Designing an Adaptive Agent for the Wumpus World

Cesar Augusto Pulido Cuervo - 20222020048

Cristian David Romero Gil - 20222020138

Universidad Distrital Francisco José de Caldas

Designing an Adaptive Agent for the Wumpus World

Introduction

Classic implementations of the Wumpus World rely heavily on static, rule-based agents that operate on binary percepts and follow deterministic decision policies, typically using propositional logic and fixed inference rules Temple University and Sciences (2023). These models struggle with generalization, adaptability, and performance in more complex or unpredictable configurations. This project is motivated by the need to explore how autonomous agents can evolve into more intelligent, learning-driven systems and now, with the environment elements randomized.

In this workshop we explore ways to refine the reward structure to improve the agent's decision-making process in uncertain environments. As the goal of the project is to design, model, and implement an adaptive agent capable of navigating and making intelligent decisions using principles from system dynamics and reinforcement learning. To achieve this, we begin by constructing a causal diagram that captures the interactions between perceptual inputs, environmental variables, and agent actions over discrete time steps besides showing a implicit feedback. About the framework chosen, recognizing the limitations of binary percepts, we extend the observation space by introducing a continuous "danger level" variable that provides a more nuanced representation of environmental threats. This, in turn, enhances the agent's ability to assess risk and make informed decisions.

System Dynamics Analysis

State representation of Wumpus World:

$$s_t = \langle x, y, b, s, g, a, w, t \rangle$$

Variables that define the system state:

- x: X coordinate
- y: Y coordinate
- b: Binary percept for breeze
- s: Binary percept for stench
- g: Binary percept for glitter
- a: Is arrow available?
- w: Is Wumpus alive?
- t: Time step

System Dynamics Model

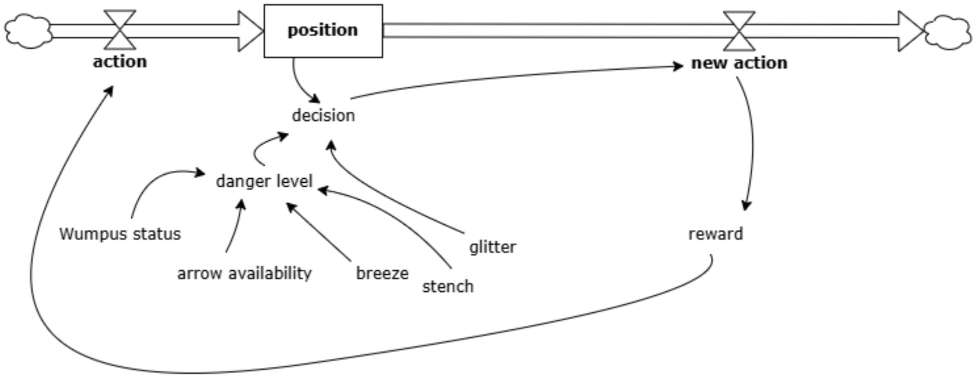

Figure 1. Causal diagram

Temporal Structure. The process occurs in time steps. At each time step:

1. The agent performs an action.
2. The position updates based on that action.
3. The new position leads to new percepts (stench, breeze, etc.).
4. A new decision is made, resulting in a new action.
5. The agent receives a reward, influencing future behavior.

Loop. This is a closed loop system; the result of each action influences the next cycle. The model implicitly supports learning or adaptive behavior.

Feedback Loop Refinement

Enhanced Control Mechanisms

To improve the agent's responsiveness to changes in the environment, we propose to introduce new control mechanisms through both additional sensors and more granular reward structures.

a. Additional Sensor: `danger_level`. A continuous variable named `danger_level` is introduced to represent the immediate threat level around the agent. As Waseem, Hameed, and Ramachandiran (2021) propose, a risk probability is useful for improving decision-making. It is computed based on the number of adjacent cells containing pits or the Wumpus. The value ranges from 0.0 (no risk) to 1.0 (maximum danger), offering a more nuanced input for decision-making than the binary percepts (breeze, stench).

This new sensor is added to the observation space and used as an input to the neural network policy (DQN). It allows the agent to distinguish between mildly dangerous and highly dangerous zones, facilitating smoother behavioral transitions.

b. Granular Reward Signals. The original reward structure was discrete and sparse. We introduced the following additional signals to increase feedback frequency and support learning:

Event	**Updated Reward**
Visiting a new, safe cell	+5
Revisiting the same cell	-0.5
Passing near danger (pit/Wumpus) but surviving	+10
Missed arrow shot (penalty reduced)	-25

The reward structure is refined to include low-magnitude, frequent signals that reinforce safe exploration, penalize redundancy, and encourage intelligent risk navigation. These modifications provide smoother learning dynamics and align with cybernetic principles of error correction, information feedback, and system adaptation.

Stability on Convergence Criteria

Bounded Behavior.

- The agent should not exhibit uncontrolled or erratic behavior such as infinite loops, excessive wandering, or oscillating between actions.
- Reinforcement learning models like Q-learning and DQN ensure that the agent eventually learns policies with bounded action choices, especially when experience replay and target networks are used to stabilize learning (Workshop 1).
- The agent incurs a step cost (reward -1) for every action. This discourages excessive movement and promotes efficient paths.

Convergent Learning Behavior. Use of reward curves, episode length, Q-value variance, and victory rate to assess convergence over time. Testing across multiple random seeds and environment layouts confirms whether learning consistently converges.

Iterative Design Outline

Project Plan Update

To implement dynamic behavior and adaptive control mechanisms, we plan to update the architecture and logic as follows:

Component	**Update**

Observation Space	Augmented with `danger_level` and a rolling memory of the last two perceptual vectors.
Reward Function	Enhanced with small positive/negative signals as described above.
Agent Network (DQN)	Input layer updated to accept expanded observation vectors.
Environment Logic	Tracks visited cells and calculates dynamic risk zones in real-time.
Training Code	Processes augmented states and updated rewards. Includes random seeds.

Testing Strategy

To evaluate these changes, we propose the following testing strategy:

Aspect	**Testing Strategy**
Random Seeds	Each training configuration is run across at least 10 different seeds to measure variance in outcomes.
Chaos Regimes	Tested with environments using varied layouts and perceptual noise to simulate dynamic volatility.
Scenario Variations	Multiple Wumpus/pit layouts used per run to assess policy generalization.
Perturbation Tests	Small changes in initial conditions to observe long-term trajectory divergence.
Metrics Tracked	Reward curves, victory rate, episode length, revisits, and Q-value stability.

This design and testing outline is intended to ensure that the Wumpus agent evolves from a static learner into a dynamic, chaos-aware decision-making system.

Framework Justification

This project integrates three key frameworks, selected not only for technical compatibility but also for their alignment with systems science principles explored throughout the course:

- Gymnasium: A modern RL toolkit that enables the implementation of discrete-time environments with well-defined observation and action spaces. Gymnasium's modularity mirrors the structure of dynamical systems, where agents perceive state variables, apply control inputs, and receive feedback. Terry et al. (2023) It supports experiments in control loops and systemic interaction over time, just as we model in cybernetic frameworks.
- PyTorch: PyTorch offers full control over the construction and training of neural networks used to approximate value functions. This directly supports adaptive agents capable of modifying behavior based on error signals—one of the core ideas in feedback regulation and learning. Its integration into the agent's policy controller reflects real-world systems that adapt over time through self-regulation, aligning with cybernetic principles.
- Pygame: Used for real-time visualization of the environment, Pygame enables observation of system behavior and feedback dynamics Pygame Community (2023). Visualization is not only helpful for debugging but also allows system dynamics to be interpreted as time-evolving graphical phenomena, a useful teaching aid and simulation technique aligned with dynamic systems modeling.

These frameworks collectively support the implementation, training, visualization, and theoretical grounding of a cybernetic agent—making them pedagogically appropriate and technically effective for this project.

References

Pygame Community. (2023). *Pygame: Python game development library.* `https://www.pygame.org`. (Version 2.5.0)

Temple University, D. o. C., & Sciences, I. (2023). *Cis 587: The wumpus world.* `https://cis.temple.edu/~giorgio/cis587/readings/wumpus.html`.

Terry, J., Chan, K., Tucker, G., Zholus, A., Gao, X. B., Agarwal, R., et al. (2023). *Gymnasium: A standard api for reinforcement learning environments.* `https://gymnasium.farama.org`. (Farama Foundation)

Waseem, C. T., Hameed, V. A., & Ramachandiran, C. R. (2021). Generating intuitive behaviour in artificial intelligence. *Journal of Applied Technology and Innovation*, *5*(2), 53–59.