Software Reliability Methods

Assignment 3

Verifast and Model Checking

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Mestrado em Engenharia Informática

2019/2020

VeriFast

Consider a class describing a set of objects. We are interested in the following methods.
Constructor that builds an empty set.

boolean isEmpty () that checks whether the set is empty.

int size () that returns the number of elements in the set.

void clear () that removes all elements from the set.

boolean contains (Object e) that checks whether a given element is contained in the set.

void add (Object e) to add an element to the set.

void remove (Object e) to remove an element from the set.

Use reference equality (==) to compare set elements. Select an appropriate representation for your class. Write an abstraction **predicate** to specify the observable behaviour of a set without exposing its internal representation. The predicate should incorporate the fact that sets do not have duplicated elements. Add pre and postconditions to all your methods based on the abstraction predicate. Make sure your code compiles with the Java compiler (as well as it passes the VeriFast verifier).
Suggestion: Work at one method at a time, by gradually adding contracts to your methods. Leave method remove to the end.

Model Checking

A printing system in a department is composed of one or more printing devices (printers) and one or more printing clients.
Printers have no identity in the system, so that clients do not name the printer they are interest at. Instead they just issue a printing request to the printing system. The first printer available answers with is name (its process identifier), so that human clients may pick printouts from the appropriate printer.
Only idle printers can answer to print requests. Due to network restrictions, documents are sent to printers page by page. When issuing a print request clients provide the number of pages they intend to print. If n is the number of pages in a print request, then the printer must be available to receive n pages *from the same client*, before becoming idle again.
All interactions in the printing system is by *message passing* (no shared memory is expected). All printers read from a common shared asynchronous channel. Here's the properties we are interest in:

No page mix-up Printers are not expect to mix-up pages from different documents, so that human clients do not need to collect different pages of the same document from different printers.

Mutual exclusion Conversely, one cannot find two clients sending pages to the same printer at the same time.

Absence of starvation Clients that plan to print a document (that is, that try to issue a print request) are expected to be eventually served, under a weak fairness regime.

No deadlock Clients are suppose to print their documents to the end; servers are expect to serve clients indefinitely.

Your expected to

1. Devise a protocol that ensures the four properties above, and model it in Promela.

2. Check the four properties of interest.

Here's a few point to be taken into consideration:

- Each client prints a single document. The client `proctype` must be parameterised on the length of the document to print.

- Each printer serves an unbounded number of clients.

- Your system must modular so that one can easily change a) the capacity of the printing system channel, b) the number of clients, and c) the number of printers. Clearly explain, in a paragraph in a comment, how to change the number of clients and the number of printers.
- Start with a model composed of two printers and three clients with documents of length 5, 7, and 3. The printing channel must have at least capacity 3.
- Use whichever means you find more convenient to check each of the four properties.
- Explain how to check the properties using Spin from a *Unix command line*. For each one of them add a paragraph within a comment explaining how to proceed in checking the property. Do not forget to explain how shall one conclude that the property holds or does not hold.
- I expect *five clearly separated block comments*: one explaining how to vary the capacity of the channel and the number of printers and clients; the others explaining how to check each property from the command line.
- Clearly distinguish model from ghost code by marking the latter with an appropriate comment (`ghost`).
- Document your source code as you would do for a Java program: an header for the file, describing its contents and author, and an header for each member (global variable or channel, **`proctype`**, **`#define`**, and so forth).

Delivering your assignment

- Deadline: January 8th, 2020.
- Your assignment is composed of *two* source files: one VeriFast (extension `java`) and one Promela (extension `pml`). Zip your code and upload it at the course's web site before the deadline.