Learning requires consequences

Reinforcement Learning Foundations

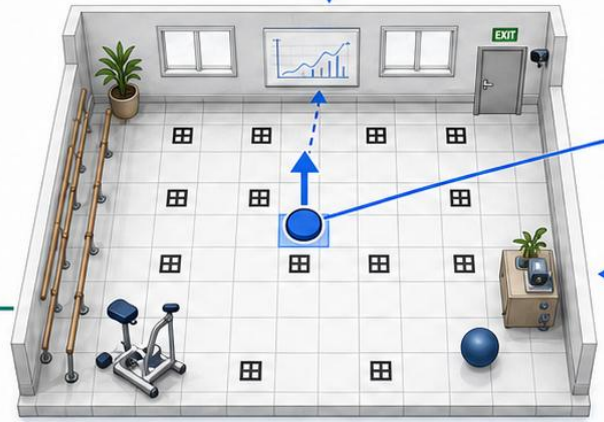
Learning from Consequences in Virtual Worlds

Environment • reward • update • evaluation

A scene becomes an experiment when actions change what happens next

environment
actions change the state and what happens next

reward
consequences are measured as numerical feedback



$t+1$ $t+2$ $t+3$ ...

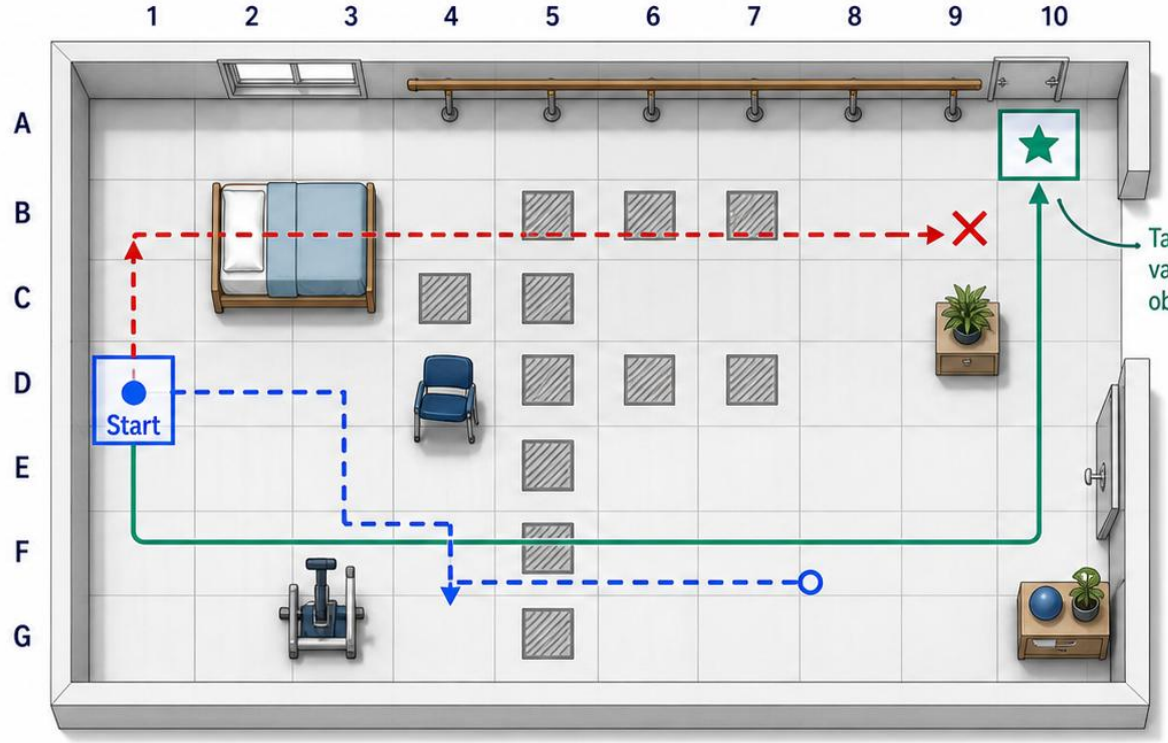
update
experience changes the estimate or policy

evaluation
does the resulting behaviour really work beyond one run?

z y x

Environment • reward • update • evaluation | A scene becomes an experiment when actions change what happens next

Which viewpoint should the agent choose?



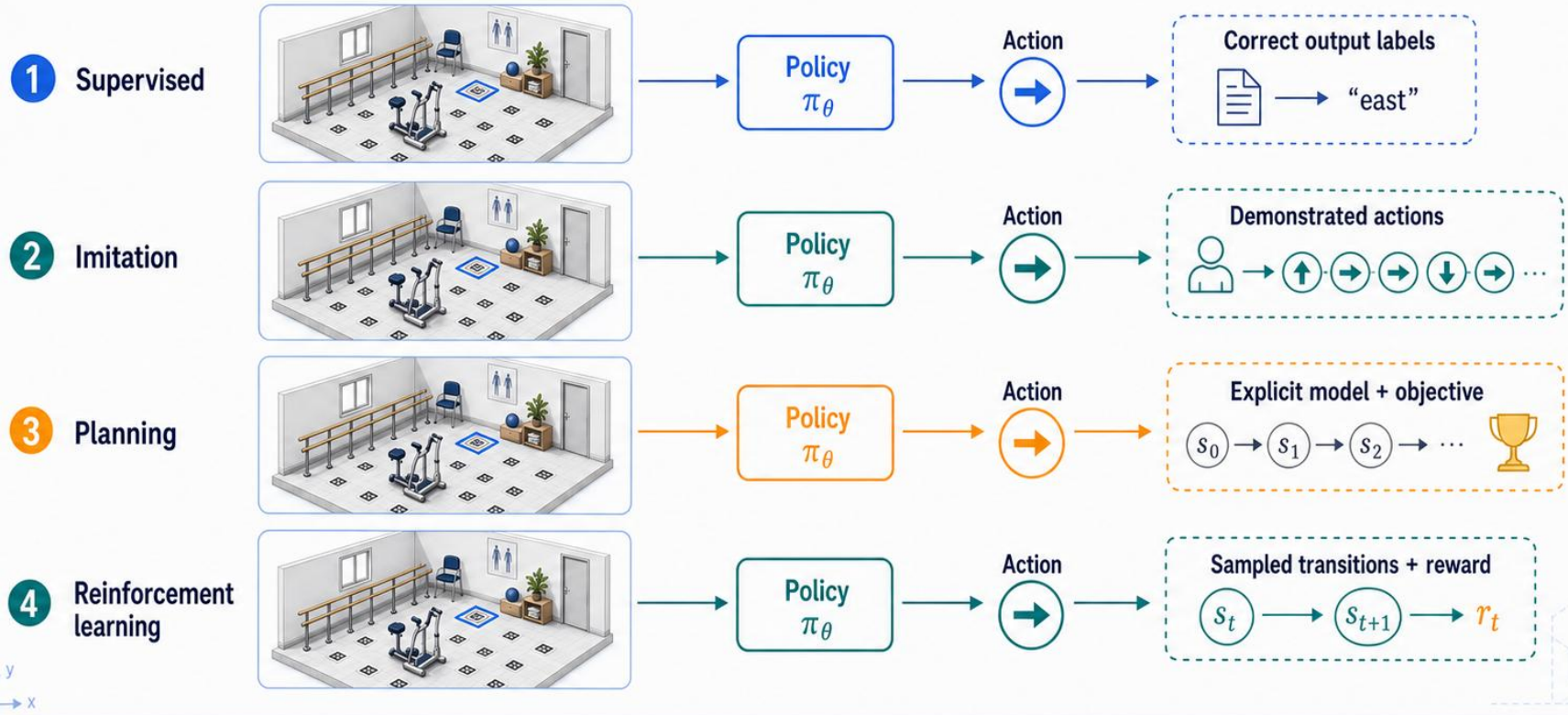
- Start: doorway
- Target: valid handrail observation point *(pedagogical viewpoint proxy)*
- Obstacles: bed, chair, equipment
- Costs: motion and limited steps

- Short but blocked ✗
- Long but safe ✓
- Incomplete ○

y
x

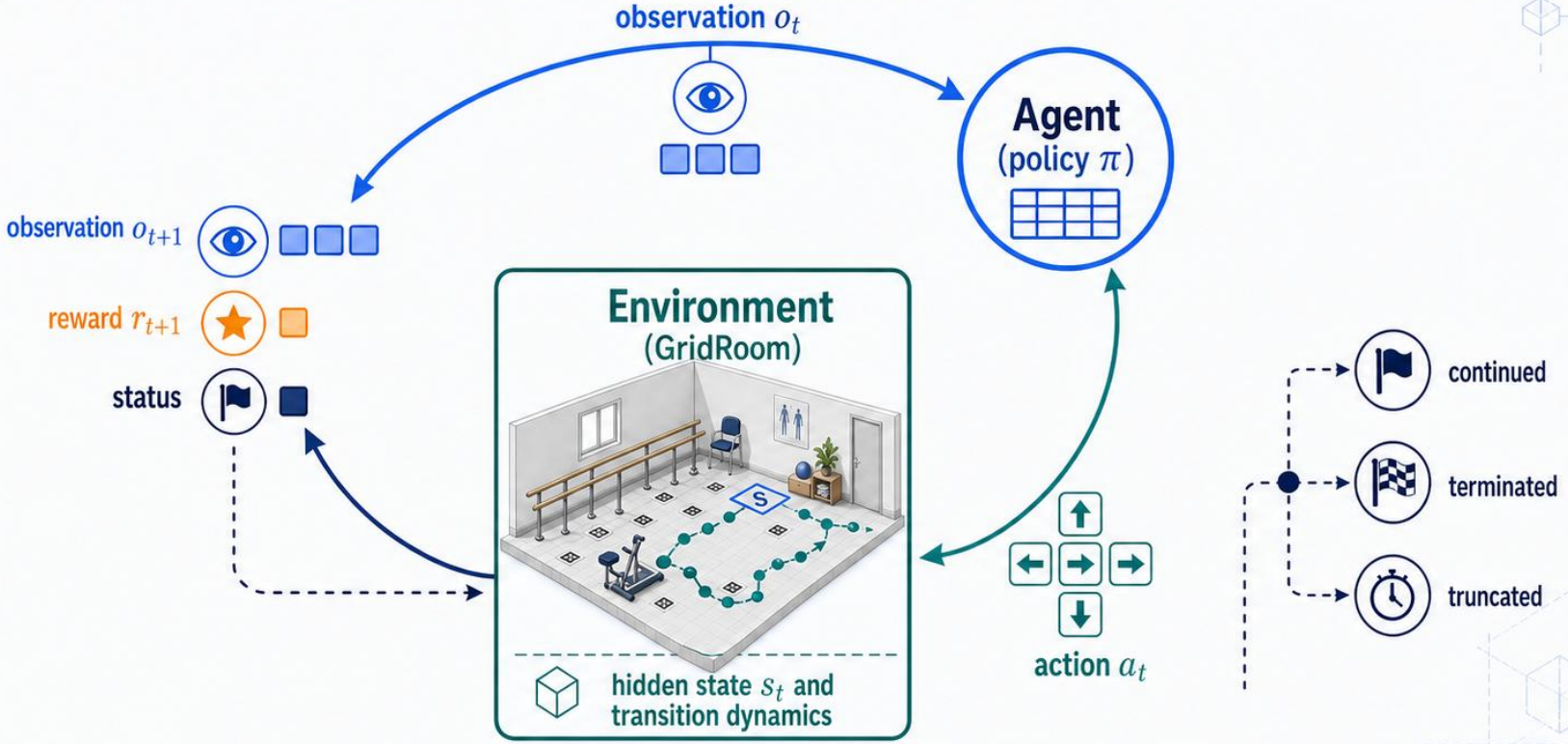
What must be specified before this can be learned?

Feedback changes the learning problem



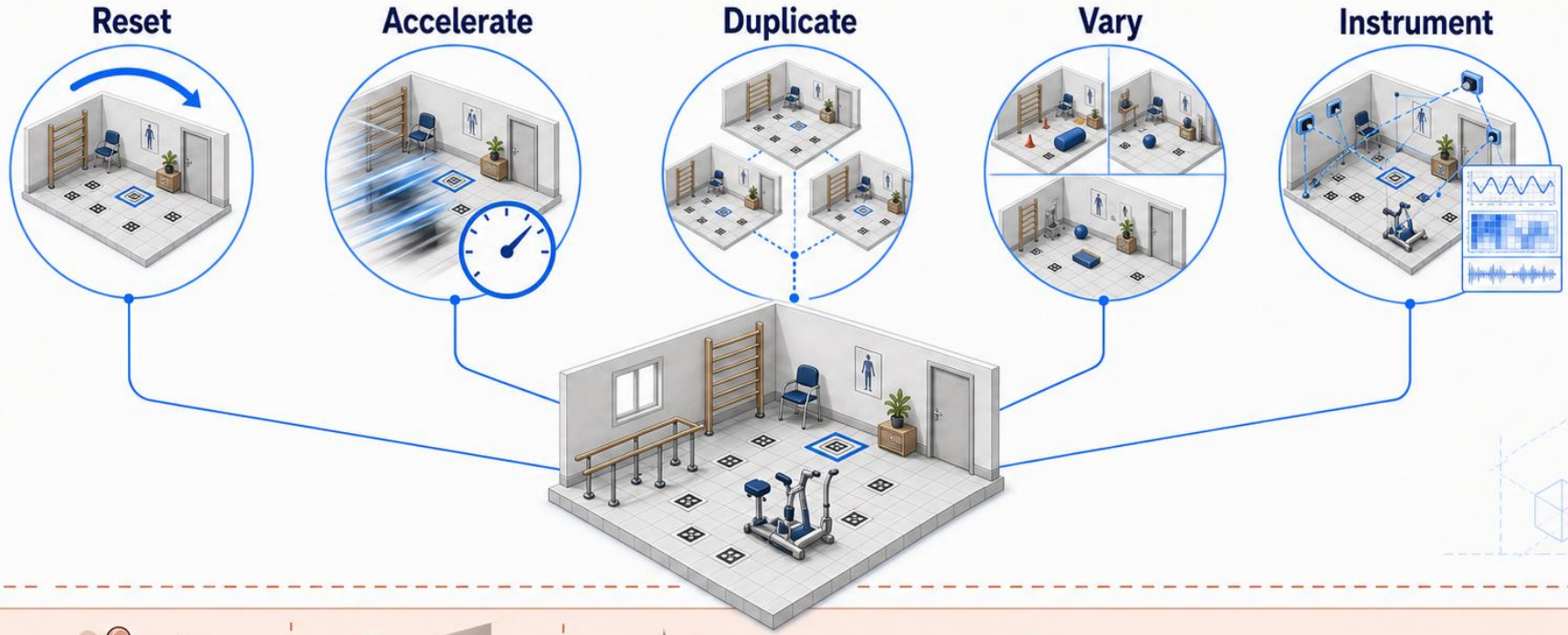
Ask where evidence for a better decision rule comes from

The agent–environment loop



Action changes the distribution of future experience

Why virtual worlds are RL instruments

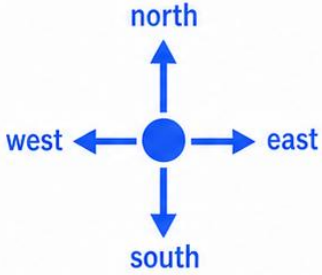




Efficiently optimizing the wrong experiment is still failure

The GridRoom task

	1	2	3	4	5	6	7
1	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	G (1,7)
2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
3	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
4	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
5	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
6	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
7	S (7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)



- finite grid; fixed layout
- actions: north • east • south • west
- blocked cells create self-transitions
- target cell ends the task
- external step budget limits classroom runs

An MDP makes the contract explicit

$\mathcal{S}$: possible states

$\mathcal{A}$: possible actions



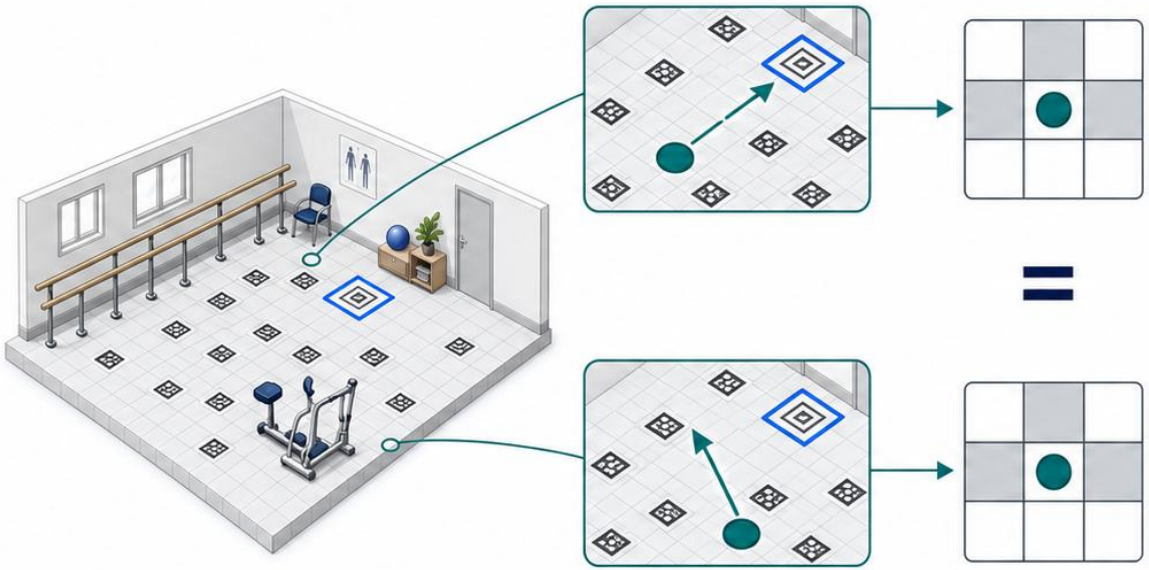
$P(s' \mid s, a)$: transition distribution

$\mathcal{R}$: reward distribution or function

γ: delayed-consequence weighting

initial state and episode boundary rules

State is not observation



State: sufficient environment information for the next-outcome model

Observation: information supplied to the policy

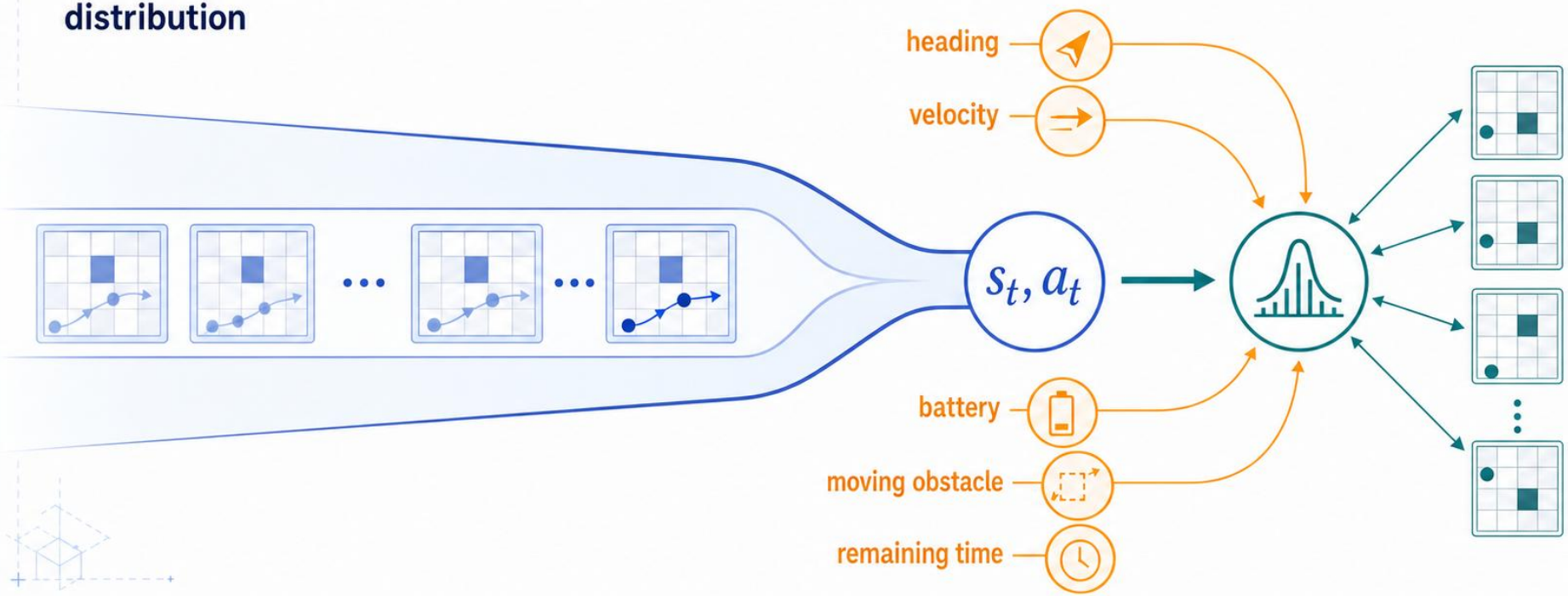
fully observable: $o_t = s_t$ is possible

partial observation: different states can look identical

memory may help; information cannot be invented

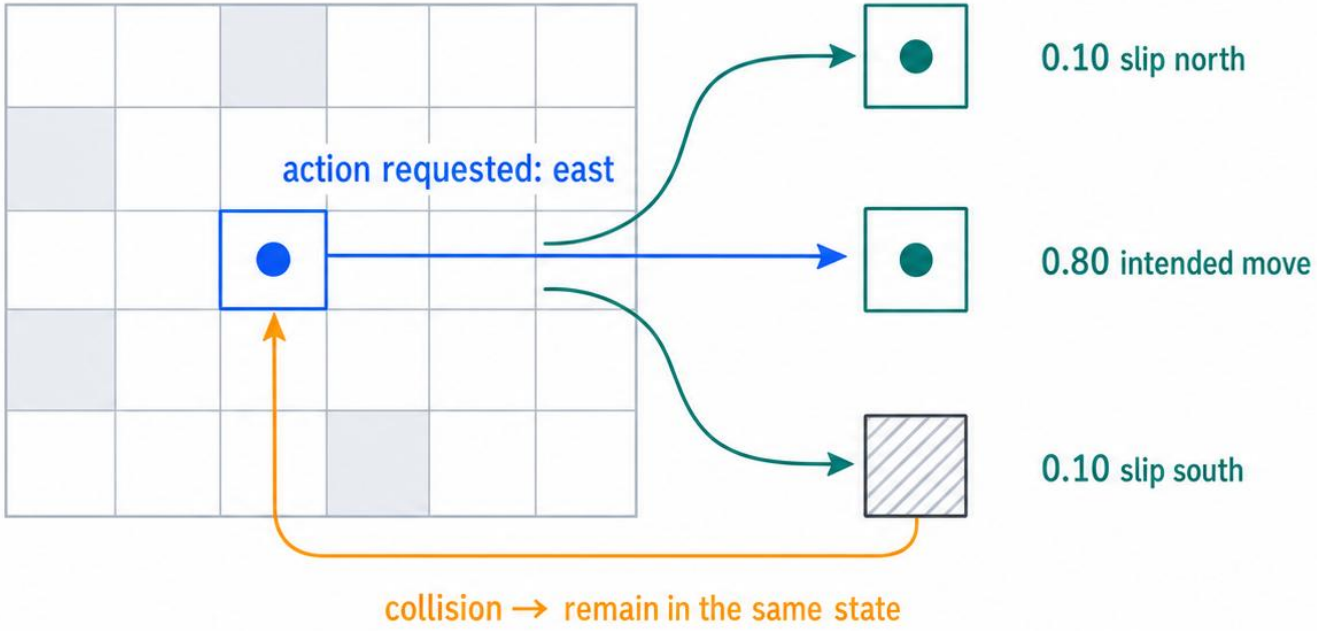
The Markov question

given s_t, a_t, additional history should not change the next-outcome distribution



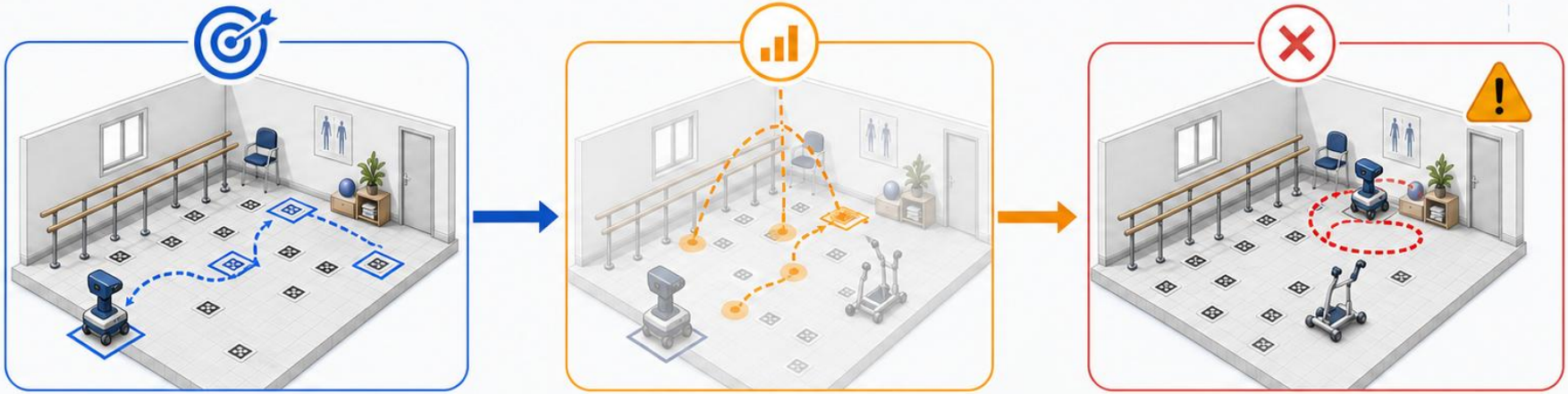
omitted variables leak history dependence → representation + task determine the answer

Actions meet uncertain transitions



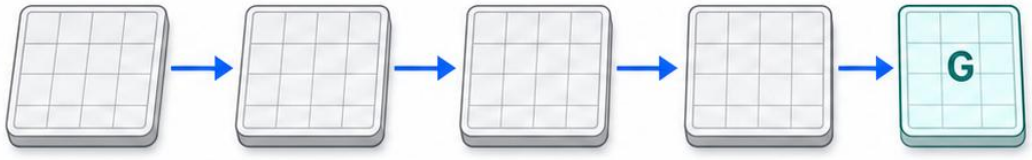
An action is not a guaranteed displacement

Reward is a signal, not the intention

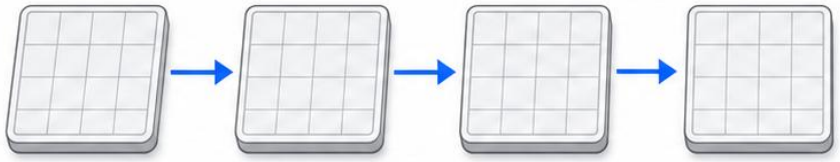


- intended task: reach a useful viewpoint safely and efficiently
- exposed signal: numeric reward after transitions
- proxy can be sparse, shaped, or exploitable
- constraints and evaluation metrics can remain separate
- The learner optimizes what the environment returns

Episodes need explicit boundaries



terminated: terminal state defined by the task



40-step external limit

truncated: external condition stops the sampled episode

reset after either status

bootstrap zero only for true task termination

Activity: specify this room MDP

State / observation

Reward



Actions / invalid-action rule

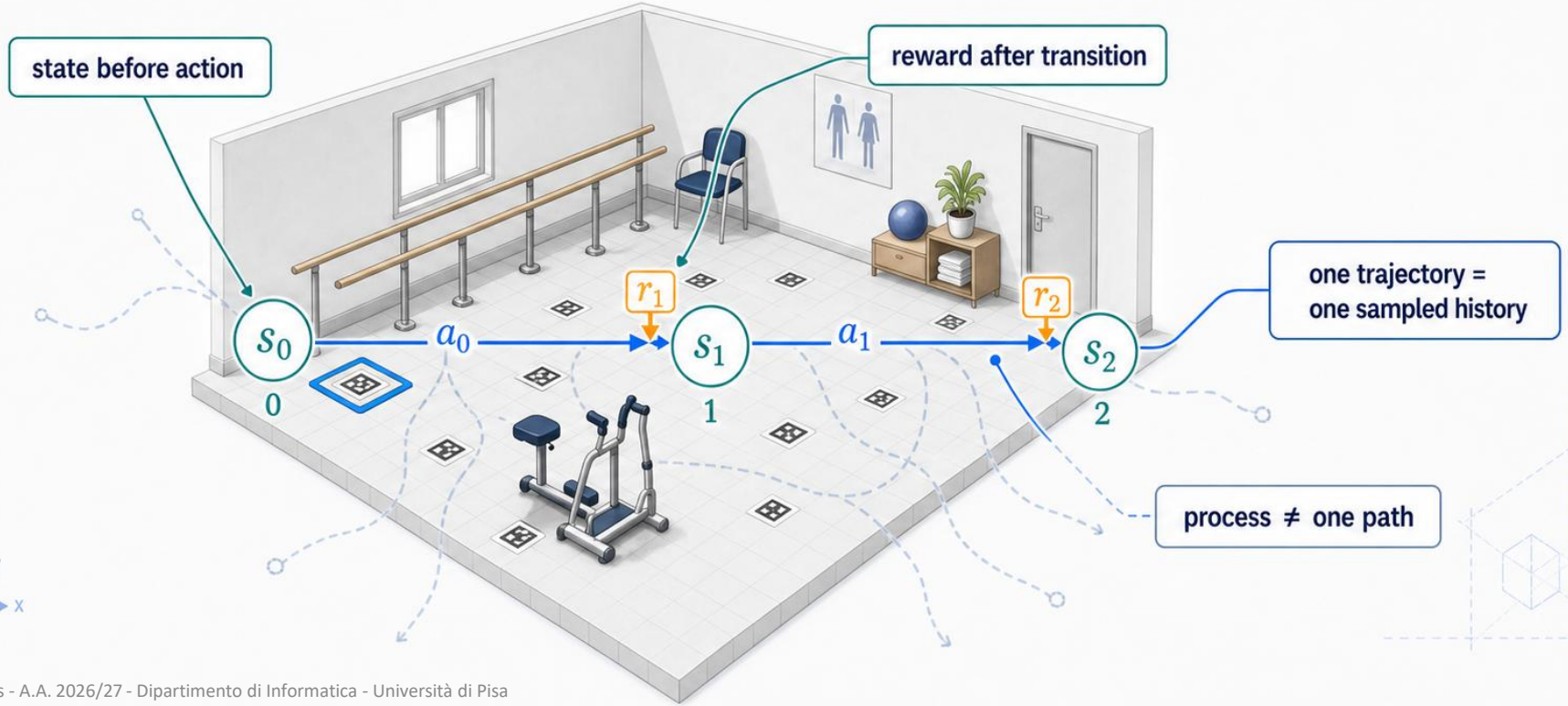
Terminated / truncated

Transition uncertainty

One assumption you would test

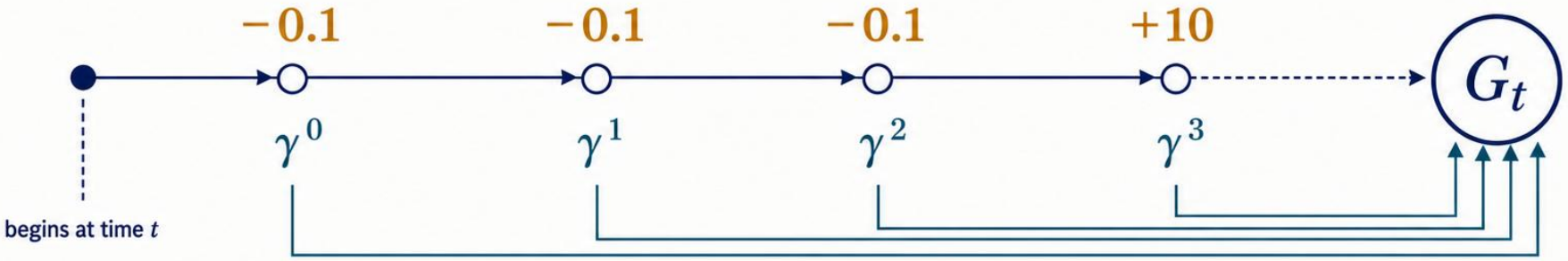
A trajectory records consequences

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots)$$



Return aggregates delayed reward

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k \, r_{t+k+1}$$



Σ sums future transition rewards

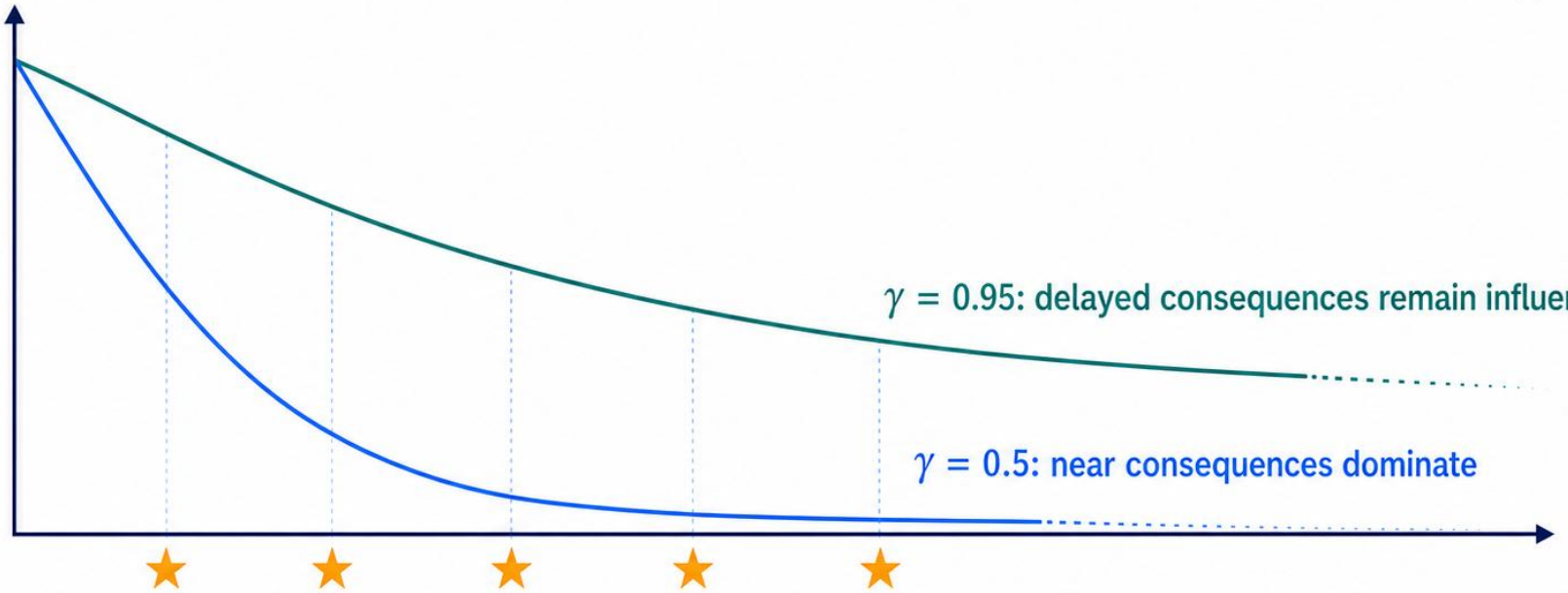
γ gamma weights delay



random trajectory → random return

Sutton and Barto, Chapter 3

Discount changes the effective horizon



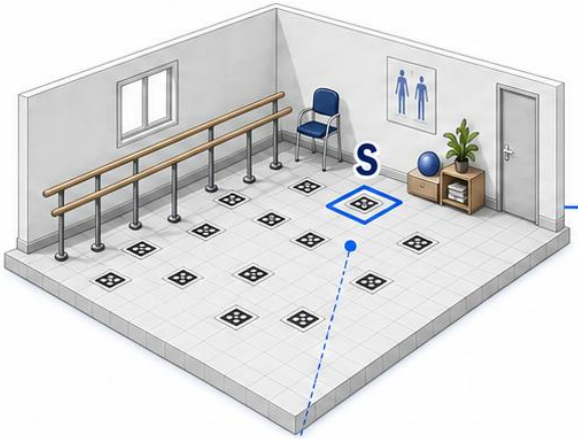
γ shapes the objective

Not interchangeable

α shapes update size

A policy maps situations to actions

$$\pi(a \mid s) = \Pr(A_t = a \mid S_t = s)$$



input: state s —
or observation o

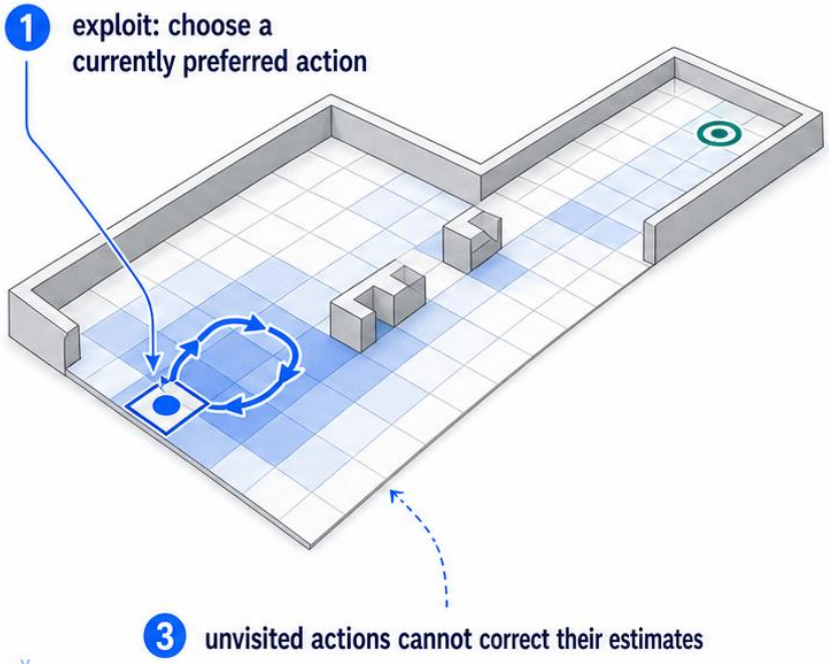
deterministic:
one selected action

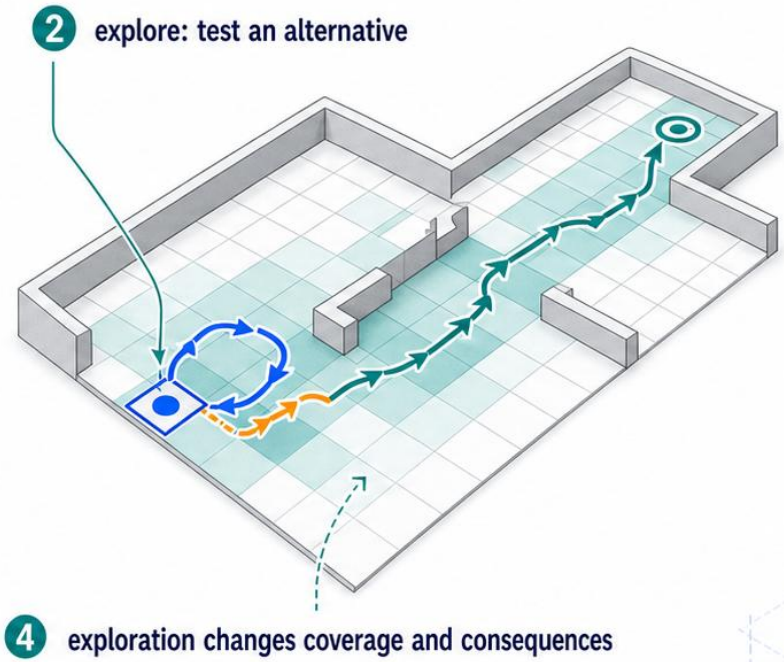
stochastic:
probability distribution
over actions

training behaviour may differ from evaluated behaviour

Sutton and Barto, Chapter 3

Exploration changes the data you see





z
y
x

5 risk and training budget still matter

Value asks: what happens from here?

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G_t \mid S_t = s\right]$$



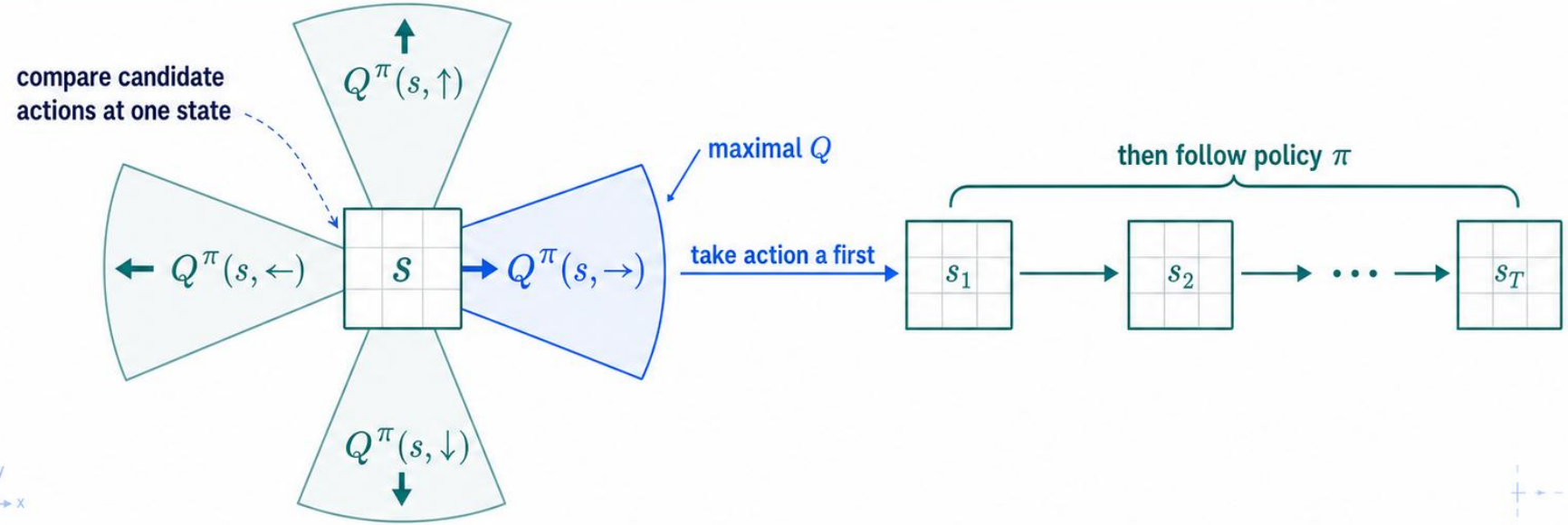


- expected return from state s
- future actions follow π
- transition randomness is averaged

→ value depends on policy, reward, dynamics, and γ

Action value asks: what if I do this?

$$Q^{\pi}(s,a) = \mathbf{E}_{\pi}\left[G_t \mid S_t = s, A_t = a\right]$$



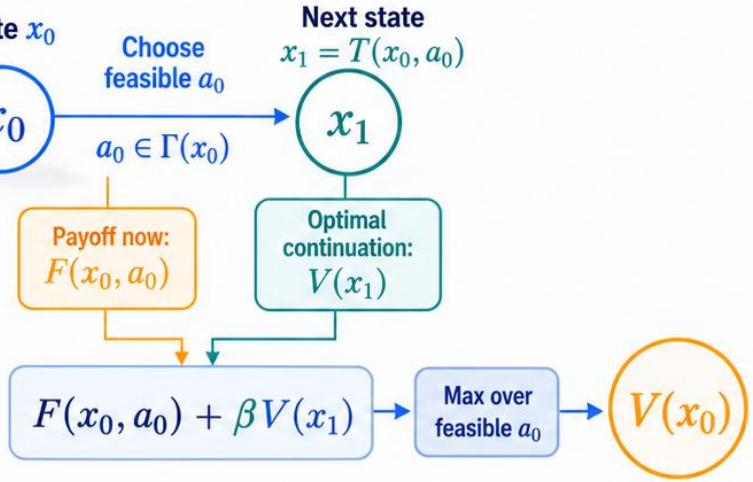
greedy action: an action with maximal Q

Bellman equation: value recurs through the next state

$$V(x_0) = \max_{a_0} \left\{ F(x_0, a_0) + \beta V(x_1) \right\}$$

$$a_0 \in \Gamma(x_0), \quad x_1 = T(x_0, a_0), \quad 0 < \beta < 1$$





x_0	:	current state
a_0	:	chosen feasible action
x_1	:	next state, $x_1 = T(x_0, a_0)$
$F(x_0, a_0)$	:	immediate payoff (undiscounted)
$V(x_1)$	:	optimal continuation value
β	:	discount factor, $0 < \beta < 1$
$\Gamma(x_0)$	:	feasible actions in state x_0
$T(x_0, a_0)$	:	deterministic state transition

1 **Optimal value = best combined payoff + discounted continuation**

$V(x_0)$ is the largest value of $F(x_0, a_0) + \beta V(x_1)$ over feasible actions $a_0 \in \Gamma(x_0)$.

2 **The same value function describes the remaining decision problem.**

Course notation: $x \leftrightarrow s$ and $\beta \leftrightarrow \gamma$.
Here V is the optimal value.

3 **Deterministic optimal-value form.**

Next: evaluate a fixed policy with uncertain transitions.

Source: Wikipedia — Bellman equation, first recursive definition (7 Oct 2026)

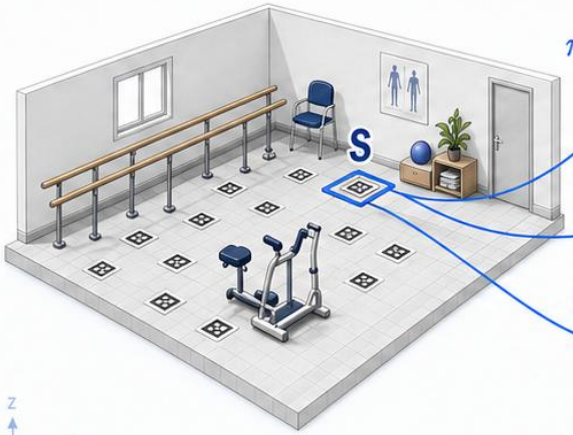
Bellman: look one step ahead

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma V^{\pi}(s') \right]$$

1 Choose action under policy

2 Sample next state + reward

3 Immediate reward + discounted successor value



$\pi(a_1|s)$ → s'_1 → $r_1 \longrightarrow r_1 + \gamma V^{\pi}(s'_1)$

$\pi(a_2|s)$ → s'_2 → $r_2 \longrightarrow r_2 + \gamma V^{\pi}(s'_2)$

$\pi(a_3|s)$ → s'_3 → $r_3 \longrightarrow r_3 + \gamma V^{\pi}(s'_3)$

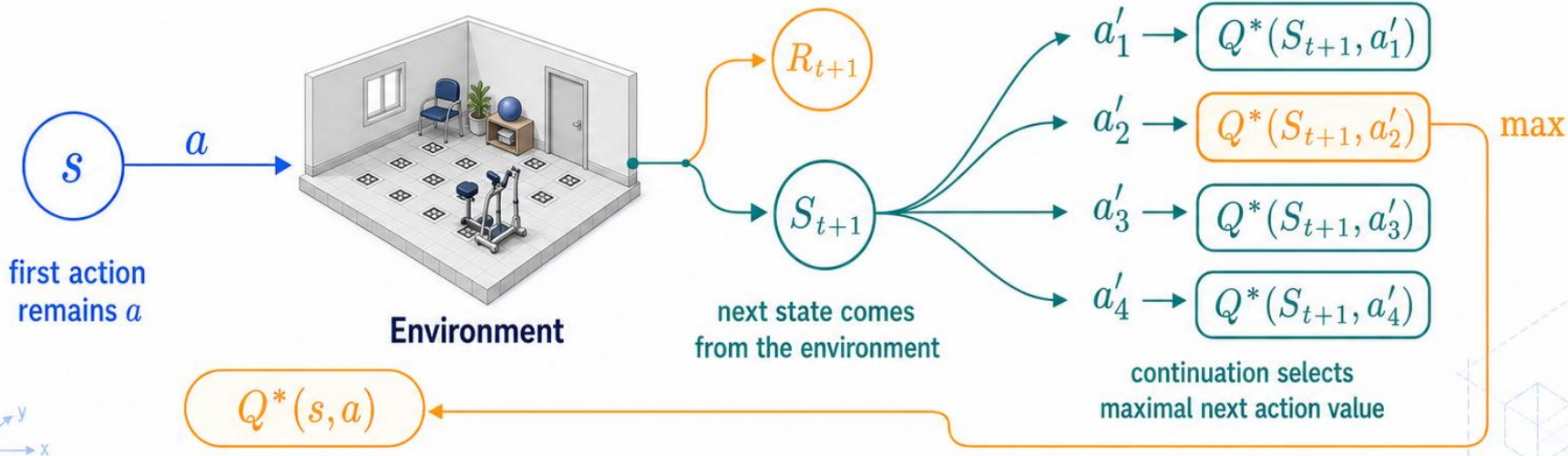
$\Sigma \rightarrow V^{\pi}(s)$

→ $p(s', r|s, a)$: transition dynamics (one-step)

Recursive consistency: value equals one-step lookahead

Optimality adds a best-next-action choice

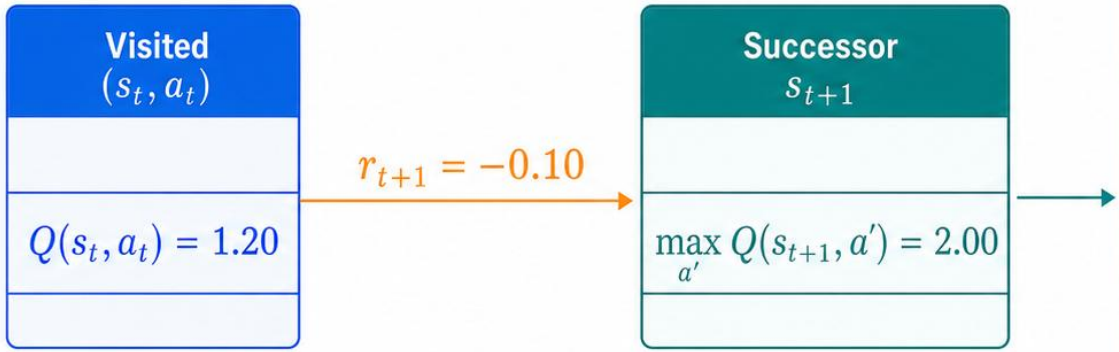
$$Q^*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a\right]$$



equation defines consistency: it is not yet an update rule

Sutton and Barto, Chapters 3 and 4.

Activity: calculate one backup



$\gamma = 0.95$ $\alpha = 0.25$

Answer reveal:

1 TARGET

$-0.10 + 0.95(2.00) = \mathbf{1.80}$

2 TD ERROR

$1.80 - 1.20 = \mathbf{0.60}$

3 UPDATED Q

$1.20 + 0.25(0.60) = \mathbf{1.35}$

TRUE TERMINAL

target = 10

no bootstrap

$\max_{a'} Q(s_{t+1}, a')$

TD learning updates before the episode ends

1 SAMPLE

$(s_t, a_t, r_{t+1}, s_{t+1})$

2 TARGET

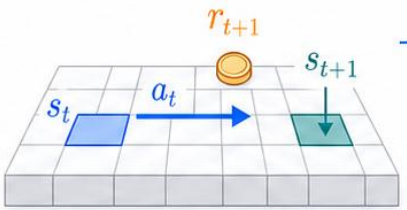
$y_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a)$

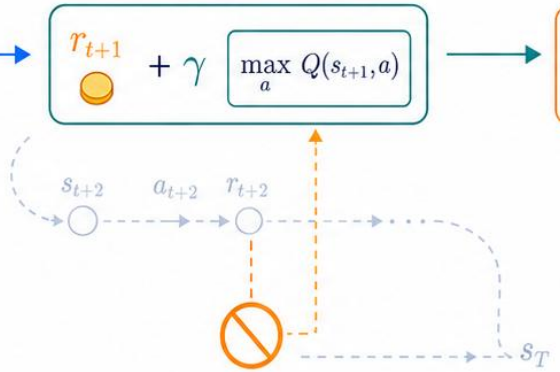
3 ERROR

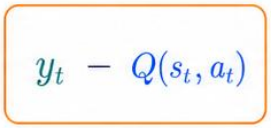
$\delta_t = y_t - Q(s_t, a_t)$

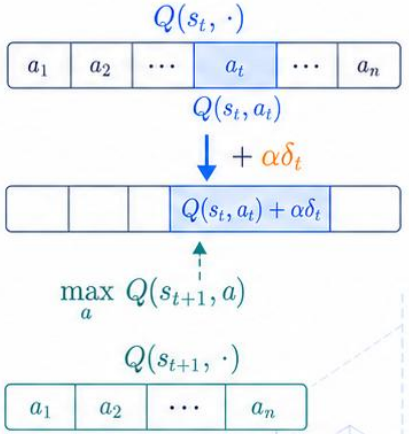
4 UPDATE

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t$











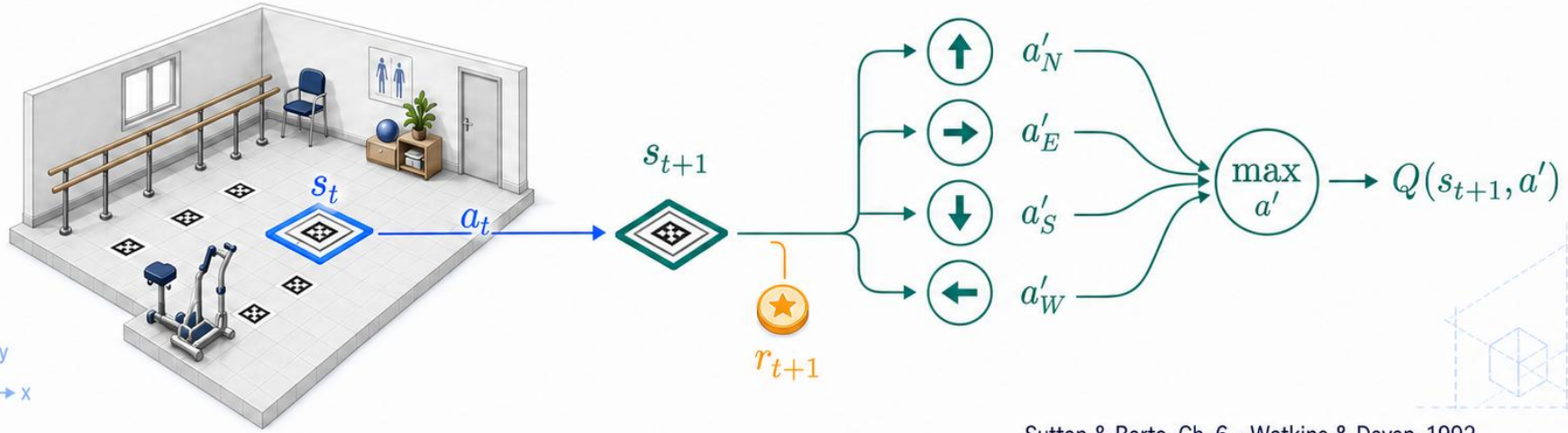
bootstrap before the full return is known

Sutton and Barto, Chapter 6

Q-learning in one equation

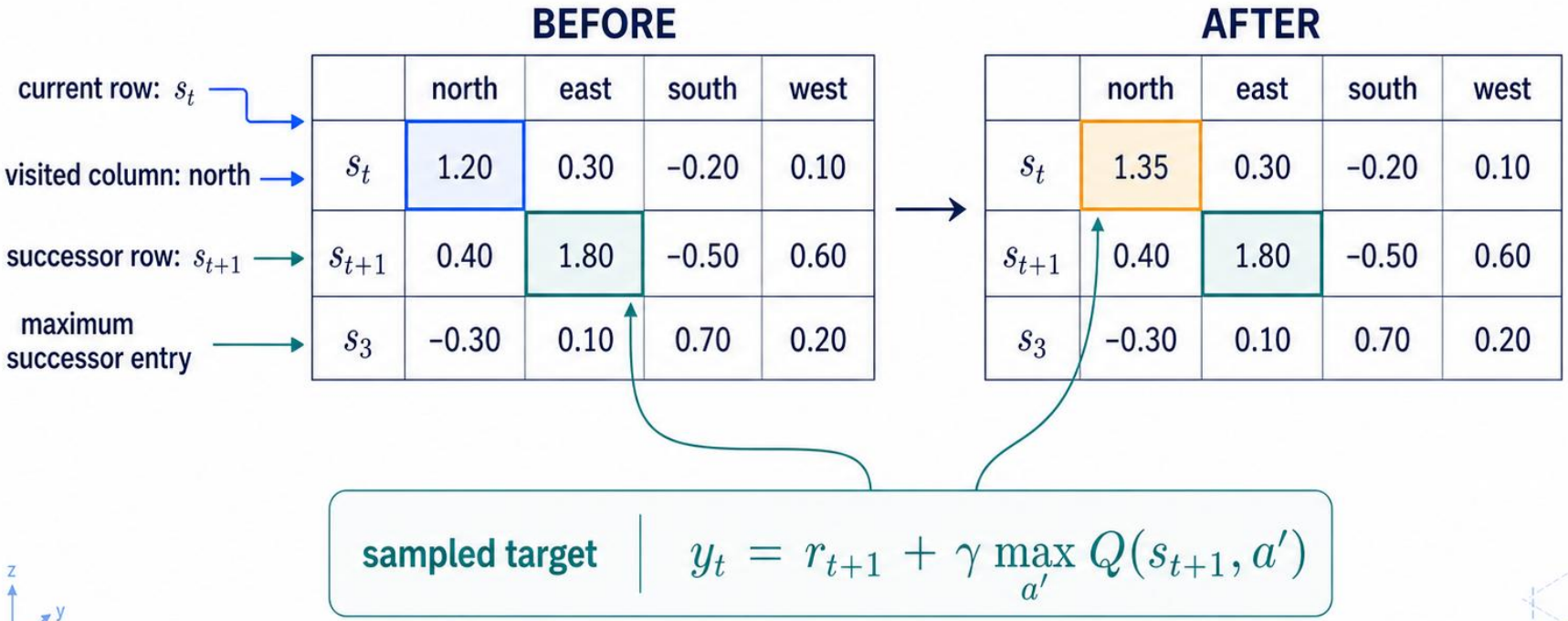
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

1. current estimate
2. sampled target
3. TD error
4. alpha-scaled correction



Sutton & Barto, Ch. 6 · Watkins & Dayan, 1992

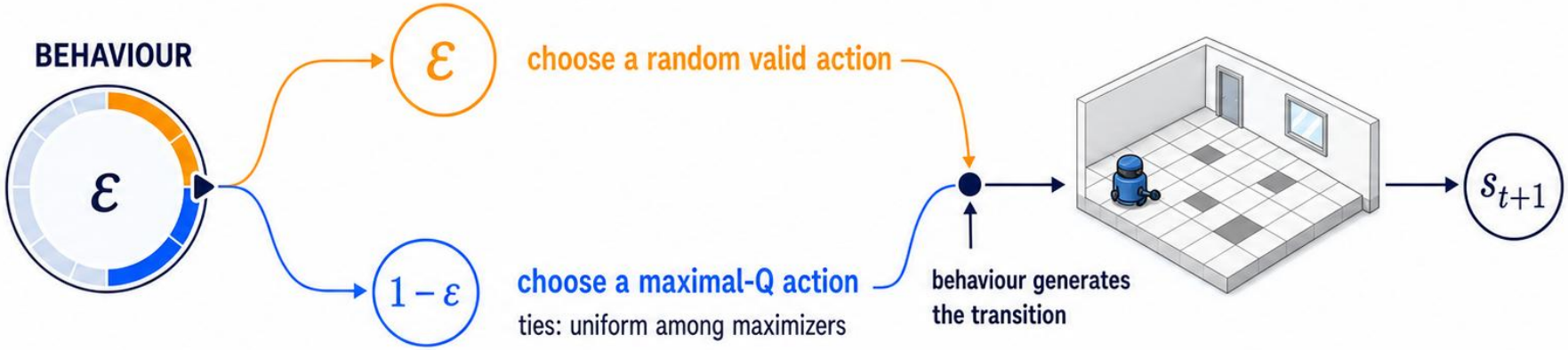
One transition changes one table cell



sampled target $\quad y_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$

only $Q(s_t, a_t)$ is updated

Epsilon-greedy separates behaviour and target



TARGET → s_{t+1} →

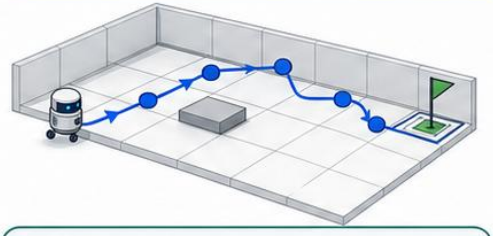
Q-learning target still uses

$$\max_{a'} Q(s_{t+1}, a')$$

record epsilon schedule and tie rule

Off-policy does not mean consequence-free

behaviour policy μ
produces experience



samples transitions

(s, a, r, s')

(s_1, a_1, r_1, s'_1)
(s_2, a_2, r_2, s'_2)
$\vdots$
(s_N, a_N, r_N, s'_N)

shared transition dataset
sampled from μ

target policy π
defines desired continuation

not used for data collection



COVERAGE

	a_1	a_2	a_3	a_4
s_1				
s_2				
s_3		×		
s_4				

ASSUMPTIONS

- relevant (s, a) pairs visited
- tabular setting
- learning rates

greedy target

$$r + \gamma \max_{a'} Q(s', a')$$

greedy continuation with respect to current Q estimates

update $Q(s, a)$

Arbitrary replay is not automatically adequate

Convergence assumptions ≠ deployment guarantee

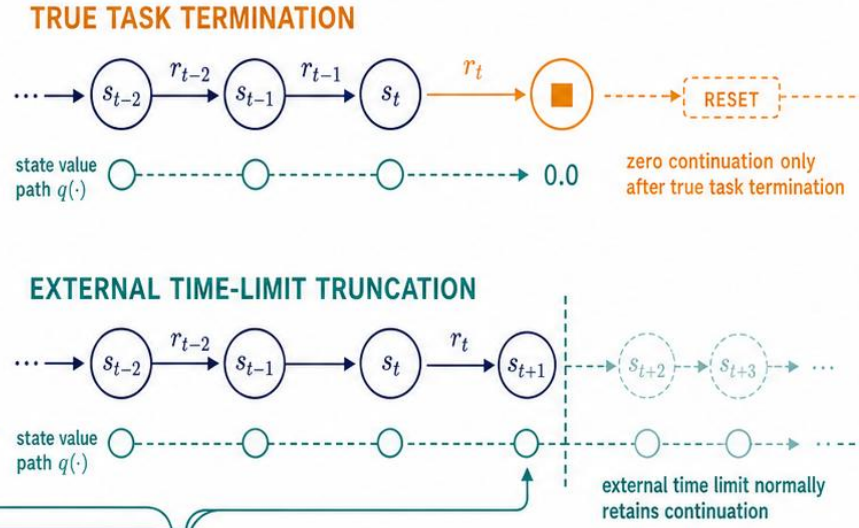
Watkins & Dayan, 1992 · Sutton & Barto, Ch. 6

Terminal and truncated steps bootstrap differently

```
ended = terminated or truncated
```

```
continuation = 0.0 if terminated
               else q[next_obs].max()
```

```
target = reward + gamma * continuation
```




```
next_obs, reward, terminated, truncated, info = env.step(action)
```

finite task horizon requires time-aware state

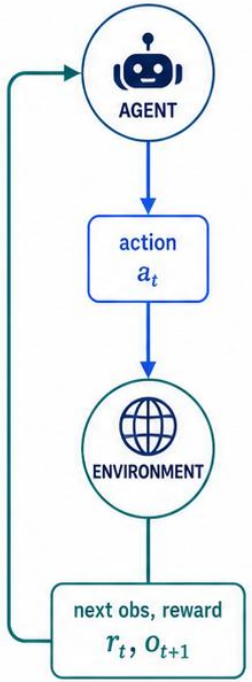
The environment loop in Python

```
1  obs, info = env.reset(seed=seed)

   while True:
2      action = epsilon_greedy(q[obs], epsilon, rng)
3      next_obs, reward, terminated, truncated, info = env.step(action)

4      continuation = 0.0 if terminated else np.max(q[next_obs])
       target = reward + gamma * continuation
5      q[obs, action] += alpha * (target - q[obs, action])

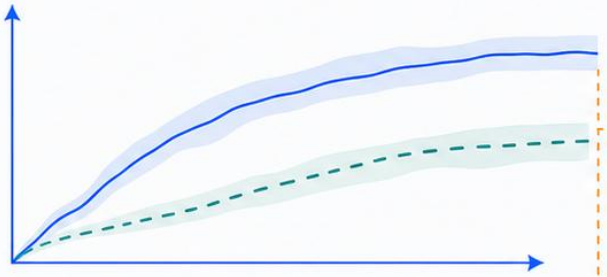
6      if terminated or truncated:
           break
       obs = next_obs
```



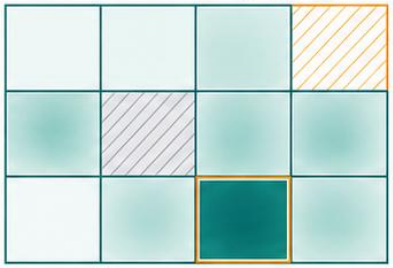
1. reset
2. behaviour
3. transition
4. target
5. update
6. boundary

The learned table becomes a room policy

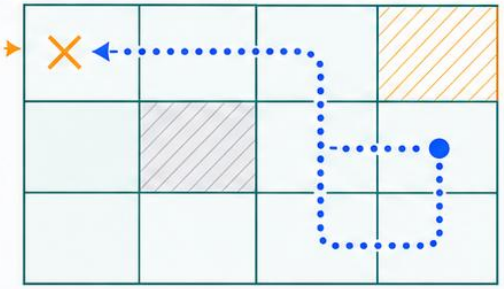
training curve: return and success versus episode



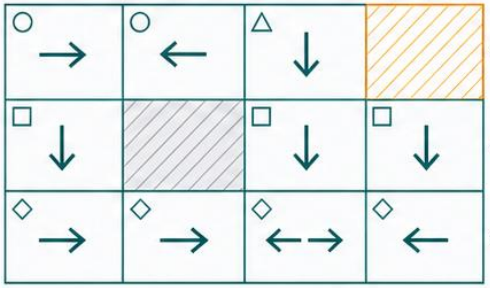
value view: $\max_a Q(s,a)$

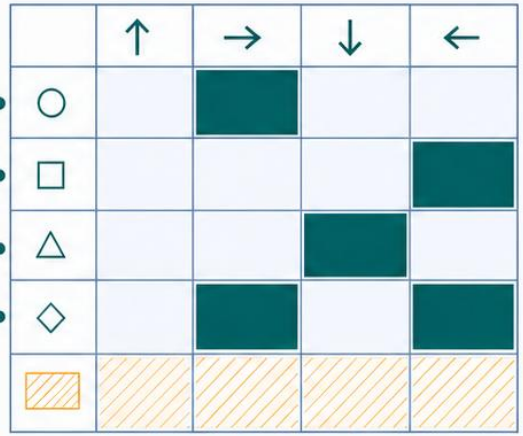


rollout view: one path with status and reward



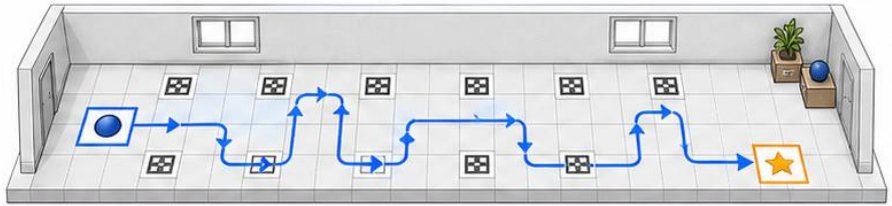
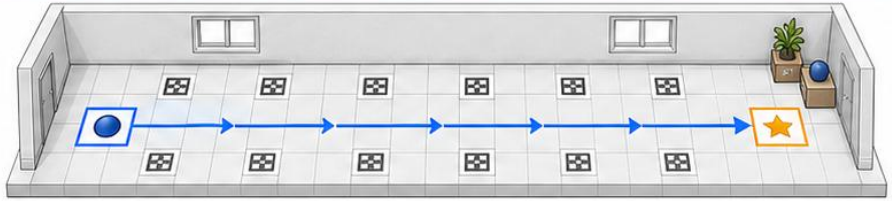
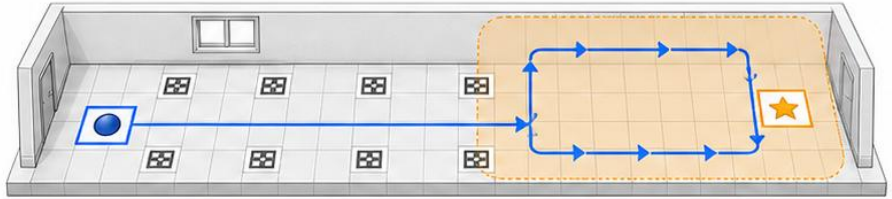
policy view: greedy action arrows





Inspect aggregate evidence and individual failure together

Activity: find the reward failure

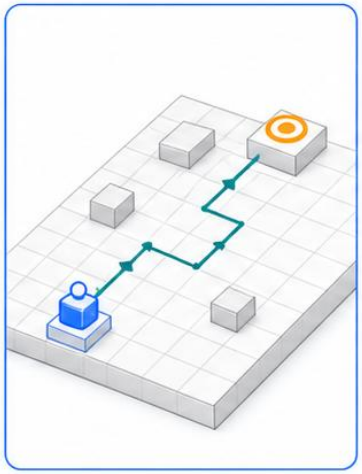
Reward		Predicted behaviour
+10 at target		?
+10 target −0.05/step		?
+1 each visible-target step		?

➡ **Redesign one of:** reward • constraint • external metric

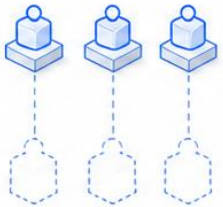
One lucky run is not evidence

“With seed 7, the policy reaches the target once. Therefore it works.”

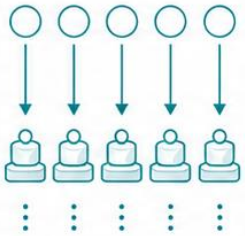
Missing:



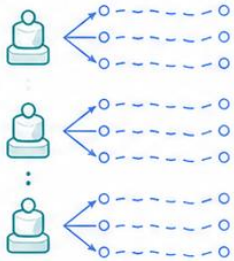
? baseline



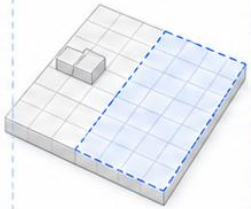
? independent training runs



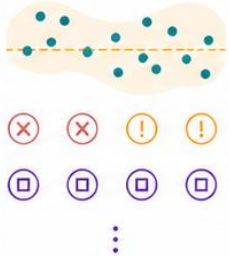
? repeated evaluation



? held-out conditions

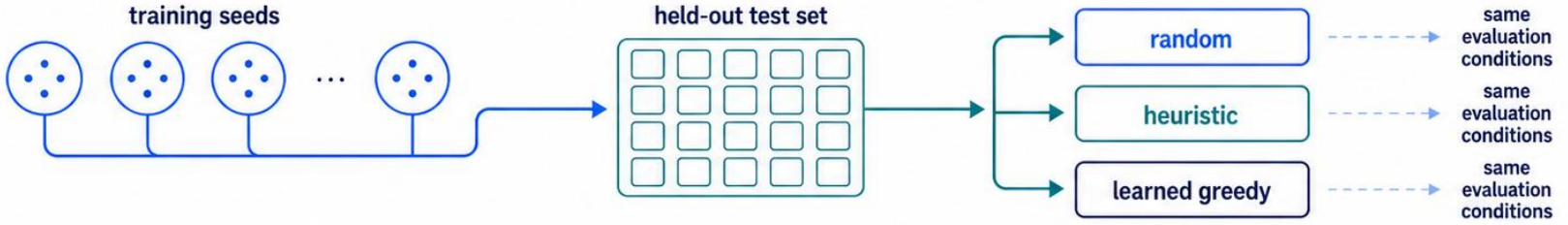


? uncertainty and failure cases



Compare random, heuristic, and learned policies

independent training seeds + repeated held-out evaluation



	random	heuristic	learned greedy
success			
steps			
return			
collisions/failures			

report variability or uncertainty

failure matrix

	random	heuristic	learned greedy
timeout	☐	☐	☐
loop	☐	☐	☐
collision attempt	☐	☐	☐
wrong target	☐	☐	☐

policies: random • heuristic • learned greedy

metrics: success • steps • return • collisions/failures

A simpler baseline can falsify the need for learning

From equations to Lesson 07

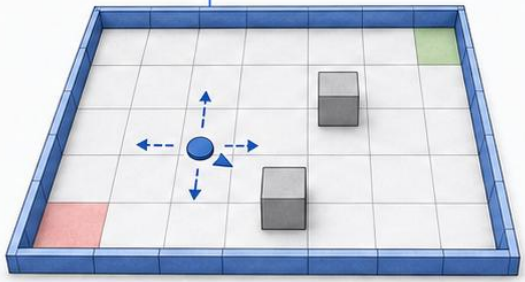
Define
GridRoom state, observation, dynamics, reward, boundaries

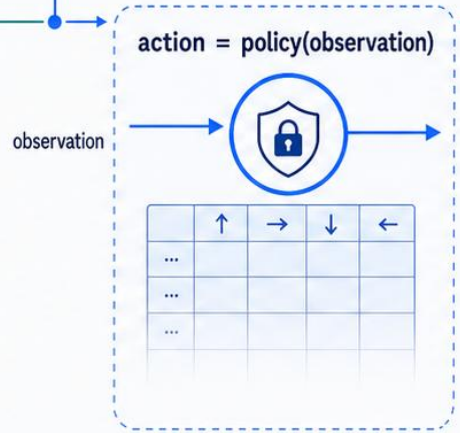
Implement
epsilon-greedy + tabular Q-learning

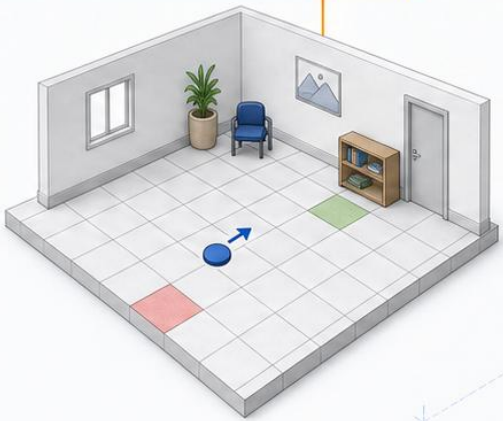
Test
transitions, terminal masking, truncation, seeds

Evaluate
random vs heuristic vs learned on held-out cases

Reuse
action = policy(observation) in Lesson 12







Exit ticket: state / observation • terminal / truncation • baseline

References: foundations and implementation

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press. Chapters 1, 3, 4, and 6.

MDP

Watkins, C. J. C. H., and Dayan, P. (1992). "Q-learning." *Machine Learning*, 8, 279–292. https://doi.org/10.1007/BF00992698

Q-learning

Farama Foundation. *Gymnasium documentation:* environment API, basic usage, migration guide, and custom-environment tutorial.

implementation semantics



References: evaluation and reproducibility

Henderson, P., et al. (2018). “Deep Reinforcement Learning That Matters.” AAAI 32(1).

Agarwal, R., et al. (2021). “Deep Reinforcement Learning at the Edge of the Statistical Precipice.” NeurIPS 34.

Stable-Baselines3. “Reinforcement Learning Tips and Tricks.”

Course-generated evidence: code version • configuration • seeds • metrics