T-79.5303 Safety Critical Systems

Case Study 5: ProB - Model Checker for B language

Teemu Tynjälä

April 10, 2008

Automated verification of B systems - ProB

ProB is an academic tool for analyzing B systems via Model Checking. In ProB, we can check for deadlocks, invariant violations and operation coverage.

You can find and install ProB from the address `www.ecs.soton.ac.uk/~mal/systems/prob.html`

ProB can be installed for Windows, Linux, Solaris, MacOS – very rare for an academic verification tool. A Java version is in the works which will expand the supported platforms even further.

So, go to the web page and install it along with AT&T GraphViz..

ProB - Basics

So, by this time you should have installed ProB in your machine... Now is the time for a few hints

- You know the mathematical syntax for B. To get the ProB equivalents, choose the **About** menu, and there the item **Summary of B syntax**

- The machines are stored in the **Machines** subdirectory. You should save your work here, to help Dotty find your graph files.

- Look in the **Machines/Book** directory to find some of the machines that we have discussed in the lectures already... A great help to learn ProB syntax as well..

ProB workflow - 1

1. Make the first model in e.g. Notepad, and save it as foo.mch (the suffix is important)
2. Open your file in ProB. If there are syntax errors, ProB will point them out to you, and you can correct them on the spot and save/reopen.
3. Start analysis. Choose the **Animate** menu, and there the **Random Animation**. You can choose the number of steps for random walk.
4. After the random walk is done, you can choose **View visited states** in the **Animate** menu to get a picture of the partial 'reachability graph'
5. If the invariant is violated, the random animation will stop, and you can choose **View -> Subgraph for invariant violation** in **Animate** menu to see how the bad state was reached.
6. To do another random simulation choose **Animate -> Reset** first.

ProB workflow - 2

Sometimes invariant violations are not the only things we are interested in. Deadlocks are states of the system where no operation is enabled. To find such things we have to do model checking as follows:

1. Choose **Verify -> Temporal Model Check**

2. Choose the properties you are interested in. Usually invariant violations and deadlocks suffice.

3. In Temporal Model-Check meny there is a choice to use **breadth-first search** or not. From my experience, easy errors are found quicker using breadth-first search (especially at beginning of system design). When the ’easy’ errors are plucked out, the breadth-first search option could be left unchecked.

ProB workflow - 3

After having done random animation or temporal model check, we may be interested to see what's our coverage of the system's complete state space. To this end we have the **Analyse** menu in the toolbar.

In that menu, **Compute Coverage** shows statistics for completely explored states, partially explored states, etc. as well as the number of times each operation was called during the random walk. The less partially explored states there are, the more convinced we can be of the system's correctness.

ProB interactively..

Sometimes we are interested in driving our system step by step rather than performing model checking or random animation.

ProB helps in this as well, because the window at the bottom of the screen entitled **EnabledOperations** shows you the operations that are enabled at any system state. The user can just double click a suitable operation, and the B machine is advanced accordingly.

Interactive operation of a B machine is a good starting point for verification. It's easy to check the typical success scenario there first before venturing into model checking for heavy-duty error-digging.

The way is open for ProB experimentation...

That’s it – this was to raise your attention and interest in ProB tool.. Now you are equipped to start using ProB and testing its use.

Read over the example machines first and practice the syntax of ProB. After that you may start doing more complex designs and doing verification on those... The sky is the limit..