
...
```

s-taliro saves the location traces in the history.

In this demo three iteration of structural coverage will run and at each time it updates the location history and chooses the location based on the unobserved location combination.

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

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

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

```
next location =
     1     1     1     2     2     1     2     2     2     1
```

And we create the new location predicate for next iteration of s-taliro

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

and updates the specification:

...

```
([]p)\/!<>_[0,inf) staliroStrCovPred1
```

...

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

Structural coverage continues the iteration for two more times where it finally tried three location combination. 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
     1     1     1     1     1     1     2     1     1     1
...
next location =
     2     1     2     1     1     2     2     2     2     1
Location Predicate:
    str: 'staliroStrCovPred2'
      A: []
      b: []
```

```
    loc: {[2]  [1]  [2]  [1]  [1]  [2]  [2]  [2]  [2]  [1]}
([]p)\/!<>_[0,inf)staliroStrCovPred2
...
...
...
Iteration number is
     3
Location History
     1     1     1     1     1     1     2     1     1     1
...
next location =
     2     1     1     2     2     2     2     2     2     2
Location Predicate:
    str: 'staliroStrCovPred3'
      A: []
      b: []
    loc: {[2]  [1]  [1]  [2]  [2]  [2]  [2]  [2]  [2]  [2]}
([]p)\/!<>_[0,inf)staliroStrCovPred3
```

At the end of simulation, this demo creates the following four 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:
     1     1     1     1     1     1     2     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     2     1     1     1     1
     1     1     1     1     1     1     1     2     1     1
     1     2     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     2
     1     1     1     1     1     1     1     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     1     1     2     2     2     2     2     2     2
     2     1     2     1     1     2     2     2     2     1
     2     1     1     2     2     2     2     2     2     2
```

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. However in this demo this list is empty which means the location predicates did not guide the system to go to the unvisited location:

```
List of location predicates that are visited:
List is empty
```

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

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

Demo 3: (Navigation Benchmark)

This demo considers the following hybrid automata as the simulation model

`navbench_hautomaton.m`

and it uses the original s-taliro for the falsification.

Testing:
To run demo the following script should be used:
`navbench_Red.m`

At first iteration it considers the following specification (not eventually visit red area):

'!<>redArea';

Which runs 100 tests and cannot falsify the specification.

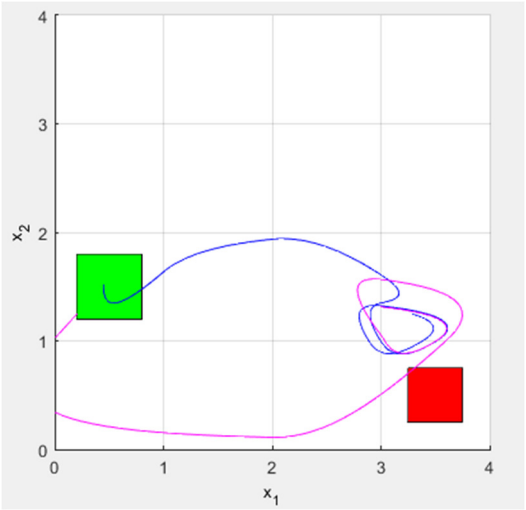
Therefore, we create the path coverage with the following specification

!<>(ploc1/\<>redAreaLoc4)';

Which specifies that “not eventually visit location 1 and then visit red area with location 4”.
This specification helped s-taliro to falsify at the second run:
…
`Best ==> <0,-0.0052011>`

```
FALSIFIED at sample ,75!
  Running time of run 1: 40.4002 sec
```

In the following figure the trajectory of first run is given by the blue line and the trajectory of second run is given by the pink line:



Demo 4: (Navigation Benchmark)

This demo considers the following hybrid automata in the simulation model

`navbench_hautomaton.m`

and it uses the original s-taliro for the falsification.

Testing:
To run demo the following script should be used:
`navbench_Yellow.m`

At first iteration it considers the following specification (not eventually visit yellow area):

'!<>yeloArea';

Which runs 100 tests and cannot falsify the specification.

Therefore, we created the path coverage with the following specification.

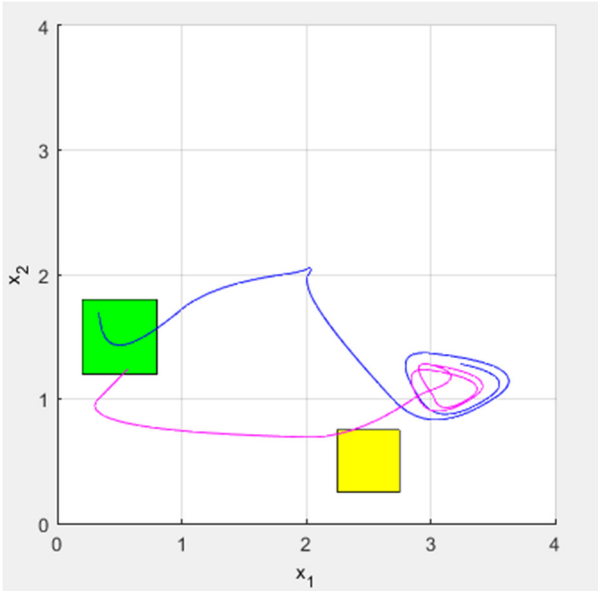
‘!<>(ploc1/\<>yeloArea)’;

Which specify that “not eventually visit location 1 and then visit yellow area”. This specification helped s-taliro to falsify at the second run:

…

```
Best ==> <0,-0.011422>
FALSIFIED at sample ,53!
  Running time of run 1: 82.0612 sec
```

In the following figure, the trajectory of first run is given by the blue line and the trajectory of second run is given by the pink line:



Demo 5: (Navigation Benchmark with Coverage Metrics)

This demo considers the following hybrid automata as the simulation model

`navbench_hautomaton.m`

but it uses the structural coverage with coverage metric for the falsification.

Testing:
To run demo the following script should be used:

`staliro_navbench_coverageTest.m`

At first iteration it considers the following specification (not eventually visit yellow area):

Phi='!<>yeloArea';

Which is considered as

```
phi_1 = !<>p64
```

After the first run of staliro we observe the visited locations and unvisited locations:

```
…
Iteration number is
     1
numOfUnVisited
     8
…
```

The variable `numOfUnVisited` specifies how many locations have not been visited so far. `numOfUnVisited` is 8 which means among 16 number of possible location, 8 locations have not been visited.

Then location predicate is chosen among the unvisited locations.

```
…

Location predicate is locPred1 and location number is 14
…
```

The location predicate is `locPred1` and its location value is 14.

Then new specification is created:

```
new phi is (!<>p64)\/!<>_[0,inf)locPred1
```

And the second run of staliro will continue and iteration of location observation and specification update continues similar to 1st iteration.

At the end of simulation, this demo creates the four following lists:
1- List of all visited location which is the history of all locations that are observed so far:

```
List of all visited locations
Columns 1 through 13
     9    10     6     7     8     4     8     7     3     4     8
7     3
  Columns 14 through 26
```

```
     4     9    10     9     5     9    10     9    10     6     7
8     7
....
```

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

```
List of location predicates that are tested:
     14
     2
     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 staliro. This list is sub set of previous list, however in this demo this list is empty which means the location predicates did not guide the system to go to the unvisited location:

```
List of location predicates that are visited:
     List is empty
```

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 that location. This list is a sub set of 2nd list:

```
List of location predicates that are not visited:
     14
      2
      1
```