DES Supervisor Application for BSCOPNBMAX and MPO

Version 1.2 08/20/2015

GENERAL USAGE NOTES

This program contains implementations of both BSCOPNBMAX and MPO, as well as a converter utility, defined as follows:

-BSCOPNBMAX (Basic Supervisory Control and Observation Problem: Non-blocking and Maximally Permissive Case) is a previously open problem of synthesizing non-blocking, deterministic, information-state property compliant, maximally-permissive supervisors for a partially observed Discrete Event System (DES). The problem—as well as an algorithmic solution for BSCOPNBMAX—are described in Xiang Yin and Stéphane Lafortune's paper "Synthesis of Maximally Permissive Supervisors for Partially-Observed Discrete-Event Systems." This program is an implementation of the algorithms found in that paper.

-MPO (Most Permissive Observer) is a Bipartite Dynamic Observer used to solve Dynamic Sensor Activation Problems for partially observed DES. The structure is utilized in Xiang Yin and Stéphane Lafortune's paper "A General Approach for Solving Dynamic Sensor Activation Problems for a Class of Properties" to synthesize either a maximal or minimal sensor activation policy for an arbitrary information state property.

-CONVERT will convert FSM files between the format listed below and the UMDES .fsm file format, which may be useful for viewing structures in DESUMA

Compiling DES_Supervisor:

Within this directory in the command terminal, run the command "make".

NOTE: this program uses features of C++ unique to C++-11 and later.

Running DES_Supervisor:

Command line execution of BSCOPNBMAX is in the form

./bin/DES_Supervisor <options…> (user-compiled)

./bin/DES_Supervisor_win <options...> (pre-compiled binary for Windows)

./bin/DES_Supervisor_linux <options…> (pre-compiled binary for Linux)

Options List:

Mode [-m] - switch between interactive mode [INTERACTIVE], the BSCOPNBMAX [no arg], the MPO [MPO] or the converter [convert]

MPO_condition [-c] - request the MPO to find a [min]imal or [max]imal solution

FSM_file [-f] - provide an FSM file for processing

Property [-p] - provide an implemented information state property

ISP_file [-i] - provide a corresponding file for the specified ISP property

Verbose [-v] - request more detailed output

Write_to_File [-w] - write all structures relevant to current mode to separate .fsm files

in the ./results folder

Help [-h] - display help menu

Examples:

Start interactive mode

./bin/DES_Supervisor -m interactive

Compute supervisor for FSM_test_4.txt w/ safety property and no output

./bin/DES_Supervisor -f ./test/FSM_test_4.txt -p safety -i ./test/safety_test_4.txt

Compute supervisor for FSM_test_6.txt w/ no property and both screen and file output

./bin/DES_Supervisor -f ./test/FSM_test_6.txt -v -w

Compute MPO for FSM_test_2.txt

./bin/DES_Supervisor -f ./test/FSM_test_2.txt -m MPO

Compute maximal MPO for FSM_test_25.fsm w/ state disambiguation property and full output

./bin/DES_Supervisor -f ./test/FSM_test_25.txt -v -w -p disambiguation -
i ./test/disambuguation_test_25.txt -m mpo

Convert a .txt FSM file to a .fsm FSM file

./bin/DES_Supervisor -f ./test/FSM_test_4.txt -m CONVERT

NOTE: All input files should be in Unix format. If unexpected results occur, try running dos2unix on the input files.

FSM_file:

The FSM_file describes the Finite State Machine that will be used to construct the supervisor. Its format is as follows:

<number_of_states> <number_of_events>

<state1> <marked (m or u)>

...

<staten> <marked (m or u)>

<event1> <controllability (c or u)> <observability (o or u)>

...

<eventn> <controllability (c or u)> <observability (o or u)>

<event state> <event state> ... //state1 transitions

...

<event state> <event state> ... //staten transitions

See the FSM_test_*.txt files in the ./test folder for examples.

ISP:

The Information State Properties currently defined for this program are safety, opacity, and state disambiguation. See "Adding ISPs" for information on how to add additional ISPs. If this is left blank or does not match any ISP, the supervisor will be constructed with a trivial ISP.

ISP_file:

The ISP_file contains additional information specific to the ISP being used in the program. If this is left blank or does not match any file, the supervisor will be constructed with a trivial ISP.

For safety, this file contains a list of the unsafe states.

For opacity, this file contains a list of secret states.

For state disambiguation, this file contains two lines of disjoint states

See ./test/<ISP>_test_*.txt for examples of valid ISP_files.

NOTE: The test numbers of the FSM and ISP files in ./test correspond to one another.

Adding ISPs:

ISPs are hardcoded in files ./src/IS_Property.cpp and ./include/IS_Property.h as derived classes of class IS_Property. You may use the existing properties as a template to construct your own, or contact the programmer.

NOTE: The function get_ISP in the mentioned files must also be modified in order to accept new command line arguments for the new ISP.

Contact Information

Max Morrison

Undergraduate Researcher Summer 2015

Univeristy of Michigan

maxrmorrison@gmail.com