CSCI 446 Artificial Intelligence Project 2 Final Report

Roy Smart Nevin Leh Brian Marsh

October 24, 2016

Abstract

Our project attempts to solve the problem of Logic and the Wumpus World by using two agents of varying sophistication to navigate the environment. The agents used are a reasoning agent and a reactive agent. The reasoning agent utilized first order logic and resolution to navigate the world while the reactive agent used its current state and its previous states to make decisions. The performance of these two agents were compared to determine which one navigated the Wumpus World in the least amount of steps. It was found that the reasoning agent performed better on Wumpus Worlds of varying size, though this difference became much more apparent on Wumpus Worlds that are larger or have a large number of obstacles.

1 Introduction

Our task was to create a reasoning agent and reactive agent to navigate a WumpusWorld. *Logic and the Wumpus World* is an artificial intelligence problem first proposed by Michael Genesereth, and described in detail by his student Stuart Russel[Russel and Norvig, 2010]. The problem involves navigating an environment known as the Wumpus World using logic to avoid dangers as the agent attempts to reach a goal. The environment is a square grid of tiles that can be empty or contain gold for the goal state, a pit that the agent will fall into, or a monster known as a wumpus. As the agent navigates the environment, a score is calculated from the various actions that the agent makes. The objective of the game is thus to maximize the score.

Our reasoning agent uses first order logic and the process of resolution to navigate the Wumpus World and find the gold. Resolution relies on the concept of proof by contradiction to deduce things about the Wumpus World and uses those deductions to make further deductions about the world.

Meanwhile, our reactive agent does not use resolution and relies solely on the information supplied at its current tile and the information from tiles it has already visited to make decisions as to how to find the gold. This agent uses a kind of directed random walk to navigate the Wumpus World.

We will be comparing the performance of the two agents to determine which one solves the Wumpus World problem more efficiently. We hypothesize that the reasoning agent will perform better than the reactive agent due to its heightened knowledge of the world and its ability to infer things about the world that the reactive agent has no way of knowing.

2 Problem Generation

3 Reactive Agent

The first agent that was implemented was the reactive agent. This agent was not attached to the logical reasoning system and made decisions using an `if/else` tree. This agent was not a purely reactive agent as it saved old states and used that information to improve its functionality[Weiss, 1999].

The reactive agent had very basic functionality. As it searched the space it would mark squares that have been visited as safe. If the agent encountered a breeze or a stench it would move to a randomly chosen safe tile. If the agent was on a tile that was not breezy and not smelly it randomly chose from the set of adjacent tiles that had not been explored yet. If all tiles squares had been explored a tile was chosen at random.

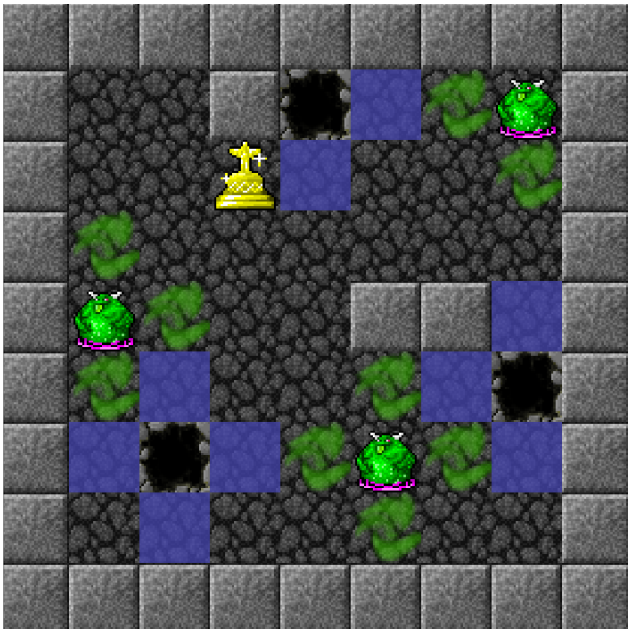

Figure 1: *An example of a wumpus world produced by our problem generator. The green haze represents stench and the blue tiles represent breezy squares.*

The reactive agent also had a special mode that was used to take chances if the space was currently searched sufficiently. To implement this we added a counter that incremented ever time the agent had to move to a previously explored tile and was set to zero every time the agent encountered an unexplored tile. If the counter reached a certain threshold a chance was taken.

4 Reasoning Agent

Our reasoning agent has the ability to navigate through each of the caves while avoiding wumpi, pits and barriers in search of the gold. We did not have time to implement a shooting routine, so wumpi and pits are notionally the same to our program. The reasoning agent is designed to navigate the caves through the use of af first-order logic *inference engine*. This inference engine is an automated theorem prover based on unification and resolution. These algorithms use clauses in conjuctive normal form (CNF) that represent the rules of the wumpus world. The inference engine can use these rules, and the knowledge gained exploring the cave to locate and avoid obstacles.

4.1 Resolution

The application of resolution to automated theorem proving was introduced by J.A. Robinson in 1965. In first-order logic, resolution forms a single inference rule that can be used to construct proofs by refutation [Robinson, 1965]. Our reasoning agent will use resolution to prove that a square in the wumpus word is safe.

Two clauses in CNF may be resolved if they contain complementary literals. In predicate-order logic, two literals are complimentary if one is the negation of the other. In first-order logic, we add the additional distinction that two literals are complimentary if one is the negation of the other following unification (variable substitution)[Russel and Norvig, 2010]. For example, the clauses

$$\text{Breezy}(s) \vee \neg\text{Pit}(s)$$
$$\text{Clear}(r) \vee \text{Pit}(r) \vee \text{Wall}(r) \vee \text{Wumpus}(r)$$

contain the complementary literals $\neg\text{Pit}(s)$ and $\text{Pit}(r)$, since r and s are taken to be variables in this example. If two clauses can be resolved, the result of the resolution is a new clause with the complementary literals removed [Russel and Norvig, 2010]. For example, resolving the clauses presented above, we get

$$\text{Breezy}(s) \vee \text{Clear}(s) \vee \text{Wall}(s) \vee \text{Wumpus}(s)$$

We can use resolution for theorem proving by using proof by contradiction. In the wumpus world, our agent asks the inference engine a yes/no question. The inference engine then negates the query and uses resolution to find either a tautology or a contradiction.

Initially, we based our resolution algorithm off of the function PL-Resolution described in Figure 7.5 of Russell and Norvig. This algorithm attempts to find a contradiction or tautology by resolving every resolvable clause with every other resolvable clause. As one might expect, this technique is very inefficient, so we turned to an algorithm known as *linear resolution*. Linear resolution improves on the exhaustive approach pf PL-Resolution by only attempting to resolve clauses that are either in the knowledge base or an ancestor of the original input query[Russel and Norvig, 2010]. By only resolving clauses with the input query, we were able to drastically reduce the search space of the resolution algorithm.

In linear resolution, the search space forms a tree, since each successive resolution step produces an arbitrary number of descendants, where each descendant will be again be resolved with the knowledge base. The performance of our resolution algorithm then directly depends on the strategy for searching this linear resolution tree. The strategy adopted by our implementation is the breadth-first strategy described by Lawrence and Starkey. For this search strategy, the inference engine resolves every clause at each layer, before descending to the next layer. This approach prevents the resolution algorithm from getting stuck in the search tree if it uses unideal rule.

To further improve the efficiency of the resolution problem, we made sure to only search local rules. In the wumpus world, the knowledge base consists of many facts associated with each location, but our rules only concern neighboring squares. Therefore, we only tried to resolve rules from the squares adjacent to the agent's current square. This prevented the resolution process from having to search the whole board for a single rule.

4.2 Unification

Resolution relies on the concept of unification to check if two clauses can be unified[Robinson, 1965]. Our unification algorithm is based off of the function Unify presented in Russell and Norvig Figure 9.1. Since many of our rules contain functions, we improved on Russell and Norvig's implementation using the unification algorithm described by [Robinson, 1965].

In unifying our rules with the knowledge base an interesting question presented itself: should we evaluate functions? Ultimately we tried it both ways. Evaluating functions led to a more robust resolution/unification algorithm, but was much less efficient than not evaluating functions. This is because evaluating functions leads to information propagation, e.g. resolution propagates across the whole board attempting to unify everything. If we left the functions unevaluated, this helped to restrict the resolution process to local scope, but had the disadvantage of having to have copies of information with different functional forms.

4.3 Pathfinding

To navigate the wumpus world, we embraced a simple recursive backtracking algorithm. This algorithm attempts to travel to only clear and unexplored squares. If a clear/unexplored square is unavailable, it backs up until a clear/unexplored square is found. If all accessible squares have been explored, the agent then does a random walk to attempt to find the gold.

It is worth mentioning here that we did not get the chance to implement a shooting routine for our agent. Unfortunately time constraints prevented us from realizing this feature.

5 Rules

Our rules are constructed to simply verify if a square is clear. Since there is the possibility of multiple pits and wumpi, there is no way to precisely determine where the obstacles are. Therefore we have to satisfy ourselves with the knowledge where the obstacles are not. The rules we implemented are as follows.

$$\text{Breezy}(s) \vee \neg\text{Pit}(\text{north}(s))$$
$$\text{Breezy}(s) \vee \neg\text{Pit}(\text{south}(s))$$
$$\text{Breezy}(s) \vee \neg\text{Pit}(\text{east}(s))$$
$$\text{Breezy}(s) \vee \neg\text{Pit}(\text{west}(s))$$
$$\text{Stinky}(s) \vee \neg\text{Wumpus}(\text{north}(s))$$
$$\text{Stinky}(s) \vee \neg\text{Wumpus}(\text{south}(s))$$
$$\text{Stinky}(s) \vee \neg\text{Wumpus}(\text{east}(s))$$
$$\text{Stinky}(s) \vee \neg\text{Wumpus}(\text{west}(s))$$
$$\text{Breezy}(s) \vee \text{Clear}(s) \vee \text{Wall}(s) \vee \text{Wumpus}(s)$$

So to navigate the wumpus world, our agent simply asks a square is clear. Then the resolution algorithm uses the rules above to determine if the square is accessible. Notice that these rules can only determine if a square is clear

6 Comparing Algorithm Performance

6.1 Experimental Approach

To measure the performance of our agents we made a scoring system that decrements each time a move is made. A move is defined as rotating left, rotating right, and moving forward. The score will start at 1000 and finding the gold will yield an additional 1000 points to be added to the score.

Using the performance measurement, we then varied the parameters of the wumpus world such as the size, number of wumpi, number of pits, etc. and plotted the relationships between the size of the world, number of obstacles, and the score. For simplicities sake we used constant values for the number of pits, barriers, and wumpi. Therefore, each world had the same number of pits, barriers, and wumpi. We ran each algorithm on graph sizes {5, 10, 15,...,25} with ten runs for each size. The number of obstacles was roughly computed to have a constant obstacle density.

We then used *Mathematica* to produce a 3D plot to display the data in terms of score vs. size vs. number of obstacles

6.2 Results

In general, the results presented exactly as hypothesized. The reasoning agent outstripped the reactive agent in terms of score regardless of the size of the world or the number of obstacles in the world. However, size and the number of obstacles did affect by how much the reasoning agent was better than the reactive agent. For very small board sizes with minimal obstacles the two algorithms scored almost evenly, but for large worlds with a large number of obstacles the reasoning agent far outstrips the reactive agent.

One reason the reactive agent was almost as good as the reasoning agent in small worlds is that the search space is so small that the reactive agent can search the whole safe space in nearly the same number of steps as the logical agent. Since the number of steps to search the space has a large effect on the score, the scores for the small worlds are almost the same.

There are two main reasons the reasoning agent is much faster on large worlds with large amounts of obstacles. The first is that the search is much better. The random walk of the reactive agent gets stuck

on large worlds and has a hard time searching the whole space efficiently. The second reason is that the reasoning agent can reason its way through obstacles and can deduce information about tiles it has not visited yet. It can also update information as more facts are reasoned out. Meanwhile, the reactive agent does not have this ability and cannot mark a tile unless it has visited that tile.

7 Summary

References

[Robinson, 1965] Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41.

[Russel and Norvig, 2010] Russel, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, New Jersey 07458, 3rd edition.

[Weiss, 1999] Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, Massachusetts.