Reinforcement *Learning*

by

Giovanni Stilo, PhD.

giovanni.stilo@univaq.it

Outline

- Course logistics
- Overview of reinforcement learning
- Introduction to sequential decision making under uncertainty

Course logistics

Lectures On January:
7 (11:50), 12 (10:10), 14 , 19, 21

Evaluation is based on a final homework.

Course/slides is/are based on the lectures of:

- David Silver
- Emma Brunskill
- Devin Schwab

- Textbooks [Freely available on the Web]:
  - David L. Poole, Alan K. Mackworth, ARTIFICIAL INTELLIGENCE FOUNDATIONS OF COMPUTATIONAL AGENTS
  - Sutton & Barto, Reinforcement Learning: An Introduction;
  - Csaba Szepesvari, Algorithms for Reinforcement Learning;
  - *Goodfellow, Bengio & Courville, Deep Learning;*

Reinforcement Learning

Learn to make good sequences of decisions

Repeated Interactions with World

Learn to make good **sequences of decisions**

Reward for Sequence of Decisions

Learn to make **good** sequences of decisions

Don’t Know in Advance How World Works

Learn to make good sequences of decisions

Fundamental challenge in AI and ML

Learning to make good decisions under uncertainty

RL, Behavior & Intelligence

Sea Squirt – Primitive Animal

Childhood: primitive brain & eye, swims around, then attaches to a rock
Adulthood: once it is settled down, he digests brain, sits

Suggests: brain is helping guide decisions
(no more decisions, no need for brain?)

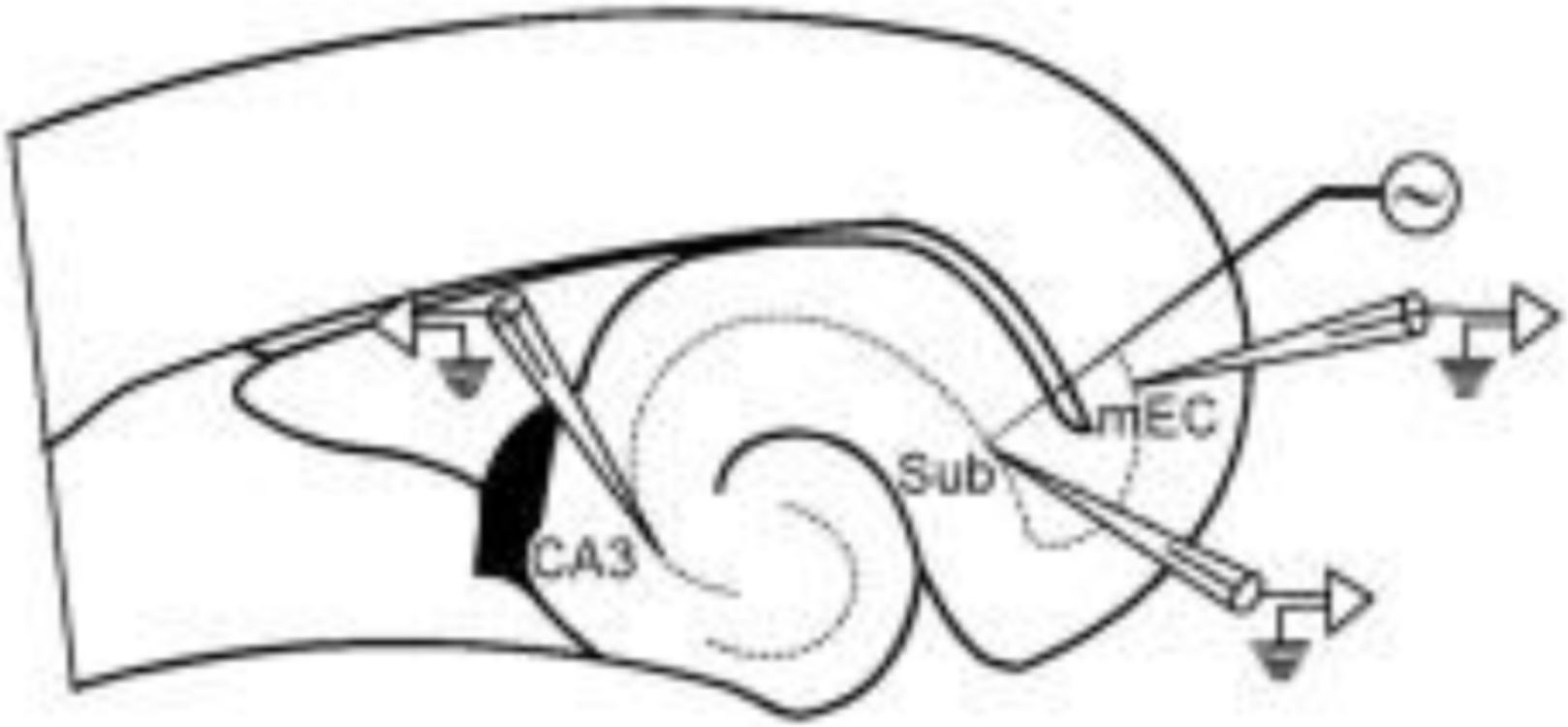
Atari Games – 2015 DeepMind

https://www.youtube.com/watch?v=V1eYniJ0Rnk

Robotics

https://www.youtube.com/watch?v=CE6fBDHPbP8

Healthcare



Adaptive control of epileptiform excitability in an in vitro model of limbic seizures.

RL Involves

- Optimization
- Delayed consequences Exploration
- Generalization

Optimization

- Goal is to find an optimal way to make decisions
  - Yielding best outcomes
- Or at least a very good strategy

Delayed Consequences

- **Today's** decisions have an **impact** on things much **later**:
  - Saving for retirement
  - Finding a key in Montezuma's revenge (Atari Game)

- Introduces two challenges:
  1. **When planning:** decisions involve reasoning about not just immediate benefit of a decision but also its longer term ramifications

  2. **When learning:** temporal credit assignment is hard (what caused later high or low rewards?)

Exploration

- Learning about the world by making decisions:
  - Agent as scientist;
  - Learn to ride a bike by trying (and failing);
  - Finding a key in Montezuma's revenge;

- Censored data:
  - Only get a reward (label) for decision made;
  - Don't know what would have happened if we had taken red pill instead of blue pill (Matrix movie reference) ;
- Decisions impact what we learn about:
  - If we choose to go to London instead of UAQ, we will have different later experiences...

Question!

- Policy is mapping from past experience to action

Why not just pre-program a policy?

Generalization

- Policy is mapping from past experience to action
- Why not just pre-program a policy?

How many possible images are there?

- $256^{100x200}$

RL Involves

- Optimization
- Exploration
- Generalization
- Delayed consequences

AI Planning (vs RL)

- **Optimization**
- **Generalization**
- ~~Exploration~~
- **Delayed consequences**

- **GOAL**: Computes good sequence of decisions

- *having/giving a model of how decisions impact world*

Supervised Machine Learning (vs RL)

- **Optimization**
- **Generalization**
- ~~Exploration~~
- ~~Delayed consequences~~

- **GOAL:** Learns from experience

- *having/providing correct labels*

Unsupervised Machine Learning (vs RL)

- **Optimization**
- **Generalization**
- ~~Exploration~~
- ~~Delayed consequences~~

- **GOAL:** Learns from experience

- *no labels from world;*
- *but having/providing some knowledge*

Imitation Learning (vs RL)

- **Optimization**
- **Generalization**
- ~~Exploration~~
- **Delayed consequences**

- **GOAL:** Learns from experience … **of the others**

- *Assumes input demos of good policies*

Imitation Learning

https://www.youtube.com/watch?v=Idn10JBsA3Q

Imitation Learning

- Reduces RL to supervised learning
- Benefits:
  - Great tools for supervised learning
  - Avoids exploration problem
  - Lots of data about the outcomes of decisions

- Limitations:
  - Can be expensive to capture the behaviours
  - Limited amount of collected data

- Imitation learning + RL promising!

How it works RL?

1. Explore the world

2. Use experience to guide future decisions

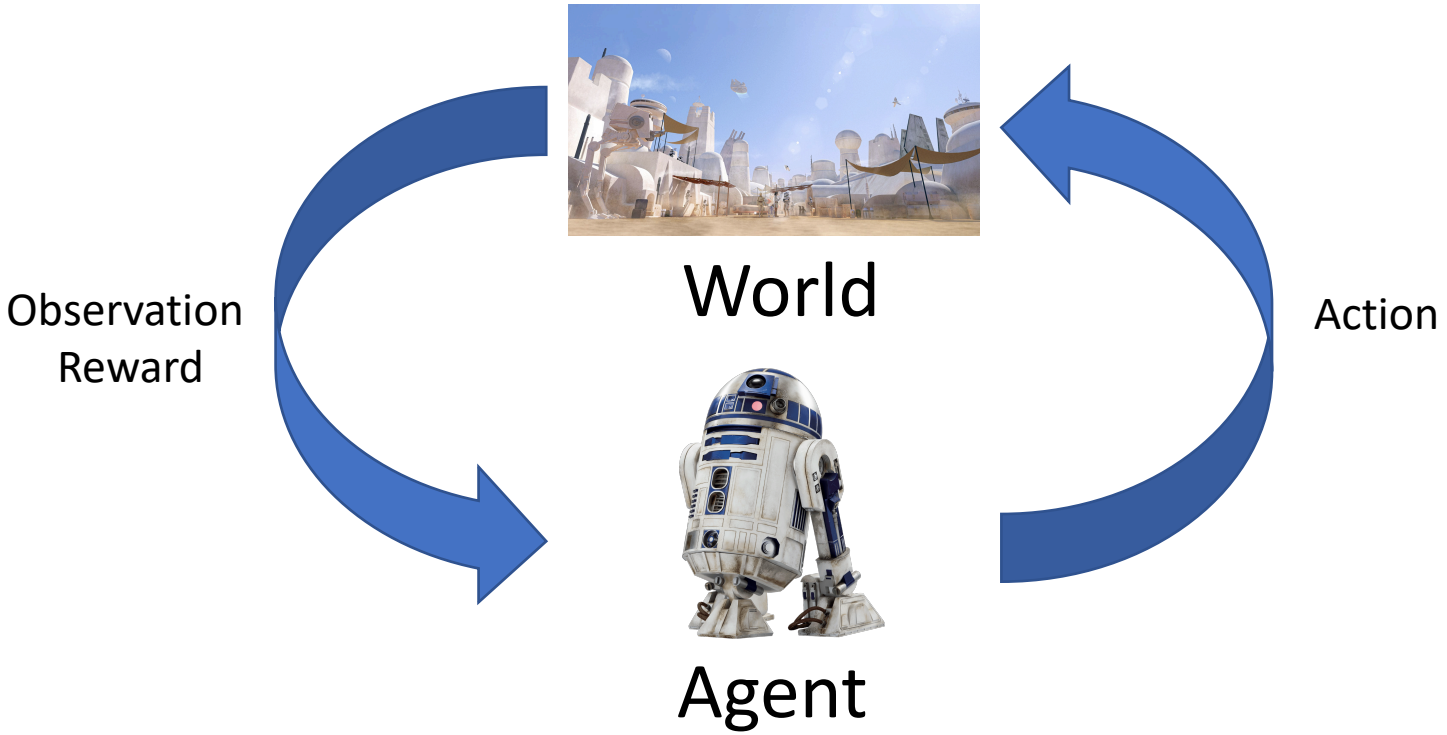
Tricky parts:

- Where do rewards come from?
  - And what happens if we get it wrong?

- Robustness / Risk sensitivity

- We are not alone...
  - Multi-agent RL

Sequential Decision Making



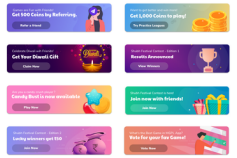
- Interactive closed-loop process;
- **Goal**: Select actions to maximize total expected future reward;
- Requires **balancing** immediate & long term **rewards**;
- Requires **strategic behaviour** to achieve high rewards;

Example: Web Advertising

World

View Time
Click on Ad

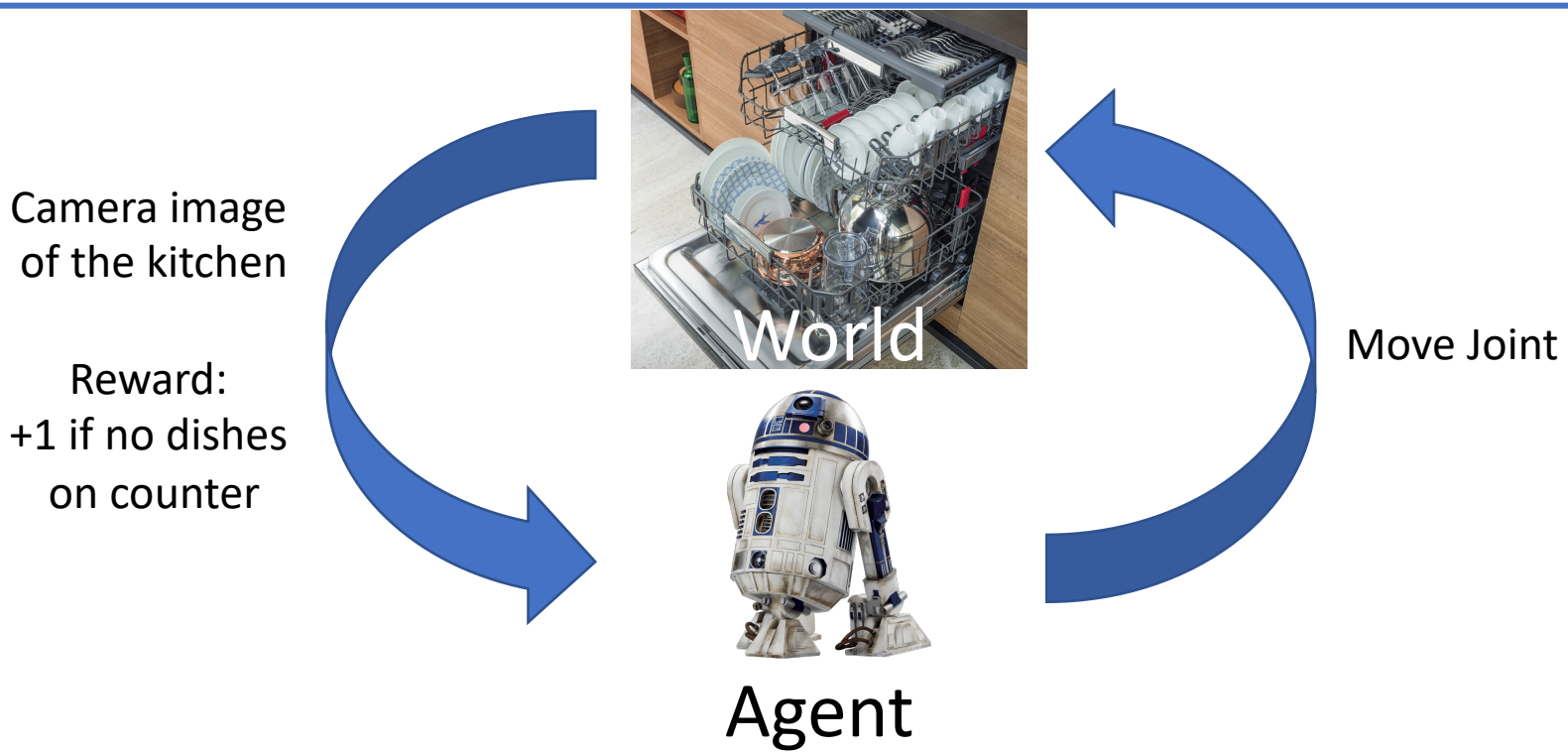
Chose a
web Ad



Agent

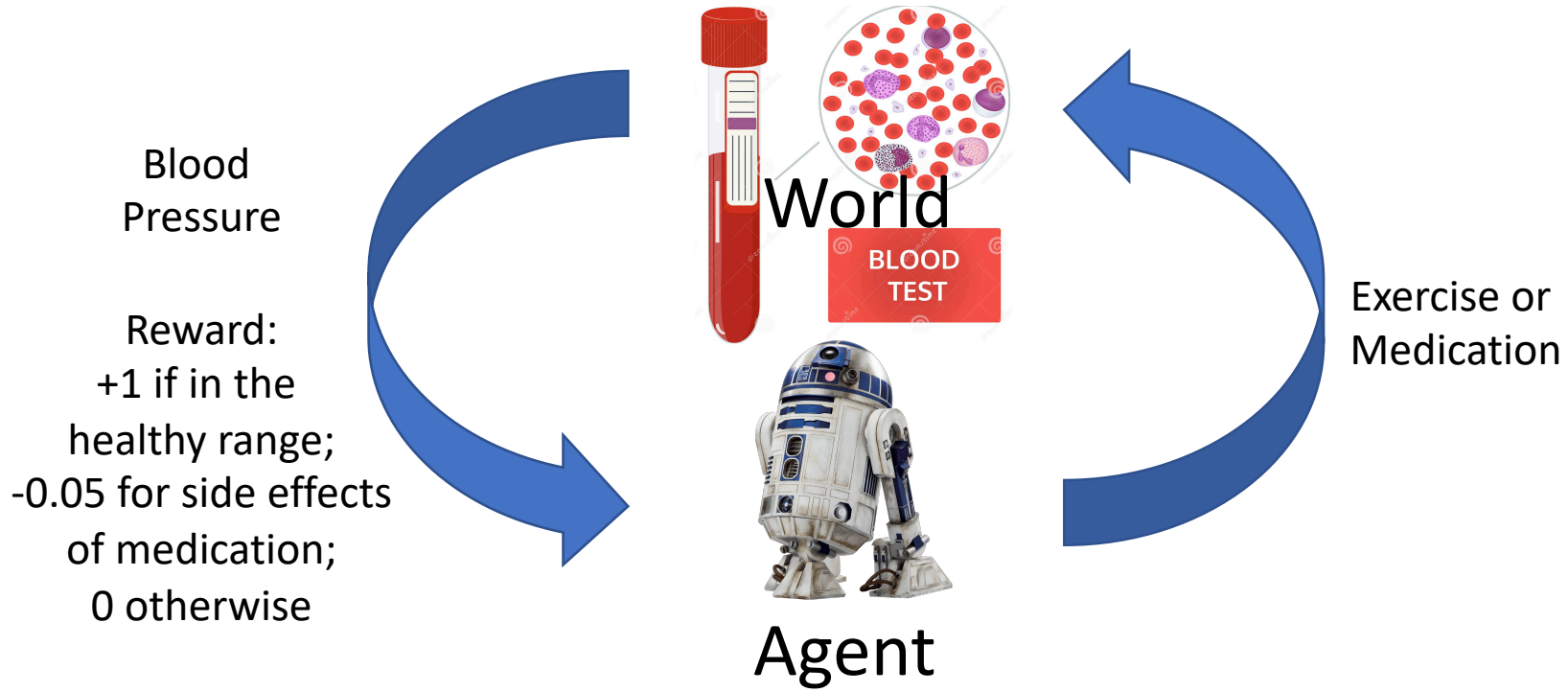
- **Goal**: Select actions to maximize total expected future reward;
- Requires **balancing** immediate & long term **rewards**;
- Requires **strategic behaviour** to achieve high rewards;

Example: Robot Unloading Dishwasher



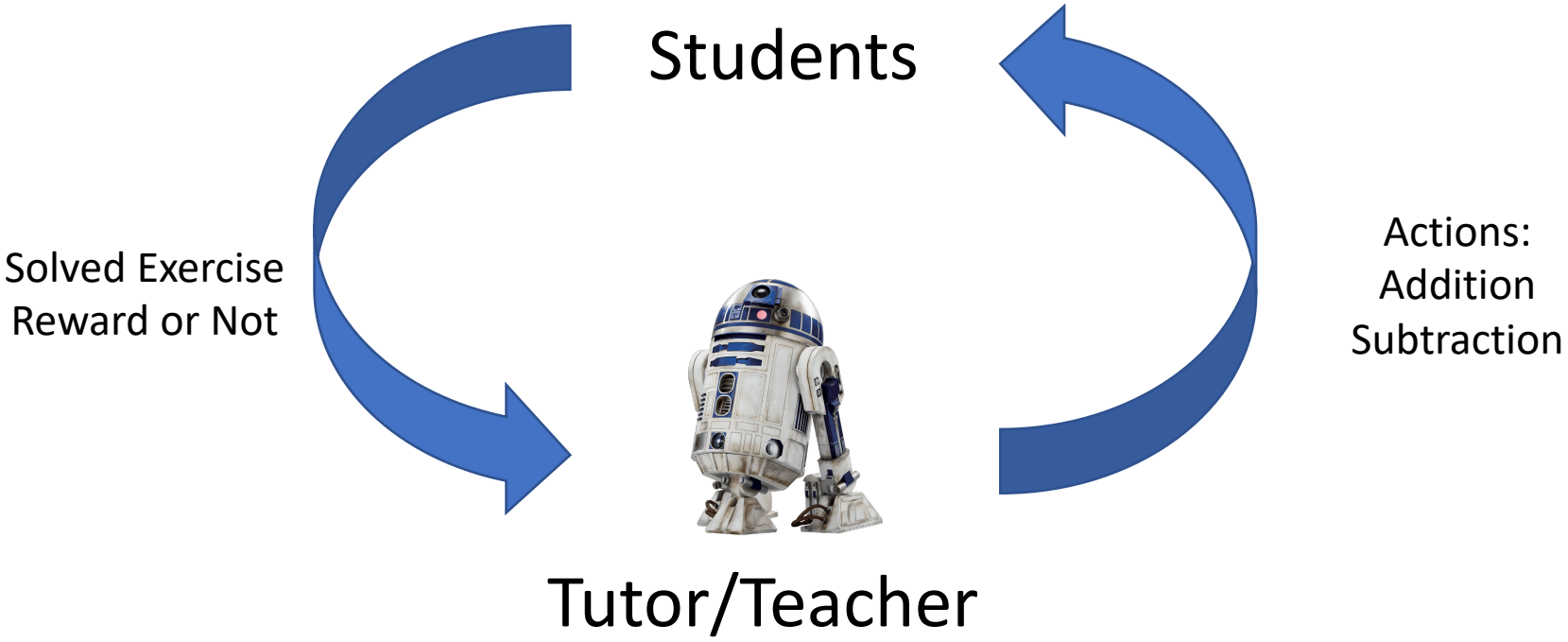
- **Goal**: Select actions to maximize total expected future reward;
- Requires **balancing** immediate & long term **rewards**;
- Requires **strategic behaviour** to achieve high rewards;

Example: Blood Pressure Control



- **Goal**: Select actions to maximize total expected future reward;
- Requires **balancing** immediate & long term **rewards**;
- Requires **strategic behaviour** to achieve high rewards;

Artificial Teacher/Tutor



- Student **need** to learn **addition** and **subtraction;**
- Teacher/Tutor can provide exercise about addition or subtraction;
- Tutor will receive feedback about the students ability to correctly solves problems;

Artificial Teacher/Tutor

Student

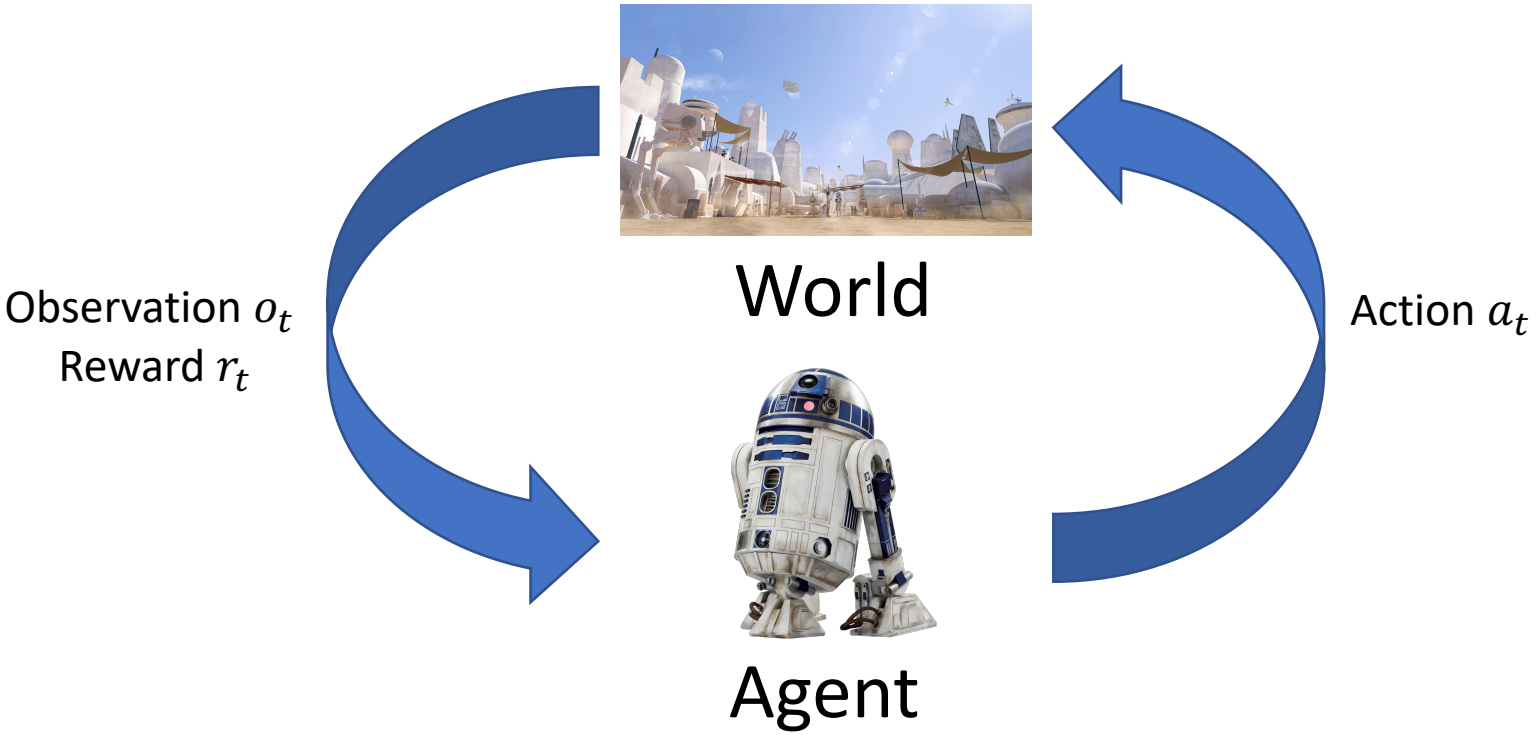
Student successfully solves the given problems?

Pick a teaching activity: Addition / Subtraction

Teaching Agent

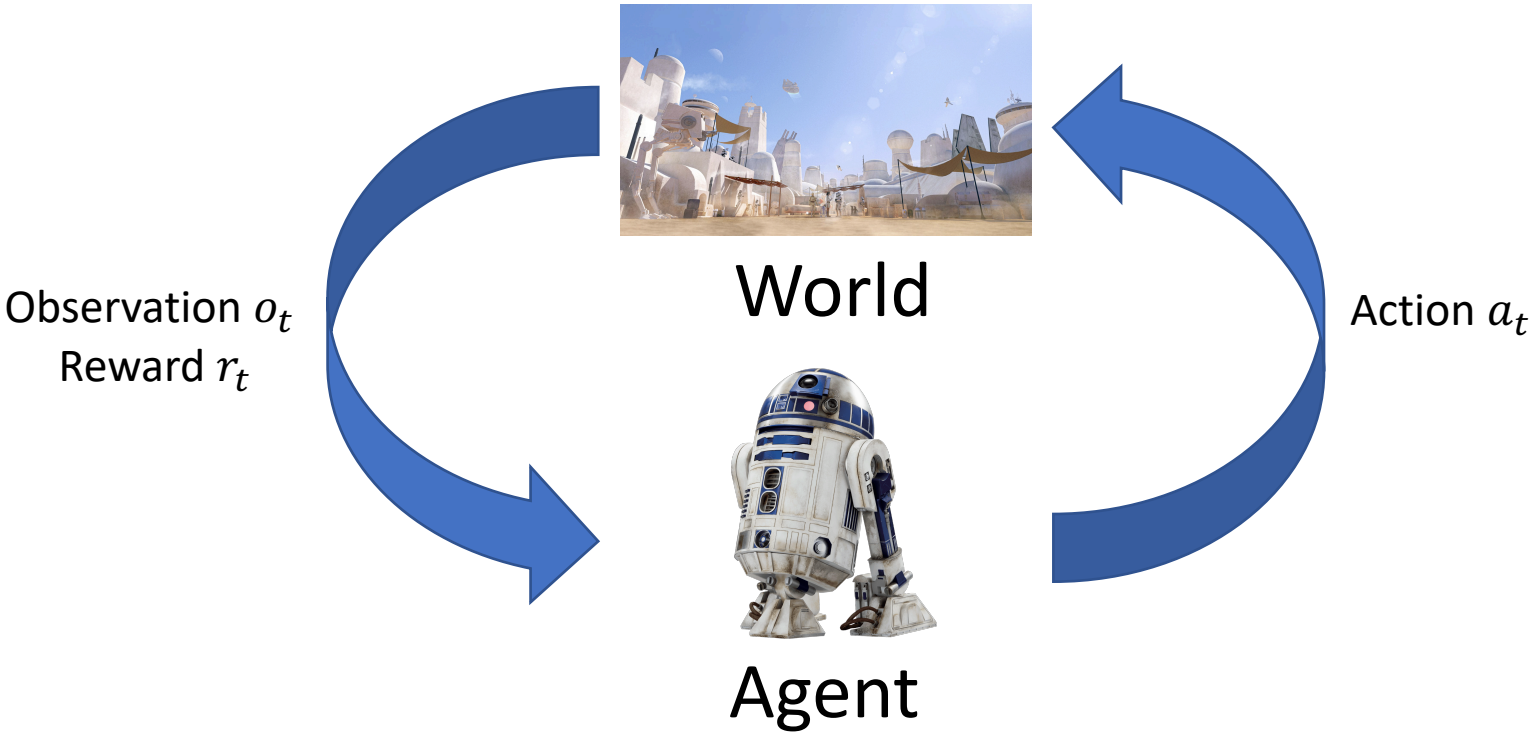
- Student initially does not know addition (easier 75%), nor subtraction (harder 50%);
- Teaching agent can provide activities about addition or subtraction;
- Agent gets rewarded for student performance: +1 if student gets problem right, -1 if get problem wrong;
- 1 - 0.75 = Addition 1 - 0.5 = Subtraction
- *Which items will agent learn to give to max expected reward? Is this the best way to optimize for learning? If not, what other reward might one give to encourage learning?*

Seq. Decision Process: Agent & the World



- Each time step t:
  - Agent takes an action a_t
  - World updates given action a_t, emits observation o_t and reward r_t
  - Agent receives observation o_t and reward r_t

History: Sequence of Past Observations, Actions & Rewards



- History is the Sequence of Past Observations, Actions & Rewards:

$$h_t = (a_1, o_1, r_1, \ldots, a_t, o_t, r_t)$$

- Agent chooses action based on history
- State is information assumed to determine what happens next:
  - Function of history: $s_t = f(h_t)$

World State

World

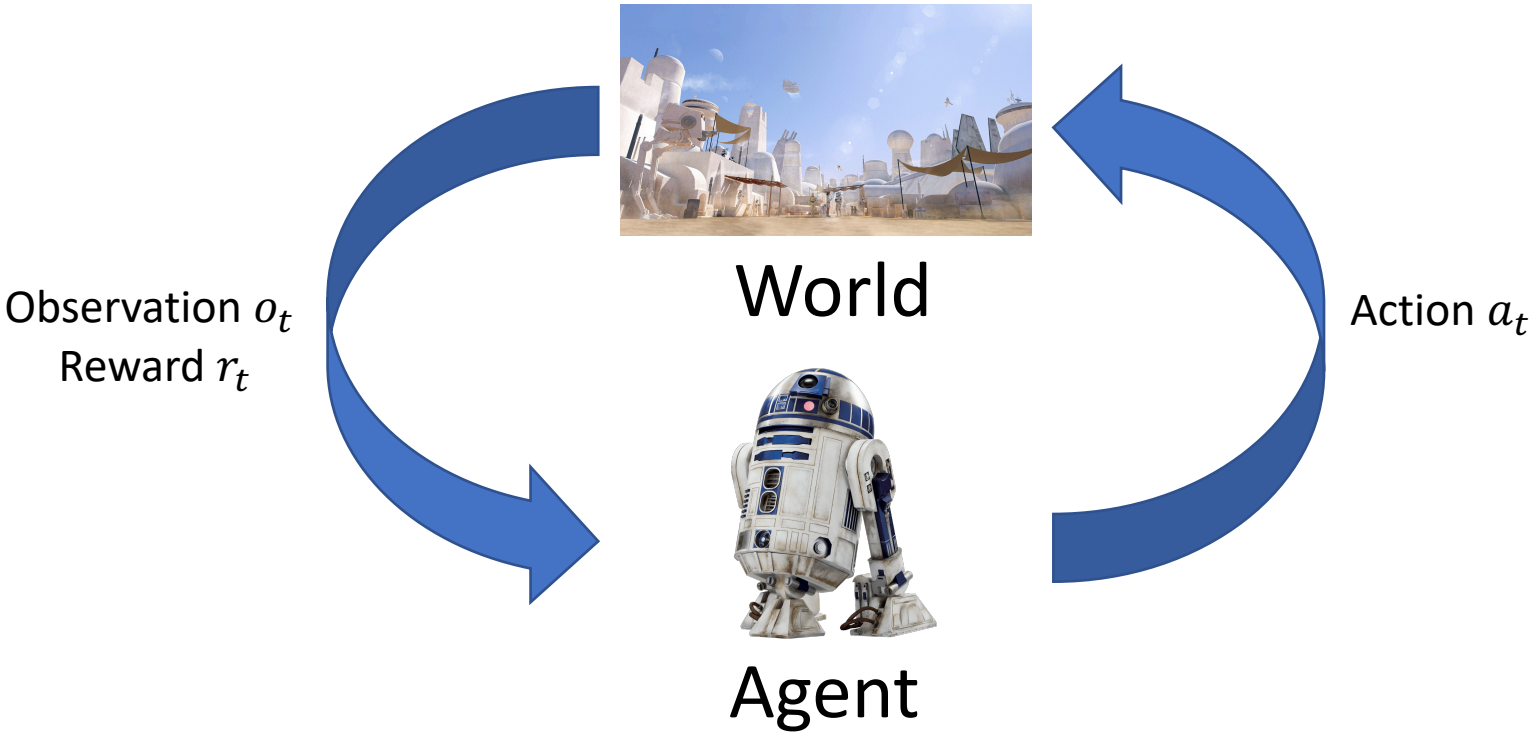
Observation o_t
Reward r_t

Action a_t

Agent

- This is the true state of the world used to determine how world generates next observation and reward;
- Often/Typically hidden or unknown to agent;
- Even if known may contain information not needed by agent;

Agent State: Agent's Internal Representation



- What the agent / algorithm uses to make decisions about how to act;
- Generally the current state is a function of the history: $s_t = f(h_t)$
- Could include meta information like state of algorithm (how many computations executed, etc) or decision process (how many decisions left until an episode ends)

Markov Assumption

- Information state: *sufficient statistic of history*
- State s_t is Markov if and only if:

  $p(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid h_t, a_t)$

- Given present, the future is **independent** of the past (if you have *sufficient statistic of history)*;

Markov Assumption for Prior Examples

- Information state: *sufficient statistic of history*

- State s_t is Markov if and only if:

$$p(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid h_t, a_t)$$

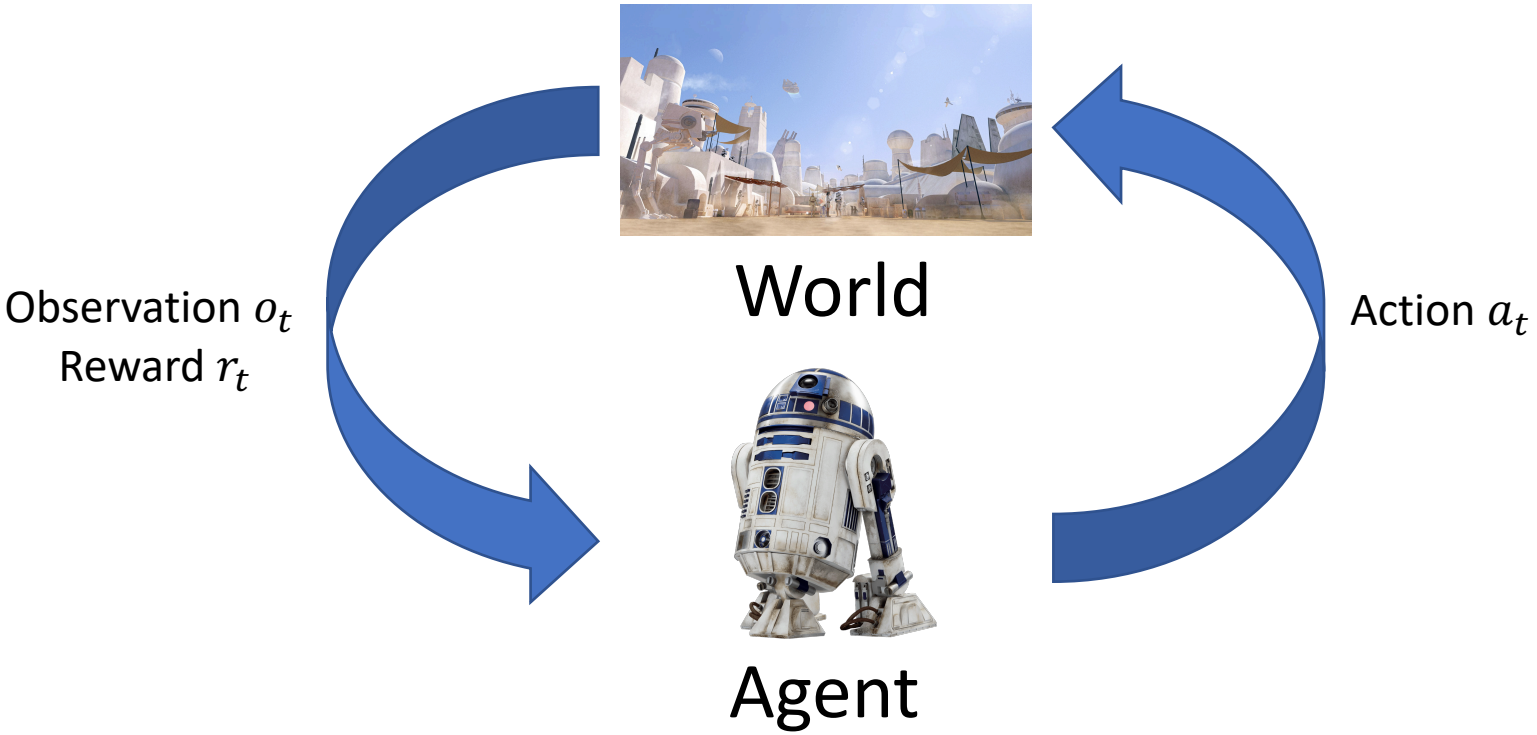
- Given present, the future is **independent** of the past;

- Hypertension control: let state be current *blood pressure*, and action be whether to *take medication* or not.
  Is this system Markov?
- Website shopping: state is *current product viewed* by customer, and action is what other product to recommend.
  Is this system Markov?

Why is Markov Assumption Popular?

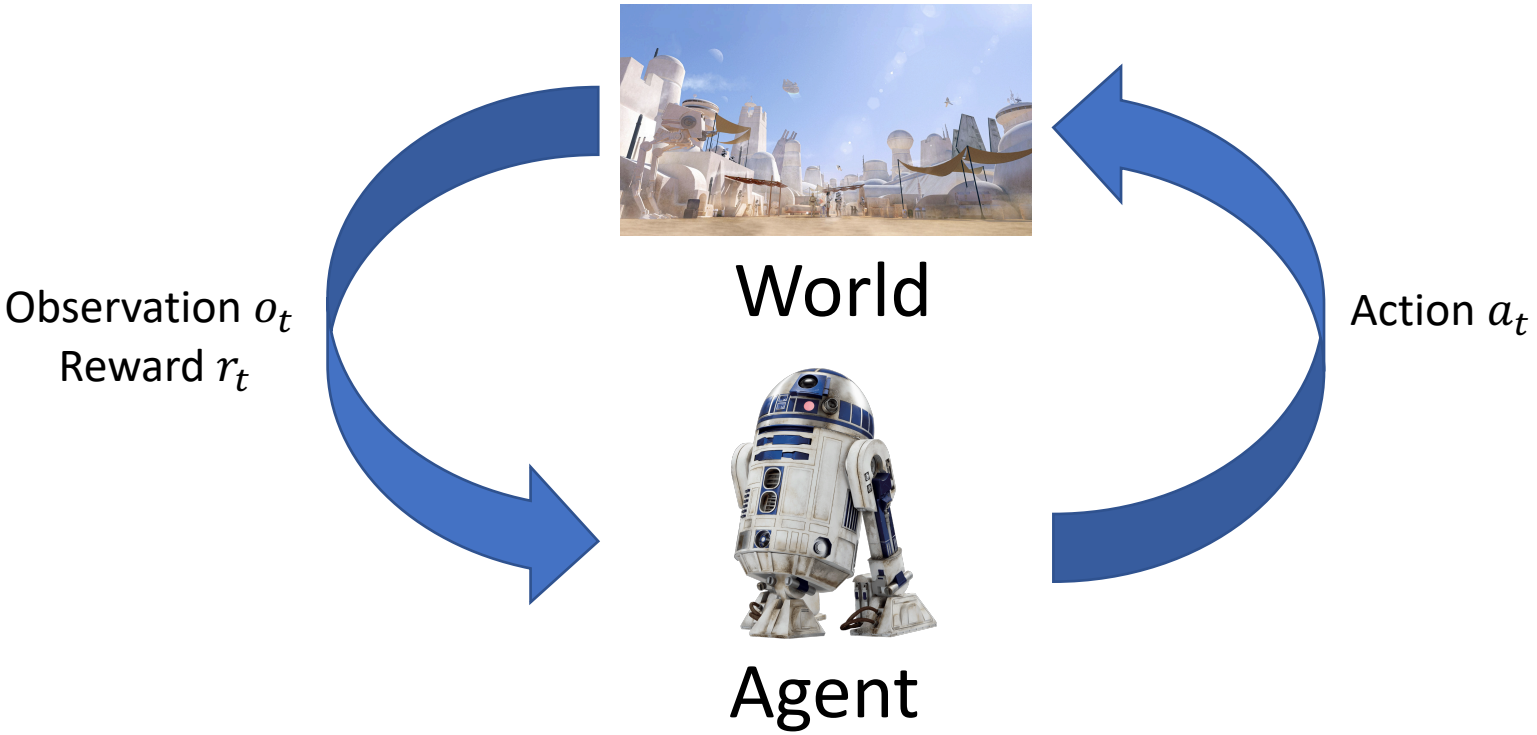
- Can always be satisfied:
  - Setting state as history always Markov: $s_t = h_t$;
- In practice often assume most recent observation is sufficient statistic of history: $s_t = o_t$;
- State representation has big implications for:
  - Computational complexity;
  - Data required;
  - Resulting performance;
- That's why you won't use the complete history;

Full Observability / Markov Decision Process (MDP)



- Environment and world state s_t = o_t

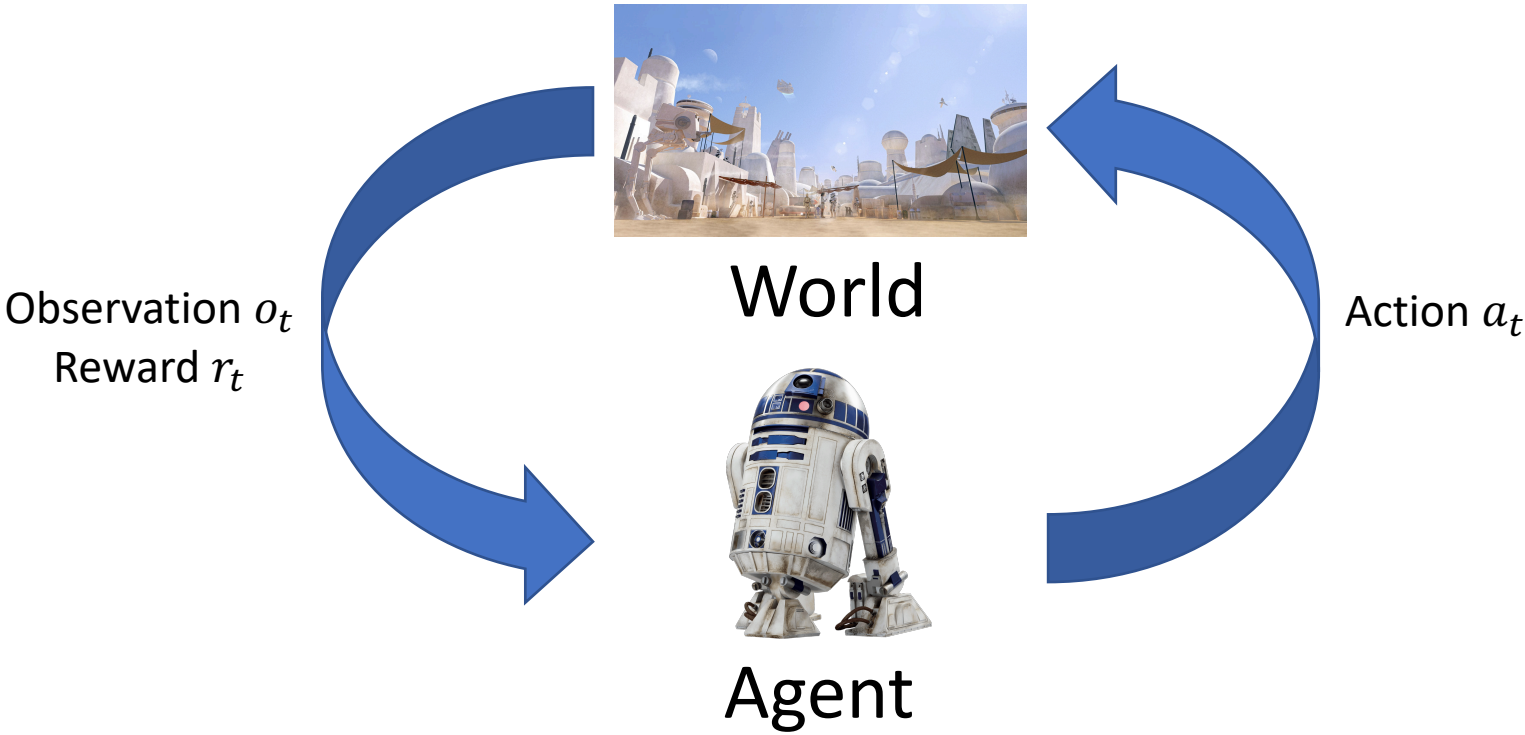
Partial Observability / Partially Observable Markov Decision Process (POMDP)



- Agent state is **not the same** as the world state
- Agent constructs its own state, e.g.
  - Use history $s_t = h_t$, or beliefs of world state, or RNN, ..

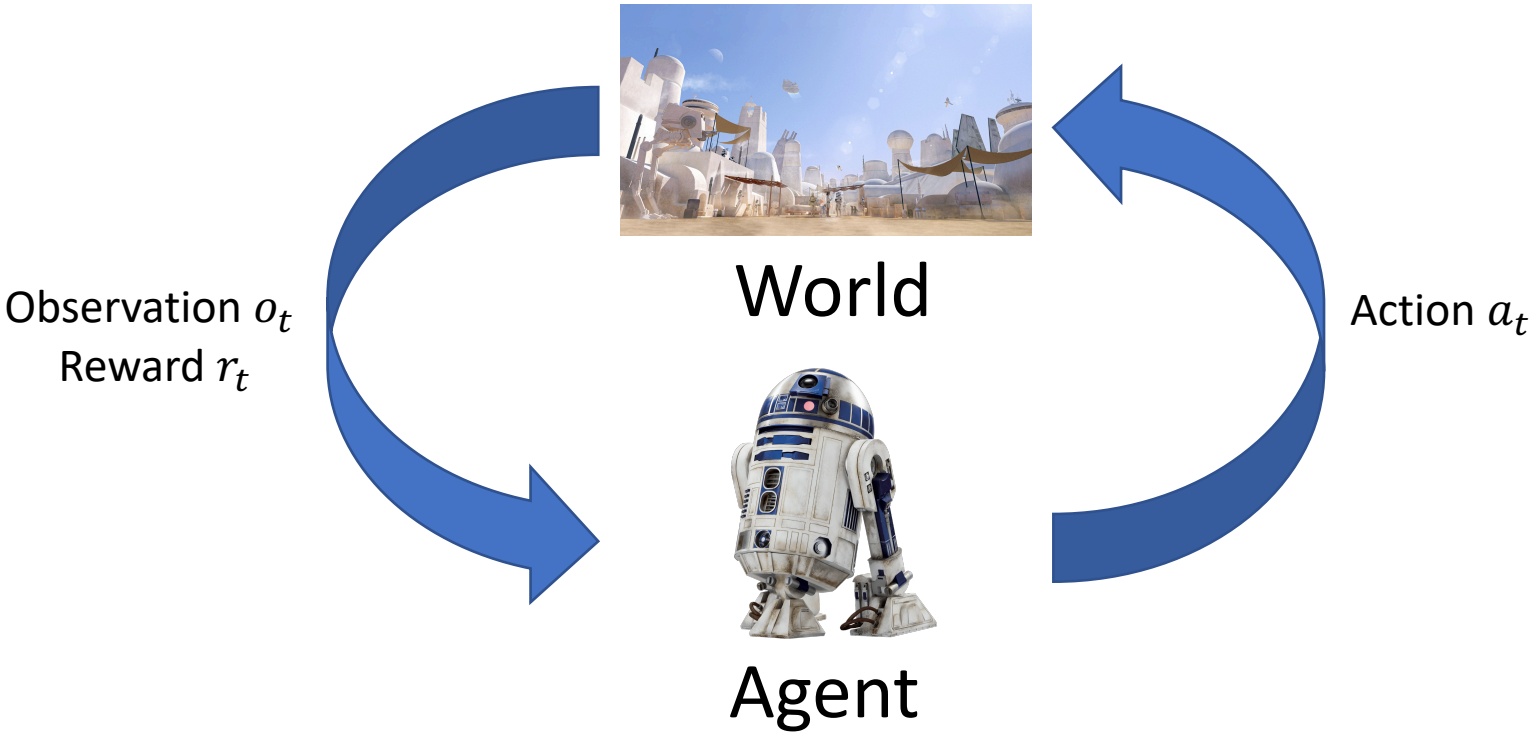
Partial Observability Examples

- Poker player (only sees own cards)
- Healthcare (don't see all physiological process)



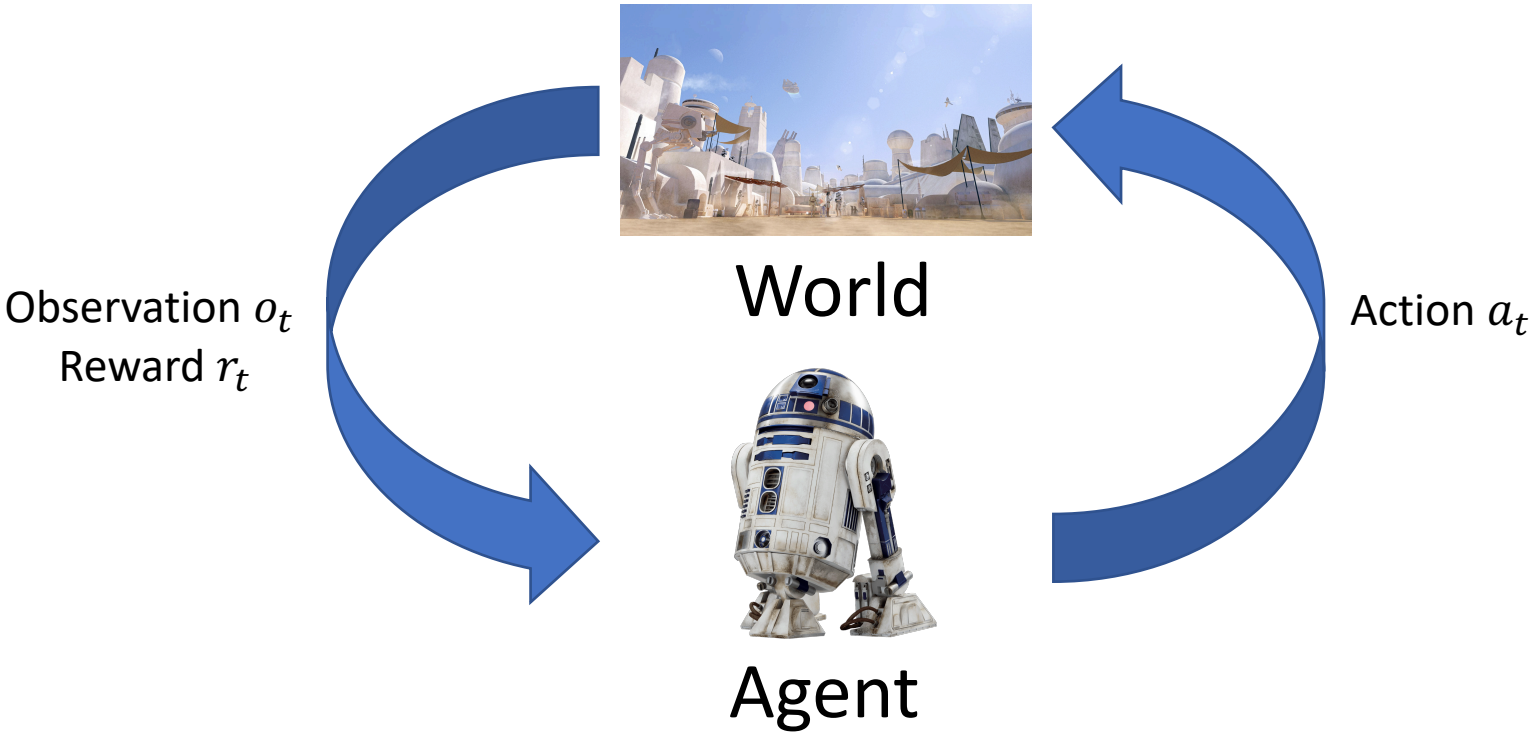
- Agent state is **not the same** as the world state
- Agent constructs its own state, e.g.
  - Use history $s_t = h_t$, or beliefs of world state, or RNN, ..

Types of Sequential Decision Processes: *Bandits*



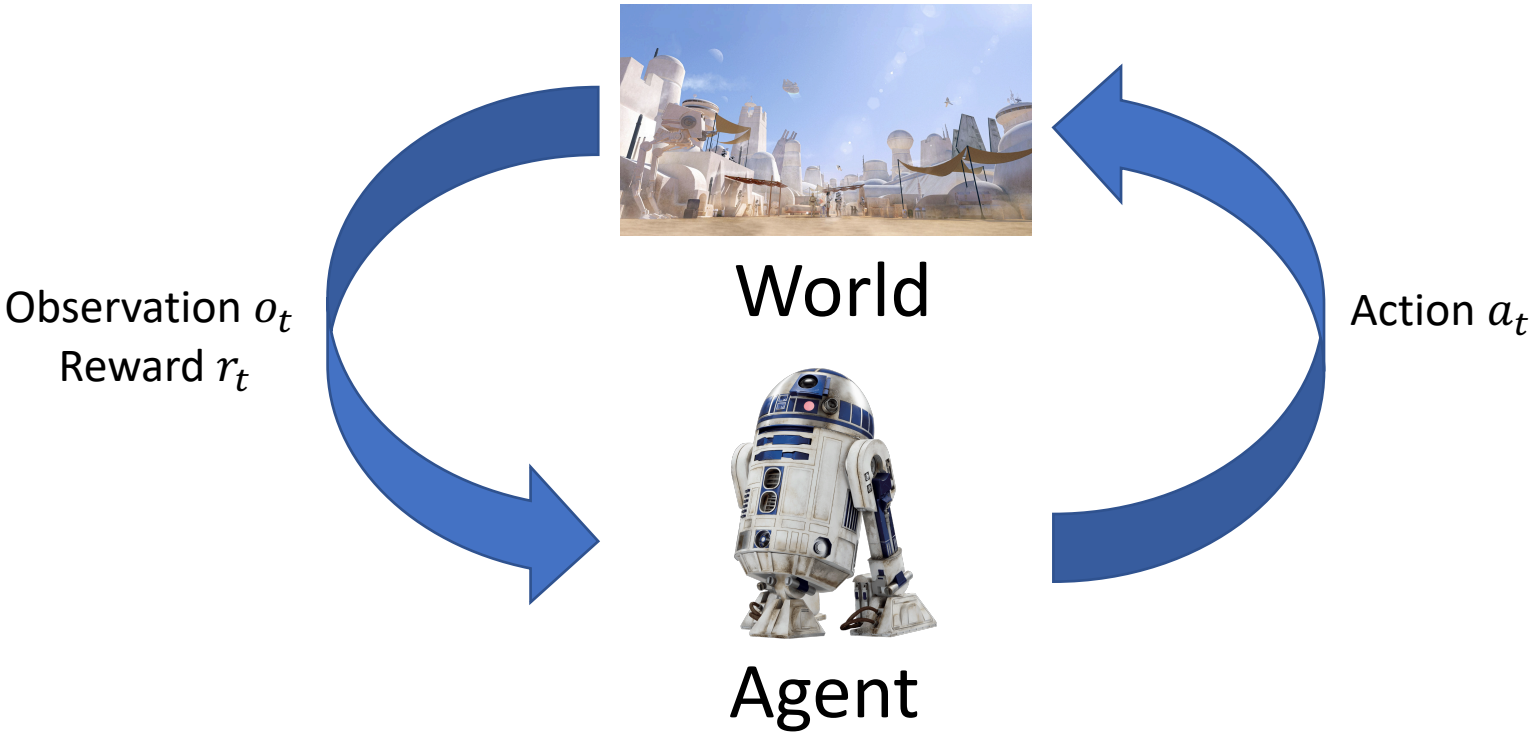
- **Bandits**: actions have no influence on next observations
- *No delayed rewards / Consequences*

Types of Seq. Decision Processes: *MDPs & POMDPs*



- Actions influence future observations
- *Credit assignment and strategic actions may be needed*

Types of Seq. Decision Processes: *How the World Changes*



- **Deterministic:** Given history & action, **single** observation & reward
  - *Common assumption in robotics and controls*

- **Stochastic:** Given history & action, **many** potential observations & rewards
  - *Common assumption for customers, patients, hard to model domains*

Example: Mars Rover as a Particular MDP

S_1	S_2	S_3	S_4	S_5	S_6	S_7
						

- **States:** Location of rover ($S_1, \ldots, S_7$)
- **Actions:** Left or Right
- **Rewards:**
  - +1 in state S_1
  - +10 in state S_7
  - 0 in all other states

RL Algorithm Components

- Can Include one or more of:
  - **Model:**

    *Representation of how the **world changes** in response to agent's action;*
  - **Policy:**

    *Function mapping agent's **states to action**;*
  - **Value function:**

    ***Future rewards** from being in a **state** and/or **action** when following a particular policy;*

Model

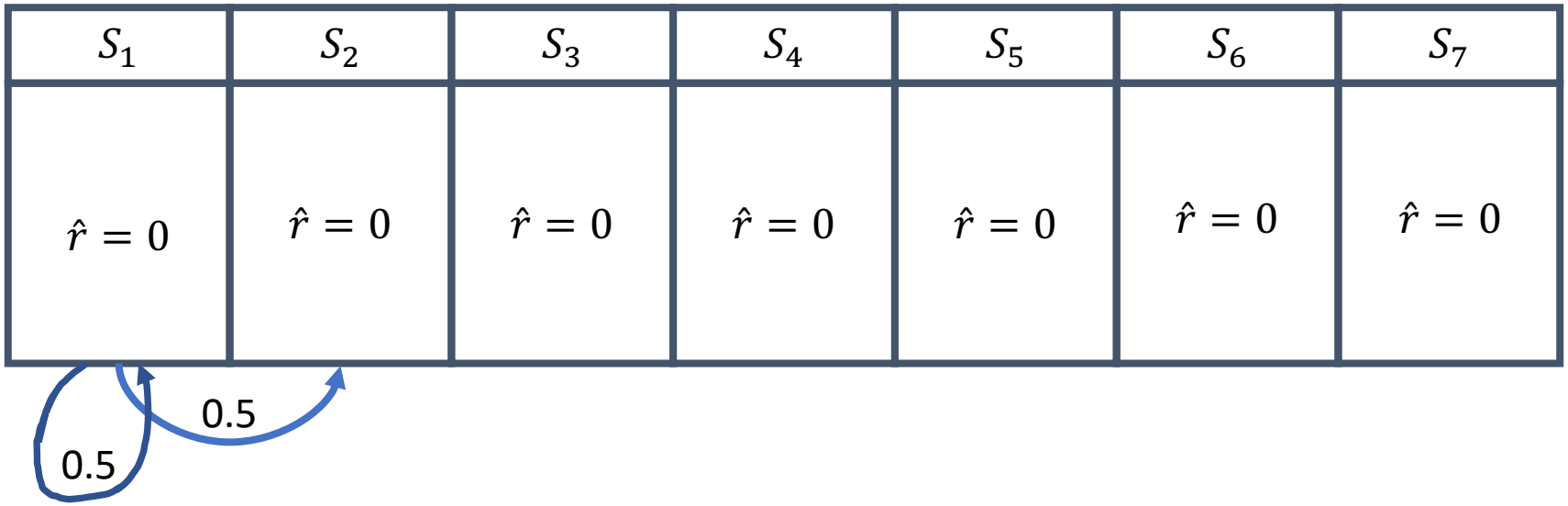
- Agent's representation of how the **world changes** in response to agent's action;
- **Transition / dynamics** model predicts next agent state

$$p(s_{t+1} = s' | s_t = s, a_t = a)$$

- **Reward** model predicts immediate reward (estimated):

$$r(s_t = s, a_t = a) = \mathbb{E}[\, r_t \mid s_t = s, a_t = a]$$

Example: Mars Rover as a Stochastic Markov Model



- Numbers above show RL agent's **reward** model
- The agent's transition model:
  - $0.5 = p(s_1|s_1, Right) = p(s_2|s_1, Right)$
  - $0.5 = p(s_2|s_2, Right) = p(s_3|s_2, Right)$
  - $\ldots$
- **Model may be wrong**

Policy

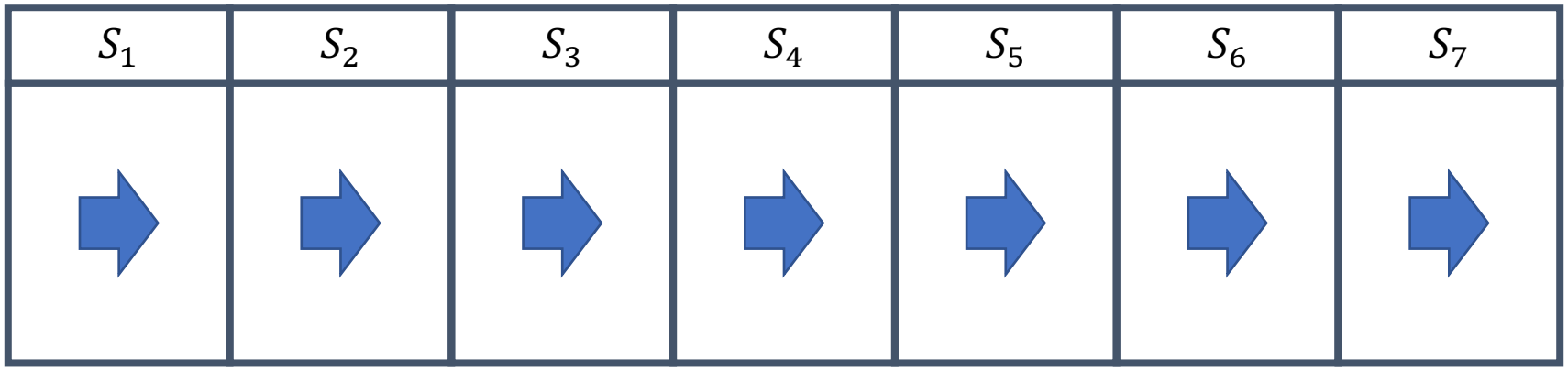
- Policy $\boldsymbol{\pi}$ determines how the agent chooses actions;
- $\pi : S \rightarrow A$, ***is mapping function from states to actions***
- **Deterministic** policy:

$$\pi(s) = a$$

- **Stochastic** policy:

$$\pi(a|s) = p(a_t = a|s_t = s)$$

Example: Mars Rover Policy



- Policy represented by arrow
- $\pi(s_1) = \pi(s_2) = \cdots = \pi(s_7) = right$
- Quick check:
  - is this a **deterministic** policy or a **stochastic** policy?

Value Function

- **Value function** V_π:
  is the expected discounted sum of future rewards following a particular policy π:

$$V_{\pi(s_t = s)} = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \mid s_t = s]$$

- **Discount factor** γ weighs immediate vs future rewards

- Can be used to **quantify goodness/badness** of states and actions:
  - Used to decide how to act by comparing policies

Example: Mars Rover as a Stochastic Markov Model

S_1	S_2	S_3	S_4	S_5	S_6	S_7

- Discount factor $\gamma = 0$;
- $\pi(s_1) = \pi(s_2) = \cdots = \pi(s_7) = right$
- Which are the values of $V_\pi(s)$ for this policy and this discount factor

$$V_{\pi(s_t = s)} = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \mid s_t = s]$$

Example: Mars Rover as a Stochastic Markov Model

S_1	S_2	S_3	S_4	S_5	S_6	S_7
$V_\pi(s_1) = 1$	$V_\pi(s_2) = 0$	$V_\pi(s_3) = 0$	$V_\pi(s_4) = 0$	$V_\pi(s_5) = 0$	$V_\pi(s_6) = 0$	$V_\pi(s_7) = 10$

- Discount factor $\gamma = 0$;
- $\pi(s_1) = \pi(s_2) = \cdots = \pi(s_7) = right$
- Numbers show value $V_\pi(s)$ for this policy and this discount factor

$$V_{\pi(s_t = s)} = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \mid s_t = s]$$

Types of RL Agents

- **Model-based:**
  - **Explicit**: **Model**
  - May or may not have policy and/or value function

- **Model-free:**
  - **Explicit: Value function** and/or policy function
  - No model

RL Agents

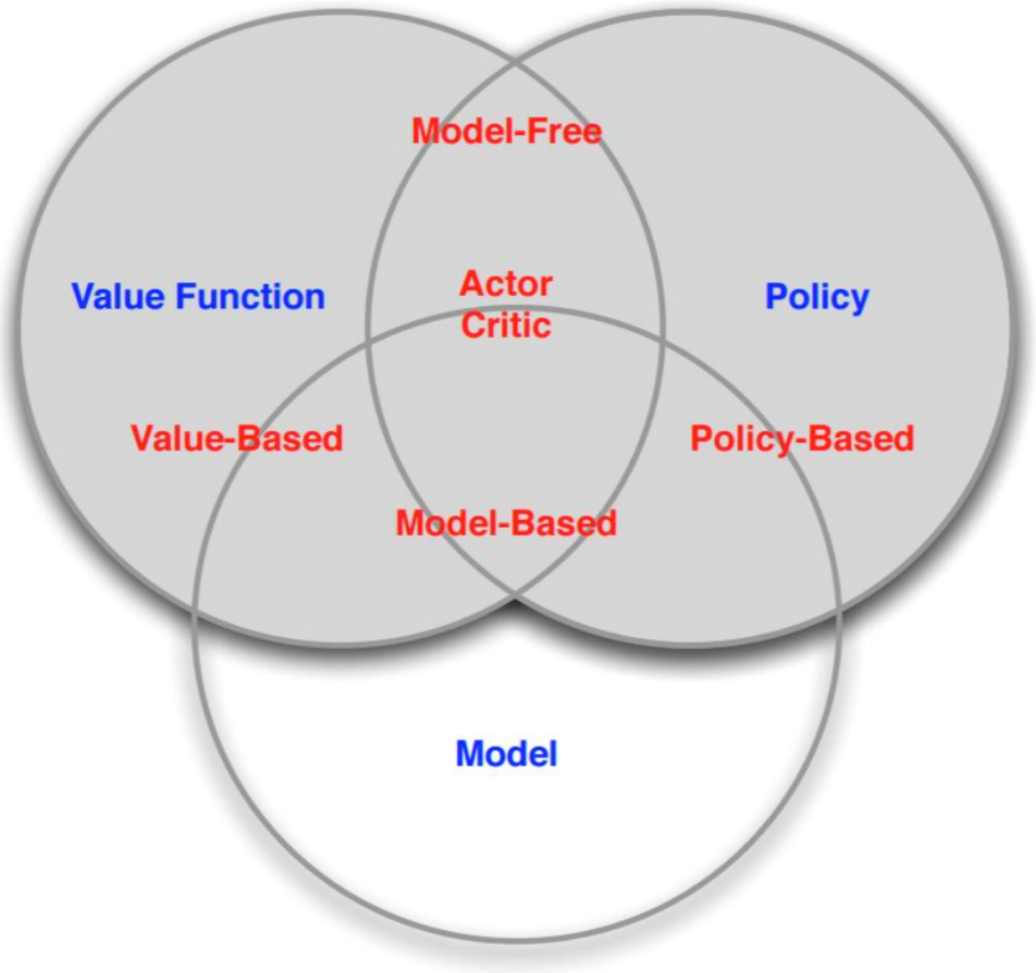


Figure: Figure from David Silver's RL course

Key Challenges in Learning to Make Sequences of Good Decisions

- In Planning (Agent's internal computation)
  - Given **model** of how the **world** works:
    Dynamics and reward model
  - Algorithm computes **how to act** in order to maximize expected reward:
    Without interacting with the real environment

Key Challenges in Learning to Make Sequences of Good Decisions

- Planning (Agent's internal computation)
  - Given **model** of how the **world** works
    - Dynamics and reward model
  - Algorithm computes **how to act** in order to maximize expected reward
    - With **no interaction** with real environment
- Reinforcement learning
  - Agent **doesn't know** how world works;
  - Interacts with world to **implicitly/explicitly learn** how it works ;
  - Agent improves **policy** (may involve planning)

Planning Example

- Solitaire: single player card game
- **Know** all **rules** of game / perfect model
- If take action *a* from state s
  - Can **compute probability distribution** over next state
  - Can **compute potential score**

- Can **plan ahead** to decide on optimal action
  - E.g. dynamic programming, tree search, ...

Reinforcement Learning Example

- **Solitaire** with no rule book
- **Learn** directly by taking actions and **seeing what happens**
- Try to **find** a good **policy** over time (that yields high reward)

Exploration and Exploitation

- Agent only **experiences** what happens for the **actions** it tries;
  - Mars rover trying to drive left learns the reward and next state for that action, but not for trying to drive right;
  - Obvious! But leads to a dilemma.

Exploration and Exploitation

- Agent only experiences what happens for the actions it tries

- How should an RL agent balance its actions?
  - **Exploration**: trying new things that might enable the agent to make better decisions in the future;

  - **Exploitation**: choosing actions that are expected to yield good reward given past experience;

- Often there may be an exploration-exploitation tradeoff (models)
  - May have to **sacrifice reward** in order to **explore** & learn about potentially better policy;

Exploration and Exploitation Examples

Movies:

- Exploitation: Watch a favorite movie you've seen before
- Exploration: Watch a new movie

Advertising:

- Exploitation: Show most effective ad so far
- Exploration: Show a different ad

Driving:

- Exploitation: Try fastest route given prior experience
- Exploration: Try a different route

Types of RL Agents: What the Agent (Algorithm) Learns

- **Model-based**
  - Explicit: Model
  - Implicit: Policy and/or value function (can use model to plan: compute a policy and/or value function)
- **Valued-based**
  - Explicit: Value function
  - Implicit: Policy (can derive a policy from value function)
- **Policy-based**
  - Explicit: Policy
  - No value function
- **Actor-Critic**
  - Explicit: Policy
  - Explicit: Value function

- Algorithms can also have an explicit model, value and policy.

Hide and Seek

https://openai.com/blog/emergent-tool-use/