Introduction to Reinforcement Learning

서울시립대학교
UNIVERSITY OF SEOUL

Presentation: Baekryun Seong

2024.05.09

Table of Contents

Ch 1.1
Reinforcement Learning

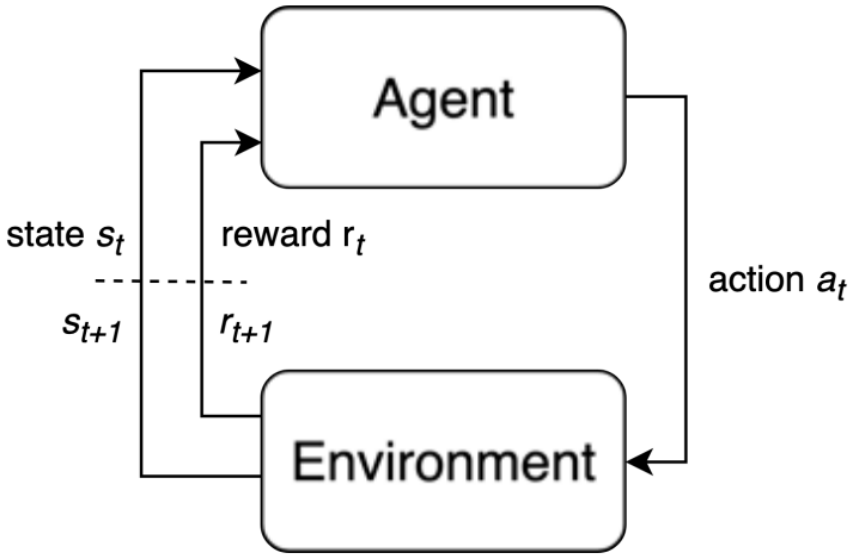


Figure 1.2 The reinforcement learning control loop

- **Environment** generates **State**
- **Agent** selects **Action**
- **Environment** accepts the **action** and returns **the next state and a reward.**
- When the cycle of (state → action → reward) completes, we say that **one time step** has passed.

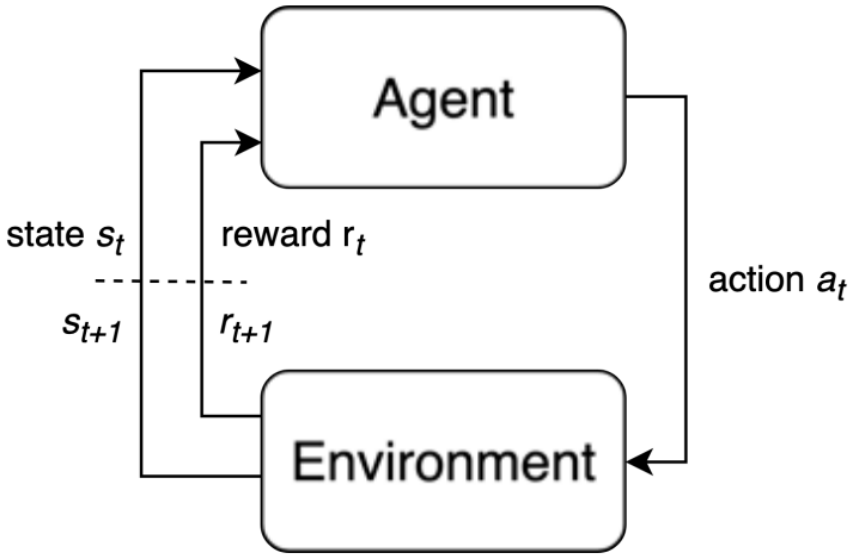


Figure 1.2 The reinforcement learning control loop

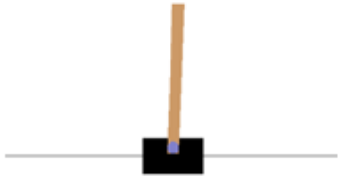
- **Policy** is an agent's action-producing function, which maps states to a actions.
- RL Problems have an **objective**, which is the sum of rewards received by an agent.
- An agent uses the reward signals it receives to **reinforce** good actions.

An **experience** is 3-tuple which consist of state, action, reward at some timestep.

$$\text{Experience} = (s_t, a_t, r_t)$$

A **trajectory** τ is a sequence of experiences over an episode.

$$\tau = \{(s_t, a_t, r_t)\}_{t=1}^{T}$$



(a) CartPole

1. **Objective**: Keep the pole upright for 200 time steps.
2. **State**: An array of length 4 which represents: [cart position, cart velocity, pole angle, pole angular velocity]. For example, [-0.034, 0.032, -0.031, 0.036].
3. **Action**: An integer, either 0 to move the cart a fixed distance to the left, or 1 to move the cart a fixed distance to the right.
4. **Reward**: +1 for every time step the pole remains upright.
5. **Termination**: When the pole falls over (greater than 12 degrees from vertical), or when the cart moves out of the screen, or when the maximum time step of 200 is reached.

$s_t \in \mathcal{S}$ is the state, $\mathcal{S}$ is the state space.

$a_t \in \mathcal{A}$ is the action, $\mathcal{A}$ is the action space.

$r_t \in \mathcal{R}(s_t, a_t, s_{t+1})$ is the reward, $\mathcal{R}$ is the reward function.

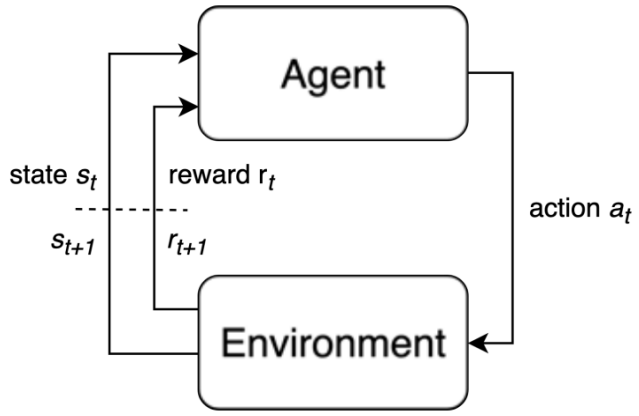


Figure 1.2 The reinforcement learning control loop

We can model the environment using probability:

$$s_{t+1} \sim P(s_{t+1} | (s_0, a_0), (s_1, a_1), \ldots, (s_T, a_T))$$

or,

$$s_{t+1} \sim P(s_{t+1} | \tau).$$

Ch1.2
Markov Decision Process

So, now we have environment model and can select action using it.

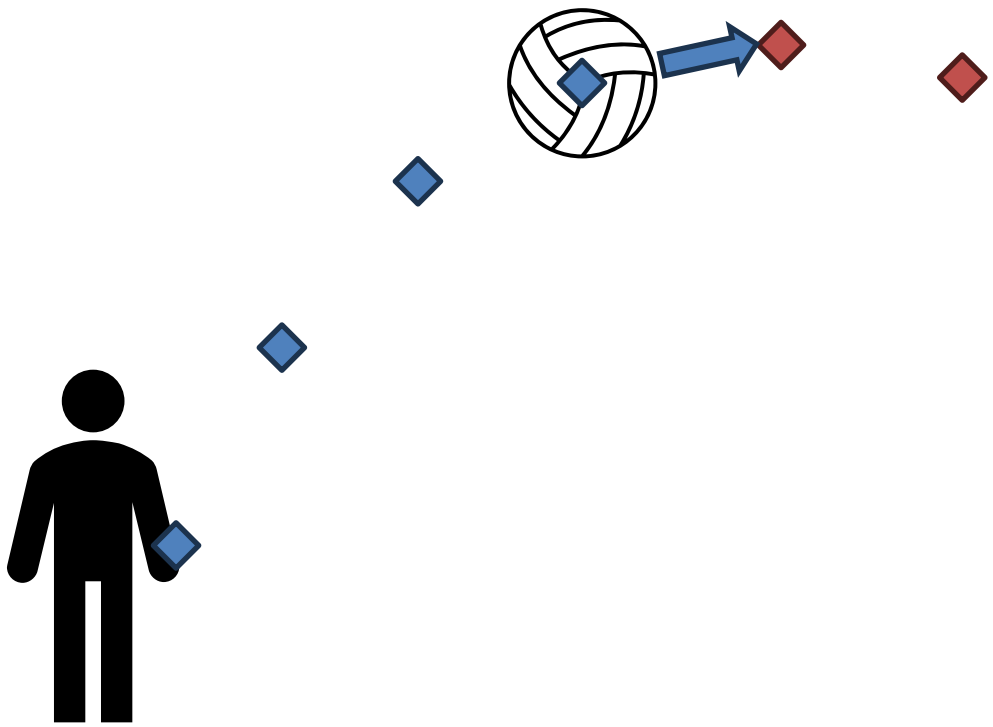
$$s_{t+1} \sim P(s_{t+1} | (s_0, a_0), (s_1, a_1), \dots, (s_T, a_T))$$

But now **the agent's action space is too large**! In worst case, the action function is:

$$a(\tau): \mathcal{S}^T \rightarrow \mathcal{A}$$

It can't be used in practice! We must reduce the action space.

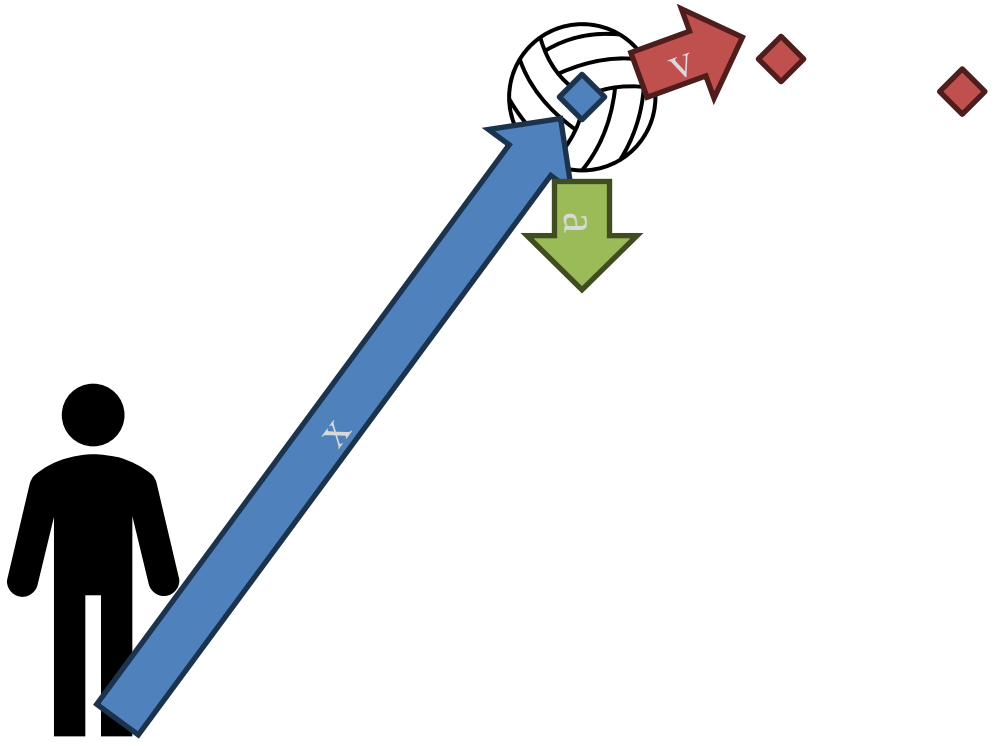
We can predict the future position of ball using position trajectory. In this example, state is 3-tuple, (x, y, z), and action function takes all trajectory (or at least 3 previous states)!
But we know the other way to predict future position.



When we define state as 9-tuple

$$(x, y, z, v_x, v_y, v_z, a_x, a_y, a_z),$$

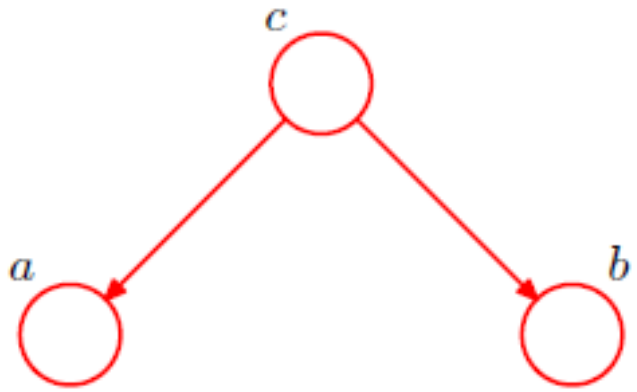
we can predict future position without using full trajectory but using only current state!



Now we define conditional probability as

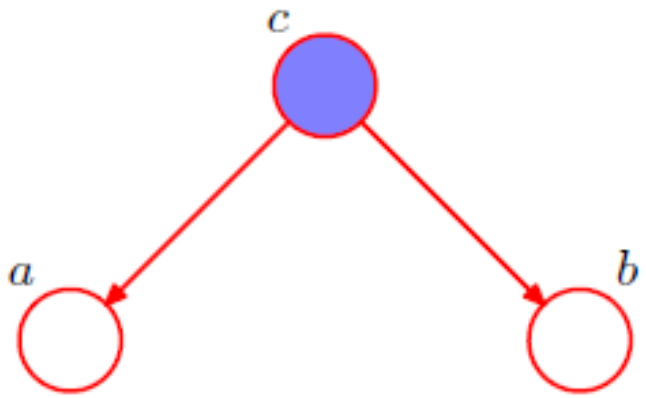
a is conditionally independent to b given c $\equiv p(a|b,c) = p(a|c)$.

We can draw a directed graph representing this:



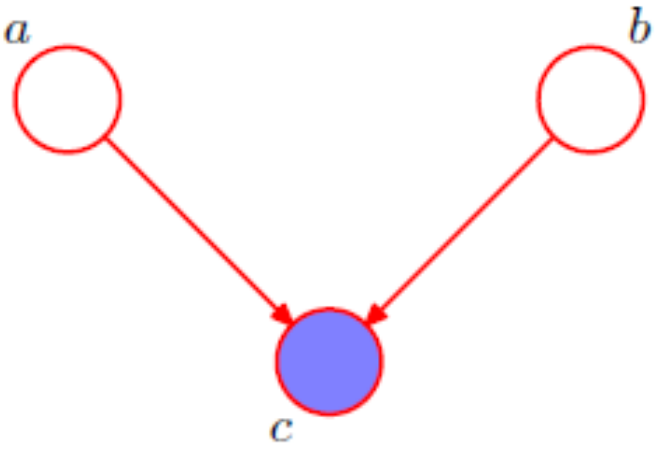
$$p(a,b|c) = p(a|c)p(b|c)$$

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.



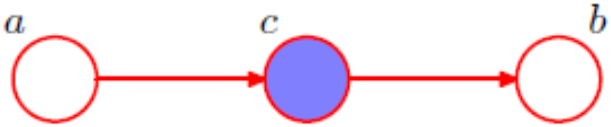
Tail-to-tail node

$$p(a, b|c) = p(a|c)p(b|c)$$



Head-to-head node

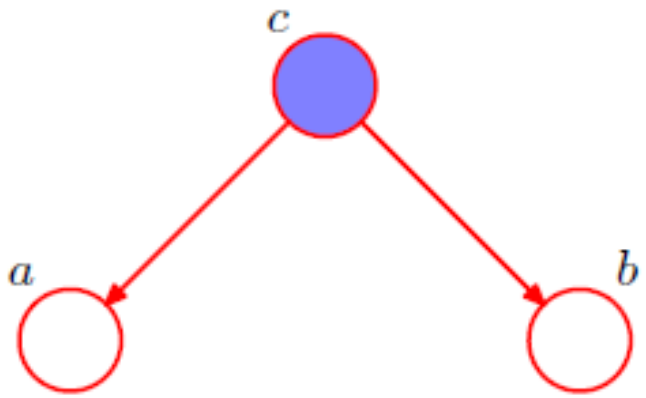
$$p(a, b|c) \neq p(a|c)p(b|c)$$



Head-to-tail node

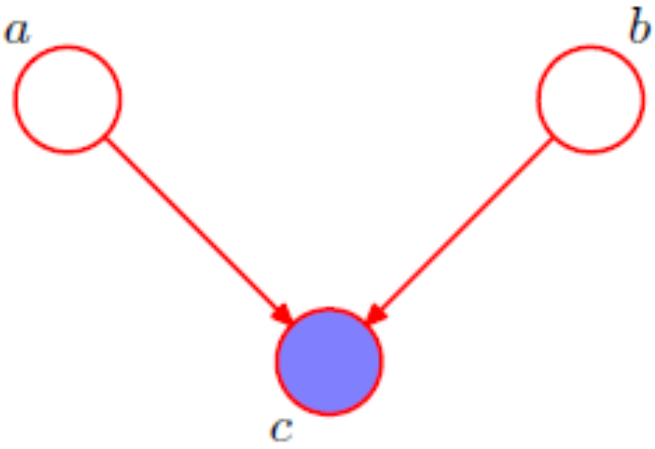
$$p(a, b|c) = p(a|c)p(b|c)$$

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.



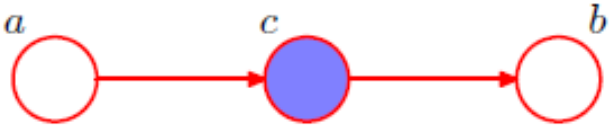
Tail-to-tail node

$$p(a, b|c) = p(a|c)p(b|c)$$



Head-to-head node

$$p(a, b|c) \neq p(a|c)p(b|c)$$



Head-to-tail node

$$p(a, b|c) = p(a|c)p(b|c)$$

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Example of head-to-head connection

Setting

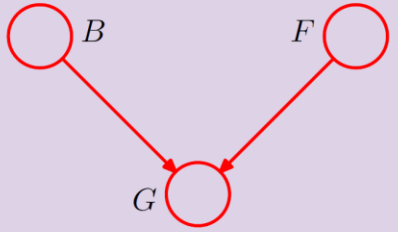
- A fuel system in a car:

  battery B, either charged ($B = 1$) or flat ($B = 0$)
  fuel tank F, either full ($F = 1$) or empty ($F = 0$)
  electric fuel gauge G, either full ($G = 1$) or empty ($G = 0$)

Conditional probability tables (CPT)

- Battery and tank have independent prior probabilities:

  $P(B = 1) = 0.9 \qquad P(F = 1) = 0.9$



- The fuel gauge is conditioned on both (unreliable!):

$$P(G = 1|B = 1, F = 1) = 0.8 \quad P(G = 1|B = 1, F = 0) = 0.2$$
$$P(G = 1|B = 0, F = 1) = 0.2 \quad P(G = 1|B = 0, F = 0) = 0.1$$

https://disi.unitn.it/~passerini/teaching/2023-2024/MachineLearning_AIS/slides/08_bayesian_networks/talk.pdf

Example of head-to-head connection

Setting

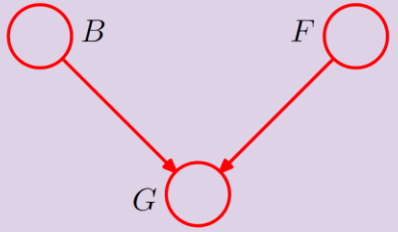
- A fuel system in a car:

  battery B, either charged ($B = 1$) or flat ($B = 0$)
  fuel tank F, either full ($F = 1$) or empty ($F = 0$)
  electric fuel gauge G, either full ($G = 1$) or empty ($G = 0$)

Conditional probability tables (CPT)

- Battery and tank have independent prior probabilities:

  $P(B = 1) = 0.9 \qquad P(F = 1) = 0.9$

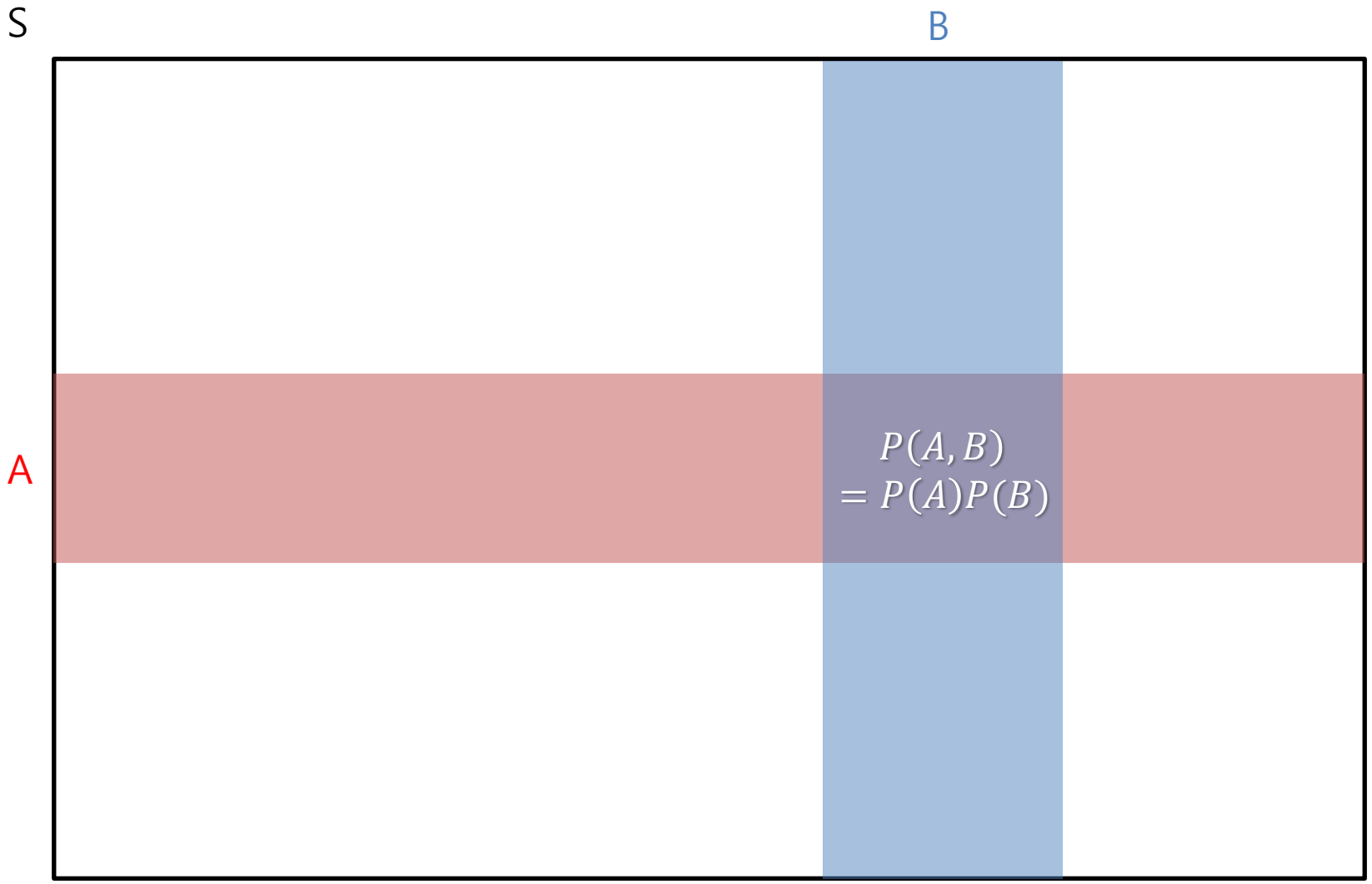


- The fuel gauge is conditioned on both (unreliable!):

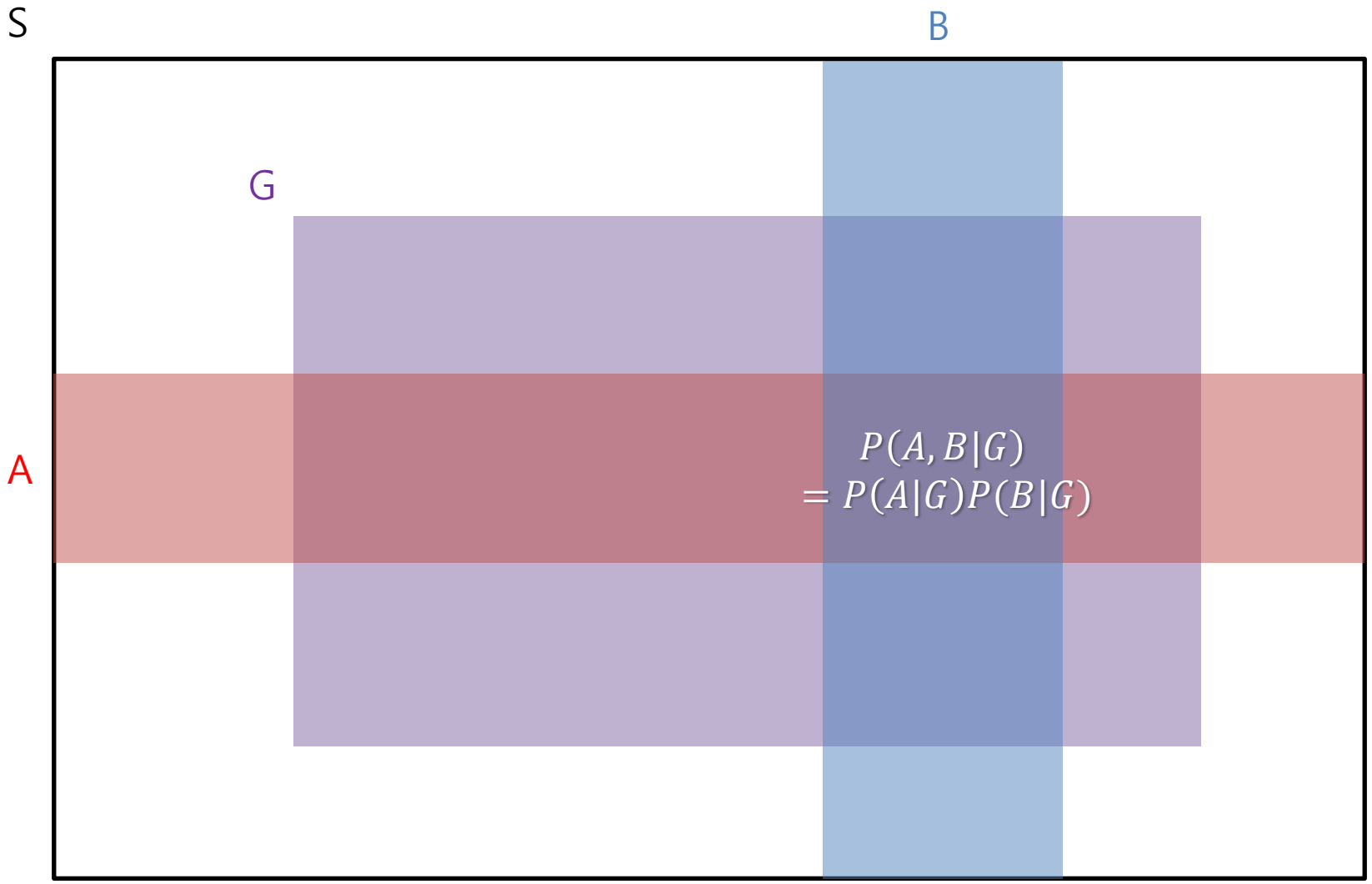
$$P(G = 1|B = 1, F = 1) = 0.8 \quad P(G = 1|B = 1, F = 0) = 0.2$$
$$P(G = 1|B = 0, F = 1) = 0.2 \quad P(G = 1|B = 0, F = 0) = 0.1$$

https://disi.unitn.it/~passerini/teaching/2023-2024/MachineLearning_AIS/slides/08_bayesian_networks/talk.pdf

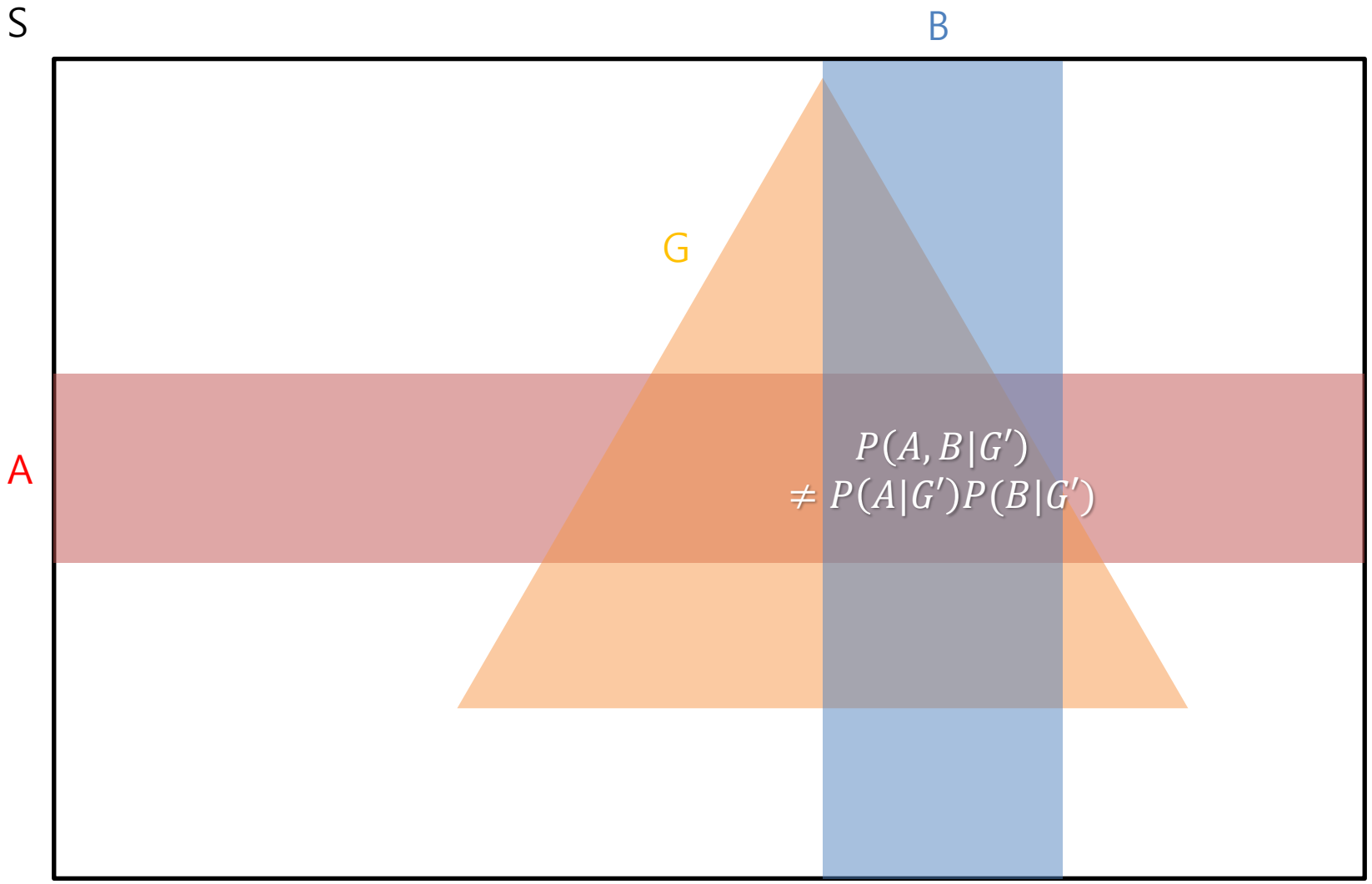
Ch 1.2 Markov Decision Process – Conditional Probability in Diagram



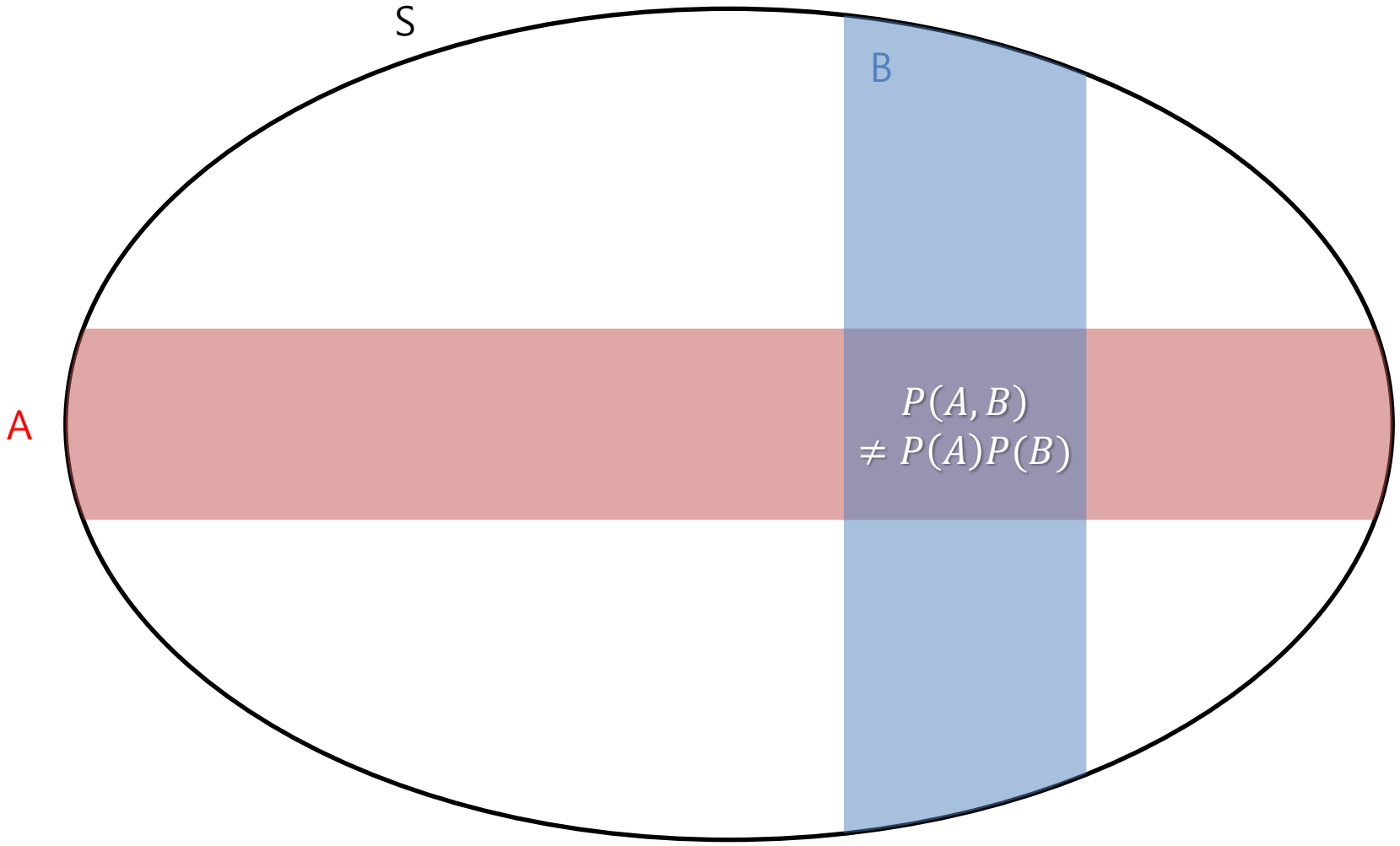
Ch 1.2 Markov Decision Process – Conditional Probability in Diagram



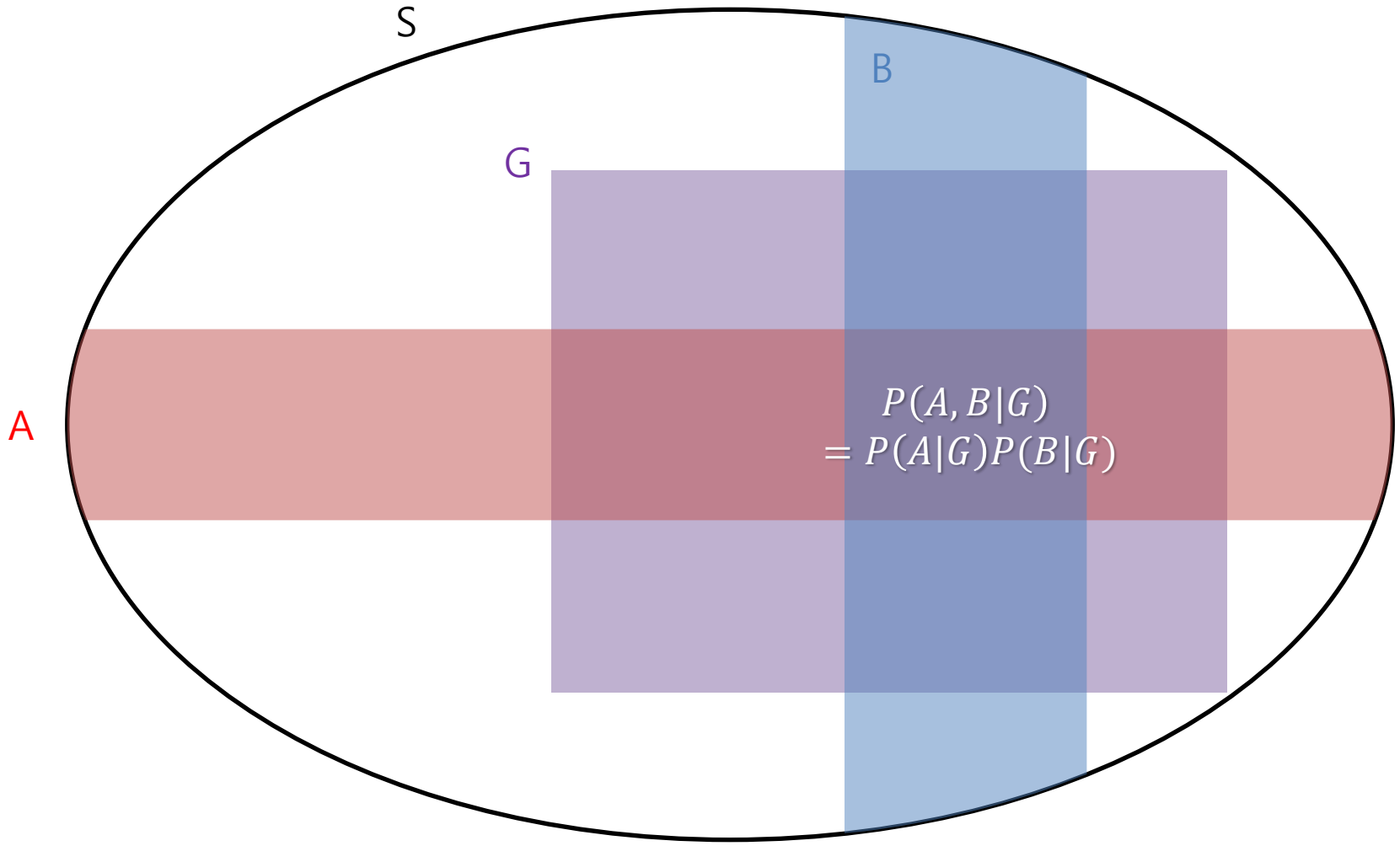
Ch 1.2 Markov Decision Process – Conditional Probability in Diagram



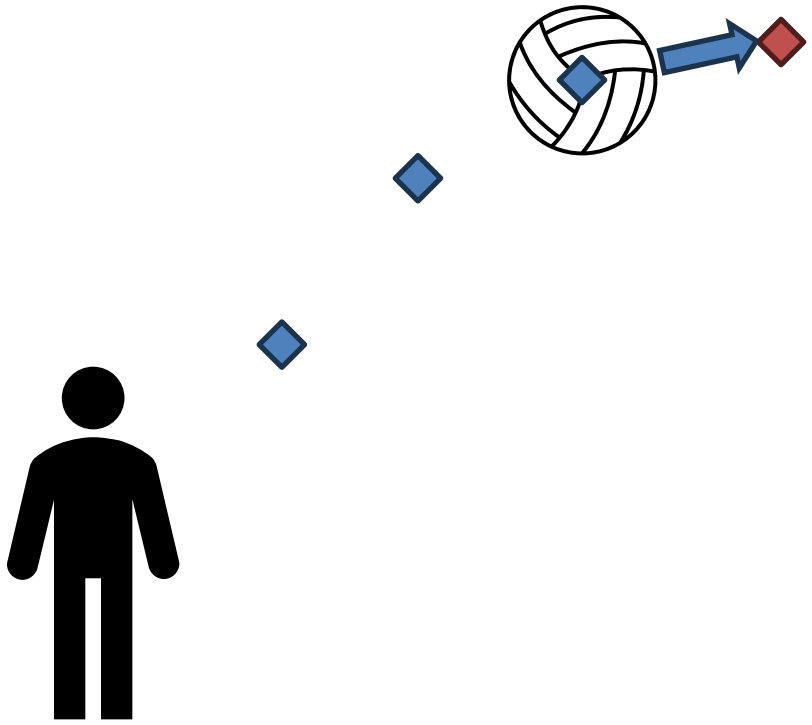
Ch 1.2 Markov Decision Process – Conditional Probability in Diagram

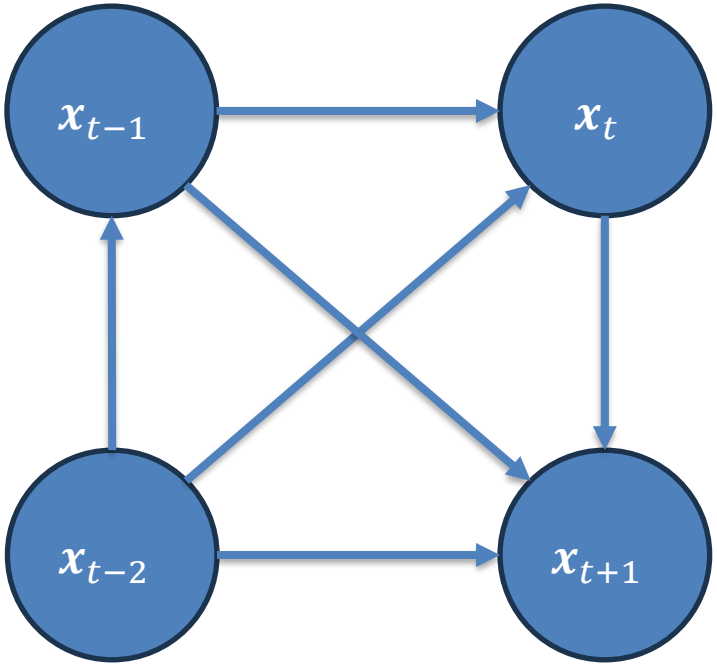


Ch 1.2 Markov Decision Process – Conditional Probability in Diagram

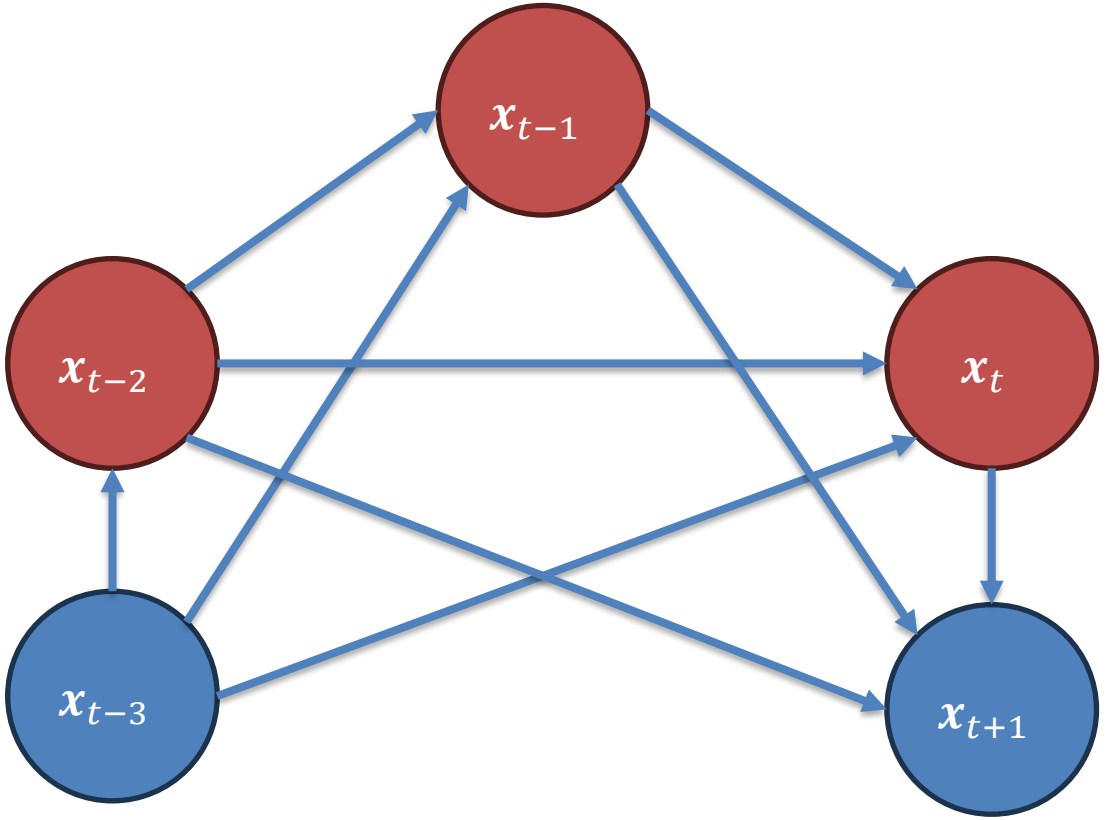


Assume we need 3 states $\{x_t, y_t, z_t\}_{t'=t-3}^{t}$ to predict next state.





When we have state $\boldsymbol{x}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t$, does $\boldsymbol{x}_{t-3}$ affect the prediction of $\boldsymbol{x}_{t+1}$?
Answer is no. $\boldsymbol{x}_{t+1}$ is not reachable from $\boldsymbol{x}_{t-3}$.



Random Process is a set of random variables indexed with parameter t.

Markov Process is a random process satisfying the equation below:

$P(x_{t+1}|x_0, x_1, \dots, x_t) = P(x_{t+1}|x_t)$ if x_k is a discrete random variable,

$P(a < x_{t+1} \leq b|x_0, x_1, \dots, x_t) = P(a < x_{t+1} \leq b|x_t)$ if x_k is a continuous random variable.

Some random variable has a **Markov Property** if it satisfies an equation above.

Alberto, L. G. Probability and Random Processes for Electrical Engineering, (1994). *University of Toronto*.

Finally, we can define Markov Decision Process.

A sequential decision problem for a **fully observable**, stochastic environment with a **Markovian transition model** and **additive rewards** is called a **Markov Decision Process**, or **MDP**, and consists of **a set of states** (with an initial state s_0); **a set ACTIONS(s)** of actions in each state; **a transition model** $p(s'|s,a)$; and **a reward function** $r(s,a,s')$.

Norvig, P. R., & Intelligence, S. A. (2002). A modern approach. *Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems*, *90*, 33-48.

Markov Decision Process, or **MDP**, is a 4-tuple

$$(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$$

where $\mathcal{S}$ is a set of states,

$\mathcal{A}$ is a set of actions,

$P(s_{t+1}|s_t, a_t)$ is a transition function of the environment,

and $\mathcal{R}(s_t, a_t, s_{t+1})$ is a reward function of the environment.

The objective of reinforcement learning is maximization of the sum of the rewards.

Graesser, L., & Keng, W. L. (2019). *Foundations of deep reinforcement learning: theory and practice in Python*. Addison-Wesley Professional.

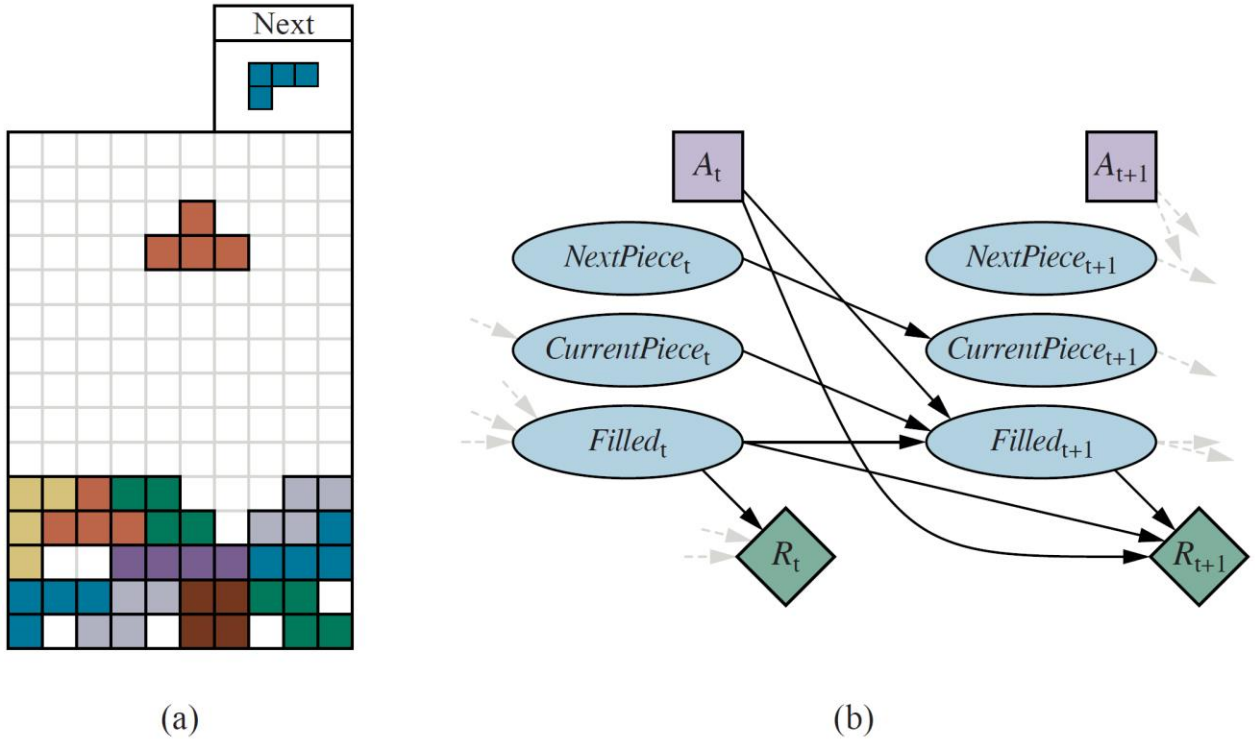


Figure 17.5 (a) The game of Tetris. The T-shaped piece at the top center can be dropped in any orientation and in any horizontal position. If a row is completed, that row disappears and the rows above it move down, and the agent receives one point. The next piece (here, the L-shaped piece at top right) becomes the current piece, and a new next piece appears, chosen at random from the seven piece types. The game ends if the board fills up to the top. (b) The DDN for the Tetris MDP.

Norvig, P. R., & Intelligence, S. A. (2002). A modern approach. *Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems*, *90*, 33-48.

The **dynamic** p of MDP is a function,

$$p(s', r|s, a) \equiv P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a).$$

A **policy** π is a function,

$$\pi(a|s) = P(A_t = a|S_t = s).$$

A **return** G_t is a sum of rewards after some timestep t,

$$G_t \equiv \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\,.$$

If $\gamma = 0$, the agent is **myopic**.

Graesser, L., & Keng, W. L. (2019). *Foundations of deep reinforcement learning: theory and practice in Python*. Addison-Wesley Professional.

State Value Function v^{π} is an expected return of a state.

$$v^{\pi}(s) = \mathbb{E}_{a\sim\pi}[G_t|s] = \mathbb{E}_{a\sim\pi}[R_{t+1} + \gamma G_{t+1}|s]$$

$$= \int_{a\in\mathcal{A}} \int_{s'\in\mathcal{S}} ds'da \cdot p(a, s'|s)\left[R_{t+1} + \gamma \mathbb{E}_{a\sim\pi}[G_{t+1}|s']\right] \because \text{def. of marginal prob.}$$

$$= \int_{a\in\mathcal{A}} \int_{s'\in\mathcal{S}} ds'da \cdot p(a|s)p(s'|s,a)\left[\int_{r\in\mathcal{R}} dr \cdot p(r|s,a,s') \cdot r + \gamma v^{\pi}(s')\right]$$

$$= \int_{a\in\mathcal{A}} \int_{s'\in\mathcal{S}} ds'da \cdot \pi(a|s)p(s'|s,a)\left[\int_{r\in\mathcal{R}} dr \cdot p(r|s,a,s') \cdot r + \gamma v^{\pi}(s')\right]$$

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Action Value Function q^{π} is an expected return about an action.

$$q^{\pi}(s,a) = \mathbb{E}_{a\sim\pi}[G_t|s,a] = \mathbb{E}_{a\sim\pi}[R_{t+1} + \gamma G_{t+1}|s,a]$$

$$= \int_{s'\in\mathcal{S}} ds' \cdot p(s'|s,a)\left[R_{t+1} + \gamma \mathbb{E}_{a\sim\pi}[G_{t+1}|s']\right]$$

$$= \int_{s'\in\mathcal{S}} ds' \cdot p(s'|s,a)\left[\int_{r\in\mathcal{R}} dr \cdot p(r|s,a,s') \cdot r + \gamma \int_{a'\in\mathcal{A}} da' \cdot p(a'|s')\mathbb{E}_{a\sim\pi}[G_{t+1}|s',a']\right]$$

$$= \int_{s'\in\mathcal{S}} ds' \cdot p(s'|s,a)\left[\int_{r\in\mathcal{R}} dr \cdot p(r|s,a,s') \cdot r + \gamma \int_{a'\in\mathcal{A}} da' \cdot \pi(a'|s')q^{\pi}(s',a')\right]$$

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.