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 three agents of varying sophistication to navigate the environment. The agents used are the Reasoning Agent, the Reactive Agent, and the Human Agent. Performance of the different agents was based upon their average score, which is improved by finding the gold and killing the wumpus and lessened by falling into pits, being killed by the wumpus, and exploring cells.

1 Introduction

2 Problem Generation

3 Reactive Agent

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

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

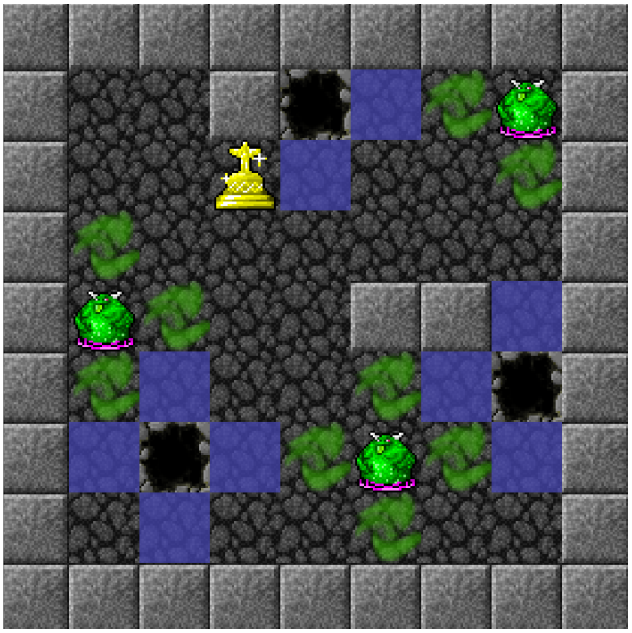

Figure 1: *An example of a wumpus world produced by our problem generator. The green haze represents stench and the blue tiles represent breezy squares.*

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

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

6 Comparing Algorithm Performance

6.1 Experimental Approach

6.2 Results

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.