Inference Engine Within the Wumpus World

- CSCI 446 Artificial Intelligence -

Group 16
James Beck, David Rice, Tyler Wright

Montana State University
Gianforte School of Computing

Abstract

Logical inference is used to determine state based off of rules rather than direct perception. This paper explores the benefits of first order logical inference through the domain of the Wumpus World: an agent is told to explore a two dimensional world of rooms with dangers such as pits and monsters with a goal state of finding the single room containing gold. The agent utilizes an inference engine that is fed perceptions about the world from the agent while providing actions for the agent to take. The actions are based off of what the inference engine can decipher about rooms without having actually visited the room.

This paper explores the process of first order axiomatic sentences, unification, and resolution through proof by contradiction. A secondary agent that does not have any logical reasoning ability is also introduced that is used to compare the effectiveness of of logic-based agent. The logic-based agent dramatically outperforms the secondary agent in every metric except for the total number of decisions that are made through a dungeon exploration.

Contents

1 Problem Statement

This project is an implementation of first order logic, unification, and resolution through the setting of the Wumpus World. The Wumpus World is an explorable dungeon with the goal state of finding the room with gold in it; but there are pits, blocked rooms, and dangerous Wumpus monsters in the way. The agent has a bow and arrow at his disposal to kill the Wumpus and a refined inference engine to decide what rooms to go to and which actions to take.

The problem handled by this project focuses on the inference engine. For dungeons of sizes 5 x 5 to 25 x 25, inference rules are used to always make the safest choice depending on information about the dungeon the agent has revealed. The challenge is to decode how to state and use first-order clauses in a programmatic fashion.

Evaluation will compare the logic-based agent to a simple, reactive agent. The reactive agent always chooses what it believes is the most safe neighbor cell is based on only the current perception at time t for room $C_{x,y}(t)$. The metrics used to compare the two agents are:

1. Number of decisions made.
2. The gold was or was not found.
3. Number of Wumpi killed by the agent.
4. Number of times the agent fell in to a pit.
5. Number of times the agent is killed by a Wumpus.
6. Number of unique cells explored.
7. The agent's score.[1]

1.1 Hypothesis

Given the knowledge based agent (KB-agent) will have rules in place to never move into a potentially dangerous room until the inference engine can assure the room is safe, one can hypothesize the KB-agent will produce a better score than the reactive agent for every map size on average. For small map sizes[2] the reactive agent may outperform the KB-agent for minimizing number of decisions made and number of cells explored, but the reactive agent will be notably outperformed in all categories as the map size increases.

Furthermore, The KB-agent will likely perform in NP time. Run time will slow down dramatically as the search space becomes large due to exponential growth of logical comparisons that will be made[1].

[1] -1 point for each movement, -10 points for firing an arrow, +10 for killing a Wumpus, -1000 for dying, +1000 points for finding the gold.

[2] 5x5 or 10x10

2 Description of Algorithms Implemented

2.1 Model-Based Reactive Agent

The model-based reactive agent is simple compared to the KB-Agent. Below contains the main loop of the reactive agent.

Algorithm 1 Model-Based Reactive Agent

```
1: while allowed moves left do
2:     procedure ReactiveAgent(actualWorld)
3:         perceivedWorld = what the agent knows about the next state
4:         model = what will happen when an agent makes a certain decision
5:         rules = a set of simple condition-action rules
6:         action = most recent action
7:
8:         state = updateState(perceivedWorld, action, sensors, model)
9:         action = Action(state, rules)
10:        return action
```

The agent is very limited in ability due to only being aware of its previous move and can only guess the state of rooms directly adjacent to it. Since the reactive agent can't look as far ahead as the KB-agent, a max moves variable is used to prevent infinite loops[3]. Naturally, this means the reactive will die...a lot.

2.2 Knowledge-Based Agent

The KB-agent is written in Java and utilizes an object oriented representation of conjunctive normal form (CNF) grammar[3] along with unification and resolution algorithms to provide answers to queries from the agent such as *'is room (3,4) safe?'* or *'should I fire an arrow, and at which cell?'*.

2.2.1 Inference Engine Design

To achieve inference through first-order logic, the KB-agent uses a inference powered knowledge base[4] which has a static list of first-order logic, universal axioms and a dynamic list of atomic sentences provided by the agent as he explores and perceives the world. Unification is used to transform the variable, universal axioms into constants relevant to the current query. Resolution takes the unified knowledge base and a query and

[3]In this implementation, the composite patter is consistently used to create a CNF representation of literals, clauses, operators, and sentences.

[4]Henceforth knowledge base and inference engine refer to the same thing.

provides a value of true or false. The inference engine also has logic built into it to decide the best action to take[5] given the current known state of the world.

As the world size increases, the number of agent-perceived atomic sentences increases exponentially. The resolution algorithm uses pairwise comparison of each atomic sentence which will result in infeasibly time-expensive procedures as the knowledge base becomes large. To deal with this, the knowledge base uses a first order logic system and contains contextual mappings of query requests mapped to a subset of axioms and atoms in the knowledge base. When a query is made, irrelevant sentences are not included in the knowledge base.

Atoms and Predicates The most basic composed logic object used is an Atom which consists of a negation value, a predicate, and a term.[6] The predicates are predefined and are as follows:

- $BLOCKED$
- $OBST$ (obstructed)
- $SHINY$
- EXP (explored)
- PIT
- $SMELLY$
- $HASGOLD$
- $SAFE$
- $WUMPUS$

Universal Axiomatic Sentences The most important axioms for this problem are sentences describing when a room has a Wumpus in it, when a room has a pit, and when a room is safe. These axioms are described in equations 1 through 3. The rest of the inferences and instructions are extensions of these three primary sentences. In the complimentary code, all axioms are written in conjunctive normal form, but for ease of reading they are written here in their equivalent implication form.

If all adjacent rooms[7] to a query square are explored and determined smelly, then the query room has a Wumpus in it.

$$\begin{aligned}&(SMELLY(Room_{x-1,y}) \wedge EXP(Room_{x-1,y})) \wedge \\ &(SMELLY(Room_{x,y+1}) \wedge EXP(Room_{x,y+1})) \wedge \\ &(SMELLY(Room_{x+1,y}) \wedge EXP(Room_{x+1,y})) \wedge \\ &(SMELLY(Room_{x,y+1}) \wedge EXP(Room_{x,y-1})) \\ &\qquad \Rightarrow WUMPUS(Room_{x,y})\end{aligned} \tag{1}$$

In similar fashion, if all adjacent rooms are explored and determined windy, then the

[5]Move to a known safe cell, fire an arrow, pick up the gold, or prematurely exit the dungeon

[6]The term is used during unification and is either a variable room or a unified reference to a specific room.

[7]Note that in the code implementation, edge cases also had to be included in the axioms. In other words, if the query room was Room [0,0], then conjunctives for Room [-1,0] and Room [0,-1] are not a part of the CNF

query room is a pit.

$$\begin{aligned}&(WINDY(Room_{x-1,y}) \wedge EXP(Room_{x-1,y})) \wedge \\ &(WINDY(Room_{x,y+1}) \wedge EXP(Room_{x,y+1})) \wedge \\ &(WINDY(Room_{x+1,y}) \wedge EXP(Room_{x+1,y})) \wedge \\ &(WINDY(Room_{x,y-1}) \wedge EXP(Room_{x,y-1})) \\ &\qquad \Rightarrow PIT(Room_{x,y})\end{aligned} \tag{2}$$

Finally, a room is known to be safe if any of the surrounding rooms have been explored and have neither a smell or wind attribute.

$$\begin{aligned}&(\neg SMELLY(Room_{x-1,y}) \wedge \neg WINDY(Room_{x-1,y}) \wedge EXP(Room_{x-1,y})) \vee \\ &(\neg SMELLY(Room_{x,y+1}) \wedge \neg WINDY(Room_{x,y+1}) \wedge EXP(Room_{x,y+1})) \vee \\ &(\neg SMELLY(Room_{x+1,y}) \wedge \neg WINDY(Room_{x+1,y}) \wedge EXP(Room_{x+1,y})) \vee \\ &(\neg SMELLY(Room_{x,y-1}) \wedge \neg WINDY(Room_{x,y-1}) \wedge EXP(Room_{x,y-1})) \\ &\qquad \Rightarrow PIT(Room_{x,y})\end{aligned} \tag{3}$$

The rest of the axioms are straight forward such as identifying if a room has gold in it:

$$SHINY(Room_{x,y}) \Rightarrow HASGOLD(Room_{x,y}) \tag{4}$$

Unification Before running resolution on a given query, the knowledge base axioms need to be unified. The unification shown in algorithm 2 follows the concepts described by Russell and Norvig [2], but takes advantage of the domain of this problem by only looking to transform atoms of type *variable* into atoms of type *constant* with allowed predicates already defined.

Algorithm 2 Unification

```
1: procedure UNIFY(kb, query)
2:     Pre: kb is the knowledge base
3:     Pre: query is an atomic sentence
4:     for each sentence in kb do
5:         for each atom do
6:             if SUBST(θ, atom) = SUBST(θ, query) then
7:                 UNIFY(atom, query)
```

Resolution Resolution follows the resolution inference rules outlined in Russell and Norvig [2]. It utilizes the fact that all statements in the knowledge base are already in conjunctive normal form and applies a proof by contradiction to assert the truth of a query. After all terms have been unified, split any conjunctions into individual sentences.

Next, do a pairwise comparison of each sentence in the knowledge base. If a new resolvent clause is generated that has not been generated before, add it to the local knowledge base. If an empty resolvent clause is generated, the proof by contradiction succeeded. Assert the query is true. If no unique resolvent clauses are generated after all pairwise comparisons, assert the query is false. The algorithm is shown in algorithm 3.

Algorithm 3 Resolution

```
1:  procedure RESOLVE(kb, query)
2:      Pre: the knowledge base, kb, has been unified with the query
3:      kb ← SPLITCONJUNCTIONS(kb)
4:      repeat
5:          for all pairwise comparisons (a, b) sentences in kb do
6:              r ← (a − resolvent_atom) ∪ (b − resolvent_atom)
7:              if r ∉ a ∨ b then
8:                  if r = ∅ then
9:                      query is True
10:                 else
11:                     kb = kb ∪ r
12:         if kb was not updated over the loop then
13:             query is False
14:     until query is true or false
```

Due to the repeated pairwise comparisons of the knowledge base, it is vital to ensure the local knowledge base only contains the necessary sentences relevant to the query as mentioned at the beginning of section 2.2.1.[8]

Request Action The agent moves through the dungeon by perceiving the room it is currently in, updating the knowledge base with the perceptions, and requesting an action to preform to the knowledge base. A frontier of unexplored cells is tracked throughout the scenario. The REQUESTACTION command follows a simple order-of-priority method of approach as follows:

1. If the agent's current room is shiny, pick up the gold and exit the simulation.
2. Move to the closest room in the frontier list that can be asserted as safe.
3. If there are no safe rooms to move to, move to a safe room adjacent to where a Wumpus is known to exist and shoot it.
4. If there are still no safe rooms, no guaranteed safe move can occur, so exit the simulation.

[8] Alternately, a more optimized algorithm should be used such as backward chaining.

3 Experimental Approach

World sizes from 5 to 25 will be run with a step increase of 5. The probability of placing a pit was set to 0.10, Wumpus to 0.05, and obstacle to 0.05. There will never been more than 1 danger in any given square.

Though the contents of most cells will be left entirely up to the probability function, the cells immediately above and to the right of the starting cell will be left empty so that at least one safe move will be guaranteed.

The agent begins to search for the gold and tracks each time it makes a decision, when it succeeded at finding the gold, when it kills a Wumpus, when it falls down a pit, when it dies, when it explores a new room, and its score. The score metric starts at zero, subtracts one point for each action made[9], subtracts ten points for firing an arrow, adds ten points for shooting an arrow, subtracts 1,000 points for dying, and adds 1,000 points for finding the gold.

Because one agent utilizes an inference engine and the other does not, the way decisions are made is notably different. Thus we look for the rate of change in average decisions made as the world size increases.

To generate statistically significant data, we run each map size for each agent 20 times and take the average result of the 20 sample runs. This data is displayed in section 4. Detailed data about the individual results of each run is provided in appendix A.

4 Experiment Results

The table following tables contain the average of 20 runs per map size. RA stands for Reactive Agent and KB stands for the Knowledge-Based Agent.

Agent	*Avg. Decisions*	*Success rate*	*Wumpi Killed*	*Pit Falls*	*Killed*	*Rooms Explored*	*Score*
			Map size 5x5				
RA	-	-	-	-	-	-	-
KB	8,569.70	0.75	0.20	0.00	0.00	12.20	722.20
			Map size 10x10				
RA	849.95	0.15	0.00	0.30	11.45	-	-2049.95
KB	53,311.20	0.70	1.80	0.00	0.00	44.90	607.00
			Map size 15x15				
RA	899.00	0.10	0.00	0.55	12.60	-	-2649.00
KB	92,312.40	0.80	3.45	0.00	0.00	100.00	609.75

[9]This includes turning clockwise and counterclockwise and moving from one room to another

			Map size 20x20				
RA	998.10	0.00	0.00	0.35	13.20	-	-2898.10
KB	95,285.25	0.65	7.60	0.00	0.00	177.50	396.80
			Map size 25x25				
RA	949.50	0.05	0.00	0.35	12.15	-	-2299.5
KB	82,260.25	0.65	8.95	0.00	0.00	195.90	387.95

5 Algorithm Behavior

5.1 Model-Based Reactive Agent

The reactive agent performed poorly, but a bit better than expected. Interestingly, the main variability lied in the decisions made and the rate at which the gold was found. Unfortunately, the reactive agent was unable to ever kill any wumpi. The rate that the agent experienced pit falls, was killed by a wumpus, and score mostly increased as the map size increased. The exceptions were pit falls for map size 15x15, score for 25x25, and being killed by wumpus for 25x25. The agent likely got lucky with an easy map. With more runs, the rate may have followed the decreasing pattern for gold and score while following an increasing pattern for everything else.

5.2 Knowledge-Based Agent

The hypothesis was primarily interested if the KB-Agent would have exponentially increasing decisions made as the world size increases. Figure 1 shows this is not the case. Although there are not enough x-axis plots to extrapolate the data to a best-fit curve, the results imply that the rate of growth of decisions-made as the world size increases is not exponential, and may even be zero.

This is due to the advantage first order axioms provide and the context mapping provided by the inference engine. The number of atomic sentences grows only in constant time with the number of rooms.

The score does decrease as world size increases. This is likely due to a higher potential of rooms with overlapped smells and winds causing the agent to struggle to find scenarios where a room could be determined not-pit or not-Wumpus due to all adjacent rooms having smells or winds.

Also the natural result of a larger frontier will cause the agent to do a lot more path traversal and turns which lowers the points earned.

5.3 Reactive vs. Knowledge-Based Agent Behavior

An unexpected result is that both the reactive agent and the KB-Agent displayed a similar rate of change in average decisions made as the map size increased.

Figure 1

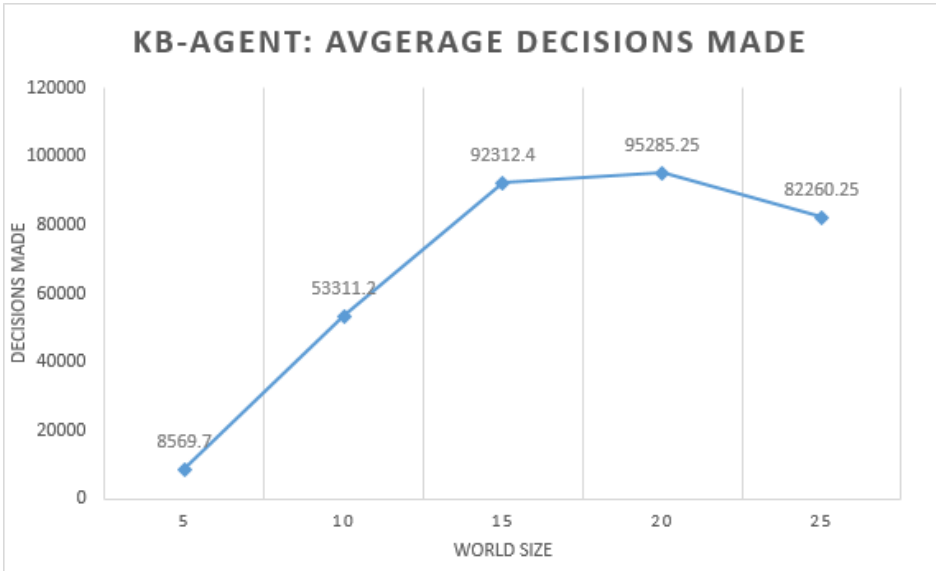
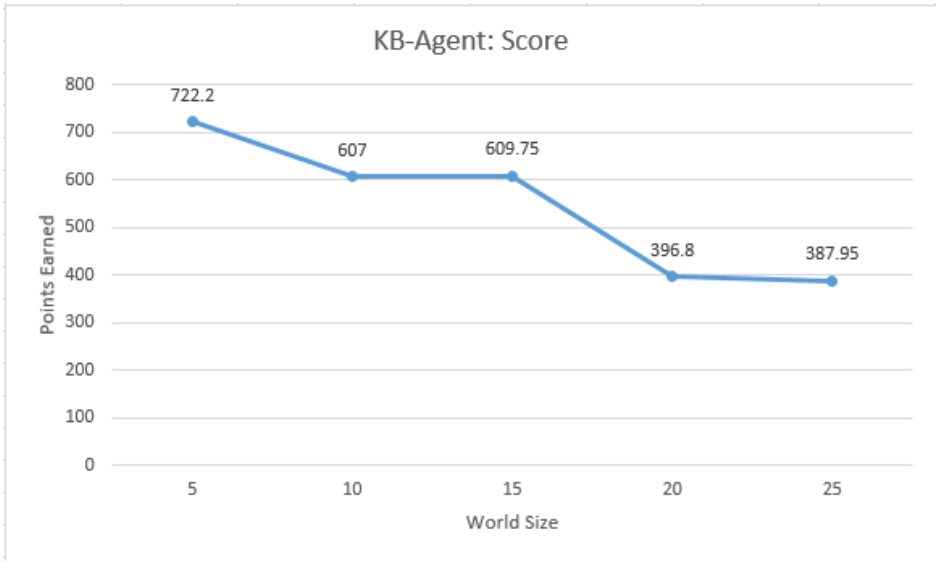


Figure 2



The fact that the inference engine keeps the KB-Agent safe from ever dying is an obvious advantage and is revealed in the difference between the average score of the two agents. The KB-Agent also flexed its logical superiority in being able to identify and kill the Wumpi with arrows reliably. Based on the difference in success rate and rooms explored, the ability to infer state and make decisions based off of the inference clearly made the KB-Agent the superior explorer.

6 Summary

As expected, the reactive agent did not perform very well on average over the range of world sizes used in testing. The lack of reasoning about cells that were outside of the agent's current percept range severely inhibited it from avoiding dangerous cells and successfully finding the gold. Even on smaller world sizes such as the initial 5x5 grid, the reactive agent failed to reach the goal state the majority of the time and was repeatedly killed, giving it little chance to finish with a positive score.

By using the reactive agent as a benchmark for performance improvement, it was possible to determine whether or not the additions to the knowledge-based agent were increasing or decreasing performance. With the exception of decisions made, which is due to the far greater complexity of the knowledge-based approach, every measured value was better for the knowledge-based agent than for the reactive agent. Perhaps one of the most telling statistics for the improvements over the reactive version is the fact that the knowledge-based agent did not die once for all iterations over all world sizes, showing that it was able to very successfully determine which unknown cells were more dangerous than others. Overall, the results for the knowledge-based agent show a great improvement over the much simpler reactive agent.

References

[1] Ahlam Ansari et al. *An Optimized Hybrid Approach For Path Finding*. International Journal in Foundations of Computer Science and Technology, 2015.

[2] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Pearson, 2009.

[3] Shichao Zhang and Ray Jarvis. *AI 2005: Advances in Artificial Intelligence*. Springer, 2005.

A Detailed Run Data

KB-Agent - Map Size: 10x10

Run #:	# Decisions Made	Gold Found?	# Wumpi Killed	# Times Fell Into Pit	# Times Killed	# Cells Explored	Score
1	13870	1	0	0	0	18	961
2	30041	1	0	0	0	27	938
3	17121	0	0	0	0	10	-32
4	98899	1	3	0	0	98	797
5	74683	1	3	0	0	65	853
6	6110	0	0	0	0	5	-16
7	88583	1	8	0	0	73	912
8	98820	1	5	0	0	101	785
9	67218	1	0	0	0	60	838
10	17052	0	0	0	0	10	-28
11	109761	1	5	0	0	77	901
12	109075	1	8	0	0	94	861
13	33121	0	0	0	0	18	-51
14	109560	1	0	0	0	72	798
15	17609	0	0	0	0	14	-38
16	6014	0	0	0	0	7	-23
17	129061	1	4	0	0	85	828
18	960	1	0	0	0	4	994
19	5795	1	0	0	0	16	972
20	32871	1	0	0	0	44	890
Avg:	53311.2	0.7	1.8	0	0	44.9	607

KB-Agent - Map Size: 15x15

Run #:	# Decisions Made	Gold Found?	# Wumpi Killed	# Times Fell Into Pit	# Times Killed	# Cells Explored	Score
1	13419	0	0	0	0	11	-34
2	97527	1	8	0	0	186	725
3	68311	1	0	0	0	42	901
4	8567	0	0	0	0	9	-27
5	99006	1	0	0	0	78	808
6	41450	0	0	0	0	14	-40
7	180642	1	0	0	0	120	749
8	94712	1	11	0	0	93	928
9	2774	1	0	0	0	10	986
10	182526	1	9	0	0	138	746
11	121254	1	10	0	0	180	690
12	164077	1	0	0	0	151	654
13	71173	1	0	0	0	76	840
14	104179	1	0	0	0	91	764
15	31803	1	0	0	0	24	946
16	6014	0	0	0	0	7	-23
17	60866	1	8	0	0	218	662
18	192015	1	9	0	0	190	636
19	174272	1	5	0	0	207	500
20	131661	1	9	0	0	155	784
Avg:	92312.4	0.8	3.45	0	0	100	609.75

KB-Agent - Map Size: 20x20

Run #:	# Decisions Made	Gold Found?	# Wumpi Killed	# Times Fell Into Pit	# Times Killed	# Cells Explored	Score
1	226790	1	15	0	0	351	500
2	73895	1	11	0	0	371	503
3	83343	1	17	0	0	173	841
4	46027	1	0	0	0	57	893
5	30259	0	0	0	0	10	-26
6	224075	1	0	0	0	181	625
7	4818	0	0	0	0	5	-16
8	226804	1	16	0	0	324	555
9	68034	0	0	0	0	26	-99
10	29376	0	0	0	0	14	-41
11	125152	1	13	0	0	383	484
12	32027	1	13	0	0	40	1075
13	122103	1	17	0	0	411	359
14	11370	0	0	0	0	8	-25
15	144680	1	15	0	0	373	495
16	6110	0	0	0	0	5	-16
17	102528	1	13	0	0	319	600
18	54333	1	13	0	0	149	877
19	22491	0	0	0	0	20	-68
20	271490	1	9	0	0	330	420
Avg:	95285.25	0.65	7.6	0	0	177.5	396.8

KB-Agent - Map Size: 25x25

Run #:	# Decisions Made	Gold Found?	# Wumpi Killed	# Times Fell Into Pit	# Times Killed	# Cells Explored	Score
1	87467	1	18	0	0	278	724
2	121980	1	0	0	0	101	793
3	20948	1	0	0	0	14	957
4	16711	0	0	0	0	11	-34
5	60319	1	19	0	0	490	442
6	112040	1	18	0	0	288	612
7	69434	0	0	0	0	35	-90
8	72424	1	17	0	0	385	595
9	278034	1	18	0	0	476	278
10	2842	0	0	0	0	5	-16
11	174681	1	13	0	0	229	634
12	28613	0	0	0	0	14	-38
13	19226	0	0	0	0	12	-34
14	55685	1	19	0	0	439	551
15	27401	0	0	0	0	16	-58
16	142431	1	20	0	0	458	528
17	6021	0	0	0	0	7	-24
18	279884	1	19	0	0	529	177
19	44296	1	18	0	0	99	823
20	24768	1	0	0	0	32	939
Avg:	82260.25	0.65	8.95	0	0	195.9	387.95