




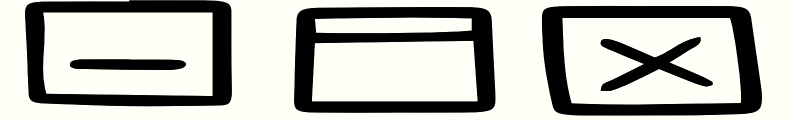
Reinforcement Learning and Autonomous Systems (AI 4102)



Lecture 2 (18/08/2023)
Lecture 3 (21/08/2023)

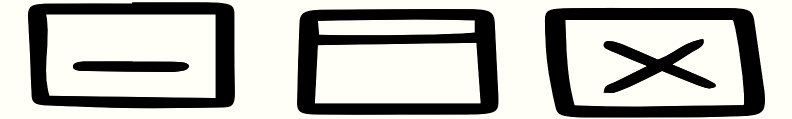
Instructor: Gourav Saha

Lecture Content



- Stochastic process (very brief introduction; only that what is required for this course).
- Markov Decision Process and Reinforcement Learning.
- Special Cases of Markov Decision Process.
- Stationary vs Non-Stationary Process.
- Objective functions in Reinforcement Learning.
- Reinforcement Learning vs

Lecture Content



- Stochastic process (very brief introduction; only that what is required for this course).
 - Visualizing stochastic process using Probabilistic Graphical Model.
 - Markov process.
- Markov Decision Process and Reinforcement Learning.
- Special Cases of Markov Decision Process.
- Stationary vs Non-Stationary Process.
- Objective functions in Reinforcement Learning.
- Reinforcement Learning vs

Stochastic Process



- A stochastic process is simply a sequence of random variables X_1, X_2, X_3, \dots . The short hand notation of stochastic process is $\{X_t\}_{t=1}^T$
- In this course, the index t of the random variable is **time** (hence the word “process” in stochastic process).
- Example of stochastic process:
 - The number of customers coming to a supermarket every hour of a day.
 - Daily weather (like temperature, humidity, precipitation etc).
 - Stock prices.

Stochastic Process

- A stochastic process is characterized by the **joint probability distribution**:

$$p(x_1, x_2, \dots, x_T)$$

- Now we can write:

- $p(x_1, x_2, \dots, x_T)$

$$= p(x_T | x_1, x_2, \dots, x_{T-1}) \underbrace{p(x_1, x_2, \dots, x_{T-1})}_{\text{blue}}$$

- $\underbrace{p(x_1, x_2, \dots, x_{T-1})}_{\text{blue}}$

$$= p(x_{T-1} | x_1, x_2, \dots, x_{T-2}) \underbrace{p(x_1, x_2, \dots, x_{T-2})}_{\text{blue}}$$

•
•
•

- $\underbrace{p(x_1, x_2)}_{\text{blue}}$

$$= p(x_2 | x_1) p(x_1)$$

Stochastic Process

Probabilistic Graphical Model (PGM)

- PGMs are **graphical representation** of the **conditional distribution** function.
- The **arrows** in the PGM represents the conditional distribution.

- A stochastic process is characterized by the **joint probability distribution**:

$$p(x_1, x_2, \dots, x_T)$$

- Now we can write:

- $p(x_1, x_2, \dots, x_T)$

$$= p(x_T | x_1, x_2, \dots, x_{T-1}) p(x_1, x_2, \dots, x_{T-1})$$

- $p(x_1, x_2, \dots, x_{T-1})$

$$= p(x_{T-1} | x_1, x_2, \dots, x_{T-2}) p(x_1, x_2, \dots, x_{T-2})$$

•
•
•

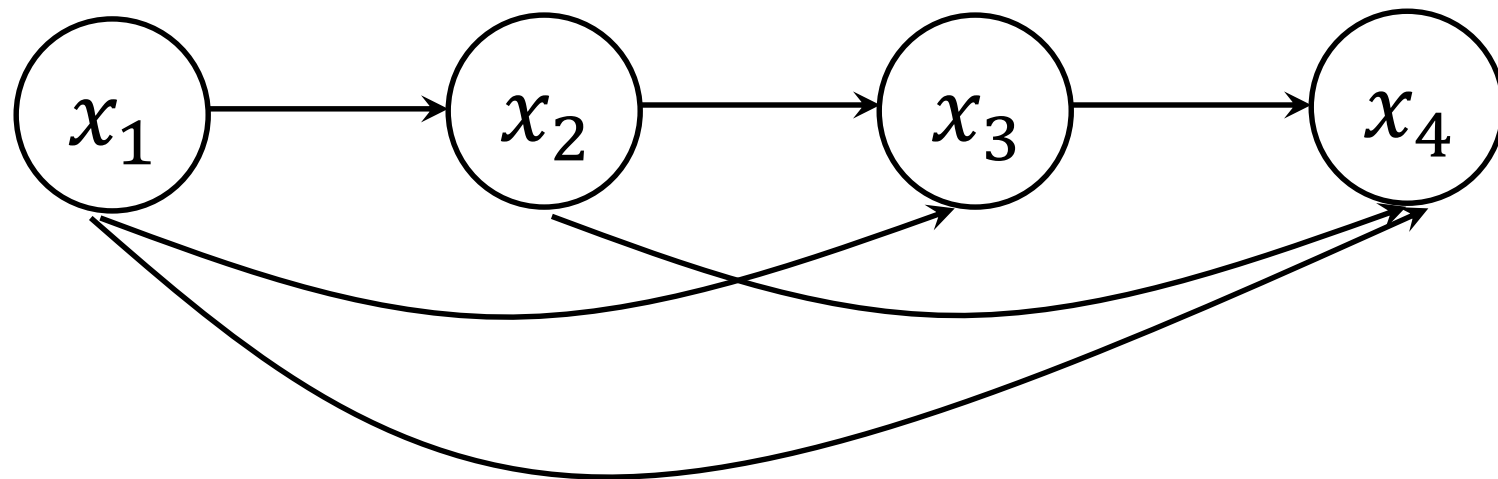
- $p(x_1, x_2)$

$$= p(x_2 | x_1) p(x_1)$$

Stochastic Process

Probabilistic Graphical Model (PGM)

- PGMs are **graphical representation** of the **conditional distribution** function.
- The **arrows** in the PGM represents the conditional distribution.



- A stochastic process is characterized by the **joint probability distribution**:

$$p(x_1, x_2, \dots, x_T)$$

- Let $T = 4$. We have,

- $p(x_1, x_2, x_3, x_4)$
 $= p(x_4 | x_1, x_2, x_3) p(x_1, x_2, x_3)$

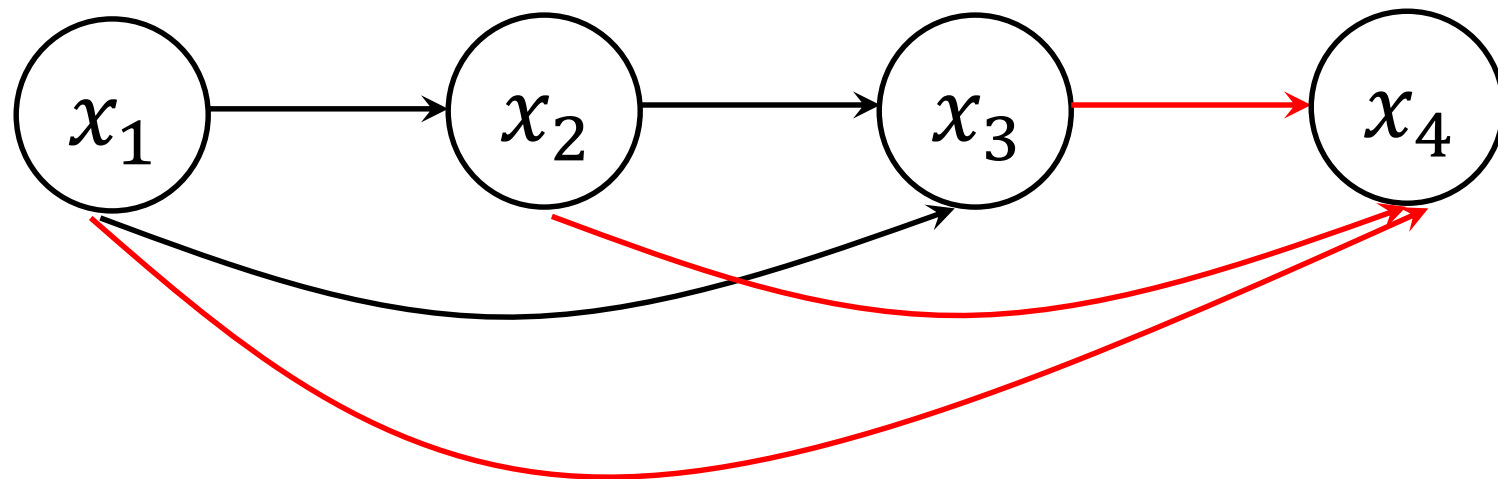
- $p(x_1, x_2, x_3)$
 $= p(x_3 | x_1, x_2) p(x_1, x_2)$

- $p(x_1, x_2)$
 $= p(x_2 | x_1) p(x_1)$

Stochastic Process

Probabilistic Graphical Model (PGM)

- PGMs are **graphical representation** of the **conditional distribution** function.
- The **arrows** in the PGM represents the conditional distribution.



The **red arrows** shows that x_4 is conditionally dependent on x_1 , x_2 , and x_3 .

- A stochastic process is characterized by the **joint probability distribution**:

$$p(x_1, x_2, \dots, x_T)$$

- Let $T = 4$. We have,

- $p(x_1, x_2, x_3, x_4)$
 $= p(x_4 | x_1, x_2, x_3) p(x_1, x_2, x_3)$

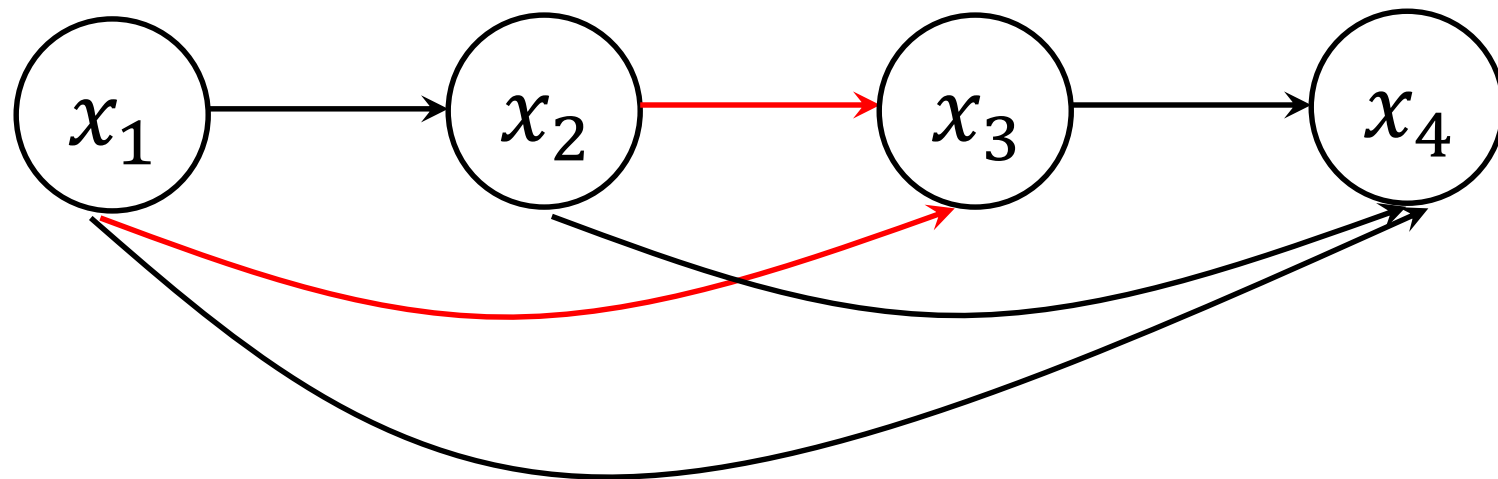
- $p(x_1, x_2, x_3)$
 $= p(x_3 | x_1, x_2) p(x_1, x_2)$

- $p(x_1, x_2)$
 $= p(x_2 | x_1) p(x_1)$

Stochastic Process

Probabilistic Graphical Model (PGM)

- PGMs are **graphical representation** of the **conditional distribution** function.
- The **arrows** in the PGM represents the conditional distribution.



The **red arrows** shows that x_3 is conditionally dependent on x_1 and x_2 .

- A stochastic process is characterized by the **joint probability distribution**:

$$p(x_1, x_2, \dots, x_T)$$

- Let $T = 4$. We have,

- $p(x_1, x_2, x_3, x_4)$
 $= p(x_4 | x_1, x_2, x_3) p(x_1, x_2, x_3)$

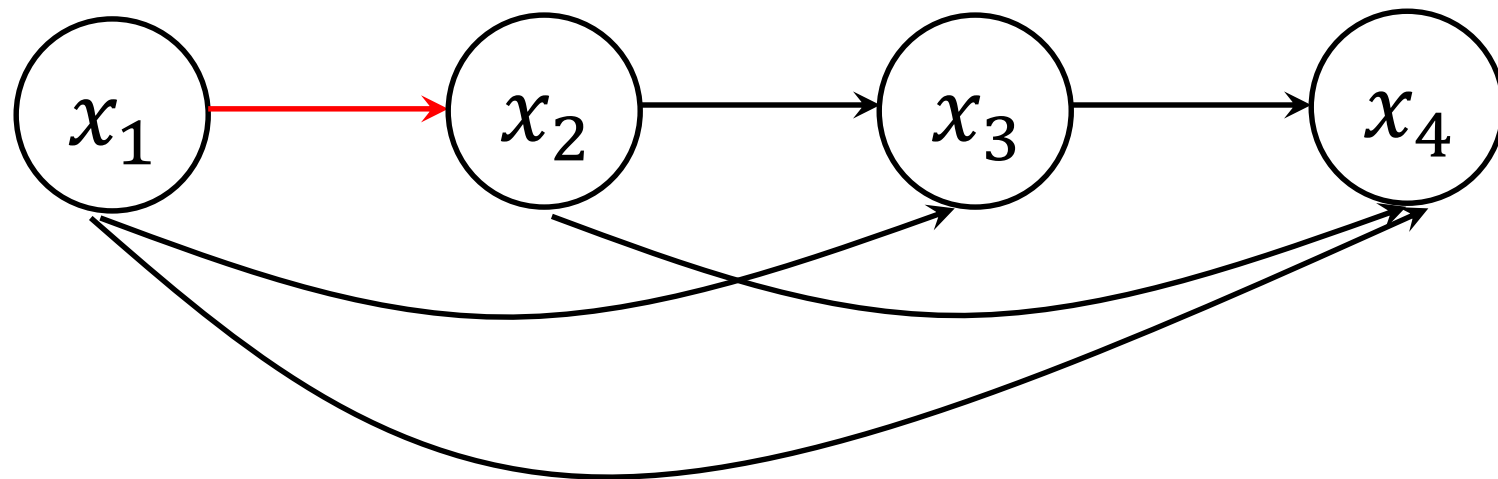
- $p(x_1, x_2, x_3)$
 $= p(x_3 | x_1, x_2) p(x_1, x_2)$

- $p(x_1, x_2)$
 $= p(x_2 | x_1) p(x_1)$

Stochastic Process

Probabilistic Graphical Model (PGM)

- PGMs are **graphical representation** of the **conditional distribution** function.
- The **arrows** in the PGM represents the conditional distribution.



The **red arrow** shows that x_2 is conditionally dependent on x_1 .

- A stochastic process is characterized by the **joint probability distribution**:

$$p(x_1, x_2, \dots, x_T)$$

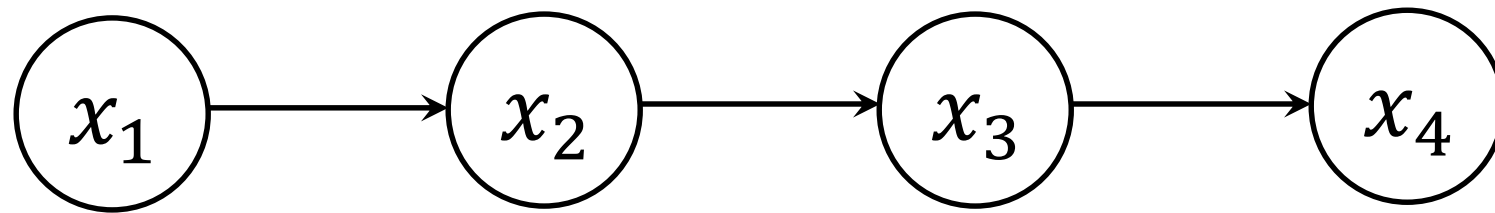
- Let $T = 4$. We have,

- $p(x_1, x_2, x_3, x_4)$
 $= p(x_4 | x_1, x_2, x_3) p(x_1, x_2, x_3)$

- $p(x_1, x_2, x_3)$
 $= p(x_3 | x_1, x_2) p(x_1, x_2)$

- $p(x_1, x_2)$
 $= p(x_2 | x_1) p(x_1)$

Stochastic Process

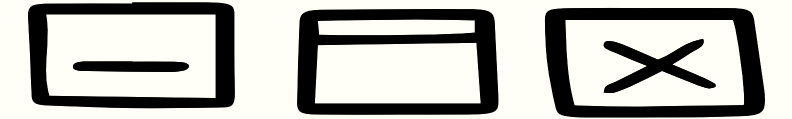


- A Markov process is a stochastic process where X_{T+1} is independent of x_1, x_2, \dots, x_{T-1} if x_T is given. Mathematically,

$$p(x_{T+1} | x_1, x_2, \dots, x_{T-1}, x_T) = p(x_{T+1} | x_T)$$

- In a simple language, the **future states** are independent of the **past states** **GIVEN** the **present state**.
- The PGM of a Markov process is shown on the left.
 - Unlike a general stochastic process, notice that the **only incoming arrow** for x_t is from x_{t-1} . This is an indication of the **Markovian property**.

Lecture Content



- Stochastic process (very brief introduction; only that what is required for this course).
- Markov Decision Process and Reinforcement Learning.
 - Components of an MDP.
 - Policies.
 - Markov Decision Process vs Reinforcement Learning.
- Special Cases of Markov Decision Process.
- Stationary vs Non-Stationary Process.
- Objective functions in Reinforcement Learning.
- Reinforcement Learning vs

NOTATION WARNING

Going forward, please be aware of the following:

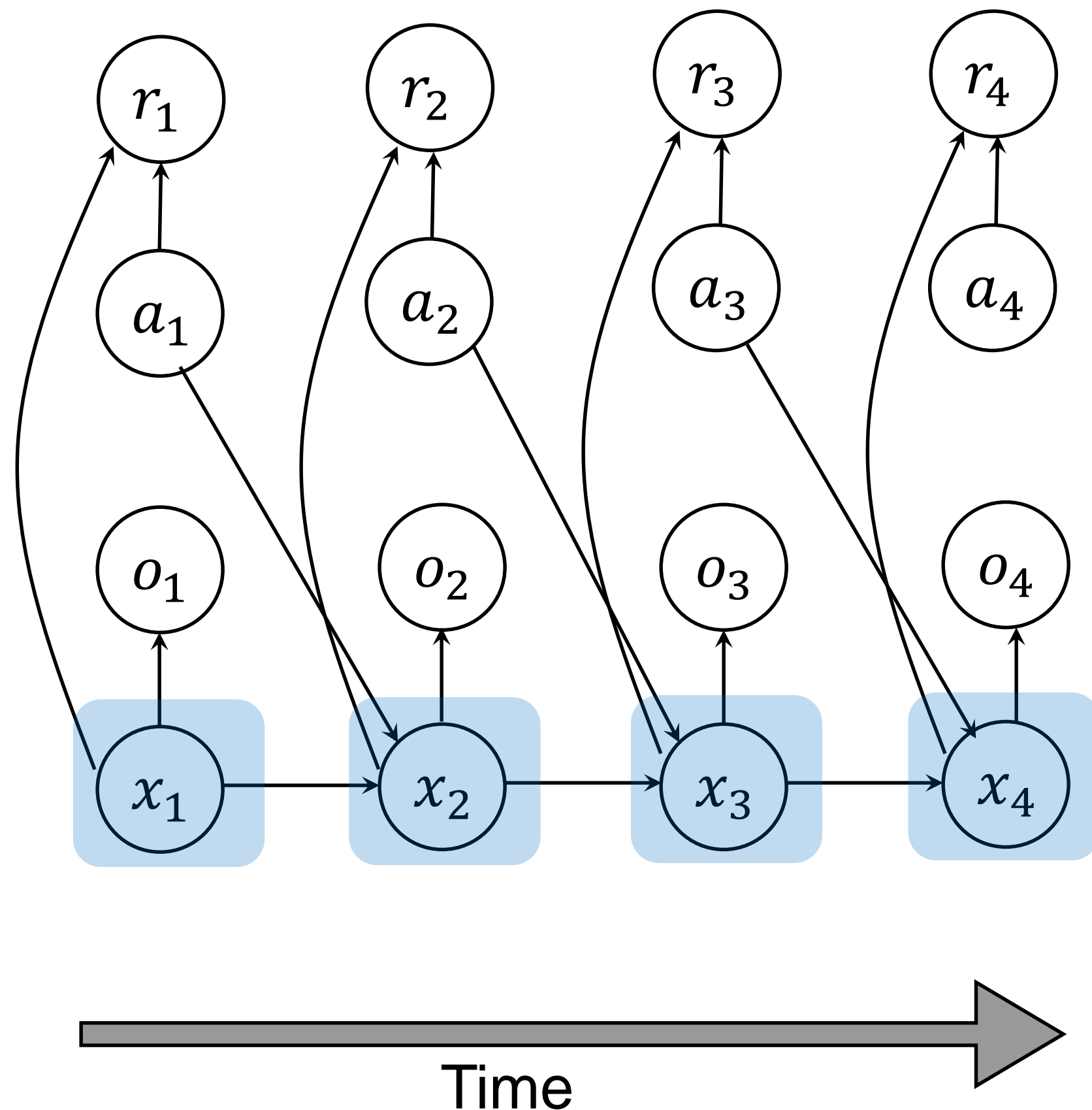
During lectures, I used s_t to denote “states”. In the slides, I have changed it to x_t because it is **more conventional**.

Also, there may be few slides where I might have used s_t instead of x_t by mistake. These are typos and my apologies for that in advance!

Markov Decision Process and Reinforcement Learning

- A Markov Decision Process (MDP) is an extension of Markov process with two main differences:
 - **Actions:** There is an agent who has to take actions (decisions). Unlike Markov process where the next state is dependent only on the current state, in a MDP, the **next state is dependent on the current state and the current action.**
 - **Rewards:** The agent earns a reward based on its actions and the objective is to maximize the reward (in some sense).

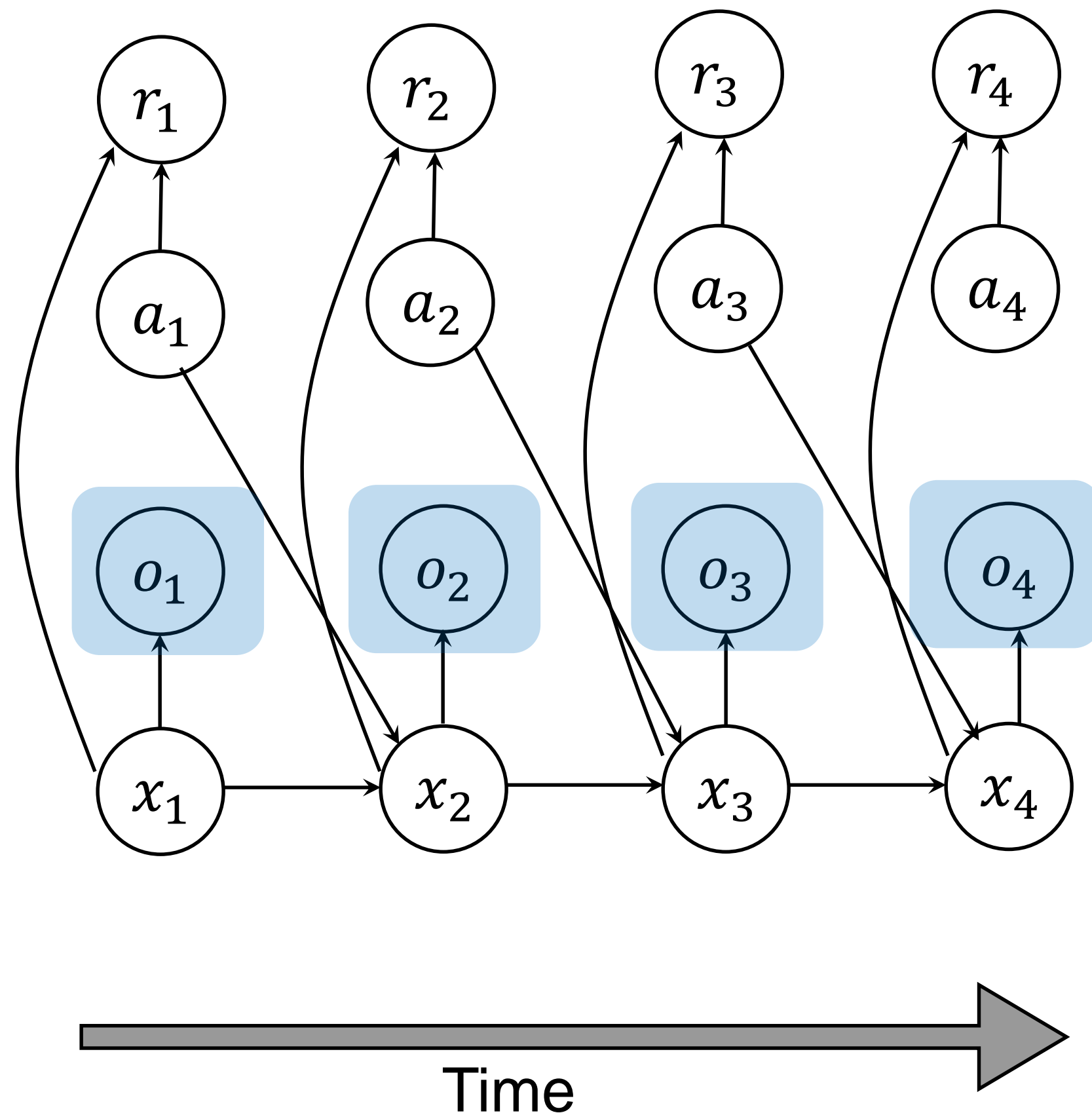
Markov Decision Process and Reinforcement Learning



- The PGM of a Markov Decision Process* (MDP) is shown in the left.
- An MDP* has the following components:
 1. **States:** It captures the dynamics of the physical world that are **relevant for making decisions**.
 - The importance of the phrase highlighted in blue can be explained using the following example. As such, the moon is imparting some force on a car on road. But since it is so negligible, we can ignore it. Hence, we don't need to capture the dynamics of the moon in order to design autonomous cars 😊!
 - Knowing what the states are requires **domain knowledge**.

*Strictly speaking, a Partially Observable MDP (POMDP). The difference between MDP and POMDP is explained after a few slides.

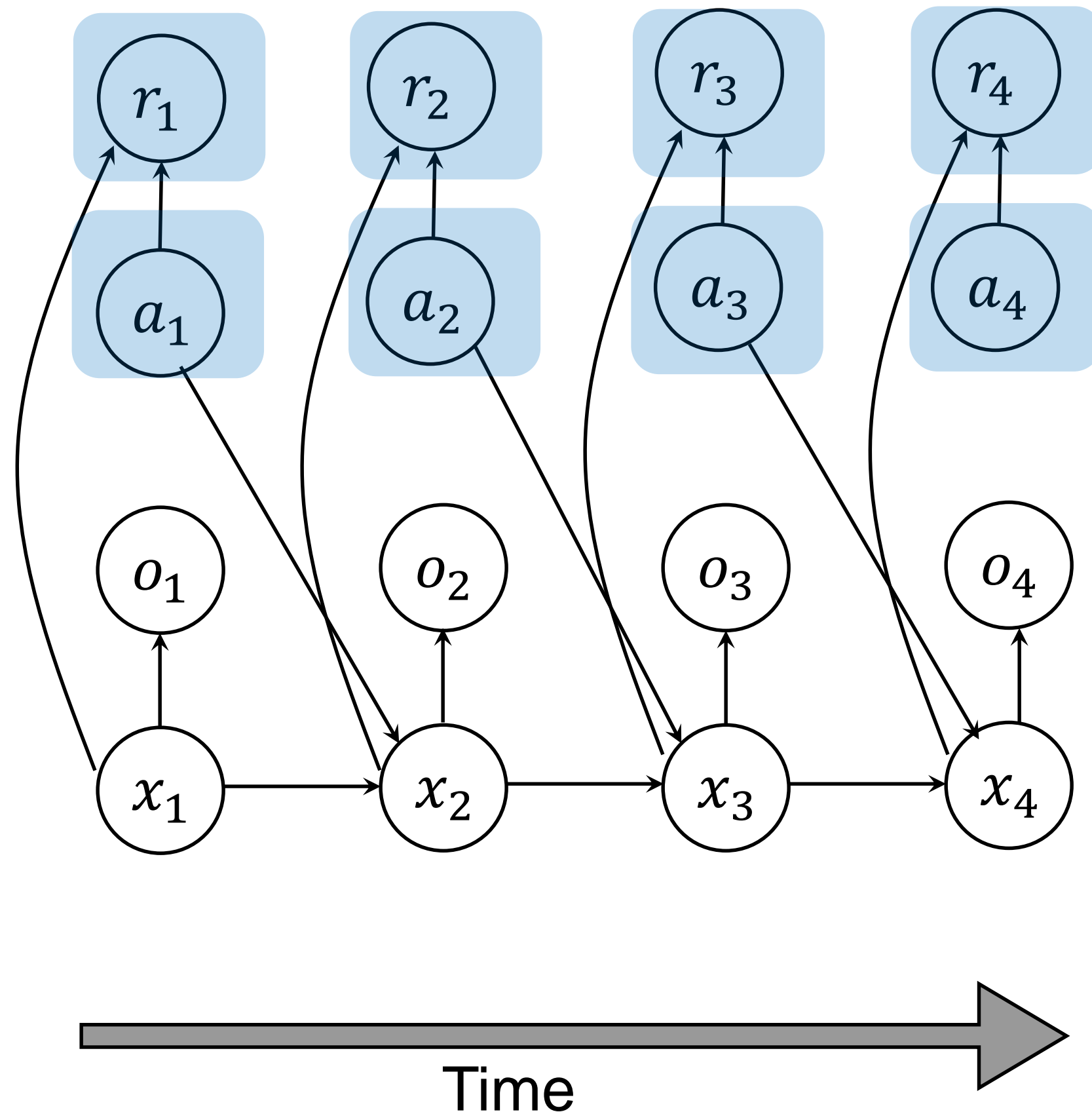
Markov Decision Process and Reinforcement Learning



- The PGM of a Markov Decision Process* (MDP) is shown in the left.
- An MDP* has the following components:
 2. **Observations:** Basically, whatever an agent observes in order to make decisions. The agent can only observe certain components of the states because of various limitations, e.g. In autonomous cars, we may only have the knowledge of the **positions** of the other vehicles and NOT their **velocities** because of sensor limitations.
 - Sometimes, because lack of domain knowledge, the agent may not observe a state because it does not know that it is relevant for decision making.

*Strictly speaking, a Partially Observable MDP (POMDP). The difference between MDP and POMDP is explained after a few slides.

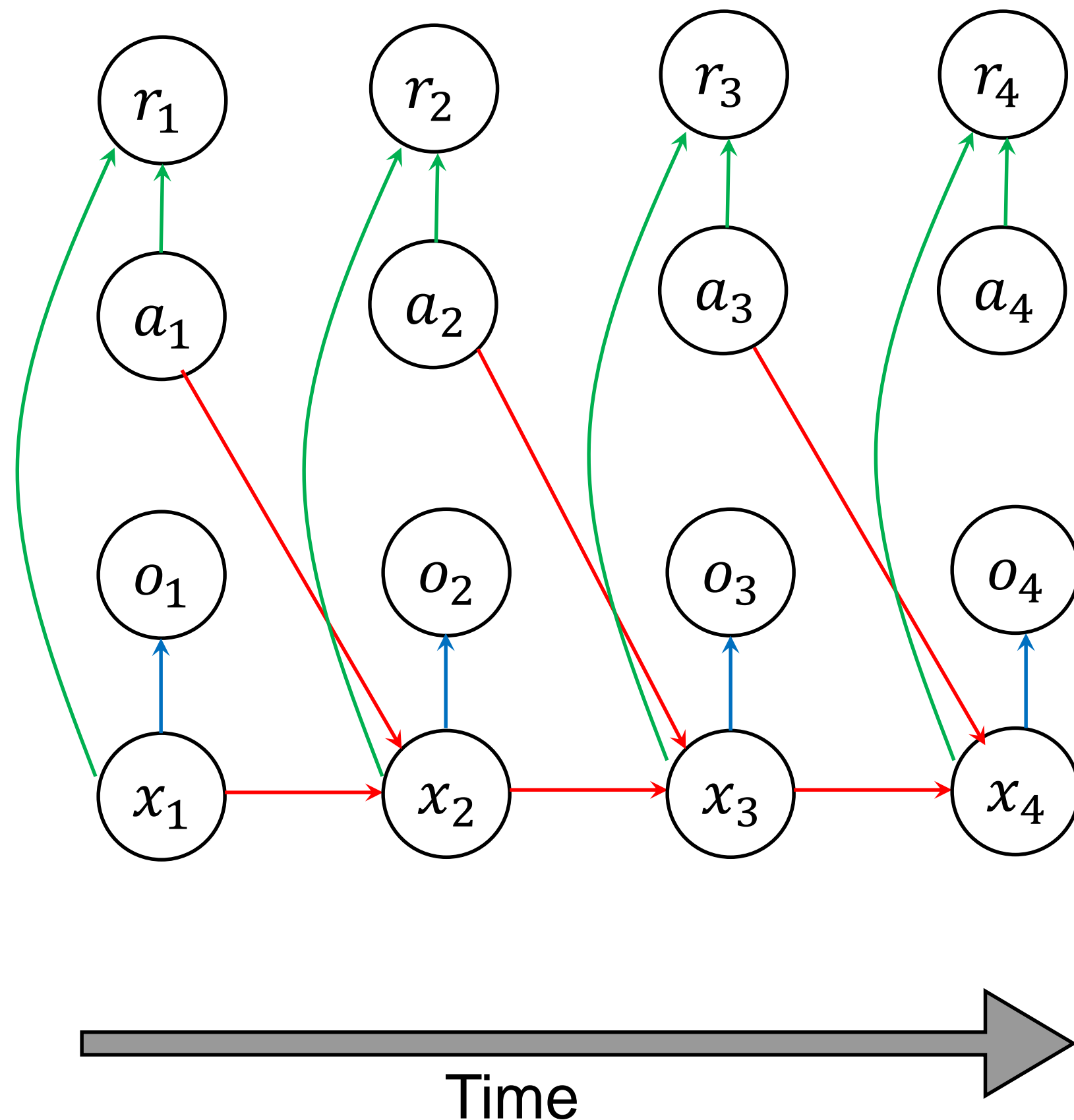
Markov Decision Process and Reinforcement Learning



- The PGM of a Markov Decision Process* (MDP) is shown in the left.
- An MDP* has the following components:
 3. **Actions:** Basically, the action taken by the agent.
 4. **Rewards:** Basically, the rewards received by the agent after taking an action.

*Strictly speaking, a Partially Observable MDP (POMDP). The difference between MDP and POMDP is explained after a few slides.

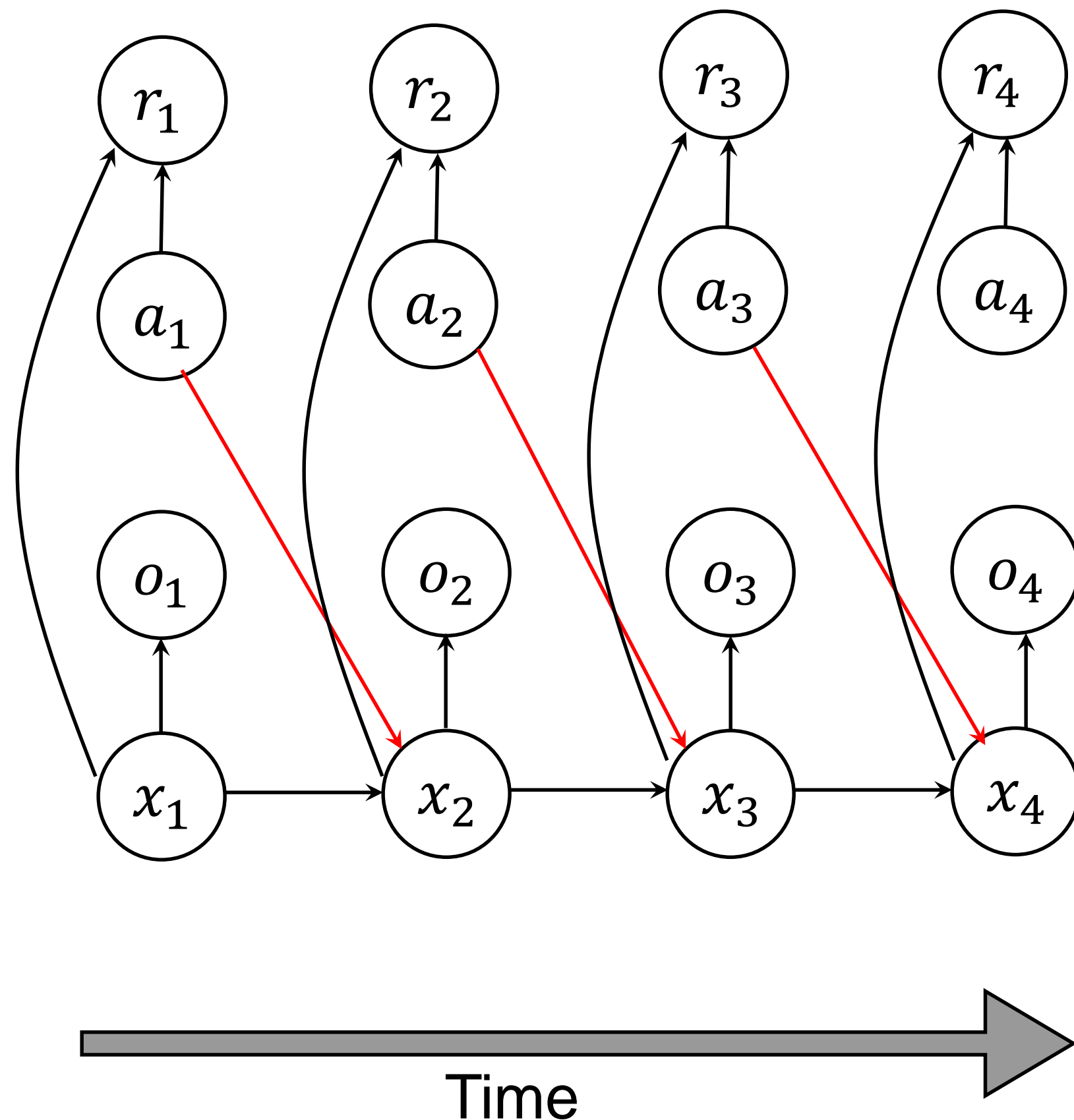
Markov Decision Process and Reinforcement Learning



- An MDP* has the following **conditional dependencies**:
1. Because of Markovian property, current state x_t is dependent on the past state x_{t-1} action and a_{t-1} . This is shown in the PGM using **red arrow**.
 2. The current observation o_t is dependent on current state x_t . This is shown in the PGM using **blue arrow**.
 3. The current reward r_t is dependent on the current state x_t and action a_t . This is shown in the PGM using **green arrow**.

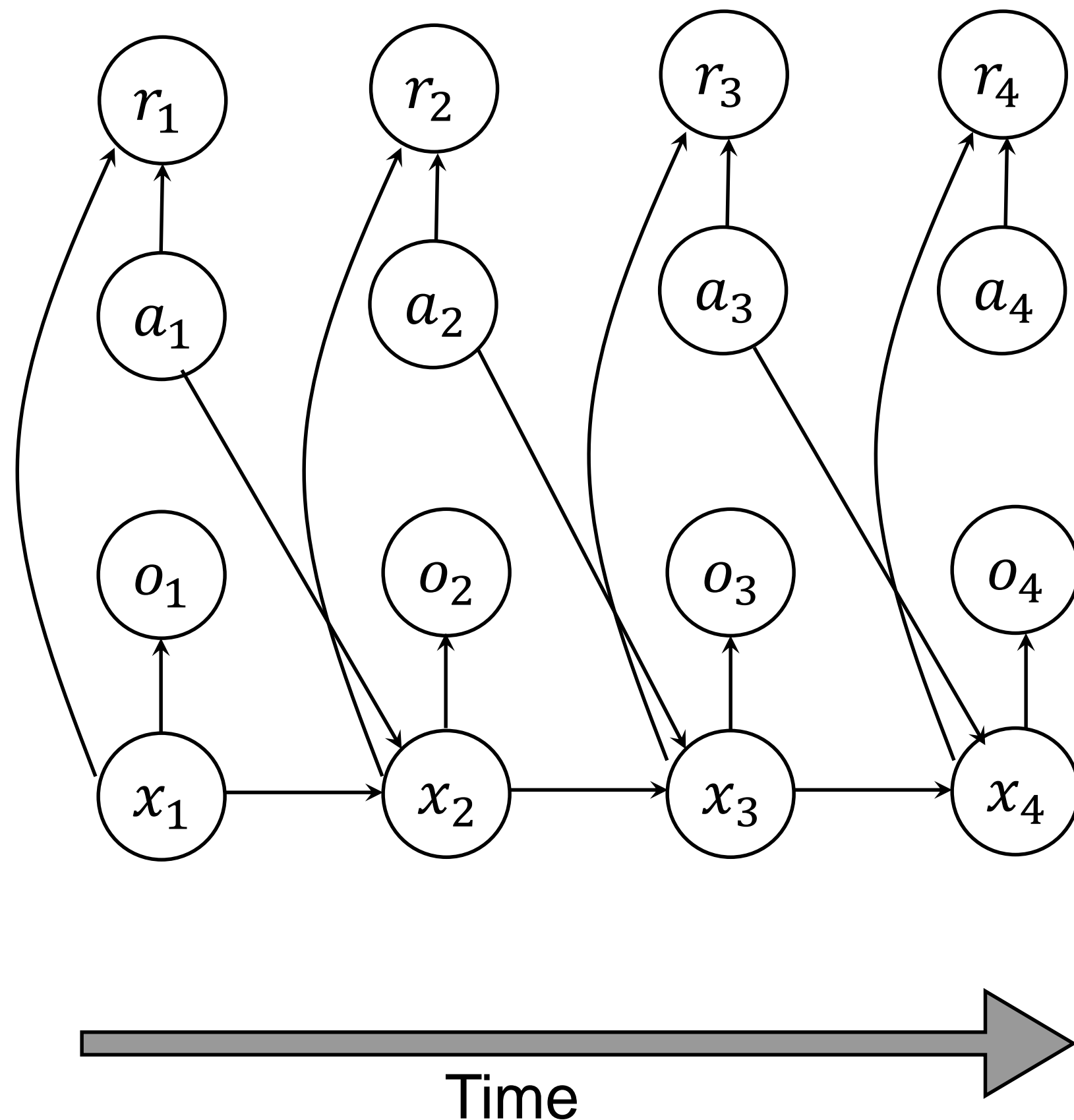
*Strictly speaking, a Partially Observable MDP (POMDP). The difference between MDP and POMDP is explained after a few slides.

Markov Decision Process and Reinforcement Learning



- Notice the **red arrows** in the PGM that shows that current action decides future states and hence the future rewards.
- Because of the presence of these red arrows that taking **greedy action** to maximize the current reward is not a optimal decision. A greedy action may lead to a future state where rewards are low.

Markov Decision Process and Reinforcement Learning



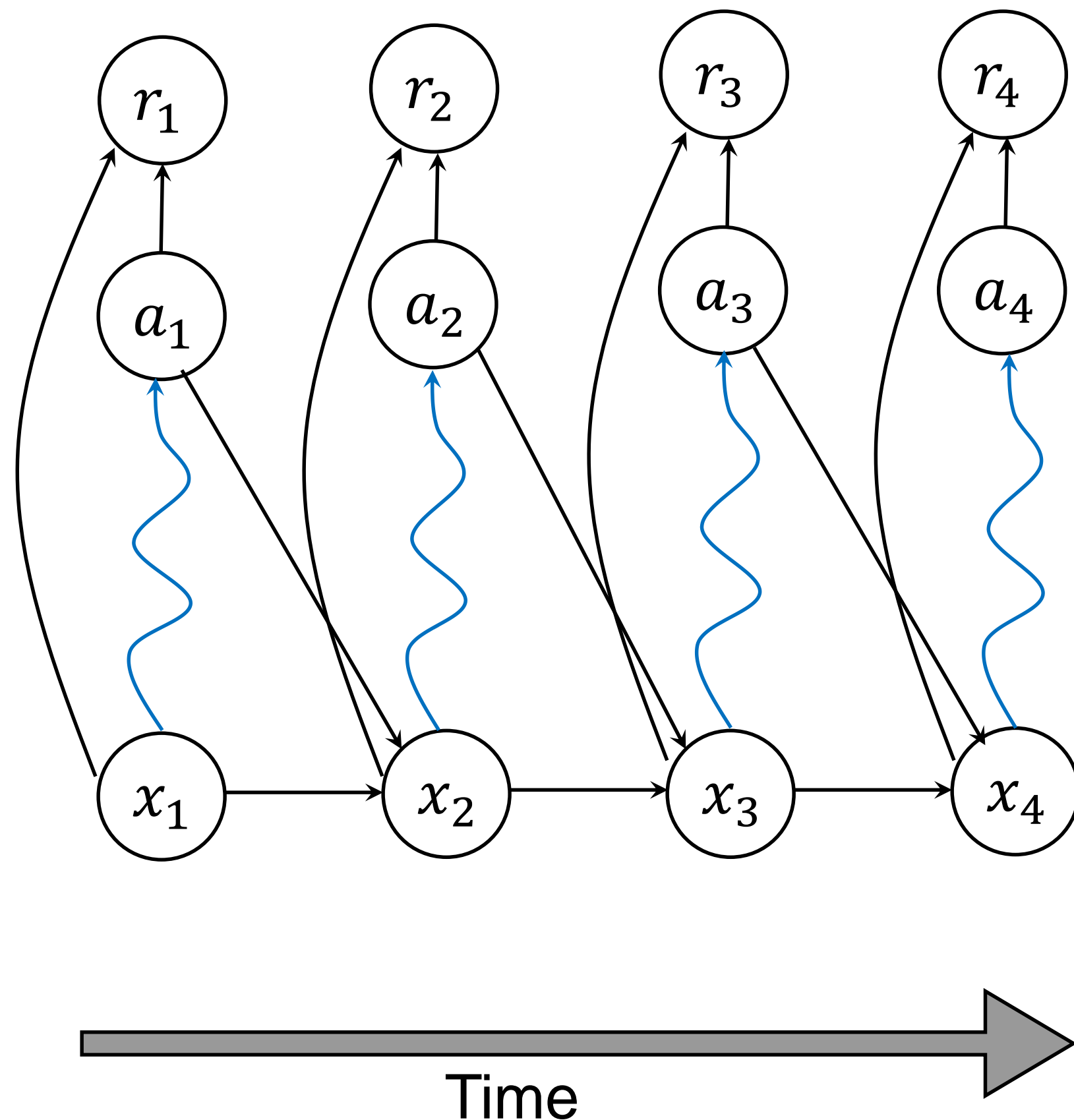
Policies

Question: Based on what does the agent decide its actions?

Answer: The agent decides its action based on a **policy**. Depending on the situation, there are two possible cases when it comes to designing policies.

(PTO)

Markov Decision Process and Reinforcement Learning



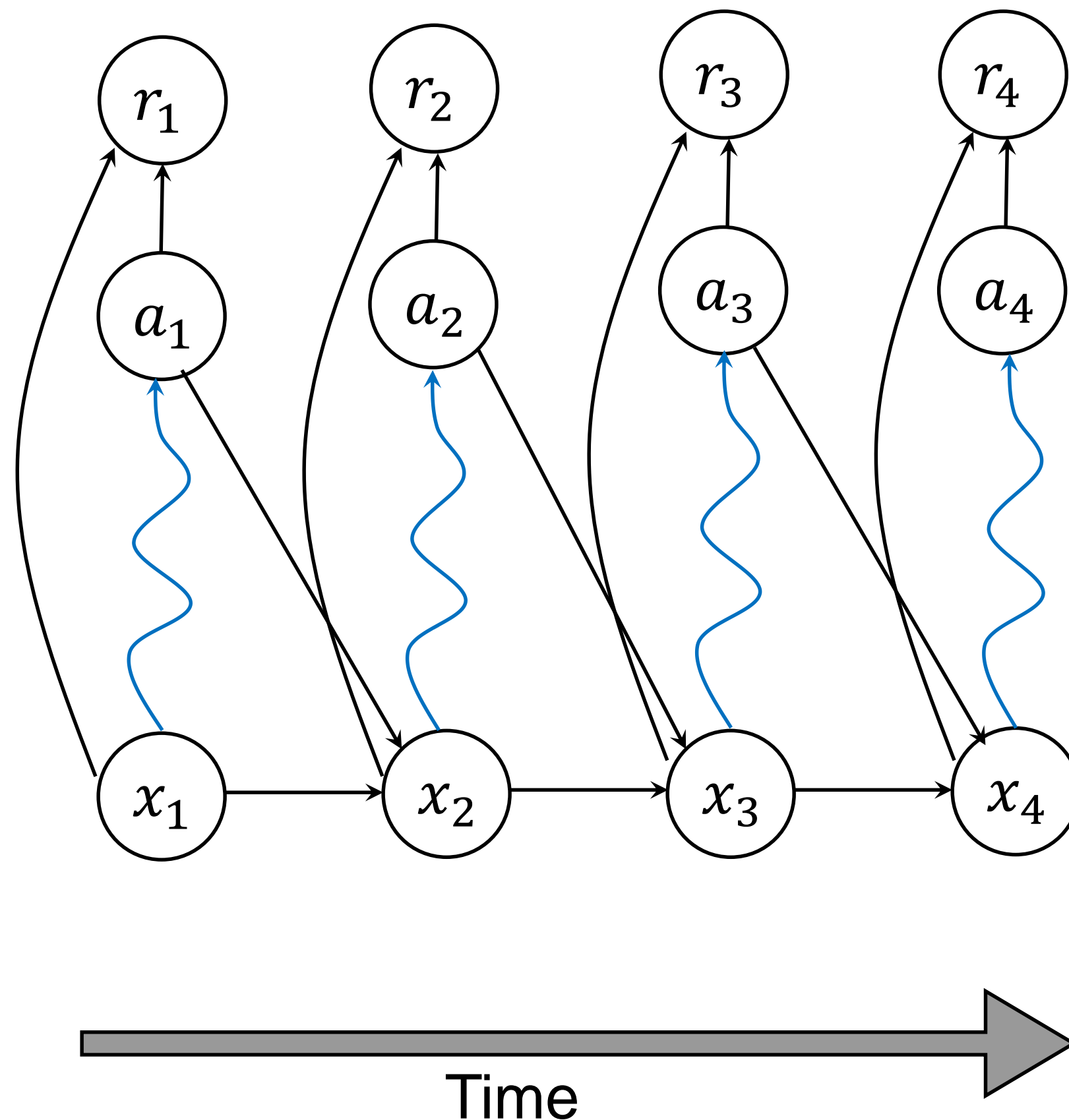
Policies

Case 1 ($\mathbf{o}_t = \mathbf{x}_t$):

- Basically, **all the states are observed**. This is called a **Markov Decision Process**.
NOTE: Since in this case \mathbf{o}_t and \mathbf{x}_t are the same, \mathbf{o}_t 's have been removed from the PGM.
- In this case, the agent decides \mathbf{a}_t based on \mathbf{x}_t . A policy π is a mapping from state \mathbf{x}_t to action \mathbf{a}_t (this is shown using the **blue squiggly arrow**). The policy is denoted as follows,

$$\pi(\mathbf{a}_t | \mathbf{x}_t)$$

Markov Decision Process and Reinforcement Learning



Policies

Case 1 ($\mathbf{o}_t = \mathbf{x}_t$):

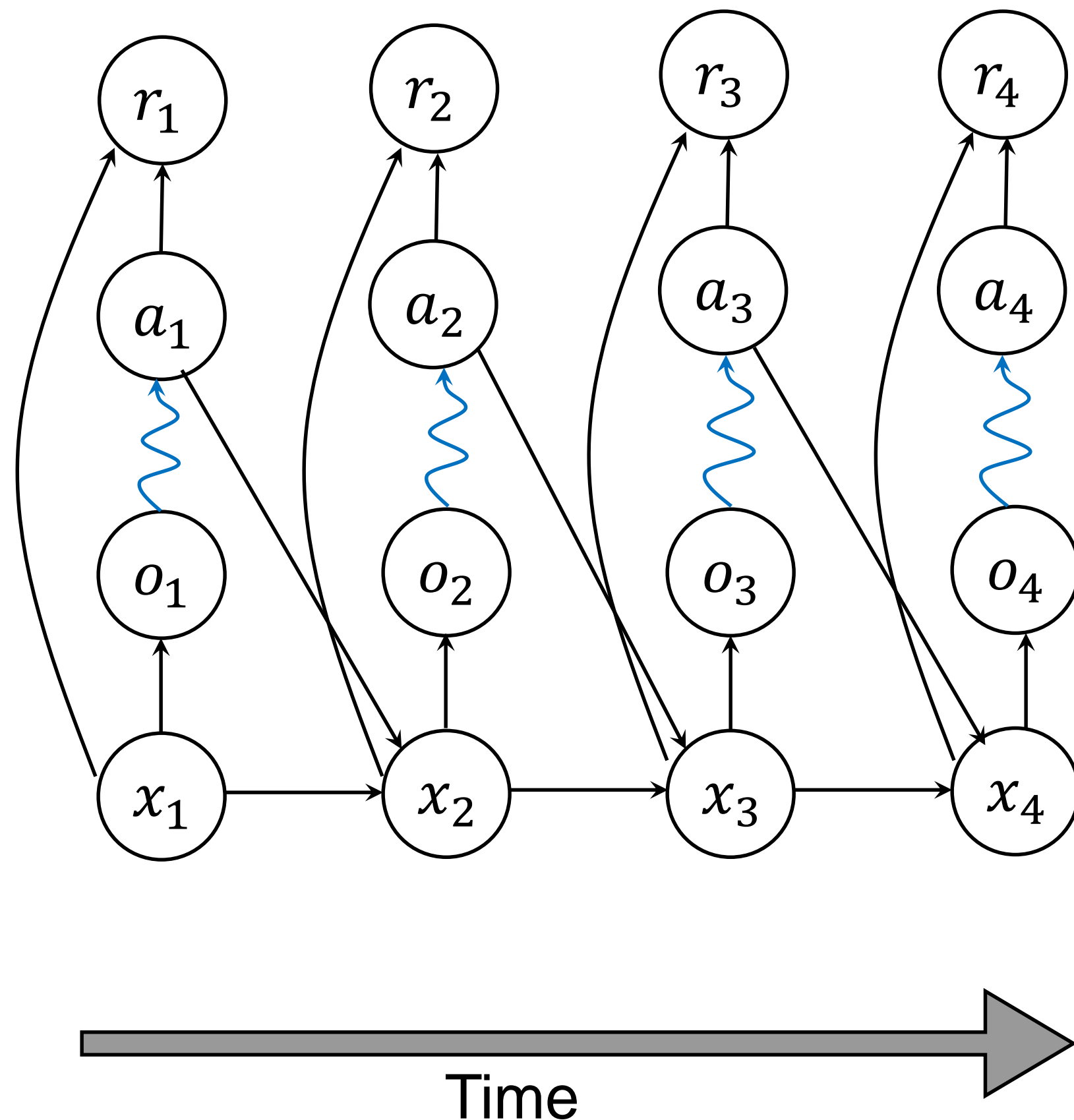
- In general, a policy should consider the **history of states** to design an optimal policy. Such a policy can be written as,

$$\pi(a_t | x_t, x_{t-1}, x_{t-2}, \dots)$$

Question: Is there ALWAYS an OPTIMAL policy of the form $\pi(a_t | x_t)$?

Answer: YES. Since the state x_t is known (completely observed), ALL the statistical property of the future states and hence future rewards can be inferred from x_t because of the **Markovian property**.

Markov Decision Process and Reinforcement Learning



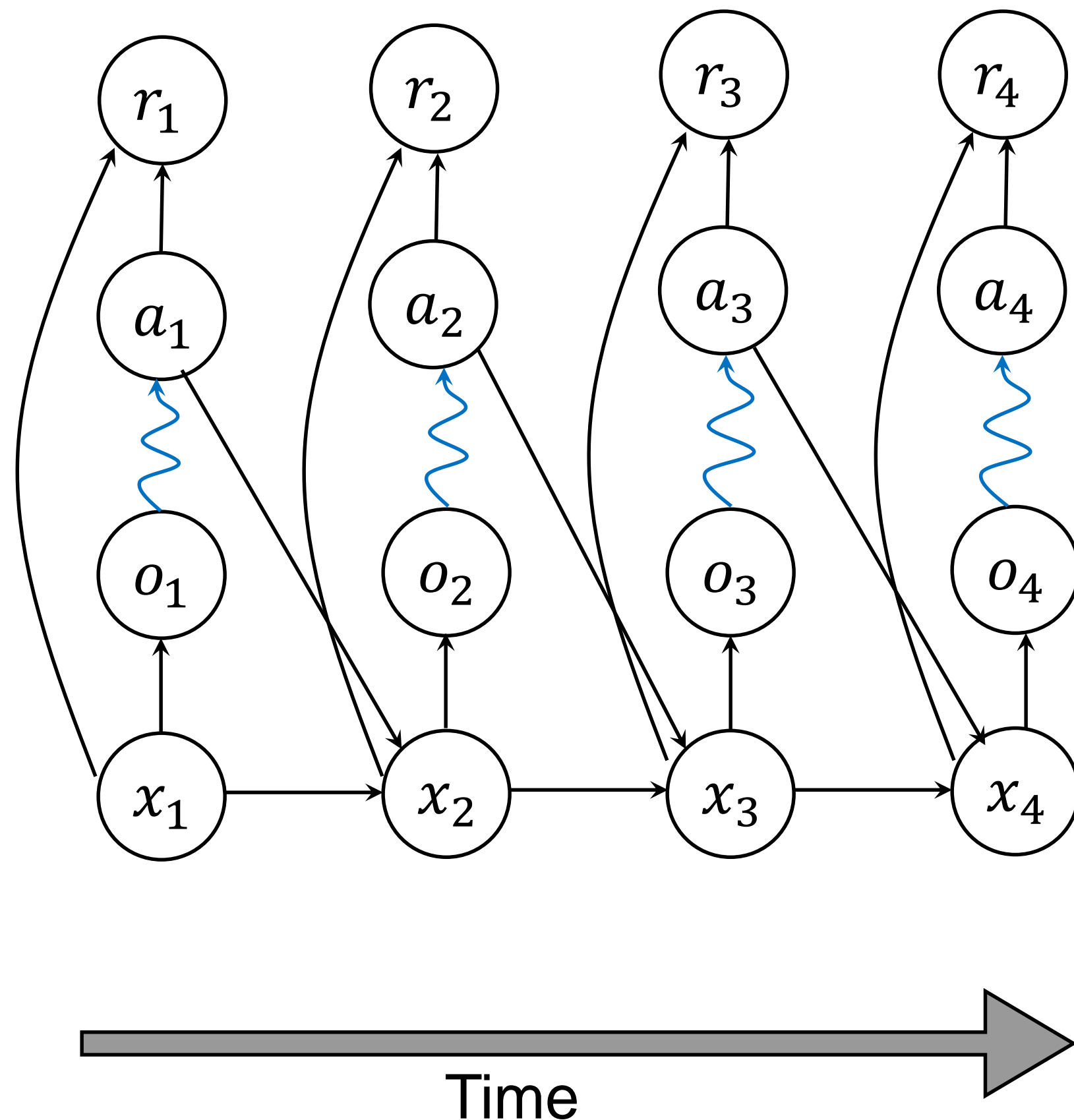
Policies

Case 2 ($\mathbf{o}_t \neq \mathbf{x}_t$):

- Basically, **all the states are NOT observed**. Hence this case is called **Partially Observable Markov Decision Process** or POMDP.
- In this case, the agent decides a_t based on o_t . A policy π is a mapping from observation o_t to action a_t (this is shown using the **blue squiggly arrow**). The policy is denoted as follows,

$$\pi(a_t|o_t)$$

Markov Decision Process and Reinforcement Learning



Policies

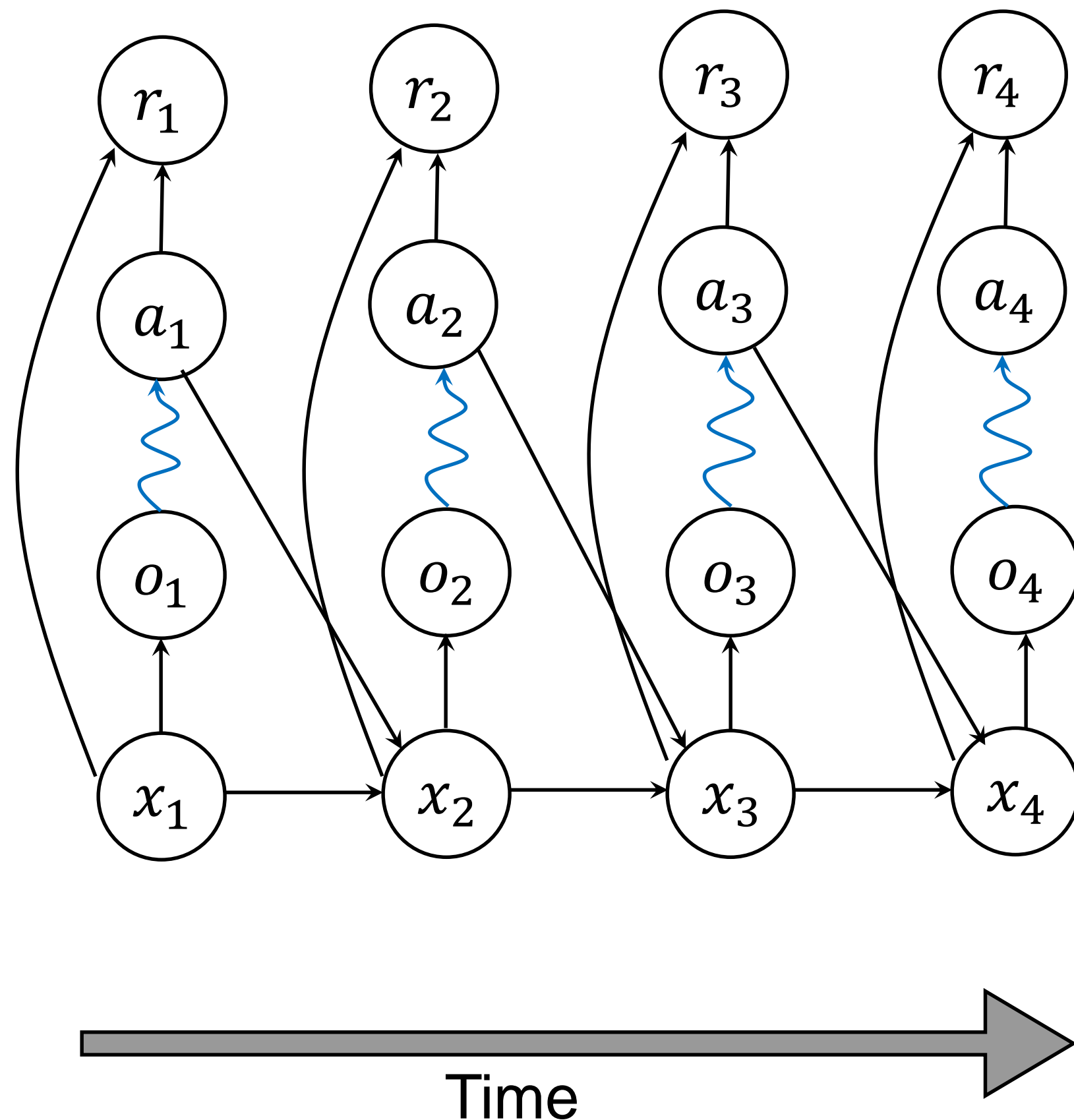
Case 2 ($o_t \neq x_t$):

- Basically, **all the states are NOT observed**. Hence this case is called **Partially Observable Markov Decision Process** or POMDP.
- In this case, the agent decides a_t based on o_t . A policy π is a mapping from observation o_t to action a_t (this is shown using the **blue squiggly arrow**). The policy is denoted as follows,

$$\pi(a_t|o_t)$$

**Most of this course deals with MDP
and NOT POMDP.**

Markov Decision Process and Reinforcement Learning



Policies

Case 2 ($\mathbf{o}_t \neq \mathbf{x}_t$):

- In general, a policy should consider the **history of observations** to design an optimal policy. Such a policy can be written as,

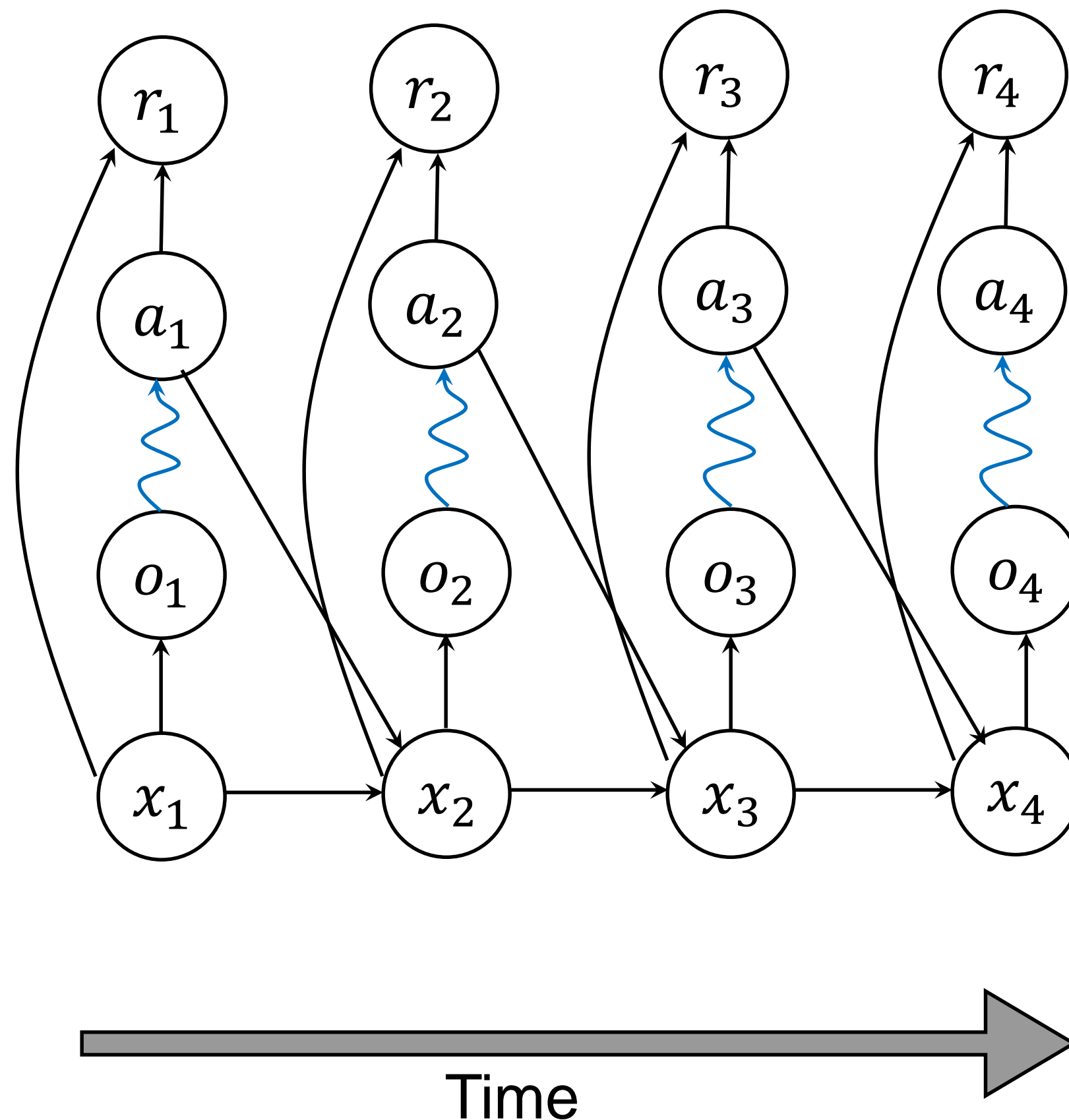
$$\pi(a_t | o_t, o_{t-1}, o_{t-2}, \dots)$$

Question: Is there ALWAYS an OPTIMAL policy of the form $\pi(a_t | o_t)$?

Answer: NO. Since the current state \mathbf{x}_t is not completely available, the Markovian property cannot be directly applied. In order to make optimal decisions, past observations may be needed. To understand why, we need the concept of **D-Separation** ([link here](#)) that gives a systematic

(Continued on next slide)

Markov Decision Process and Reinforcement Learning



Policies

Case 2 ($\mathbf{o}_t \neq \mathbf{x}_t$):

- In general, a policy should consider the **history of observations** to design an optimal policy. Such a policy can be written as,

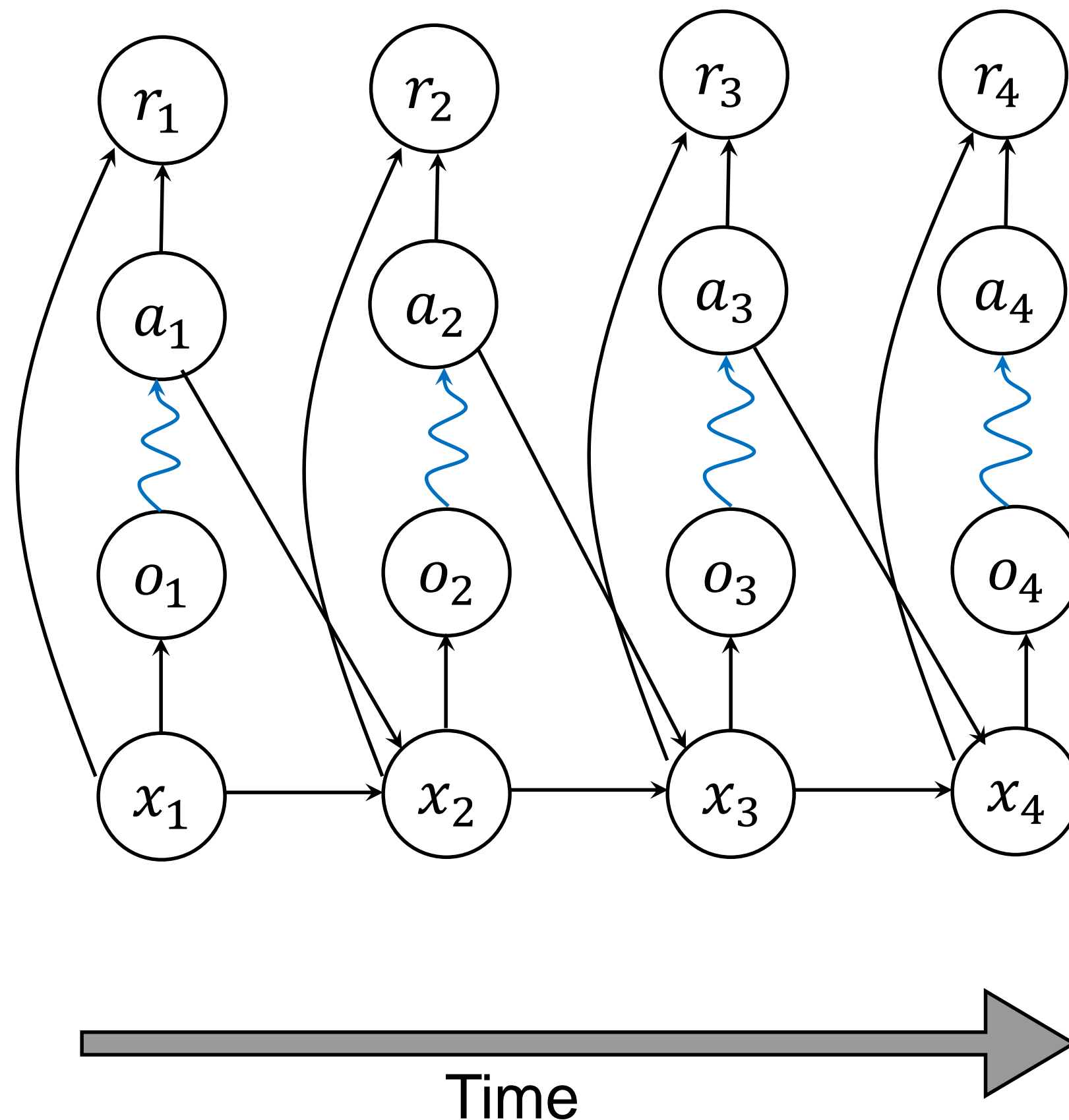
$$\pi(a_t | o_t, o_{t-1}, o_{t-2}, \dots)$$

Question: Is there ALWAYS an OPTIMAL policy of the form $\pi(a_t | o_t)$?

Answer: procedure to understand conditional dependences in PGM. However, D-Separation is beyond the scope of this syllabus. Here we can understand it using an example. In the autonomous driving example in lecture 1, the agent needs both position and velocity of vehicles to make optimal

(Continued on next slide)

Markov Decision Process and Reinforcement Learning



Policies

Case 2 ($o_t \neq x_t$):

- In general, a policy should consider the **history of observations** to design an optimal policy. Such a policy can be written as,

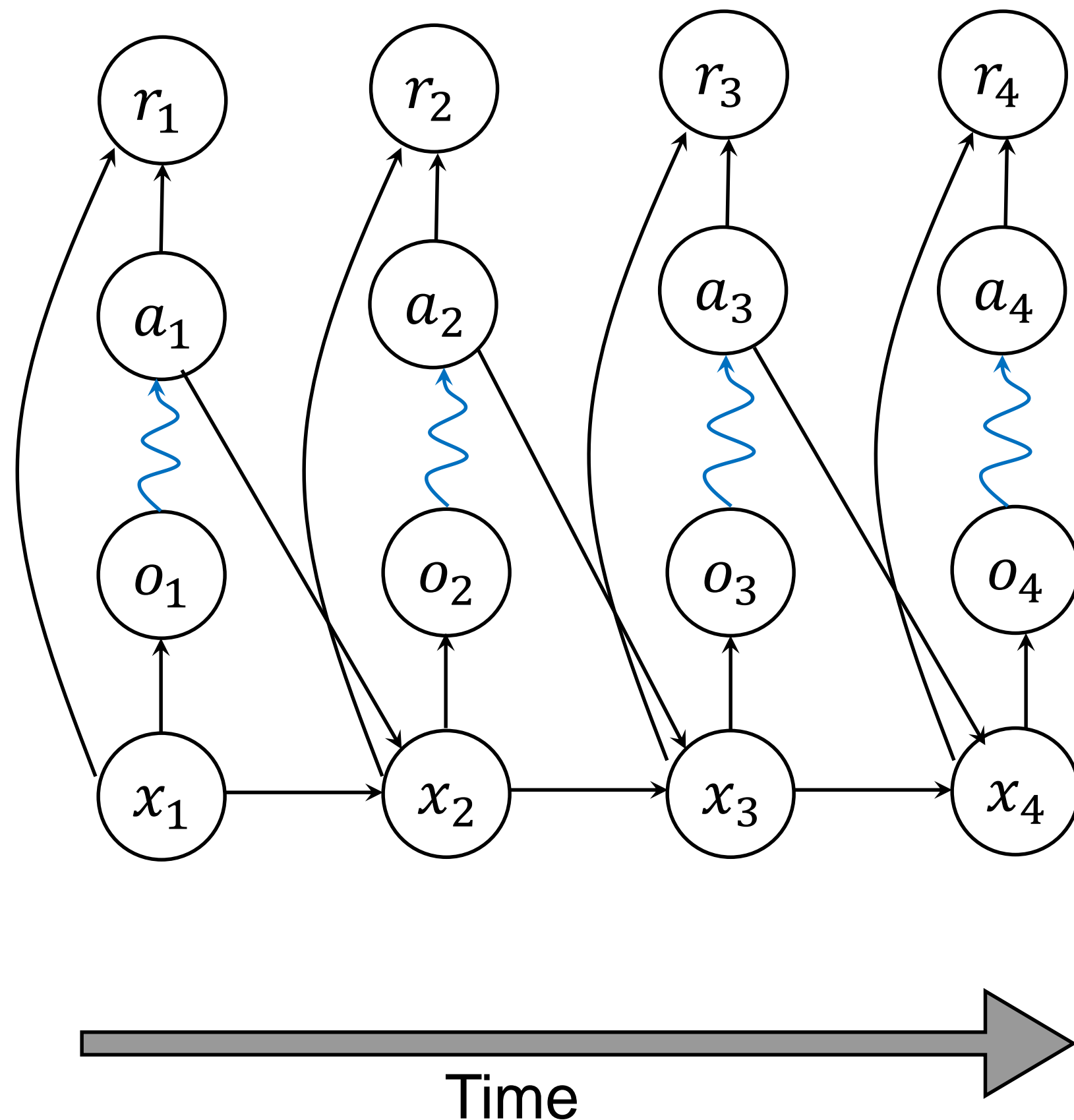
$$\pi(a_t | o_t, o_{t-1}, o_{t-2}, \dots)$$

Question: Is there ALWAYS an OPTIMAL policy of the form $\pi(a_t | o_t)$?

Answer: decisions. However, if only the position of the vehicles are available, then we will need both current and **previous position** to estimate the velocity of the vehicle.

(Continued on next slide)

Markov Decision Process and Reinforcement Learning



Policies

Case 2 ($\mathbf{o}_t \neq \mathbf{x}_t$):

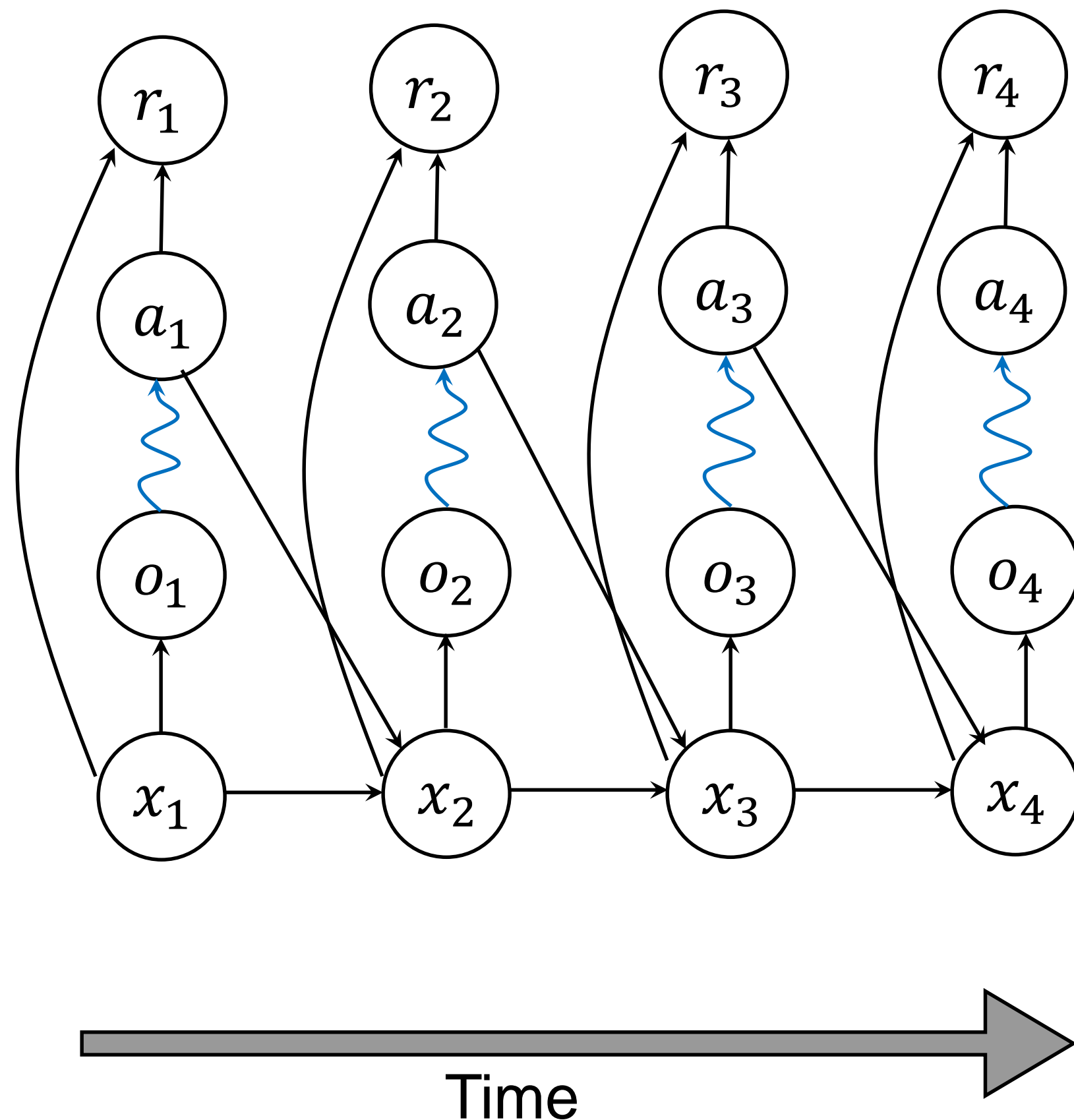
- In general, a policy should consider the **history of observations** to design an optimal policy. Such a policy can be written as,

$$\pi(a_t | o_t, o_{t-1}, o_{t-2}, \dots)$$

Question: Is there ALWAYS an OPTIMAL policy of the form $\pi(a_t | o_t)$?

Side Note: During exams, I may ask questions related to this issue for which logical thinking is enough. But, the knowledge of D-Separation may help you a little bit.

Markov Decision Process and Reinforcement Learning



Policies

- Consider the policies for both the MDP and POMDP case:

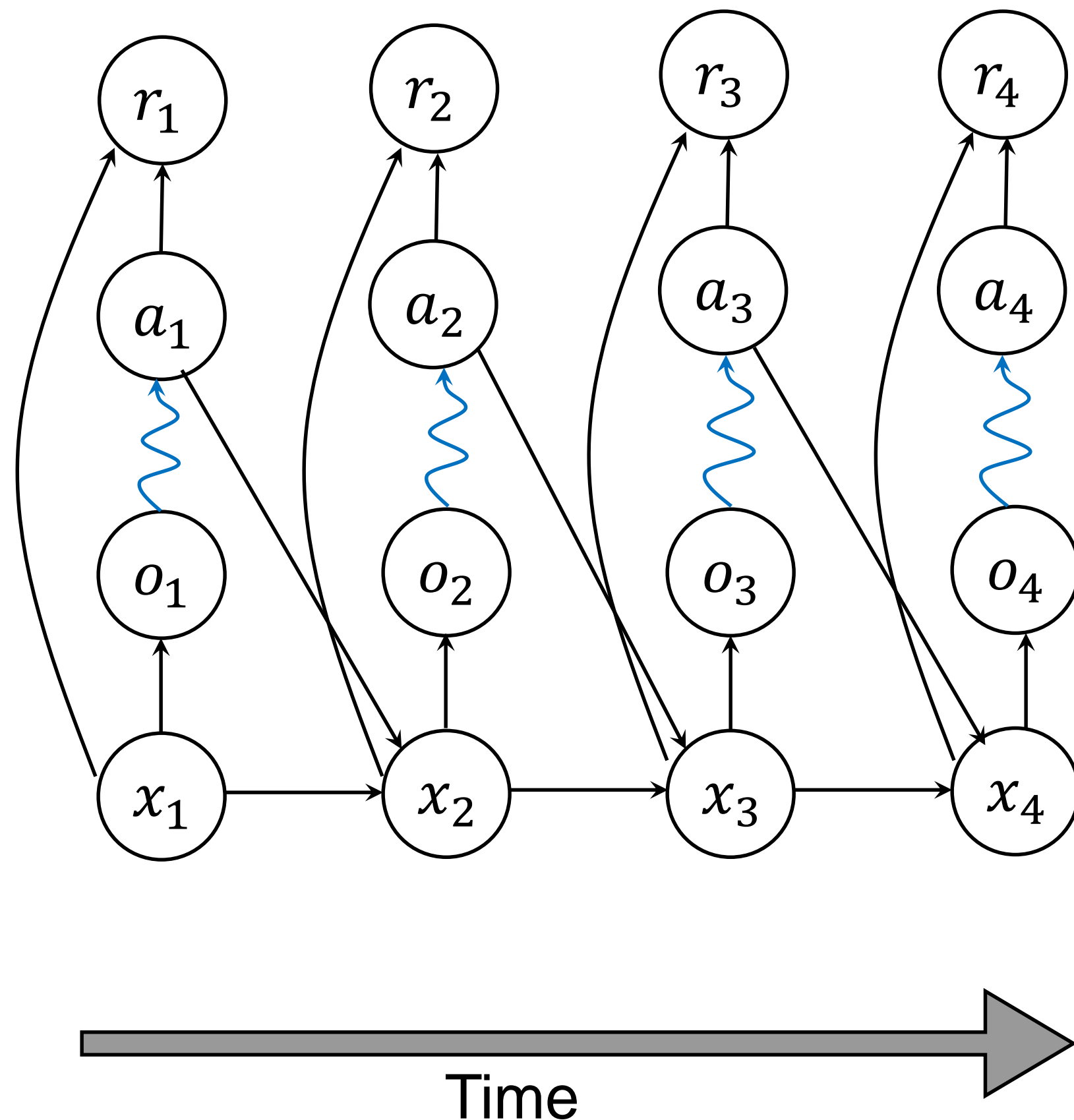
$$\pi(a_t | x_t)$$

$$\pi(a_t | o_t)$$

- For both these cases, the policies can be categorized as either deterministic vs randomized.
 - **Deterministic policy:** For a given x_t or o_t , the policy outputs a fixed a_t . *I believe that most of the algorithms* you might have encountered till now would have been a deterministic algorithm.*

*Algorithm and policy are synonyms.

Markov Decision Process and Reinforcement Learning



Policies

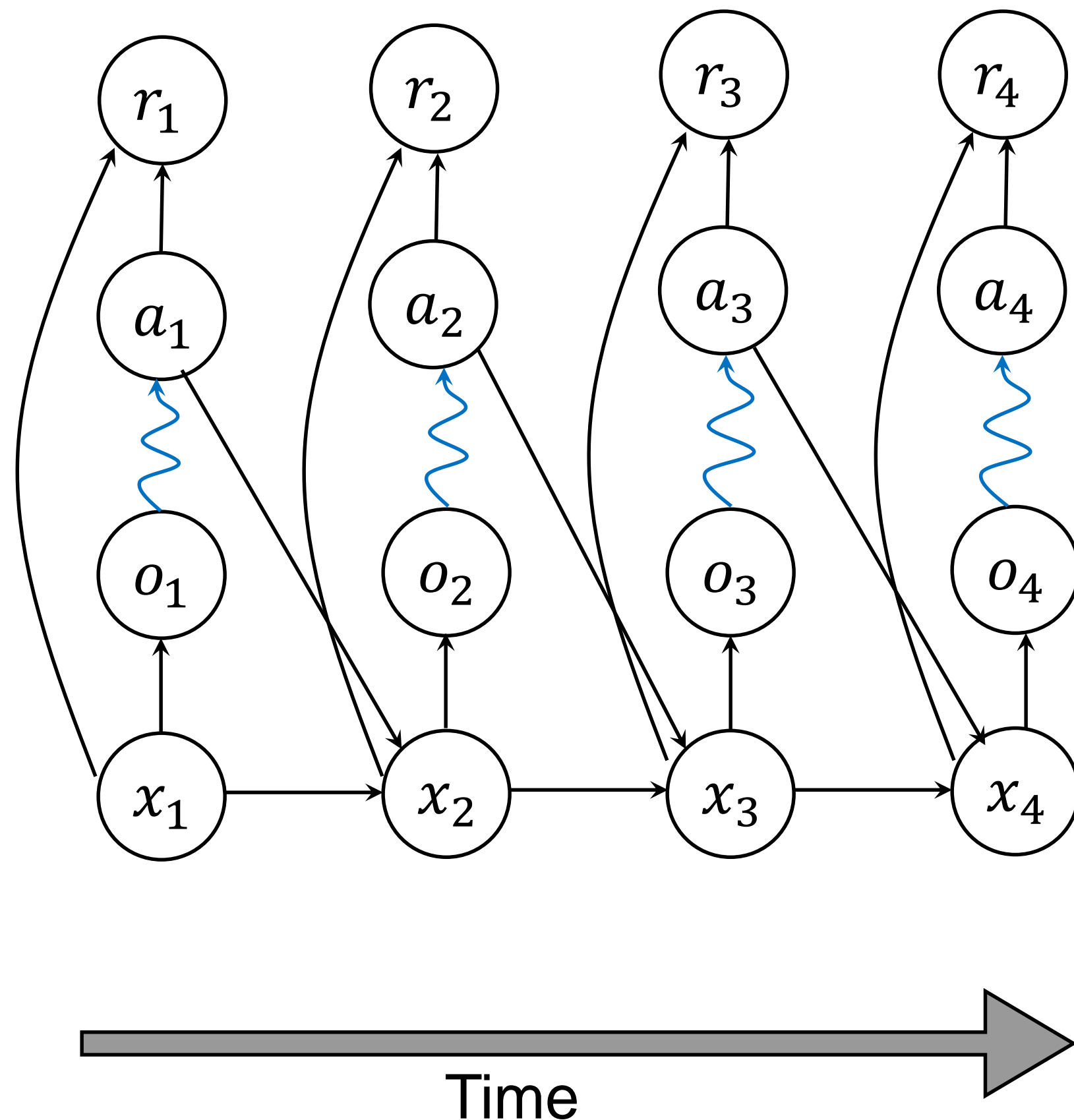
- Consider the policies for both the MDP and POMDP case:

$$\pi(a_t|x_t)$$

$$\pi(a_t|o_t)$$

- For both these cases, the policies can be categorized as either deterministic vs randomized.
 - **Randomized policy:** For a given x_t or o_t , the action a_t is a random variable that depends on x_t or o_t . Hence, a_t is NOT fixed for a given x_t or o_t . E.x. for the warehouse routing example, when **weather condition is rainy**, take **routes 1, 2, and 3** with probability **0.2, 0.5, 0.3** respectively.

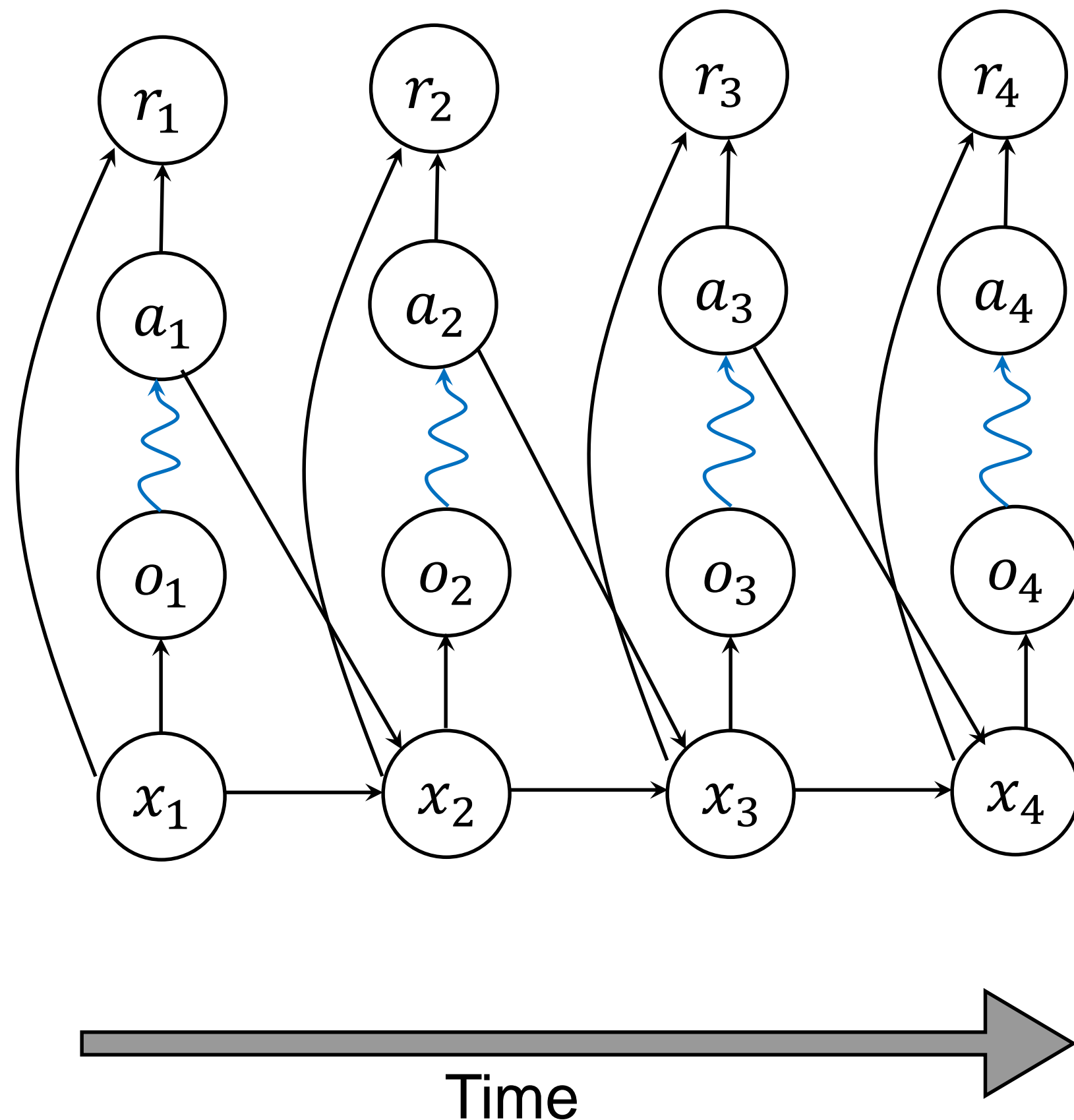
Markov Decision Process and Reinforcement Learning



MDP (POMDP) vs Reinforcement Learning

- Recall that the arrows in the PGM represents **conditional distribution**. These conditional distributions is basically the **probabilistic model** of the uncertainties associated with the system. Based on whether the conditional distribution is know or not, we have the following situations:
 - If the **conditional distributions are known**, we are interested in solving a “**planning problem**” where the objective is to maximize the reward. This is called an **Markov Decision Process (MDP)**. This is dealt in **Module 2 (1st half)**.

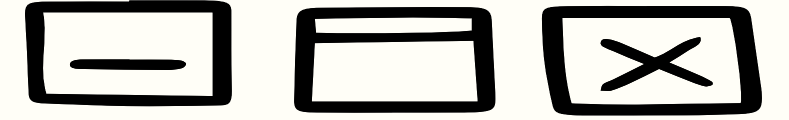
Markov Decision Process and Reinforcement Learning



MDP (POMDP) vs Reinforcement Learning

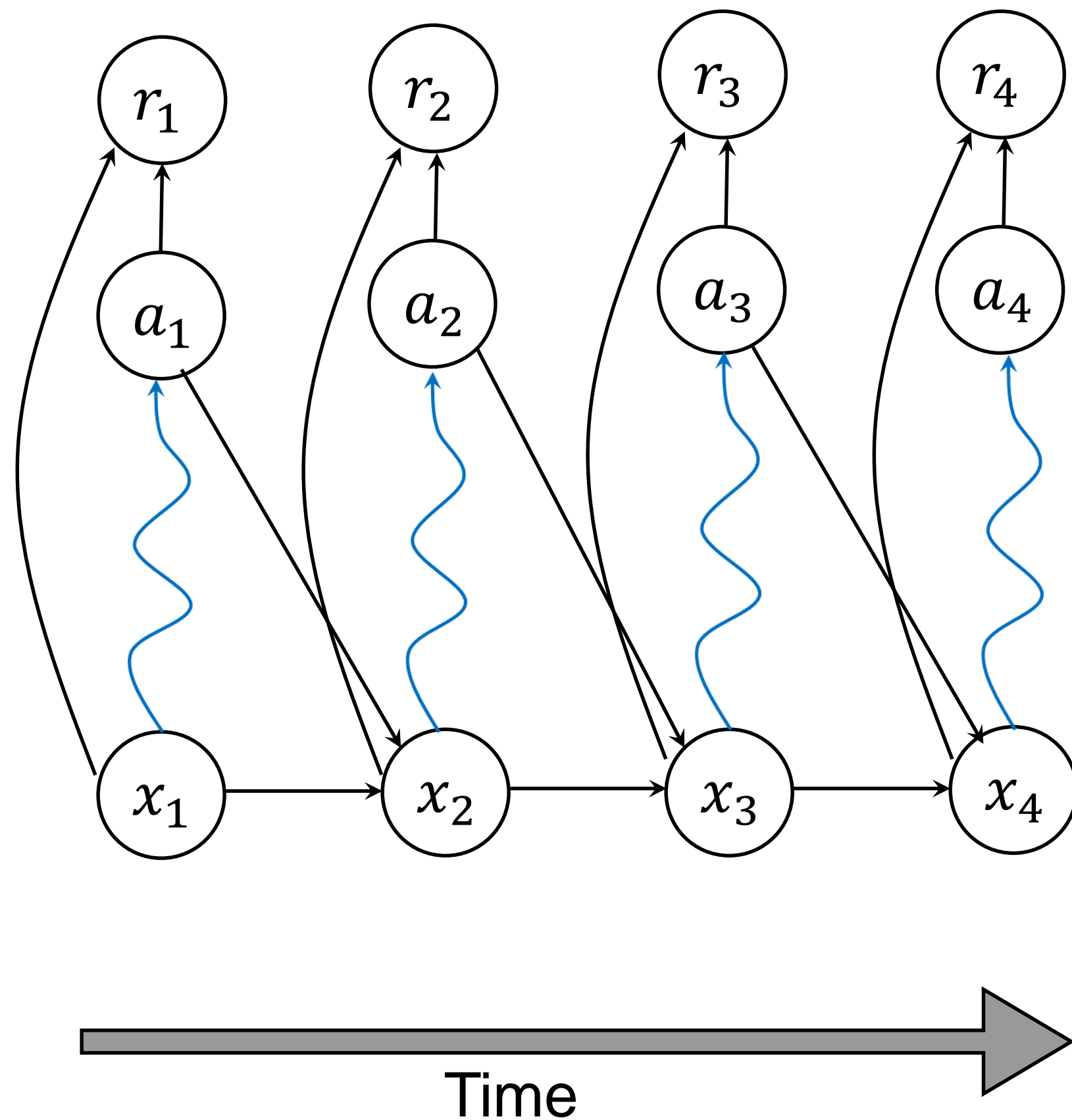
- Recall that the arrows in the PGM represents **conditional distribution**. These conditional distributions is basically the **probabilistic model** of the uncertainties associated with the system. Based on whether the conditional distribution is know or not, we have the following situations:
 - If the **conditional distributions are NOT known**, we are interested in solving a **“planning + learning problem”** where the objective is to maximize the reward which in turn needs the agent to learn the conditional distributions (either directly or indirectly). This is called **Reinforcement Learning**. This is dealt in **Module 2 (2nd half), Module 3, and Module 4**.

Lecture Content



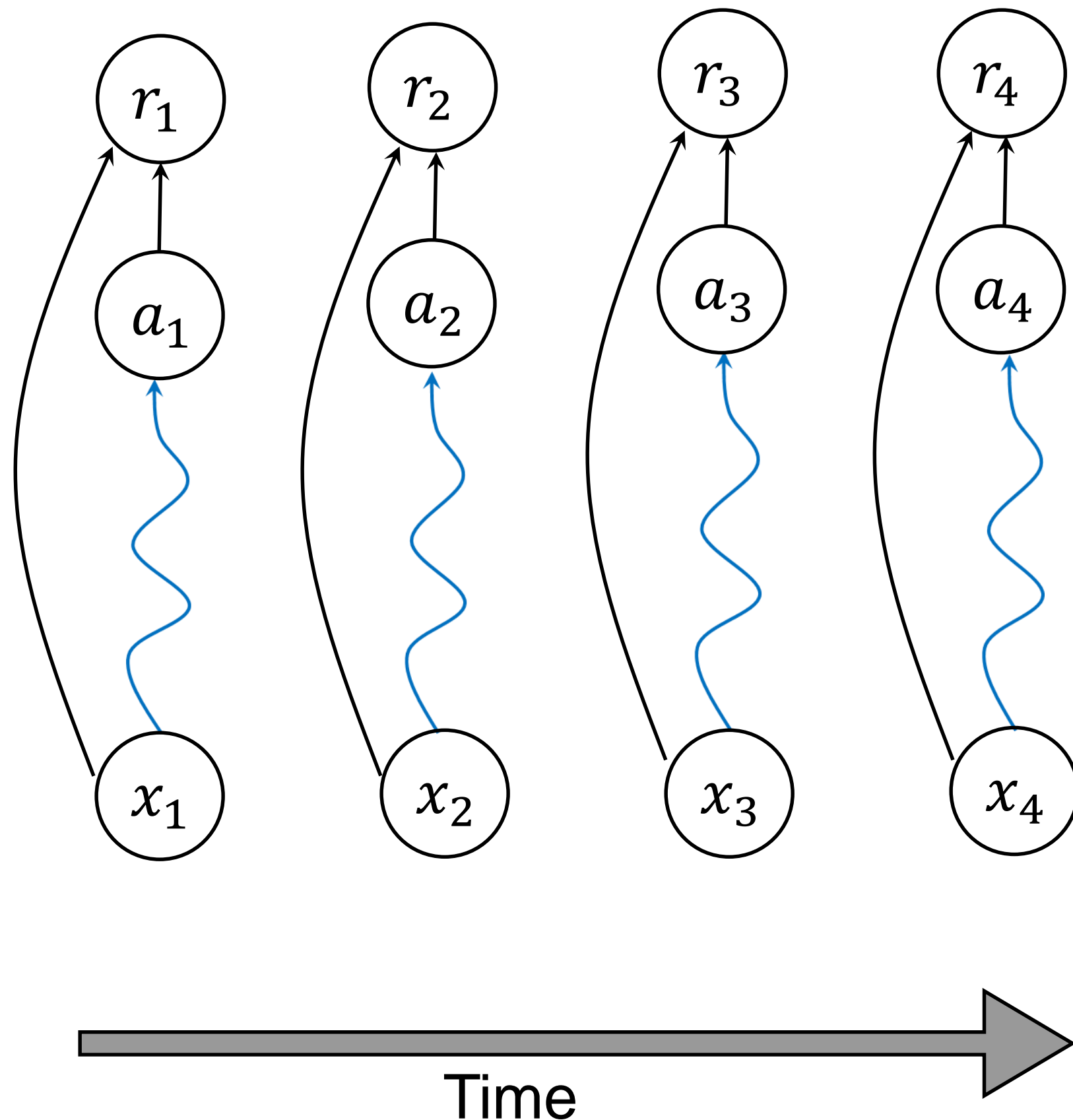
- Stochastic process (very brief introduction; only that what is required for this course).
- Markov Decision Process and Reinforcement Learning.
- Special Cases of Markov Decision Process.
 - Contextual Bandits.
 - Multi-Armed Bandits.
- Stationary vs Non-Stationary Process.
- Objective functions in Reinforcement Learning.
- Reinforcement Learning vs

Special Cases of Markov Decision Process



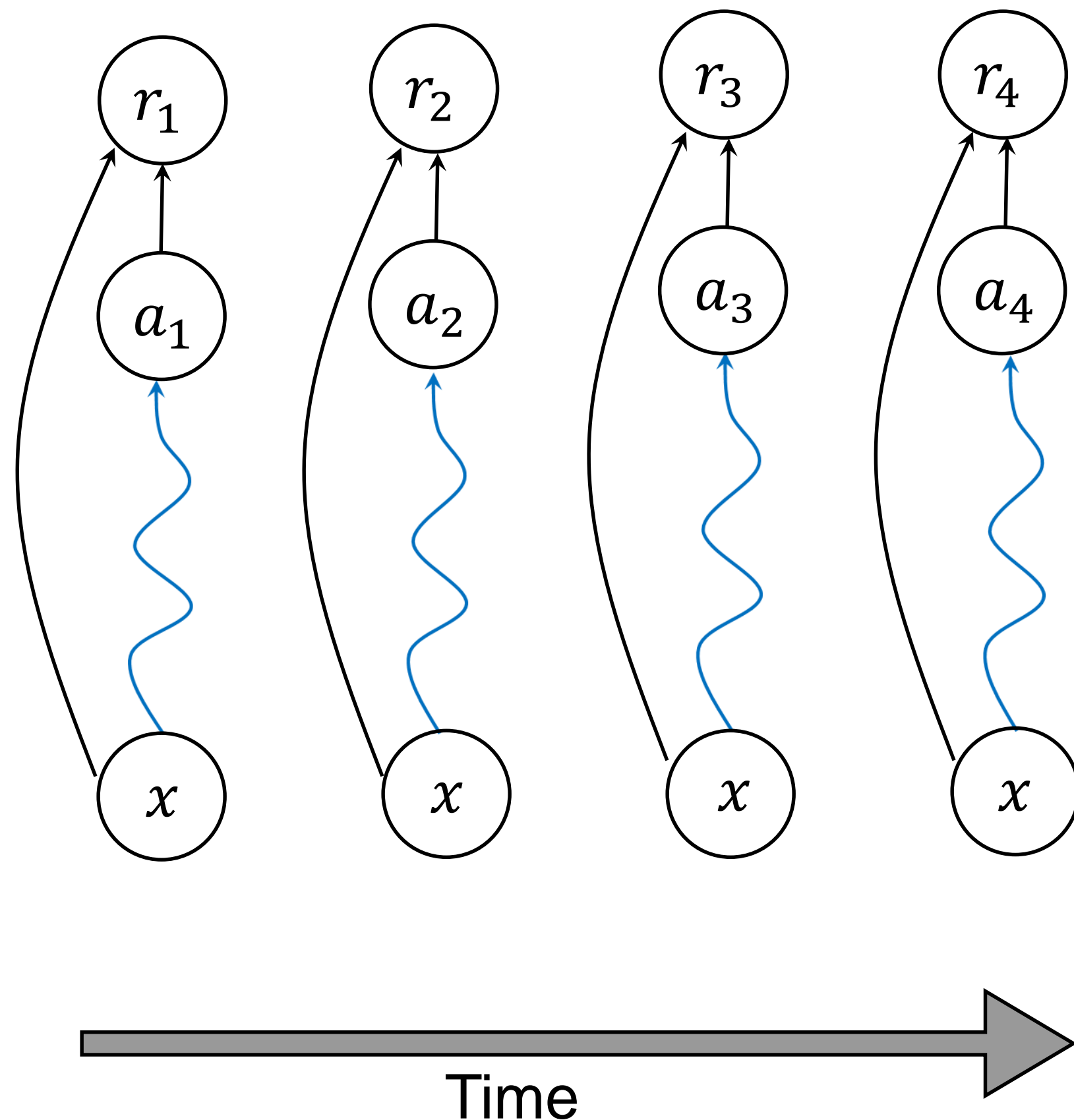
➤ The PGM of a MDP is shown in the left.

Special Cases of Markov Decision Process



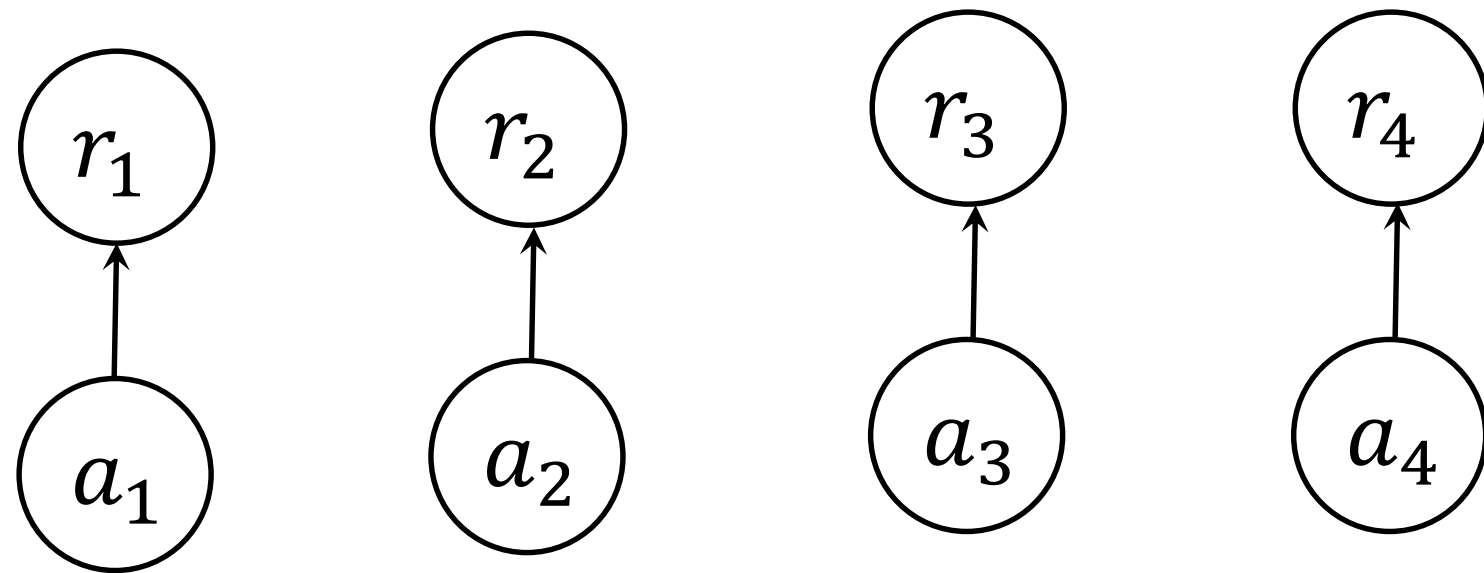
- A contextual bandit is a special case of MDP where there is no **temporal dependence** (dependence over time) between current states and actions to future state.
- Since, there is no temporal dependence, we can make **greedy decisions** in every time step and yet come up with an **optimal policy**.
- The states are called “**context**”.
- We will learn about contextual bandits in Module 1.

Special Cases of Markov Decision Process



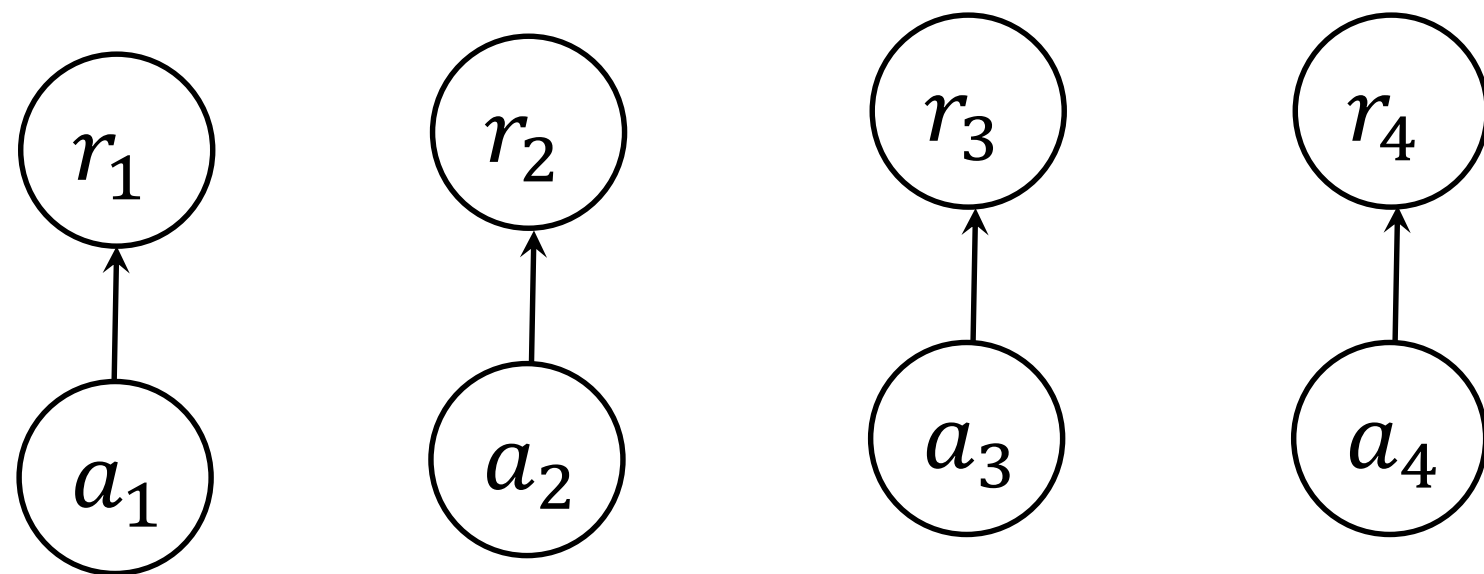
- A multi-armed bandit is special case of contextual bandit where there is only **one state** and hence the state does not change with time.

Special Cases of Markov Decision Process



- A multi-armed bandit (MAB) is special case of contextual bandit where there is only **one state** and hence the state does not change with time.
- Since there is only one state x , we can ignore the state.
 - This is similar to supervised learning. If there is a feature (a column) that has same entry for all the rows, we can drop that feature.
- So, in essence the reward only depends on the action.
- We will learn about multi-armed bandits in Module 1.

Special Cases of Markov Decision Process

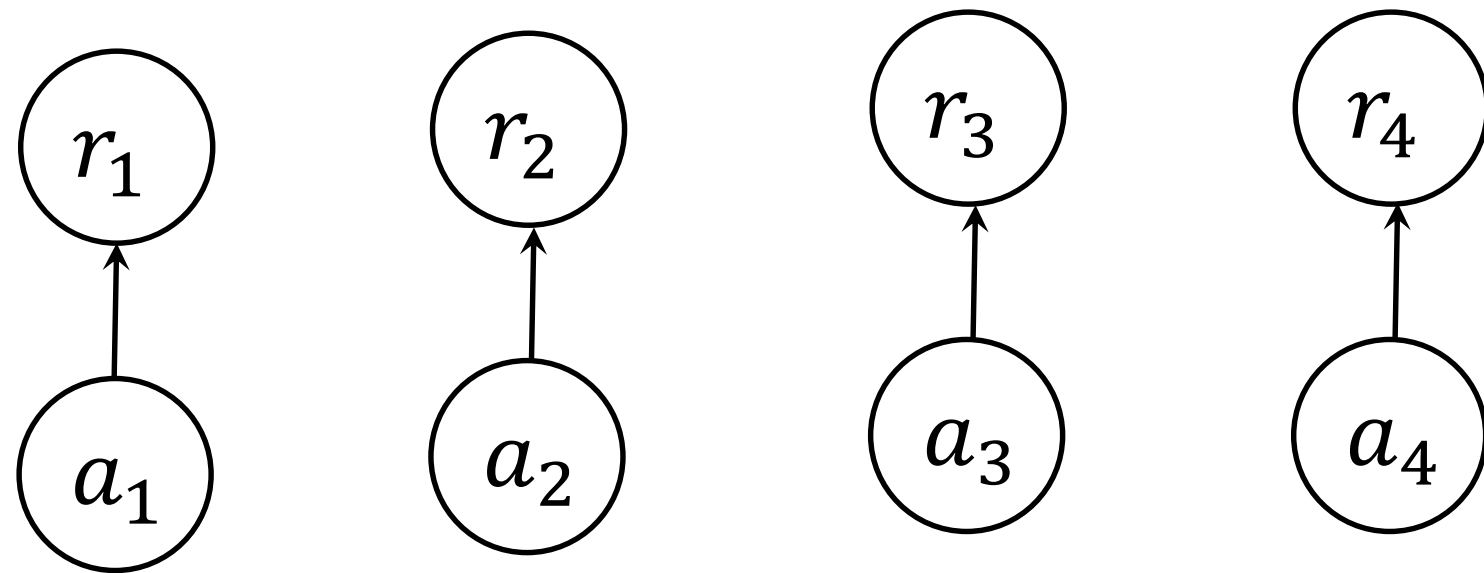


- For MAB, if we know the **expected reward** for all the actions, we would always select the action with the maximum expected reward.
- Similarly for contextual bandits, if we know the expected reward for all the actions **of a given state**, we would always select the action with the maximum expected reward **for that given state**.
- Hence, the **planning problem*** for both MAB and contextual bandits is TRIVIAL!
- **Important:** The phrase multi-armed bandit (or MAB) and contextual bandit **by default** means **learning + planning problem**, i.e. the probabilistic model of the system is **not known**.

*Recall that planning problem is the case when we know the probabilistic model of the system.

Time

Special Cases of Markov Decision Process



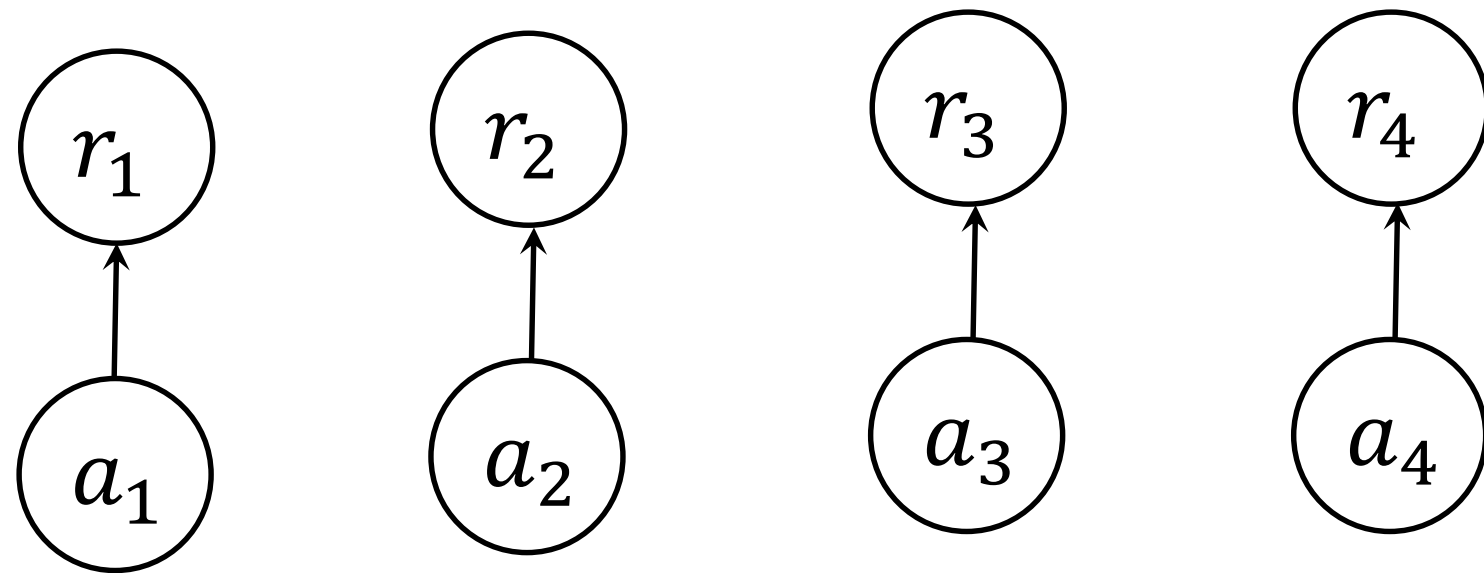
- For MAB, if we know the **expected reward** for all the actions, we would always select the action with the maximum expected reward.
- Similarly for contextual bandits, if we know the expected reward for all the actions **of a given state**, we would always select the action with the maximum expected reward **for that given state**.
- Hence, the **planning problem*** for both MAB and contextual bandits is TRIVIAL!
- **Important:** The phrase multi-armed bandit (or MAB) and contextual bandit **by default** means **learning + planning problem**, i.e. the probabilistic model of the system is **not known**.

*Recall that planning problem is the case when we know the probabilistic model of the system.

Special Cases of Markov Decision Process

A Conceptual Dilemma about Bandits

For MAB and contextual bandit, does the current action decides future reward?



Time

Special Cases of Markov Decision Process

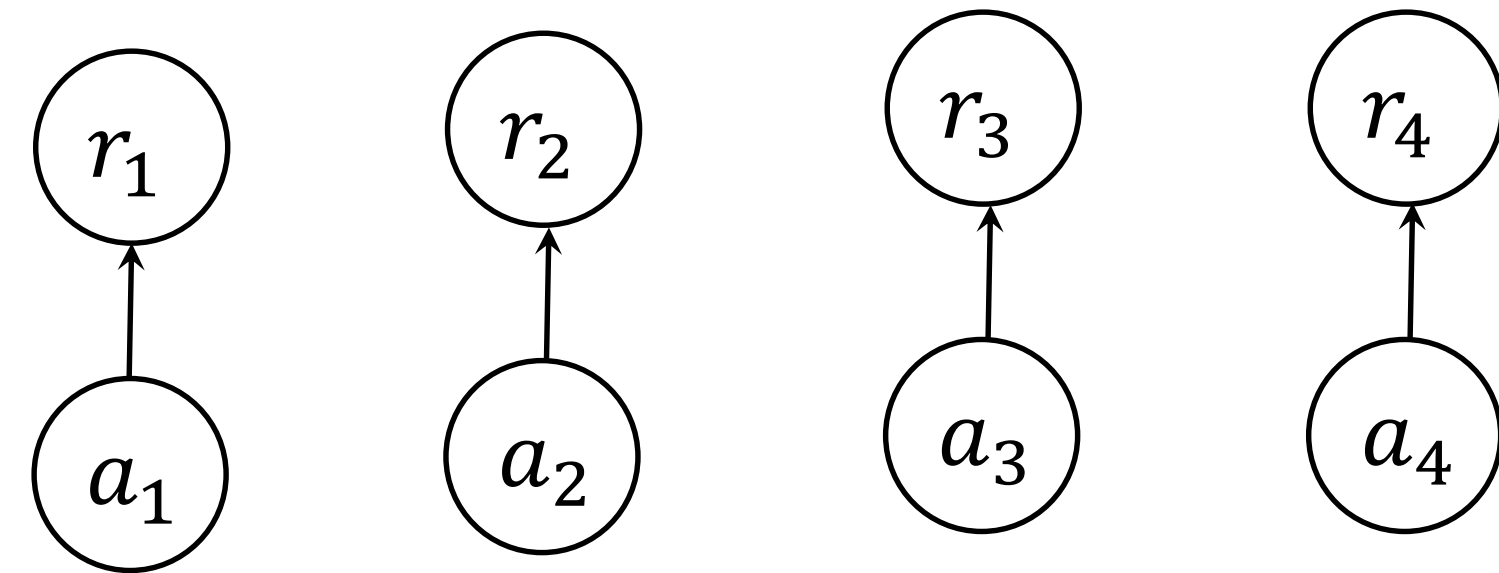
A Conceptual Dilemma about Bandits

For MAB and contextual bandit, does the current action decides future reward?

This question leads to the following dilemma:

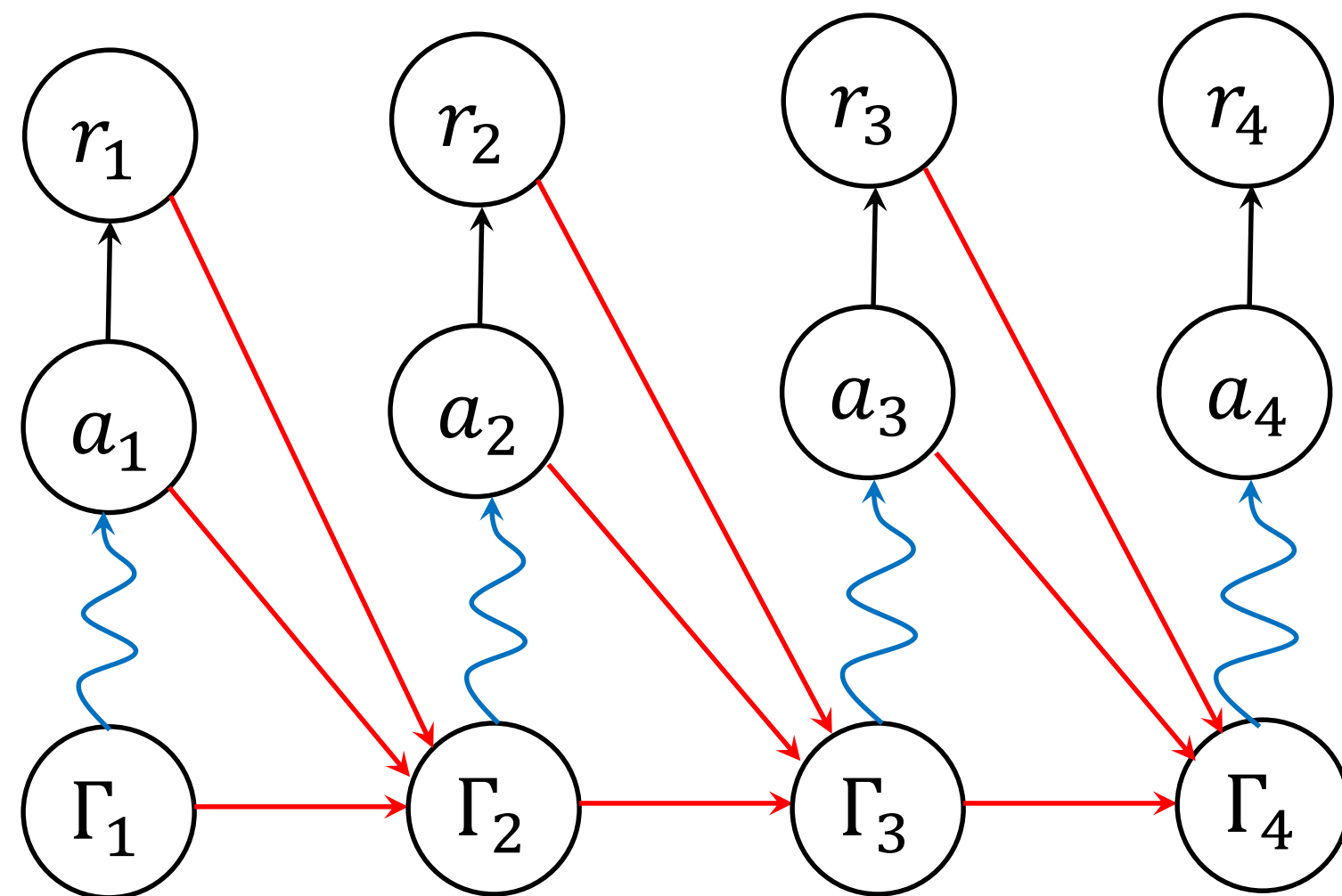
- In MAB and contextual bandits there is no temporal dependence (as it is clear from the PGM on the left). **So, it seems that the current action should not decide future rewards.**
- However, consider the warehouse routing example. If we travel a route that we didn't travel before and had a good experience, we are likely to travel that route in the future. **This shows that current actions (traveling a route) definitely decides the future actions and hence the future rewards (travel time in future).**

The
dilemma



Time

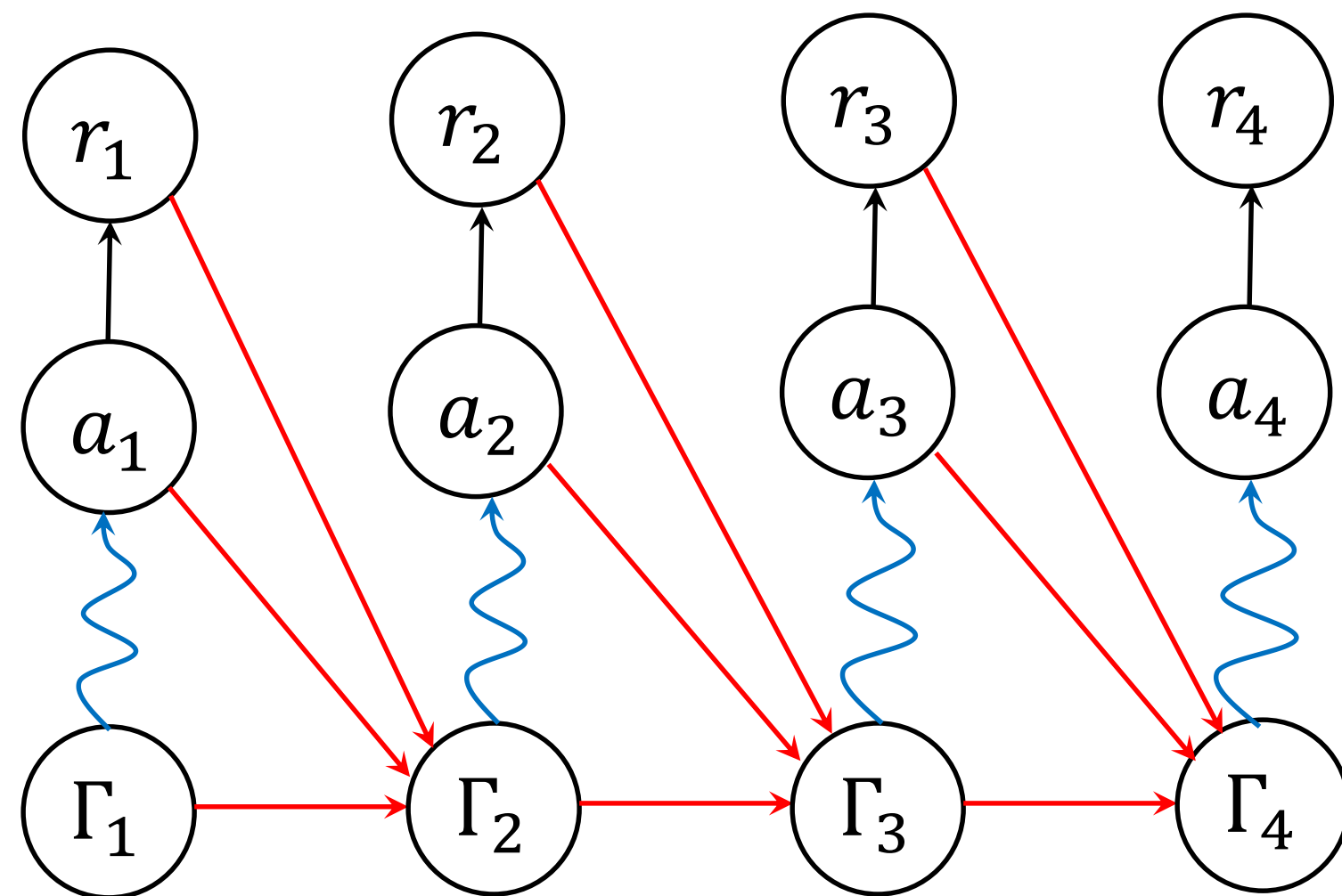
Special Cases of Markov Decision Process



A Conceptual Dilemma about Bandits

- This contradiction arises because the PGM in the previous slide has to be modified for the planning + learning scenario.
- In the planning + learning scenario, the **“experience”** is used to update the agents estimate of the probabilistic model of the system. Let Γ_t denote the estimate of the probabilistic model at time t .
- Γ_{t+1} (updated model), is dependent on Γ_t (current model), a_t (the current action that the agent takes), and r_t (the current experience).
- The action at time t is dependent on the estimate of the probabilistic model at time t (blue arrow).

Special Cases of Markov Decision Process



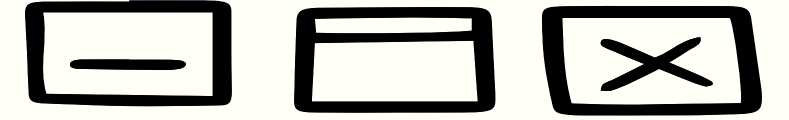
$$R_{discounted} = r_0 + \beta r_1 + \beta^2 r_2 + \beta^3 r_3 + \dots$$

Time

A Conceptual Dilemma about Bandits

- As it can be clearly seen from the PGM, there is clearly a temporal dependence for MAB and contextual bandits. Hence, the **current action does decide future reward**.

Lecture Content

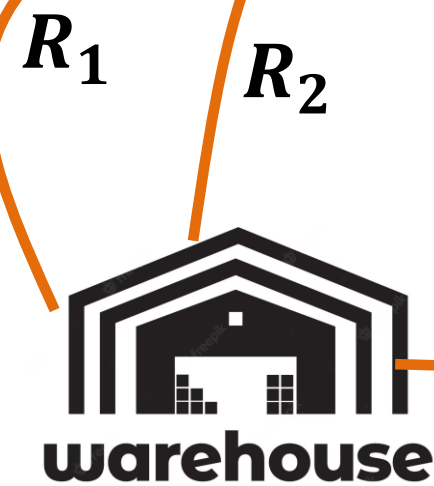


- Stochastic process (very brief introduction; only that what is required for this course).
- Markov Decision Process and Reinforcement Learning.
- Special Cases of Markov Decision Process.
- Stationary vs Non-Stationary Process.
- Objective functions in Reinforcement Learning.
- Reinforcement Learning vs

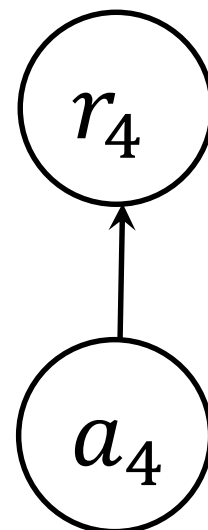
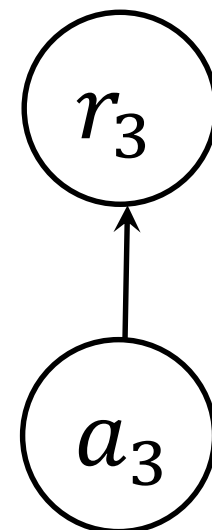
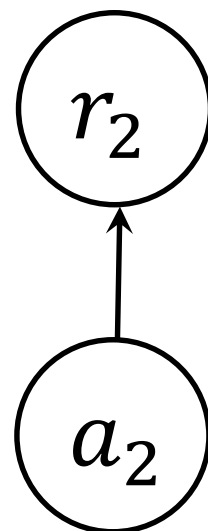
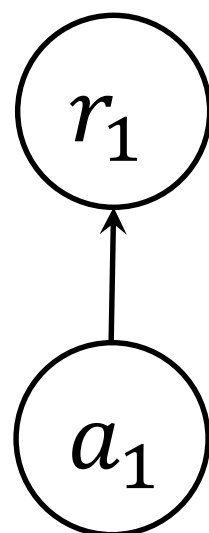
Stationary vs Non-Stationary Process

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University



R_3

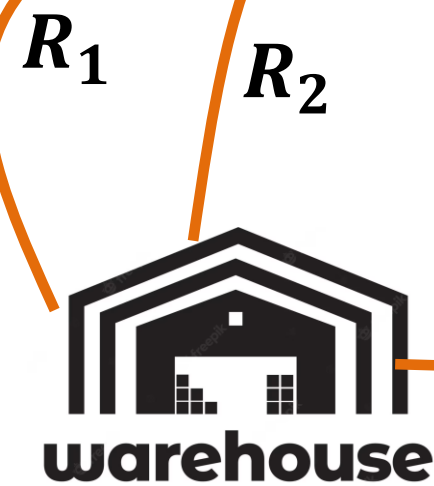


- We will be dealing with stationary stochastic processes for like 95% of this course.
- The actual definition of Stationary/Non-Stationary Stochastic process is quite involved and so I will only explain it with an example over here.
- Consider the warehouse routing example which can be formulated as a MAB.
- Let time step t denote the t^{th} hour of the day, e.x. from 10:00 am to 11:00 am.
- The action at time step t is $a_t \in \{R_1, R_2, R_3\}$.

Stationary vs Non-Stationary Process

NOTE: Routes R_1 and R_2 merges at one point

Mahindra University

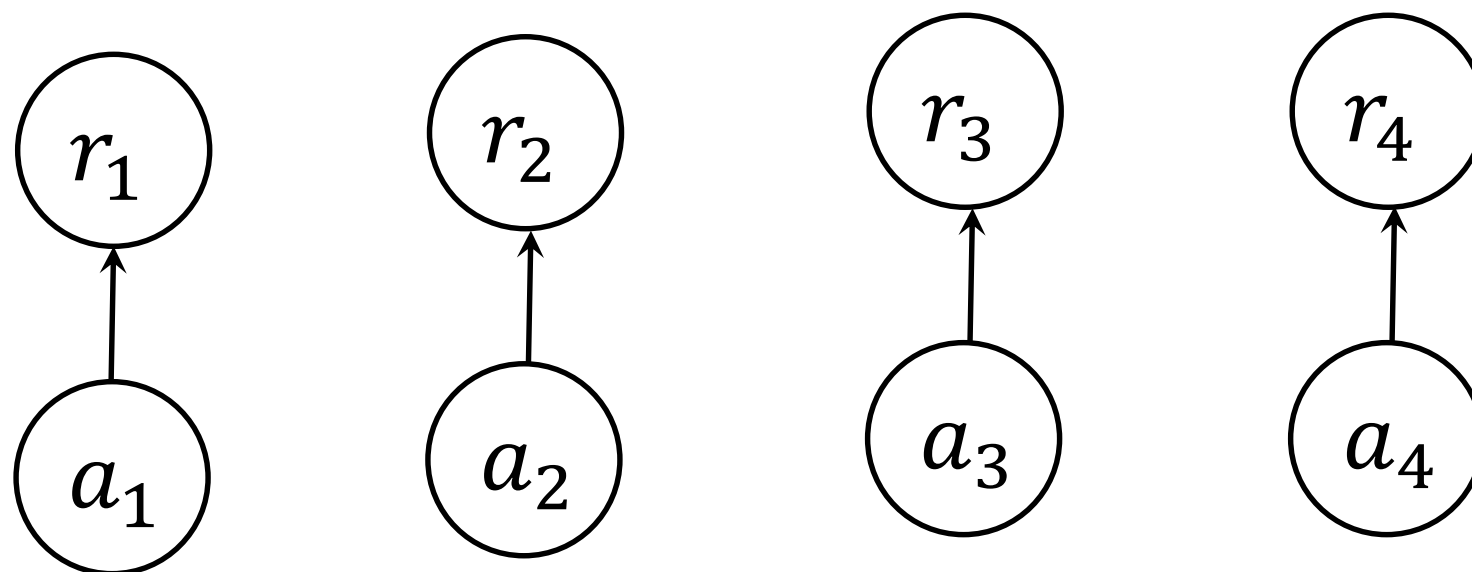


R_3

Let's consider two different costs:

1. Fuel cost

- For a given route, the fuel cost at different time of the day **does NOT change** much the hour of the day.
- Let θ_t denote the fuel cost at time t .
- Let $F_k = P[\theta_t | a_t = k]$ denote the probability density function of the fuel cost for route k .
- Notice that probability density function F_k is **NOT dependent on t** . This is because the statistical property of fuel cost does not change (much) with the hour of the day. Hence, a **stationary process**.
- **Important:** Even though F_k is NOT dependent on t , θ_t that is sampled from F_k (denoted by $\theta_t \sim F_k$) **DOES CHANGE** with time.



Stationary vs Non-Stationary Process

NOTE: Routes R_1 and R_2 merges at one point

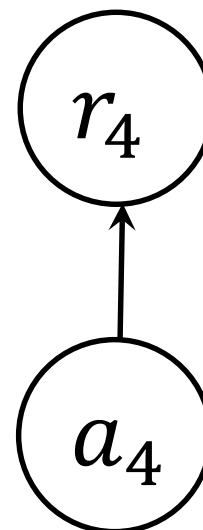
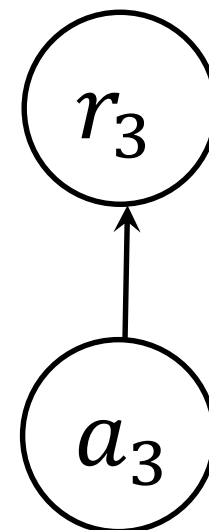
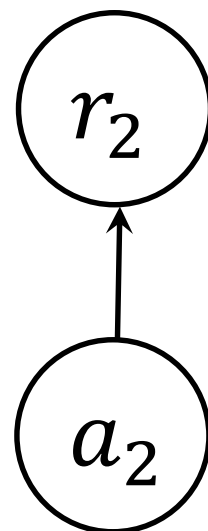
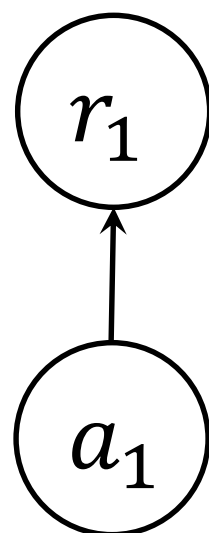
Mahindra University



R_1

R_2

R_3

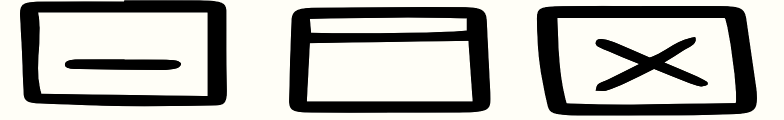


Let's consider two different costs:

2. Travel time

- For a given route, the travel time at different time of the day **does change** with the hour of the day.
- Let φ_t denote the travel time at time t .
- Let $G_{k,t} = P[\varphi_t | a_t = k]$ denote the probability density function of the travel time for route k .
- Notice that probability density function $G_{k,t}$ **is dependent on t** . This is because the statistical property of travel time does change with the hour of the day. Hence, a **non-stationary process**.

Lecture Content



- Stochastic process (very brief introduction; only that what is required for this course).
- Markov Decision Process and Reinforcement Learning.
- Special Cases of Markov Decision Process.
- Stationary vs Non-Stationary Process.
- Objective functions in Reinforcement Learning.
- Reinforcement Learning vs

Objective functions in Reinforcement Learning

Since reinforcement learning (RL) is essentially an optimization problem, one of the first step is to decide the objective function. The following objective functions are widely used in RL.

1. Sum of rewards

$$R_{sum} = \sum_{t=0}^T r_t$$

➤ Here T is the time horizon and it is **finite**.

Objective functions in Reinforcement Learning

Since reinforcement learning (RL) is essentially an optimization problem, one of the first step is to decide the objective function. The following objective functions are widely used in RL.

2. Average reward

$$R_{avg} = \liminf_{T \rightarrow \infty} \frac{1}{T+1} \sum_{t=0}^T r_t$$

➤ Here, the time horizon T is **infinite**.

Objective functions in Reinforcement Learning

Since reinforcement learning (RL) is essentially an optimization problem, one of the first step is to decide the objective function. The following objective functions are widely used in RL.

3. Discounted reward

$$R_{discounted} = \sum_{t=0}^{\infty} \beta^t r_t$$

where $\beta \in (0,1)$ is the discount factor.

- Similar to average reward, the time horizon is **infinite**.
- The expanded form of discounted reward is

$$R_{discounted} = r_0 + \beta r_1 + \beta^2 r_2 + \beta^3 r_3 + \dots$$

Since $\beta \in (0,1)$, the weight β^t associated with reward decreases with time. In other words, for a given time step, the reward in that time step is weighted more than reward in future time step. This qualitatively means that **reward today is better than reward tomorrow**.

Objective functions in Reinforcement Learning

Since reinforcement learning (RL) is essentially an optimization problem, one of the first step is to decide the objective function. The following objective functions are widely used in RL.

3. Discounted reward

$$R_{discounted} = \sum_{t=0}^{\infty} \beta^t r_t$$

where $\beta \in (0,1)$ is the discount factor.

- In this course, we will be dealing only with discounted rewards and not average rewards because of three reasons.

Reason 1: The **policy for average reward can be obtained from the policy for discounted reward** by simply letting $\beta \rightarrow 1$.

Reason 2: From mathematical perspective, it is **much easier** to deal with discounted reward than average reward. Average reward has many “corner cases” that makes its analysis tedious.

Objective functions in Reinforcement Learning

Since reinforcement learning (RL) is essentially an optimization problem, one of the first step is to decide the objective function. The following objective functions are widely used in RL.

3. Discounted reward

$$R_{discounted} = \sum_{t=0}^{\infty} \beta^t r_t$$

where $\beta \in (0,1)$ is the discount factor.

- In this course, we will be dealing only with discounted rewards and not average rewards because of three reasons.

Reason 3: When the underlying stochastic process is **non-stationary**, discounted reward is likely to generate better outcome than average reward. This is because when the statistical property of the system is changing with time, it is a good idea to focus on the current reward (and maybe rewards few steps ahead) and not planning for the rewards much ahead of the current time step. Discounted reward achieves this by assigning more weightage to current reward than future rewards (average reward assigns equal weightage to all rewards).

