Ning

Models and Verification

19 September 2019

Class Summary

[OPENING]: This class is a theoretical, mathematical, based class on formal descriptions of computer science models.

[GRADING]: There is an oral exam (counting for two-thirds of your grade) and six (6) or seven (7) TPs (counting for one-third of your grade).

[SUMMARY]: Think about software development like a bridge. We should draw a blueprint, draw calculations, and verify *before* we go into production. Modeling vs Design: modeling is independent of the functions (modeling is what rather than how). Verification vs Validation: verification asks if we are producing the product correctly (quality), rather than producing the right product (business-driven question). In modeling, we have functional vs non-functional parts. Functional being service related, and non-functional being non-service aspects (such as security, or performance). The difficult part of verification is due to complexity issues (ie: chess and go board states are too big to confirm all the states. There are not enough atoms in the universe to simulate a 1-to-1.) Managing complexity is key. You can check systems to prevent reaching “bad states” (deadlocks). Try to achieve Liveness: progress in system (continues to run). Fairness assumption: splitting time to processes, no process is starved of CPU time. We care about automated verification that systems did its job, as long as we can achieve that, we call that correctness.

[SUMMARY2]: Observer: an operation that observes what the structure does. Modifier: modifies a structure. Generator: generates a structure. These are the three categories we use to describe operations. S-sorted sets are just sets that we partition based on different type names (ie: sets containing different operations). Signature is a couple of an s-sorted set and a function (ie: F bool, bool : meaning a function that takes a bool and outputs a bool). Can we write expressions with our Types? Expressions are applications of functions on other functions and so on (ie: push(3, push(2, empty))). Terms of a signature (there’s a provided formal math definition, but it’s pretty much a fancy way of saying we’re applying some signatures in a proper way). Axioms are the “contracts” that apply to the terms (ie: they’re statements that should hold if your terms are correct. I’m guessing these become useful later in verification). There are lots of axiom practices we did in class that solidify s-sorted sets, signatures, terms, and axioms all-together holistically.