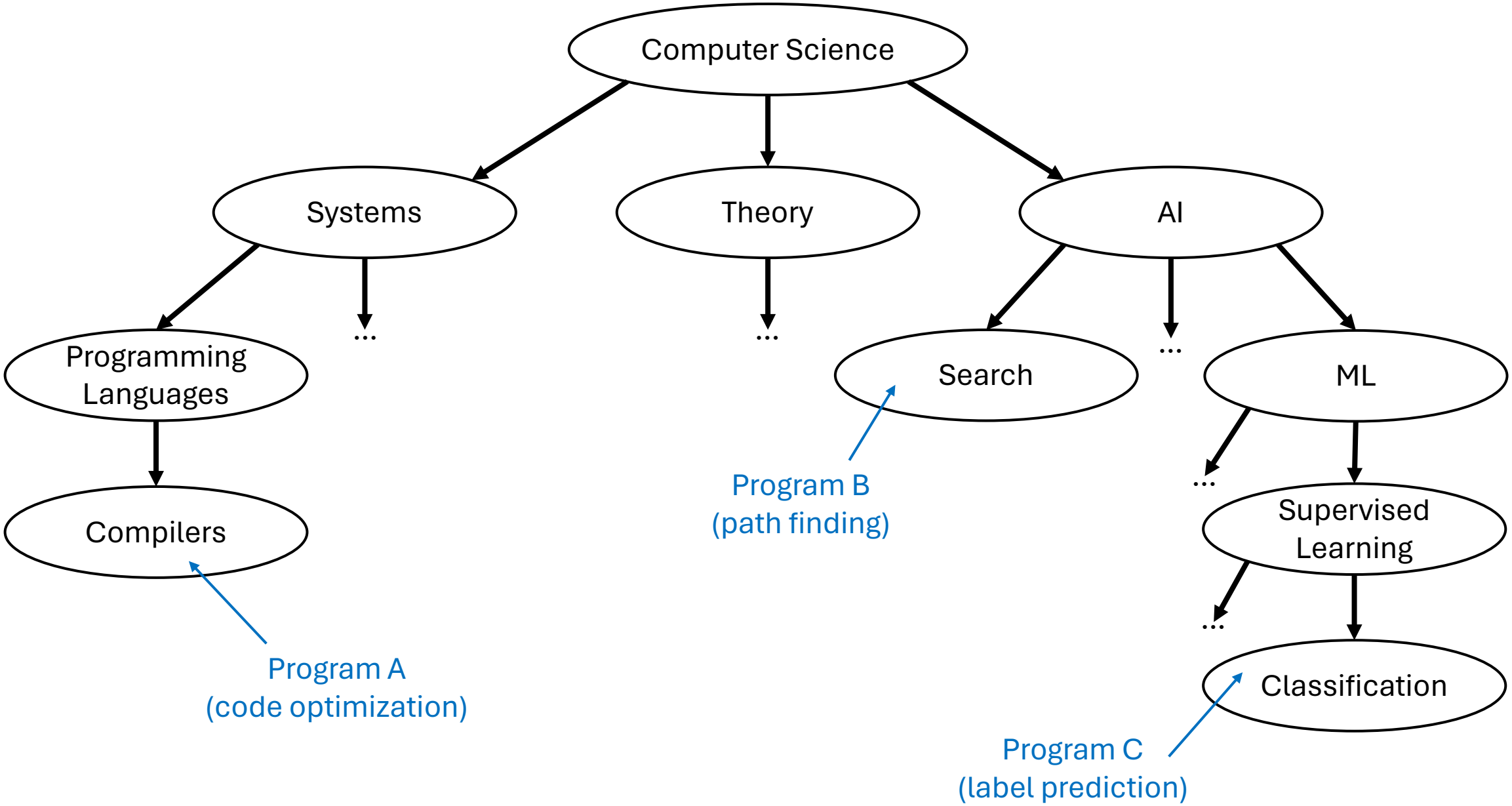
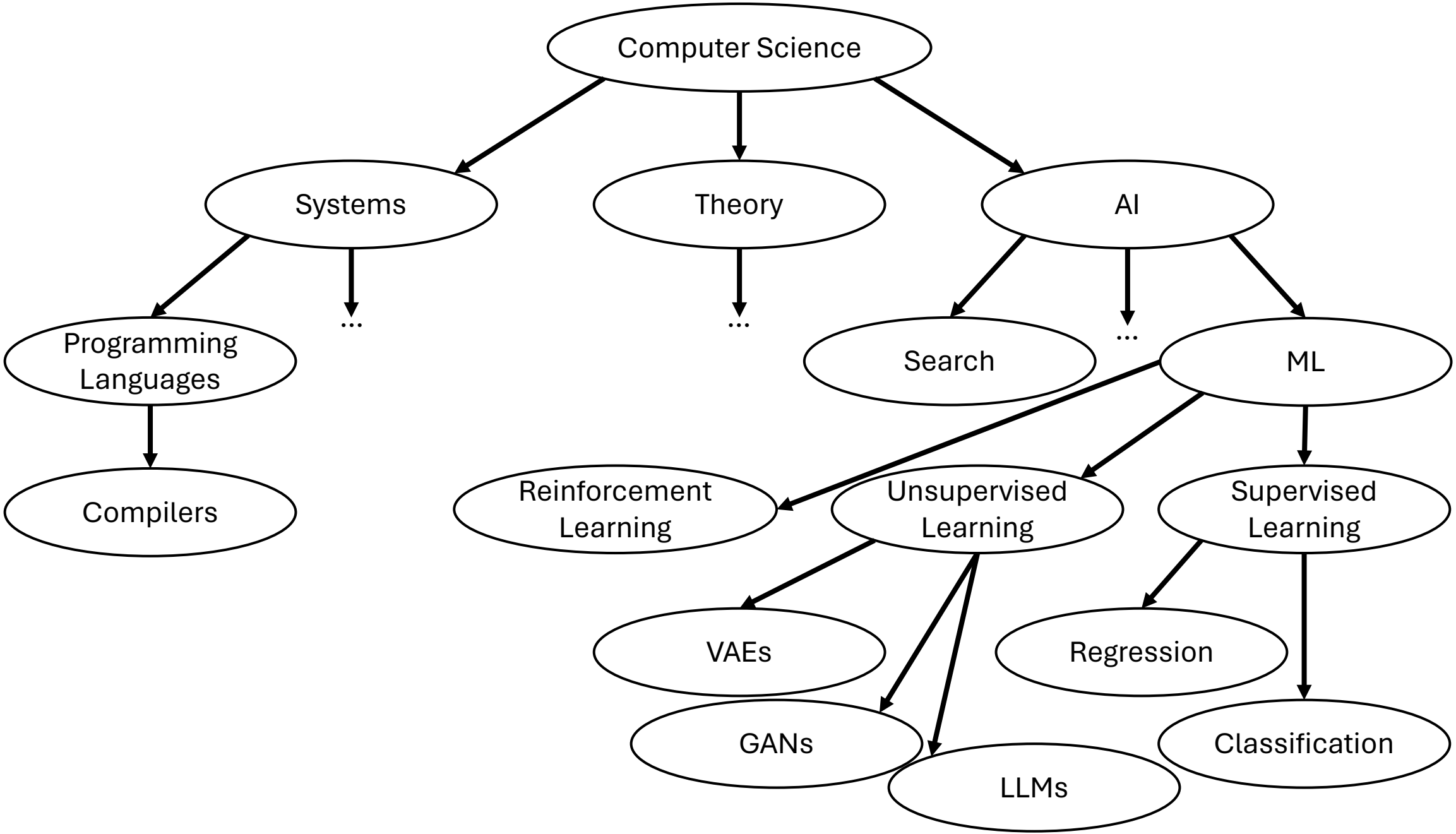
COMPSCI 389
Introduction to Machine Learning

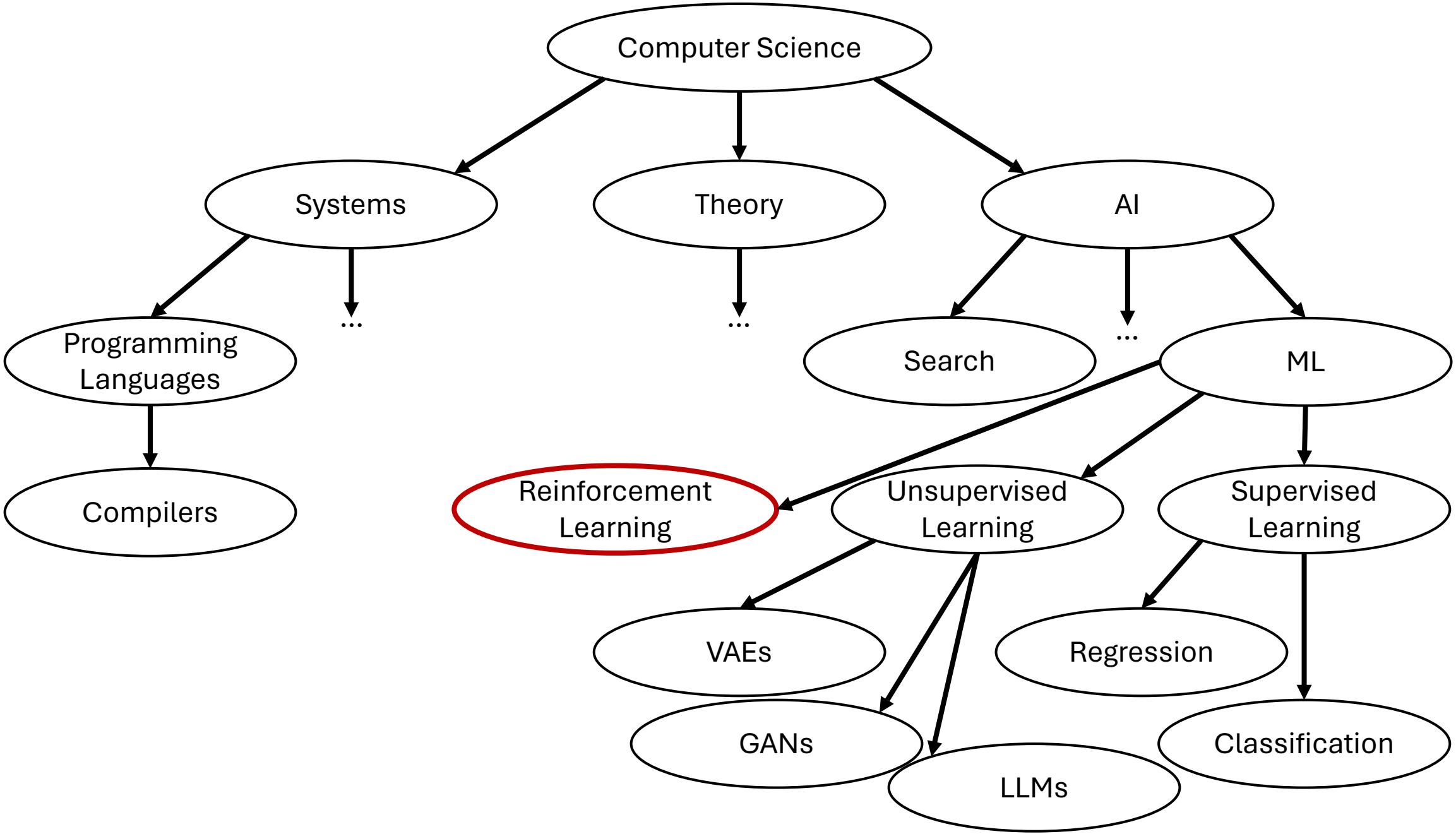
Days: Tu/Th. **Time:** 2:30 – 3:45 **Building:** Morrill 2 **Room:** 222

Topic 13.0: Reinforcement Learning

Prof. Philip S. Thomas (pthomas@cs.umass.edu)


Computer Science
Systems
Theory
AI
...
...
...
Programming Languages
Search
ML
...
...
...
Compilers
Supervised Learning
...
Classification
Program A
(code optimization)
Program B
(path finding)
Program C
(label prediction)


Computer Science
Systems
Theory
AI
...
...
...
Programming Languages
Search
ML
Compilers
Reinforcement Learning
Unsupervised Learning
Supervised Learning
VAEs
Regression
GANs
Classification
LLMs


Computer Science
Systems
Theory
AI
...
...
...
Programming Languages
Compilers
Search
ML
Reinforcement Learning
Unsupervised Learning
Supervised Learning
VAEs
GANs
LLMs
Regression
Classification

Supervised and Unsupervised Learning

- Algorithm learns from a fixed data set
- Supervised: Data includes labels
- Unsupervised: Data does not include labels
- **Semi-Supervised Learning**: Some data includes labels
  - Use unlabeled data to learn a representation (e.g., features)
  - Use labeled data to train a model using the learned representation
  - Not discussed further in this class

Reinforcement Learning

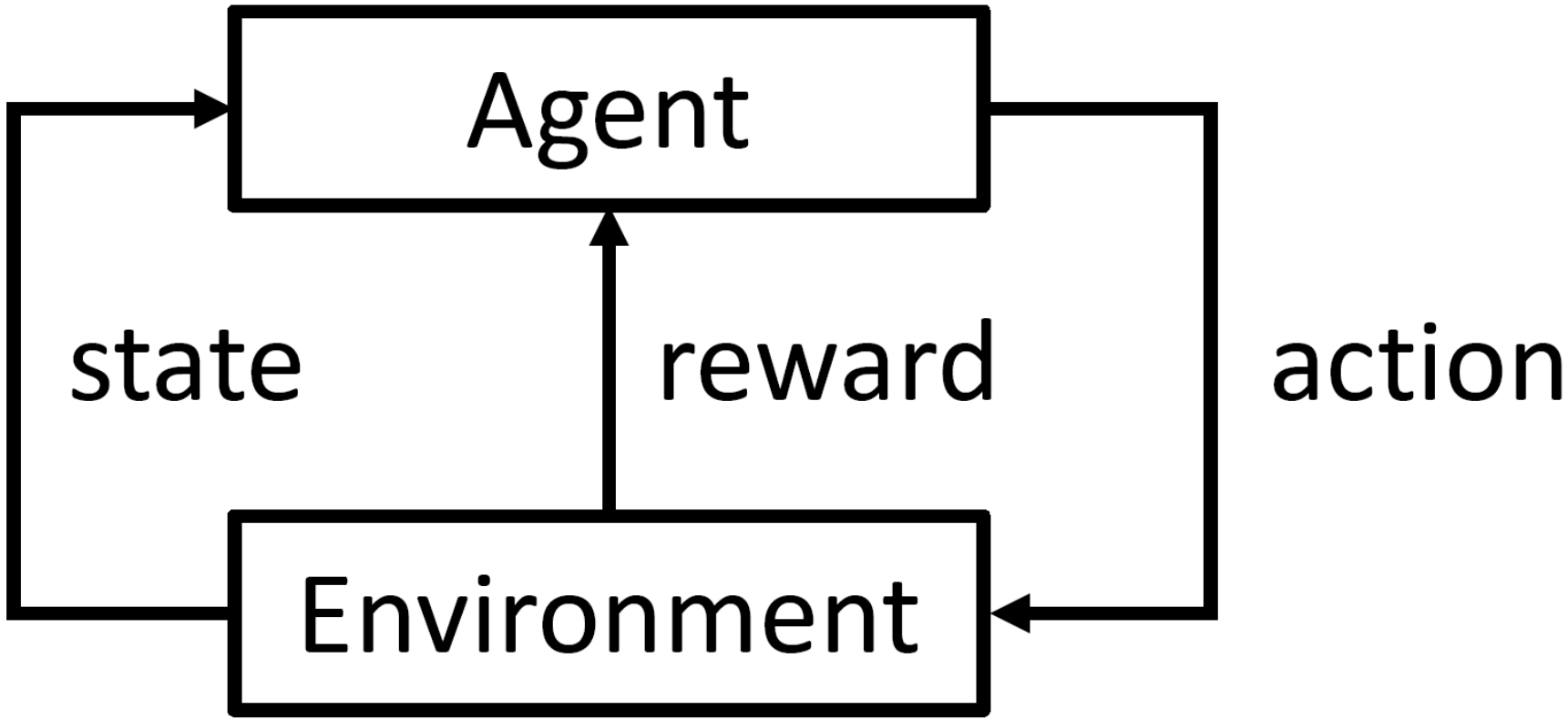
- There is no fixed data set.
- The decisions (predictions) made by the agent change the data the agent receives!
- Modeled as an agent interacting with an environment

Reinforcement learning is an area of machine learning, inspired by behaviorist psychology, concerned with how an agent can learn from interactions with an environment

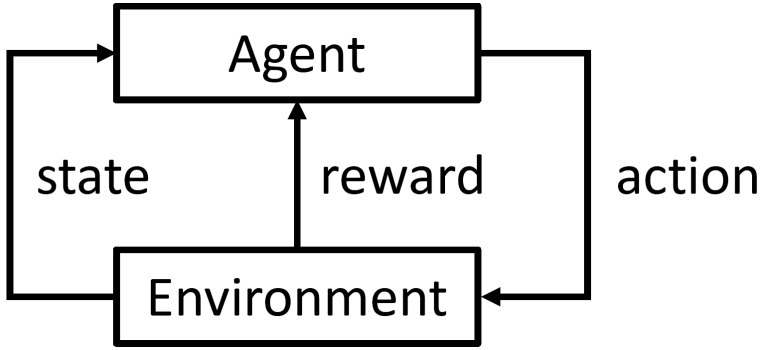
– Wikipedia / Sutton&Barto / Phil

How rewards and punishments shape our behavior.

Agent-Environment Diagram

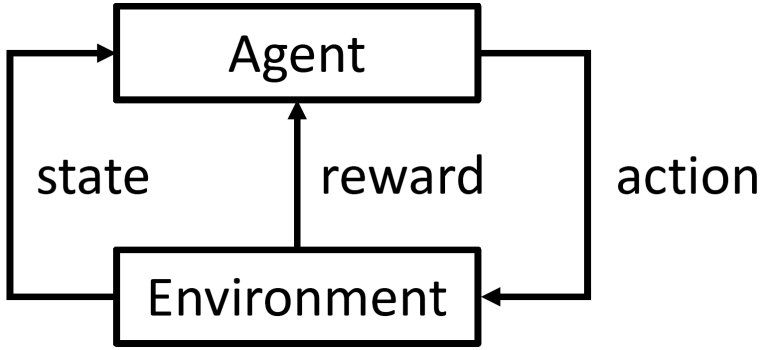


RL Problem Description



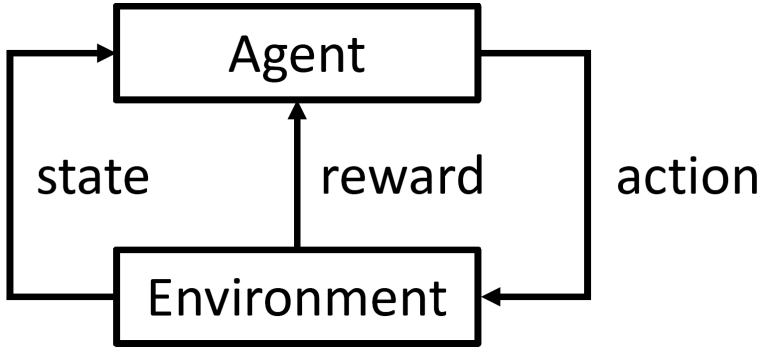
- The agent interacts with the environment over time $t \in \{0,1, \dots\}$.
- At each time the agent observes the **state** of the environment
  - For now, we assume that it observes the full state of the environment.
    - This is called the **fully observable** setting.
  - In general, the agent might only make a partial (noisy) observation about the state of the environment through its sensors.
    - This is called the **partially observable** setting.
- Based on its observation of the state, the agent selects an **action**.
  - The "parametric model" in RL is the mechanism in the agent that takes a state as input and produces an action as output.
  - This mechanism is called a **policy.**
    - Worse, in RL, **model** means something completely different! (A *model* of the environment)
  - It can be deterministic (always producing the same action given a state) or stochastic (producing a distribution over actions given the state).

RL Problem Description



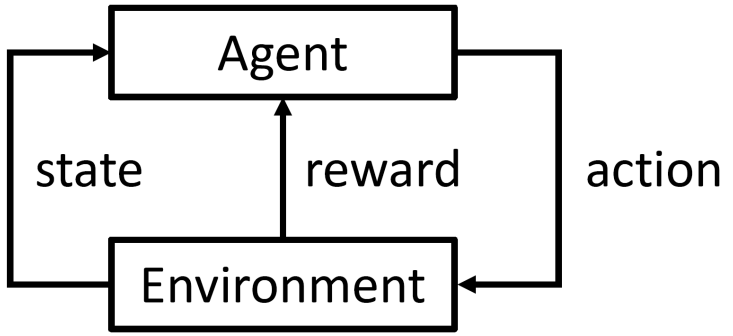
- The agent interacts with the environment over time $t \in \{0,1, \dots\}$.
- At each time the agent observes the **state** of the environment
- Based on its observation of the state, the agent selects an **action**.
  - The **policy** is the mechanism that determines the action given the state.
- The action causes the state of the environment to change.
  - This is called a **state transition**.
- When the state transitions, the environment also emits a scalar **reward**.
  - Intuitively, this reward indicates how "good" the current state is in the short term.
  - Sometimes it is called the **immediate reward** to emphasize the short-term nature of its evaluation.
- The sequence of agent-environment interactions can end, and the process restarts.
  - Each sequence of agent-environment interactions starting from time 0 is called an **episode**
  - This is the **episodic setting**. If the sequence of interactions never ends, it is called the **continuing setting**.
- The agent's goal is to find a policy that maximizes the total amount of reward that it receives.
  - The **return** is the sum of rewards that the agent receives during one episode.
  - The same policy can produce different returns during different episodes due to stochasticity in the state transitions, rewards, and policy.
  - The agent's goal is to maximize the **expected return**.

RL Problem Description



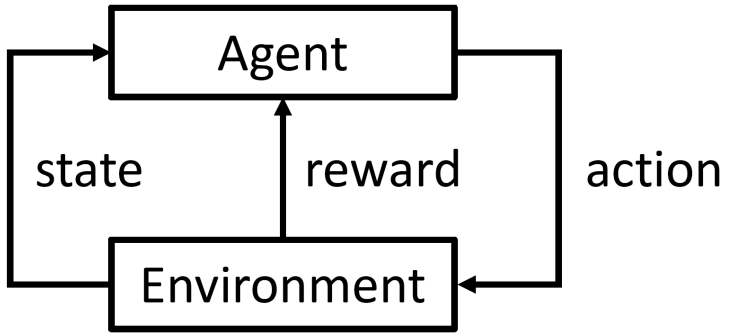
- The agent interacts with the environment over time $t \in \{0,1, \dots\}$.
- At each time the agent observes the **state** of the environment
- Based on its observation of the state, the agent selects an **action**.
  - The **policy** is the mechanism that determines the action given the state.
- The action causes a **state transition**.
- When the state transitions, the environment also emits a scalar **reward**.
- Each sequences of agent-environment interactions starting from time 0 is called an **episode**. Episodes can end (terminate).
- The **return** is the sum of rewards that the agent receives during one episode.
- The agent's goal is to maximize the **expected return**.

Key Properties of RL



- **Evaluative feedback, not instructive feedback**
  - **Instructive feedback** tells an agent what the correct decisions would have been
    - Labels in supervised learning provide instructive feedback.
  - **Evaluative feedback** tells an agent how good its decisions were
    - Rewards in RL provide evaluative feedback.
    - Evaluative feedback can be noisy (random)
    - The range of possible feedback values may not be known.
      - Is a reward of $+10$ good or bad? The agent must interact with the environment to figure this out!
- **Sequential**
  - The agent's goal is to maximize the expected return (expected sum of rewards).
  - This can require it to forgo larger short-term rewards to obtain larger rewards in the future.

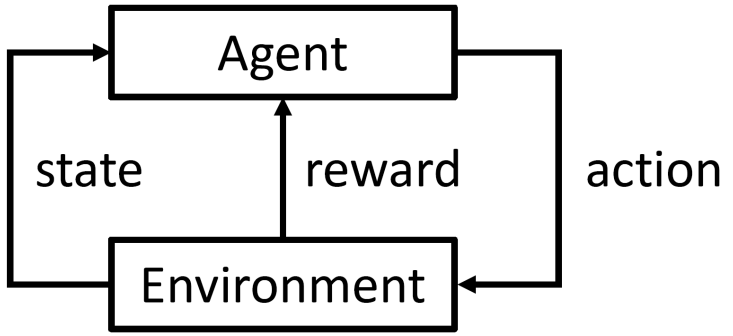
RL Examples



- **Agent**: Child
- **Environment**: World
- **Goal**: The child may learn to grasp an object or get a parent's attention

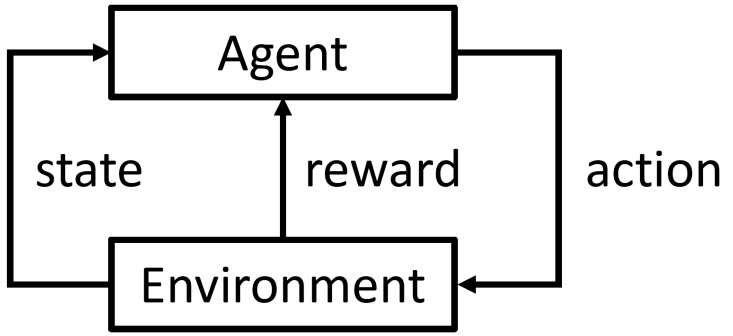
- **State**: The state of the world around the child (partially observed!)
- **Action**: Decision of how much to activate each muscle
- **Reward**: Positive when an object is picked up, negative when an object is dropped, positive when a parent responds, etc.

RL Examples



- **Agent**: Dog
- **Environment**: World
- **Goal**: Learn to fetch or catch

RL Examples

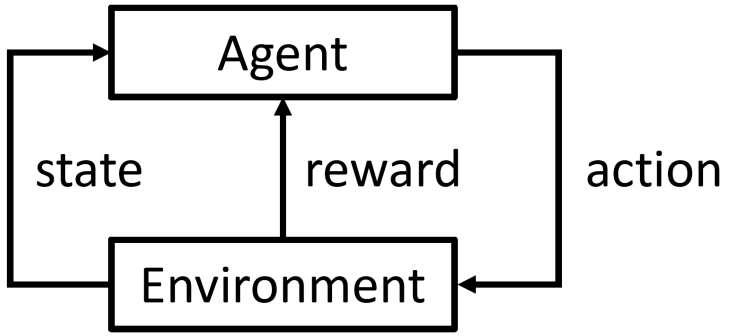


- **Agent**: Dog
- **Environment**: World
- **Goal**: Learn to fetch or catch

- **State**: The state of the world around the dog (partially observed!)
- **Action**: Decision of how much to activate each muscle
- **Reward**: Positive when food is obtained

- **Note**: Each catching attempt can be viewed as an **episode**.

RL Examples



- **Agent**: Robot
- **Environment**: Lab
- **Goal**: Lift a heavy object

The learned policy exploits the container dynamics.