18-642 Recitation
Week #10

November 1, 2019
Updates

• Homework:
  – Last week homeworks graded on canvas
  – Next week homework due Wednesday night
    • Project workload has been INCREASING
    • Homework workload is DECREASING (only three HWs left)
  • Homework grading
    – Points are mostly for effort
    – Read comments on canvas, even if you got full points
    – Full points does not mean you got the right answer
      • We’ll try to cover some common issues in recitation
      • If you’re not sure – ask!
Updates

• Projects:
  – Project 7 graded on canvas
    • If you handed in late, it might take a while longer
  – Project 8 due tonight
  – Project 9 is live

• Don’t forget the weekly survey!
Today

• Project 8 questions?
• Project 9

• Be sure to fill out weekly surveys!
Project #8 Questions?

- Peer reviews on code
- Unit Testing
- Check, but do not fix warnings
- Have a look at how your maze solving is going
Project 9

• Fix compiler warnings
• Fix any failing unit tests
• Peer review of unit tests
• First step for monitoring/invariants

• Preview of Project 10:
  – More monitoring/invariants
Compiler Warning Flags

• Same as in Project 8

```c
# Warning flags for Projects 8,9,10
target_compile_options(ece642rtle_student PUBLIC
  "-Werror" # Do not comment this out after Project 8!
  "-Wextra"
  "-Wall"
  "-Wfloat-equal"
  "-Wconversion"
  "-Wparentheses"
  "-pedantic"
  "-Wunused-parameter"
  "-Wunused-variable"
  "-Wunused-type"
  "-Wunused-function"
  "-Wunused-decls"
  "-Wunused-type"
  "-Wunused-decls"
  "-Wunused-value"
  "-Wswitch-default"
  "-Wunused-init-self"
)}
```
Compilation Notes

• Remember: Makefiles only rebuild source files if they have changed (or if they haven’t successfully compiled before)
• If you don’t change any files, warnings won’t appear on second compile
• Our solution: keep the –Werror flag
  – Turns all warnings into errors
  – Compilation won’t succeed until all warnings are removed
• If you disable the –Werror flag, remember to re-enable it and check build for warnings
• Turn in Project 9 code with all warning flags, especially –Werror, enabled
  – Make sure you compile it with –Werror before turning it in
Unit Tests

• In Project 8, you wrote unit tests to achieve 100% transition, 100% branch, and sufficient data coverage
• For Project 9, you must maintain this coverage and have your tests pass
Unit Test Peer Review

- Groups assigned on canvas
- Same organization/etiquette as previous projects
- Review testing coverage and traceability
  - Have your state chart printed or on a tablet screen
- Create a unit test peer review checklist as before
  - For each transition: check that the test exists and has a commented annotation in code, and that the test is correct
  - Check testing code style, but test coverage is the main emphasis
  - Check whether pass/fail criteria are correct
Runtime monitors

- In general: some sort of external entity that plugs into a system’s interface to monitor system output and state
- For ece642rtle: a separate ROS node that listens to messages sent by the turtle system
- You are given an interface for handling message interrupts
- By paying attention to the sequence of messages, can infer state and detect improper behavior
Logging Monitor

- Example provided to you to show you how message interrupt handling works
- Just prints every message it sees
- Doesn’t record state or check invariants
Invariants

• A condition that is always true during proper operation
• A monitor can check for invariant violations by keeping track of the state of the system
Step Monitor

• Invariant: turtle shall not move more than one square at a time

• Example provided to you to show how monitor can log state to check invariants

• Uses ROS_WARN to indicate an invariant violation
Turn monitor

• Invariant: turtle shall not turn more than 90 degrees at a time

• You must write this invariant

• Project page gives some hints

• Run it with your turtle
  – If invariant is violated: is it because of your turtle code, or a bug in the monitor?
  – You do not have to fix invariant violations for Project 9
Project 9 summary

• Fix warnings
• Fix failing unit tests
• Peer review unit tests (not the turtle code itself)
• Invariant monitor for 90 degree turn limit
• Lots of choices to manage your time:
  – Unit tests compile independent of ROS project, so you can fix warnings first OR fix unit tests first
  – Write monitor at any time
• Make sure what you turn in compiles successfully and that the documentation is up-to-date
Looking ahead to Project 10

• Solve several new mazes (acceptance tests)
• Update documentation/traceability/unit tests
• Write more invariants
• Peer reviews of the invariant monitors
Build Packages

• Projects require a gzip tar file
  – All the files needed to be added to a blank course virtual machine image
  – Scripts to compile & run unit tests
  – Script to run acceptance test on one maze
    • Maze file put in as a script parameter

• If you have trouble ask at office hours
  – Very basic scripting, but might be new to you
Project Questions?
Homework Questions on Parallel/Serial Reliability?

27-1a. (1 point) What is the reliability of the aircraft in terms of probability of both engines failing during a 12 hour trans-Pacific flight? Put your final answer in terms of 1 failure out of X flights. (Show your work)

27-1b. (1 point) What is the chance that the aircraft loses only a SINGLE engine during a 12-hour flight? (1 failure out of X flights)

27-1c. (1 point) Assume that after a single engine failure the aircraft diverts to the nearest airport that is 3 hours away. What is the probability that this one engine will still be working when the aircraft lands 3 hours later? (1 failure out of X such 3-hour flights)

27-1d. (1 point) Briefly explain the concept of ETOPS-180 operation for aircraft. (For this particular question Wikipedia is an acceptable source.) Pasting in a picture that illustrates the concept.

27-1e. (1 point) Given the previous answers, what is the probability of catastrophic aircraft loss (crashing because of loss of both engines) if ETOPS-180 rules are followed? (1 failure out of X flights)

27-1f. (2 points) Is this engine reliable enough to meet the 1 catastrophic failures per billion flight hours with ETOPS-180 flight rules assuming that all missions are 12 hour flights?