Multicore Computing – CS 5966 / 6966 – Week 4
http://www.eng.utah.edu/~cs5966

Contents

1 Overview of this week

This week, we will learn thread programming, how debugging thread programs using testing tools can miss bugs, and how formal verification tools can help locate these bugs with assurance.

Formal verification tools are also subject to exponential blow-ups, although they often try to avoid wasted searches. They give you measurable coverage according to precise coverage criteria. However, you must use formal verification tools with care – else they will also explode. In particular,

much like with testing, you must downscale your problems before you attempt to formally verify.

By "downscale" we mean: dial down the values of the `#define` constants, and also make suitable adjustments to the overall code. Without this, *all* your formal verification attempts will bite the dust! With some downscaling, formal verification tools can be your best friends!

How much downscaling is needed? You should discover that using your engineering acumen!! There is no free lunch. We won't automatically find your bugs! We only **help** you find them.

2 Assignment

Assignment 3 (due 2/10): Please DO NOT post your solutions WHERE INDICATED. Instead, **FOR THOSE PROBLEMS**, please keep a webpage containing your solutions in PDF form, and please email me the web URL. For all other questions, do post the solutions / discussions online! Those are very valuable and all of us get to learn from each other.

5%: Try to `gcc -lpthread -lssl file.c` on the

Code examples courtesy of LLNL, Blaise Barney

`watch-count-small-buggy.c, watch-count-small.c, bug1-small.c, bug1fix-small.c`

Run the resulting `a.out`. Did you encounter any bugs?

10%: Key for discussions: bug1ix-small

Does `bug1fix-small.c` really fix the bug? Use `Inspect` to find out. Also run `Inspect` on the other examples. (Also demo-ed in class.)

5%: Key for discussions: prodcons under gcc

Try to `gcc -lpthread -lssl prodcons.c` and run the `a.out`. Can you find any bugs?

30%: Please do not post discussions on this question. Put it on the PDF on your URL, instead.

Formally verify the `prodcons.c` code using Inspect. Exactly describe the bug.

20%: Please do not post discussions on this question. Put it on the PDF on your URL, instead.

Fix the bug in `prodcons.c` and re-verify using Inspect.

10%: Key for discussions: Boehm – Threads as a Library

Read and summarize (in a page) "Threads Cannot be Implemented as a Library" by Boehm

`http://www.eng.utah.edu/~cs5966/LECTURES/Week4/boehm-threads-as-a-library.pdf`

10%: Key for discussions: Boehm – Reordering Pthread-Style Locks

Read and summarize (in a page) "Reordering Constraints for Pthread-Style Locks" by Boehm

`http://www.eng.utah.edu/~cs5966/LECTURES/Week4/boehm-reordering-pthread-style-locks.pdf`

5%: Please do not post! Put it on URL.

Code-up the Sieve program in Boehm's Threads as a Library paper (see above) using Pthreads/C and show that indeed you are able to get some speed-up on our machines.

5%: Please do not post! Put it on URL.

Redo the above part using Cilk (must be easier to do time measurements using Cilk).