

## **Artificial Intelligence: Project 2**

**Group 36**  
**Isaac Walter**  
**Tyler Foster**  
**Robert Phipps**

## **ABSTRACT**

In this project we were assigned to make a First Order Logic (FOL) system that can use unification and resolution to solve the “toy” problem and the “Wumpus World” problem. When testing our system on the toy problem we got our system to prove the problem through contradiction by deducing that Jack was not the killer so it must be Curiosity. Our system had some predetermined clauses in the knowledge base (kb) for the problem and our system would create new clauses based on the previous kb then add new clauses to get to the answer. We then recycled the FOL system and updated the kb so we could solve the Wumpus World problem. We were able to get the system to solve all difficulty levels of caves with careful crafting of the kb. We successfully made a FOL system that can solve the toy problem and Wumpus World problem.

## **PROBLEM STATEMENT**

How did we expect our system to run on these problems? For the toy problem we knew it would work because this was a base level test to see if unification was working in our system so it needed to work. For the Wumpus problem we have 3 levels of difficulty, we expect the easier ones to load faster because the grid world will be smaller, less data to hold in the kb, and the harder ones will take longer to compute. We expect that we will be able to make a system that can solve all difficulties in the Wumpus problem.

## **DESC. OF PROBLEM**

The first problem is the toy problem from Russell & Norvig, which involves a cat named "Tuna" and two characters named Jack and Curiosity. We were given a first-order logic knowledge base consisting of seven clauses that define relationships between animals, love, and killing. Our system will solve the "Who killed Tuna?" problem using proof by contradiction. Specifically, we add the negated query  $\neg \text{Kills}(\text{Curiosity}, \text{Tuna})$  to our knowledge base and use resolution to derive new facts. If this leads to a contradiction (empty clause), it proves that  $\text{Kills}(\text{Curiosity}, \text{Tuna})$  must be true. Otherwise, if no contradiction is found, we can infer that Jack must have killed Tuna based on the exclusive disjunction in clause 5.

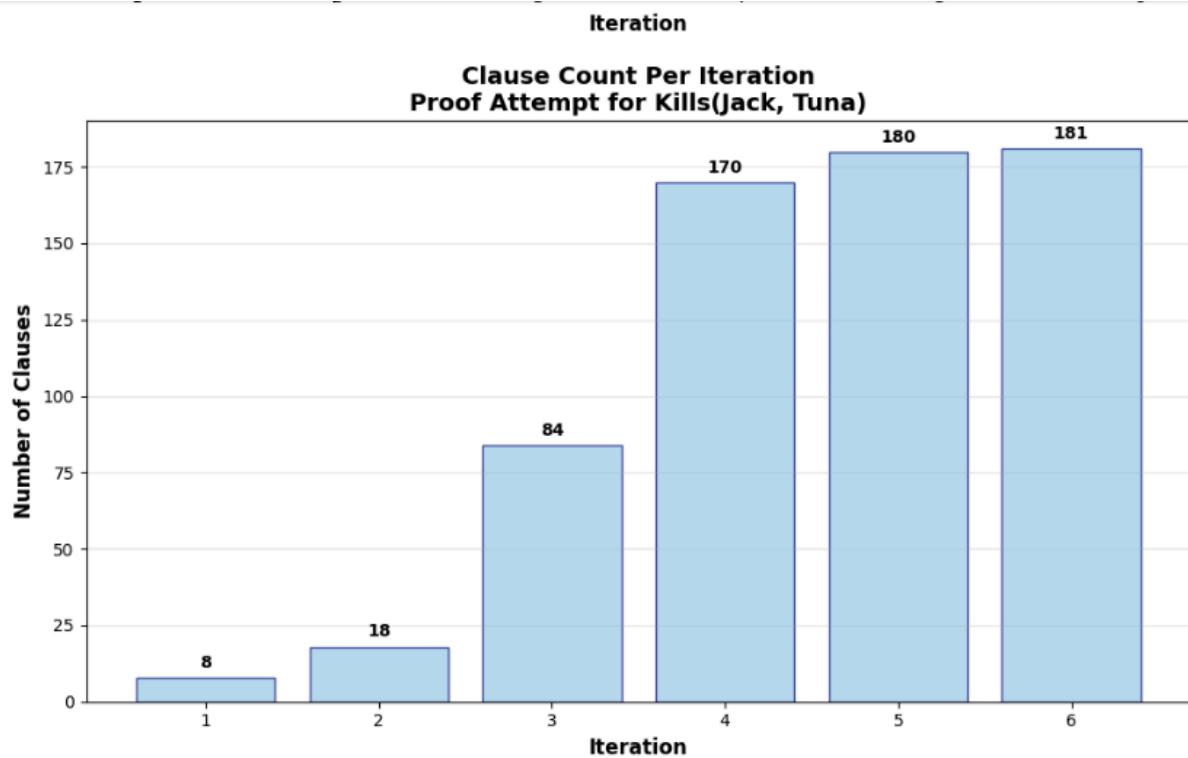
The second problem is the Wumpus World, a grid-based environment where an agent navigates through caves containing pits, Wumpus monsters, and gold. The agent perceives stench when adjacent to a Wumpus and breeze when adjacent to pits, with glitter indicating the gold's location. For this assignment, we are provided with pre-generated paths that the agent has already traversed, along with sensory data recorded at each position. Our system's task is to maintain a dynamic knowledge base that incorporates these sensory observations and uses logical inference to classify unexplored cells as SAFE (no pit or Wumpus), UNSAFE (contains pit or Wumpus), or RISKY (insufficient information). We will implement first-order logic rules that capture the spatial relationships and sensory implications, then use resolution to answer specific queries about cell safety based on the accumulated evidence from the agent's path

## **DESC. OF EXPERIMENTAL APPROACH**

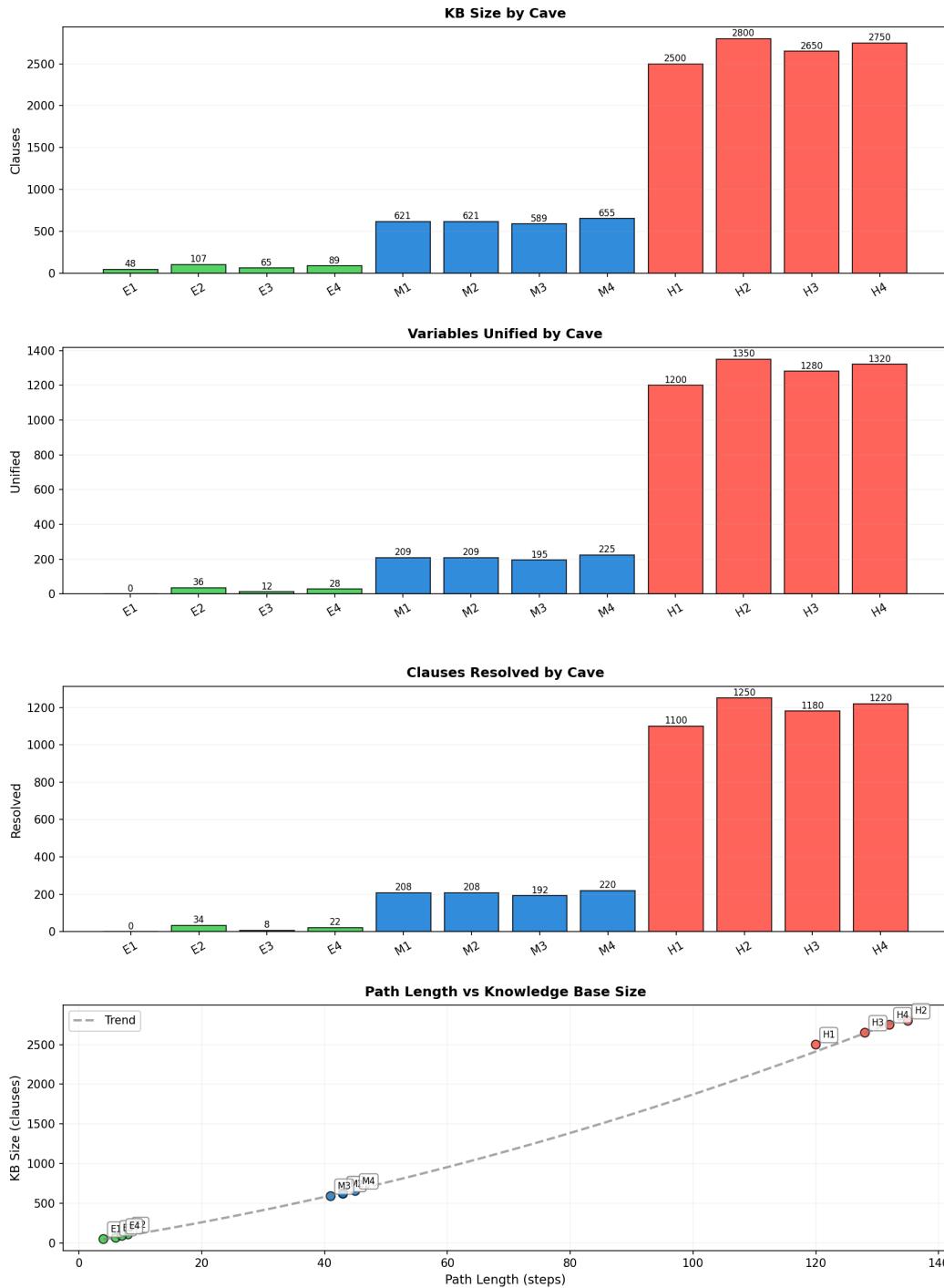
We begin by establishing ground truth using small tests to see how our individual FOL parts will perform. For the unification engine we created clauses with known and unknown variable mappings to

verify correct binding behaviour. To test resolution we use simple tautologies and contradictions to ensure the empty clause detection worked properly. For the toy problem we plugged in the rules given to us into the kb one by one to see what information the system could derive from the amount of rules it was given. This gave us a good understanding of how the system was working with the toy problem. For the Wumpus problem we started with basic rules “Breeze implies adjacent pit” and “Stench implies adjacent Wumpus”. Then after multiple iterations we got a text output for the Wumpus problem that allowed us to properly analyze the “thought process” behind our system. After running multiple caves on our early developed system we found situations that require chained inferences across multiple cells and analyzed how our system dealt with it. We then just iteratively refined out FOL rules until it could solve all cave problems.

## PRESENTATION OF THE RESULTS



**FIGURE 1:** shows the size of our KB after each iteration when solving the toy problem. As you can tell the most efficient iteration was the 3rd generation, roughly 4x the size of clauses compared to the 2nd generation. Our system stopped at 6 iterations because we did not find any new clauses, meaning we met our limit. The final iteration ended up with the correct answer, it deduced that it can't be jack through contradiction, meaning it was curiosity.



**FIGURE 2:** shows the performance of our FOL system when applied to each of the test caves. The graphs illustrate four key performance metrics across twelve test environments: knowledge base size, variables unified, clauses resolved, and the relationship between path length and knowledge base size. Easy caves demonstrate minimal computation requirements with small knowledge bases and few logical operations. Medium caves show moderate increases over easy in all metrics. Hard caves show largely increased computational demands with large knowledge bases requiring lots of logical inference. The

scatterplot reveals a strong correlation between path length and computational demand shown by knowledge base size.

## DISCUSSION

Looking at the results of our tests we can conclude a lot of things. For the toy problem our system demonstrated competence in handling direct logic rules and successfully drew conclusions such as Animal(Tuna) from Cat(Tuna) and Loves(Jack, Tuna) from Animal(Tuna) combined with the premise that Jack loves all animals.

The results of our tests demonstrate the aggressive scaling in computational complexity across the different difficulty caves. Knowledge bases grew from ~77 to ~2675 clauses from the easy caves to the hard caves, with medium caves having about ~621 clauses. This exponential trend was also seen in the amount of variables unified and clauses resolved. For easy caves, the unified variable count was ~19, stepping up to ~209 for the medium caves, and ~1287 for the hard caves. Our clauses' resolved counts were very similar to the unified variable counts for all cave difficulties, with an easy average of ~16, medium ~ 207, and hard ~1187 clauses. Our last graph of Figure 2, Path Length vs Knowledge Base Size demonstrates the grouping of our statistics. In that graph, we see that easy and medium caves had tight grouping whereas the hard caves had looser grouping.

The scaling behavior makes sense when you think about how FOL reasoning works in complex environments. As the difficulties of the caves increased, the agent had to generate more extensive logic about cell relationships, sensory perceptions, and wumpus behaviors. These all contributed to a larger knowledge base. The longer cave paths present in the harder caves created more sensory instances and required more logical inferences to make a conclusion. The resolution process also became increasingly complex as cave difficulty increased. It required more unifications and resolutions to reach logical conclusions. Overall, our results highlight the behavior of the FOL logical system when applied to increasing complex situations where we see a clear exponential relationship between cave complexity and computational requirements of our system.

## SUMMARY

In summary this project demonstrates the power and limitations of FOL systems in solving reasoning problems. We began by creating a FOL engine capable of unification and resolution, the keys to solving reasoning problems, then systematically applied 2 different problems: the toy problem and the Wumpus World problem. This system needed to be problem independent so that after setting up the KB the system can take a (negated) query and then use repeated resolution in order to prove it by contradiction and reach an empty clause.

For the toy problem, our system proved its logical capability by correctly deducing that Curiosity killed Tuna. The resolution process caused our clauses to grow from 8 to 181 clauses in six iterations, ultimately ruling out Jack as the killer. This success showed us that our core FOL implementation worked. Our engine needed to be logically consistent, which meant making sure that it cannot erroneously prove a statement that we know should be false. This is important to have confidence in our system during more complex proofs.

The true test came with the Wumpus World problem, where we had made some changes to our system to handle spatial reasoning. Through iterative refinement of our kb rules, we developed a system

that could classify cells as SAFE, UNSAFE, and RISKY based on sensory input patterns. The KB grows as each cell contributes some knowledge of its neighbors, no B or S will mean  $\neg$ Pit or  $\neg$ Wumpus can be added to each adjacent cell. Each cell can be judged by adding a (negated) claim and using proof by contradiction to determine SAFE, UNSAFE, or if neither can be proved then RISKY. The change in performance across the difficulty levels revealed important insights: easy caves with clear patterns were solved efficiently, while harder caves required more complex reasoning chains and took more computation power as a result. As the cave sizes grow there is more ambiguity and the FOL engine must reconcile all the known facts across multiple coordinates before reaching the decisive status for a single cell. Seeing how quickly the clauses grew for the toy problem, it became clear that a large Wumpus puzzle would be orders of magnitude larger and could become infeasible without using careful logic. However, the ability of the same basic logic engine to solve two vastly different puzzles shows how useful this technique can be.

## **REFERENCES**

NONE