COMPSYS-705

Assignment 1

Bisimulation Implementation and Research

Due: 22 September 2014, by 5pm

Submit in Dropbox called COMPSYS 705 Assignment 1

1 Bisimulation Implementation

1.1 Requirements

You are required to implement LTS bisimulation covered in class using the Java programming language. Given two LTS P and Q, your program must be able to find if they are bisimilar.

You must adhere strictly to the given input and output formats as assignments will be auto-marked to test successful completion.

This implementation is worth **15** % of your total marks for this course. You will also submit some take home exercises worth **10** % and a verification assignment using UPPAAL worth **10** %.

1.2 Inputs

The program should ask for two files from the user that contain the information for processes P and Q. Each line of an input file has a fixed format:

state(int),action(String):successor_state(int)

A sample file that represents a LTS with 3 states (1,2,3), an action set having 2 elements (a,b) and 3 transitions ((1,a,2),(1,b,3),(2,a,3)) is given as follows:

```
1,a:2
1,b:3
2,a:3
!
```

Note: The "!" represents the end-of-file. Any text after this line must be ignored. The source state stated in the first transition given in the input file is to be assumed the start state of the process.

1.3 Output

The output is also in form of a file (in the current working directory) which must have the following format:

```
Process P
S = state(int),state(int)
A = action(String),action(String)
T = (state(int),action(String),state(int)),(state(int),action(String),state(int)
Process Q
S = state(int),state(int)
A = action(String),action(String)
T = (state(int),action(String),state(int)),(state(int),action(String),state(int)
Bisimulation Results
state(int),—
state(int),— —
—
—
state(int),—
Bisimulation Answer
Yes/No
```

S, A and T represent the LTS sets S (the set of states), A (set of actions) and $\rightarrow$ (set of transitions). The Bisimulation Results section details the refined sets of states obtained after bisimulation equivalence algorithm is executed on the given LTSs. The Bisimulation Answer section contains either a Yes (if $P \sim Q$) or No (if they are not bisimilar).

The output of your program must be stored in a user-provided file-name.

1.4 Assessment

The marks allotted (out of 15). We will perform an automatic marking to check the functional correctness of your code. We will also look at how readable is your code.

1.5 Submission Details

A dropbox for code and pdf submission (of the take-home exercises) called *COMSYS 705 Assignment 1* has been setup. Submit the code (in java) before the deadline in this dropbox.

1.6 Additional Requirements

The submitted code must also meet the following specifications:

- The main public class must be named BisimulationChecker (contained in the default package (no package declaration)) and must have the following methods:

    1. *public BisimulationChecker()* : the constructor is used to initialize any internal variables.
    2. public void readInput(String fileP, String fileQ): This method reads the contents of the files pointed to by the Strings fileP and fileQ. If any argument is null (or a file with that name does not exist), the user is (repeatedly) asked to enter a filename until a valid file is found.
    3. *public void performBisimulation()*: performs partition-based LTS bisimulation. This method must check if valid input files (using the readInput() method) have been read by the BisimulationChecker object. If files have not yet been read, display an error message on the console.
    4. *public void writeOutput(String filename)*: Saves the results of the bisimulation by overwriting the contents of the file pointed to by the String filename. If the filename is null then user is asked to enter a filename on the console.

2 Additional Problems

Some additional problems which will be worth 15% will be relased before 1st September. Some of these problems will be based on the lecture material and also based on the Argos paper. Solutions to these problems will have to be submitted also by this deadline.

3 Conclusions

This assignment consists of three parts: Implementation of Bisimulation checker (15%), and take-home exercises (15%) and a model checking assignment in UPPAAL (10%). All components are to be submitted by the deadline of 23 September. This is an individual assignment and no group work or collaboration is allowed. We will be strictly checking for any copying / code duplication issues.