# CISC 440 - Artificial Intelligence & Robotics
## Spring 2020

## Assignment 07: Due Thursday, 4/30 by 11:59pm

For this assignment, you will finish a hybrid agent that plays a slightly different version of the game "StudentWorld". The difference is that, instead of 5 students randomly placed on the board, each cell (other than (0,0), (0,1), (1,0), and the door cell) contains a student with a probability of 0.2. This implementation will use a "probabilistic inference engine" that, when the best cell options to visit next are all risky, identifies which of those is the least risky.

Begin by forking and cloning the provided assignment as described in the "Assignment Setup and Submission" document in Canvas. You will need to finish the implementation of the StudentWorldProbabilisticInferenceEngine class.

### StudentWorld

StudentWorld PEAS task environment description:

**Performance measure**
- Escaping through the exit door in the fewest number of moves without being consumed by a student.

**Environment**
- 5x5 grid of cells with 1 player starting in the upper left cell (0,0), 1 randomly selected cell containing a door, and all other cells containing a student with probability 0.2. Neither a student nor the door can be placed in (0,0), (0,1), or (1,0). The student and the door cannot be placed in the same cell.

**Actuators**
- The player can move up, right, down, or left.

**Sensors**
- All cells adjacent to a student will smell. All cells adjacent to the door will glow. The size of the grid is known. The agent knows which cells have been visited.

### StudentWorldProbabilisticInferenceEngine (studentworld)

**getTargetCell(frontierCells : List<String>, smellyCells : List<String>) : String**

This static method is called when all cells on the frontier are risky (or some are known to contain students). It is ultimately responsible for determining and returning the cell to visit next in the format "student (3,3) = [0,1]". It receives two lists of cells from the constraint solver:

- "frontierCells" contains a list of all cells on the frontier (including those that have students). The format for a cell that is known to contain a student looks like "student (2,3) = 1". The format for a risky cell looks like "student (3,3) = [0,1]".

- "smellyCells" contains a list of all cells known to be smelly. The format for these cells looks like "smelly (1,3) = 1".

This method will need to call the provided "generateRiskyCells", which does 2 things:

- populates a list of risky cells (separating them from those known to contain students)
- renames those known to contain students as "fixed-student (2,3) = 1" (for the sake of the constraint solver, which was difficult to implement. I just wanted you to know that.)

"getTargetCell" then needs to use "calculateStudentProbabilityForCell" to determine the probability of each risky cell containing a student. Using that information, this method identifies and returns the cell to visit.

**calculateStudentProbabilityForCell(cell : String, frontierCells : List<String>, smellyCells : List<String>) :double**

This static method accepts 3 parameters: the cell for which to calculate the probability of containing a student, a list of all cells on the frontier, and a list of all smelly cells. The syntax of these parameter Strings matches that described in the description for "getTargetCell".

This method should call the provided "caclulateProbabilitySum" method as needed to determine the probability that the given cell contains a student. The "calculateProbabilitySum" method uses the "StudentWorldProbabilisticInferenceEngineConstraintSolver" 😎 (which again was difficult to implement) to return the sum of the probabilities of all possible risky cell states, given a pre-determined student presence (indicated using the "hasStudent" parameter) for one of those risky cells (indicated using the "cell" parameter).

"calculateStudentProbabilityForCell" needs to work with "calculateProbabilitySum" as described by the textbook and the final few slides in the corresponding lecture when revisiting wumpus world. Understanding this process is intended to be a key takeaway from this assignment. It will take some time to understand it.

## Odds and Ends

Your StudentWorldProbabilisticInferenceEngine.java file will be used along with the other provided files when grading. Your grade be determined largely by the number of tests passed in StudentWorldProbabilisticInferenceEngineTest.java. While these tests are a good indicator of correctness, they do not guarantee a fully correct implementation. In other words, passing all tests may not translate to a 100% on the assignment.

Submit your assignment by pushing your code to your gitlab repository fork by the deadline.