A4b - Logical Agent using Resolution Theorem Proving

CS 4300
Fall 2016
Leland Stenquist - 1,3,5
Ryan Keepers - 2,4,6

1. Introduction

In this experiment a Hybrid Wumpus Agent is used to explore the Wumpus World game. The point of the Hybrid Wumpus Agent is to test the effectiveness of using logic as an agent for solving problems. In particular this agent uses a knowledge base filled with the physics of its world and information gathered from percepts. Using this knowledge base the agent tries to prove theories about its world to help it make decisions.

This strategy for solving Wumpus World creates some interesting inquiries:

1. Does logic help the agent be more successful at solving the Wumpus World?
2. It seems logic would not be computationally efficient. Is this true?
3. If logic is computationally inefficient, is it worth the time that it takes to find a solution?

2. Method

The purpose of this assignment was to implement three primary functions: ASK, TELL, and HYBRID-WUMPUS-AGENT (henceforth HWA). ASK and TELL are both relatively simple functions. ASK posed a theorem and, using the Resolution Theorem Prover algorithm (provided by the professor), attempted to confirm the theorem against the current knowledge base. ASK returns a boolean, so if the theorem was confirmed then it returns true, otherwise it returns false.

TELL updates the knowledge base with a new set of clauses. Within the HWA function, TELL was utilized to add the clauses created by the current percepts. So, for example, if the current percept showed a breeze but no stench, then clauses were constructed for BREEZExy and ~STENCHxy, and those clauses were given to TELL to add to the knowledge base. The TELL function works by unioning its parameters, so that the knowledge base can never contain duplicate information.

HWA therefore contains the vast majority of the assignment, only calling TELL once, and ASK a handful of times where necessary. The behavior of HWA at a high level depends on two operational cases. First, the agent may be in the middle of a current plan of action (saved in HWA as a persistent variable). In this case, the function skips most of its operations and continues the agent along its plan. The only actions taken are to TELL the knowledge base about information from newly visited locations.

If the agent does not have a current plan of action, HWA must decide the next plan. The function can choose from four fundamental options at this point: grab the gold and leave, try to shoot the wumpus, explore a new location, or give up and leave. Once it decides on a course of action, the function utilizes the A-Star algorithm (also provided by the professor) to pick a route through the board from the current location to the goal location.

Grabbing the gold is a simple scenario: the agent needs to know that it is standing on top of the location containing the gold. This is signified with a GLITTER percept. If the current location glitters, the agent picks up the gold, plots a path back along its previously visited locations towards the starting location, and climbs out.

The function may have the agent try to shoot the wumpus. This only occurs in the event that the agent doesn't believe any of the unvisited locations are currently safe. If the agent still has an arrow, HWA can attempt to create a safe path by killing the wumpus. The function will parse all unvisited locations on the board, ASKing for potential locations of a wumpus. Those potential locations are get randomly sampled, and each sample will try to find a totally safe path to a position where the agent can shoot its arrow at presumed wumpus. If it is impossible for the agent to reach a shooting position from a totally safe path, another location is sampled. If no wumpus location allows for a safe shooting position, the agent declines to shoot. Otherwise, a plan is created for the agent to path along previously visited locations to the shooting position, face the wumpus, and finally fire the arrow.

The decision for the agent to move into unvisited locations is dependent on two tiers of safety. An unvisited location can be considered probably safe, or probably unsafe, depending on information from nearby percepts. In this implementation of HWA, a location was considered probably safe if none of its adjacent, visited locations contained a STENCH or BREEZE. A location was considered probably unsafe if one or more adjacent, visited locations contained a STENCH or BREEZE. These considerations could change in the event that a wumpus was successfully shot: after a SCREAMED percept occurs, location safety checks ignore STENCH and only consider BREEZE.

If a probably safe location is discovered, a plan is created for the agent to move to that location. In the event that all unvisited locations are probably unsafe, the function picks one of these locations at random, and creates a plan where the agent paths towards that random location. The decision to move into a probably safe location is generally unlikely to cause the agent to die, since a lack of surrounding percepts imply the safety of the location. Therefore it is significantly more likely for the agent to die by moving towards a probably unsafe location.

As a last ditch preservation of life: if the function decides that all unvisited locations are absolutely, certainly dangerous (which is to say, it ASKs and finds that all unvisited locations contain either a wumpus or pit) then the agent has not choice but to plan a course back to the initial location, and climb out, without getting the gold.

3. Verification of Program

CS4300_ASK

In this implementation of the CS4300_ASK function the RTP function is simply called and the result is then turned into a true or a false return statement. RTP was verified extensively in A4a. Therefore CS4300_ASK can be verified by referring to A4a.

CS4300_TELL

Tell is implemented by calling the function CS4300_cnf_union. The function CS4300_cnf_union is used extensively in RTP. Therefore this function has also gone through extensive testing in A4a. But for extra verification CS4300_TELL was run against choice CNF clauses with new clauses to be appended. The function's answer was compared to a hand written answer.

HUMAN		CS4300_TELL	
CNF 1	**CNF 2**	**CNF 1**	**CNF 2**
1 2 3 -1 2 1 2	3 4 3 4	c=[] c(1).clauses=[1 2 3] c(2).clauses=[-1 2] c(3).clauses=[1] c(4).clauses=[2]	d=[] d(1).clauses=[3] d(2).clauses=[4] d(3).clauses=[3 4]
Solution 1 2 3 -1 2 1 2 3		**Solution** ans(1).clauses=[1 2 3] ans(2).clauses=[-1 2] ans(3).clauses=[1] ans(4).clauses=[2] ans(5).clauses=[3]	

4 3 4		ans(6).clauses=[4] ans(7).clauses=[3 4]	
CNF 1	**CNF 2**	**CNF 1**	**CNF 2**
1 2 3 4 5 6 -1 2 1 2	3 4 -1 2	c=[] c(1).clauses=[1 2 3 4 5 6] c(2).clauses=[-1 2] c(3).clauses=[1] c(4).clauses=[2]	d=[] d(1).clauses=[3] d(2).clauses=[4] d(3).clauses=[-1 2]
Solution 1 2 3 4 5 6 -1 2 1 2 3 4		**Solution** ans(1).clauses=[1 2 3 4 5 6] ans(2).clauses=[-1 2] ans(3).clauses=[1] ans(4).clauses=[2] ans(5).clauses=[3] ans(6).clauses=[4]	

CS4300_HWA

This function was tested by validating it on the three provided boards. It was run 100 times against each board. Statistics were then used to make sure that it is getting different results. The trials can vary in result. This is because of the situation where an agent needs to make a potentially dangerous decision and there is no clear answer. In this situation a random choice is selected. For verification the statistics were compared to see if they matched an analysis of what a human would expect. For example:

1. **Board 1:** It would be expected that a Hybrid Wumpus Agent could always get out of board 1 with the gold. After 100 trials this experiment had 100 % success as expected.
2. **Board 2:** This board gets to the point where the Agent does not have a known safe place to visit. Because of this it will try to shoot the Wumpus. If it succeeds then it will easily find the gold. If not it would have to guess. A high probability of success would be expected for this situation. In an experiment of 100 trials 100% success was achieved. This was actually a surprise, but by testing with variations of board two we were able to feel confident in our results.
3. **Board 3:** This board is the hardest to solve it requires an attempted shot at the Wumpus and if that fails the agent attempts to proceed forward to its inevitable doom. Due to this, this board would likely have a lower probability of success.. Under 100 trials it succeeded 25% of the time. This meets with expectations.

Even though the Verification for Hybrid Wumpus Agent is not as well defined as some validations may be. It seems that through extensive runs this Agent has proven to be efficient at solving the Wumpus World problem.

4. Data and Analysis

100 trials on the 3 boards in CS4300_WW3.

Board	0 0 0 3 0 0 0 0 2 1 0 0 0 0 0 0	0 0 0 1 3 2 1 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 3 2 0 0 0 1 0 0
Success Rate	1.00	1.00	0.25
Death Rate	0.00	0.00	0.75
Average Time	1.73	5.13	2.25

Boards 1 and 2 seem relatively flawless in result. Board 1 seems like an obvious result: with relatively few dangers and a gold position that has little interference with percepts like breezes or stenches, it's only a matter of the agent exploring safe, unvisited locations to find the gold. Similarly, Board 3 is unsurprising. Even when the wumpus is successfully shot (which is not a guarantee), the knowledge base can't guarantee that either path is safe. Ultimately, the agent is always resigned to stepping forward and hoping for the best.

However, we were surprised to see that Board 2 was so completely successful (and remained so, across other trials as well). Therefore, we decided to conduct the further tests on some variants of board 2, shown below.

50 trials on board 2 variants

Variants	0 0 0 1 1 2 1 0 0 0 0 0 0 0 3 0	0 0 0 1 1 2 3 0 0 0 0 0 0 0 1 0	0 0 0 3 1 2 1 0 0 0 0 0 0 0 1 0
Success Rate	1.00	0.25	0.3
Average Score	978	-585.84	-466.84
Average Time	5.08	4.72	4.74

Some tinkering with the layout of the board showed that the proximity of the wumpus, relative to the pits, is a primary driving factor of success for our HWA function. The variant 1 board shows that if the agent is stuck in a position that can be resolved by shooting the wumpus, it manages to do so successfully and gain the gold. This situation circumvents the randomization of visiting probably unsafe locations, because killing the wumpus opens up portions of the board to be viewed as “safe” again. However, put the agent between two properly spaced pits, as seen in variants 2 and 3, and it falls back on randomly choosing from potentially unsafe locations. In this situation there are three potentially unsafe and unvisited spots to attempt (3,1; 2,2; 1,3), and with only one safe choice it makes sense that the overall success rate drops to about 30%.

5. Interpretation

In this experiment three questions were asked. The first delved into the effectiveness of using logic to help an agent solve the Wumpus World. Though only three boards were tested it is clear from the test that logic was successful in helping the agent to solve the Wumpus World. The prove of this is in the results. For board 1, 100% of the attempts were successful in bringing back the gold. For board 2, 100% of the attempts were successful in bringing back the gold. For board 3 25% of the attempts were successful in bringing back the gold. Clearly the agent had a good amount of success using logic.

The second question asked was about the efficiency of the algorithm. In this interpretation this is the time efficiency of the algorithm. Still it could be argued that logic has a strong potential to also be inefficient in its memory usage. To check this an average amount of time was gathered for running CS4300_WW3. An average time was also gather for running against boards 1, 2, and 3. The average time it took to run CS4300_WW3 was 16.2977 seconds. Board 1 seemed to take about 1.7 seconds to run. Board 2 seemed to take about 5.13 seconds and board 3 took about 2.25 seconds. Since these are small boards and the Wumpus World is not a hugely complicated problem it seems that using logical agents can be extremely time consuming.

The last question asked is about whether or not logic is worthwhile. Is logic efficient enough that it is worth a hit in the speed of the agent and the resources required by the agent. The answer to this question depends on other methods. If there are better methods for solving the problem(methods that are faster and require less resources), the answer is definitely no. Logic is not worth it. But logic in the most part was very successful in this experiment. Therefore, if infinite resources are provided and time is not a problem them it would seem that logic is a good solution to a problem.

6. Critique

The primary takeaway from this assignment is to learn that I, and my coding partner, are not more clever than the book. On two separate occasions we read the provided pseudocode, thought about what it sought to achieve, and then decided to to deviate into what we considered a more robust or more efficient take on the subject. Our third time sitting down with the program we decided to simply write what was provided for us, and were finally successful.

The majority of our hardship came, we believe, from not having a complete comprehension of how the logic could affect the decision making process. Most of our exercises in variant algorithms attempted to shortcut costly actions (such as multiple ASK calls) or dodge the issue of inconclusive proposals (we tried many ways to prove, for example, that a location did or did not contain a pit or wumpus, rather than let some ASK calls tell us it couldn't be proven). These trials were somewhat successful, but tended to steer us towards overfitting for particular test scenarios rather than provide a robust solution.

We are still undecided that the algorithm couldn't be improved by extrapolating knowledge from our given percepts. For example, a location which has no breeze must necessarily be surrounded by locations that do not contain pits, according to the rules of Wumpus World. We can sometimes resolve the same conclusion without the aforementioned information, and there are times where this inclusion could not help us (such as board 3 in the first data table). But overall we still find the consideration compelling as a possible way to improve the program. And in considering this stipulation it seems like there may be some (perhaps many) other tweaks and additions of similar style which we haven't yet thought of, which could also help. Then again, as a nod to Occam's razor, this leads to the question of whether these accumulated tweaks would, at large, improve the algorithm, or simply convolute it.

7. Log

Ryan : 18 hours
Sections 2, 4, 6
Leland : 18 hours
Sections 1, 3, 5