Multi-Agent Reinforcement Learning

Games: Models of Multi-Agent Interaction

Stefano V. Albrecht, Filippos Christianos, Lukas Schäfer
Slides by: Leonard Hinckeldey

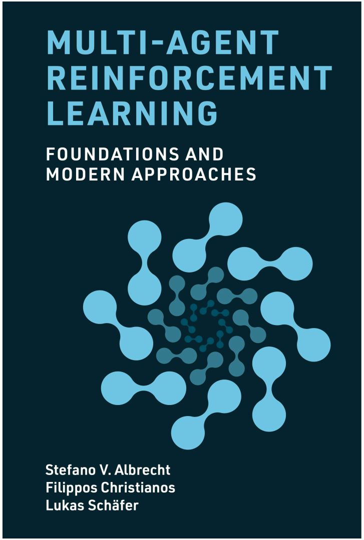
This lecture is based on

Multi-Agent Reinforcement Learning: Foundations and Modern Approaches

by Stefano V. Albrecht, Filippos Christianos and Lukas Schäfer

MIT Press, 2024

Download book, slides, and code at:
www.marl-book.com



Lecture Outline

Part 1: Game Models

- Normal-form games
- Stochastic games
- Partially observable stochastic games

Part 2: Modeling Communication

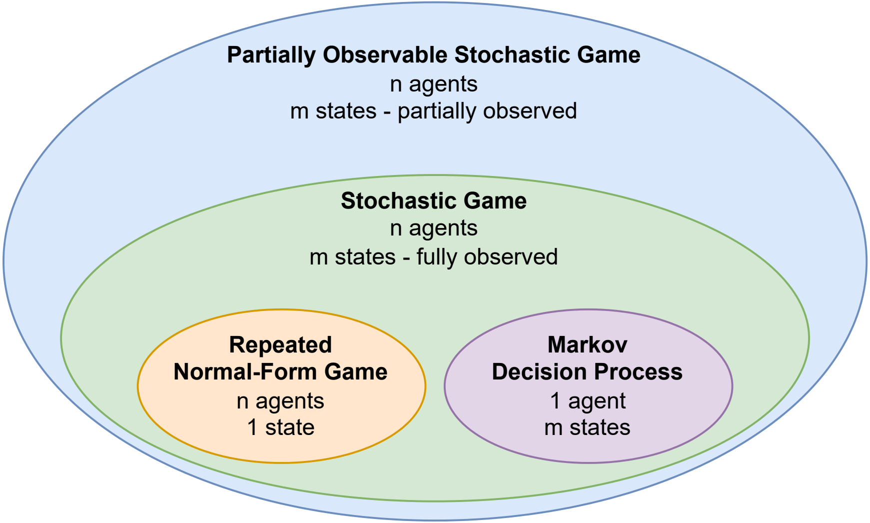
- Communication as an action
- Communication with observation functions

Part 3: Assumptions

- Game theory vs MARL assumptions

Game Models

Hierarchy of Games



Normal-Form Games

Normal-form games define a **single** interaction between two or more agents, providing a simple kernel for more general games to build upon.

Normal-Form Games

Normal-form games define a **single** interaction between two or more agents, providing a simple kernel for more general games to build upon.

Normal-form games are defined as a 3 tuple $(I, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I})$:

- I is a finite set of agents $I = \{1, ..., n\}$

Normal-Form Games

Normal-form games define a **single** interaction between two or more agents, providing a simple kernel for more general games to build upon.

Normal-form games are defined as a 3 tuple $(I, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I})$:

- I is a finite set of agents $I = \{1, ..., n\}$
- For each agent $i \in I$:
  - A_i is a finite set of actions
  - $\mathcal{R}_i$ is the reward function $\mathcal{R}_i : A \to \mathbb{R}$ where $A = A_1 \times ... \times A_n$ (set of **joint** actions).

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

1. Each agent samples an action $a_i \in A_i$ with probability $\pi_i(a_i)$

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

1. Each agent samples an action $a_i \in A_i$ with probability $\pi_i(a_i)$
2. The resulting actions from all agents form a **joint action**, $a = (a_1, ..., a_n)$

Normal-Form Game Process

In a normal-form game, there are **no time steps or states**. Agents choose an action and observe a reward.

The game proceeds as follows:

1. Each agent samples an action $a_i \in A_i$ with probability $\pi_i(a_i)$
2. The resulting actions from all agents form a **joint action**, $a = (a_1, ..., a_n)$
3. Each agent receives a reward based on its **individual** reward function and the **joint action**, $r_i = \mathcal{R}_i(a)$

Classes of Games

Games can be classified based on the relationship between the agents' reward functions.

- In **zero-sum games**, the sum of the agents' reward is always 0
  i.e. $\sum_{i \in I} \mathcal{R}_i(a) = 0, \forall a \in A$

Classes of Games

Games can be classified based on the relationship between the agents' reward functions.

- In **zero-sum games**, the sum of the agents' reward is always 0
  i.e. $\sum_{i \in I} \mathcal{R}_i(a) = 0, \forall a \in A$
- In **common-reward** games, all agents receive the same reward $(R_i = R_j; \forall i, j \in I)$

Classes of Games

Games can be classified based on the relationship between the agents' reward functions.

- In **zero-sum games**, the sum of the agents' reward is always 0 i.e. $\sum_{i \in I} \mathcal{R}_i(a) = 0, \forall a \in A$
- In **common-reward** games, all agents receive the same reward ($R_i = R_j; \forall i, j \in I$)
- In **general-sum** games, there are no restrictions on the relationship between reward functions.

Matrix Games

Normal-from games with 2 agents are also called **matrix games** because they can be represented using reward matrices.

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

Rock-Paper-Scissors

	A	B
A	10	0
B	0	10

Coordination Game

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoner's Dilemma

Normal-from games with 2 agents are also called **matrix games** because they can be represented using reward matrices.

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

Rock-Paper-Scissors
zero-sum

	A	B
A	10	0
B	0	10

Coordination Game

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoner's Dilemma

Matrix Games

Normal-from games with 2 agents are also called **matrix games** because they can be represented using reward matrices.

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

Rock-Paper-Scissors
zero-sum

	A	B
A	10	0
B	0	10

Coordination Game
common-reward

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoner's Dilemma

Normal-from games with 2 agents are also called **matrix games** because they can be represented using reward matrices.

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

Rock-Paper-Scissors
zero-sum

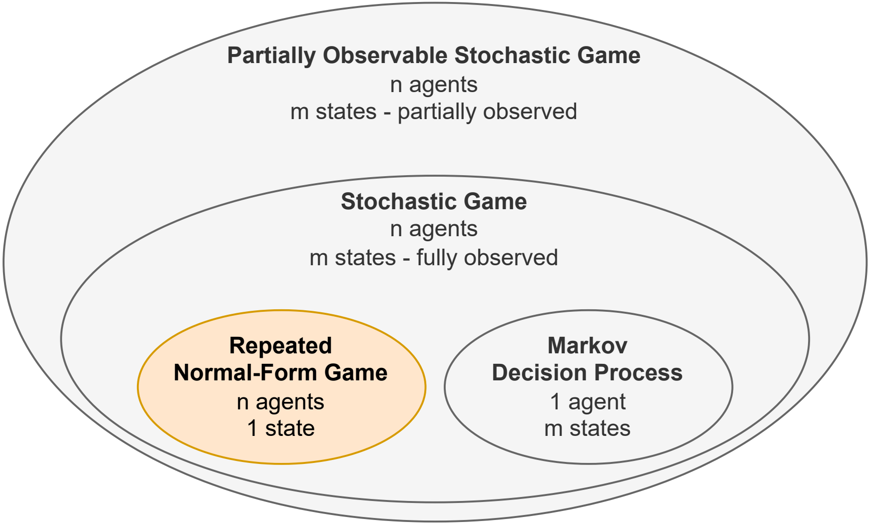
	A	B
A	10	0
B	0	10

Coordination Game
common-reward

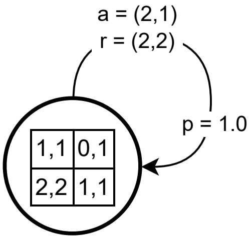
	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Prisoner's Dilemma
general-sum

Repeated Normal-Form Games

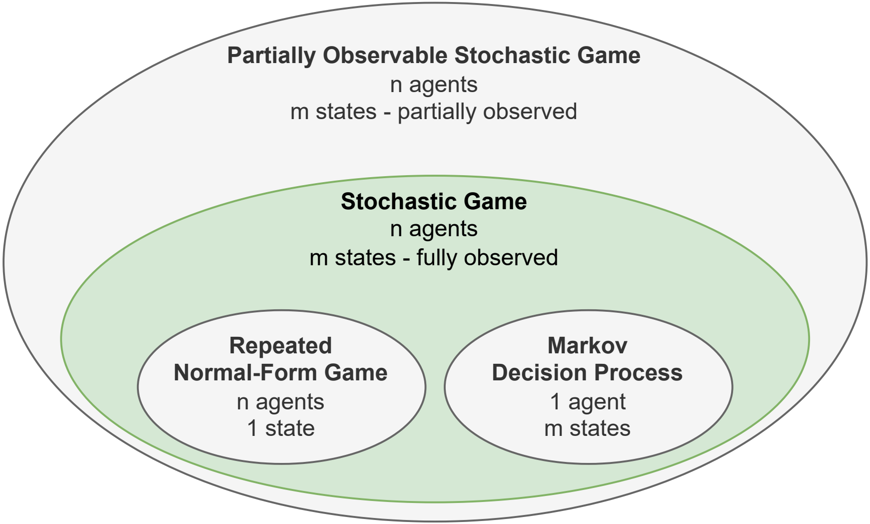


To extend normal-form games to **sequential** multi-agent interaction, we can repeat the same game over T timesteps.



- At each time step t an agent i samples an action a_i^t
- The policy is now conditioned on a **joint-action** history $\pi_i(a_i^t|h^t)$ where $h^t = (a^o, ..., a^{t-1})$
- In special cases, h^t contains n last joint actions. E.g. in a tit-for-tat strategy (Axelrod and Hamilton 1981), the policy is conditioned on a^{t-1}

Stochastic Games



Stochastic Games

Stochastic games introduce the notion of **states** and are defined as a 6 tuple $(I, S, \{A_i\}_{i\in I}, \{\mathcal{R}_i\}_{i\in I}, \mathcal{T}, \mu)$

Stochastic Games

Stochastic games introduce the notion of **states** and are defined as a 6 tuple $(I, S, \{A_i\}_{i\in I}, \{\mathcal{R}_i\}_{i\in I}, \mathcal{T}, \mu)$

- I is a finite set of agents

Stochastic Games

Stochastic games introduce the notion of **states** and are defined as a 6 tuple $(I, S, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I}, \mathcal{T}, \mu)$

- I is a finite set of agents
- S is a finite set of states with subset of terminal states $\bar{S} \subset S$

Stochastic Games

Stochastic games introduce the notion of **states** and are defined as a 6 tuple $(I, S, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I}, \mathcal{T}, \mu)$

- I is a finite set of agents
- S is a finite set of states with subset of terminal states $\bar{S} \subset S$
- For each agent $i \in I$:
  - A_i is a set finite set of actions
  - $\mathcal{R}_i$ is the reward function $\mathcal{R}_i : S \times A \times S \rightarrow \mathbb{R}$ where A is the set of **joint** actions $A = A_1 \times ... \times A_n$

Stochastic Games

Stochastic games introduce the notion of **states** and are defined as a 6 tuple $(I, S, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I}, \mathcal{T}, \mu)$

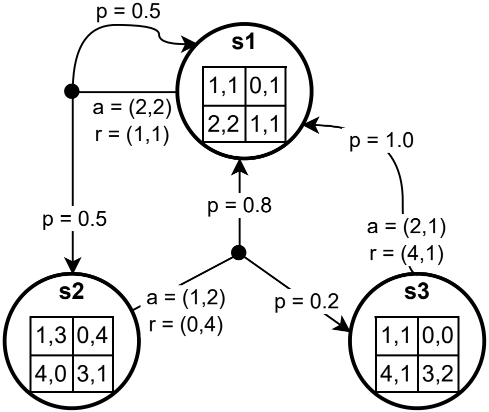
- I is a finite set of agents
- S is a finite set of states with subset of terminal states $\bar{S} \subset S$
- For each agent $i \in I$:
  - A_i is a set finite set of actions
  - $\mathcal{R}_i$ is the reward function $\mathcal{R}_i : S \times A \times S \to \mathbb{R}$ where A is the set of **joint** actions $A = A_1 \times ... \times A_n$
- $\mathcal{T}$ is the state transition function $\mathcal{T} : S \times A \times S \to [0, 1]$

Stochastic Games

Stochastic games introduce the notion of **states** and are defined as a 6 tuple $(I, S, \{A_i\}_{i \in I}, \{\mathcal{R}_i\}_{i \in I}, \mathcal{T}, \mu)$

- I is a finite set of agents
- S is a finite set of states with subset of terminal states $\bar{S} \subset S$
- For each agent $i \in I$:
  - A_i is a set finite set of actions
  - $\mathcal{R}_i$ is the reward function $\mathcal{R}_i : S \times A \times S \to \mathbb{R}$ where A is the set of **joint** actions $A = A_1 \times ... \times A_n$
- $\mathcal{T}$ is the state transition function $\mathcal{T} : S \times A \times S \to [0, 1]$
- μ is the initial state distribution $\mu : S \to [0, 1]$

Stochastic Games - Continued



- Each **state** can be viewed as a **non-repeated normal-form game**
- Stochastic games can also be classified into: zero-sum, common-reward or general-sum
- The figure on the left shows a general-sum case

Stochastic Game Process

1. Initial state $s_0 \in S$ is samples from μ.

Stochastic Game Process

1. Initial state $s_0 \in S$ is samples from μ.
2. At time t each agent $i \in I$ observes the current state $s^t \in S$ and chooses an action $a_i^t \in A_i$ with probability $\pi_i(a_i^t|h^t)$

Stochastic Game Process

1. Initial state $s_0 \in S$ is samples from μ.
2. At time t each agent $i \in I$ observes the current state $s^t \in S$ and chooses an action $a_i^t \in A_i$ with probability $\pi_i(a_i^t | h^t)$
   - $h^t = (s^0, a^0, s^1, a^1, ..., s^t)$ is the **state-action history** containing the current state s^t and past **joint** actions and states

Stochastic Game Process

1. Initial state $s_0 \in S$ is samples from μ.
2. At time t each agent $i \in I$ observes the current state $s^t \in S$ and chooses an action $a_i^t \in A_i$ with probability $\pi_i(a_i^t | h^t)$
   - $h^t = (s^0, a^0, s^1, a^1, ..., s^t)$ is the **state-action history** containing the current state s^t and past **joint** actions and states
3. Given s^t and a^t, the game transitions into the next state $s^{t+1} \in S$ with probability $\mathcal{T}(s^{t+1} | s^t, a^t)$

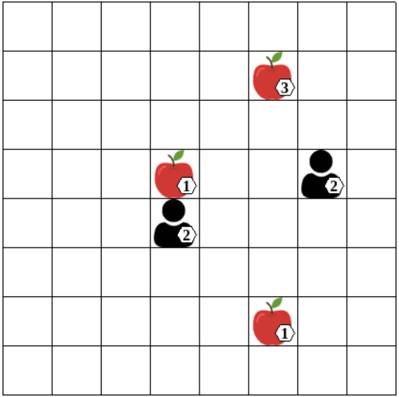
Stochastic Game Process

1. Initial state $s_0 \in S$ is samples from μ.
2. At time t each agent $i \in I$ observes the current state $s^t \in S$ and chooses an action $a_i^t \in A_i$ with probability $\pi_i(a_i^t|h^t)$
   - $h^t = (s^0, a^0, s^1, a^1, ..., s^t)$ is the **state-action history** containing the current state s^t and past **joint** actions and states
3. Given s^t and a^t, the game transitions into the next state $s^{t+1} \in S$ with probability $\mathcal{T}(s^{t+1}|s^t, a^t)$
4. Each agent receives a reward according to their reward function $r_i^t = \mathcal{R}_i(s^t, a^t, s^{t+1})$

Stochastic Game Process

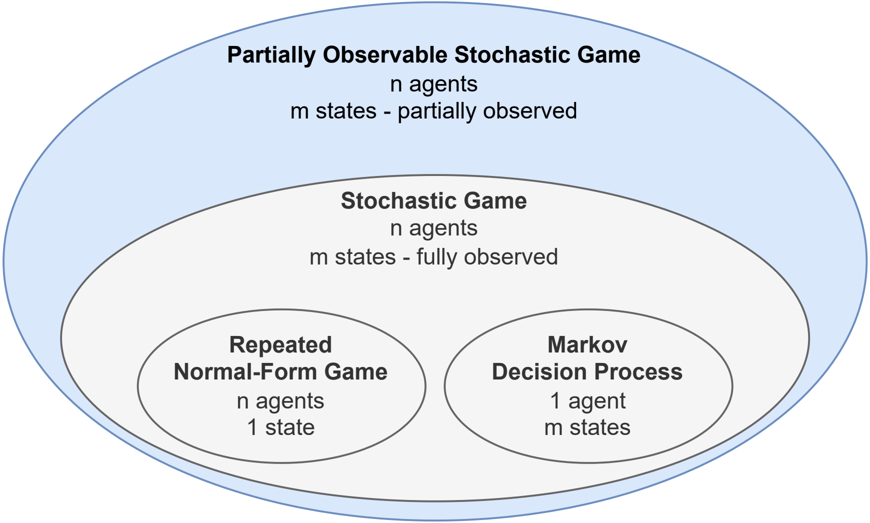
1. Initial state $s_0 \in S$ is samples from μ.
2. At time t each agent $i \in I$ observes the current state $s^t \in S$ and chooses an action $a_i^t \in A_i$ with probability $\pi_i(a_i^t | h^t)$
   - $h^t = (s^0, a^0, s^1, a^1, ..., s^t)$ is the **state-action history** containing the current state s^t and past **joint** actions and states
3. Given s^t and a^t, the game transitions into the next state $s^{t+1} \in S$ with probability $\mathcal{T}(s^{t+1} | s^t, a^t)$
4. Each agent receives a reward according to their reward function $r_i^t = \mathcal{R}_i(s^t, a^t, s^{t+1})$
5. These steps are repeated until a terminal state $s^t \in \bar{S}$ is reached or a maximum number of T time steps is completed

Example: Level-Based Foraging



- $s \in S$: vector of x-y positions for agents/items, binary collection flags, levels for agents/items
- $a_i \in A_i$: move in four directions, collect item, or no operation (noop)
- $\mathcal{T}$: joint actions update state, e.g., two agents collecting an item switch its flag
- $\mathcal{R}$:
  - common-reward: +1 reward for any item collected by any agent
  - general-sum: +1 reward only for agents directly involved in item collection

Partially Observable Stochastic Games (POSG)



Partially Observable Stochastic Games (POSG)

At the top of the game model hierarchy, the most **general** model is the POSG

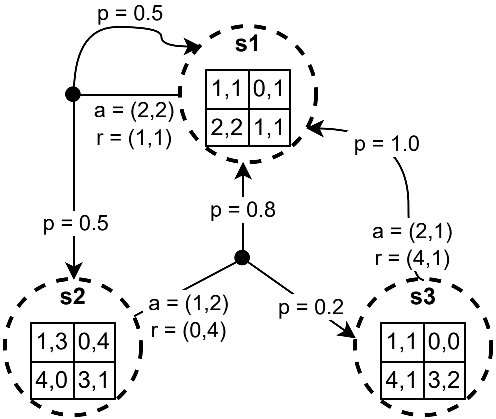
- POSGs represent complex decision processes with **incomplete information**
- Unlike in stochastic games, agents receive **observations** providing **incomplete information** about the state and agents' actions
- POSGs apply to scenarios where agents have limited sensing capabilities

  $\Rightarrow$ e.g. autonomous driving

  $\Rightarrow$ e.g. strategic games (e.g. card games) with private, hidden information

POSG Definition

POSG is defined in the same way stochastic games are, with two additions. Thus it is defined as a 8 tuple $(I, S, \{A_i\}_{i\in I}, \{\mathcal{R}_i\}_{i\in I}, \mathcal{T}, \mu, \{O_i\}_{i\in I}, \{\mathcal{O}_i\}_{i\in I})$



For each agent *i* we additionally define:

- a finite set of observation O_i
- an observation function $\{\mathcal{O}_i\}_{i\in I}$ such that $\mathcal{O}_i : A \times S \times O_i \to [0,1]$ and $\forall a \in A, s \in S : \sum_{o_i \in O_i} \mathcal{O}_i(a, s, o_i) = 1$

POSG Process

1. Initial state s^0 sampled from $\mu(s)$

POSG Process

1. Initial state s^0 sampled from $\mu(s)$
2. At each t every agent i receives an **observation** $o_i^t \in O_i$, with probability given by the observation function $\mathcal{O}_i(o^t|s^t, a^{t-1})$

POSG Process

1. Initial state s^0 sampled from $\mu(s)$
2. At each t every agent i receives an **observation** $o_i^t \in O_i$, with probability given by the observation function $\mathcal{O}_i(o^t|s^t, a^{t-1})$
3. Each agent then chooses an action a_i^t based on its policy π_i, which is conditioned on the agent's observation history, $h_i = (o_i^0, ..., o_i^t)$. All agents' actions give the joint action $a^t = (a_1^t, ..., a_n^t)$

POSG Process

1. Initial state s^0 sampled from $\mu(s)$
2. At each t every agent i receives an **observation** $o_i^t \in O_i$, with probability given by the observation function $\mathcal{O}_i(o^t | s^t, a^{t-1})$
3. Each agent then chooses an action a_i^t based on its policy π_i, which is conditioned on the agent's observation history, $h_i = (o_i^0, ..., o_i^t)$. All agents' actions give the joint action $a^t = (a_1^t, ..., a_n^t)$
4. Given the joint action, the environment transitions into the next state s^{t+1} with probability $\mathcal{T}(s^{t+1} | s^t, a^t)$ and each agent receives a reward based on its reward function $r_i^t = \mathcal{R}_i(s^t, a^t, s^{t+1})$

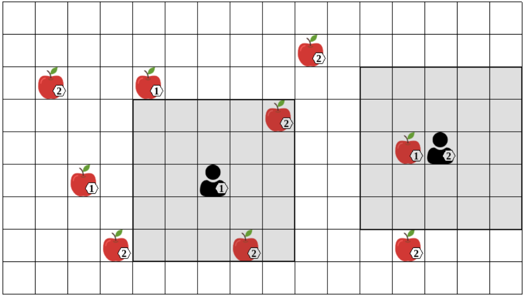
POSG Process

1. Initial state s^0 sampled from $\mu(s)$
2. At each t every agent i receives an **observation** $o_i^t \in O_i$, with probability given by the observation function $\mathcal{O}_i(o^t|s^t, a^{t-1})$
3. Each agent then chooses an action a_i^t based on its policy π_i, which is conditioned on the agent's observation history, $h_i = (o_i^0, ..., o_i^t)$. All agents' actions give the joint action $a^t = (a_1^t, ..., a_n^t)$
4. Given the joint action, the environment transitions into the next state s^{t+1} with probability $\mathcal{T}(s^{t+1}|s^t, a^t)$ and each agent receives a reward based on its reward function $r_i^t = \mathcal{R}_i(s^t, a^t, s^{t+1})$
5. This is done until a terminating state $s^t \in \bar{S}$ is reached or a maximum number of time steps is completed

The Observation Function

POSG can represent diverse observability conditions. For example:

- modeling noise by adding uncertainty in the possible observation
- to limit the visibility region of agents (see LBF example)

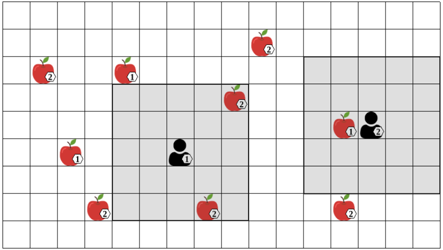


- Here, the agent can only see some parts of the state and joint action
- $o_i^t = (\bar{s}^t, \bar{a}^t)$ where $\bar{s}^t \subset s^t, \bar{a}^t \subset a^t$

Belief States

In partially observable settings, it becomes more challenging to infer optimal actions. For example:

- Optimal action for agent 1 is to move left towards level 1 apple
- But level 1 apple is not directly observable
- Agent 1 can hold a **belief state** b_i^t, a probability distribution over possible state $s \in S$
- Agent 1 might have seen the level-1 apple previously and can thus 'remember' its location



Single Agent Belief Update

To simplify, let's consider the single-agent perspective:

- The initial belief state is given by $b_i^0 = \mu$
- After taking action a_i^t and observing o_i^{t+1}, the belief state b_i^t is updated to b_i^{t+1} using a Bayesian update:

$$b_i^{t+1}(s') \propto \sum_{s \in S} b_i^t(s) \mathcal{T}(s'|s, a_i^t) \mathcal{O}_i(o_i^{t+1}|a_i^t, s')$$

Single Agent Belief Update

To simplify, let's consider the single-agent perspective:

- The initial belief state is given by $b_i^0 = \mu$
- After taking action a_i^t and observing o_i^{t+1}, the belief state b_i^t is updated to b_i^{t+1} using a Bayesian update:

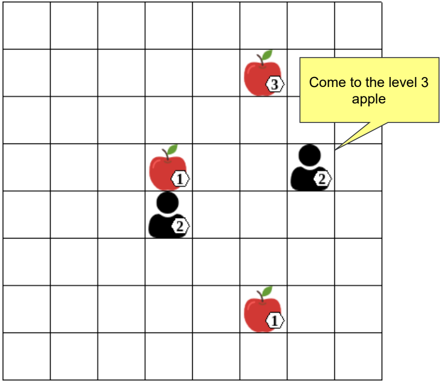
$$b_i^{t+1}(s') \propto \sum_{s \in S} b_i^t(s)\mathcal{T}(s'|s, a_i^t)\mathcal{O}_i(o_i^{t+1}|a_i^t, s')$$

In MARL this type of update is typically **infeasible**:

- High-dimensional state spaces make storage and updates of beliefs intractable
- In MARL for POSG, agents assumed not to know $(S, \mathcal{T}, \mathcal{O}_i)$
- Deep learning can be used to approximate state information (see later lectures)

Modeling Communication

Modeling Communication



- Using games, we can model more complex agent interactions, such as communication
- We can view communication as a type of action that other agents can observe without affecting the state of the environment
- Agents learn communication meanings through trials and observations, identical to environment actions
- This can lead to the evolution of a shared language or protocol

Communication Actions

To model communication, we can extend the action set of agents:

$$A_i = X_i \times M_i$$

- Where M_i is a set of possible messages $\{m1, m2, m3, ...\}$ and X_i is the set of environment actions
- The action a_i can thus be expressed as $(x_i, m_i) \in A_i$

Communication in Stochastic Games

- Agents observe the current state s_t and previous joint action a_{t-1}
- Communication action m^i_{t-1} by agent i is part of a_{t-1} and observed by all agents
- State transitions are independent of the joint communication actions $M = \times_{i \in I} M_i$

$$\forall s, s' \in S \forall a \in A, m \in M : T(s'|s, a) = T(s'|s, \langle (a_1, m_1), \ldots, (a_n, m_n) \rangle)$$

Communication in POSG

- In POSG we can use the observation function $\mathcal{O}_i$ to model noisy or unreliable communication
- We can define the observation as $o_i^t = [\bar{s}^t, w_1^{t-1}, ..., w_n^{t-1}]$
  - $\bar{s}^t$ is some partial information about the state
  - w_j^{t-1} is a message from the agent j at time step $t-1$ which has been augmented by $\mathcal{O}_i$
  - E.g. $w_j^{t-1} = f(m_j^{t-1})$ where $f(m_j^{t-1}) = m_j^{t-1} + \eta$, and η is some random noise component.
- You could also model $\mathcal{O}_i$ to hide messages such that $w_1^{t-j} = \emptyset$ if agent i is too far from agent j

Assumptions in Games

Game Theory Assumption

- In game theory, we typically assume that all agents know all components of the game **(complete knowledge games)**
- Agents know all agents' **action spaces and reward functions**
- Knowledge of other agents' reward functions may be used for informing the agent's **best response** action (we will cover this in more depth in the next lecture)
- Knowledge of the transition function (T) allows for predicting state changes and **planning** actions multiple steps ahead

MARL Assumptions

- In MARL, we assume **limited knowledge**, i.e. no knowledge of transition function $\mathcal{T}$ and no knowledge of agents' reward functions $\mathcal{R}_i$
- Additional assumption can be added and specific knowledge of the game can be held **mutually** or **asymmetrically**
- We usually assume the **number of agents to be fixed**, although recent research has looked at *open* multi-agent systems, this will not be covered in these lectures

Dictionary: Reinforcement Learning $\leftrightarrow$ Game Theory

RL		GT
environment	$\leftrightarrow$	game
agent	$\leftrightarrow$	player
reward	$\leftrightarrow$	payoff, utility
policy	$\leftrightarrow$	strategy
deterministic X	$\leftrightarrow$	pure X
probabilistic X	$\leftrightarrow$	mixed X
joint X	$\leftrightarrow$	X profile

- **Environment/Game**: Model with actions, observations, rewards, state dynamics.
- **Agent/Player**: Decision-maker, possibly with specific roles.
- **Reward/Payoff, Utility**: Scalar value received after an action
- **Policy/Strategy**: Assigns probabilities to actions; 'pure strategy' may refer to actions
- **Deterministic X/Pure X**: Assigns probability 1 to X e.g. X = equilibrium or policy
- **Probabilistic X/Mixed X**: Assigns probabilities ≤ 1 to X
- **Joint X/X Profile**: Tuple representing collective aspects, e.g., rewards or policies

Summary

We covered:

- Game models
- Modelling agent communication
- Assumptions of game models

Next we’ll cover:

- Solution concepts for games