

CS 6110 Software Correctness, Spring 2022

Lec9

Ganesh Gopalakrishnan
School of Computing
University of Utah
Salt Lake City, UT 84112

URL: bit.ly/cs6110s22



Slides for Lec9 : Agenda

- Roadmap
 - Where are we headed?
 - Where all will we park – or breeze by?
- Model-Checking Overview
- Model-Checking in Murphi
- Anyone who wants to peek at what I did last year ?
 - bit.ly/CS6110S21
 - I will largely follow that overall approach (except we are not rushing as much)

Roadmap of CS 6110 now on

- Basics

- So far we got a refresher on basic Mathematical Logic and learned some LTL
- We then looked at some half-finished protocols
 - We did not question how they were designed
 - “Seat of pants” which does not surprise us
 - Think of it as a “game” of sorts
 - Games have rules and games can be model-checked.... So why not?
 - Others may say (in more advanced practices) that
 - We have an invariant in mind
 - We don’t shy away from breaking it (when messages are in flight)
 - We then restore them (when things settle)
- It is possible to derive these protocols “top-down” [good project material]
 - E.g. specify it as a “more atomic” activity (steps “snap together” without gaps)
 - Then coarsen these atomic steps into disjoint steps
 - To gain speed (e.g. in a pipelined processor)
 - That is how “reality works” (things are truly distributed and asynchronous, often)

Roadmap of CS 6110 now on

- Examples of atomic -> multi-step
 - Pipelined processors
 - Can talk about how to prove pipelining to be correct
 - Cache coherence
 - Can even synthesize correctly
 - In 2002 we did something similar
 - <https://dl.acm.org/doi/abs/10.1023/A%3A1012916831123>
a copy can be obtained from here, as Kluwer is gone! But FMSD has revived.
 - <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.621&rep=rep1&type=pdf>
 - Latest work in this genre
 - <https://homepages.inf.ed.ac.uk/vnagaraj/papers/isca20.pdf>
 - They don't cite us (because our area is so busy publishing and not searching)
 - But the senior authors are friends and they acknowledge the omission + will cite in future!
 - Other protocols? (e.g. computer networking)

Roadmap of CS 6110 now on

- We can even do parametric proofs
 - Model 2 proctypes explicitly
 - Model the remaining (N-2) proctypes as one “ghost” proctype
 - Refine the ghost till it faithfully models the N-2 ghosts
 - Voila! We now have a proof for N proctypes
 - Good work in this area [nice project topic ; done in Murphi fully !!!]
 - <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.5271>
 - Later the lead author (Dr. Ching-Tsun Chou) helped us do something like this
 - <https://ieeexplore.ieee.org/document/4392796>
 - which appeared in an extended form also
 - <https://dblp.org/rec/journals/fmsd/ChenYGC10.html?view=bibtex>
- Some of the above posted on the class website (fmcad04 tutorial)

Today (2/8/22)

- Make you engage with the Promela locking protocol
 - We will collectively do the exploration of `locking-buggy.pml`
 - Hopefully you'll then be able to fix the protocol (Asg3) with assurance
- Show you how it is coded in Murphi, and some fun facts
 - Most of these facts will come up during the lecture
 - I have some slides to follow

What is coming?

- Thu 2/10/22 on (for about 3 lectures)
 - Encourage you to read the next Bradley/Manna Chapter
 - Ask you to encode Kenken in Z3py
 - those who have done this will get another assignment
 - Then, Alloy, and Model-Finding
 - Will teach first-order logic using Alloy
 - Will review relations, preorders, etc, using Alloy
 - Then we will write some Alloy models
- Then
 - Verification using Dynamic Symbolic Execution (KLEE)
 - Hoare-logic proofs (e.g. using VeriFast)
 - Static Analysis

Now today's material

- Make you interact on a Promela scenario
 - Hopefully you can finish Asg-3
- Then onto reading a Murphi model
 - Next slide carries intuitions about Murphi ...

Why Murphi? What else is there?

- Murphi - not Murphy. It evolved as Mur\$\phi\$
 - but now everyone wants to avoid Greek-letters (that put off real designers!)
- In the genre of
 - TLA+ - free TLA+ book + tons of examples
 - <https://lamport.azurewebsites.net/tla/book.html>
 - Also see PlusCal <https://lamport.azurewebsites.net/pubs/pluscal.pdf>
- In the style of Unity
 - <https://www.cs.utexas.edu/users/misra/psp.dir/Marktoberdorf-88.pdf>
- I encourage you to “think TLA+/Unity/PlusCal”
 - But you might code in Murphi finally!!
- Rumur’s site has a Python-based “toy model checker”
 - Shows you how Murphi works
- Also see
 - <http://mclab.di.uniroma1.it/publications/papers/papers/Melatti2006.pdf>
 - Let’s read its pseudo-code – Figure 1 – to know what Murphi does
- SPIN does a whole lot more (nested DFS, LTL, ...)
 - Murphi does only Safety
 - Rumur has some limited liveness (I’ve not investigated what subset)
- Real-world
 - Liveness is nice
 - But when things get tough, dumb-down your expectations !!! ☺ (told by Shaz Qadeer to me..)