Analysis and Design of an Autonomous Predator-Prey Adaptive Agent System

Juan Diego Grajales Castillo (20221020128)
Nicolas Castro Rivera (20221020055)

Abstract

This document presents the second version of the system analysis and design for an autonomous predator-prey adaptive agent simulation. Following feedback from Workshop 1, we introduce contextual background, clearly define system components, simplify reward structures, refine use cases with loop diagrams, expand stability analysis, and discuss suitable algorithms. Implementation details and specific technologies are deferred to later stages.

1 Introduction

The predator-prey agent system simulates interactions between two adaptive agents–predator and prey–within a discrete two-dimensional environment. By integrating cybernetic feedback loops and dynamical systems theory, our design aims to model realistic behaviors such as pursuit, evasion, and adaptive learning. This version emphasizes system context, component definitions, and design abstractions, deferring implementation details and technology choices to subsequent workshops.

2 System Functional Specifications

2.1 Sensors

Agents perceive their surroundings via proximity-limited sensors with a fixed grid radius of 5×5 cells:

- **Predator Sensor**: Detects prey, walls, and traps.
- **Prey Sensor**: Detects predator, walls, and traps.

2.2 Actuators

Actuators represent actions that modify the environment or agent state:

- **Movement Control**: Agents move one cell per time step in four cardinal directions.

2.3 Reward Functions

We simplify rewards to core objectives:

- **Predator**: $+10$ for capturing prey; -0.1 per step without capture.
- **Prey**: $+0.1$ per step evading capture; -5 if captured.

3 Use Cases and Feedback Loop Diagrams

Use cases describe actor interactions and refer to loop diagrams illustrating adaptive feedback.

1. **Predictive Pursuit**: Predator detects prey within range and navigates obstacles to capture; objective: optimize path strategy (see Figure 1).
2. **Reactive Evasion**: Prey detects predator proximity and selects evasion route avoiding traps; objective: maximize survival time (see Figure 2).
3. **Environment Exploration**: Agents explore the map to gather state information; objective: support learning of environment topology.

4 Dynamical Model

4.1 State Variables

Positions $x_p(t), x_h(t) \in \mathbb{R}^2$ and velocities $v_p(t), v_h(t)$ define the agent states. Distance is $d(t) = \|x_p(t) - x_h(t)\|$.

4.2 Discrete Motion Equations

The following table shows the discrete motion update equations used by the agents, along with detailed interpretations of each.

Table 1: Discrete motion equations with detailed interpretations.

Equation	Detailed Explanation
$\pi_p(t+1) = \pi_p(t) + \Delta t\, r_p(t)\, v_p(t)$	Updates predator position by adding the product of time-step Δt, predator speed $r_p(t)$, and control direction $v_p(t)$ (a unit vector indicating movement direction) to the current position $\pi_p(t)$.
$\pi_h(t+1) = \pi_h(t) + \Delta t\, r_h(t)\, u_h(t)$	Updates prey position using analogous terms: current position $\pi_h(t)$, time-step Δt, prey speed $r_h(t)$, and movement direction $u_h(t)$.
$r_p(t) = \nu_{\max} + \dfrac{\alpha}{1 + e^{-kd(t)}}$	Models predator speed adaptation: base-speed $\nu_{\max}$ plus a logistic function scaled by α, where k controls sensitivity to distance $d(t)$, yielding faster movement as prey distance increases.
$P_{\text{evade}}(t) = P_{\text{base}}\left(1 - e^{-\beta t}\right)$	Defines probability of successful evasion for the prey: starts at zero at $t = 0$ and asymptotically approaches P_{base} at rate determined by β, reflecting learning over time.

5 Stability Analysis

Stability ensures that agent interactions converge toward equilibrium or exhibit bounded oscillations. Due to the dynamic behavior of the agents, a balance is expected between the duration of a episode. The prey must strive to survive as long as possible, while the predator, focused on its objective, will end the session once captured.

5.1 Equilibrium Points

The agents' equilibrium will be determined by the agent's regulation system. Each agent has its own loop through which it will adjust its behavior, theoretically achieving stable behavior. The system can be sensitive to initial conditions, as the initial positions can have different results in a episode. An example is that a simulation episode can end with the predator capturing the prey because it was too close from the start of the simulation. Nonlinear factors can generate chaos in the system, and the results should be analyzed to see how the agents' behaviors stabilize.

5.2 Phase Diagrams

In figure 1 and 2 there is the Predator phase diagram and Prey phase diagram, respectively. The states of two agents will change over time and taking in count the information that obtain from proximity sensor.

For the predator to change its state if it detects the prey, it will increase its speed to catch it and choose a direction of movement.

For the prey can changes its state if detects the predator, will take an opposing direction to avoid the predator. Also, if detects a trap, will have a probability of being trapped by a trap or not, depending on the time elapsed in the episode.

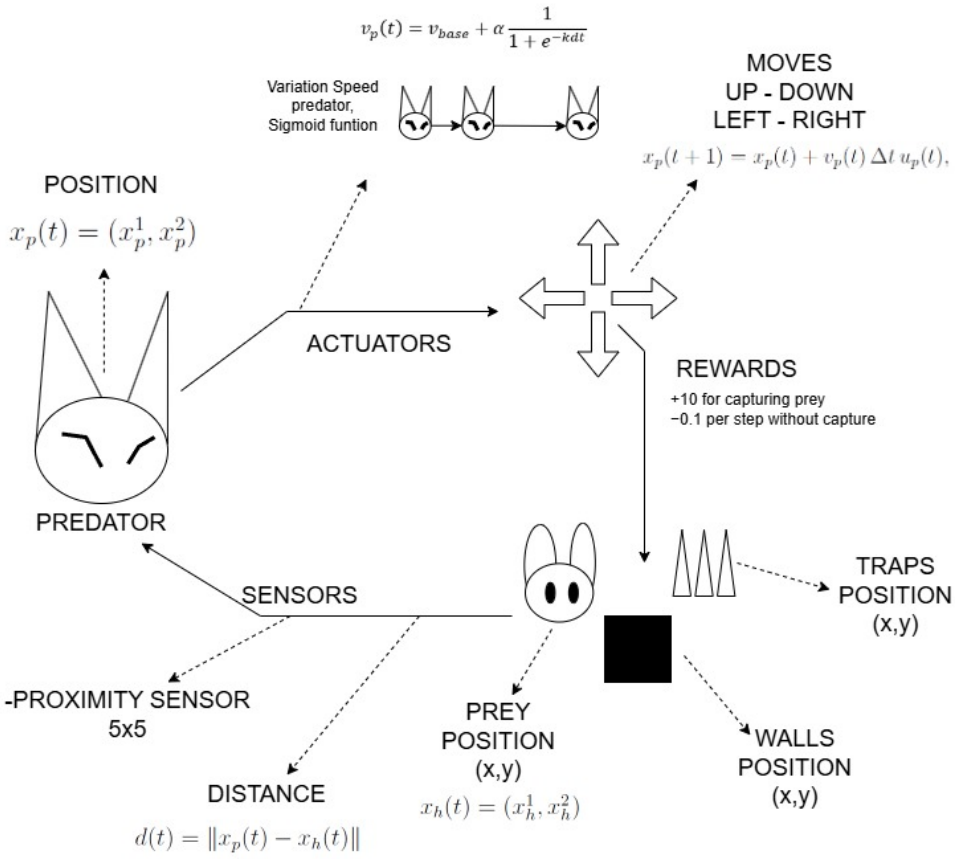

Figure 1: Predator phase diagram.

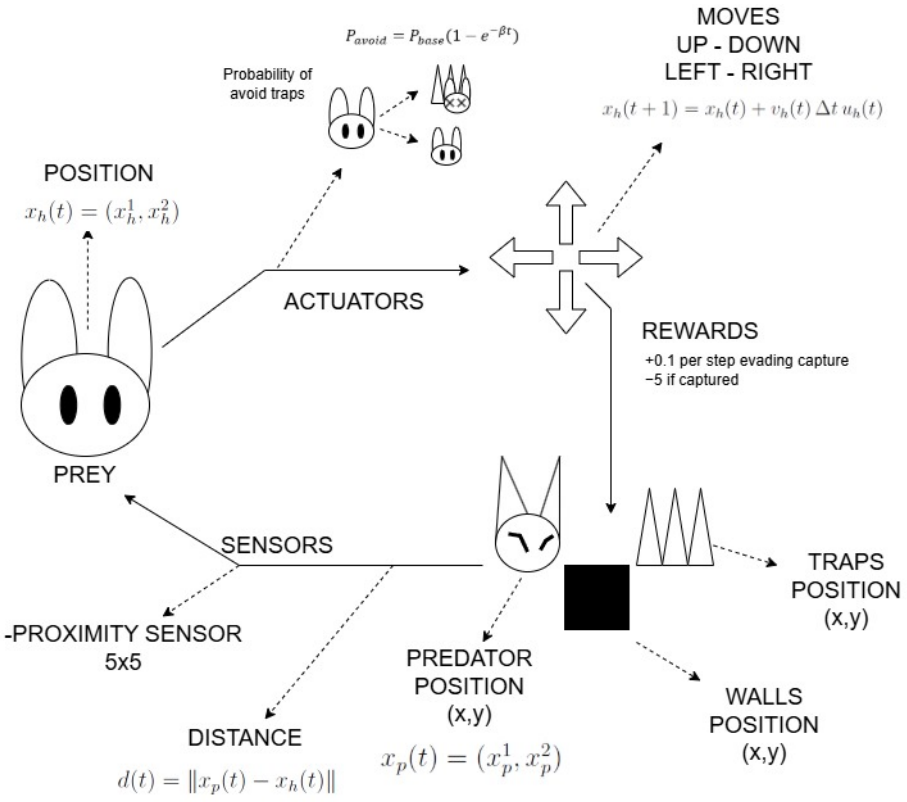

Figure 2: Prey phase diagram.

6 Suggested Algorithms

For future implementations, the following reinforcement learning algorithms are appropriate:

- **Q-Learning**: Suitable for discrete, fully observable state spaces, with straightforward value-function updates [1].
- **Deep Q-Network (DQN)**: Extends Q-Learning to high-dimensional inputs using deep neural networks, handling partial observability and large value tables [2].
- **SARSA**: An on-policy variant that mitigates action-value overestimation by learning from the current policy's transitions [3].
- **Actor-Critic**: Combines value-based updates with parameterized policies, offering stable convergence in continuous action spaces [4].

References

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529-533.

[3] Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. *Technical Report*, Cambridge University Engineering Department.

[4] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 229-256.