18-642 Recitation #4

February 9, 2018
Updates

• Homework:
  – Homeworks #5/6/7/8 graded on canvas
  – Homeworks #10/11/12 due Sunday night @ 11pm

• 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
Updates

• Projects:
  – Project 3 graded on canvas
  – Project 4 due tonight @ 11:00 PM
    • Remember that code must comply with every Project 3 Checklist item
  – Project 5 released, due 2/16 @ 11:00 PM
  – Project 6 will be a two-week project
Updates

• Be sure to fill out question sheet to receive attendance points today!
• Exam #1 in less than a month (Wednesday 3/7 in class)
Today

• Project 5
• Homework 5/6/7/8 review
• Homework discussion
Project #4 Questions?
Project 5

• Keep track of how many times the turtle has visited a cell
• Split code into turtle and maze components
• Must build and solve m1.maze
• Try running on new maze: m2.maze
  – You do not have to solve this maze
Cell Visit Counts

- You are given a new parameter for displayTurtle: displayTurtle(int nw_or, int visits);
- Call this function before you return in your main routine
- Function is for debugging/visualization only – accuracy depends on how well you are keeping track of data locally
student_maze and student_turtle

- **student_maze.cpp**
  - Does not decide how turtle moves
  - Translates moves from student_turtle into absolute coordinates
  - Think of this like building a simulator for the turtle

- **student_turtle.cpp**
  - Does not know its absolute coordinates
  - **cannot ask student_maze where it is or absolute orientation**
    - or call `bump(x1,y1,x2,y2)`
    - `student_maze` should tell `student_turtle` if a move resulted in a bump
  - Decides how to move based on current state and whether it has bumped
  - This is how a real robot (or human dropped into a maze) would behave
Example with 3x3 Maze

- Turtle is dropped into maze
  - Doesn’t know where or which cardinal direction it’s facing
  - Will draw its own map of the maze (2D array) – like a human carrying graph paper around
Drawing its own maze

- Turtle doesn’t know where it is: could be anywhere in a 3x3 maze
- Knows it can move at most two squares in any direction from start point
- Therefore needs only a 5x5 array to keep track of any moves it makes (assuming it starts in the center of its array)
- Doesn’t know which direction it faces
  - Can assume north

Real maze could be any of these (in rotation) or more!
Translating Actions: Start

Ground Truth
(student_maze)
(x,y,orient) = (1,0,east)

Turtle’s local model
(student_turtle)
(x’,y’,orient’) = (2,2,north)
Translating Actions: turn right

- Student_turtle decides to turn right
- Updates its local orientation or’
- Tells student_maze it turned left, which updates absolute orientation or

Ground Truth
(student_maze)
(x, y, or) = (1, 0, south)

Turtle’s local model
(student_turtle)
(x’, y’, or’) = (2, 2, east)
Translating Actions: check bump

- Student_turtle wants to check for bump
- Student_maze sends whether there’s a wall in front of turtle based on x,y,or

\[
\begin{align*}
\text{Turtle’s local model} & \quad (x’,y’,or’) = (2,2,\text{east}) \\
\text{Ground Truth} & \quad (x,y,or) = (1,0,\text{south}) \\
\end{align*}
\]

Bump = false

Cool, I can move forward
Translating Actions: Move Straight

- Student_turtle decides to move straight
- Updates its local coordinates \( x',y' \) and local array of visits counts
- Tells student_maze it moved straight, which updates absolute coordinates \( x,y \) and visits count at \( x,y \)
Review:

• student_maze calls decision function in student_turtle
  – Hint: pass return value of bumped() to turtle using this function
• student_turtle decides how to move based on internal logic
• student_turtle updates internal array of cell visit counts
• student_turtle returns an action to student_maze
• student_maze translates to absolute coordinates, calls displayTurtle(), etc
Sequence Diagrams Preview

- You will see this in class this week
- Describes a scenario (interaction between components)
- Time flows top to bottom
- Boxes indicate components
- Arrows indicate messages/calls from one component to another
- Use them to describe interaction between components in Proj 5 writeup
Project 5 Questions?

• Review:
  – Keep track of the times a cell has been visited
  – Split into maze and turtle
  – Run on new maze (but don’t have to solve)

• Builds a foundation for Project 6
  – Will introduce more mazes not solvable by LHR/RHR
Homework Review

• SCC vs MCC
• Equivalence Classes
• MCDC
• State Charts
MCC and SCC

• McCabe’s cyclomatic complexity
  – Counts # of if/while/for conditionals in the code

• Strict cyclomatic complexity
  – Includes +1 for every condition within a branch

• \( \text{if} \ (a < 0 \ \&\& \ b > 0) \) adds +1 to MCC, +2 to SCC
MCC/SCC in student.cpp

• Check whether your tool computes MCC or SCC

• Find branch statements with multiple conditions

```cpp
mod = true;
if(z == true && aend == false) {
    if (nw_or == 1) pos_.setY(pos_.y() - 1);
}
```

• Add # of extra statements to MCC to get SCC
  – Or subtract from SCC to get MCC

• For student.cpp, SCC = MCC + 1
Equivalence Classes

• Any input in an equivalence class is expected to cause the code to behave the same
  – If input a and input b are in the same equivalence class, they will both take the same branches in the code
• Always remember special case values:
  – NULL for pointers
  – NaN for floats
  – Overflow values (INT_MIN, INT_MAX)
To achieve MC/DC coverage

if((a > 5) && ((b < 17) || (b > 97) || (b == 42)))

• Inputs to make entire decision true and false
• Inputs to make each conditional true and false
  – a > 5, b < 17, b > 97, b == 42
• Each condition affects decision independently
  – Hold all other conditionals fixed at some value
  – Changing Boolean value of the condition changes the outcome of the decision
  – Ex: b < 17 affects decision?
    • Hold a = 6 (making a > TRUE), b < 42 (making b > 97 FALSE, b == 42 FALSE)
    • If b < 17, decision is TRUE, else decision is FALSE
State Charts

- Used to express stateful behavior
- Number the states
- Name the states
- Input to system causes state transition
- Each state sets all output variables
- Avoid complex behaviors within state subroutine
- Avoid actions on transition

http://www.ece.cmu.edu/~ece642/lectures/08_modalstatechart.pdf
Questions?
Homework Discussion