
TorXakis Documentation

Release 0.3

TNO, Radboud University

Oct 31, 2017

Contents:

1	Tutorial	1
----------	-----------------	----------

TODO.

Trace and replay functionality

TorXakis offers the possibility of writing the trace of a test, simulator, or stepper run to a file. This trace can be used subsequently to replay a test, which is useful in case a given error needs to be debugged. We illustrate this by means of the [Adder](#) example.

Start the [Adder SUT](#) and the run TorXakis:

```
tester Adder SutConnection
test 10
```

This will produce an output similar to this:

```
TXS >> .....1: IN: Act { { ( Action, [ Plus(-7059,-2147474793) ] ) } }
TXS >> .....2: OUT: Act { { ( Result, [ -2147481852 ] ) } }
TXS >> .....3: IN: Act { { ( Action, [ Plus(2245,-2147477795) ] ) } }
TXS >> .....4: OUT: Act { { ( Result, [ -2147475550 ] ) } }
TXS >> .....5: IN: Act { { ( Action, [ Plus(-2662,-2147474703) ] ) } }
TXS >> .....6: OUT: Act { { ( Result, [ -2147477365 ] ) } }
TXS >> .....7: IN: Act { { ( Action, [ Minus(-2147477744,-2147481994) ] ) } }
TXS >> .....8: OUT: Act { { ( Result, [ 4250 ] ) } }
TXS >> .....9: IN: Act { { ( Action, [ Plus(-2147473923,-7450) ] ) } }
TXS >> .....10: OUT: Act { { ( Result, [ -2147481373 ] ) } }
TXS >> PASS
```

Which corresponds with the output observed at the SUT:

```
Starting 1 adders.
Starting an adder listening on port 7890
Adders on port 7890 received input: Plus(-7059,-2147474793)
-2147481852
```

```
Adders on port 7890 received input: Plus(2245,-2147477795)
-2147475550
Adders on port 7890 received input: Plus(-2662,-2147474703)
-2147477365
Adders on port 7890 received input: Minus(-2147477744,-2147481994)
4250
Adders on port 7890 received input: Plus(-2147473923,-7450)
-2147481373
```

Then save the trace and exit TorXakis:

```
trace purp $> AdderPurpose.txs
exit
```

The `trace` command will produce a TorXakis source file with the following contents:

```
PROCDEF replayProc [Action :: Operation; Result :: Int]() HIT
::=
Action ! Plus(-7059,-2147474793)
>-> Result ! -2147481852
>-> Action ! Plus(2245,-2147477795)
>-> Result ! -2147475550
>-> Action ! Plus(-2662,-2147474703)
>-> Result ! -2147477365
>-> Action ! Minus(-2147477744,-2147481994)
>-> Result ! 4250
>-> Action ! Plus(-2147473923,-7450)
>-> Result ! -2147481373
>-> HIT
ENDDEF
```

Here we see that the actions on the `replayProc` process correspond with the output we observed when running the tests.

Using the trace (in the form a process) generated by TorXakis, together with an [AdderReplay](#) purpose definition:

```
-- | Adder model that uses the trace generated as test purpose.
PURPDEF AdderReplay ::=
  CHAN IN   Action
  CHAN OUT  Result
  -- Process `replayProc` will be generated by running TorXakis with the
  -- `Adder` model and the SUT, as follows:
  --
  -- > tester Adder Sut
  -- > test 10
  -- > trace purp $> AdderPurpose.txs
  -- > exit
  --
  -- Therefore this file should be loaded together with the generated purpose
  -- above (`AdderPurpose.txs`).
  GOAL replayAdd ::= replayProc [ Action, Result ] ( )
ENDDEF
```

we can replay this test by restarting the SUT and executing the following commands in the TorXakis command-line:

```
tester Adder AdderReplay SutConnection
test 11
```

This will produce the following output:

```

TXS << test 11
TXS >> .....1: IN: Act { { ( Action, [ Plus(-7059,-2147474793) ] ) } }
TXS >> .....2: OUT: Act { { ( Result, [ -2147481852 ] ) } }
TXS >> .....3: IN: Act { { ( Action, [ Plus(2245,-2147477795) ] ) } }
TXS >> .....4: OUT: Act { { ( Result, [ -2147475550 ] ) } }
TXS >> .....5: IN: Act { { ( Action, [ Plus(-2662,-2147474703) ] ) } }
TXS >> .....6: OUT: Act { { ( Result, [ -2147477365 ] ) } }
TXS >> .....7: IN: Act { { ( Action, [ Minus(-2147477744,-2147481994) ] ) } }
TXS >> .....8: OUT: Act { { ( Result, [ 4250 ] ) } }
TXS >> .....9: IN: Act { { ( Action, [ Plus(-2147473923,-7450) ] ) } }
TXS >> ....10: OUT: Act { { ( Result, [ -2147481373 ] ) } }
TXS >> ....11: OUT: No Output (Quiescence)
TXS >> Goal replayAdd: Hit
TXS >> PASS

```

Since we ran the `Adder` model with an `AdderReplay` purpose the possible actions of the model are constrained by the latter, allowing us to replay the behavior observed when running the tests. Note `TorXakis` still does one extra check, so we specified one extra step (11 instead of 10) to account for this check.