A4b

Resolution Theorem Proving

Trung Le and Johnny Le

10/18/2016

CS 4300 - 001

1. Introduction - Johnny Le

The objective of the Hybrid Wumpus Agent is to develop an agent who can decipher percepts to determine where safe squares are located. It has the ability to transform percepts into sentences which can be passed into the knowledge base. Utilizing the knowledge base and a series of queries, implications can be summarized. Once done, the agent employs A* path finding to move from one location to another and also uses this to move into a proper location to shoot at a possible wumpus square. Ask and Tell functions allow the agent to query and add to the knowledge base.

With these sets of abilities, the agent evolves their knowledge base and uses it to find the best route.

Questions:

1. What is the number of average number of steps it takes to run to completion?
2. How often is the gold found?
3. How often is the Wumpus shot versus the number of misses?

2. Method - Trung Le

Our hybrid wumpus agent follows the pseudocode given in figure 7.20 however it does not factor in time. It incorporates a knowledge base and uses propositional logic to infer statements about the world.
The knowledge base is initialized initially with atemporal statements about the world.
The hybrid wumpus agent is modeled in **CS4300_Hybrid_Wumpus_Agent**. It first tells the knowledge base the percepts it perceives at a given square and these percepts are created into CNF sentences by **CS4300_Make_Percept_Sentence**. **CS4300_Make_Percept_Sentence** calls **CS4300_Get_Index** to encode the percept at the given square to a specified integer and returns the sentence. **CS4300_Get_Index** takes in an x,y a negation and a type. Details about the encoding are given below. The agent then asks whether it's neighbors which have not been deemed as safe yet, are safe. This is added to a board 4x4 matrix. It asks the knowledge base if there is a glitter. If there is, the agent grabs the glitter and calls **CS4300_Plan_Route** to return to [1,1] and climb out. **CS4300_Plan_Route** uses A* search to reach the target cell using only squares it knows is safe. We modified A* search in **CS4300_Wumpus_A_Star_Safe** which only looks at safe squares.
If the agent does not perceive a glitter, then it plans a route to a safe fringe square on the frontier. That being a neighbor square that has been deemed safe and has not been visited. It adds this to the plan.
If there are no safe squares, the agent tries to make a safe square by shooting an arrow and killing a wumpus. It asks where it cannot be proven that there is not a wumpus, meaning it looks

for a square where there is potentially a wumpus. It then calls **CS4300_Plan_Shot** to line up a shot.
If there are no squares it can make safe, then it will have to take a risk and plan a route to a potentially unsafe square. That is, a square where it cannot be proven that it is not safe.
If there are no such squares, the pseudo-code calls for the agent to turn around and give up however we implemented such that we take a random guess on any unvisited squares. It returns the action that is popped from plan.

Encoding information

Pits	[1,16]
Breezes	[17,32]
Stench	[33,48]
Wumpus	[49,65]
Gold	[66,81]

Types

Pits	0
Breeze	1
Stench	2
Wumpus	3
Gold	4

Board matrix: Keeps track of safe squares, unsafe squares and squares which you do not know about.
1: Unsafe
-1: Unknown
0: Safe

Visited matrix: Keeps track of visited squares
1: Visited
0: Unvisited

Frontier matrix: Keeps track of neighbor squares
1: On frontier
0: Not on frontier

3. Verification - Johnny Le

To verify the solution, a grid was presented with a clear path to the gold. This was found quickly.

0	0	0	0
0	0	0	0
0	0	0	0
0	2	0	0

<u>Steps:</u>

0, 1, 4, 3, 3, 1, 6

Furthermore, a grid with an inaccessible gold was given to test failure.

0	0	0	0
0	2	0	0
1	0	0	0
0	1	1	0

<u>Steps:</u>

0, 4, 3, 1

Finally, a grid that only had one clear path was given to verify success. Success was seen less than 50% of the time with one trial shown below. This was attributed to the chance of going in either direction. Behaved as expected.

0	1	1	1
0	1	1	1
0	1	1	1
0	2	0	0

Steps:

0, 1, 3, 3, 1, 2, 1, 2, 5, 3, 1, 1, 2, 1

4. Data and Analysis - Trung Le

Using the first board

0	0	0	3
0	0	0	0
2	1	0	0
->	0	0	0

The agent achieved a score of 995 on this board. It performed the optimum number of steps. It's actions are recorded as:

Turn left	-1
Go forward	-1
Grab	
Turn left	-1
Turn left	-1
Go forward	-1
Climb	+1000

The KB is initialized with atemporal statements about the world.
After the first call to agent with the percept [0,0,0,0,0] the KB consists of the atemporal statements plus -33, -17,- 65 to indicate there is no breeze, no gold and no stench.
The second call to the agent will be with the same percept as the agent has not relocated.
The third call to the agent with be with [0,1,1,0,0] and the KB is appended with -37,21,69 to indicate there is a breeze, no stench and glitter.

The agent then detects there is a gold and grabs the gold. It then figures out a way to return to [1,1] by calling **CS4300_Plan_Route**.
The fourth call to the agent returns the action to turn left. Then the fifth calls says to turn left again. The sixth call says to go forward back to [1,1]. The KB is unchanged as we have not visited any new squares. The agent then arrives at [1,1] then climbs out and we are done.

For the second board

0	0	0	1
3	2	1	0
0	0	0	0
->	0	1	0

The agent achieved a score of 984 on this board.
It’s actions are recorded as:

Turn left	-1
Go forward	-1
Turn left	-1
Turn left	-1
Go forward	-1
Turn left	-1
Go forward	-1
Turn left	-1
Go forward	-1
Go forward	-1
Grab	-1
Turn left	-1
Turn left	-1
Go forward	-1
Go forward	-1
Turn right	-1
Go forward	-1
Climb	+1000

Call to Agent	State	Percept	Action	KB Additions
1	[1,1,0]	[0,0,0,0,0]	3	-33,-17,-65
2	[1,1,1]	[0,0,0,0,0]	1	
3	[1,2,1]	[1,0,0,0,0]	3	37,-21,-69
4	[1,2,2]	[1,0,0,0,0]	3	
5	[1,2,3]	[1,0,0,0,0]	1	
6	[1,1,3]	[0,0,0,0,0]	2	
7	[1,1,0]	[0,0,0,0,0]	1	
8	[2,1,0]	[0,1,0,0,0]	3	-34,18
9	[2,1,1]	[0,1,0,0,0]	1	
10	[2,2,1]	[0,0,0,0,0]	1	-22,-70
11	[2,3,1]	[1,1,1,0,0]	4	42,26,74

The agent then calls **CS4300_Plan_Route** to return to [1,1].

For board 3

0	0	0	0
0	0	0	0
3	2	0	0
->	1	0	0

For this board, the agent performed the following actions:

Shot	-10
Turn left	3
Go forward	1

Sadly, he died after shooting at the pit (2,1) then turning left and running into the wumpus at (1,2).

5. Interpretation - Johnny Le

According to the data, when the agent had a clear path to the gold, the agent was able to locate the gold and navigate back in the swiftest manner. When multiple obstacles were presented, the agent opted to take safer steps first before continuing which increased the number of steps but still located the gold.

Due to randomization, on board 5 the agent had a 50% chance of surviving as the agent had to decide between shooting in the ‘0’ direction or the ‘1’ direction. If the ‘0’ was ever chosen, the agent would not hear a scream and would thus have the unfortunate choice of choosing between a pit and a wumpus. However, if the wumpus was shot, in every scenario the agent would eventually find the gold.

With these findings, it demonstrates the power of the hybrid wumpus agent in locating the gold. As long as enough information is present to infer an answer, the agent will locate the gold without fail.

For situations with two equally viable options, the agent would not be able to make a more educated guess than a simple random choice. As can be seen with the first board, the optimum number of steps was easily achieved at 7 with a score of 995. The second board proved more challenging as it resulted in a score of 978 with the final board scoring slightly lower than that trending around 973.

The average scores for each board were 992 for 11 average steps, consistently 981 for 19 average steps, and the third board came in two varieties.

On the third board, when the agent died with a score of -1012 for 4 steps each time. When the agent survived, he/she had a score of 968 with 26 steps each time.

6. Critique - Trung Le

A4b taught important concepts regarding the knowledge base and its use. Determining the logical backend of the concept in conjunction with sentences created a difficult scenario that will surely only get more difficult to comprehend when probability becomes involved. The most powerful tool gained was in regards to the creation of numerous data structures for tracking various details such as which squares were safe or potentially dangerous. Navigating through the tricky logic of avoiding contradictions while querying the right sentences was insightful.

This problem would be most benefitted from truly random selection and additional data structures associated with each square such as a value that marks how dangerous it is. It would allow implementation of a priority queue and the highest chance of survival when deciding between two or more unprovable squares. Further modifications include increasing the size of the board for different problem sets.

7. Log - Johnny Le

Interpretation
- Johnny: 3 Hours
- Trung: 3 Hours

Coding
- Johnny: 15 Hours
- Trung: 18 Hours

Analysis
- Johnny: 2 Hours
- Trung: 2 Hours

Pseudocode in figure 7.20

```
function HYBRID-WUMPUS-AGENT(percept) returns an action
inputs: percept, a list, [stench,breeze,glitter,bump,scream]
persistent: KB , a knowledge base, initially the atemporal “wumpus physics”

t , a counter, initially 0, indicating time plan, an action sequence, initially empty

TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t)) TELL the KB the temporal “physics” sentences for time t
safe ←{[x,y] : ASK(KB,$OK_{tx,y}$) = true}
if ASK(KB,$Glitter_t$) = true then

plan ←[Grab] + PLAN-ROUTE(current,{[1,1]},safe) + [Climb] if plan is empty then

unvisited←{[x,y]:ASK(KB,$L_{t'}$ )=falseforallt′≤t} x,y

plan ← PLAN-ROUTE(current,unvisited ∩safe,safe) if plan is empty and ASK(KB,$HaveArrow_t$) = true then

possible wumpus ←{[x,y] : ASK(KB,¬ $W_{x,y}$) = false}

plan←PLAN-SHOT(current,possible wumpus,safe) if plan is empty then // no choice but to take a risk

not unsafe ←{[x,y] : ASK(KB,¬ $OK_{tx,y}$) = false}

plan ← PLAN-ROUTE(current,unvisited ∩not unsafe,safe) if plan is empty then

plan ←PLAN-ROUTE(current,{[1,1]},safe) + [Climb] action ← POP(plan)
TELL(KB, MAKE-ACTION-SENTENCE(action, t)) t←t+1

return action
```