CS 581

Advanced Artificial Intelligence

March 18, 2024

Announcements / Reminders

- **Please follow the Week 09 To Do List instructions (if you haven't already)**

- **Programming Assignment #02 due on Sunday (04/07) at 11:59 PM CST**
- **Written Assignment #03 due on Sunday (03/31) at 11:59 PM CST**

Plan for Today

- **Probabilistic Reasoning over Time**
- **A quick detour: Reinforcement Learning**

Inference in Temporal Models

Agent Belief State

Belief state: a **set of all possible environment states that the agent can be in** and needs to keep track of to handle uncertainty.

Problems:

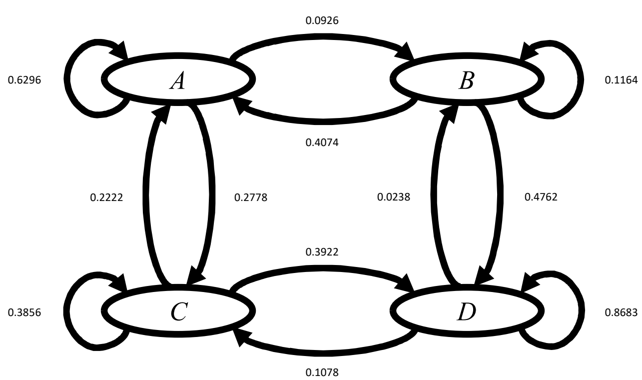
- **agent needs to consider every possible state some are going to be unlikely**
- **agent needs plans for every eventuality**
- **there may be no known plan, agent needs to act**

Agents and Belief State

Sensor Model -> Belief State

S1: A = 4, B = 7, C = 0 $\rightarrow$ P(S1)
S2: A = 4, B = 7, C = 1 $\rightarrow$ P(S2)
S3: A = 4, B = 7, C = 2 $\rightarrow$ P(S3)
S4: A = 4, B = 7, C = 3 $\rightarrow$ P(S4)

Transition Model



Sensor

A = 4 B = 7

Partially observable

A = 4

B = 7

C = 0

Agent can consult its internal representation of the world / environment to choose action

Actuator

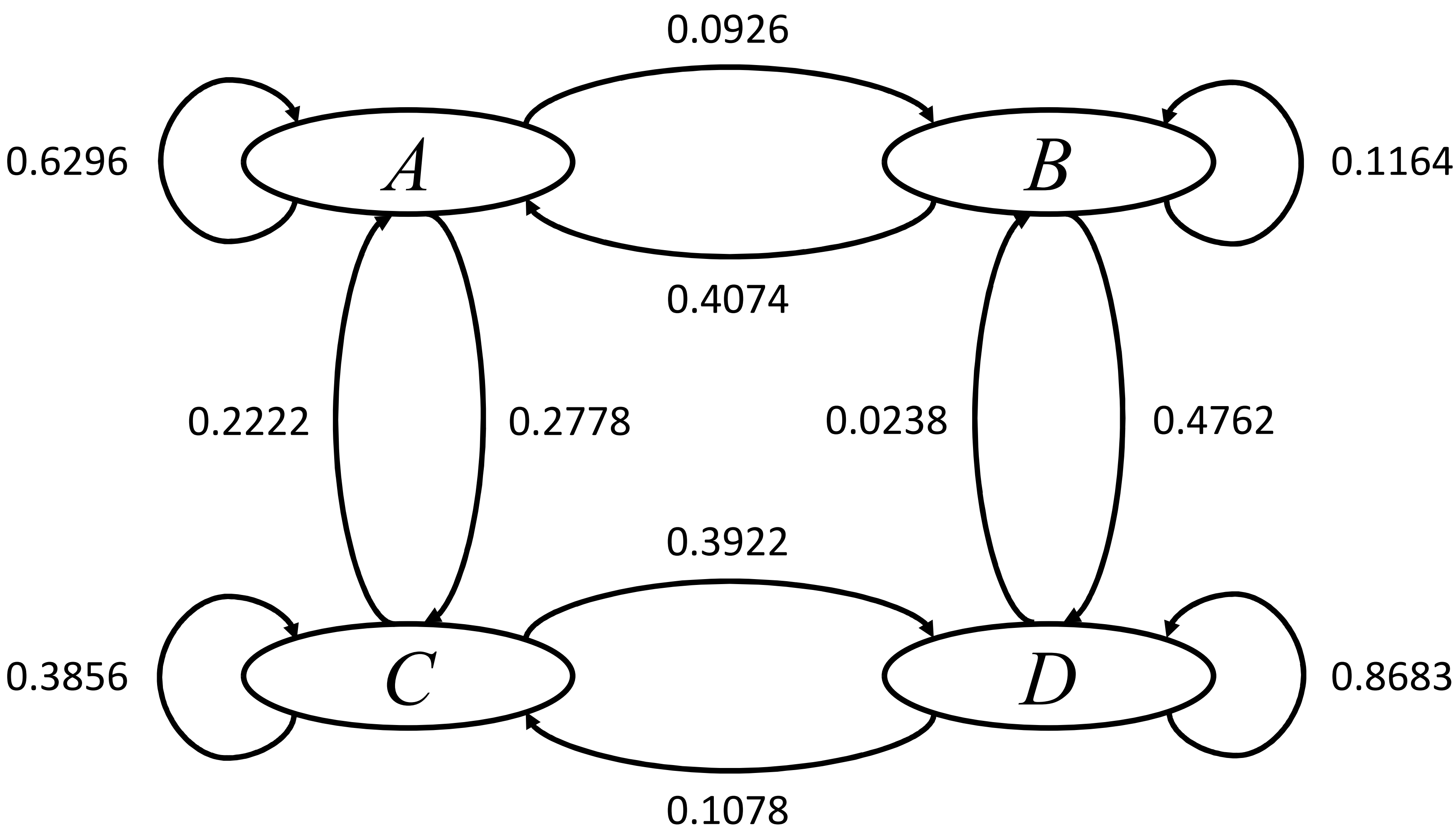
ACTION(S)

Agent

Environment

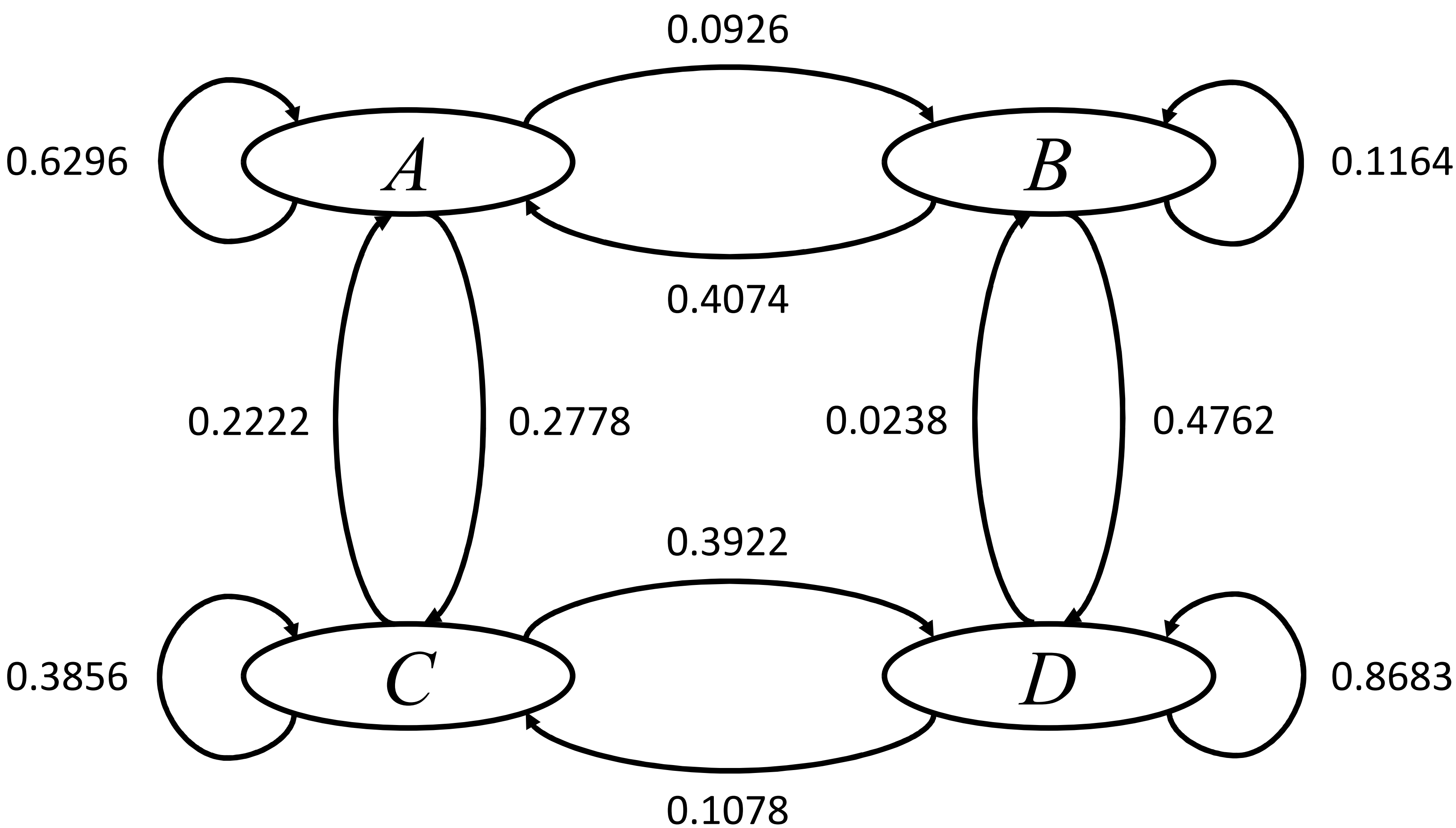
Assume: D_C = {0,1,2,3}

State Space and Transition Model



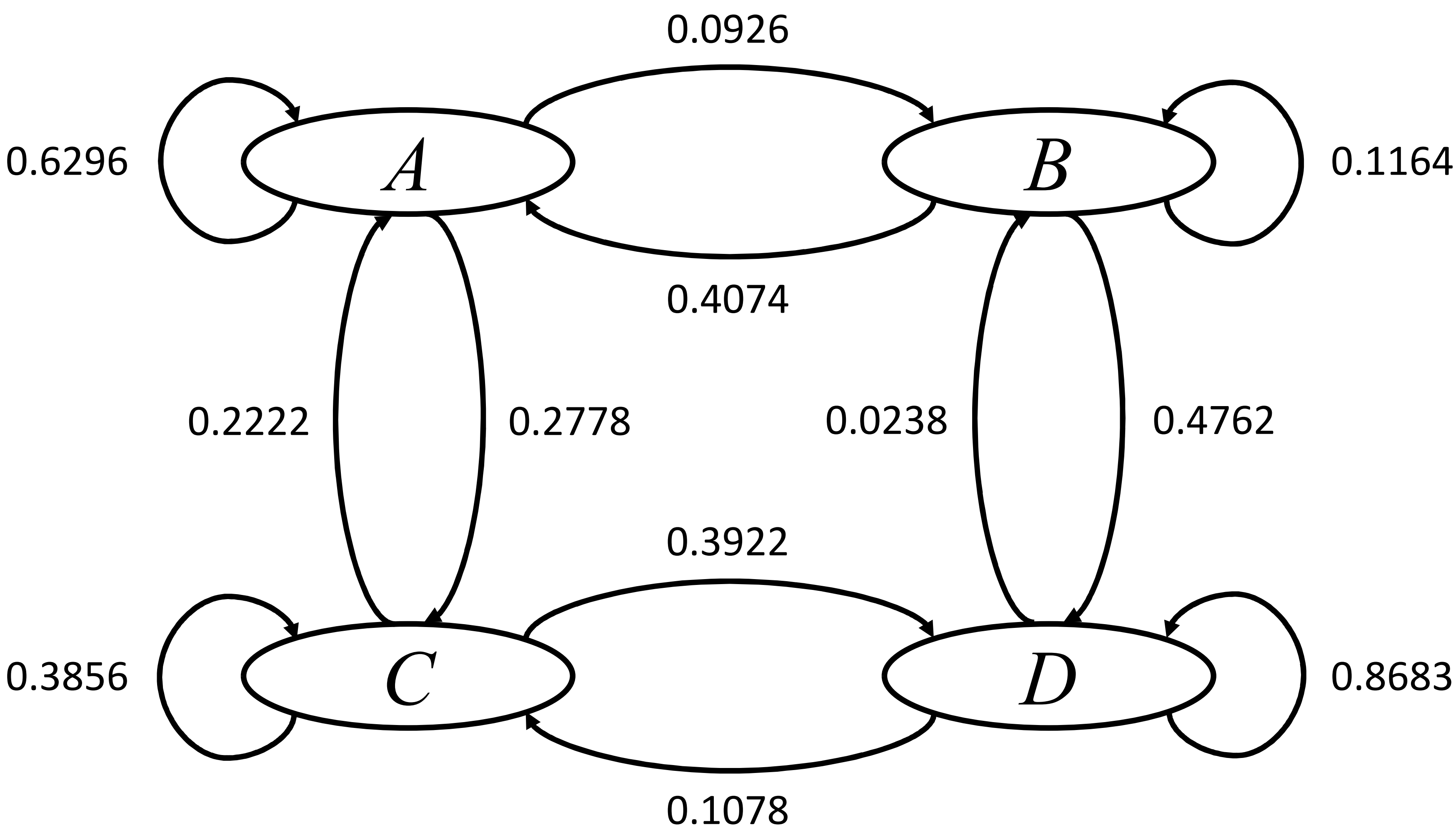
Belief State + Transition Model = Prediction

State Space and Transition Model



[Where I KNOW the WORLD CAN BE] + [How the WORLD WORKS] = [Where the WORLD CAN BE NEXT]

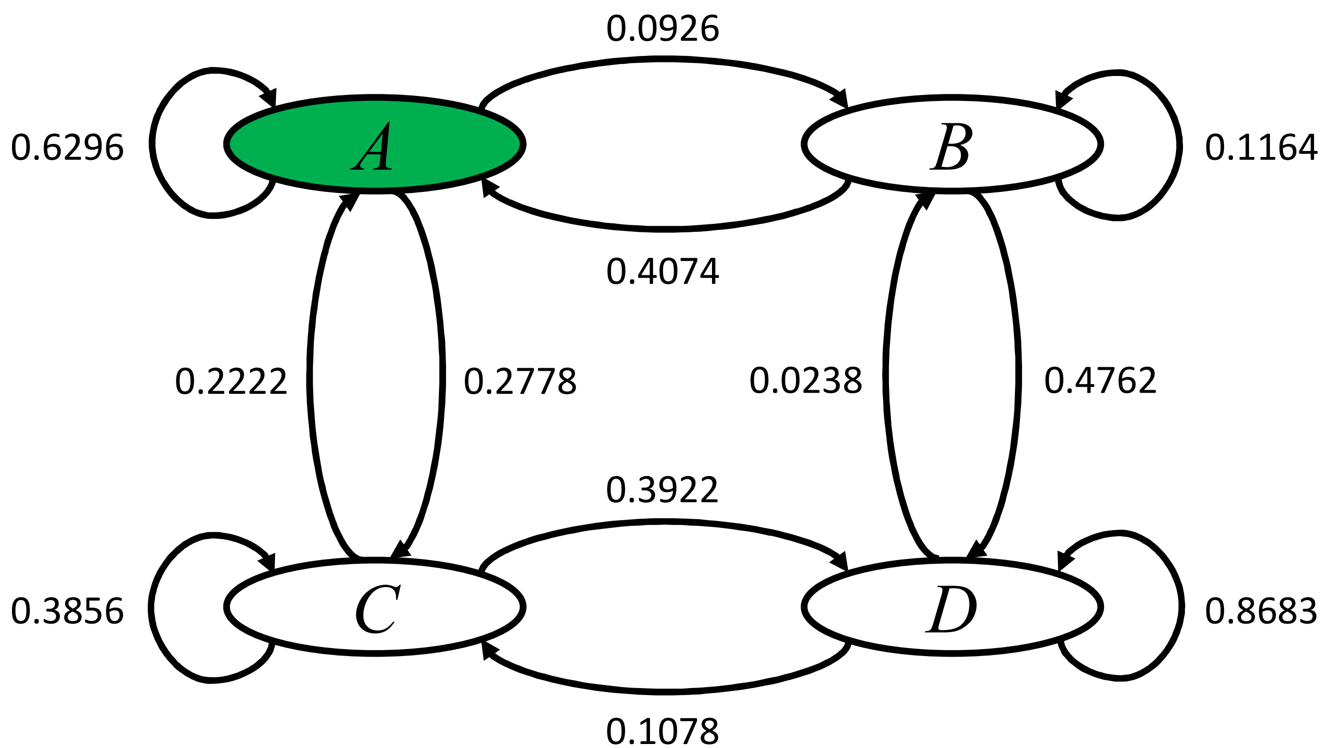
State Space and Transition Model

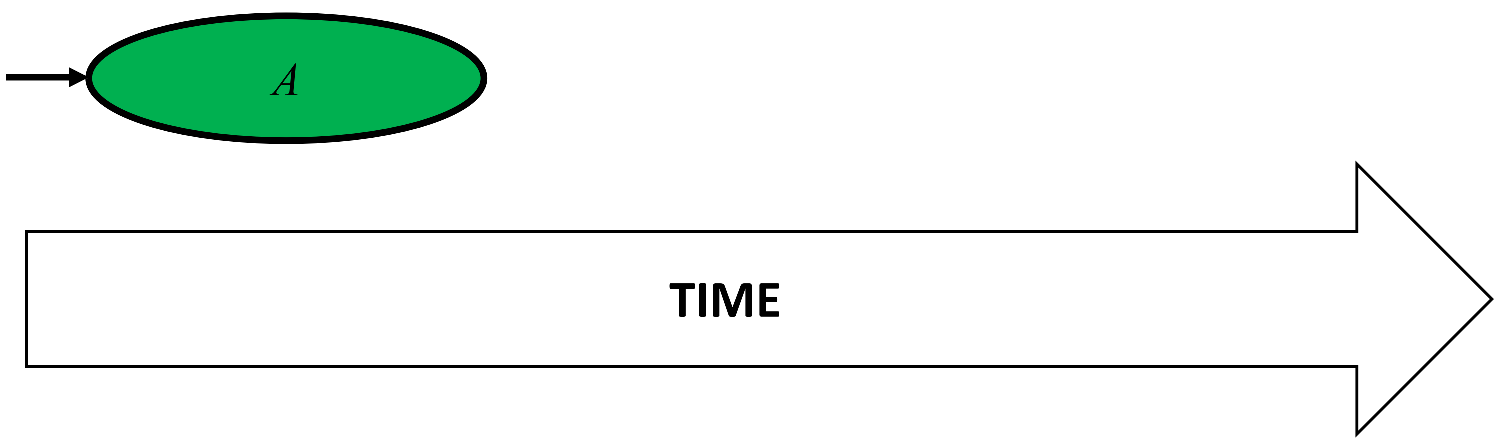


[Where I KNOW the WORLD CAN BE] + [How the WORLD WORKS] = [Where the WORLD CAN BE NEXT]

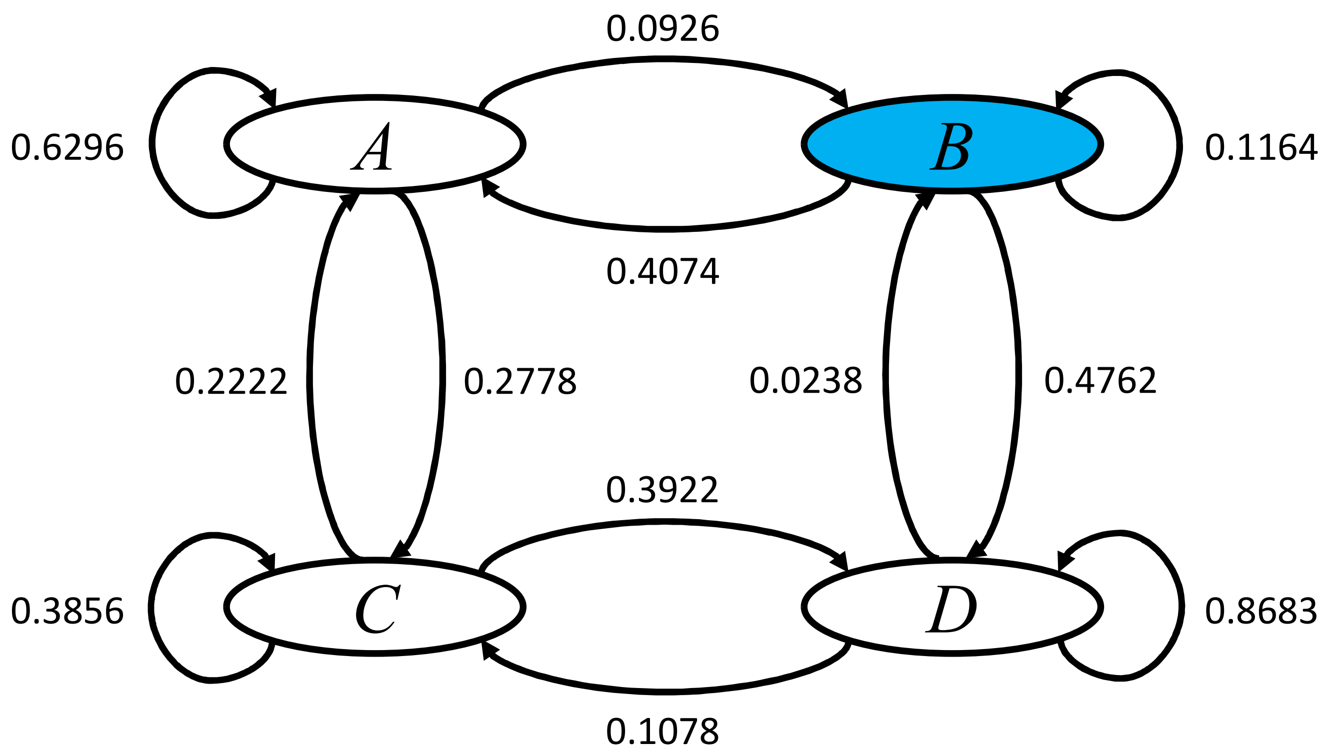
[ENVIRONMENT / WORLD is KNOWN to AGENT]

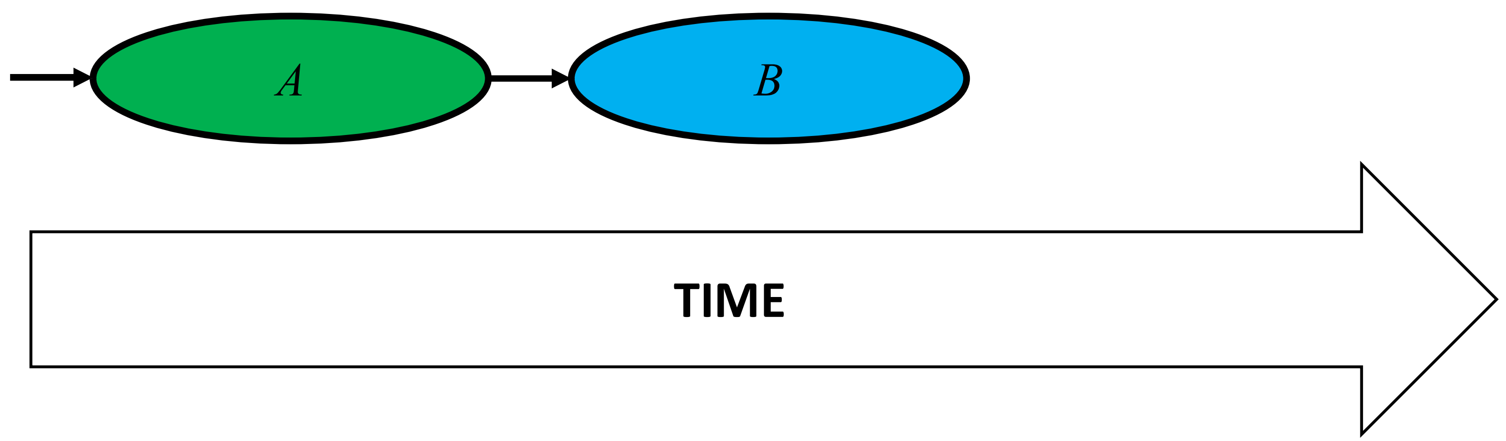
State Transitions in Time



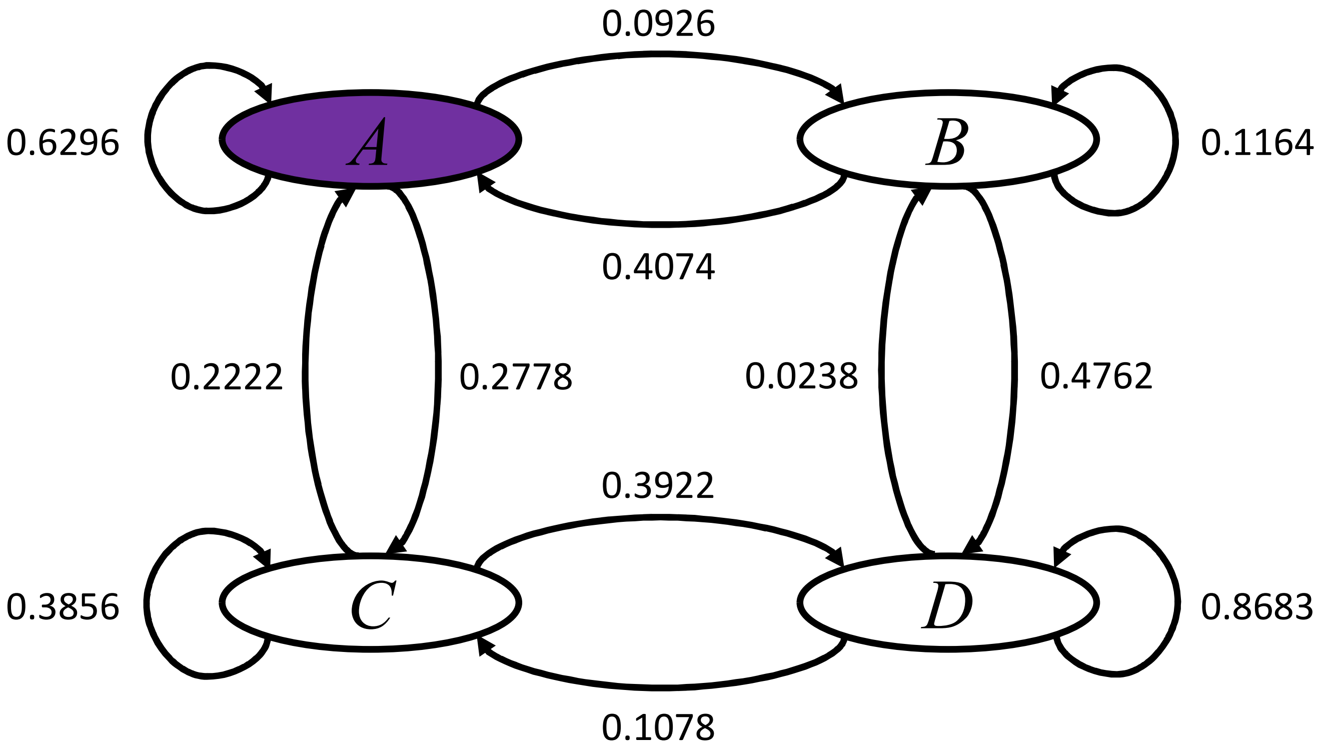


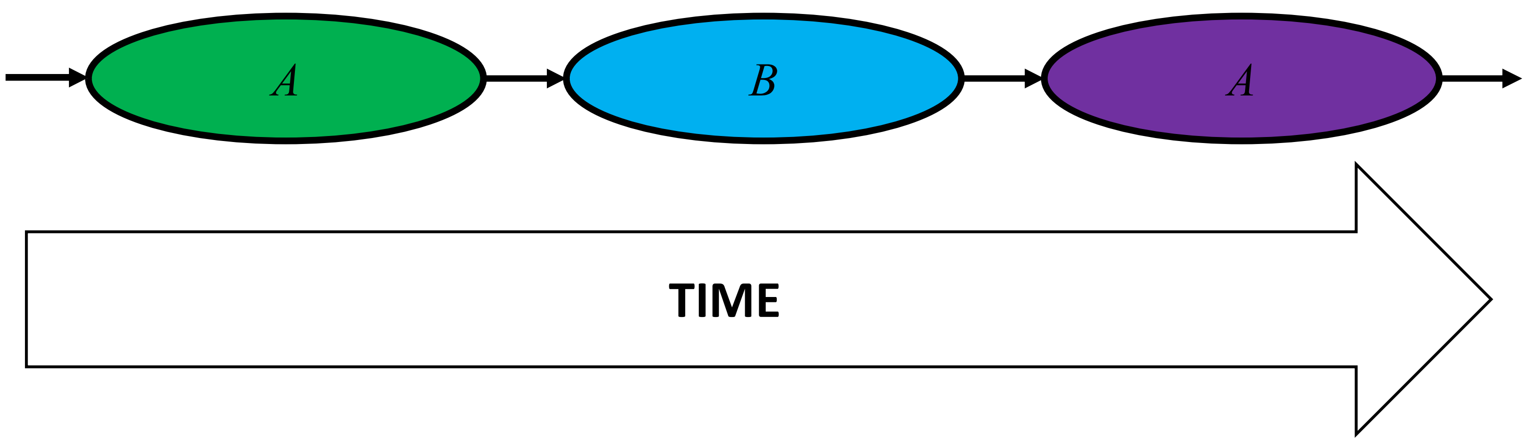
State Transitions in Time



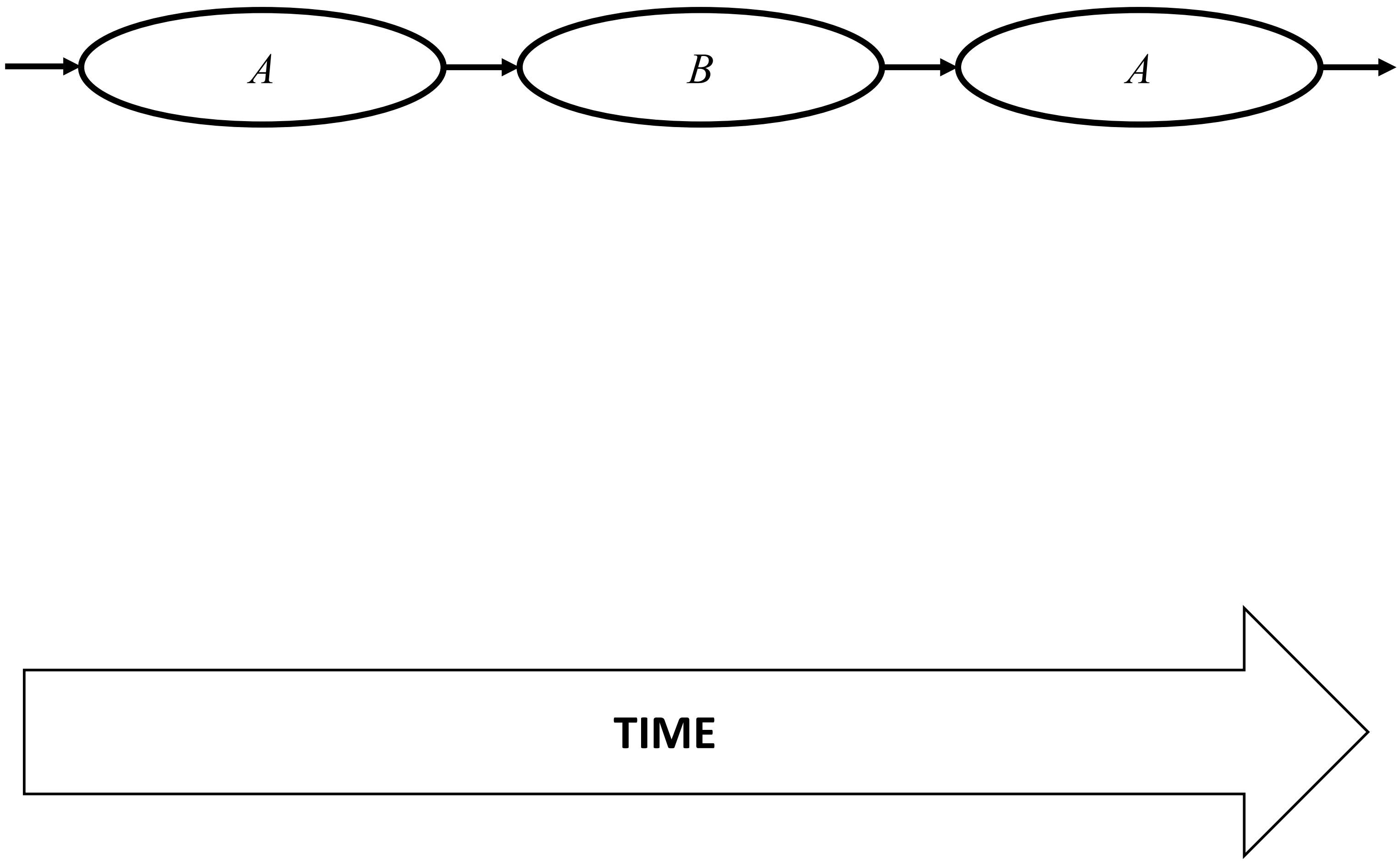


State Transitions in Time

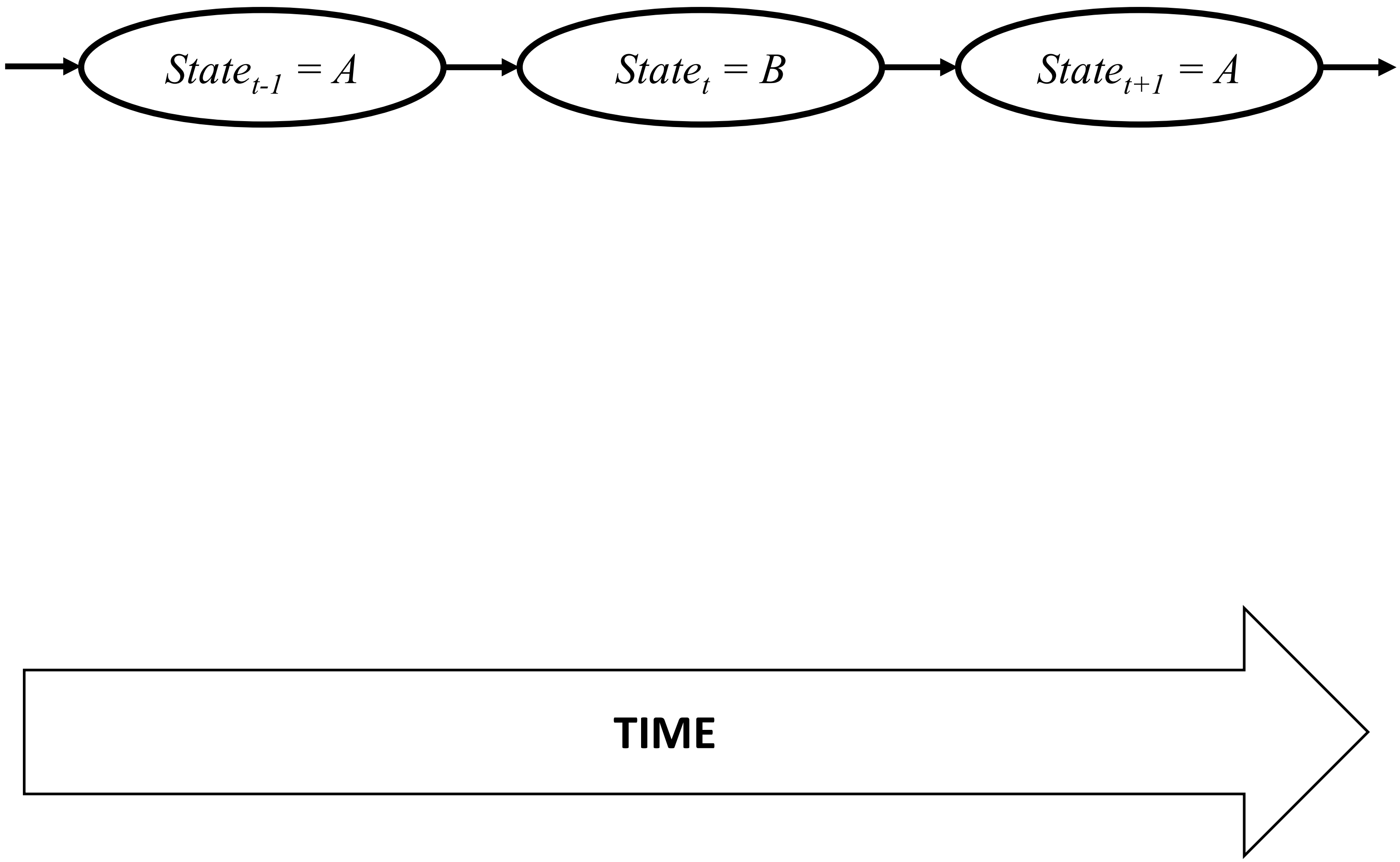




State Transitions in Time

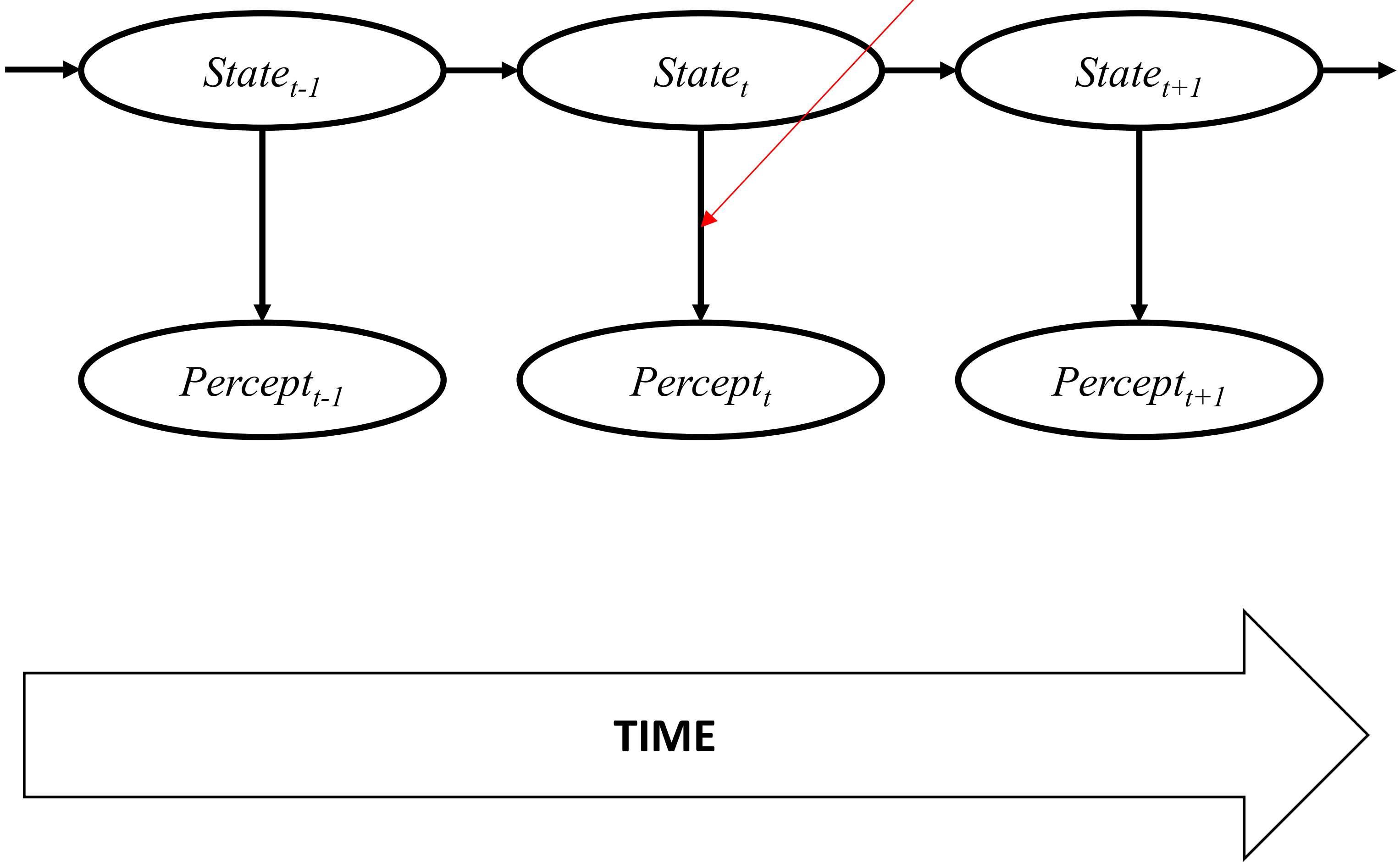


State Transitions in Time



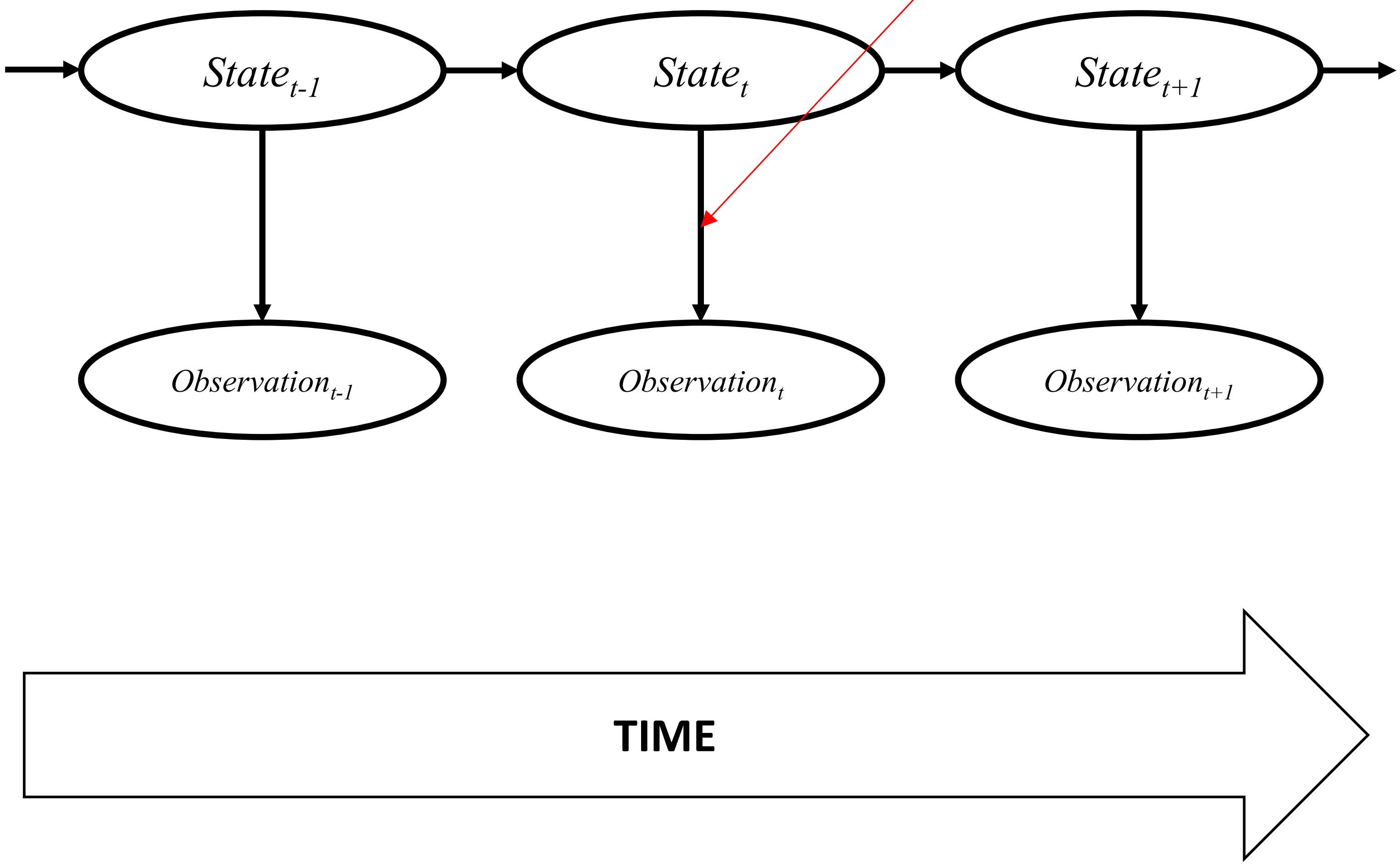
States and Observations

Environment/World **STATE** “CAUSES” the sensors to observe **PARTICULAR EVIDENCE**



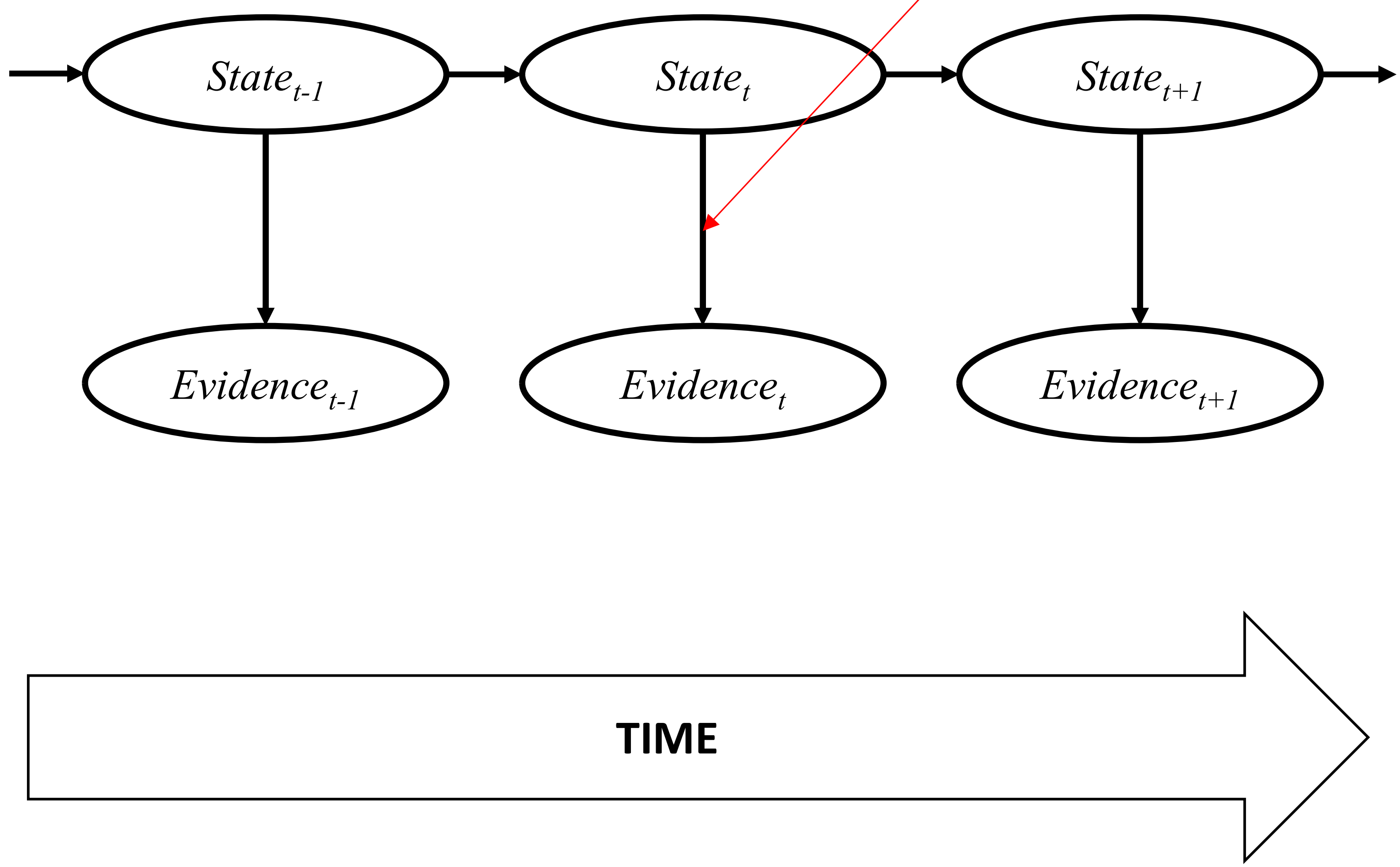
States and Observations

Environment/World **STATE** “CAUSES” the sensors to observe **PARTICULAR EVIDENCE**

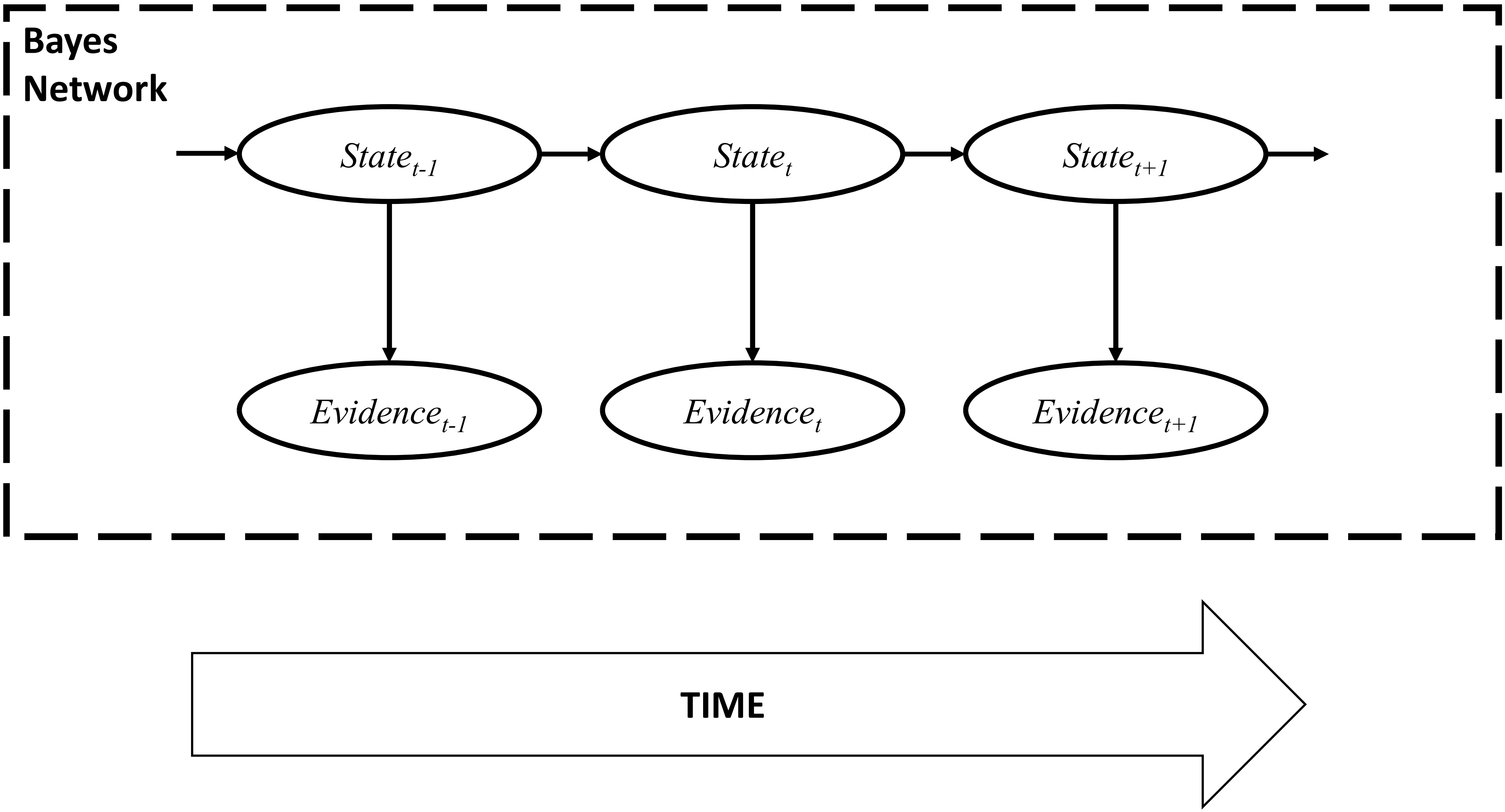


States and Observations

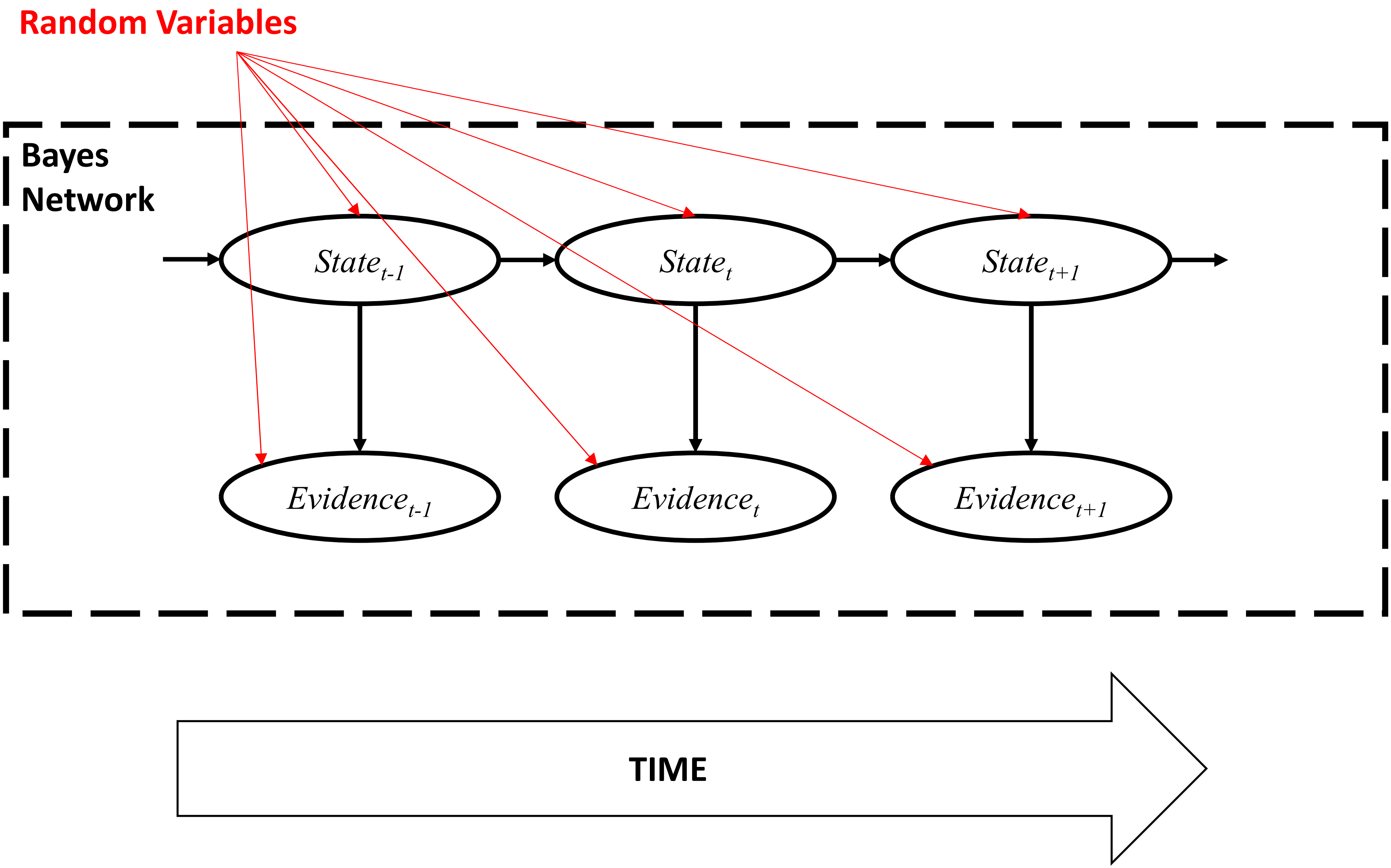
Environment/World **STATE** "CAUSES" the sensors
to observe **PARTICULAR EVIDENCE**



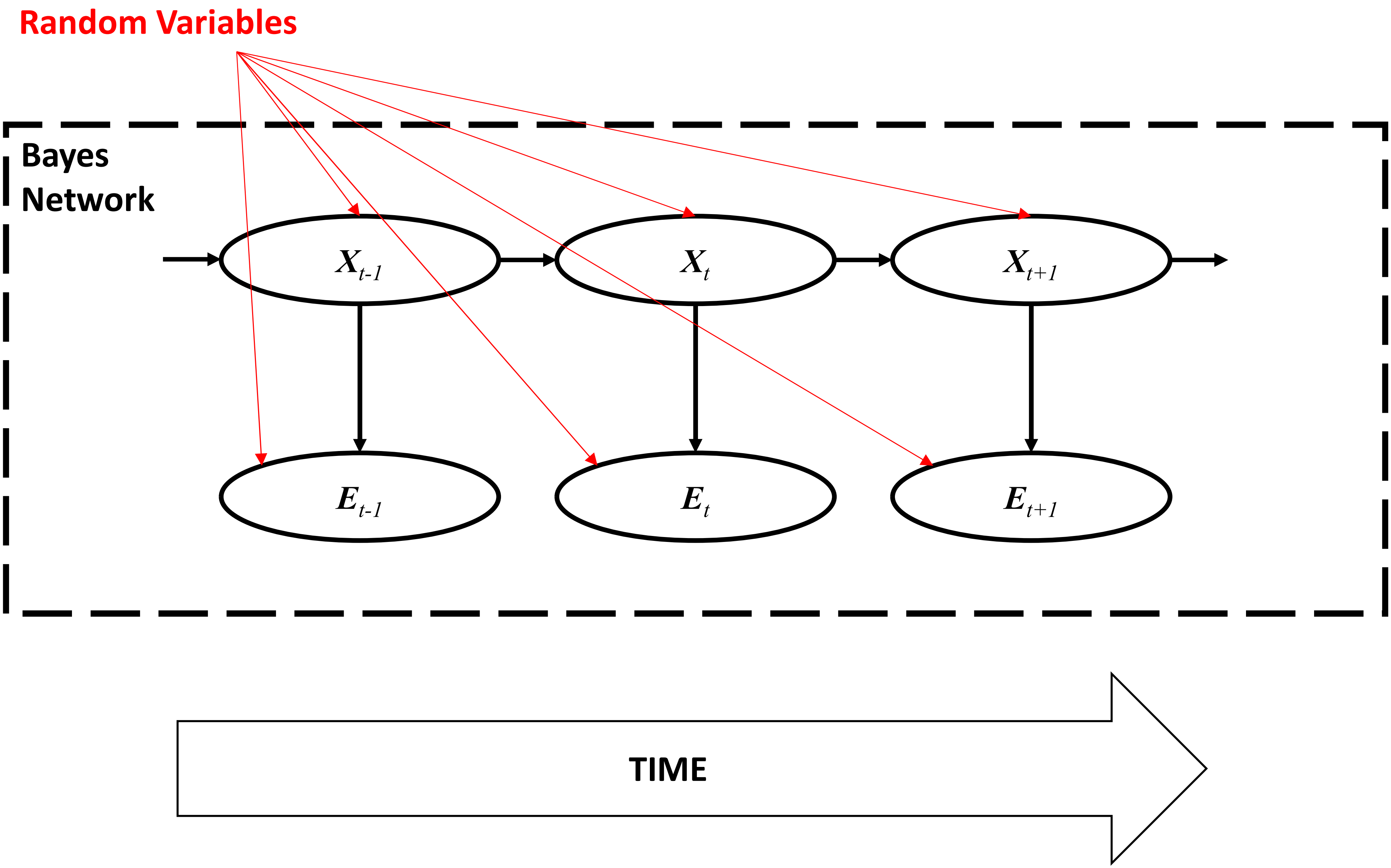
States and Observations



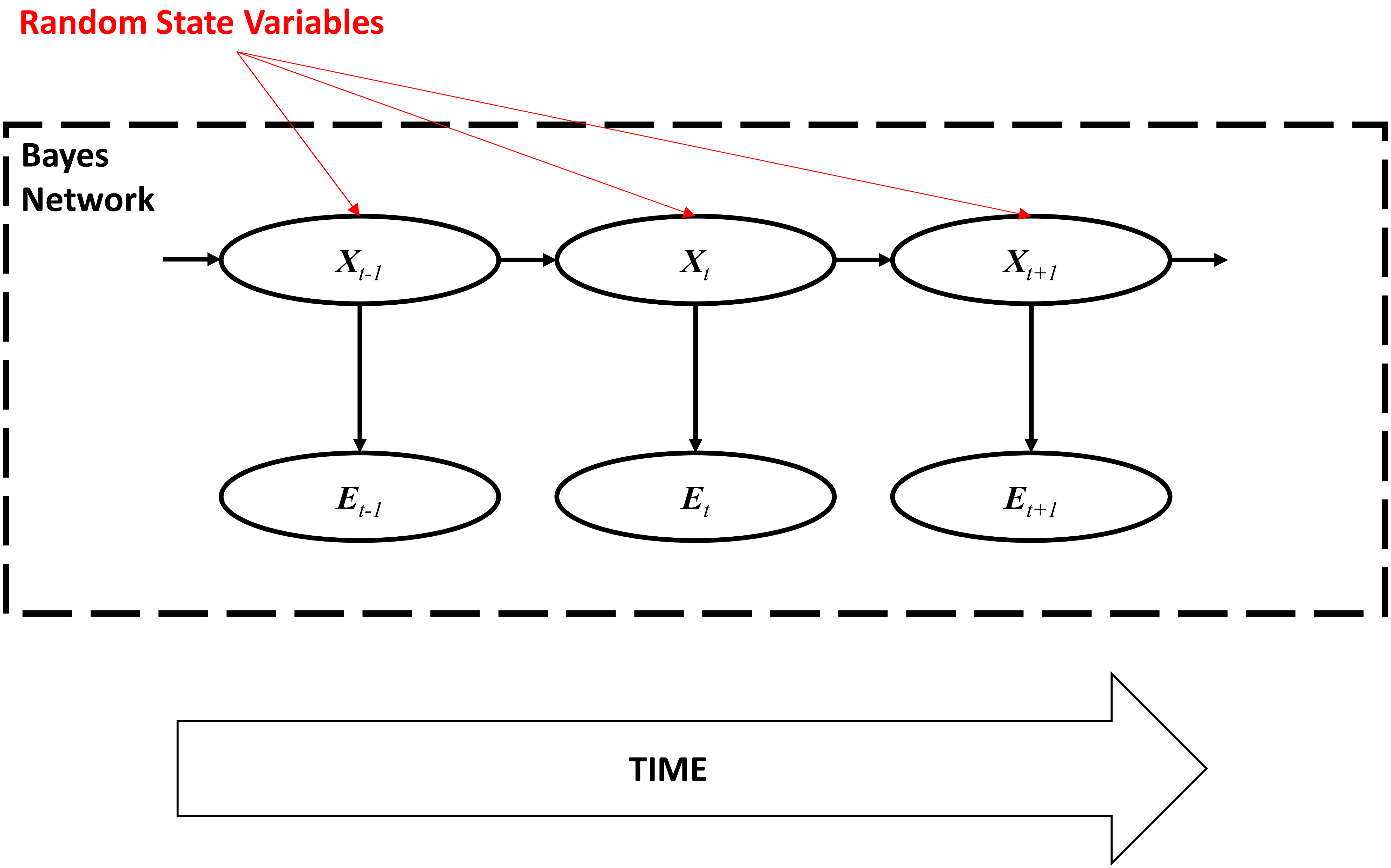
States and Observations



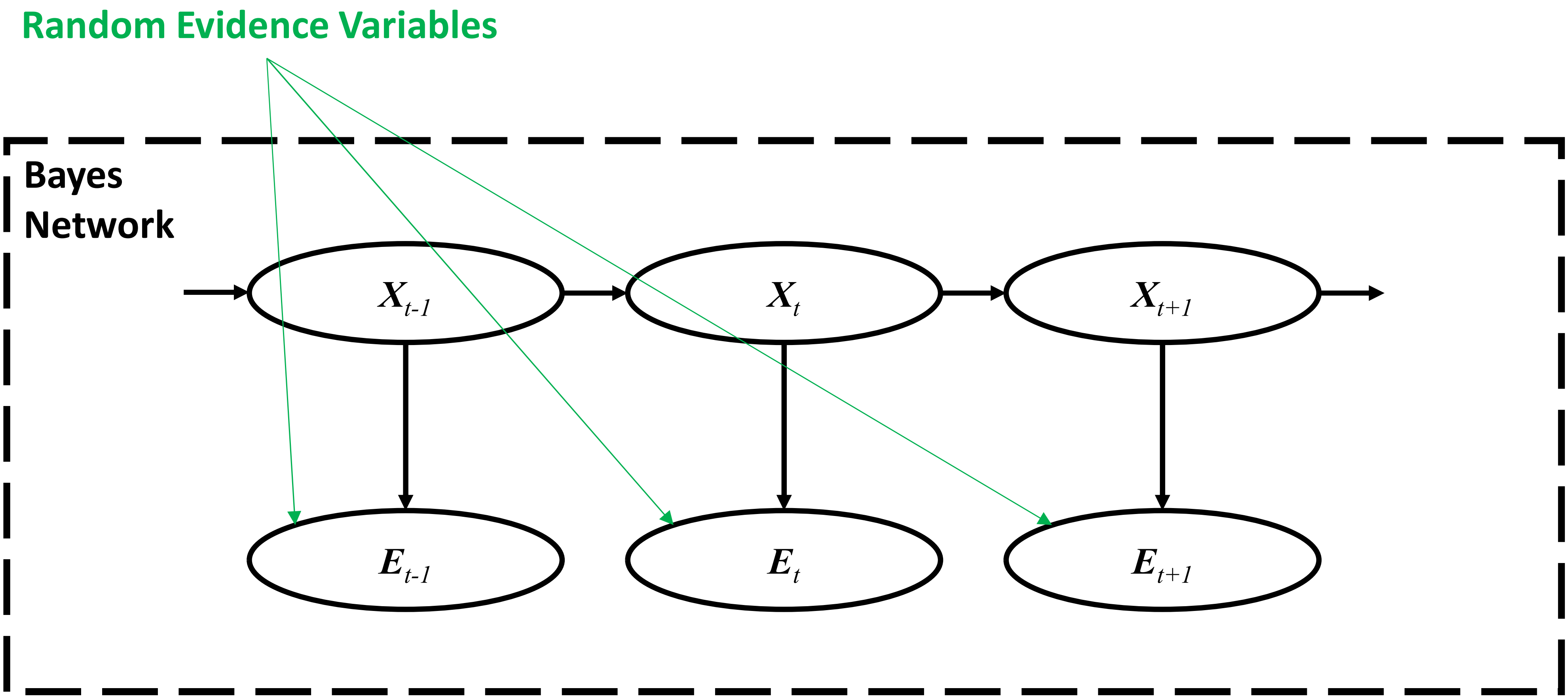
States and Observations

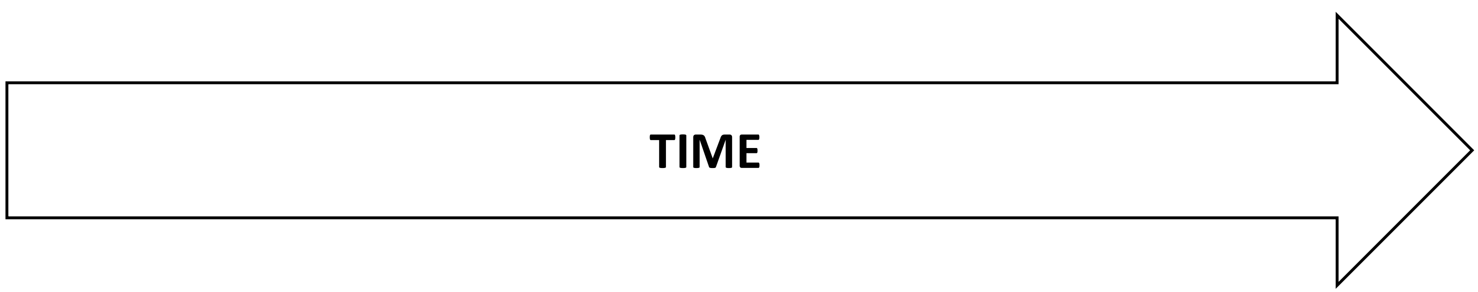


States and Observations

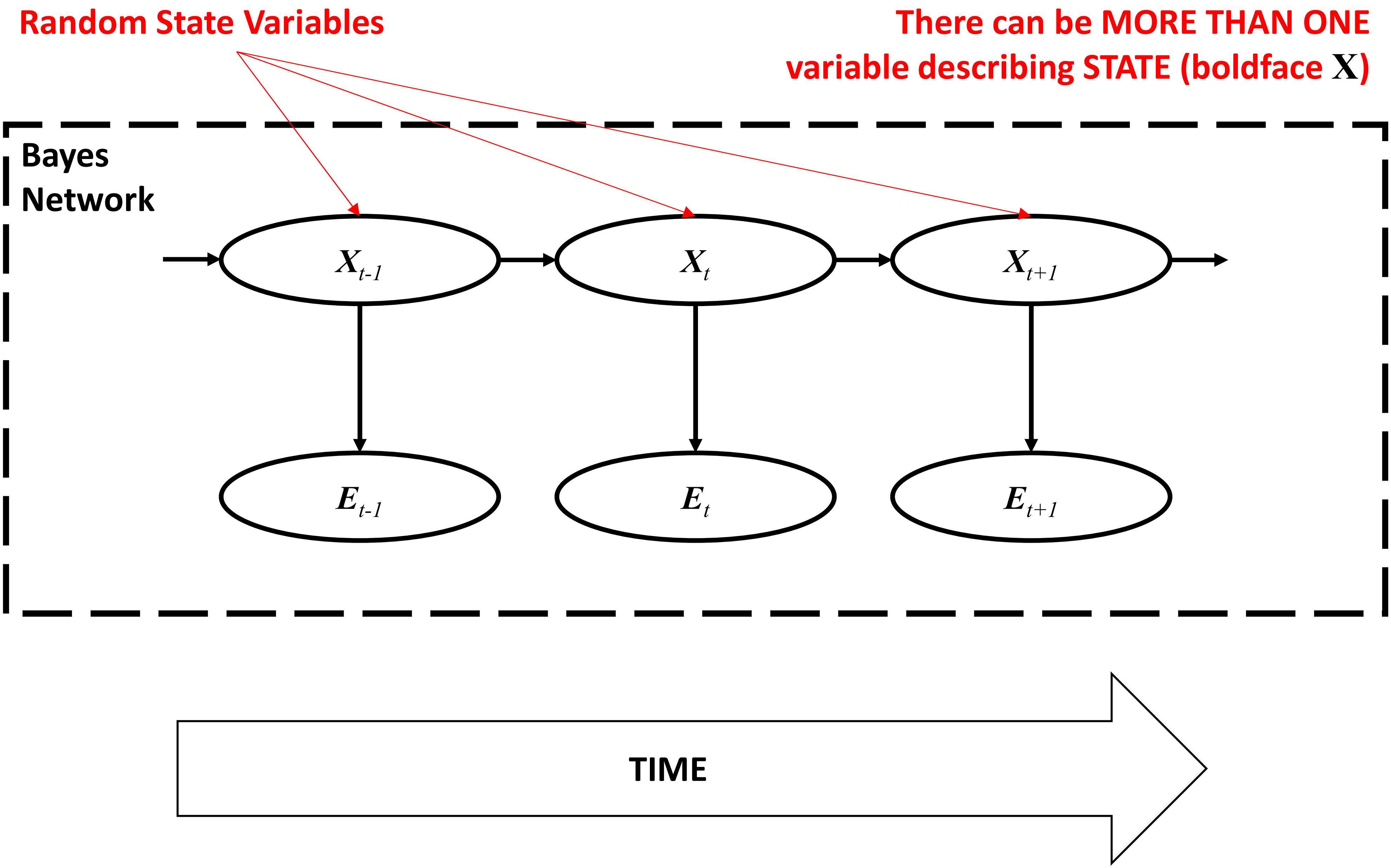


States and Observations

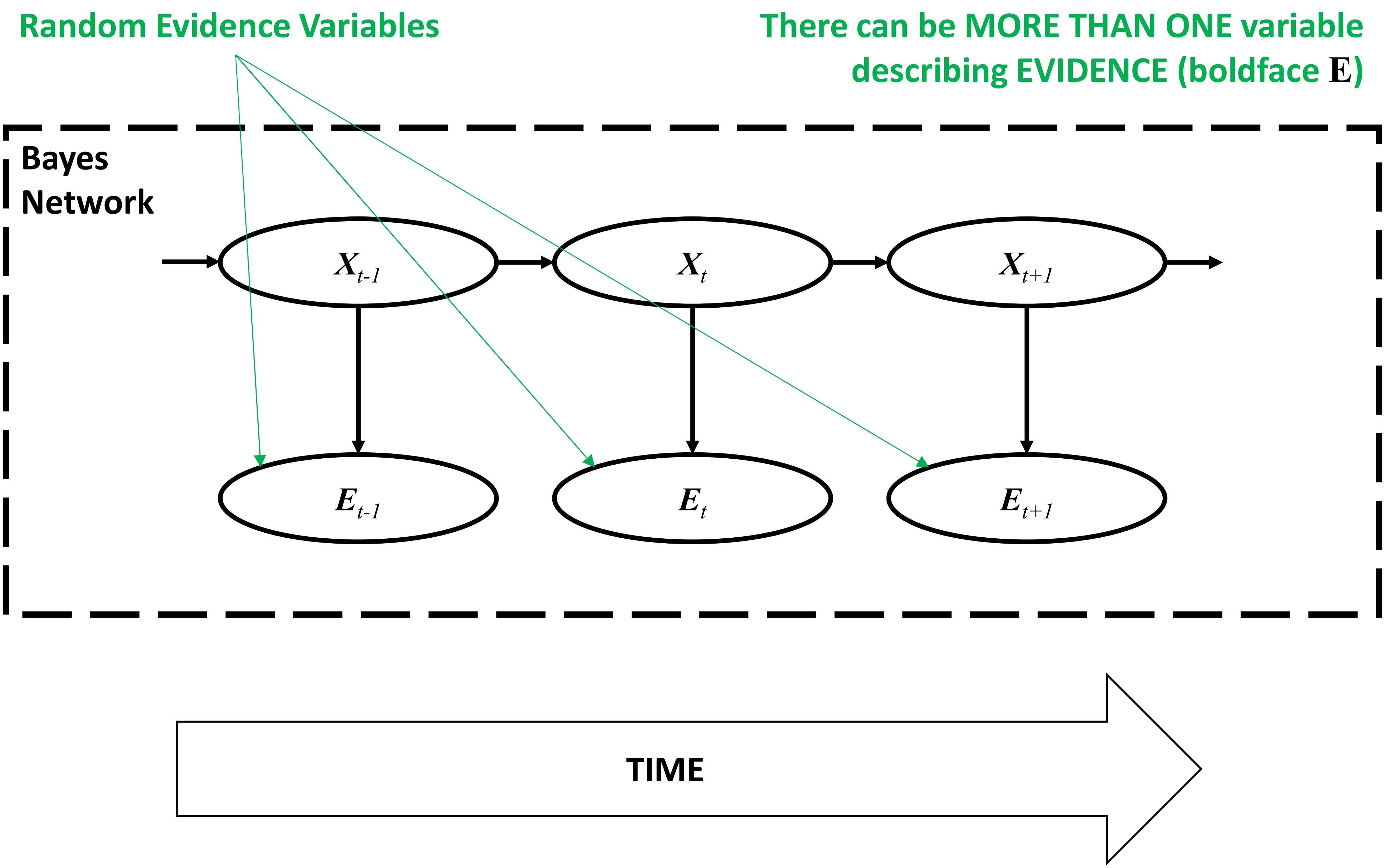




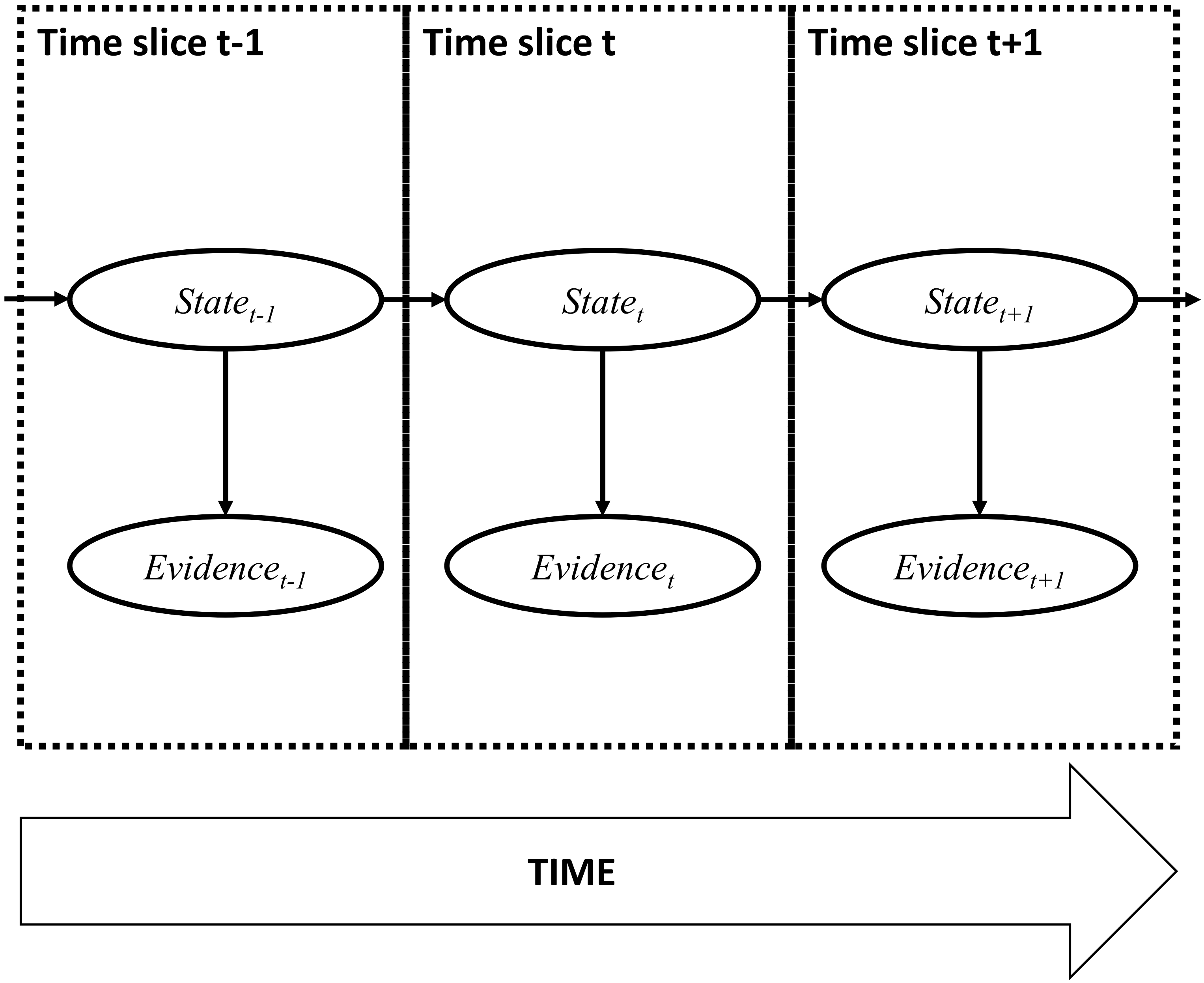
States and Observations



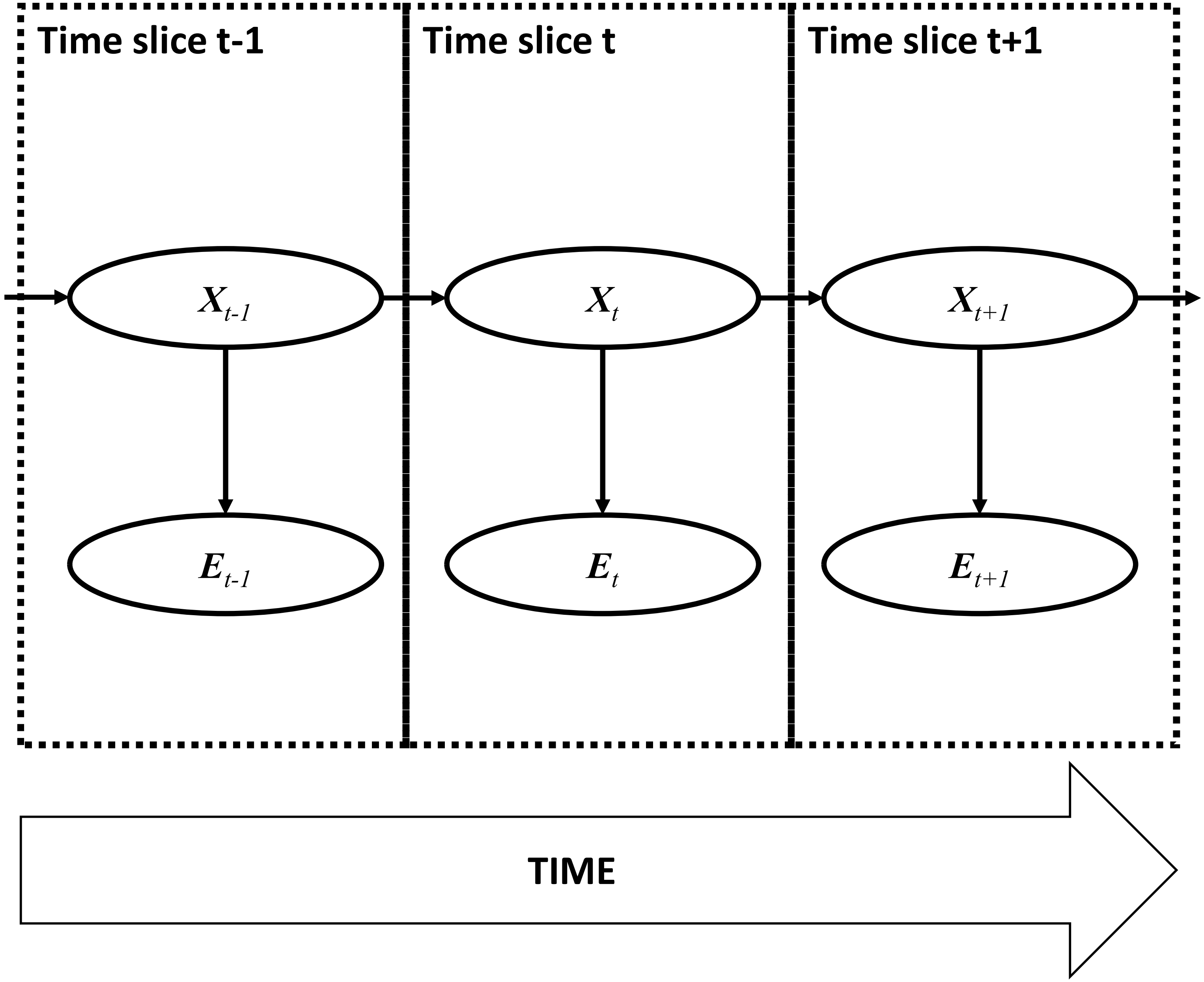
States and Observations



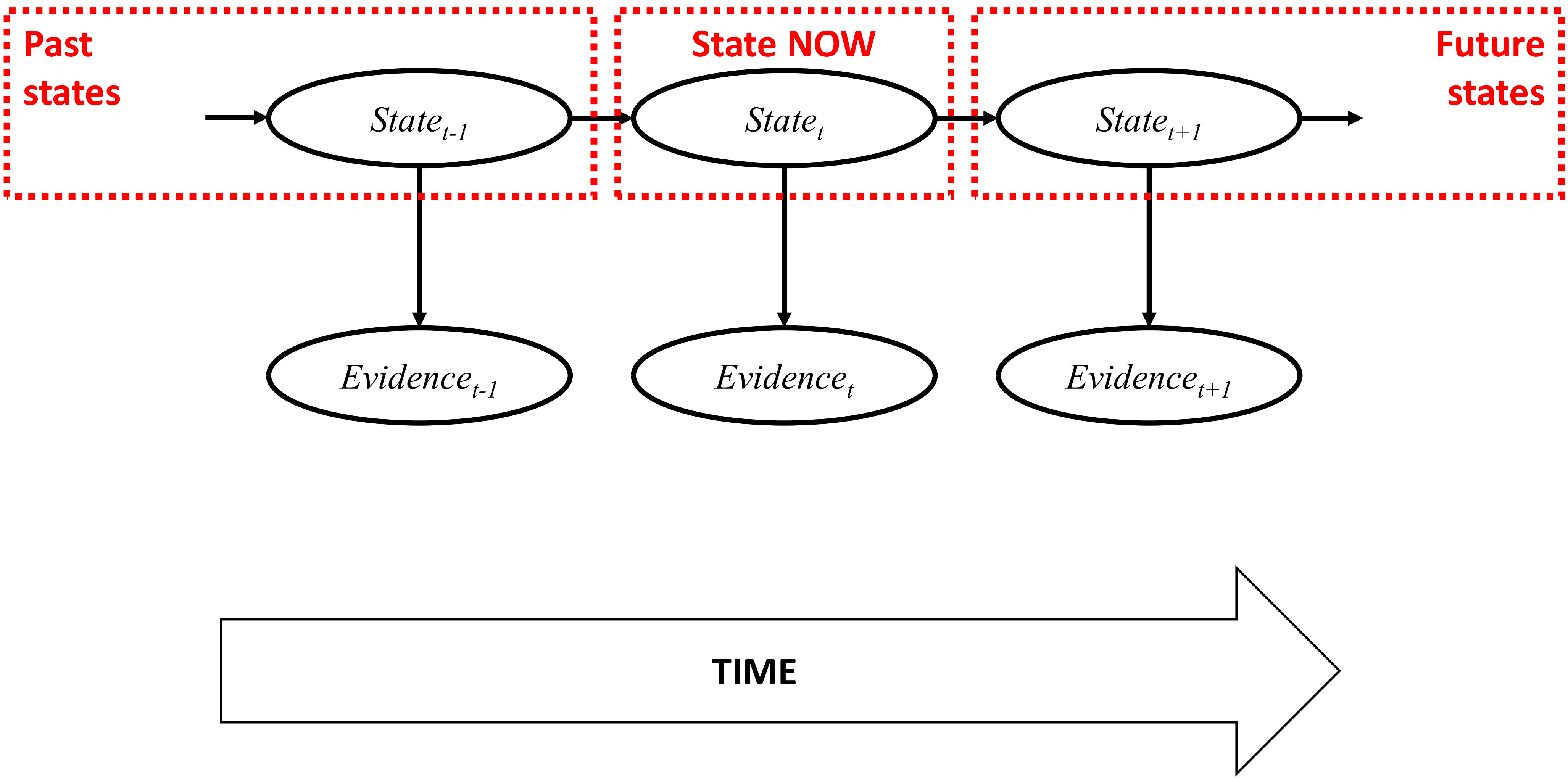
Discrete Time Model



Discrete Time Model

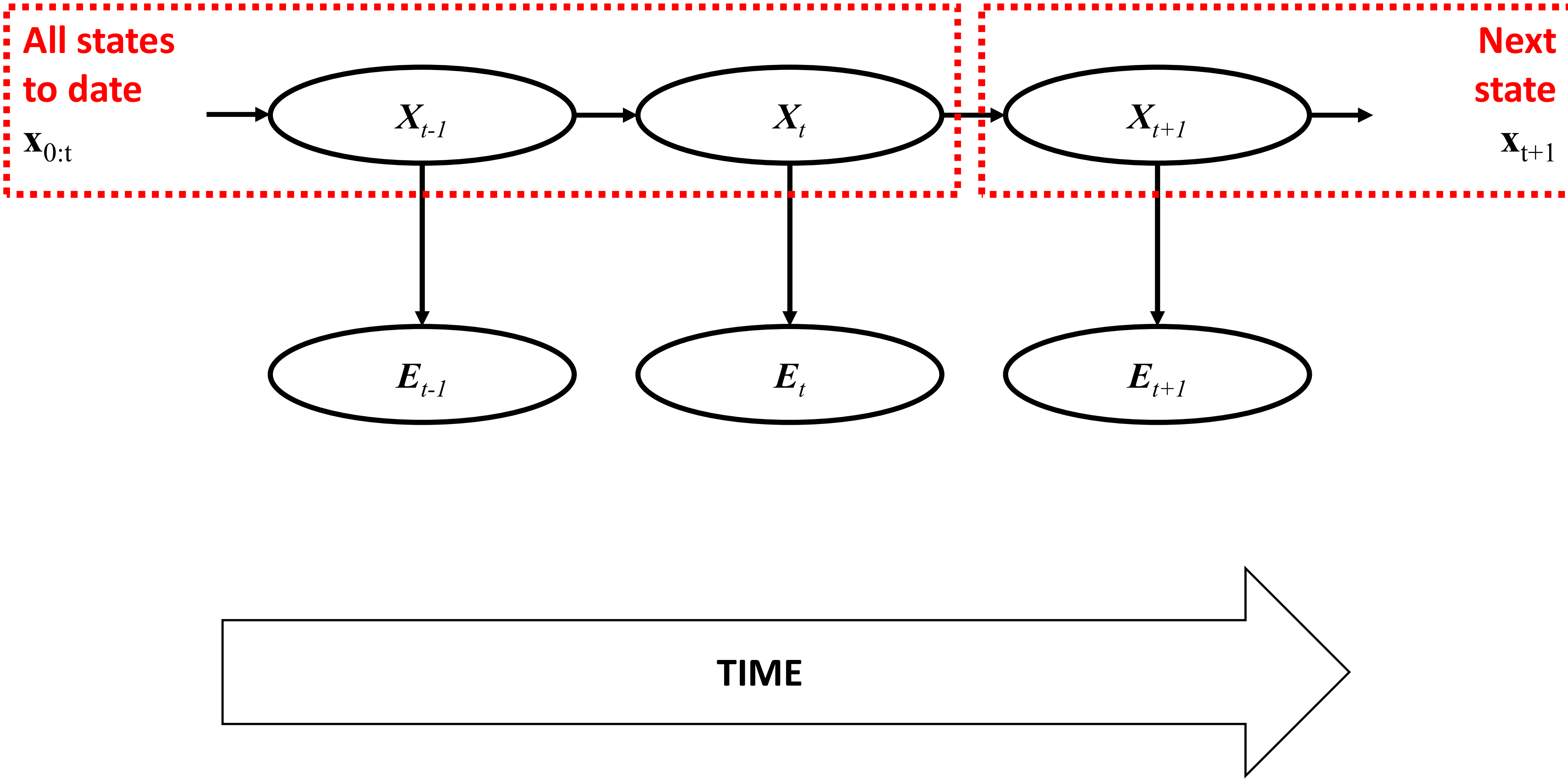


States and Observations

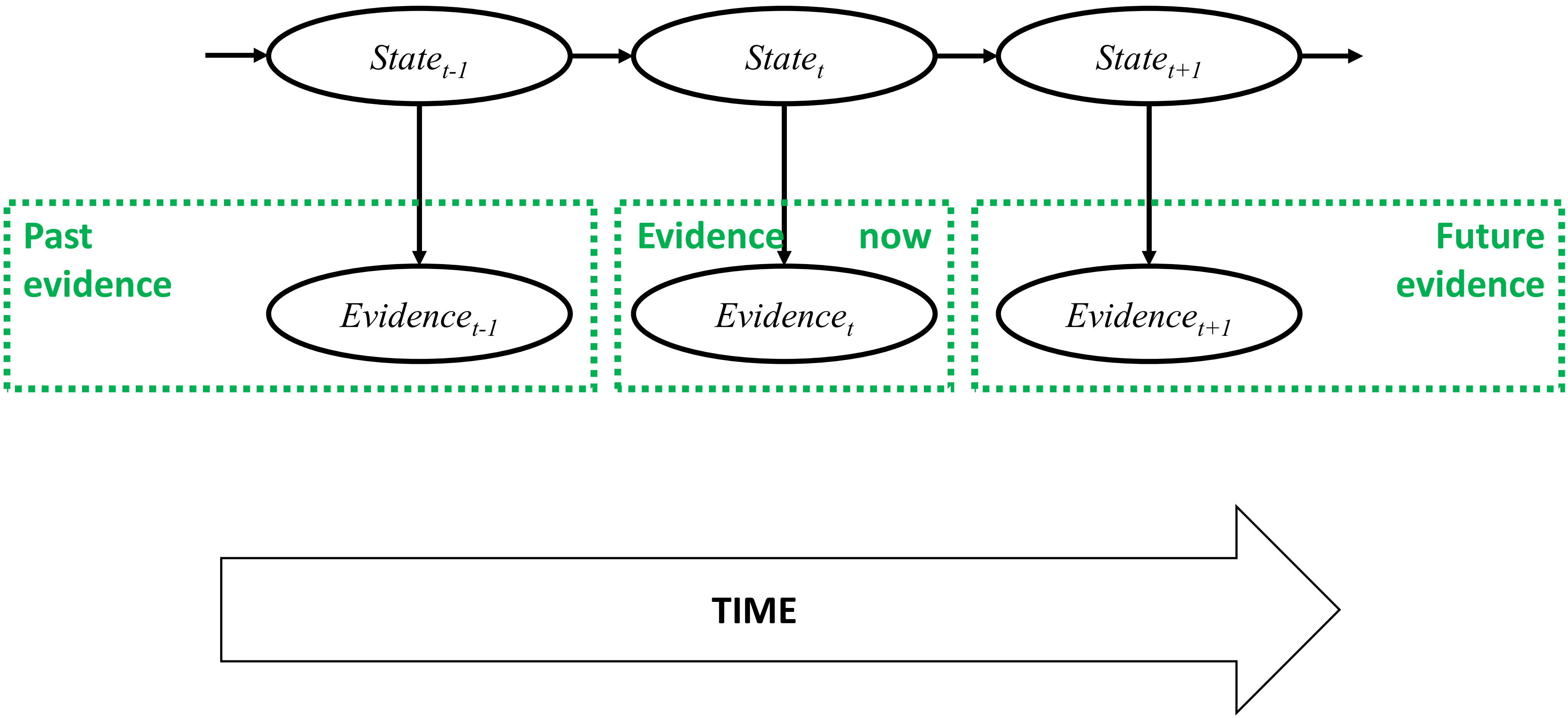


States and Observations

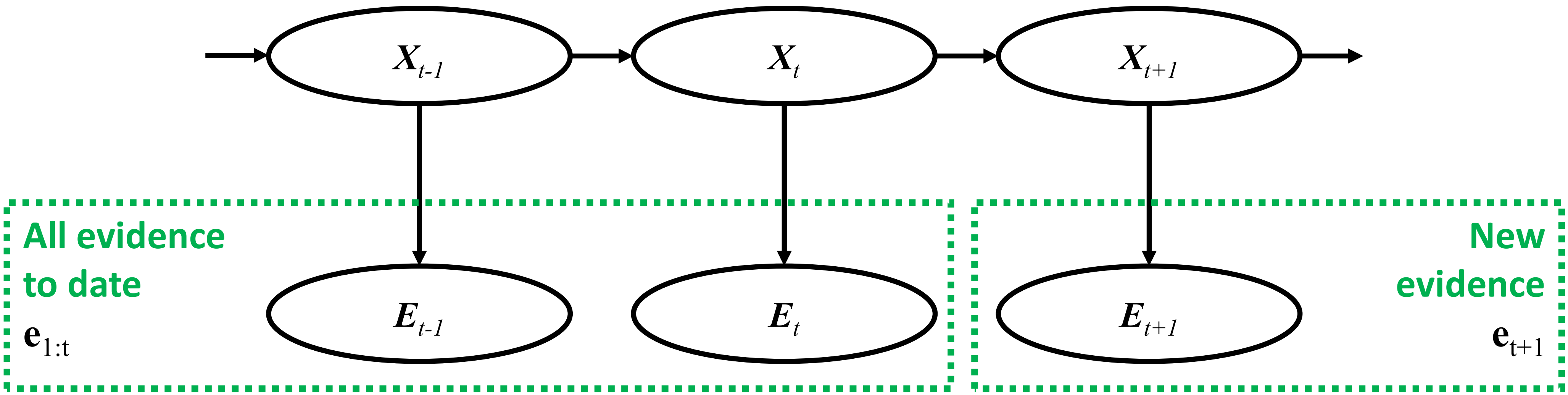
$$\mathbf{x}_{0:t} = \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t = \mathbf{x}_0 \wedge \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 \wedge \ldots \wedge \mathbf{x}_{t-1} \wedge \mathbf{x}_t$$



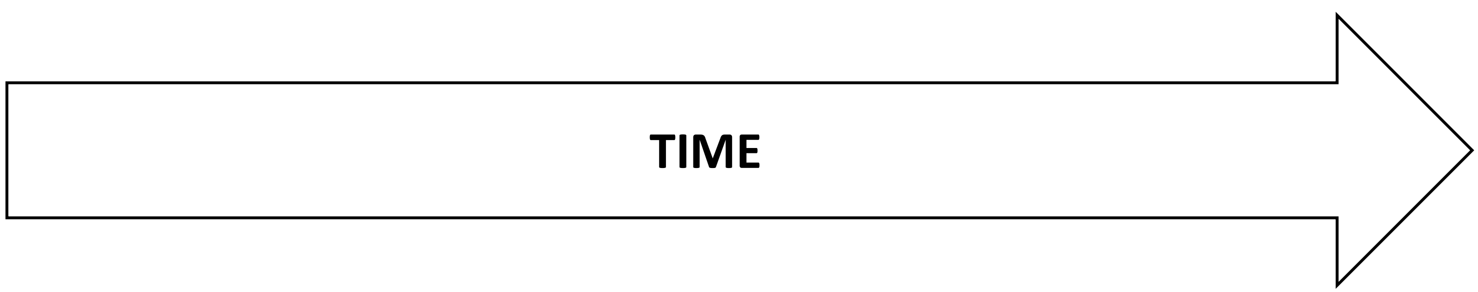
States and Observations



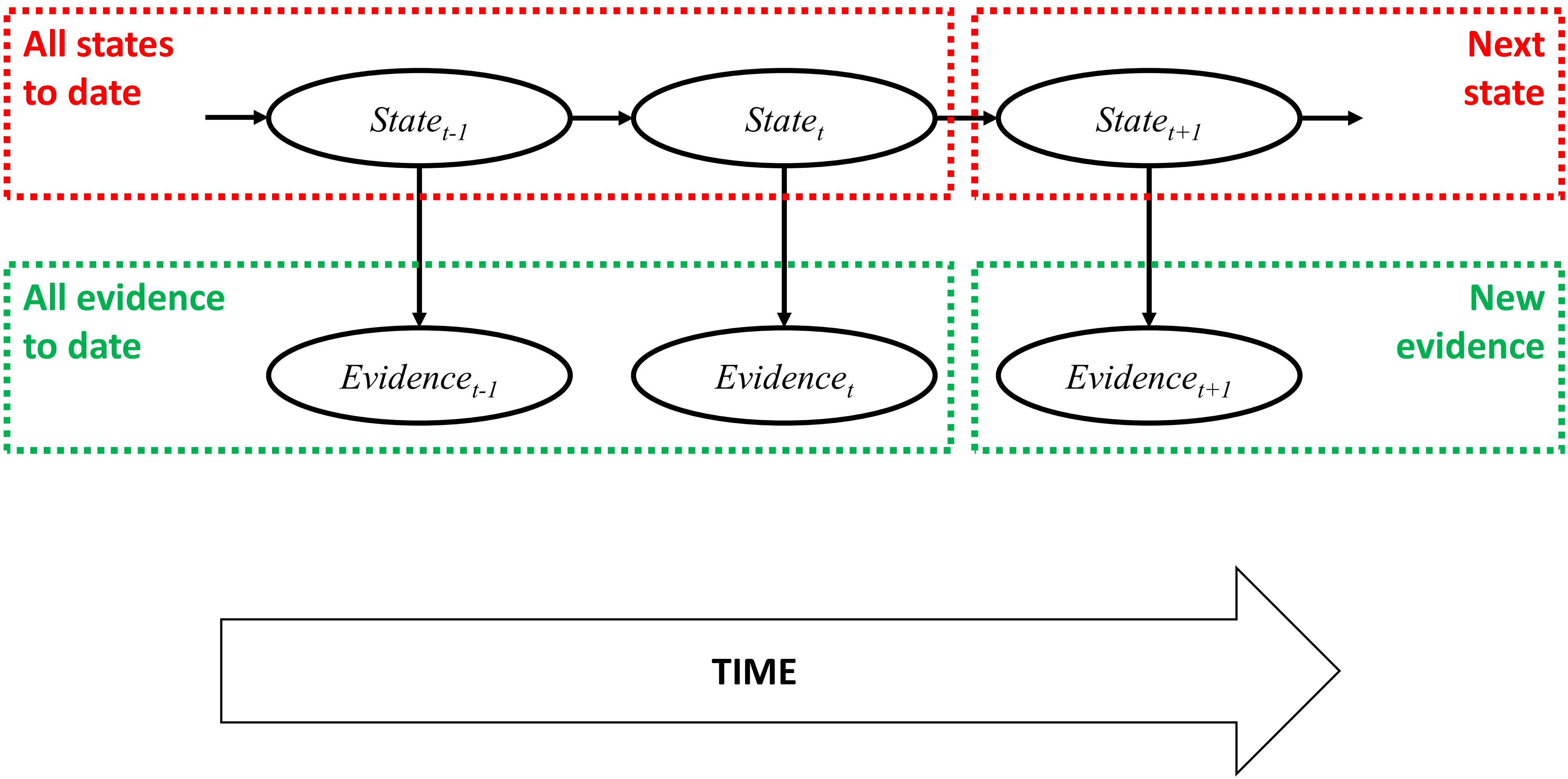
States and Observations



$\mathbf{e}_{1:t} = \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_{t-1}, \mathbf{e}_t = \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \ldots \wedge \mathbf{e}_{t-1} \wedge \mathbf{e}_t$

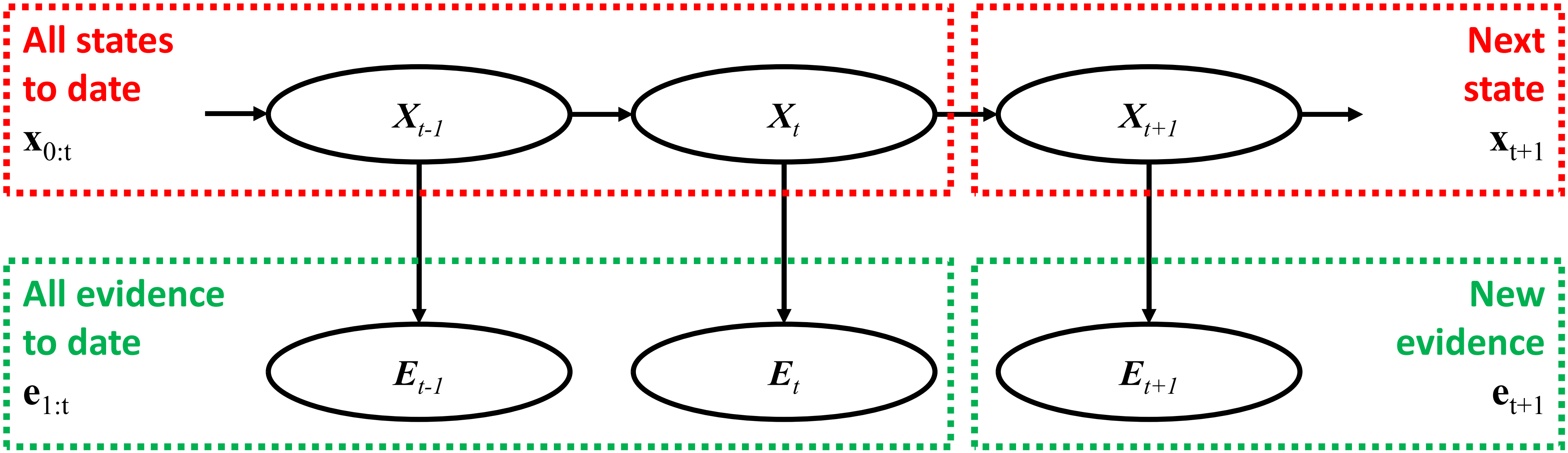


States and Observations

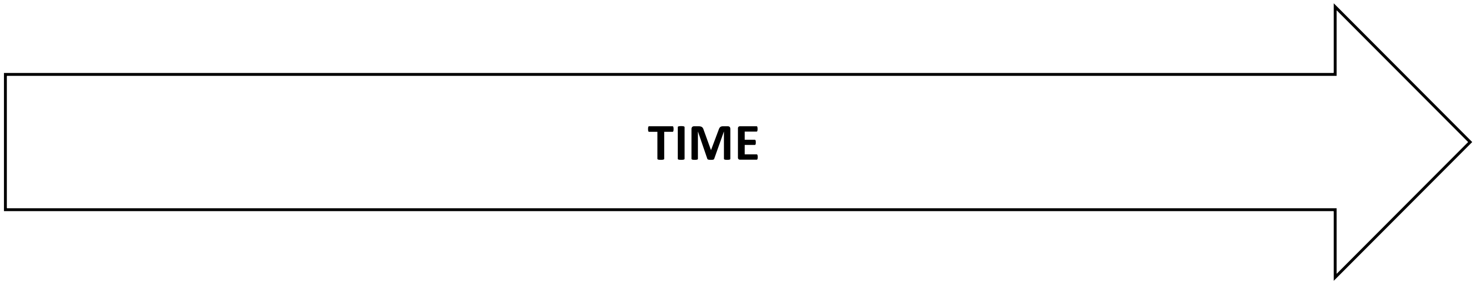


States and Observations

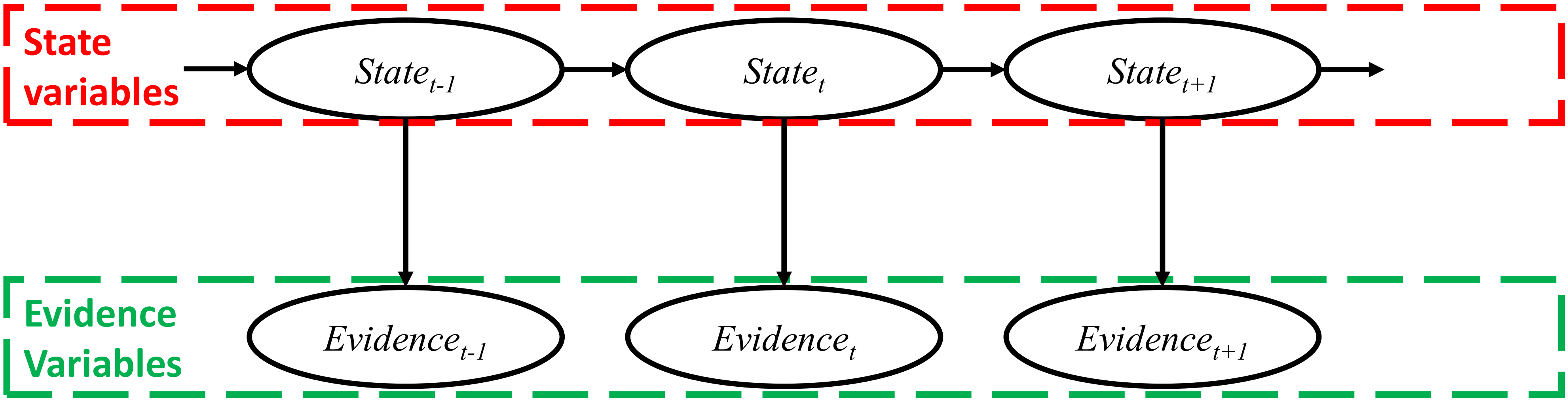
$\mathbf{x}_{0:t} = \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t = \mathbf{x}_0 \wedge \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 \wedge \ldots \wedge \mathbf{x}_{t-1} \wedge \mathbf{x}_t$



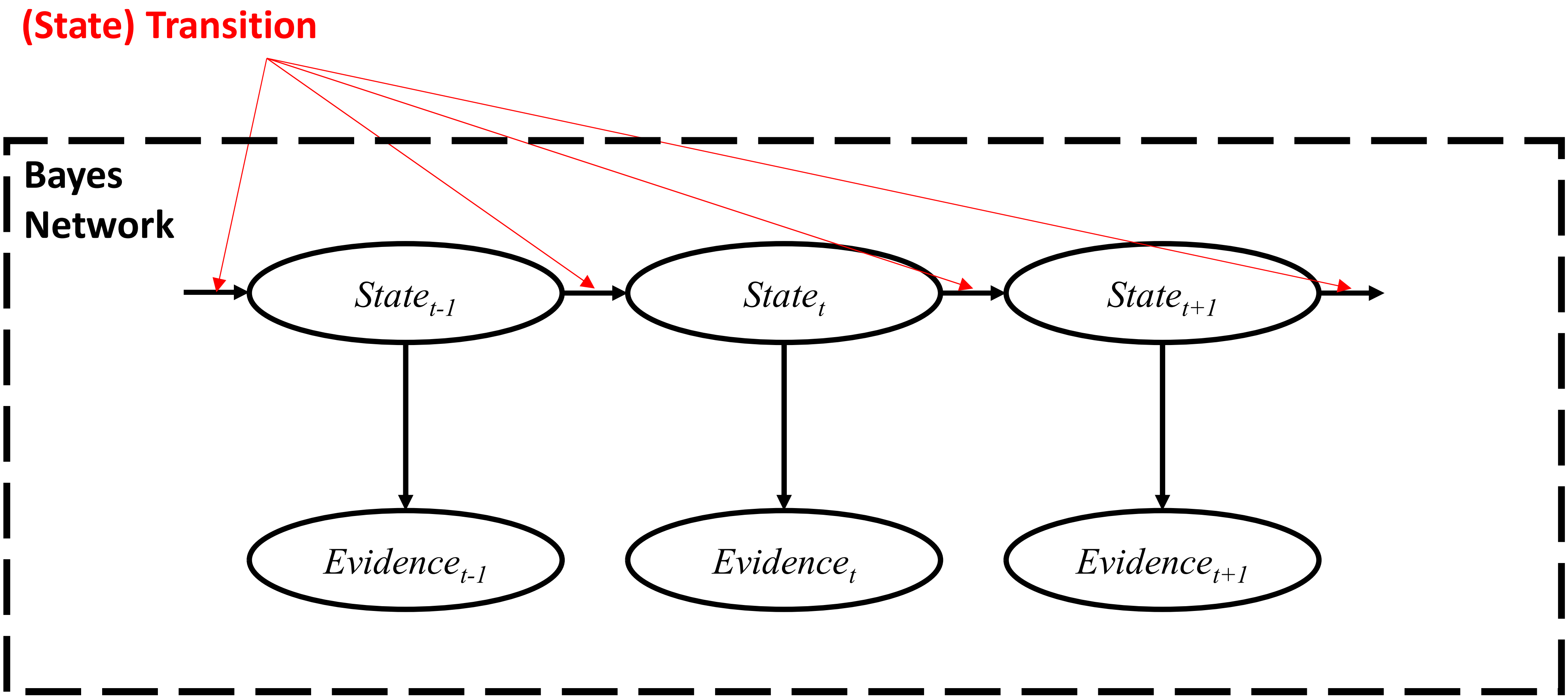
$\mathbf{e}_{1:t} = \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_{t-1}, \mathbf{e}_t = \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \ldots \wedge \mathbf{e}_{t-1} \wedge \mathbf{e}_t$



State (Hidden)/Evidence (Observable)

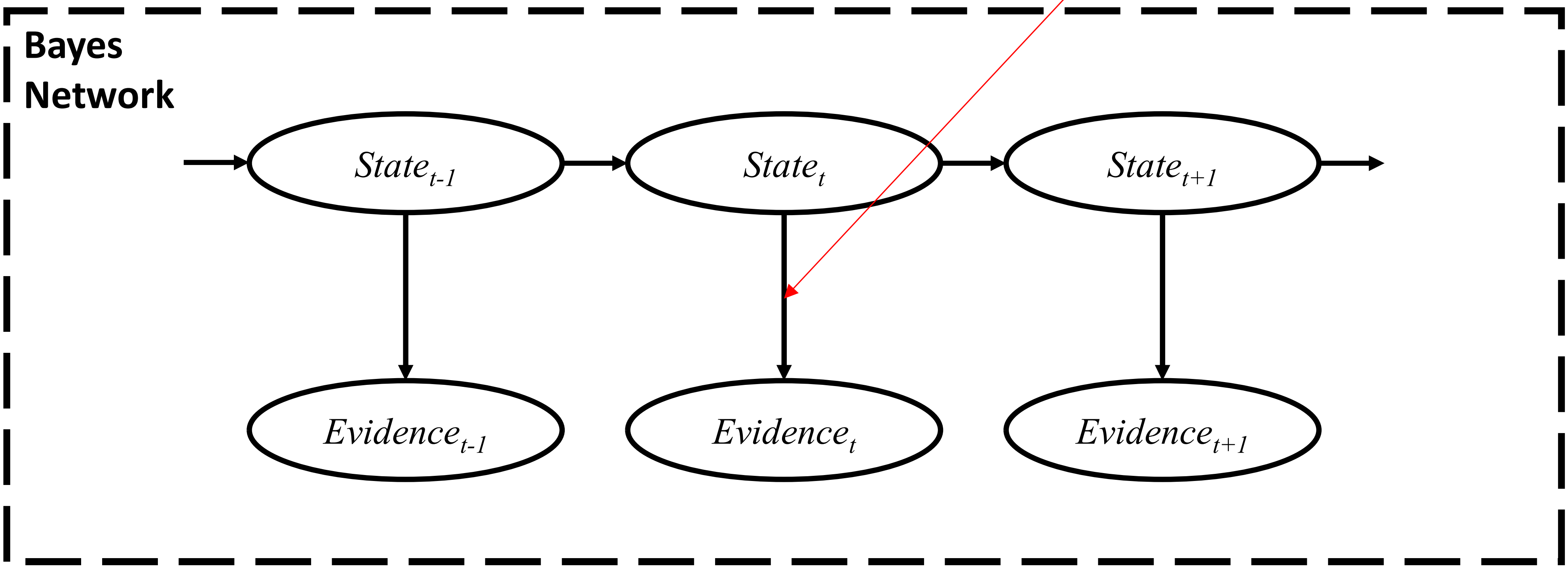


Transition and Sensor Models

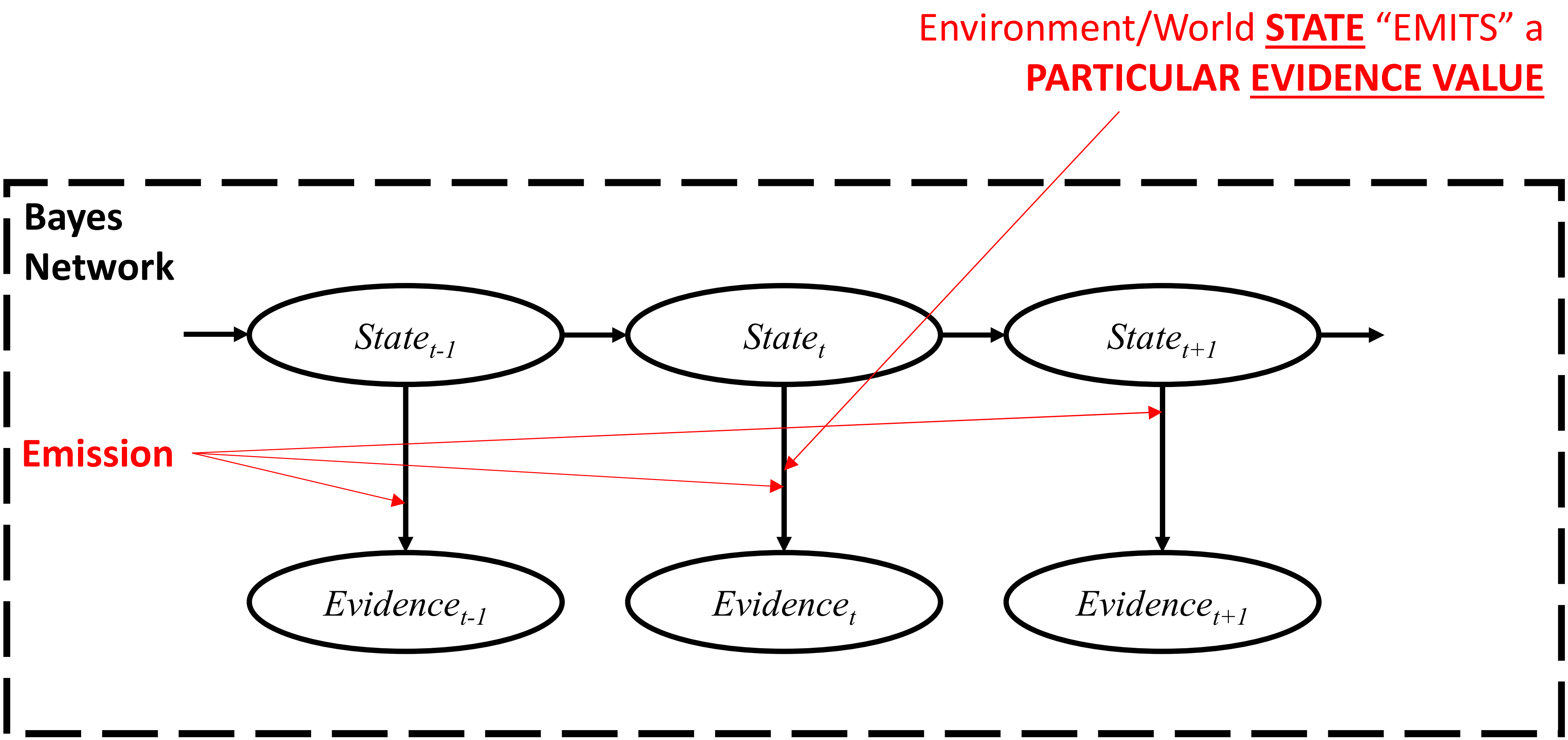


Transition and Sensor Models

Environment/World **STATE** "CAUSES" the sensors to observe **PARTICULAR EVIDENCE**



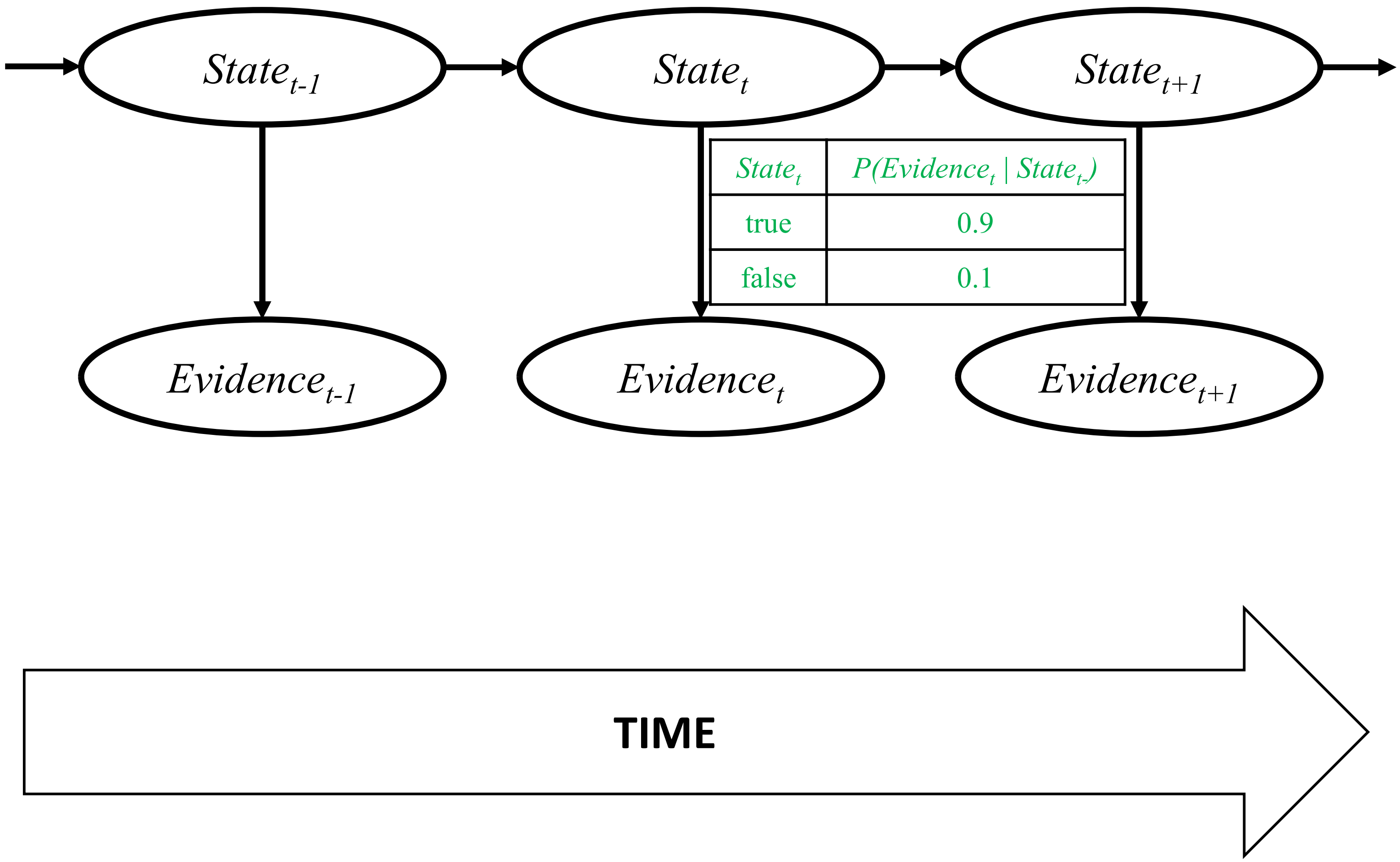
Transition and Sensor Models



Transition and Sensor Models

$State_{t-1}$	$P(State_{t-1} \mid State_{t-1})$
true	0.7
false	0.3

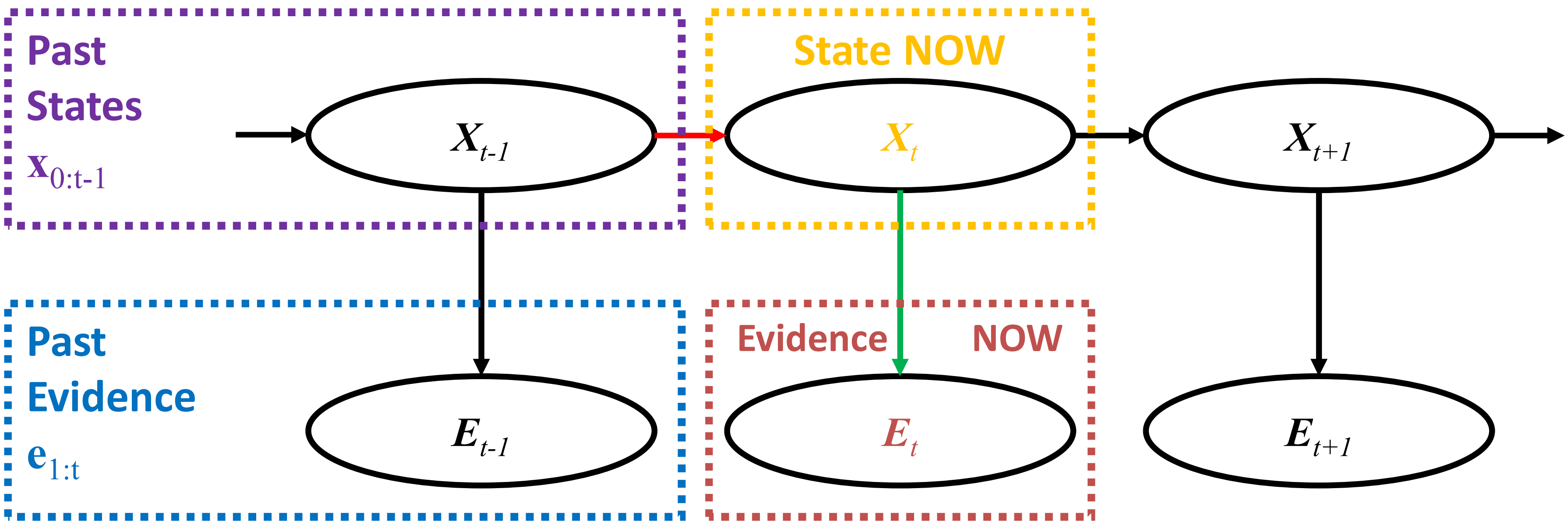
$State_t$	$P(Evidence_t \mid State_{t-})$
true	0.9
false	0.1



Transition and Sensor Model

The transition model specifies the probability distribution over the latest state variables, given the previous values:

$$P(X_t \mid X_{0:t-1})$$



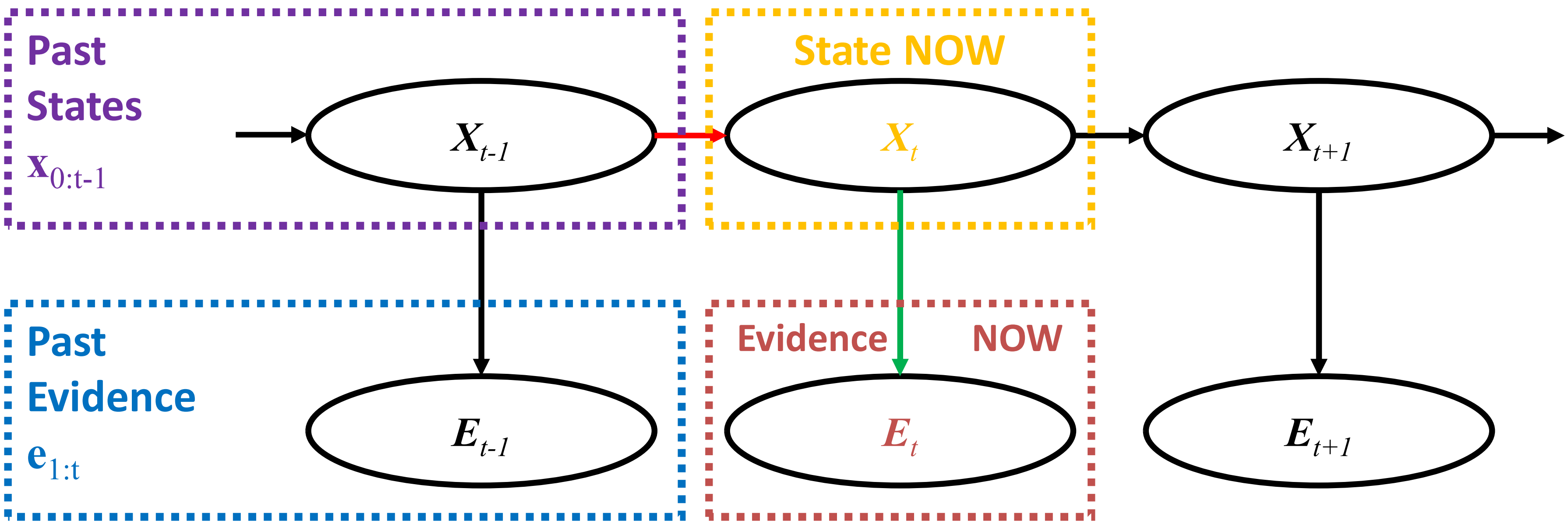
The sensor model: current evidence variables could depend on previous (past) evidence values as well as previous (past) and current state values

$$P(E_t \mid X_{0:t}, E_{1:t-1}) = P(E_t \mid X_{0:t-1}, X_t, E_{1:t-1})$$

Transition and Sensor Model

The transition model specifies the probability distribution over the latest state variables, given the previous values:

$$P(X_t \mid X_{0:t-1})$$



The sensor model: current evidence variables could depend on previous (past) evidence values as well as previous (past) and current state values

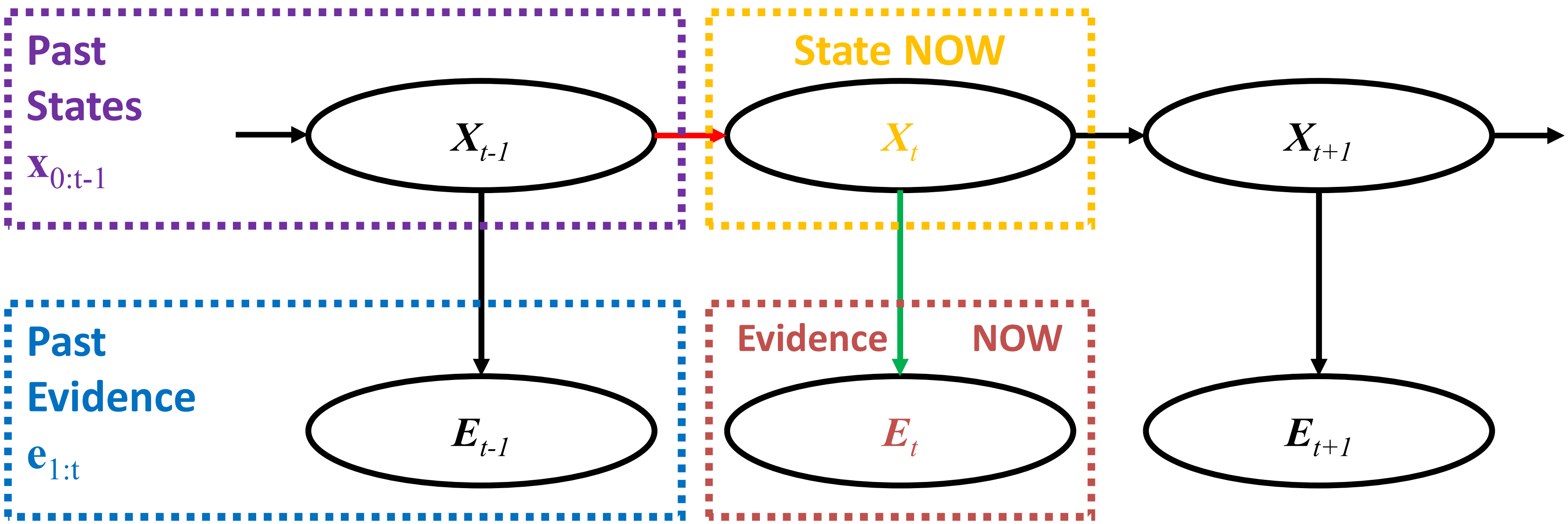
$$P(E_t \mid X_{0:t}, E_{1:t-1}) = P(E_t \mid X_{0:t-1}, X_t, E_{1:t-1})$$

What is the problem here?

Transition and Sensor Model

The transition model specifies the probability distribution over the latest state variables, given the previous values:

$$P(X_t \mid X_{0:t-1})$$



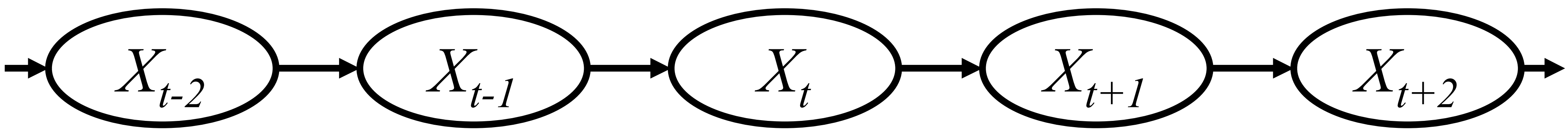
The sensor model: current evidence variables could depend on previous (past) evidence values as well as previous (past) and current state values

$$P(E_t \mid X_{0:t}, E_{1:t-1}) = P(E_t \mid X_{0:t-1}, X_t, E_{1:t-1})$$

Unbounded sets as t grows!

Markov Assumption

Markov Process (Chain) is a random process that generates a sequence of states:



Bayesian Network?? Anyone? Indeed!

$$P(X_{t+1} \mid X_t, X_{t-2}, X_{t-2}) = P(X_t \mid Parents(X_t)) = P(X_{t+1} \mid X_t)$$

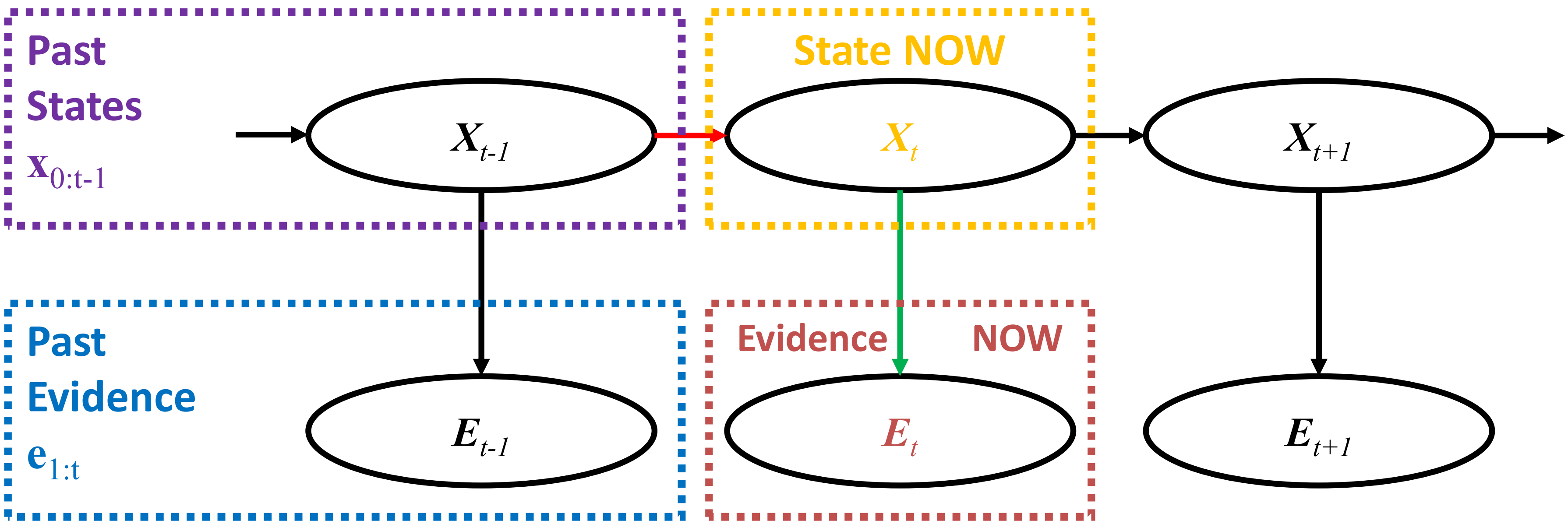
$$P(X_{t+1} \mid X_t, X_{t-2}, X_{t-2}) = P(X_{t+1} \mid X_t)$$

(First-order) Markov ASSUMPTION

T / S Models /w Markov Assumption

The transition model specifies the probability distribution over the latest state variables, given the previous values:

$$P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$$



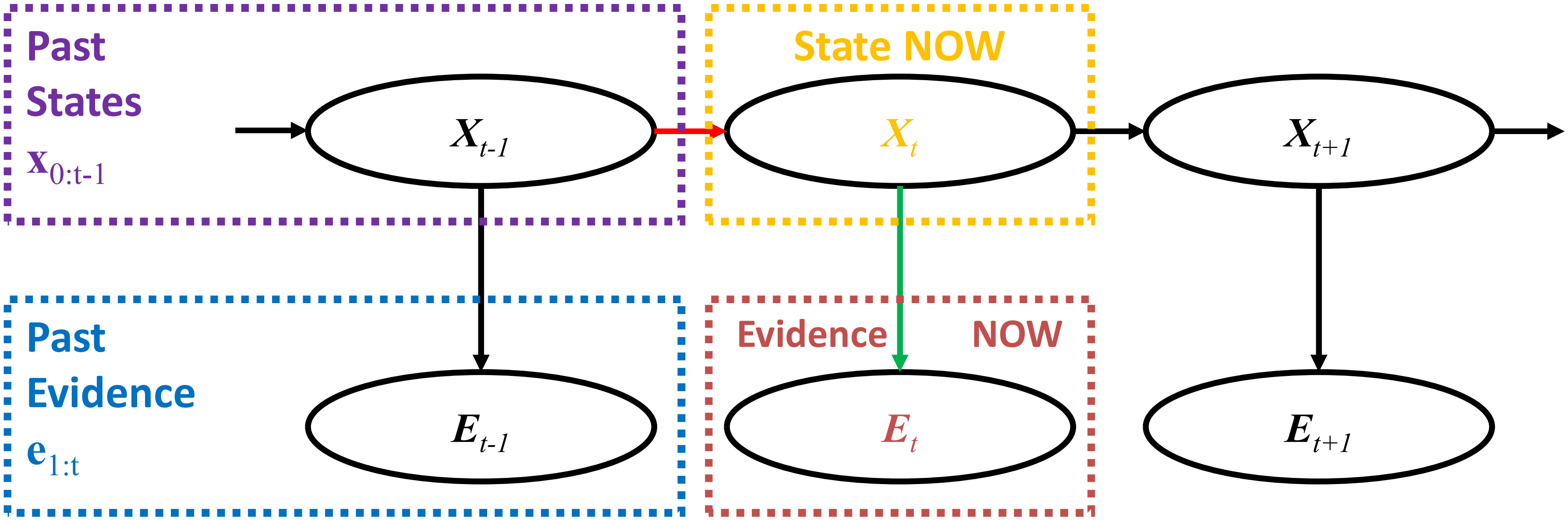
The sensor model: current evidence variables could depend on previous (past) evidence values as well as previous (past) and current state values

$$P(E_t \mid X_{0:t}, E_{1:t-1}) = P(E_t \mid X_t)$$

T / S Models /w Markov Assumption

The transition model specifies the probability distribution over the latest state variables, given the previous values:

$$P(X_t \mid X_{t-1})$$



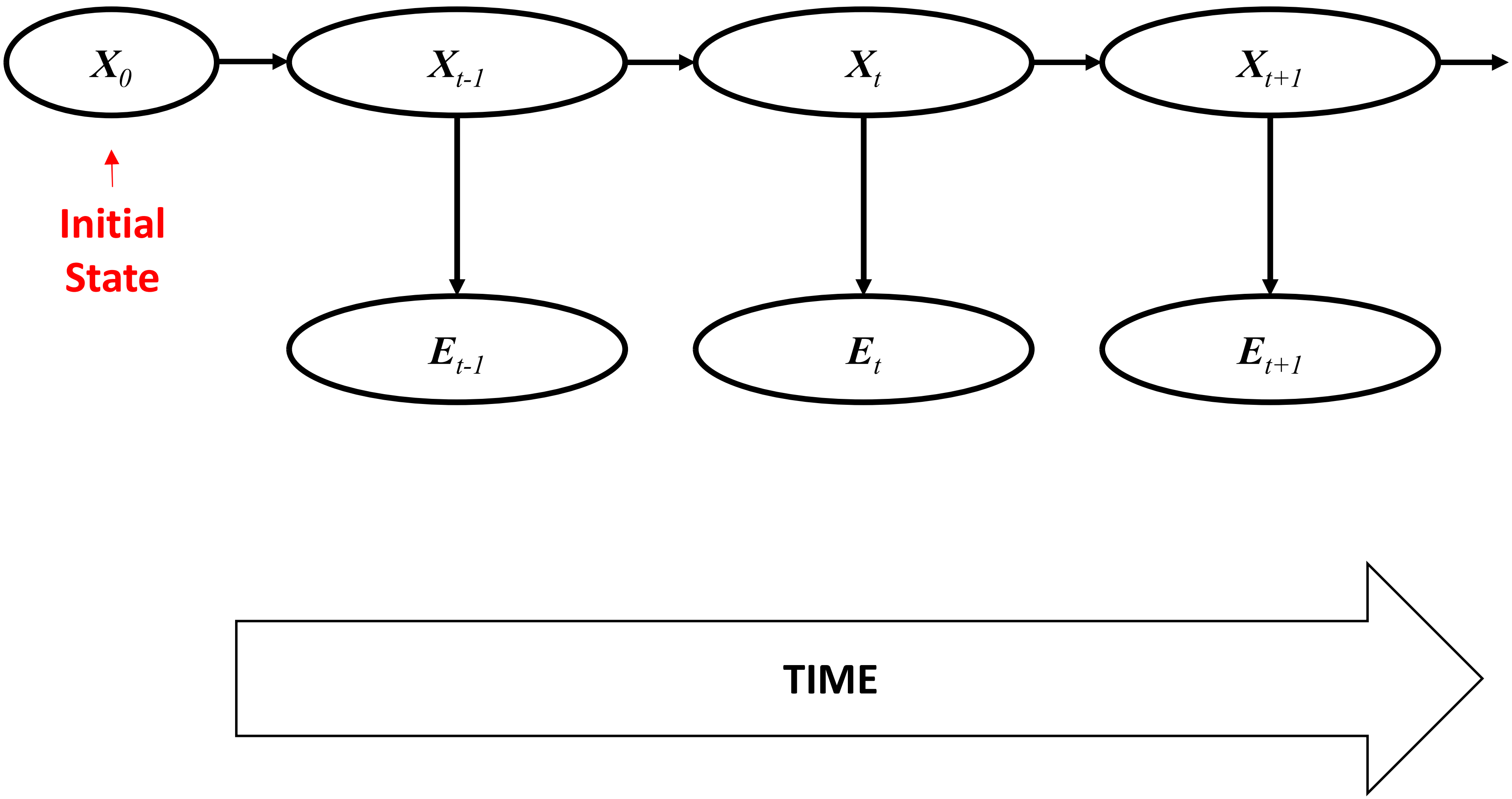
The sensor model: current evidence variables could depend on previous (past) evidence values as well as previous (past) and current state values

$$P(E_t \mid X_t)$$

Complete Joint Distribution

The complete (including initial state distribution | for any t) joint probability distribution for a sequence of transitions and emissions:

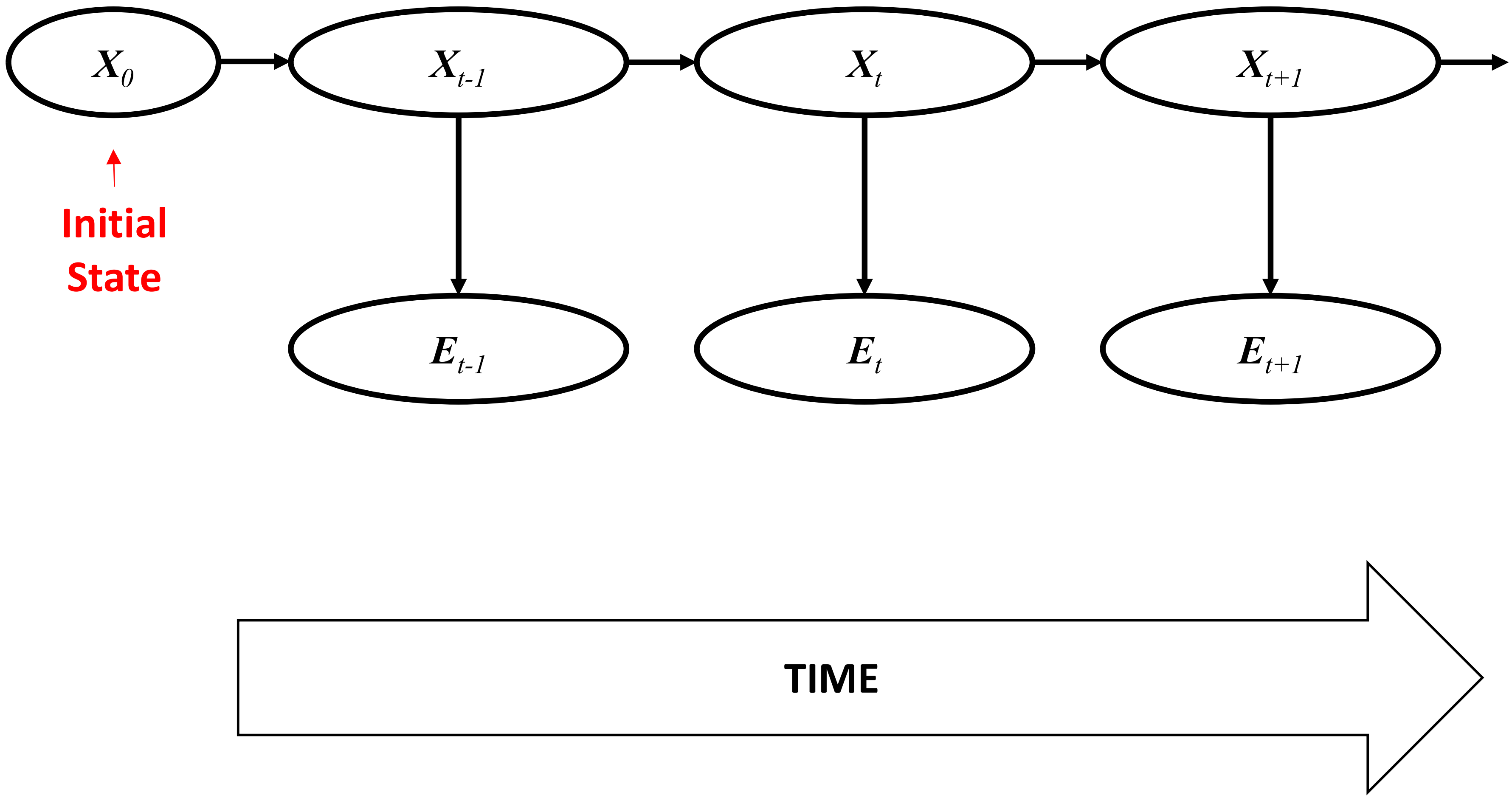
$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \mathbf{P}(\mathbf{X}_0) \prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1}) \, \mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$$



Complete Joint Distribution

The complete (including initial state distribution | for any t) joint probability distribution for a sequence of transitions and emissions:

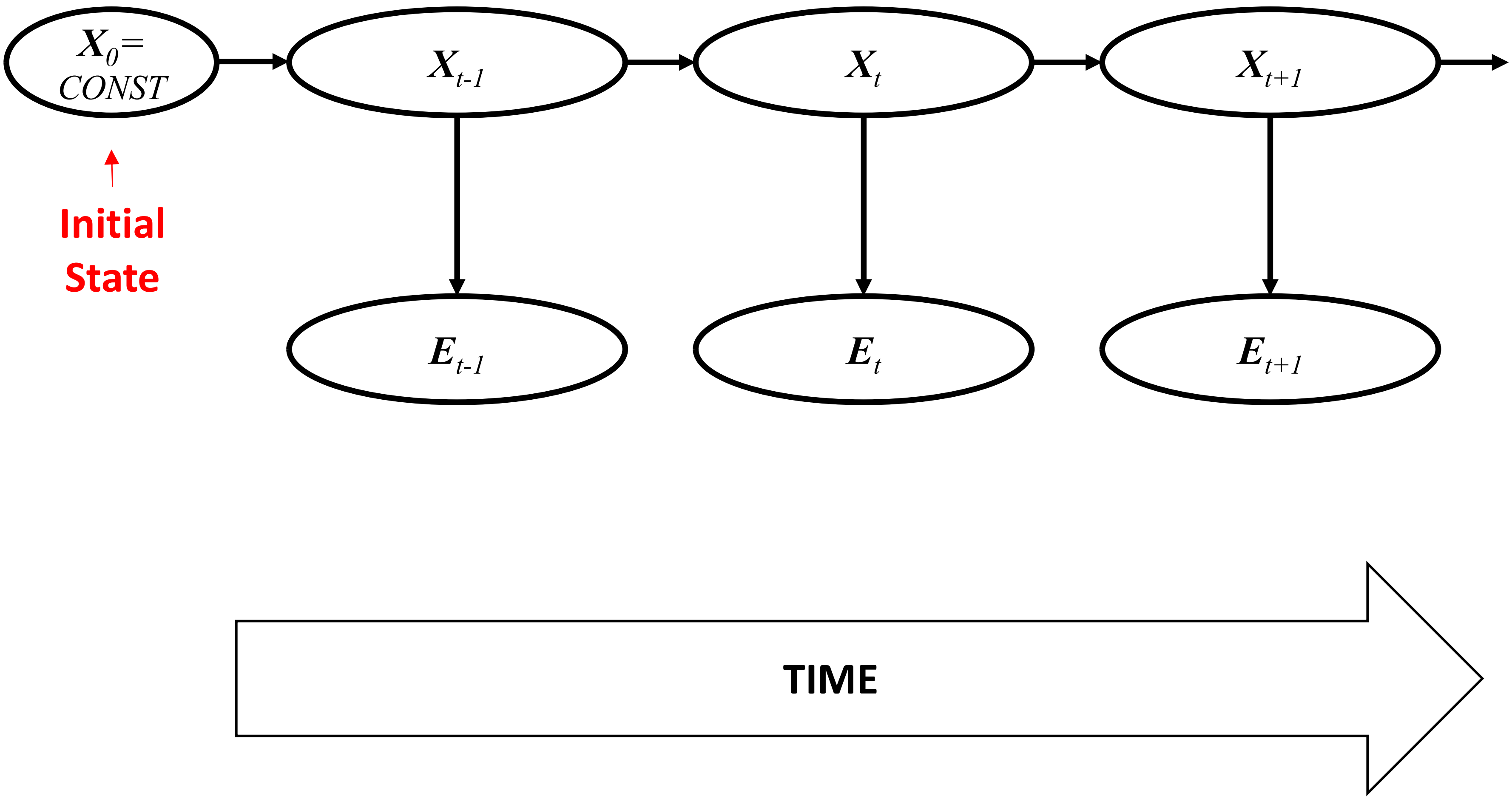
$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \underbrace{\mathbf{P}(\mathbf{X}_0)}_{\text{Initial}} \prod_{i=1}^{t} \underbrace{\mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})}_{\text{Transition}} \underbrace{\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)}_{\text{Emission}}$$



Complete Joint Distribution

The complete (including initial state distribution | for any *t*) joint probability distribution for a sequence of transitions and emissions:

$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \prod_{i=1}^{t} \underbrace{\mathbf{P}(\mathbf{X}_i \,|\, \mathbf{X}_{i-1})}_{\text{Transition}} \underbrace{\mathbf{P}(\mathbf{E}_i \,|\, \mathbf{X}_i)}_{\text{Emission}}$$



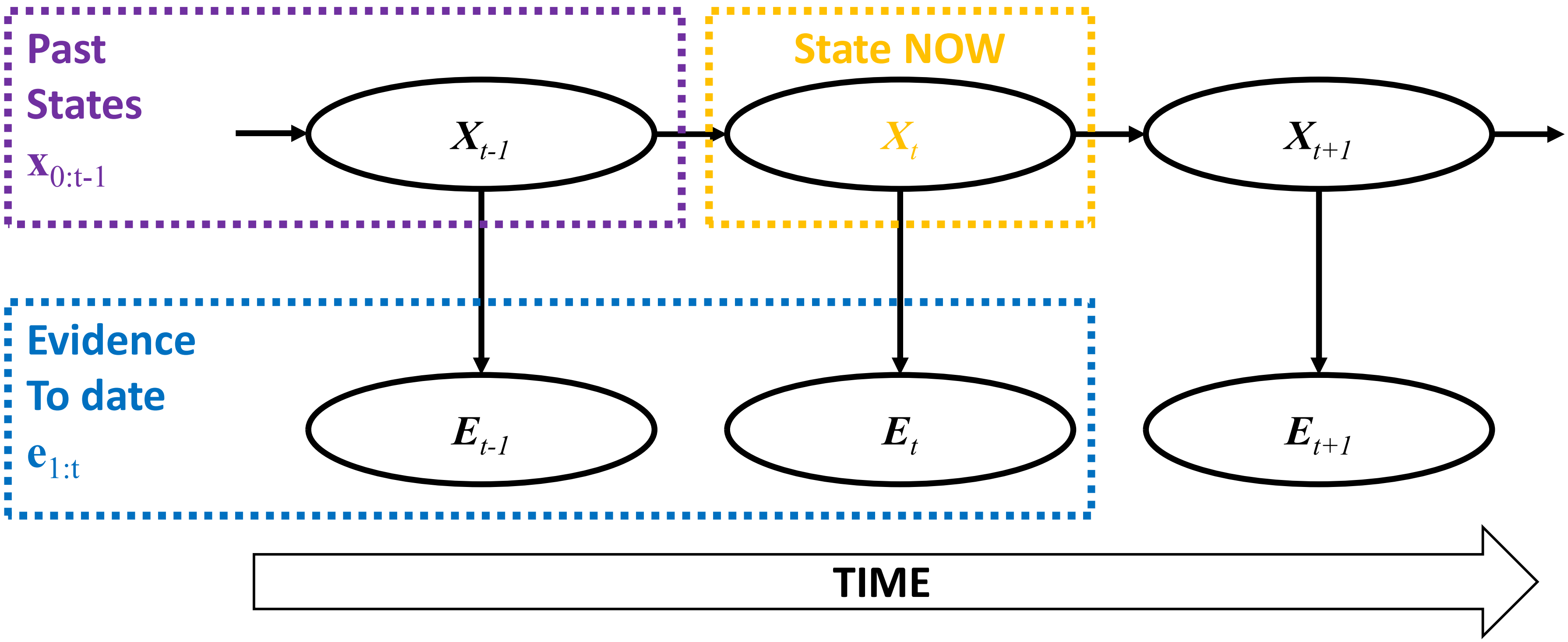
Inference in Temporal Models

- **Filtering / State Estimation**
  - **Current Belief State given Evidence/Percept/Observations so far**
- **Prediction**
  - **Future Belief State given Evidence/Percept /Observations so far**
- **Smoothing**
  - **Past Belief State given Evidence/Percept/Observations so far**
- **Most likely explanation:**
  - **Use sequence of observations to find sequence of states that generated them**
- **Learning:**
  - **Learn the transition and sensor models based on observations ("emissions")**

Inference: Filtering

This is the task of computing the belief state — the posterior distribution over the most recent state — given all evidence to date. Filtering is also called STATE ESTIMATION.

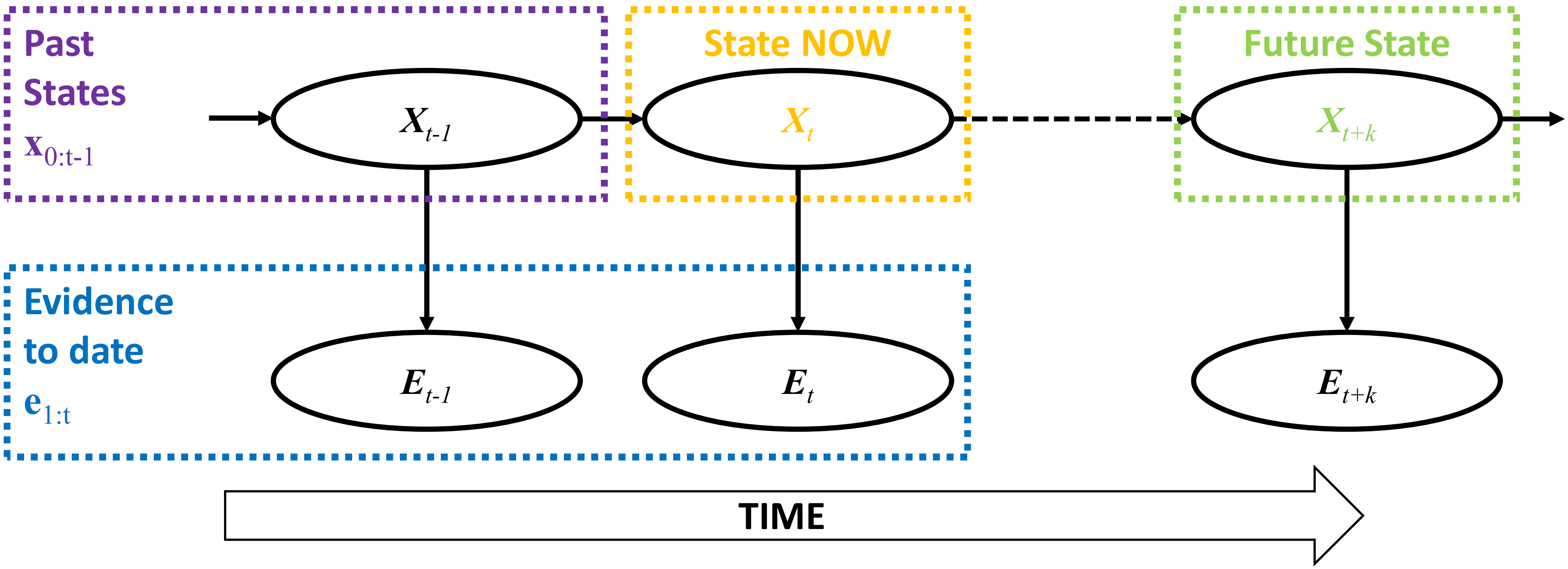
$$P(X_t \mid \mathbf{e}_{1:t})$$



Inference: Prediction

This is he task of computing the posterior distribution over the future state (time $t+k$, for some $k > 0$), given all evidence to date. Useful for evaluating possible courses of action.

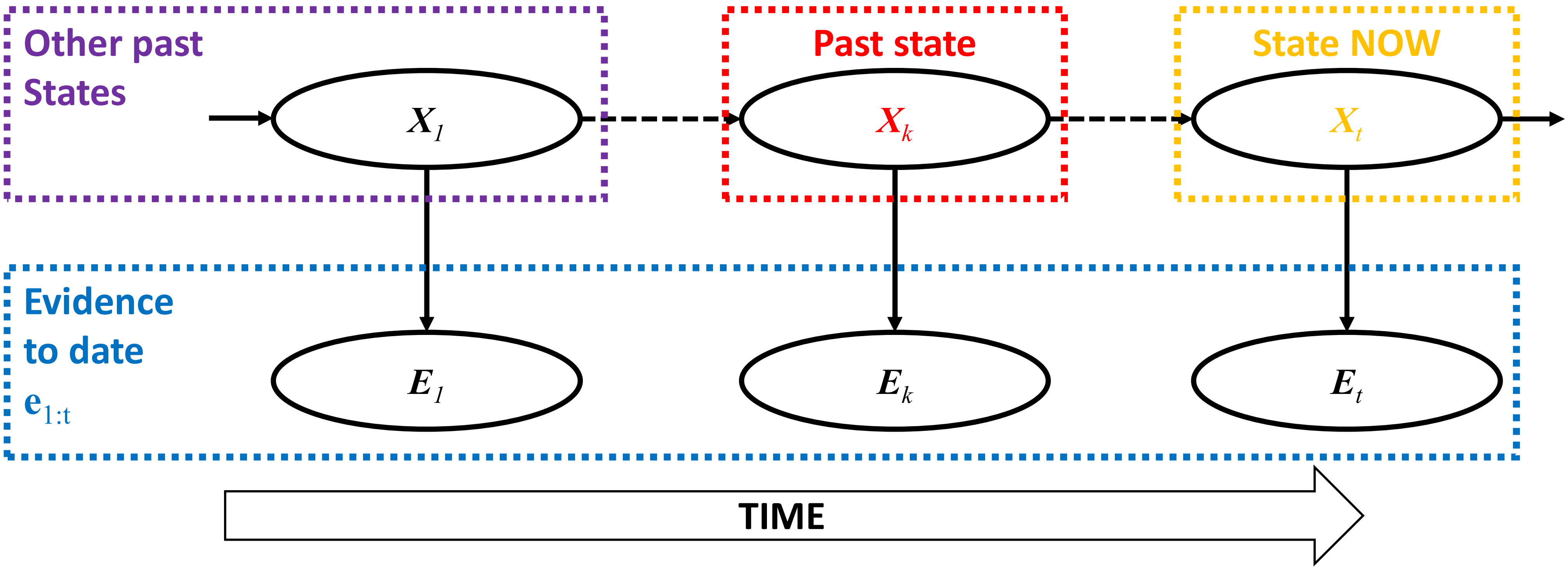
$$P(X_{t+k} \mid \mathbf{e}_{1:t})$$



Inference: Smoothing

This is the task of computing the posterior distribution over the past state (time k, for some $0 \leq k < t$), given all evidence to date. Provides a better state estimate of, because it incorporates more evidence.

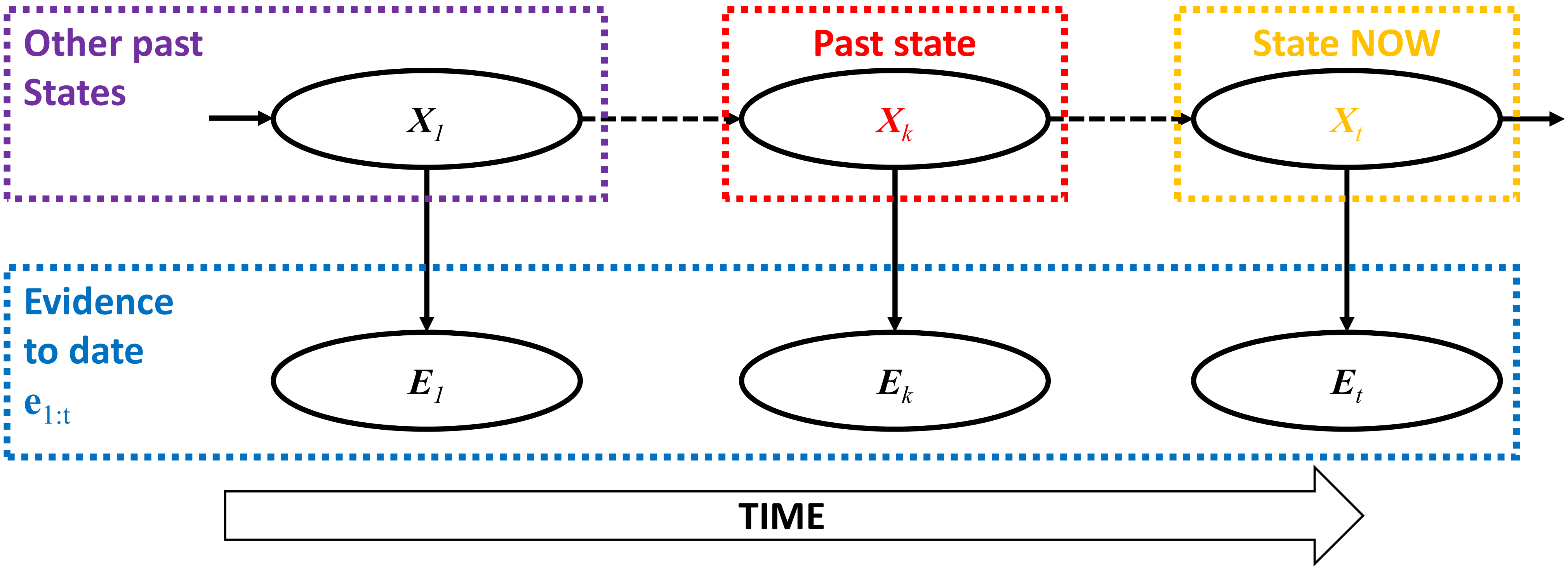
$$P(X_k \mid \mathbf{e}_{1:t})$$



Inference: Smoothing

This is the task of computing the posterior distribution over the past state (time k, for some $0 \le k < t$), given all evidence to date. Provides a better state estimate of, because it incorporates more evidence.

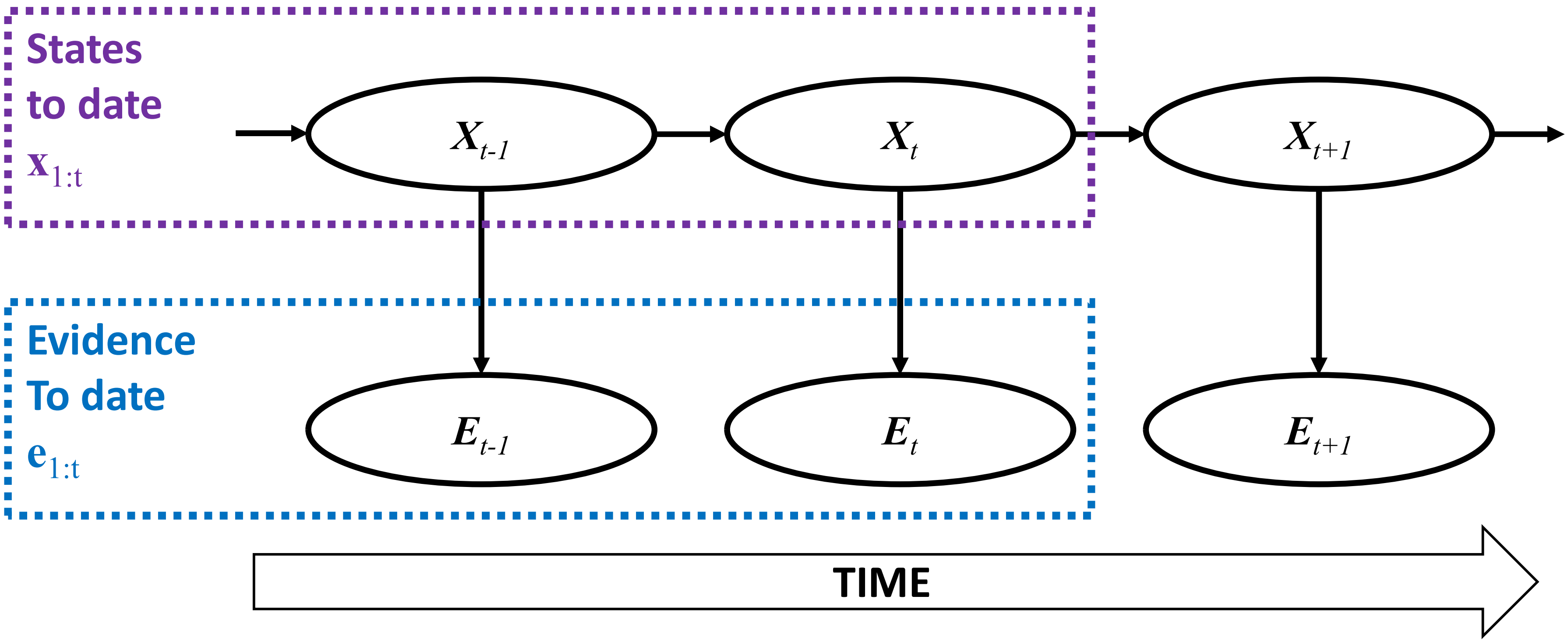
$$P(X_k \mid \mathbf{e}_{1:t})$$



Inference: Most Likely Explanation

Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations.

$$\operatorname{argmax}_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$$



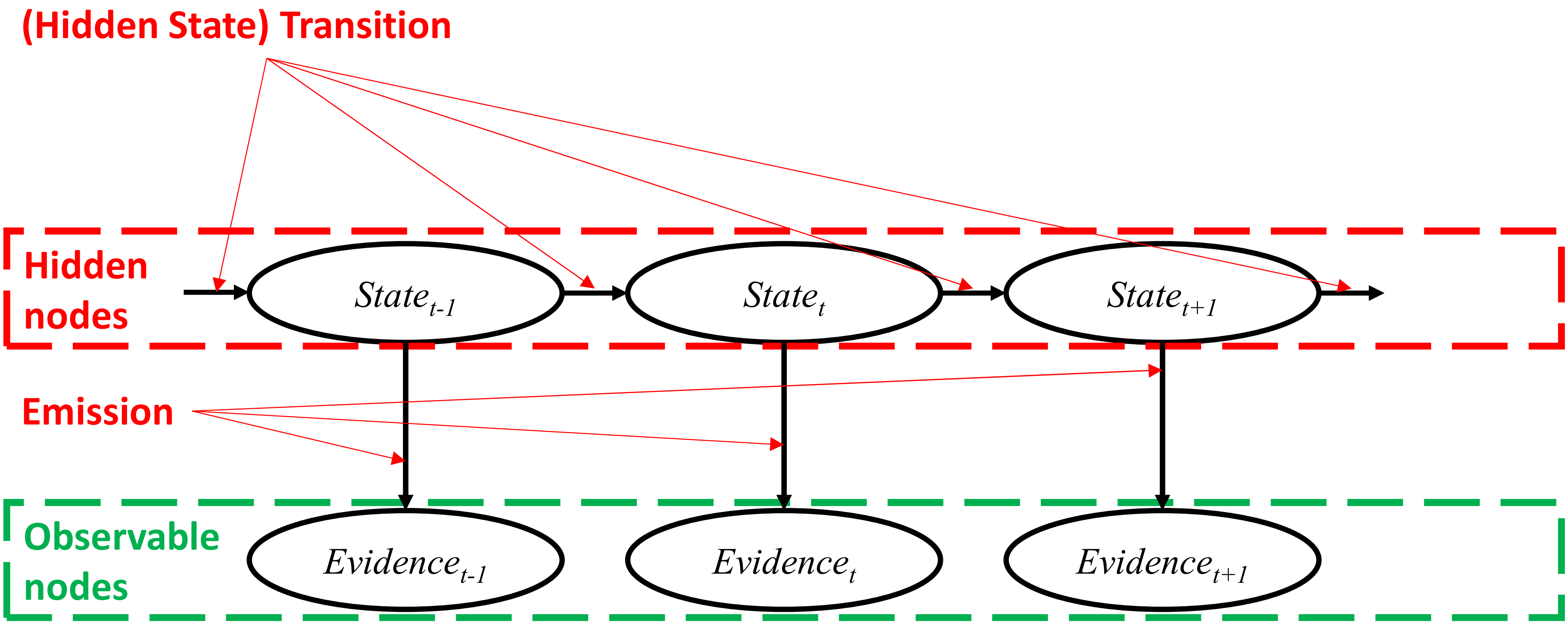
Inference: Most Likely Explanation

Hidden Markov Model

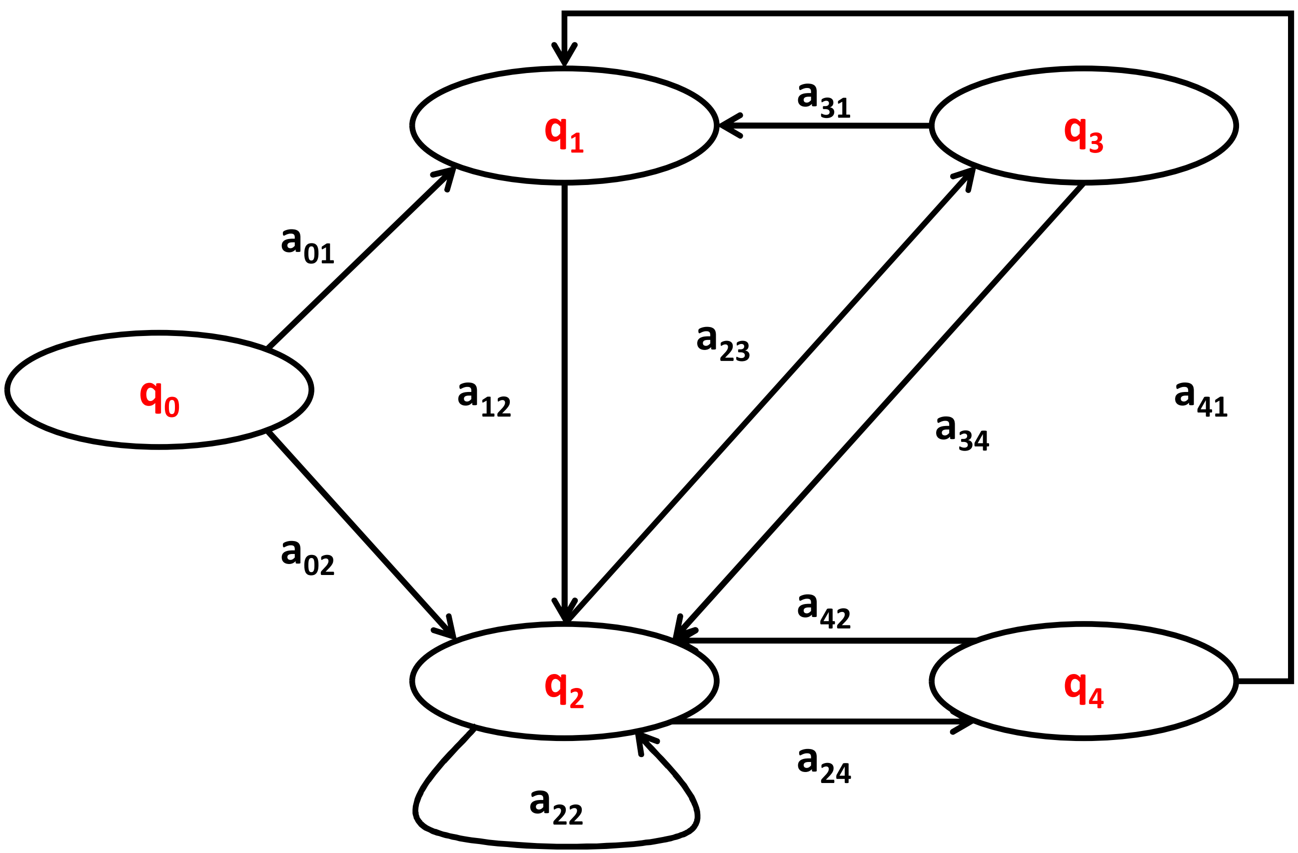
+

Viterbi Algorithm

Transition and Sensor Models



Hidden Markov Model



HMMs are specified with:

- A set of N states:

  $Q = \{q_1, q_2, \dots, q_N\}$

- A transition probability matrix A, where each $a_{i,j}$ represents the probability of moving from state q_i to state q_j

- A sequence of T observations O:

  $O = o_1, o_2, \dots, o_T$

- A sequence of observation likelihoods (emission probabilities): probability of observation o_t being generated by a state q_i:

  $B = b_i(o_t)$

- Special start (<s>) and end (final: not here) states

  q_0 and q_F

Transition probability matrix A

	q_0	q_1	q_2	q_3	q_4	Notes
q_0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	row sum = 1
q_1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	row sum = 1
q_2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	row sum = 1
q_3	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	row sum = 1
q_4	$a_{4.,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	row sum = 1

Hidden Markov Models: Decoding

The task of determining which sequence of variables is the underlying source of some sequence of observations is called the decoding:

Given as input an HMM $\alpha = (A, B)$ *and a sequence of observations* $o_1, o_2, \ldots, o_T$ *find the most probable sequence of states* $q_1, q_2, \ldots, q_T$.

or in our case:

Given as input an HMM $\alpha = (A, B)$ *and a sequence of* ***words*** $w_1, w_2, \ldots, w_T$ *find the most probable sequence of* ***tags/states*** $C_1, C_2, \ldots, C_T$.

A - transition probabilities matrix

B - emission probabilities matrix

Viterbi Algorithm: Pseudocode

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Reinforcement Learning

Main Machine Learning Categories

Supervised learning	Unsupervised learning	Reinforcement learning
Supervised learning is one of the most common techniques in machine learning. It is based on **known relationship(s) and patterns within data** (for example: relationship between inputs and outputs). Frequently used types: **regression**, and **classification**.	**Unsupervised learning** involves finding underlying patterns within data. Typically used in **clustering** data points (similar customers, etc.)	Reinforcement learning is inspired by behavioral psychology. It is **based on a rewarding / punishing an algorithm**. Rewards and punishments are based on algorithm's action within its environment.

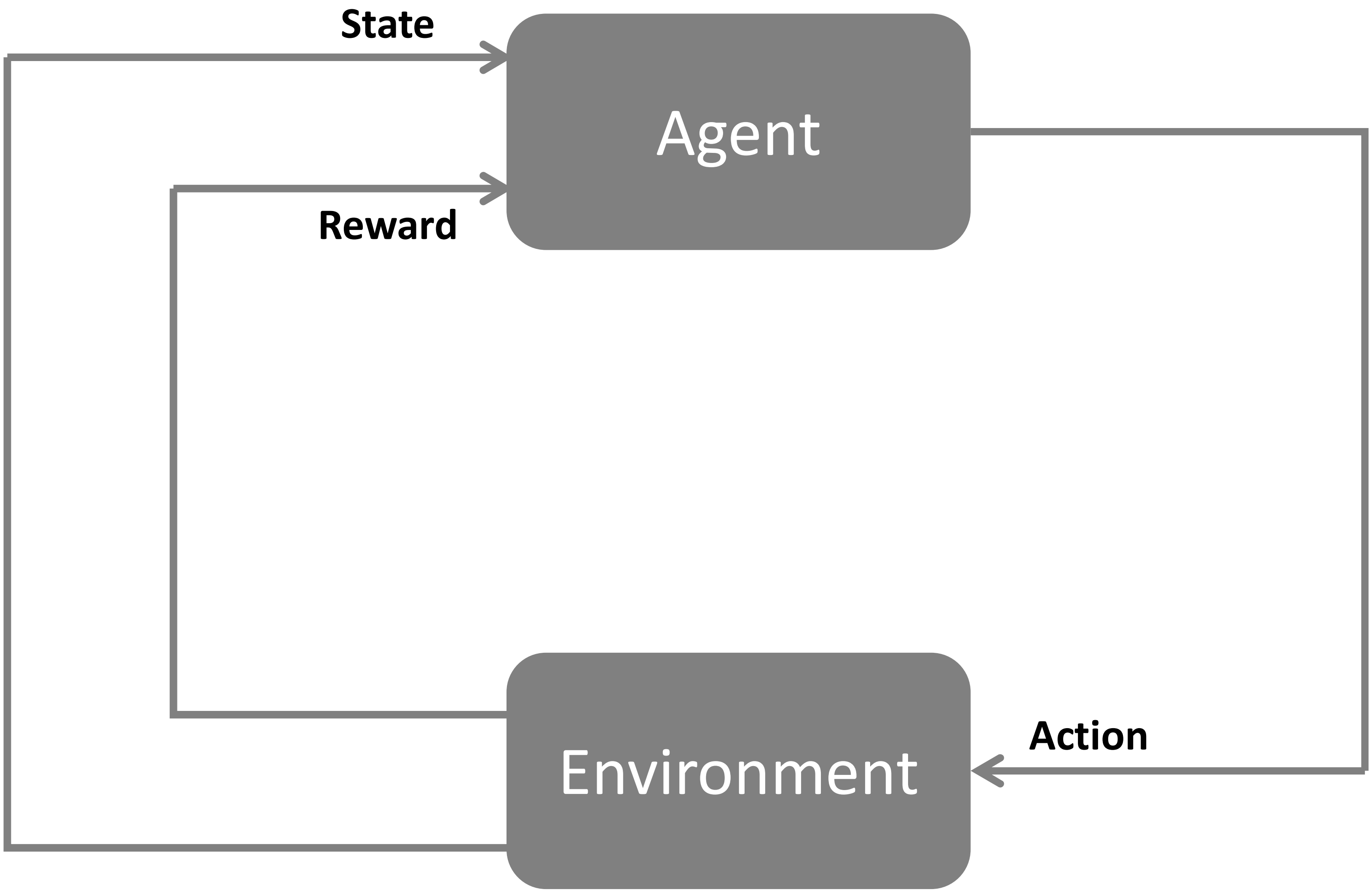
What is Reinforcement Learning?

Idea:

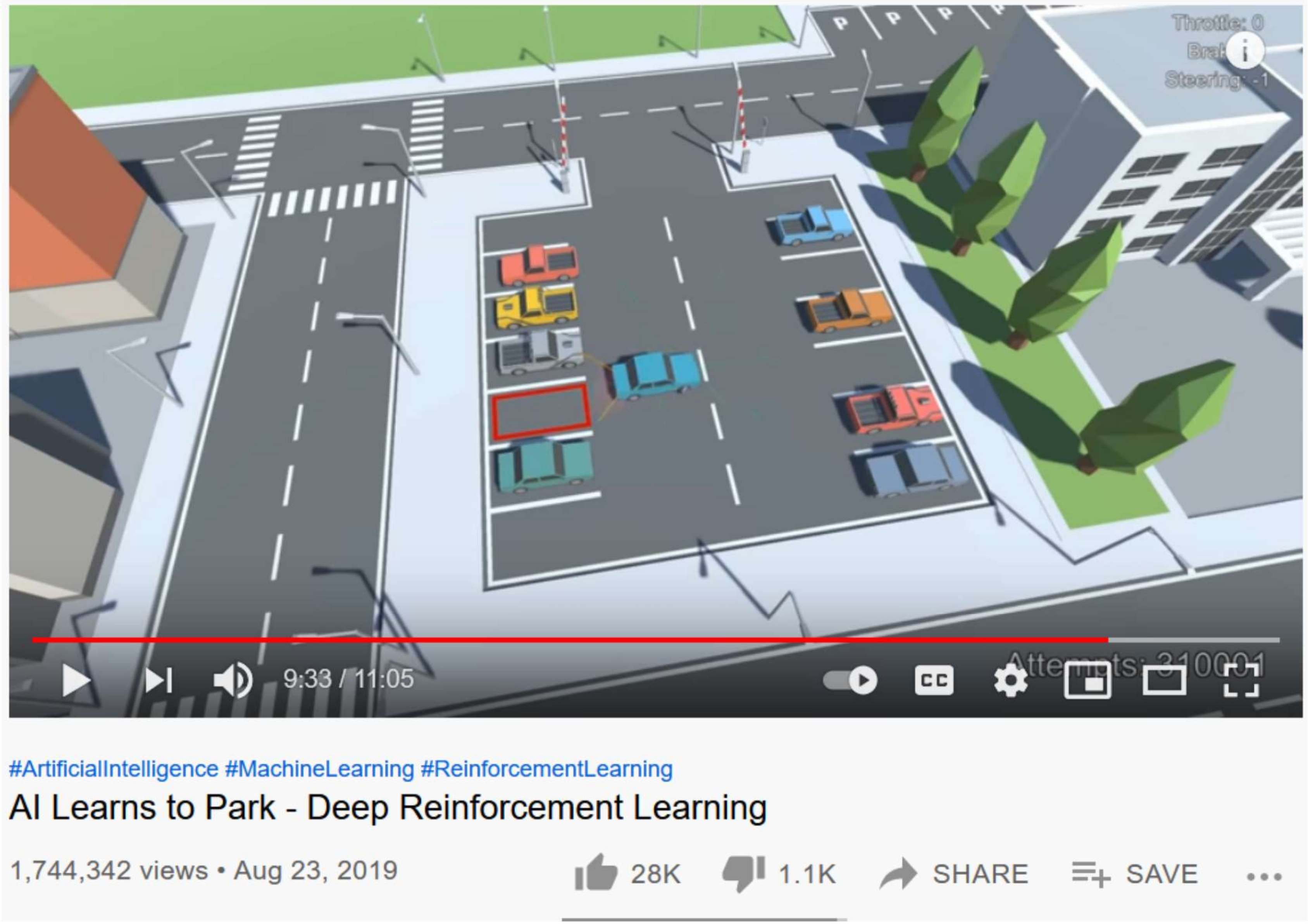
Reinforcement learning is inspired by behavioral psychology. It is **based on a rewarding / punishing an algorithm**.

Rewards and punishments are based on algorithm's action within its environment.

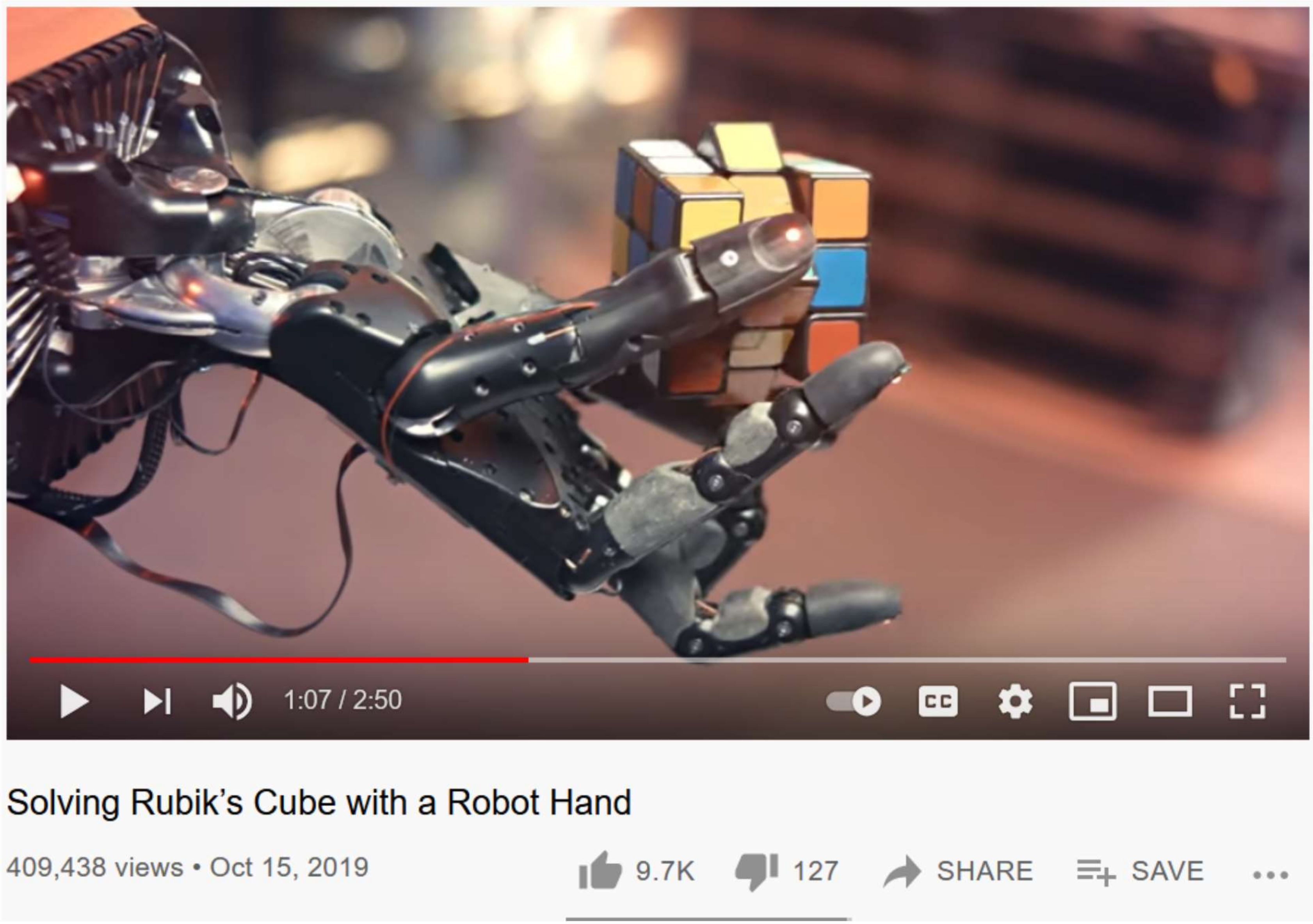
RL: Agents and Environments



Reinforcement Learning in Action



Reinforcement Learning in Action

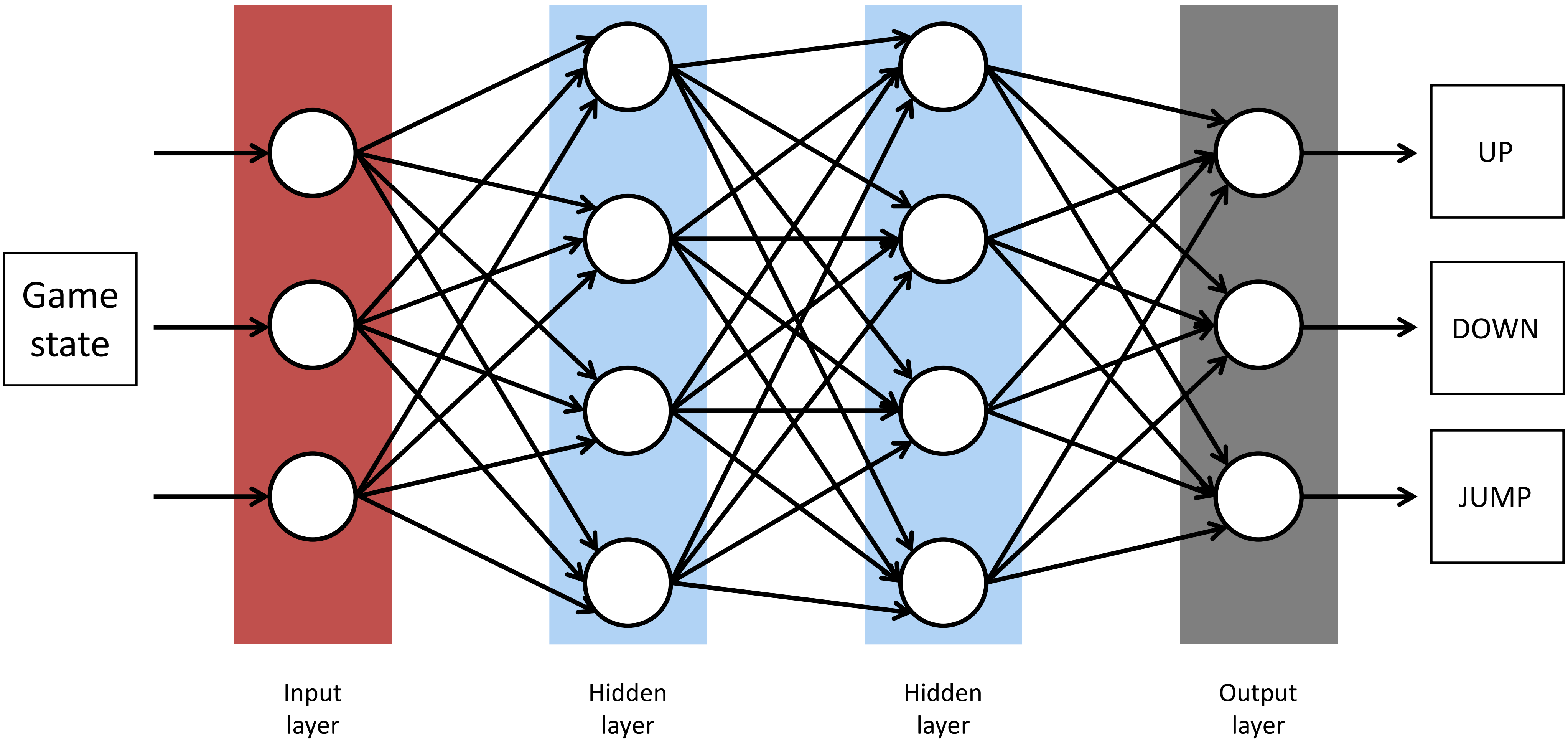


Source: https://www.youtube.com/watch?v=x4O8pojMF0w

Reinforcement Learning in Action

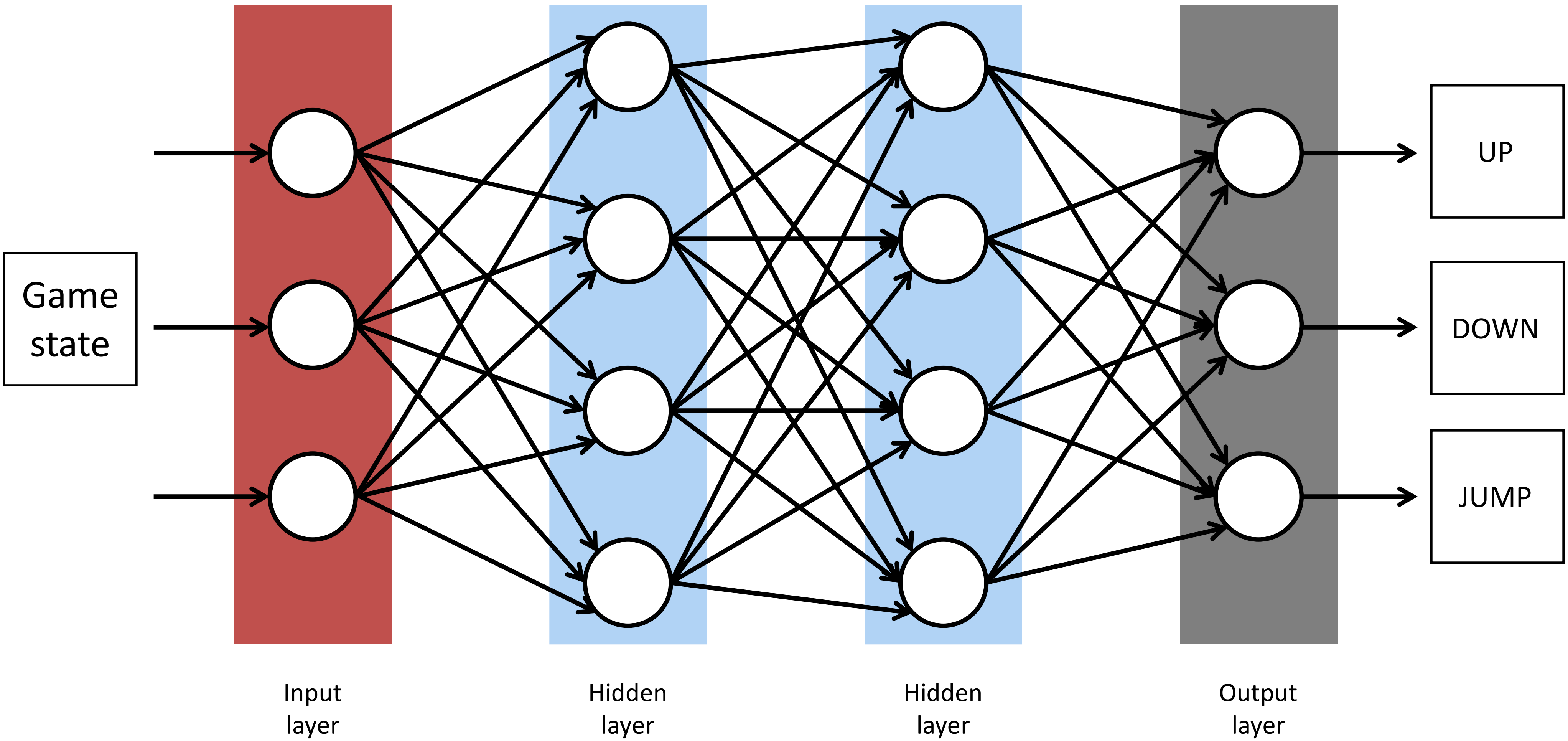
Source: https://www.youtube.com/watch?v=kopoLzvh5jY

ANN for Simple Game Playing



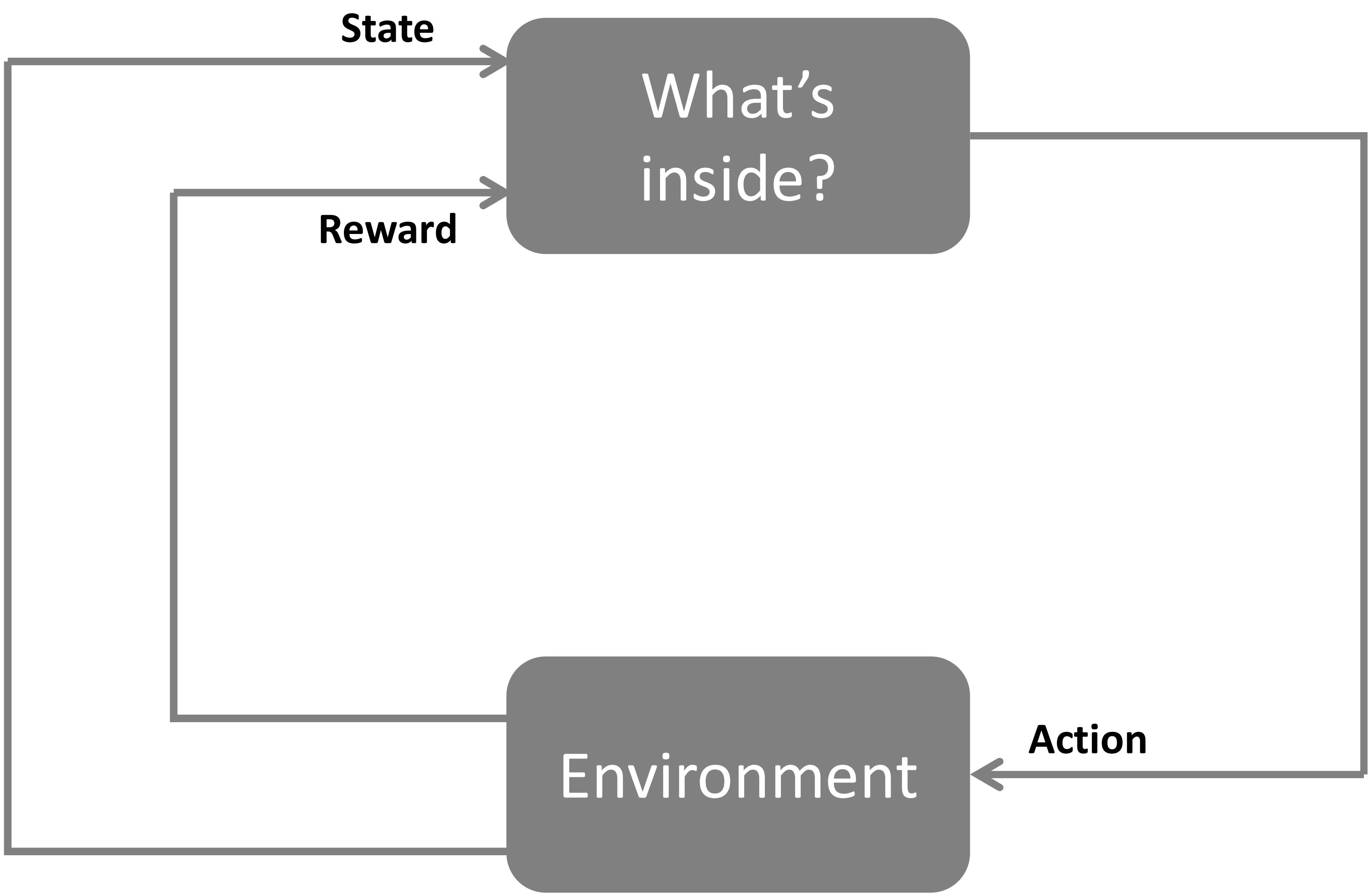
ANN for Simple Game Playing

Current game is an input. Decisions (UP/DOWN/JUMP) are rewarded/punished.

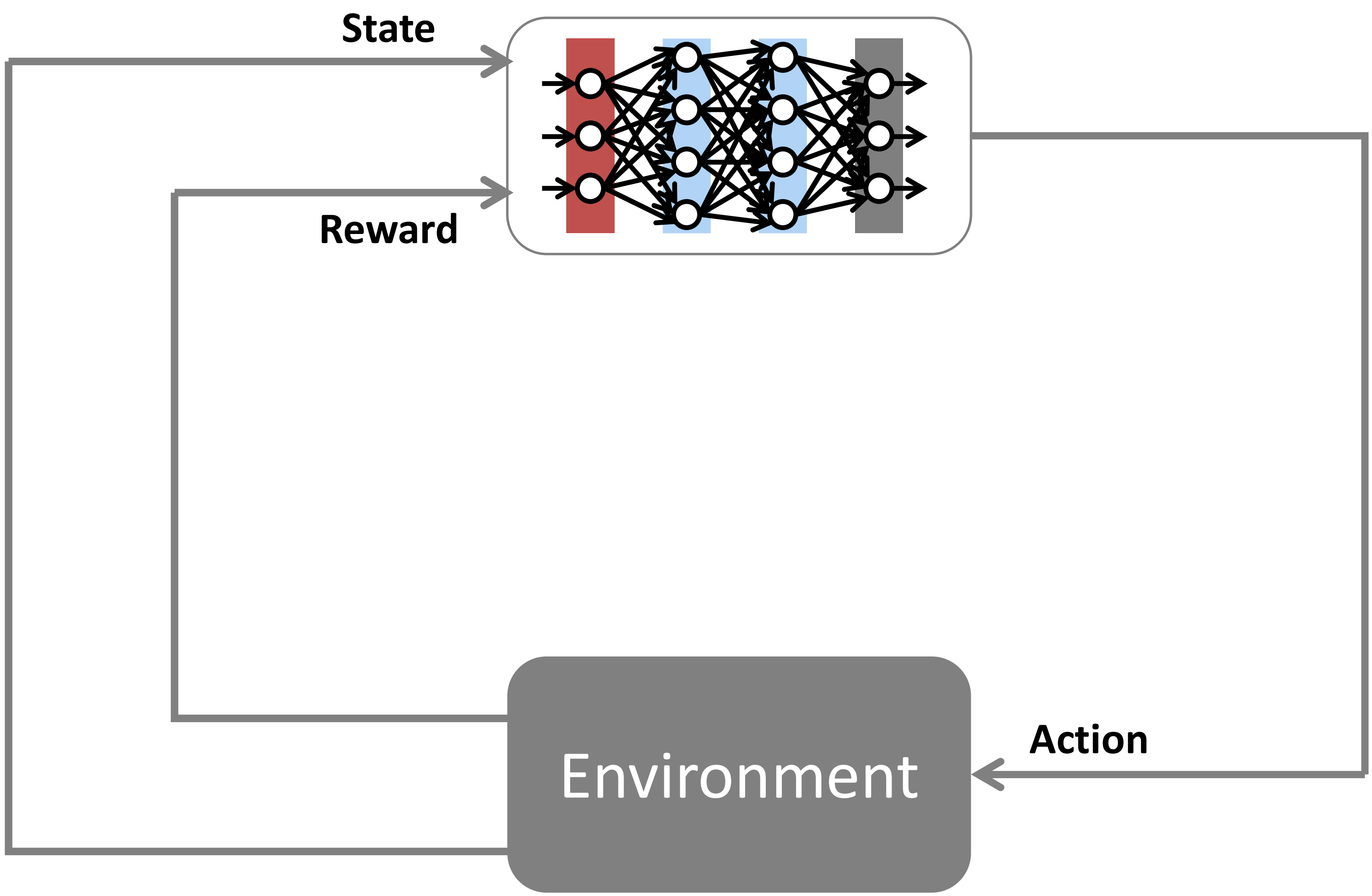


Correct all the weights using Reinforcement Learning.

RL: Agents and Environments



RL: Agents and Environments



K-Armed Bandit Problem

K-Armed Bandit Problem

The K-armed bandit problem is a problem in which a fixed limited set of resources must be allocated between competing (alternative) choices in a way that maximizes their expected gain.

Each choice's properties are only partially known at the time of allocation, and may become better understood as time passes or by allocating resources to the choice.

K-Armed Bandit Problem

In the problem, each machine provides a random reward from a probability distribution specific to that machine, that is not known a-priori.

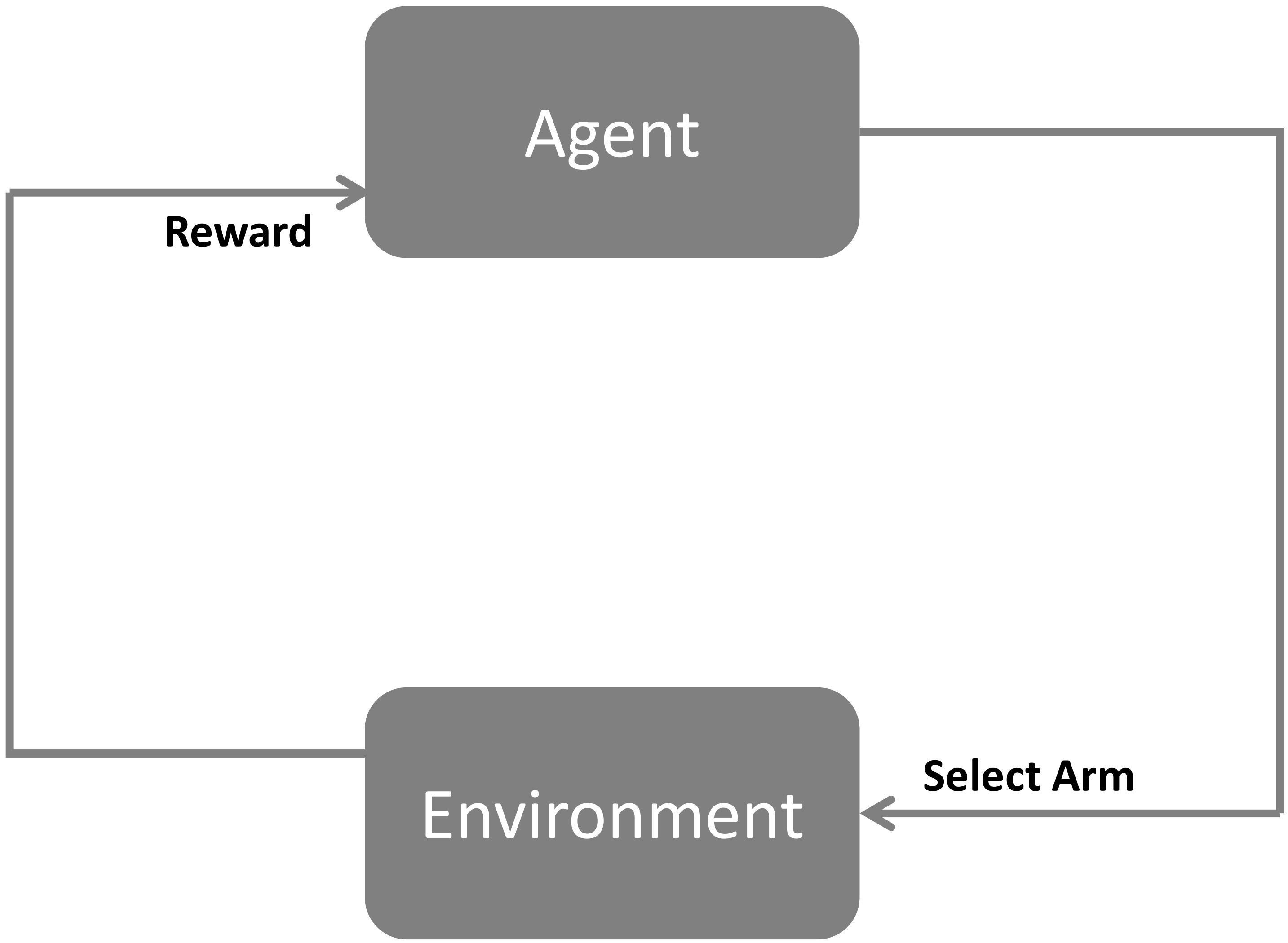
The objective of the gambler is to maximize the sum of rewards earned through a sequence of lever pulls.

K-Armed Bandit Problem

Bandit/Arm 1	Bandit/Arm 2	Bandit/Arm 3
33 %	52 %	78 %
current success (win) rate	**current** success (win) rate	**current** success (win) rate

Which bandit shall we play next?

K-Armed Bandit



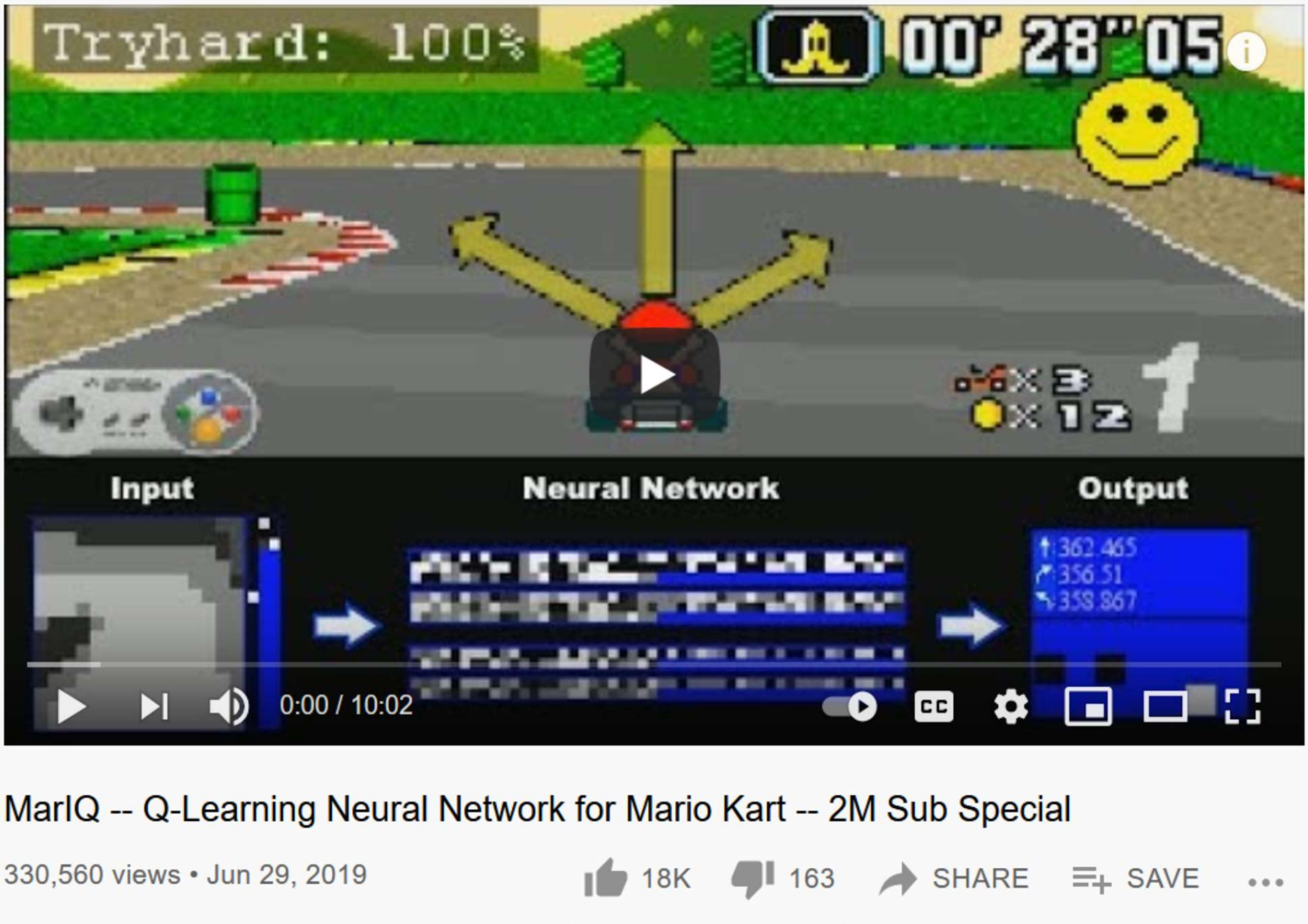
Exploration vs. Exploitation

The crucial tradeoff the gambler faces at each trial is between "exploitation" of the machine that has the highest expected payoff and "exploration" to get more information about the expected payoffs of the other machines.

ε-greedy Algorithm

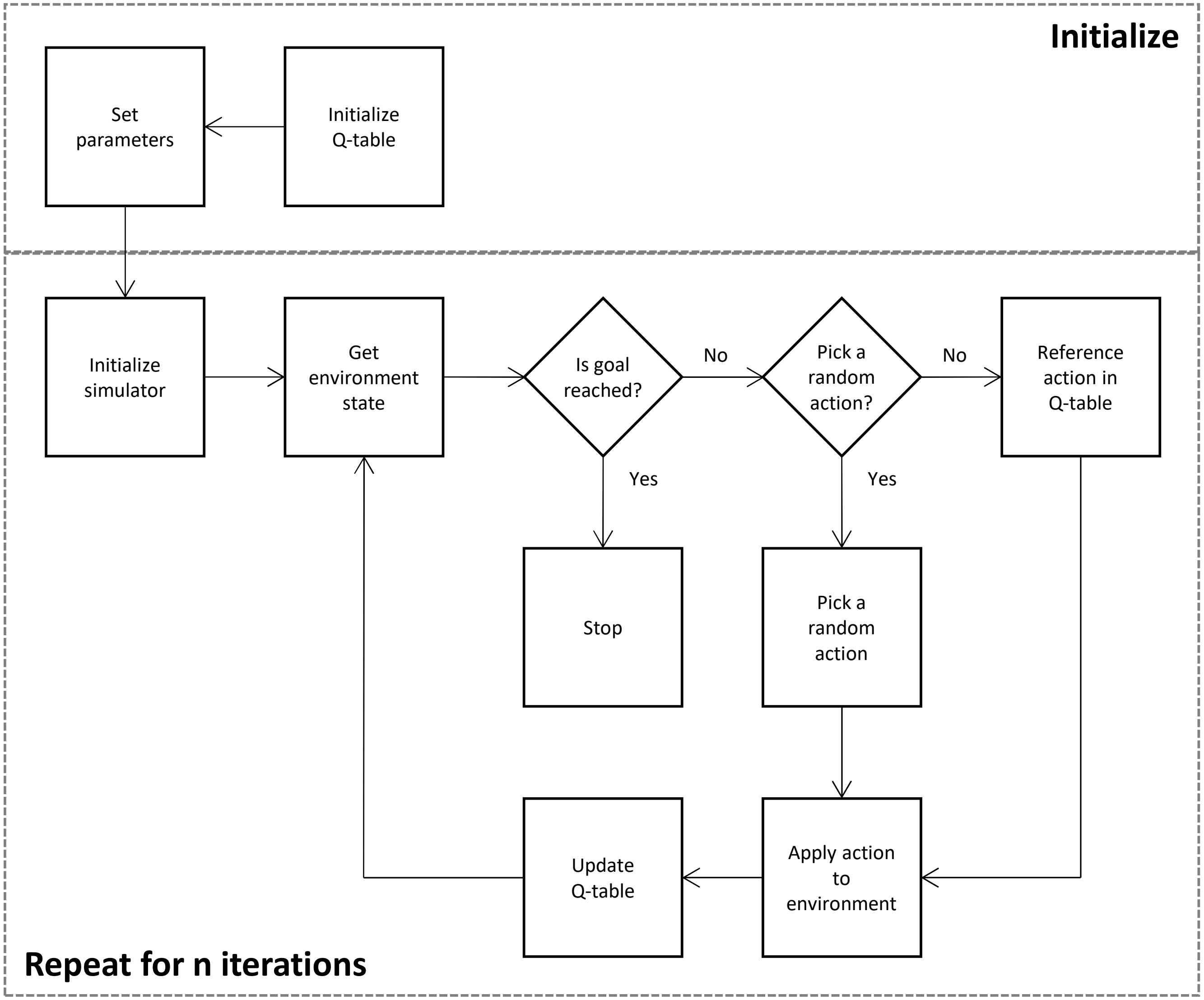
```
generate random number p ∈ [0,1]

if (p < ε)          // explore

    select random arm

else                // exploit

    select current best arm

end
```

Reinforcement Learning in Action

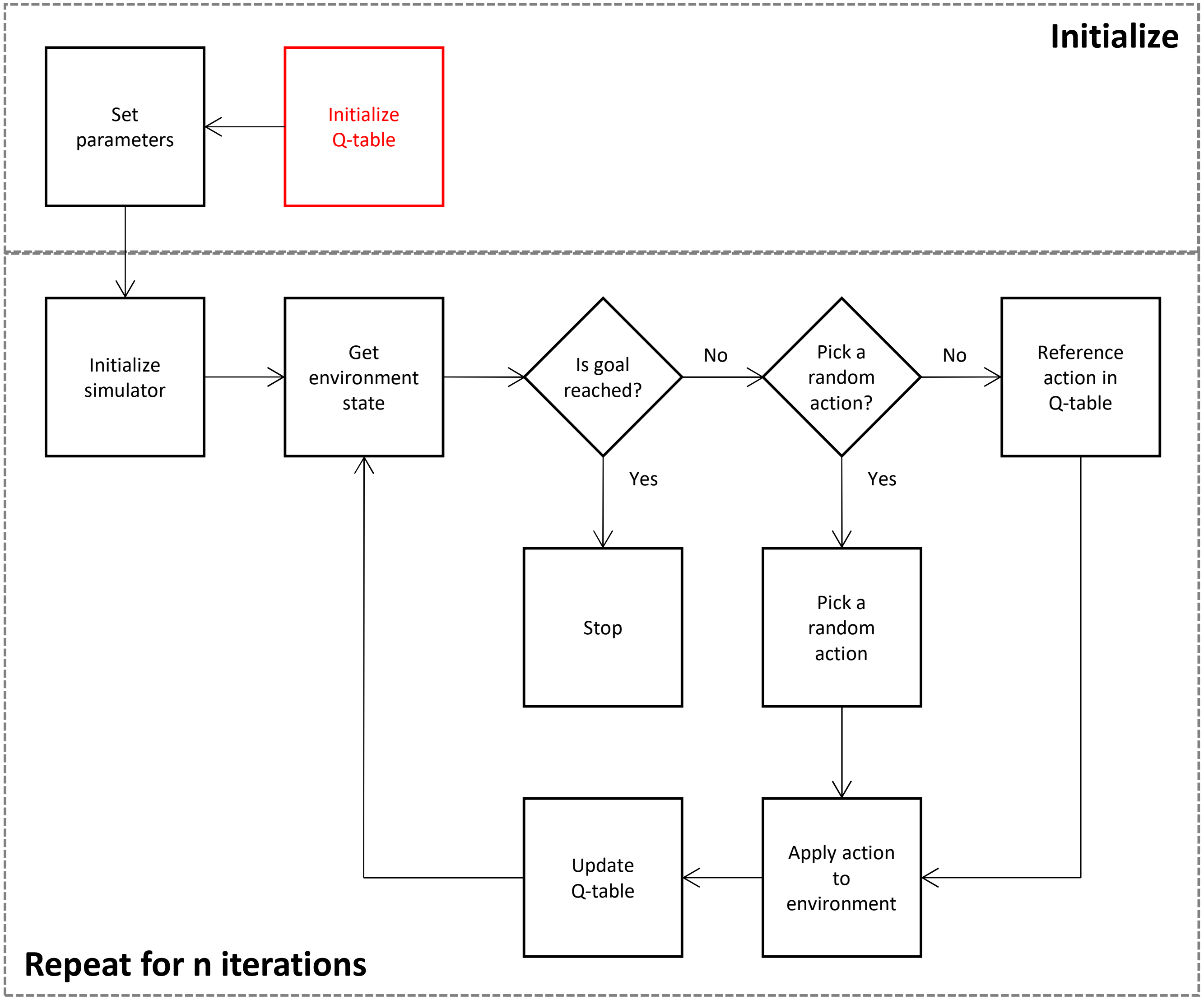


Source: https://www.youtube.com/watch?v=Tnu4O_xEmVk

Q-Learning Algorithm



Q-Learning Algorithm



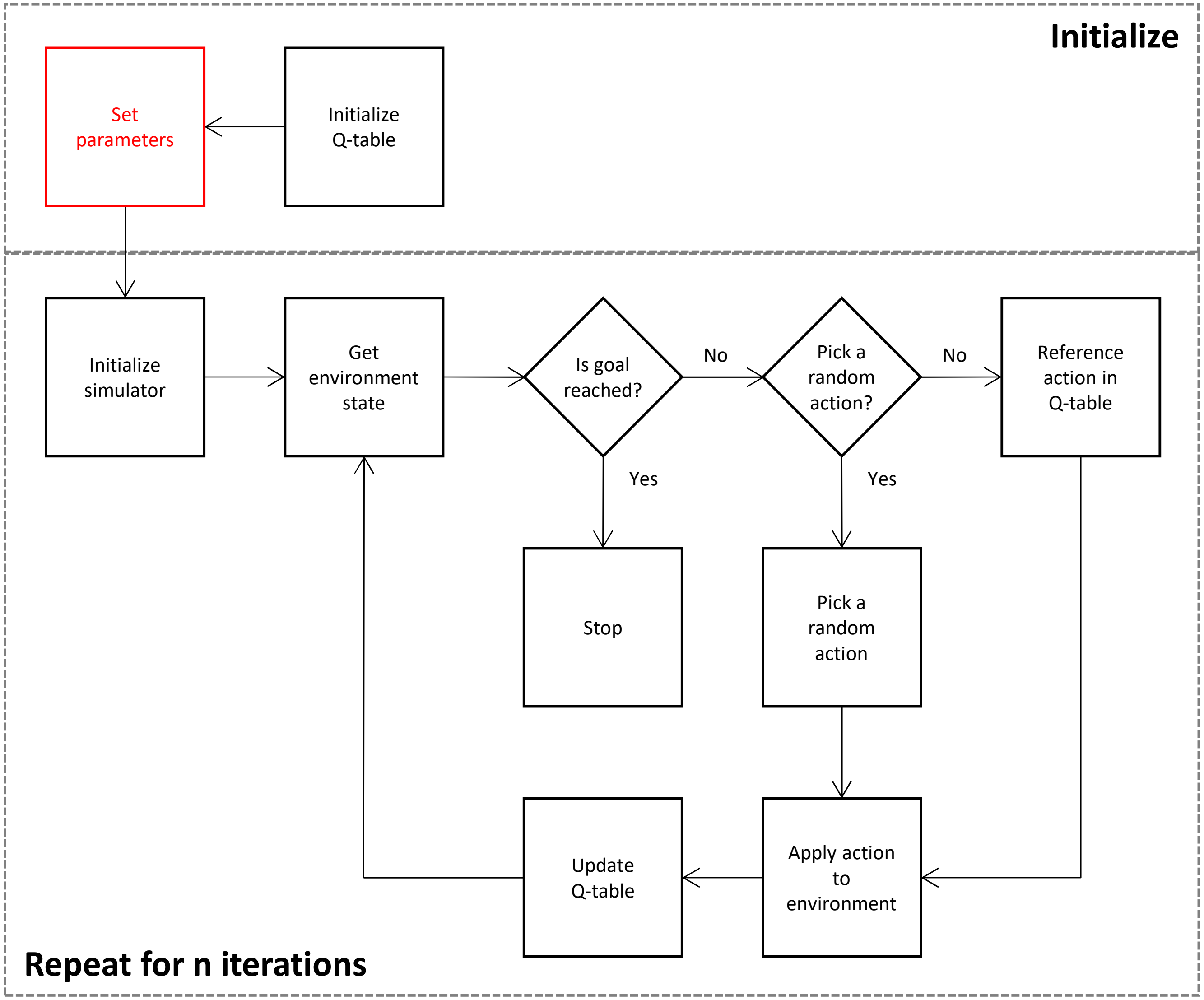
Initialize Q-table:

Set up and initialize (**all values set to 0**) a table where:

- rows represent **possible states**
- columns represent **actions**

Note that additional states can be added to the table when encountered.

Q-Learning Algorithm

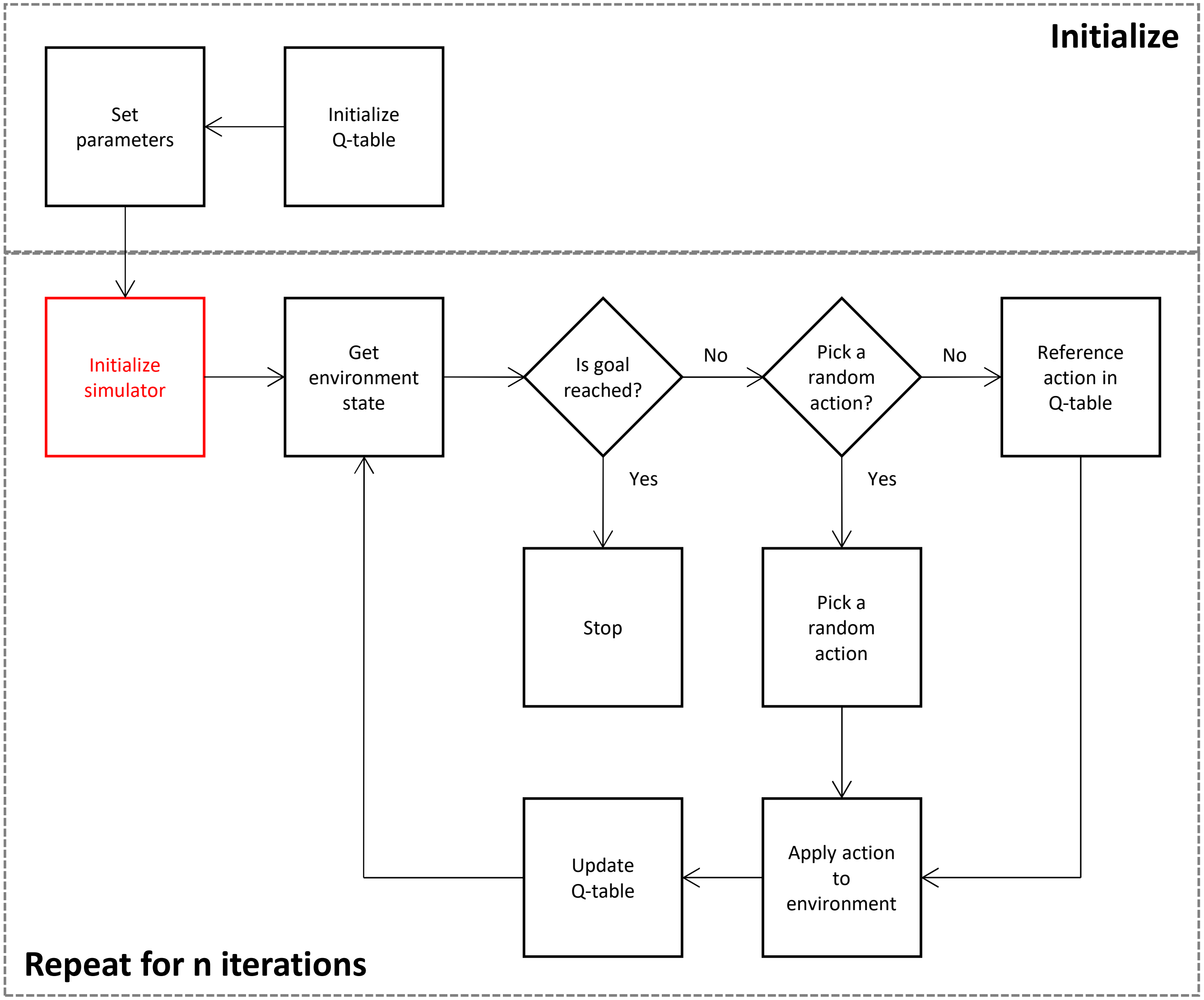


Set parameters:
Set and initialize **hyperparameters** for the Q-learning process.

Hyperparemeters include:

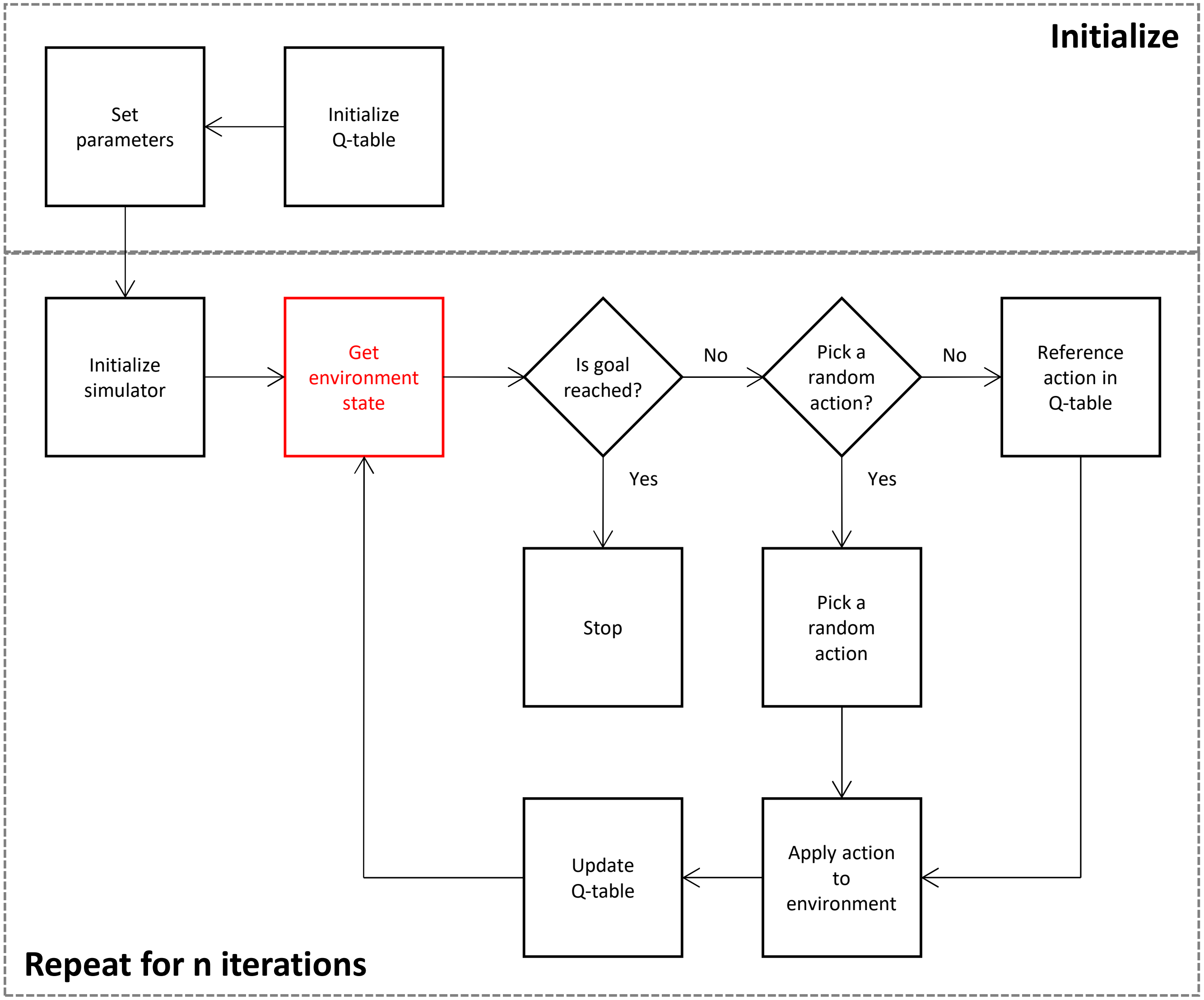
- **chance of choosing a random action**: a threshold for choosing a random action over an action from the Q-table
- **learning rate**: a parameter that describes how quickly the algorithm should learn from rewards in different states
  - high: faster learning with erratic Q-table changes
  - low: gradual learning with possibly more iterations
- **discount factor**: a parameter that describes how valuable are future rewards. It tells the algorithm whether it should seek "immediate gratification" (small) or "long-term reward" (large)

Q-Learning Algorithm



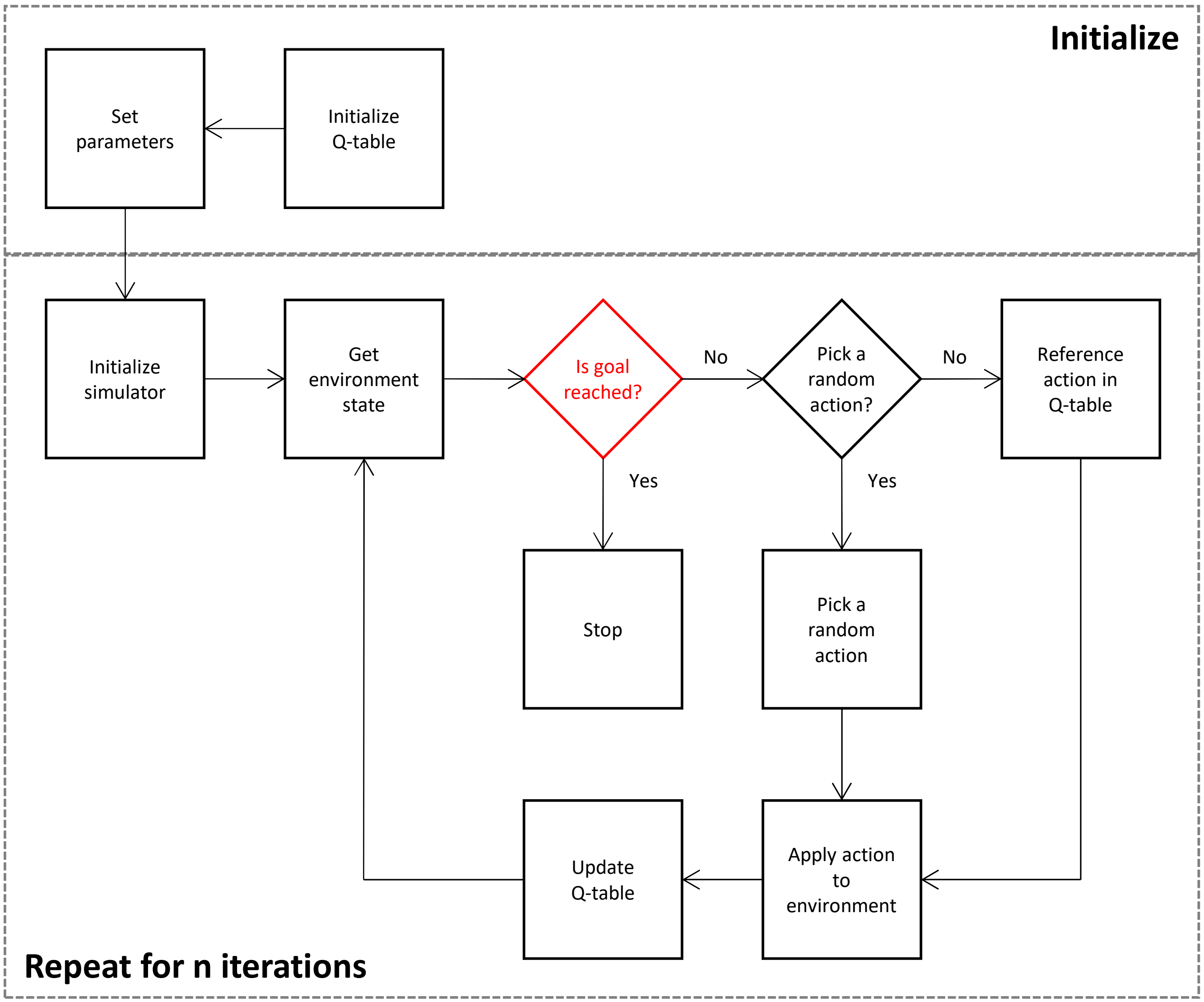
Initialize simulator:
Reset the simulated environment to its initial state and place the agent in a neutral state.

Q-Learning Algorithm



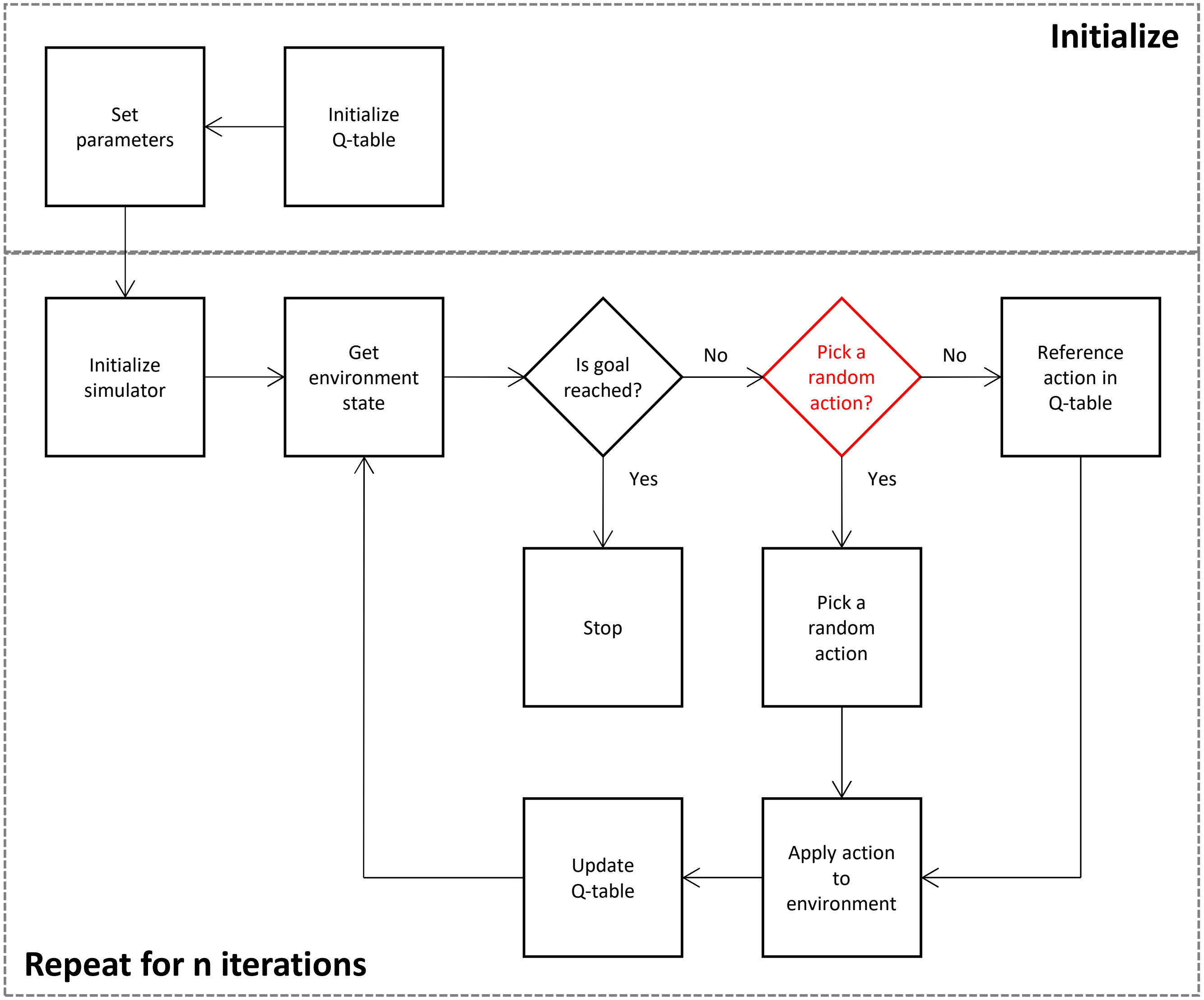
Get environment state:
Report the current state of the environment. Typically a vector of values representing all relevant variables.

Q-Learning Algorithm



Is goal reached?:
Verify if the goal of the simulation has been achieved. It could be decided with the agent arriving in expected final state or by some simulation parameter.

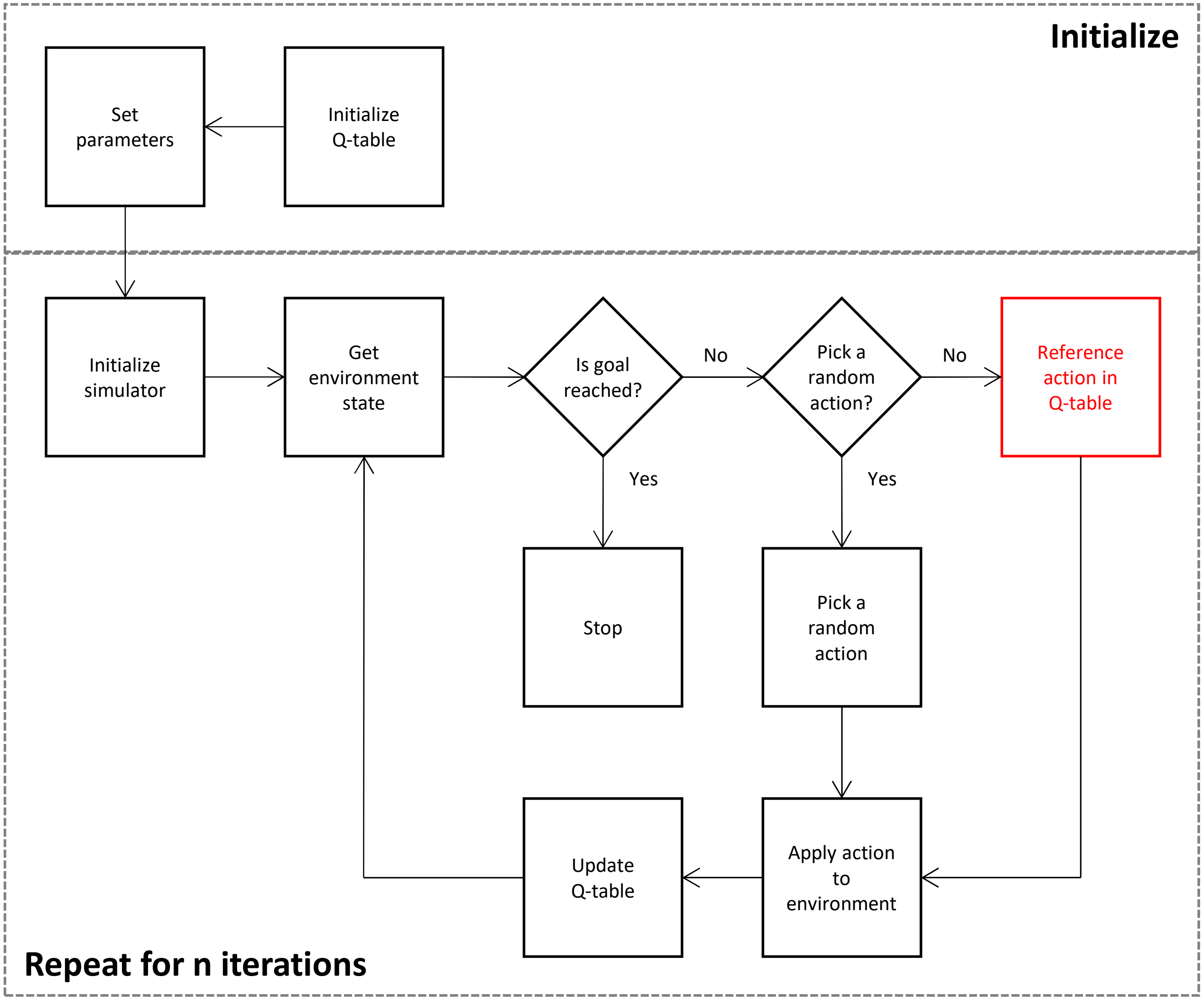
Q-Learning Algorithm



Pick a random action?:
Decide whether next action should be picked at random or not (it will be selected based on Q-table data then).

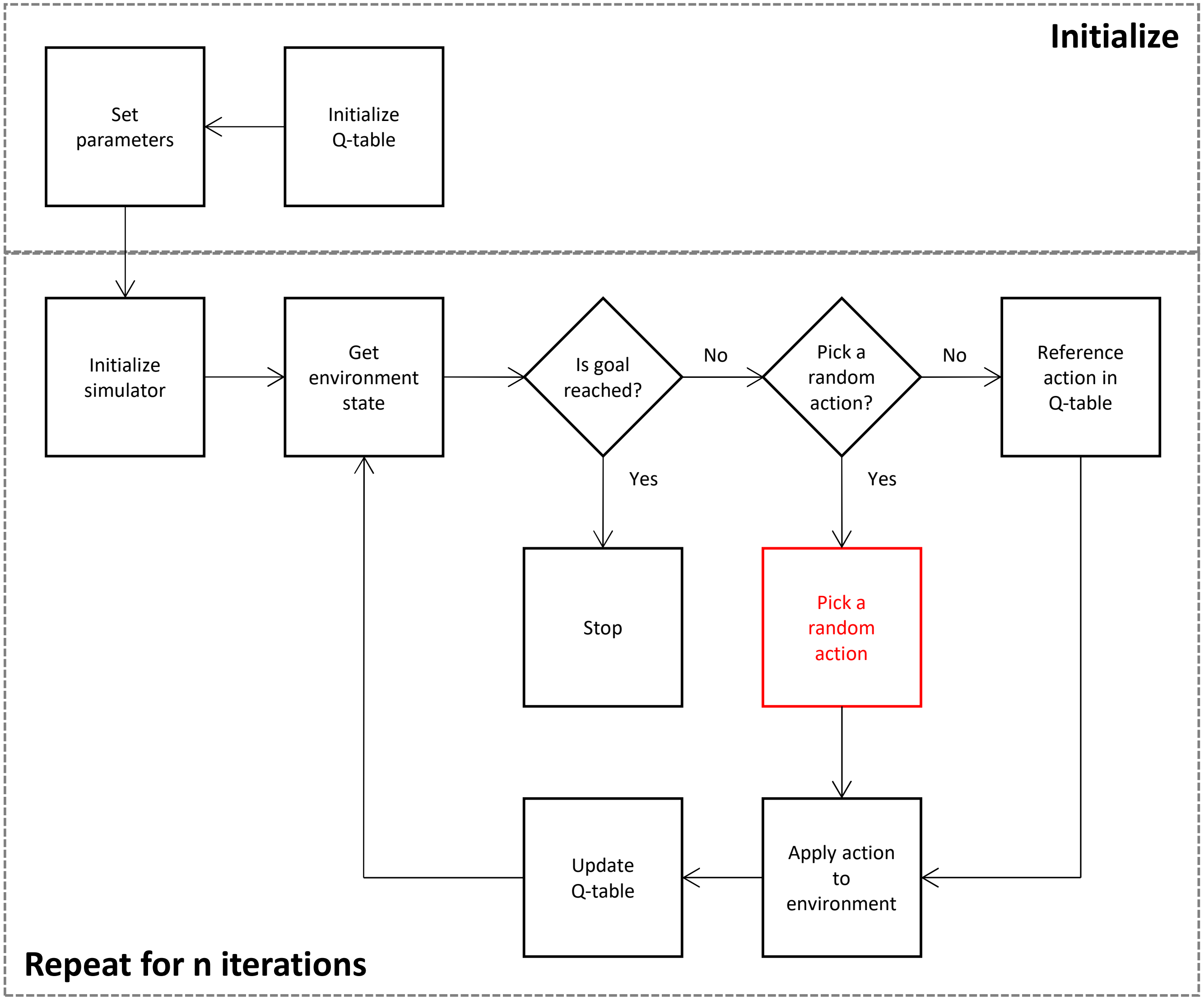
Use the **chance of choosing a random action** **hyperparameter** to decide.

Q-Learning Algorithm



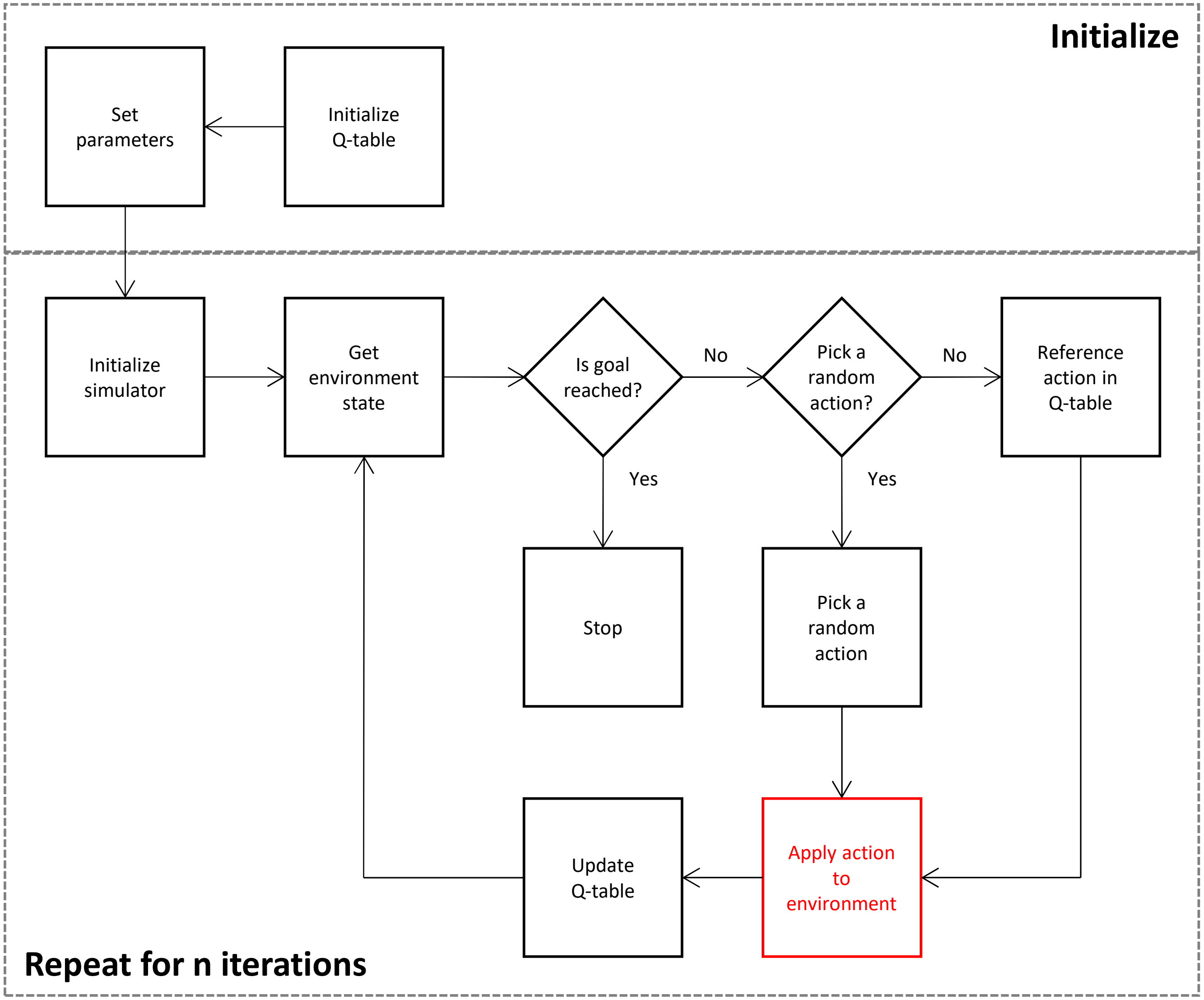
Reference action in Q-table:
Next action decision will be based on data from the Q-table given the current state of the environment.

Q-Learning Algorithm



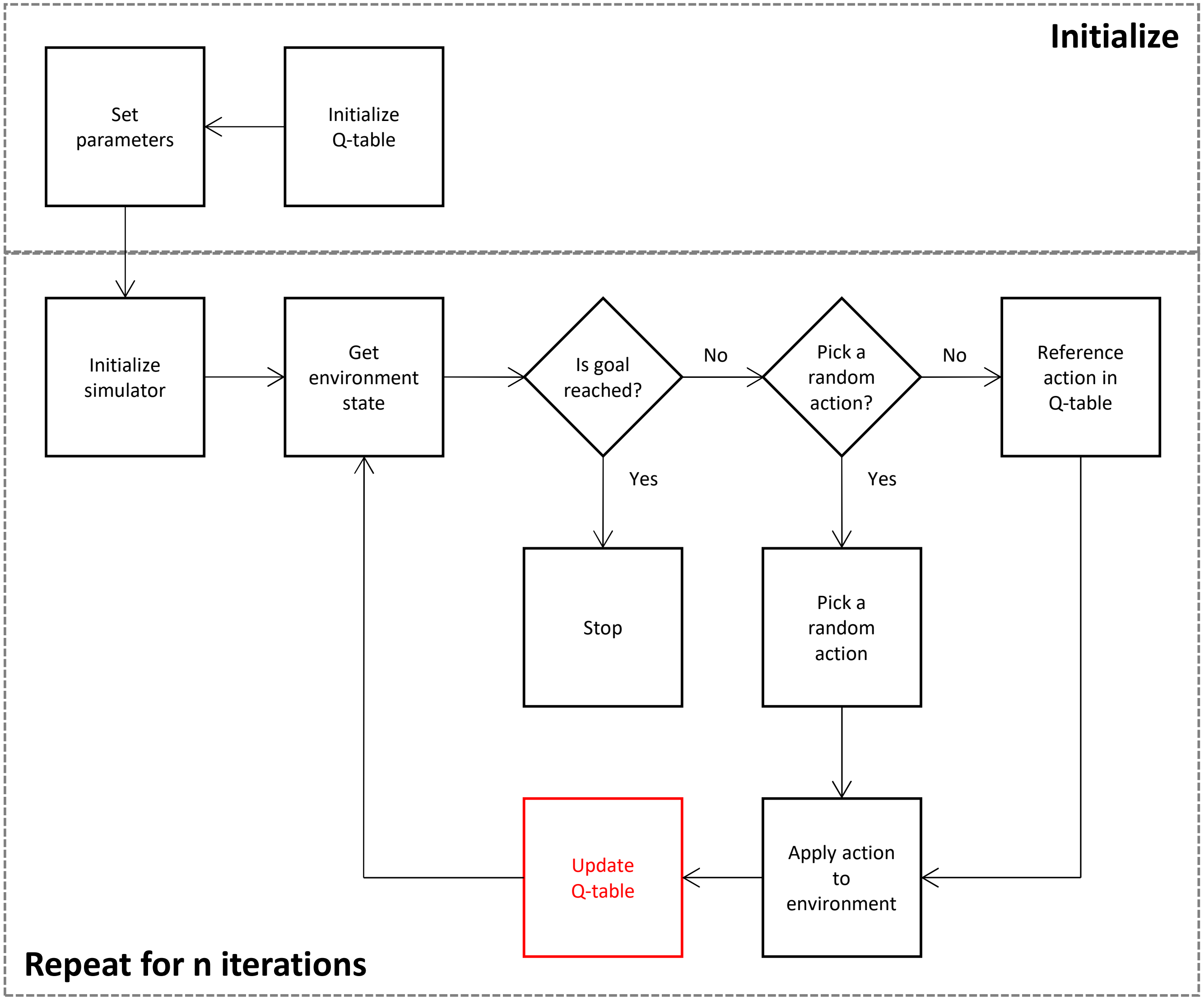
Pick a random action:
Pick any of the available actions at random. Helpful with exploration of the environment.

Q-Learning Algorithm



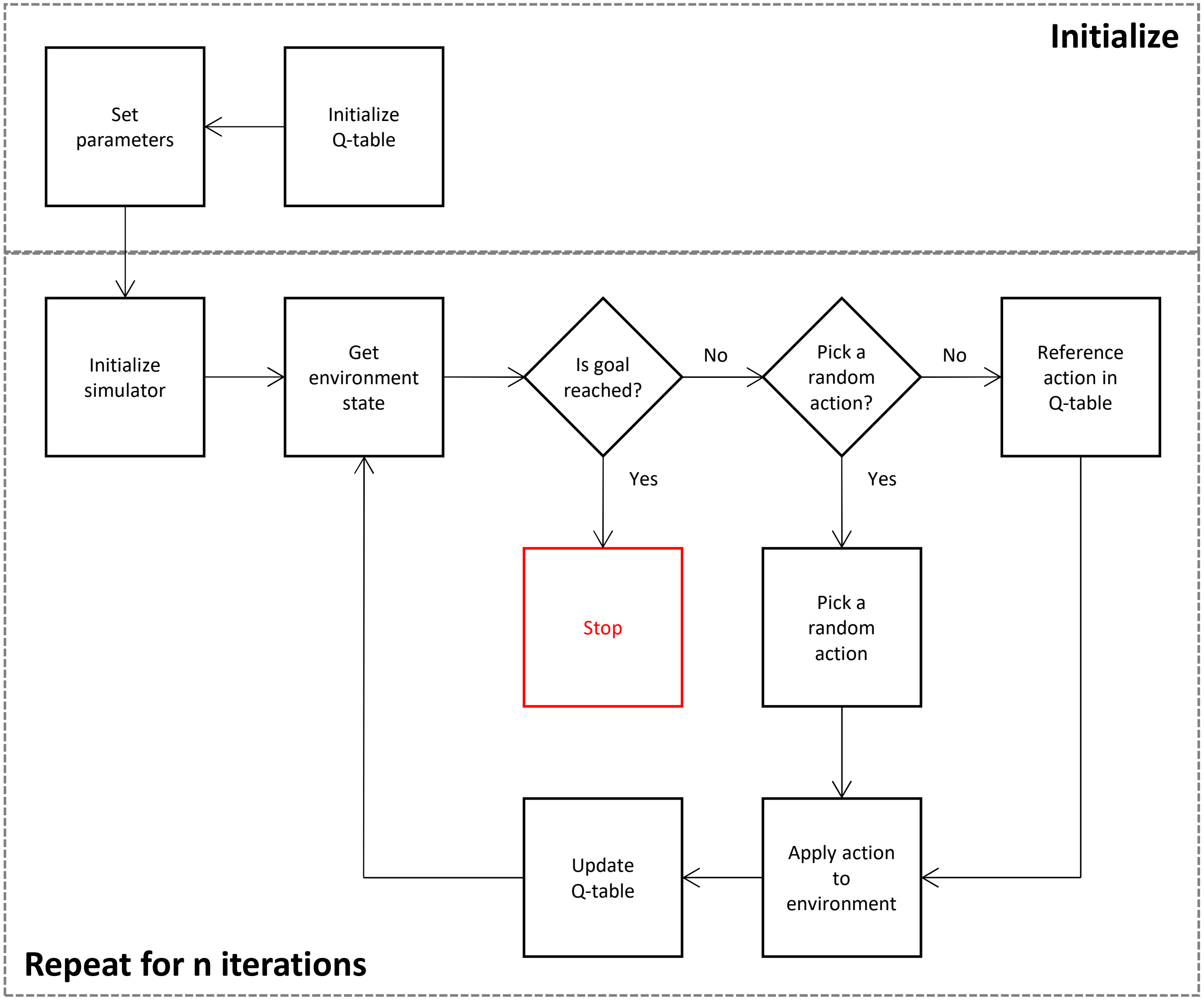
Apply action to environment: Apply the action to the environment to change it. Each action will have its own reward.

Q-Learning Algorithm



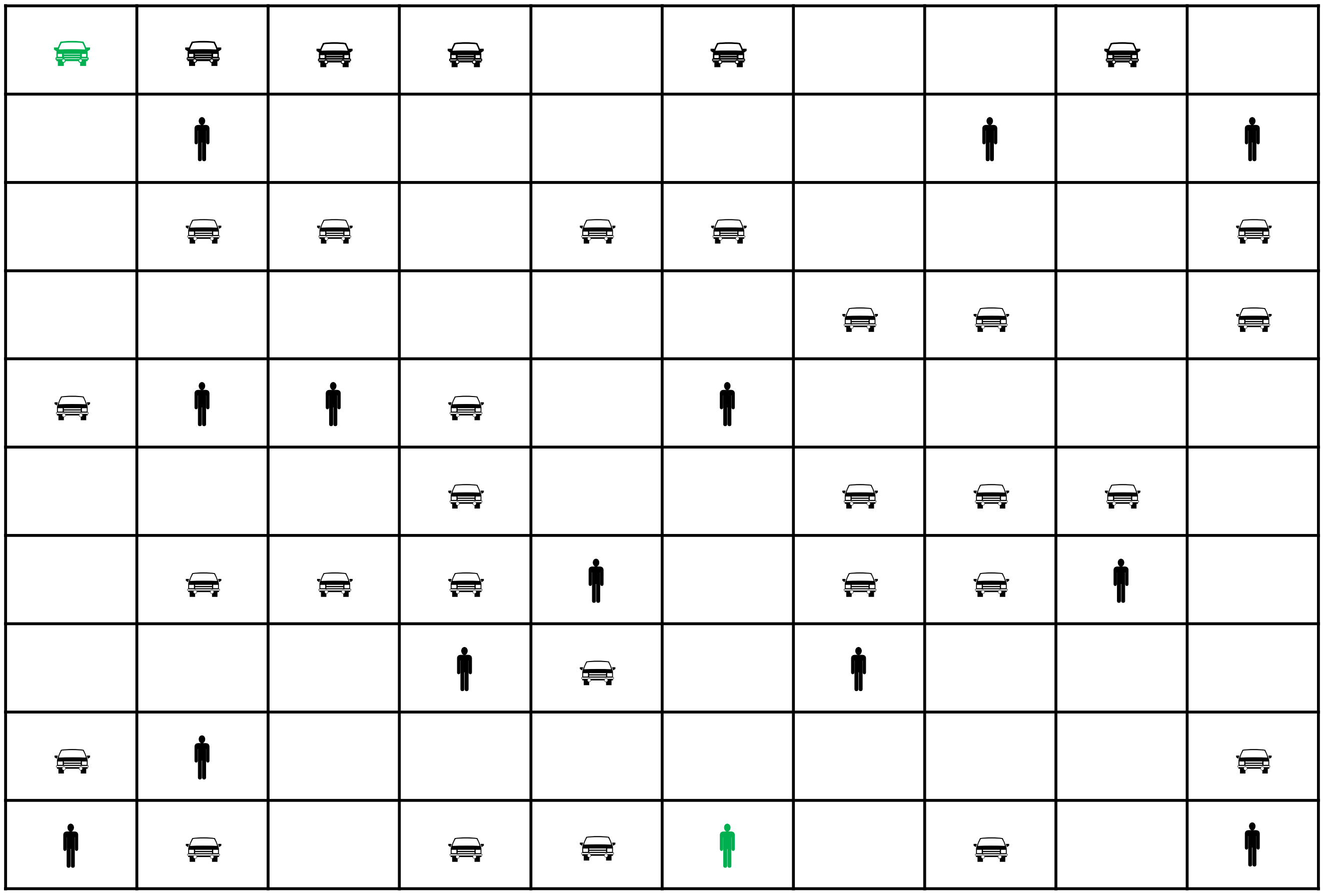
Update Q-table:
Update the Q-table given the reward resulting from recently applied action (feedback from the environment).

Q-Learning Algorithm



Stop:
Stop the learning process

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

Action:

Reward:

Q-table value:

Learning rate

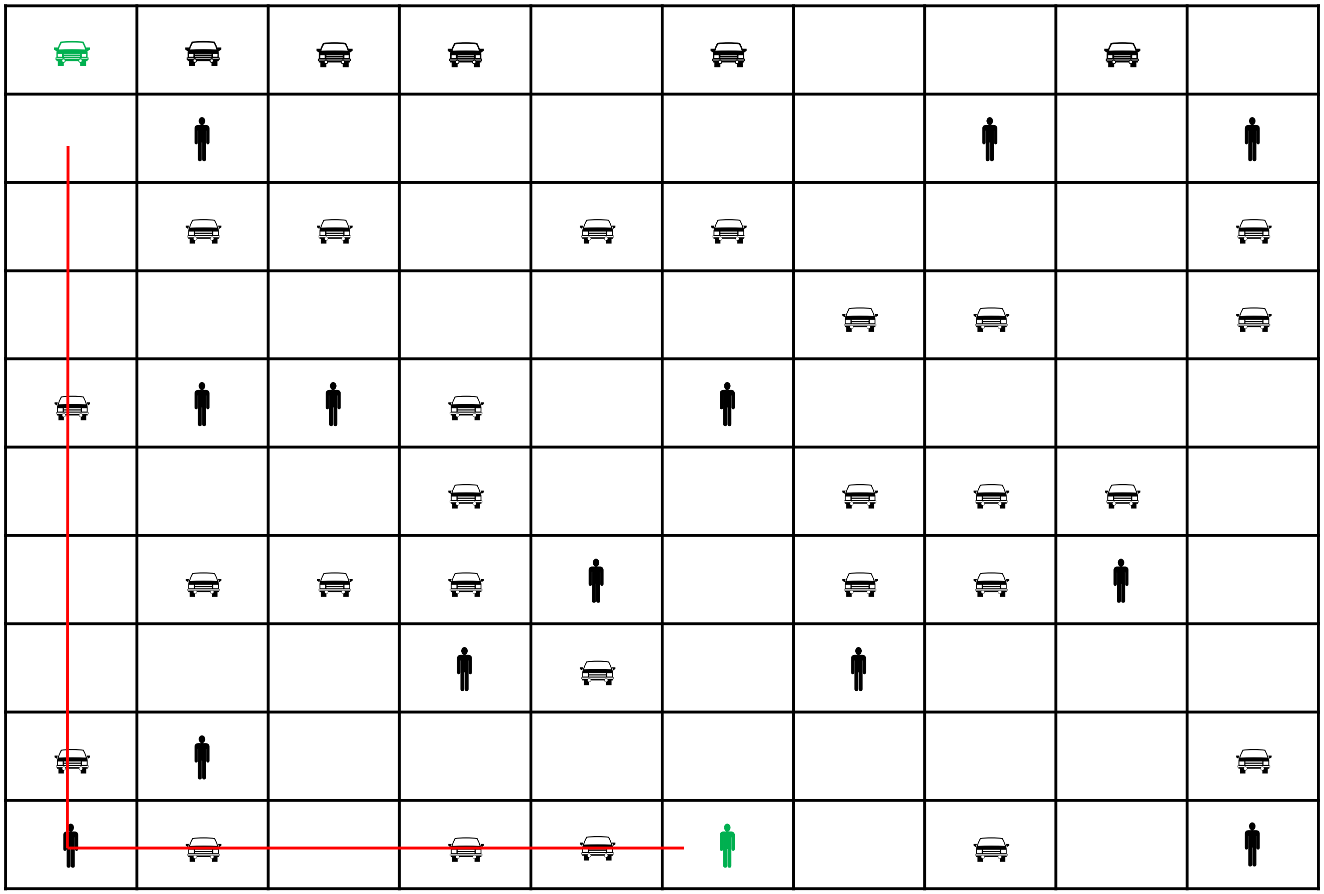
Discount

$$Q(\text{state}, \text{action}) = (1 - \text{alpha}) * Q(\text{state}, \text{action}) + \text{alpha} * (\text{reward} + \text{gamma} * Q(\text{next state}, \text{all actions}))$$

Current value

Maximum value of all actions on next state

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

Action:

Reward:

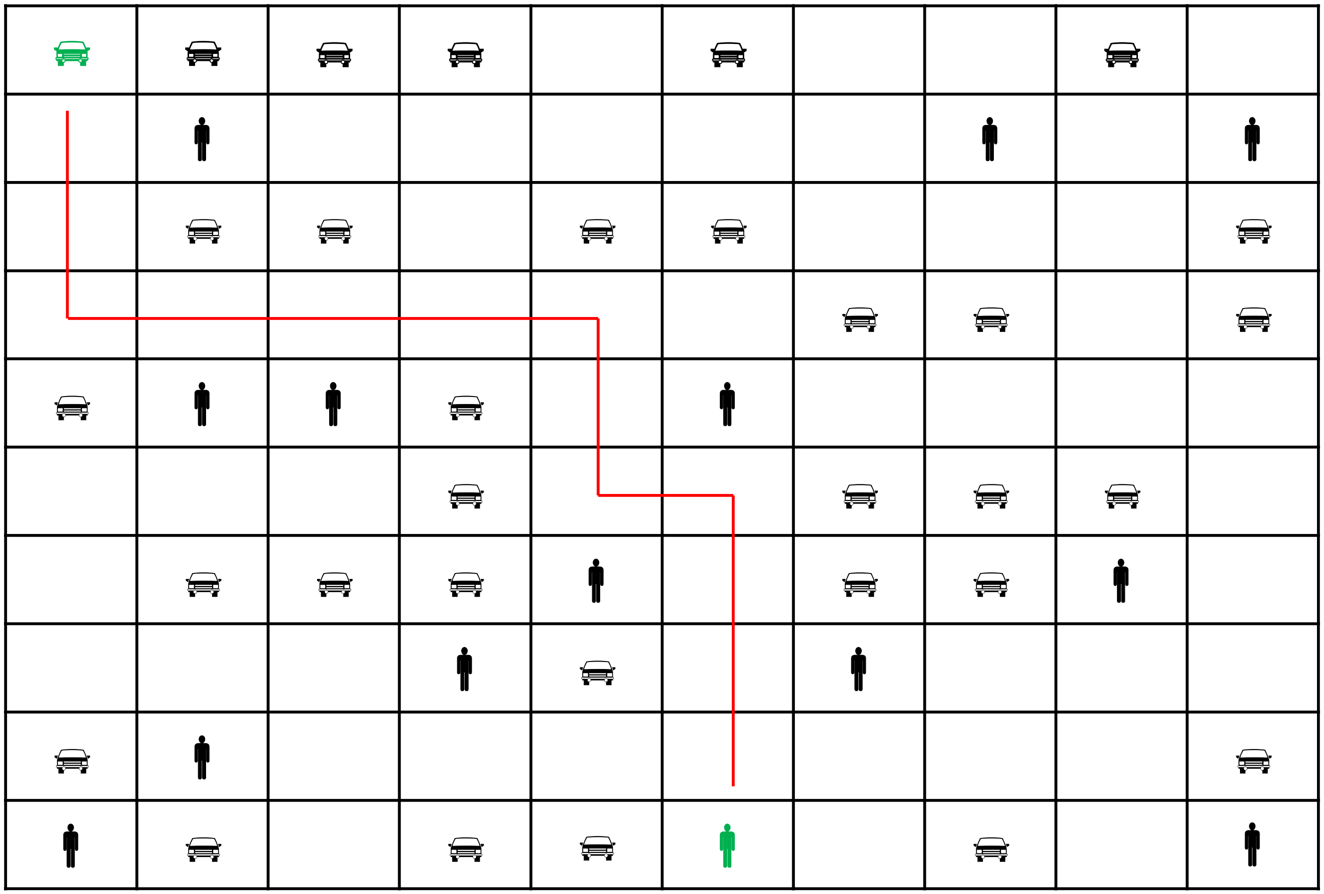
Q-table value:

Learning rate　　　　Discount

$$Q(\text{state, action}) = (1 - \text{alpha}) * Q(\text{state, action}) + \text{alpha} * (\text{reward} + \text{gamma} * Q(\text{next state, all actions}))$$

Current value　　　　Maximum value of all actions on next state

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:

Move into car: -100

Move into pedestrian: -1000

Move into empty space: 100

Move into goal: 500

Action: **Reward:**

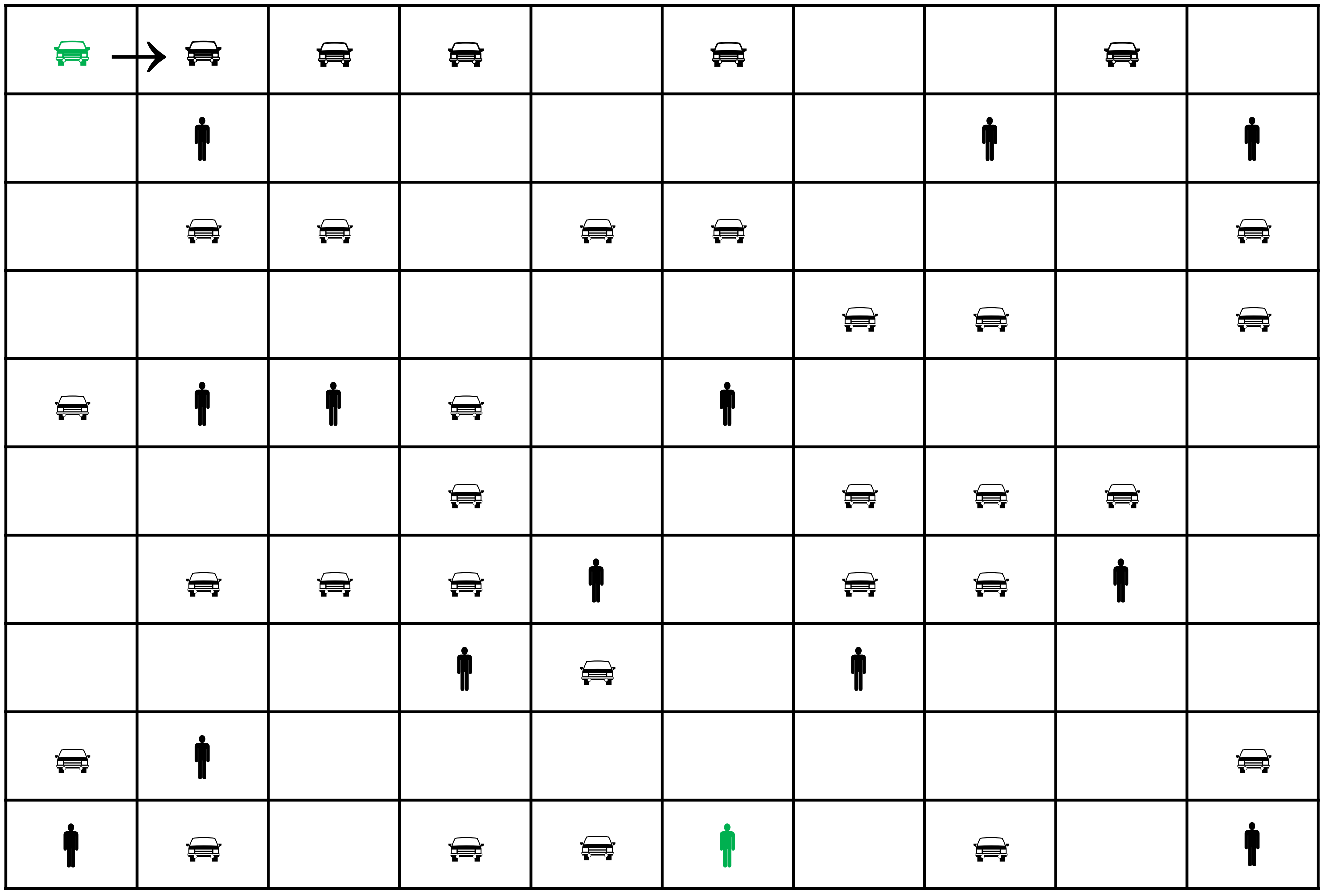
Q-table value:

Learning rate Discount

$$Q(\text{state}, \text{action}) = (1 - \text{alpha}) * Q(\text{state}, \text{action}) + \text{alpha} * (\text{reward} + \text{gamma} * Q(\text{next state}, \text{all actions}))$$

Current value Maximum value of all actions on next state

Q-Learning Algorithm



Q-table

States		Actions ↑	Actions ↓	Actions →	Actions ←
	1	0	0	0	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

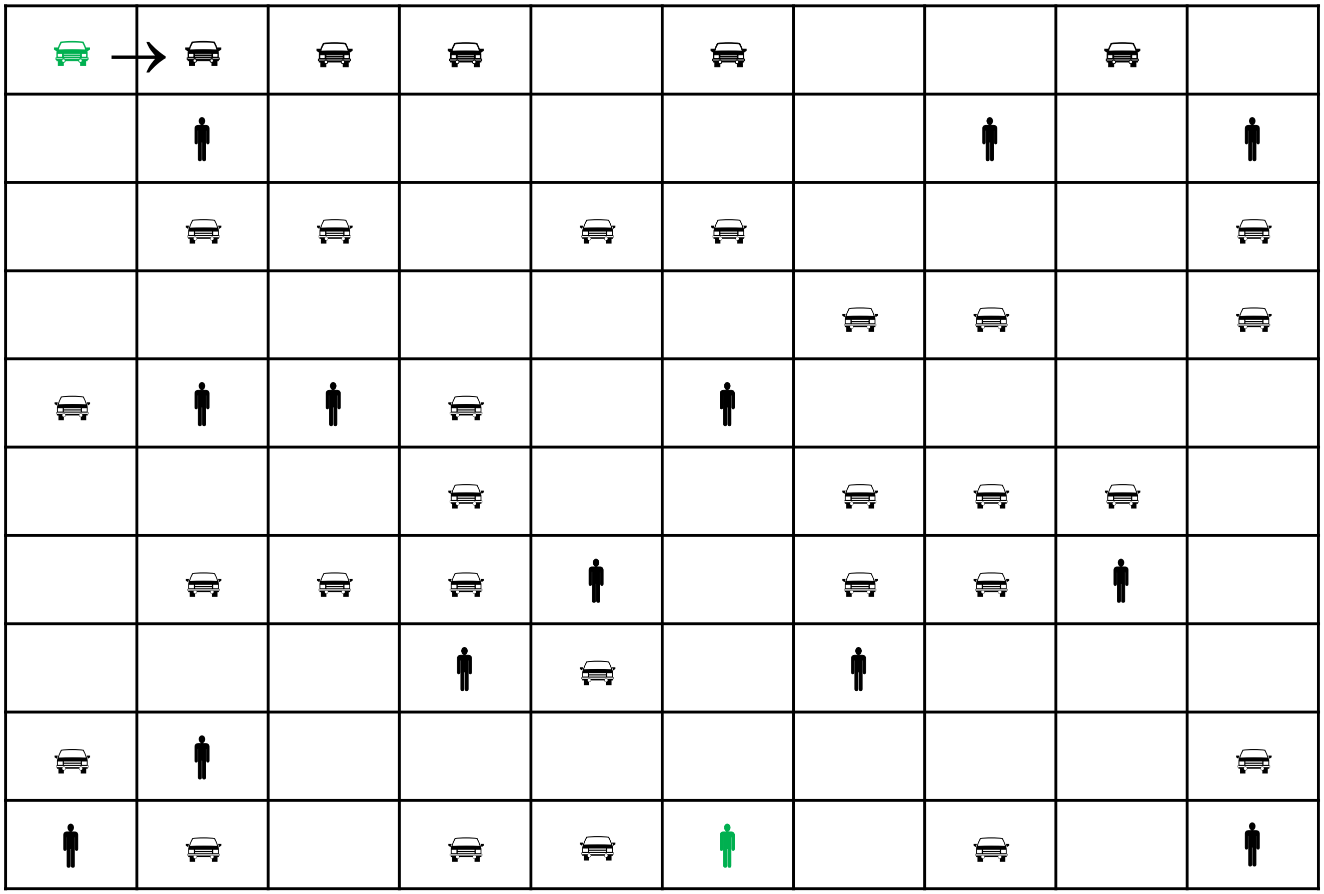
Action: →

Reward: -100

Q-table value:

$$Q(1, \text{east}) = (1 - 0.1) * 0 + 0.1 * (-100 + 0.6 * \text{max of } Q(2, \text{all actions}))$$

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	-10	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

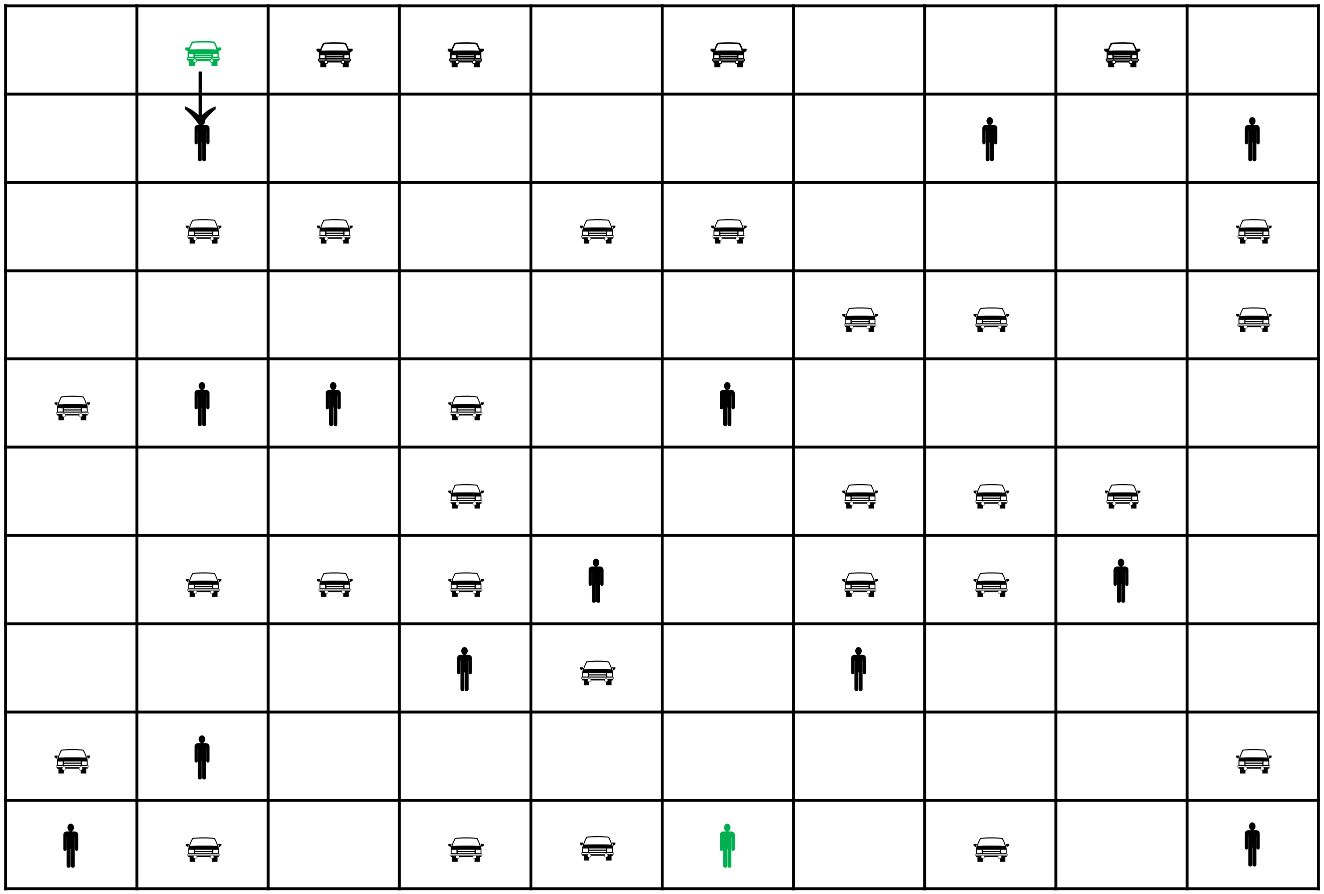
Action: →

Reward: -100

Q-table value:

$$Q(1, \text{east}) = (1 - 0.1) * 0 + 0.1 * (-100 + 0.6 * 0) = -10$$

Q-Learning Algorithm



Q-table

		Actions			
		↑	↓	→	←
States	1	0	0	-10	0
	2	0	0	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:
Move into car: -100
Move into pedestrian: -1000
Move into empty space: 100
Move into goal: 500

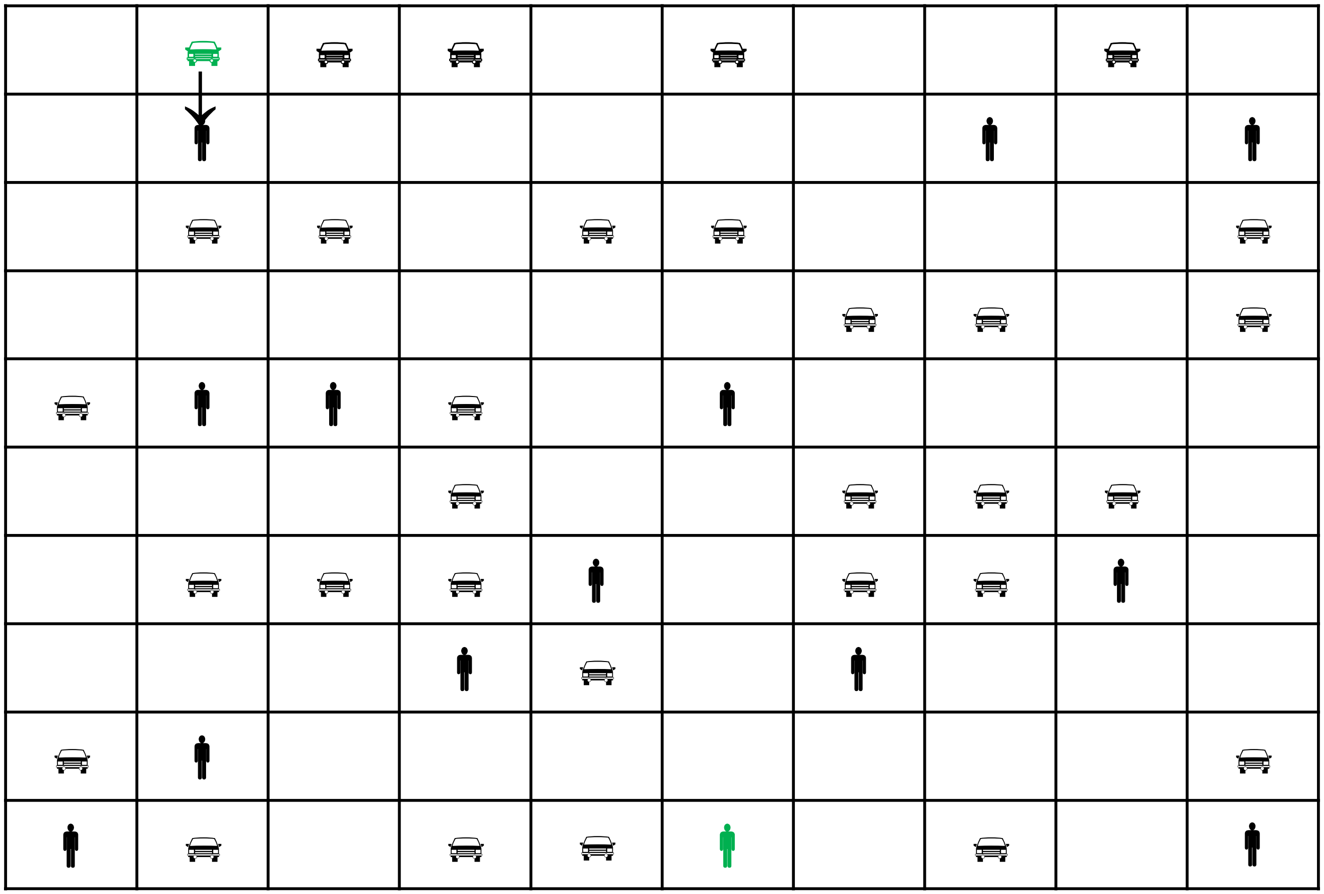
Action: →

Reward: -1000

Q-table value:

$$Q(2, \text{south}) = (1 - 0.1) * 0 + 0.1 * (-1000 + 0.6 * \text{max of } Q(3, \text{all actions}))$$

Q-Learning Algorithm



Q-table		Actions			
		↑	↓	→	←
States	1	0	0	-10	0
	2	0	-100	0	0
	…	…	…	…	…
	n	0	0	0	0

Rewards:

Move into car: -100

Move into pedestrian: -1000

Move into empty space: 100

Move into goal: 500

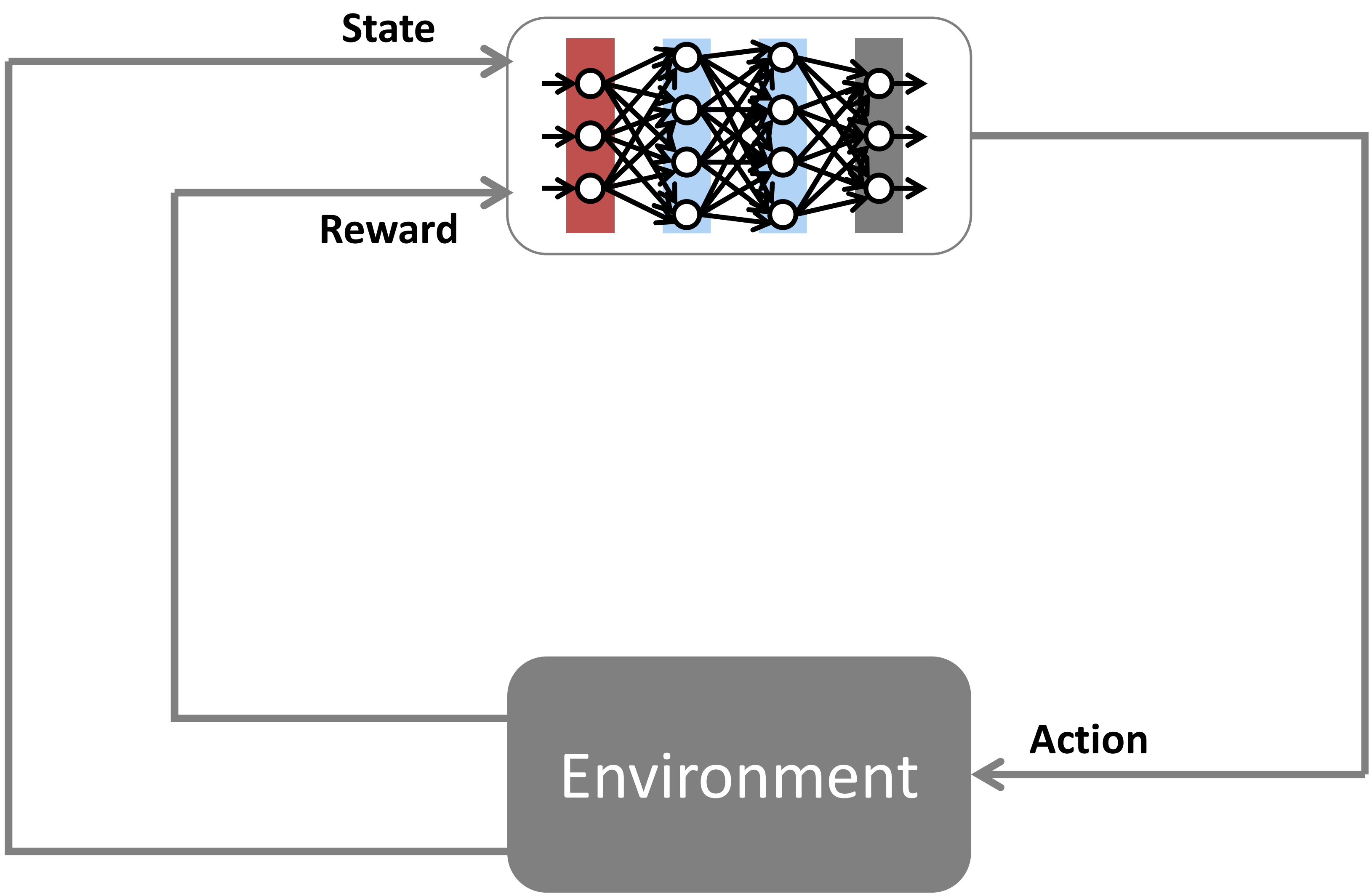
Action: →

Reward: -1000

Q-table value:

$$Q(2, \text{south}) = (1 - 0.1) * 0 + 0.1 * (-1000 + 0.6 * 0) = -100$$

Deep Reinforcement Learning



RL: Agents and Environments