Wumpus World Report

Kayla Laufer
Spring 2024 Artificial Intelligence (CISC-6525-L01)
May 12, 2024

Overview:

This project focuses on developing a rational agent tasked with navigating and excelling in the Wumpus World game. The approach combines truth-table enumeration-based entailment (model checking) with reinforcement learning techniques. This project aims to significantly enhance the agent's decision-making abilities by utilizing a given simulation environment (wwsim.py) and a basic agent (wwagent.py) that initially makes random moves.

The initial objective was to develop an agent capable of sensing its environment and making informed decisions based on the perceived data. To achieve this, model checking through truth-table enumeration was employed to identify safe adjacent rooms. This method provided a solid foundation for the agent to make decisions based on direct sensory inputs from its immediate surroundings.

As the project progressed, probabilistic model checking was integrated to handle scenarios where decisions could not be made with absolute certainty. This allowed the agent to make calculated decisions based on the probability of room safety rather than relying solely on binary safe-unsafe assessments. This approach enhanced the agent's ability to navigate environments with higher levels of uncertainty.

The final phase of the project involved integrating planning with reinforcement learning. Instead of mere guesswork, probabilistic model checking served as the basis for decision-making, which was further refined through the application of temporal difference reinforcement learning techniques. Over 200 simulated episodes, the agent continuously learned and adapted its strategy to improve performance.

After extensive simulation runs, a comparative analysis was conducted between different agents:

- The original random decision-maker
- The probabilistic model checker
- The reinforcement learning-enhanced agent

This analysis aimed to measure the growth in rational decision-making capabilities and overall effectiveness across different decision-making paradigms.

Project Description

The objective is to transform the provided basic agent into a rational one that not only survives but excels in the Wumpus World by finding gold and avoiding hazards like pits and the Wumpus itself. The agent must fulfill the following performance criteria:

1. Navigate through the Wumpus World to locate and retrieve gold, thereby winning the game.
2. Always opt for a 100% guaranteed safe move if available.
3. In the absence of a completely safe move, the agent should choose the relatively safest move based on the calculated probabilities of encountering a pit or a Wumpus.
4. Employ indirect model-based reinforcement learning (RL) to refine its decision-making process, enhancing its ability to select safer moves and overall performance in the game environment.

The agent's reasoning capabilities are to be built using truth-table enumeration, a form of model checking, to promptly and reliably determine the safety of a location based on available percepts. This model-checking approach is critical to the project and must be implemented explicitly without using alternative methods.

Version 1 – Model Checking

Model checking plays a pivotal role in enabling the Wumpus World agent to evaluate the safety of potential moves. This process involves constructing and validating models based on the agent's perceptions, which consist of both direct sensory input from the agent's current location and deductions made from historical data.

Map Maintenance and Perceptual Updates:

The agent maintains a perceptual map (self.map) of the Wumpus world, wherein each cell documents percepts like stench, breeze, and glitter. As the agent navigates the world, this map is continuously updated with fresh sensory data, enabling the construction of logical models that hypothesize what the world might look like beyond the immediate observations.

Model Construction and Deduction:

For each potential move, the agent predicts various configurations of the world. These models incorporate deductions based on the percept map—for instance, a breeze in a room implies the potential presence of a pit in adjacent rooms. The agent generates symbols representing potential hazards in each room and explores every combination of these symbols to map out all plausible world states.

Symbol Generation and Adjacency Considerations:

Symbols for potential hazards are generated for rooms adjacent to the agent's current position, facilitating a focused assessment of the immediate environment. It reduces the number of rooms needed for search by incorporating the knowledge of known safe rooms. This strategic limitation in symbol generation helps the agent prioritize its immediate surroundings while also leveraging known information to make broader logical inferences about where dangers like Wumpuses or pits might be located. It also helps to reduce the state space needed for search.

Recursive Model Checking:

The modelcheck method assesses the validity of each model through a recursive process that assigns truth values to each symbol. It employs a divide-and-conquer strategy, assigning truth values to symbols and validating the resultant world models against the agent's knowledge base. This method checks each model for consistency with known safe rooms and verifies that the logical implications of each model do not contradict the accumulated perceptual data.

Decision Making Based on Model Validity:

If a model consistently aligns with the known safe conditions and logical deductions, it is considered valid, and the safety of a target room (alpha) is confirmed by ensuring it neither implies a pit nor a Wumpus. The agent then chooses its move based on the safest path suggested by these models. In cases where no entirely safe model can be validated, the agent may opt to make cautious moves or rotations based on the least risky assumptions.

Challenges and Enhancements:

A significant challenge was encouraging the agent to explore more of the world rather than revisiting known safe rooms excessively. To address this, a 'visited' list tracks how often each room is entered, discouraging repetitive visits and prompting exploration. Moreover, the prioritization of unvisited rooms when determining the next alpha encourages exploration and ensures that safe, unvisited rooms are considered preferentially, enhancing the agent's efficiency in mapping and securing the entire environment.

Conclusion:

Model checking is instrumental in guiding the WWAgent through the Wumpus World by simulating various potential states and logically determining the safest routes. This capability is crucial for the agent's survival and success, particularly in scenarios where sensory information is incomplete or ambiguous, thus significantly enhancing its operational effectiveness.

Version 2 – Probabilistic Model Checking

Probabilistic model checking represents an enhancement to the basic model checking approach by incorporating probabilistic assessments into the decision-making process. While the original method evaluates the safety of moves strictly based on binary logical deductions, this enhanced version quantifies uncertainties, thus enabling the agent to make more informed decisions under conditions of ambiguity.

Integration of Probabilities:

In this approach, the agent not only determines whether a particular state or action is permissible based on the rules and percepts but also calculates the likelihood of each state leading to a safe outcome. This probability is derived from the number of models where a hypothesis (alpha) holds true against the total number of possible consistent models evaluated.

Model Evaluation with Probabilities:

For each potential action (alpha), the agent systematically constructs all conceivable models of the world that align with the accumulated sensory data and logical rules. Each model represents a unique configuration of pits, Wumpuses, and other percepts within the adjacent rooms:

- True Models Count: The agent counts how many of these models do not violate any known percepts or logical inferences (i.e., models that are consistent with the existing knowledge base).
- Alpha True Count: Within the subset of true models, the agent further counts how many models support the safety of the room in question (alpha). This involves checking models where alpha does not imply the presence of a pit or a Wumpus.

Probability Calculation:

The probability of safety for a room is calculated by dividing the number of models where alpha is safe by the total number of consistent models. This ratio provides a quantifiable measure of confidence in the safety of moving into a room, enhancing the agent's ability to navigate risky or less-understood areas of the Wumpus World.

Decision Making:

The agent utilizes these probabilities to prioritize its movements. Rooms with higher safety probabilities are preferred.

- If a room is safe (100% probability), then the agent chooses to move into the safe room, assuming it is within the visited threshold.
- Comparative Selection: In situations where no room meets the absolute safety threshold, the agent may choose the room with the highest relative safety probability, thus managing risks in a controlled manner.

Enhancing Exploration:

Probabilistic model checking also aids in balancing exploration with safety. By quantifying the uncertainty of unexplored rooms, the agent can make calculated risks, potentially leading to greater rewards like discovering the gold or successfully mapping more of the world.

Conclusion:

The probabilistic model checking approach equips the agent with the capability to operate effectively in uncertain environments by providing a statistical basis for decision-making. This method not only improves the agent's adaptability and performance in dynamic settings but also enhances its overall strategic depth, allowing for more nuanced and calculated decisions based on the probabilistic outcomes of its actions.

Version 3 – Reinforcement Learning

In the third phase of the project, the agent incorporates reinforcement learning techniques, specifically using SARSA (State-Action-Reward-State-Action) temporal difference learning to enhance decision-making capabilities. This approach is integral to evolving the agent from relying solely on static model checking to an adaptive strategy based on learning from interactions with the environment.

The implementation starts by initializing a Q-table, a four-dimensional array managed within the `wwsim.py` as a global variable. This Q-table is pivotal as it stores and updates the value estimates for every state-action pair encountered during the simulation. The agent uses these values to make informed decisions about the best actions given a particular state.

Key parameters for the reinforcement learning algorithm include:

- **Epsilon**: Starts at 0.1, facilitating a balance between exploration (trying new actions) and exploitation (using known good actions), gradually increasing to 0.25 to allow more exploration as the agent gains confidence in its learned values.
- **Learning Rate (alpha)**: Set at 0.3, this parameter controls the rate at which new information overrides old information, allowing the agent to adjust to changes and new strategies it learns.
- **Discount Factor (gamma)**: At 0.8, it determines the importance of future rewards, enabling the agent to evaluate the long-term benefit of its actions.

During the simulation, actions are chosen using an epsilon-greedy strategy, where the agent selects either the probabilistic model checking choice (planning), or the best-known action (learning) based on the Q-table. The probabilistic model checking still plays a role, especially in initial phases or less certain situations, by providing a 90% preference at the start, which ensures that decisions are grounded in the safest and most logical assessments of the environment.

The Q-table is updated after every action based on the SARSA update rule, which adjusts the values based on the reward received and the estimated value of the next state-action pair. This continuous update allows the agent to learn and refine its strategy within a single episode without needing to complete the entire episode to adjust its approach.

Rewards are structured to reinforce beneficial actions sharply: small negative rewards for each step encourage efficiency, a significant penalty for "dying" (falling into a pit or being caught by the Wumpus), and a substantial reward for retrieving the gold. This structure helps guide the agent towards successful strategies while avoiding dangerous mistakes.

The process is iterative, with epsilon gradually increasing over 200 episodes to encourage a shift from safer, model-based decisions to more adventurous explorations that test the limits of the learned strategies. After each run, all settings and learned values are serialized into a JSON file, ensuring that progress is saved and can be built upon in subsequent simulations.

This integration of SARSA and planning via model checking represents a sophisticated approach in artificial intelligence, allowing the agent not only to learn from its environment but also to apply logical deductions to navigate and succeed in the complex and hazardous Wumpus World. This learning mechanism is continually active, ensuring that with every new game, the agent evolves, adapting its strategies based on cumulative experiences.

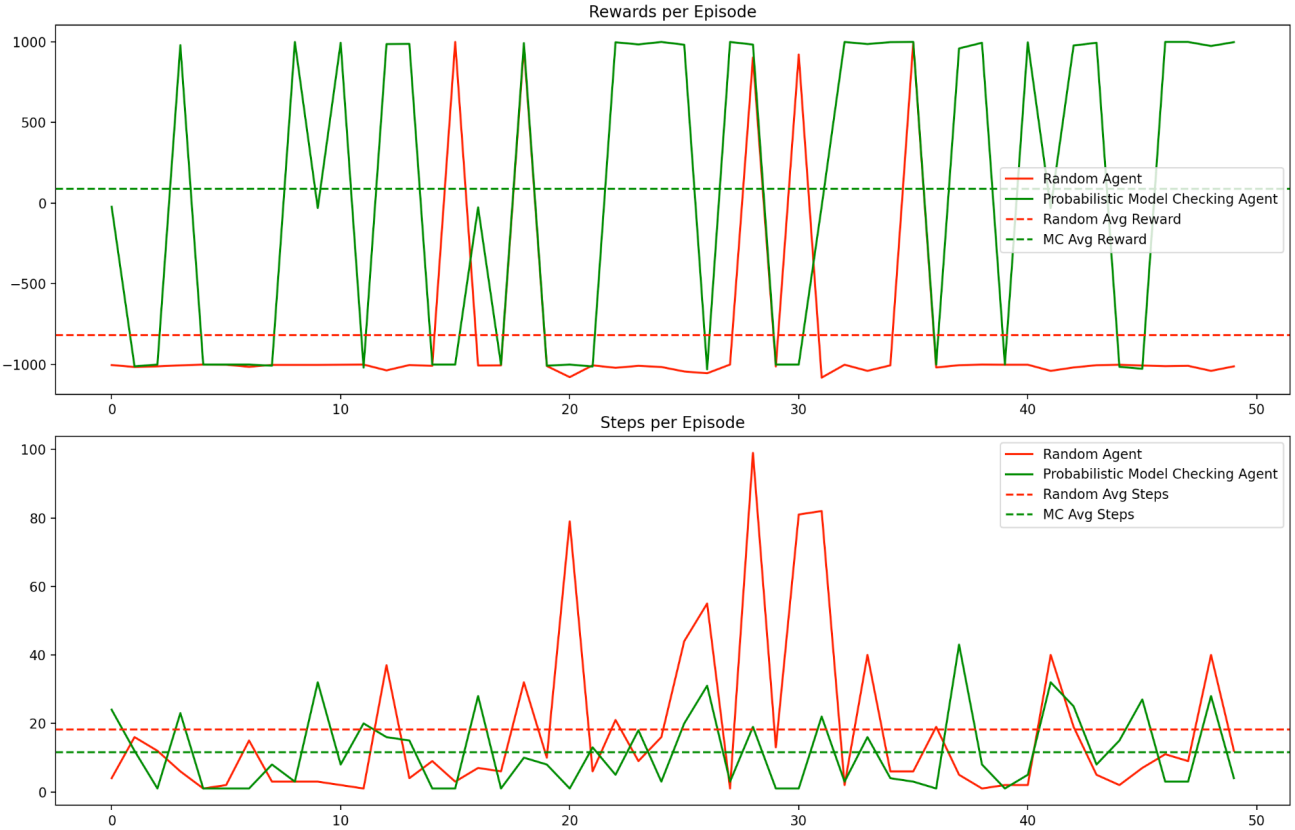
Experiments

Each agent was subjected to 50 simulation runs in the Wumpus World environment, providing a robust dataset for analysis.

Key Metrics Collected:

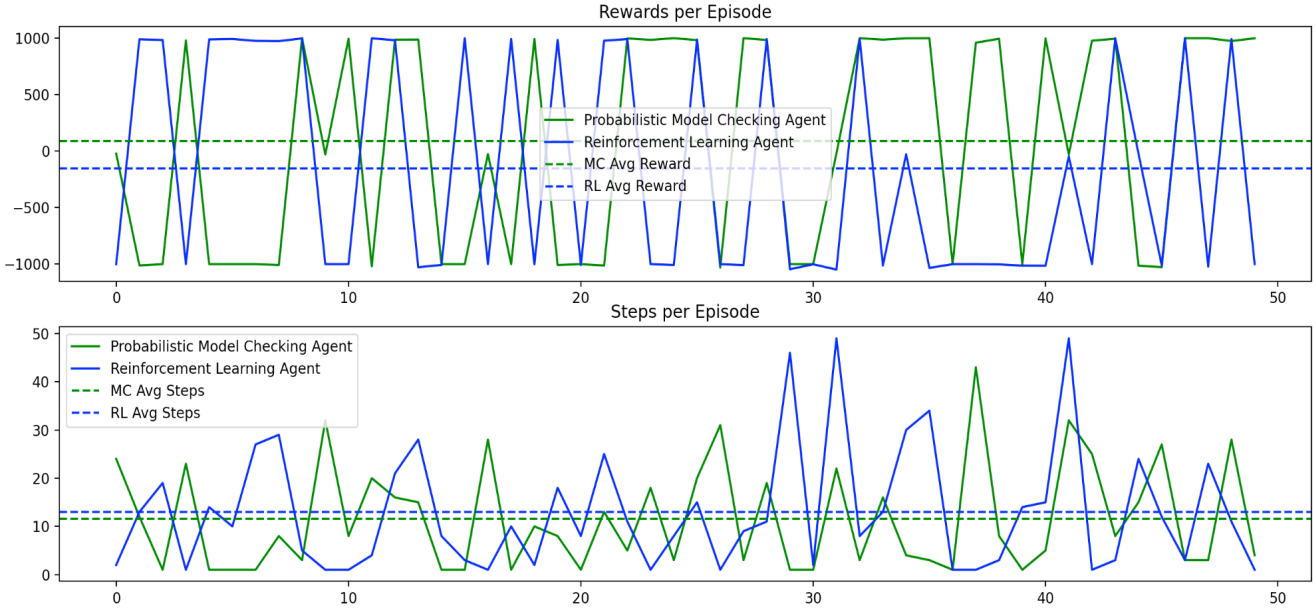
- **Rewards per Episode:** Total rewards accumulated by the agent during each simulation, reflecting its overall performance and effectiveness in navigating the environment and achieving objectives.
- **Steps per Episode:** Total number of moves (steps) taken by the agent to complete an episode, which serves as a measure of efficiency. Fewer steps indicate a more efficient pathfinding strategy by the agent.
- **Success:** The number of times the agent successfully retrieved the gold. This metric directly reflects the survival and goal-completion capabilities of each agent.

Random Agent vs Probabilistic Model Checking Agent:

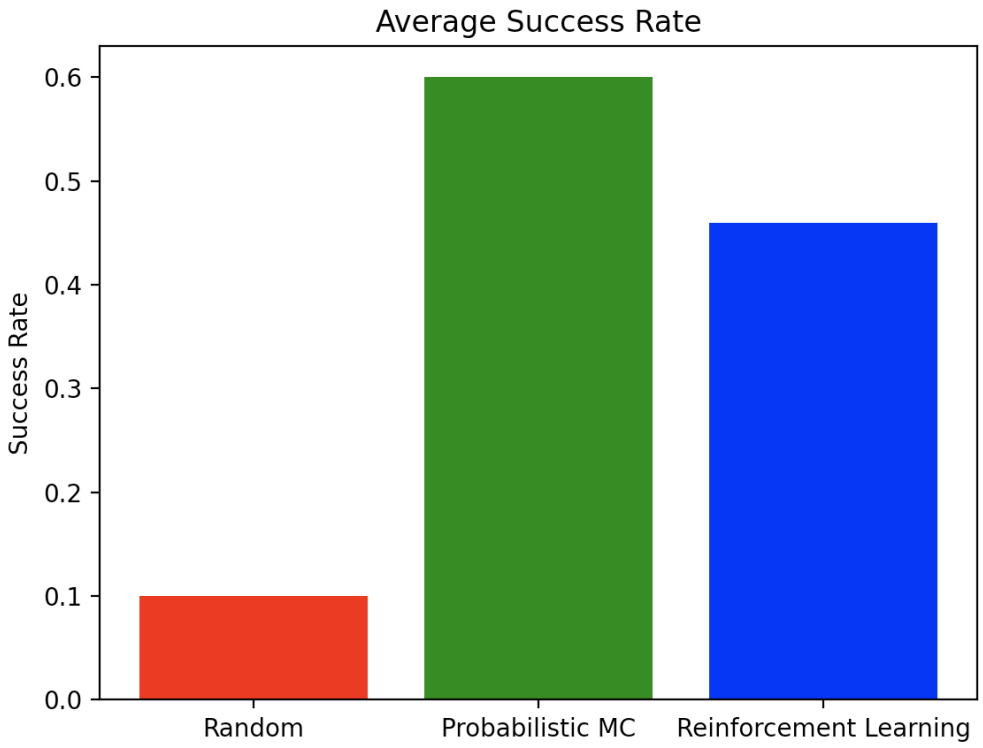


As you can see, the probabilistic model checking agent outperformed the random agent as expected. It has a much higher reward average with less steps per episode.

Probabilistic Model Checking Agent vs Reinforcement Learning Agent



We can see that the reinforcement learning agent performs slightly worse than the probabilistic model checking agent. However, their performance is comparable and many factors can contribute to this trend.




```
Random Agent:
Average Reward: -818.1
Average Steps: 18.2
Average Success: 0.1
MC Agent:
Average Reward: 89.0
Average Steps: 11.6
Average Success: 0.6
RL Agent:
Average Reward: -152.52
Average Steps: 12.98
Average Success: 0.46
```

As expected, both the probabilistic and reinforcement learning agents greatly outperformed the random agent.

Experimental Setup and Results:

Random vs. Probabilistic Model Checking:

- Random Agent: Operates without a strategic plan, making decisions based solely on random choice. This approach generally results in poor performance, with low efficiency and high risk.
- Probabilistic Model Checking Agent: Utilizes a systematic approach to decision-making, incorporating model checking techniques that assess the safety of potential moves based on the known risks of pits and Wumpuses in adjacent squares. This agent is expected to perform significantly better than the random agent, with fewer steps and higher success rates due to its ability to avoid known hazards.

Results: As anticipated, the probabilistic model checking agent outperformed the random agent across all metrics. The probabilistic model checking agent's ability to make informed decisions based on environmental percepts led to a higher win rate and more efficient paths. Graphs comparing the rewards per episode and steps per episode clearly show the superiority of the probabilistic model checking approach over random decision-making.

Probabilistic Model Checking vs. Reinforcement Learning:

- Reinforcement Learning Agent: Implements a learning-based approach, using SARSA (State-Action-Reward-State-Action) to dynamically adapt its strategy based on the outcomes of its actions. This agent is expected to improve its performance over time, adjusting its strategy to maximize rewards and minimize risks based on past experiences.

Results: While the probabilistic model checking agent provides a strong baseline with its logical deductive capabilities, the reinforcement learning agent is designed to surpass this by learning from each interaction within the environment. Initial runs may show comparable performance to probabilistic model checking, but as the reinforcement learning agent learns, it should ideally demonstrate superior decision-making, leading to higher rewards and more consistent success. In this experiment, you can see that the reinforcement learning agent is performing suboptimally to the probabilistic model checking agent. This can be due to many factors such as needing more simulations to learn, adjusting reinforcement learning parameters, the agent choosing to explore more dangerous paths, or even the fairness of the game simulations. Even though the reinforcement learning agent did not outperform the probabilistic model checking agent, it still performed much better than the random agent.

Graphical Analysis:

- Graphs comparing the performances of the agents illustrate the progression in agent capabilities. These visualizations provide clear, empirical evidence supporting the efficacy of advanced decision-making and learning algorithms in autonomous agent navigation within hazardous environments.

Conclusion:
The experimental results strongly validate the hypothesis that both probabilistic model checking and reinforcement learning significantly enhance agent performance compared to random decision-making. Furthermore, the ongoing adaptation and learning exhibited by the reinforcement learning agent suggest a promising direction for future enhancements, potentially leading to even greater efficiencies and success rates in more complex or dynamically changing environments.

What to improve on

Throughout the development of this project, several enhancements were identified that could potentially elevate the performance and efficiency of the agent. One key area identified for future exploration involves the integration of known percepts into the symbols list used for model checking. While initial experiments in this direction showed promise, improving the accuracy of probability predictions, they also introduced computational overhead and sporadic errors in the logic handling:

- **Integration of Known Percepts:** Current implementation filters out known safe rooms from model checking to reduce computational load. However, incorporating all known percepts (not just safety indicators but also potential hazards like breezes and stenches) could provide a richer, more nuanced model of the environment. This could allow the agent to make more informed decisions based on a comprehensive understanding of the world. The challenge will be to manage the increased complexity without significant performance degradation.
- **Optimization and Error Handling:** The increase in computational time and the emergence of occasional errors with the integration of full percept data suggest a need for optimization. Future work could focus on refining the data structures and algorithms used for model generation and checking, perhaps by implementing more efficient logic processing techniques.

Further advancements could also come from exploring alternative reinforcement learning strategies and adjusting the parameters involved in the learning process:

- **Exploration of Q-learning:** While the current implementation uses SARSA for reinforcement learning, experimenting with Q-learning could provide insights into the benefits of off-policy learning in this context. Q-learning, which decouples the policy being evaluated from the policy being improved, might lead to better long-term decision-making under certain game configurations.
- **Monte Carlo Reinforcement Learning:** Another promising area is the application of Monte Carlo methods for reinforcement learning, which could offer advantages in environments with high uncertainty and variability, such as the Wumpus World. These methods focus on learning directly from complete episodes, which might be particularly

useful when actions have long-term consequences not immediately apparent from initial percepts.

- **Tuning of RL Parameters:** The parameters governing the reinforcement learning process, such as the learning rate, discount factor, and exploration rate (epsilon), significantly influence agent behavior. Systematic experimentation with these parameters could help identify optimal settings for balance between exploration and exploitation, ensuring that the agent not only learns effectively but also adapiles robustly to different challenges within the Wumpus World.
- **Performance Metrics and Evaluation:** To rigorously assess the impact of these improvements, it will be crucial to establish a set of performance metrics. These metrics should not only evaluate the agent's success rate but also consider its efficiency (e.g., number of steps taken, computational resources used) and reliability (e.g., consistency across multiple runs, resilience to changes in the game setup).

By addressing these areas, the next iterations of the project could significantly enhance the agent's capability to navigate the Wumpus World more effectively and reliably. The goal is to create an agent that not only performs well on average but also consistently makes intelligent decisions that can be adapted to a wide range of scenarios, thereby approaching the benchmark of human-like strategic thinking in game environments.

Conclusion

This project successfully developed a sophisticated rational agent capable of effectively navigating the challenging environment of the Wumpus World. By integrating model checking based on truth-table enumeration with advanced reinforcement learning techniques, the agent not only improved its decision-making capabilities but also adapted dynamically to complex scenarios. The use of probabilistic model checking enhanced the agent's performance in uncertain conditions by enabling informed decision-making, while reinforcement learning facilitated continuous adaptation and learning from the environment.

These achievements underscore the effectiveness of combining traditional artificial intelligence strategies with modern learning algorithms to enhance both the efficiency and strategic depth of autonomous systems. The success observed in the simulated Wumpus World environment suggests promising applications for these methods in real-world scenarios that require rapid and informed decision-making based on partial information.

Despite these successes, the project encountered challenges such as balancing exploration and exploitation and managing the computational demands of probabilistic model checking. These challenges were addressed through iterative refinement and testing of the algorithms, highlighting the importance of adaptability and continuous improvement in AI development.

Looking forward, the project will explore alternative reinforcement learning algorithms, such as Q-learning and Monte Carlo methods, to further enhance the agent's decision-making

processes. Additionally, more detailed integration of complex perceptual data into the model checking framework could provide a more nuanced analysis of environmental risks and safety.