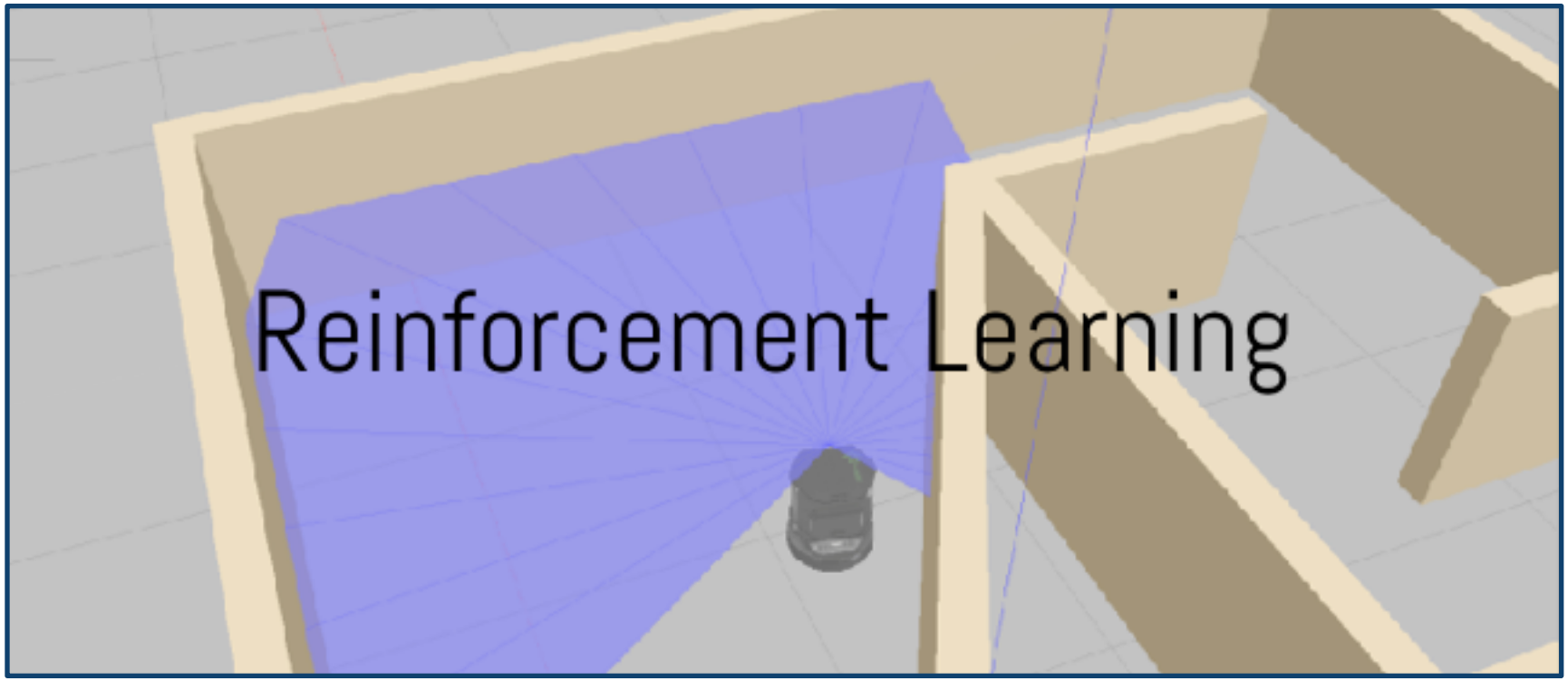
EINFÜHRUNG REINFORCEMENT LEARNING

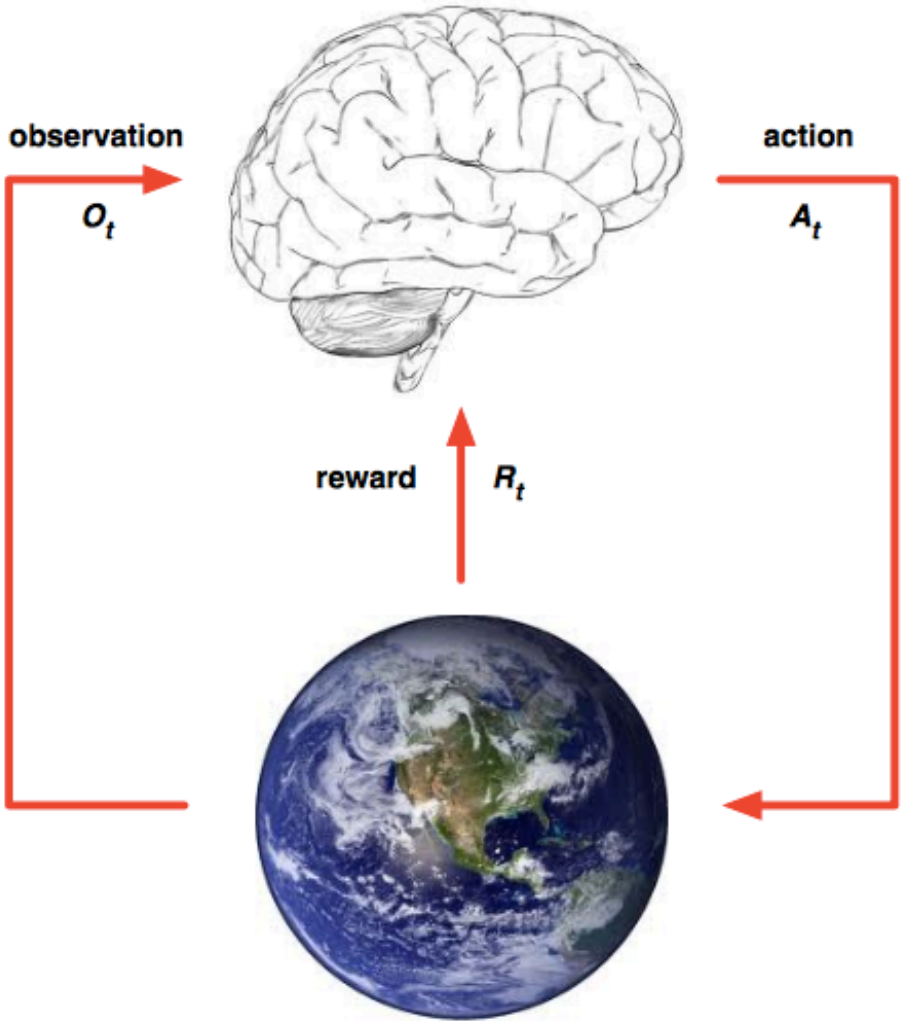
JAN-HENDRIK RUETTINGER



AGENDA

- Formelles Framework
- Bellman Gleichungen
- Lösung mit Model
- Lösung ohne Model (Monte Carlo Methoden)
- Lösung ohne Model (Temporal Difference L.)
- Lösung ohne Model mit Funktionsapproximation
- Ausblick

PROBLEMUMGEBUNG



MARKOV DECISION PROCESS (MDP)

$$MDP := < S, P, R, A, \gamma >$$

$S = \text{Set of states}$

$P = \text{State action probability matrix}$

$R = \text{Reward function}$

$A = \text{Set of actions}$

$\gamma = \text{Discount factor}$

HERLEITUNG MARKOV DECISION PROCESS (MDP)

$$\text{Markov Process} := < S, P >$$

$$S = \text{Set of states}$$

$$\text{P} = \text{State transition probability matrix}$$

HERLEITUNG MARKOV DECISION PROCESS (MDP)

$$\text{Markov Process} := < S, P, R >$$

$$\text{Markov Reward Process} := < S, P, R, \gamma >$$

$S = \text{Set of states}$

$\text{P} = \text{State transition probability matrix}$

$R = \text{Reward function}$

$\gamma = \text{Discount factor}$

HERLEITUNG MARKOV DECISION PROCESS (MDP)

$$\text{Markov Process} := < S, P, R >$$
$$\text{Markov Reward Process} := < S, P, R, \gamma >$$
$$\text{Markov Decision Process} := < S, P, R, A, \gamma >$$

$S = \text{Set of states}$

$P = \text{State action probability matrix}$

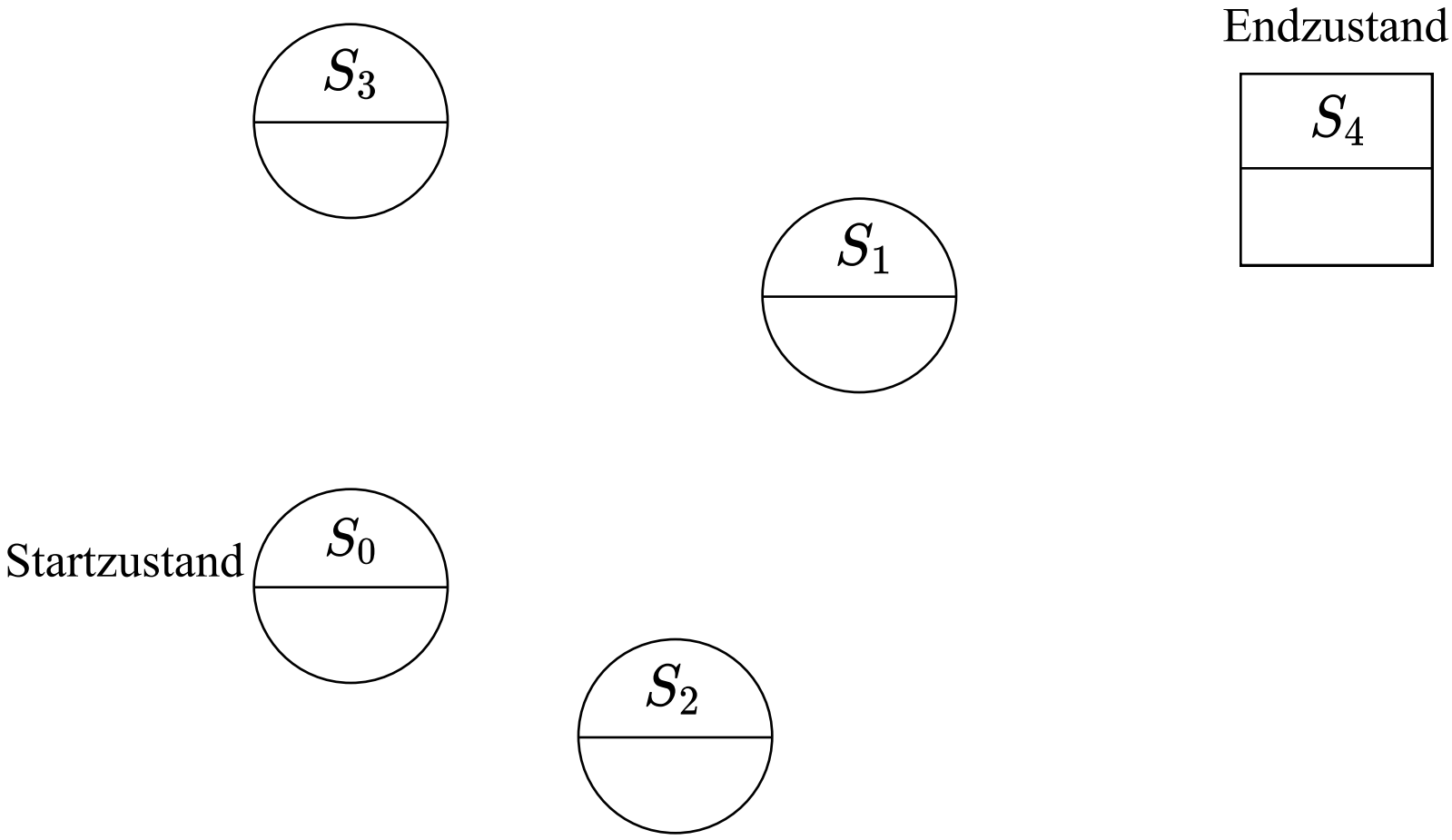
$R = \text{Reward function}$

$A = \text{Set of actions}$

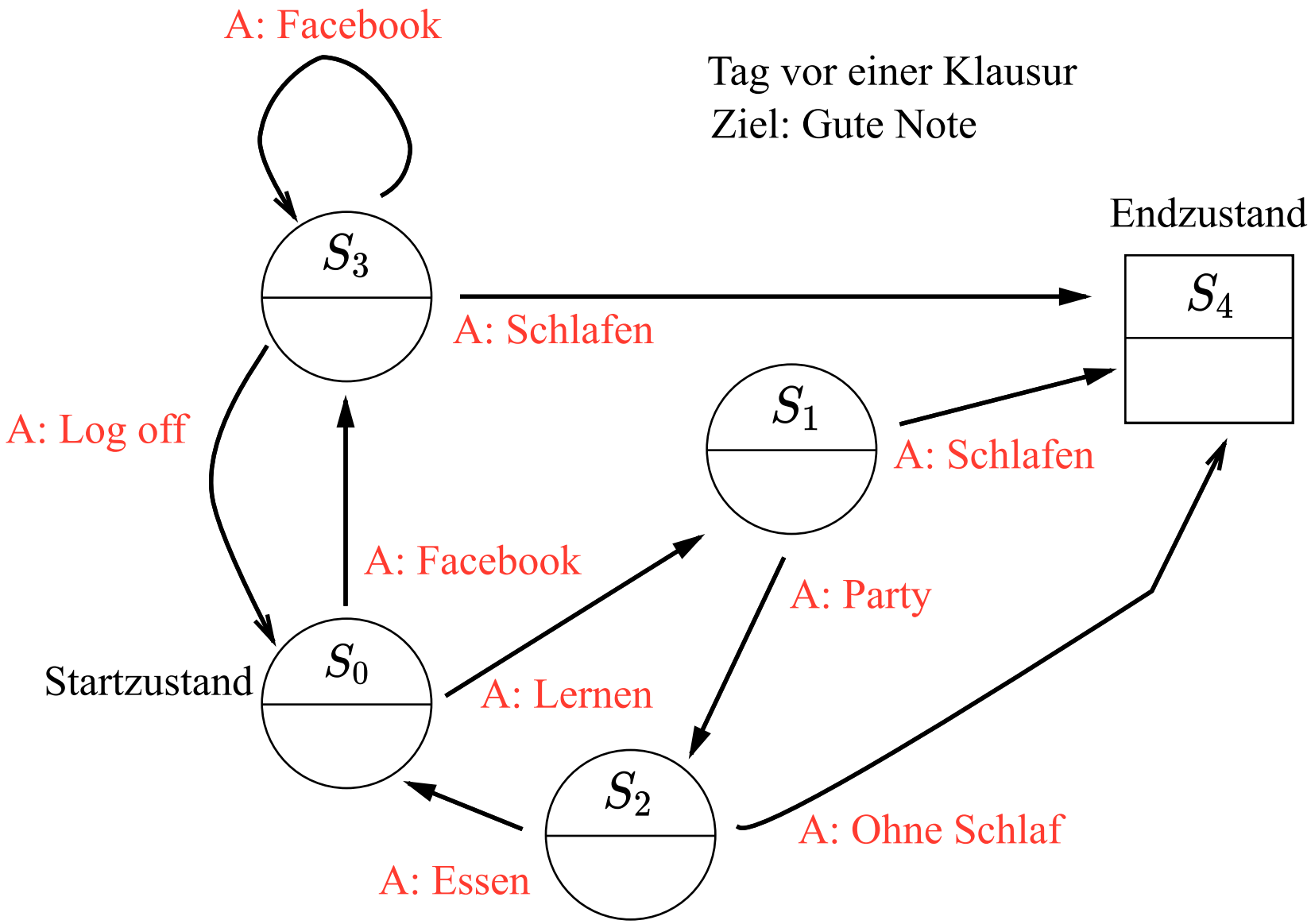
$\gamma = \text{Discount factor}$

NEUER BLICK AUF DIE PROBLEMUMGEBUNG

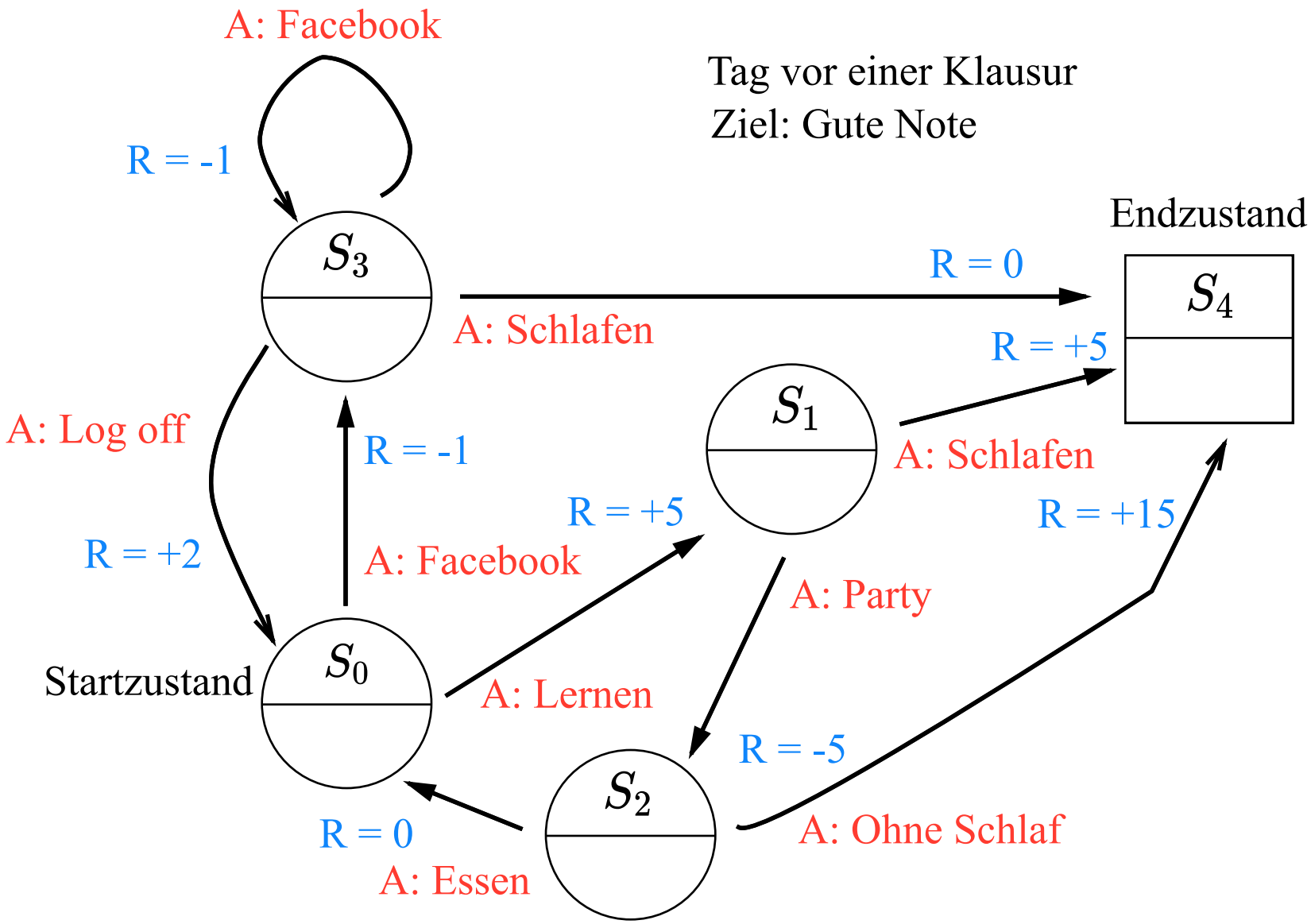
Tag vor einer Klausur
Ziel: Gute Note



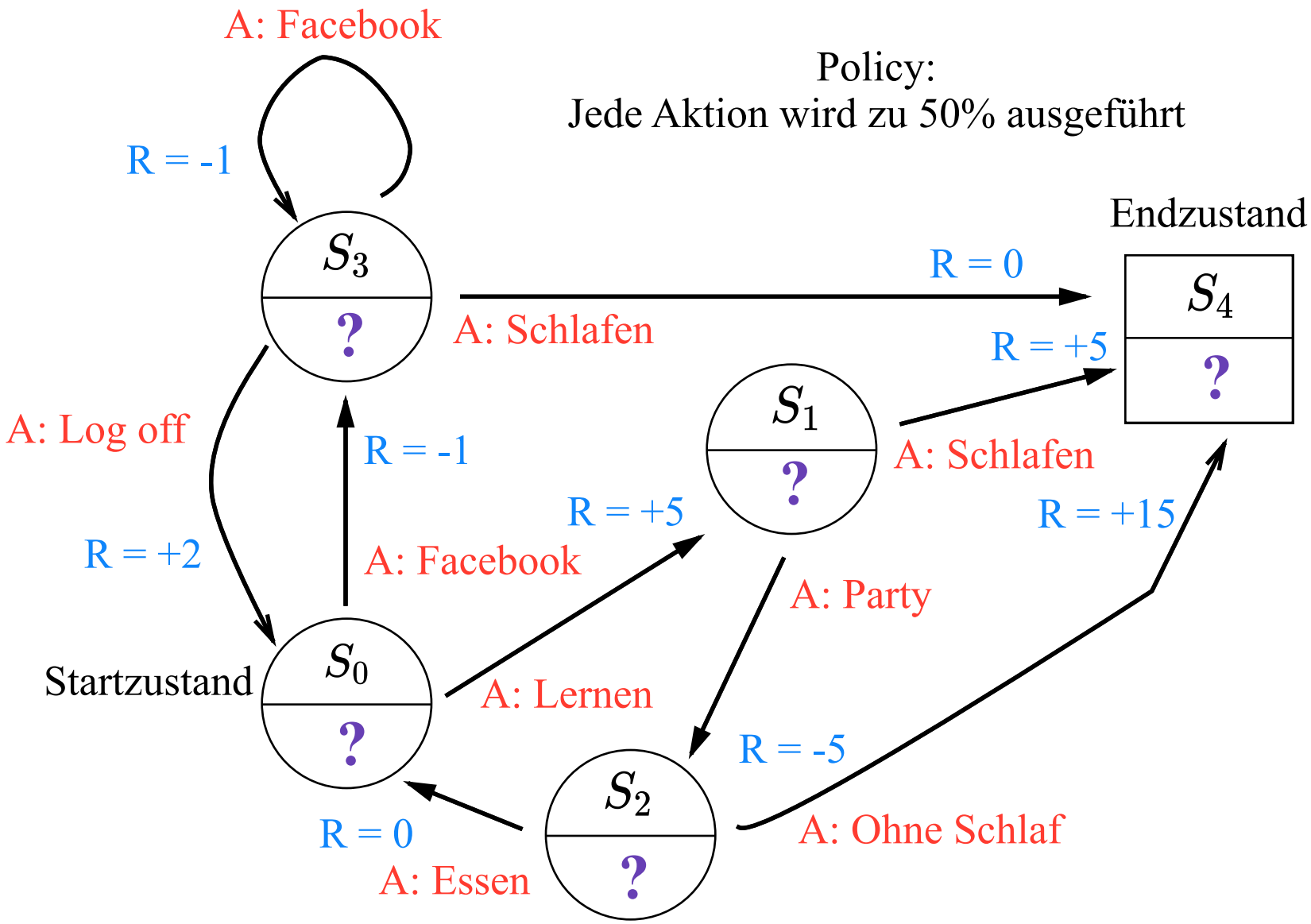
NEUER BLICK AUF DIE PROBLEMUMGEBUNG



NEUER BLICK AUF DIE PROBLEMUMGEBUNG



POLICY GEGEBEN => WERT VON JEDEM ZUSTAND UNBEKANNT



STATE VALUE FUNCTION

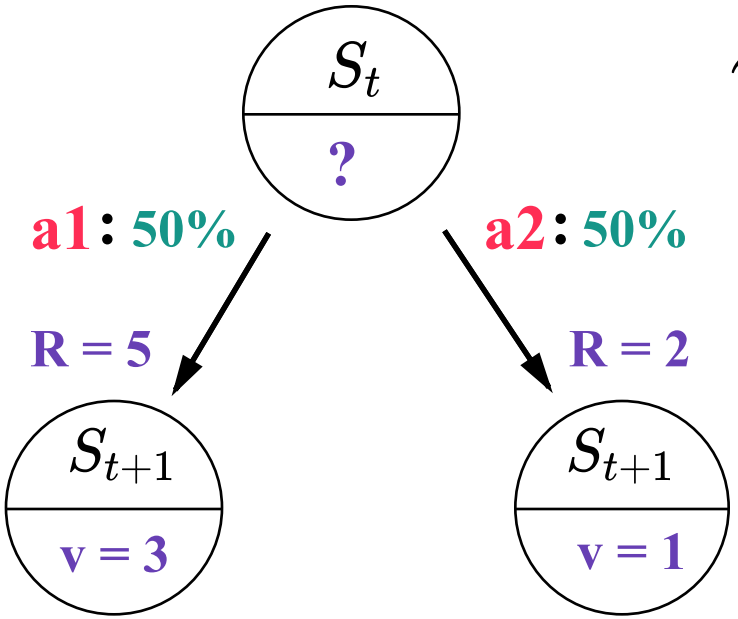
- Policy: $\pi(a|s) = P(A_t = a|S_t = s) = 0.5$
- Return in der Zukunft: $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- State Value Function: $v_\pi(s) = \mathbb{E}\{G_t|S_t = s\}$

STATE VALUE FUNCTION => BELLMAN EQUATION

- Policy: $\pi(a|s) = P(A_t = a|S_t = s) = 0.5$
- Return in der Zukunft: $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- State Value Function: $v_\pi(s) = \mathbb{E}\{G_t|S_t = s\}$
- Bellman expectation equation:

$$v_\pi(s) = \mathbb{E}\{R_{t+1} + \gamma v_\pi(s_{t+1})|S_t = s\}$$

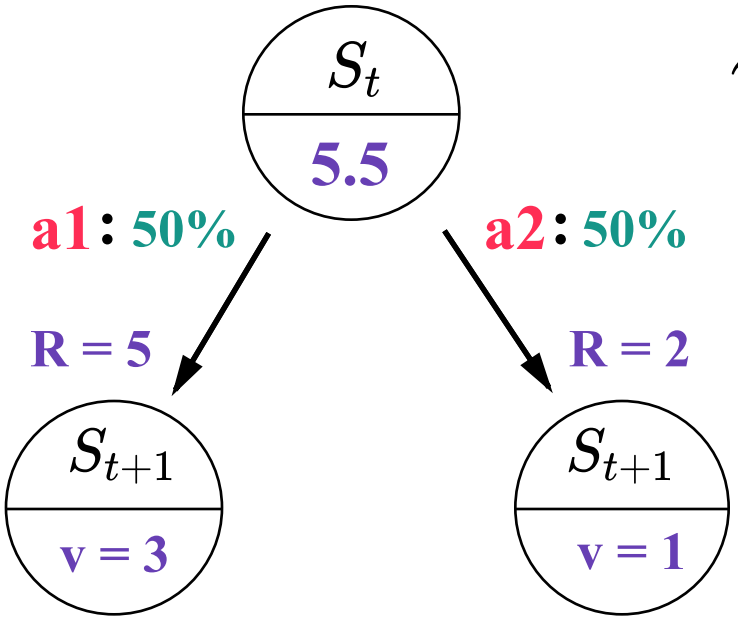
BELLMAN EXPECTATION EQUATION (BEISPIEL)



$$\gamma = 1$$

$$v_\pi(s) = \mathbb{E}\{R_{t+1} + \gamma v_\pi(s_{t+1}) | S_t = s\}$$

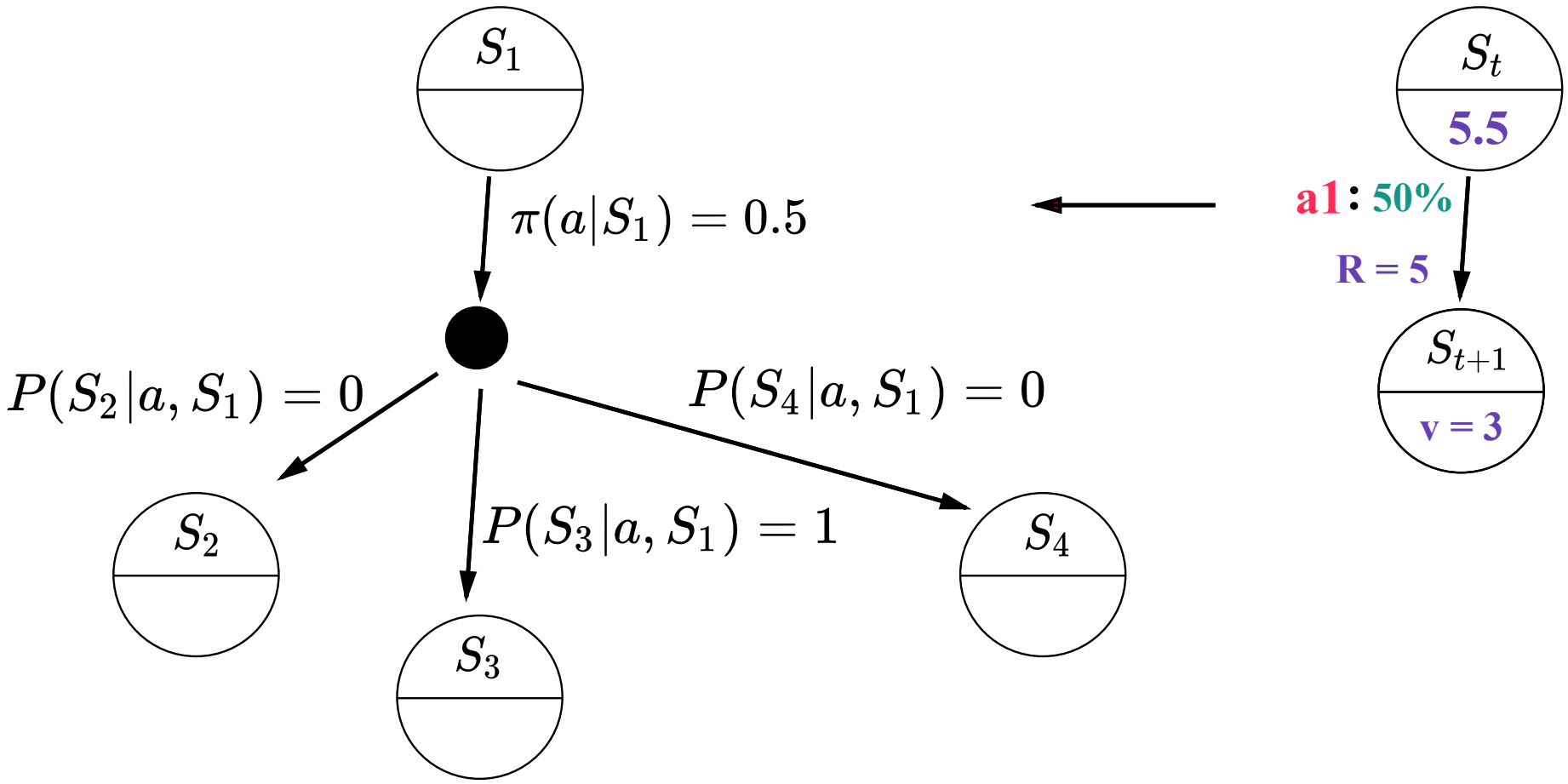
BELLMAN EXPECTATION EQUATION (BEISPIEL)



$$\gamma = 1$$

$$v_\pi(s) = \mathbb{E}\{R_{t+1} + \gamma v_\pi(s_{t+1}) | S_t = s\}$$

AKTION BESTIMMT NICHT ZWANGSLÄUFIG ZIELZUSTAND



$$v_\pi(s) = \mathbb{E}\{R_{t+1} + \gamma v_\pi(s_{t+1}) | S_t = s\}$$

MARKOV DECISION PROCESS (MDP)

$$MDP := < S, P, R, A, \gamma >$$

$S =$ Set of states

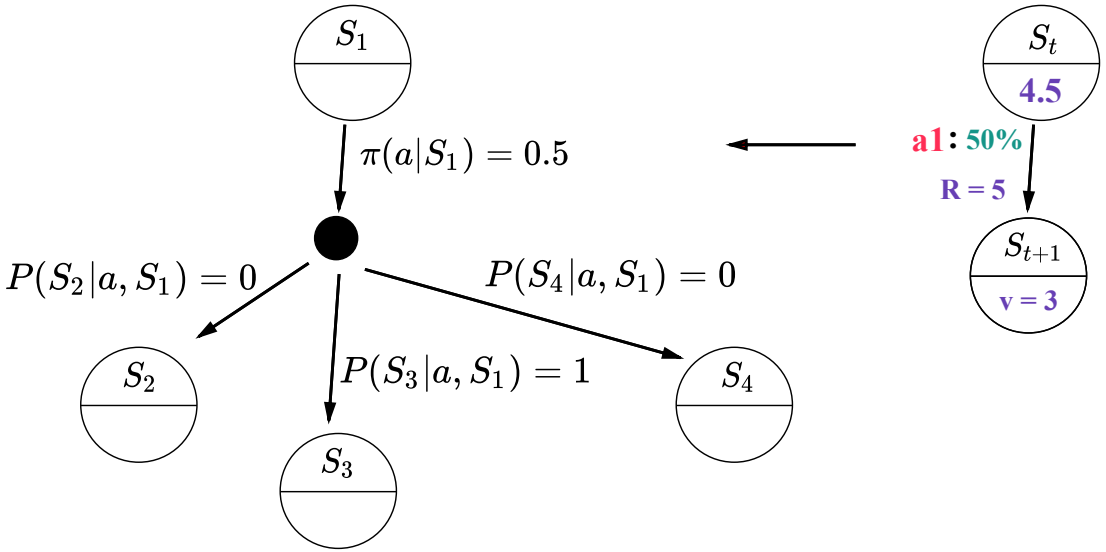
$P =$ State action probability matrix

$R =$ Reward function

$A =$ Set of actions

$\gamma =$ Discount factor

BELLMAN EXPECTATION EQUATION



$$v_\pi(s) = \mathbb{E}\{R_{t+1} + \gamma v_\pi(s_{t+1}) | S_t = s\}$$

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s_{t+1}} P(s_{t+1}|a, s)[R_{t+1} + \gamma v_\pi(s_{t+1})]$$

BELLMAN OPTIMALITY EQUATION

- Bellman expectation equation gilt nur unter der Voraussetzung, dass eine gegebene Policy immer befolgt wird

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s_{t+1}} P(s_{t+1}|a, s)[R_{t+1} + \gamma v_\pi(s_{t+1})]$$

BELLMAN OPTIMALITY EQUATION

- Bellman expectation equation gilt nur unter der Voraussetzung, dass eine gegebene Policy immer befolgt wird

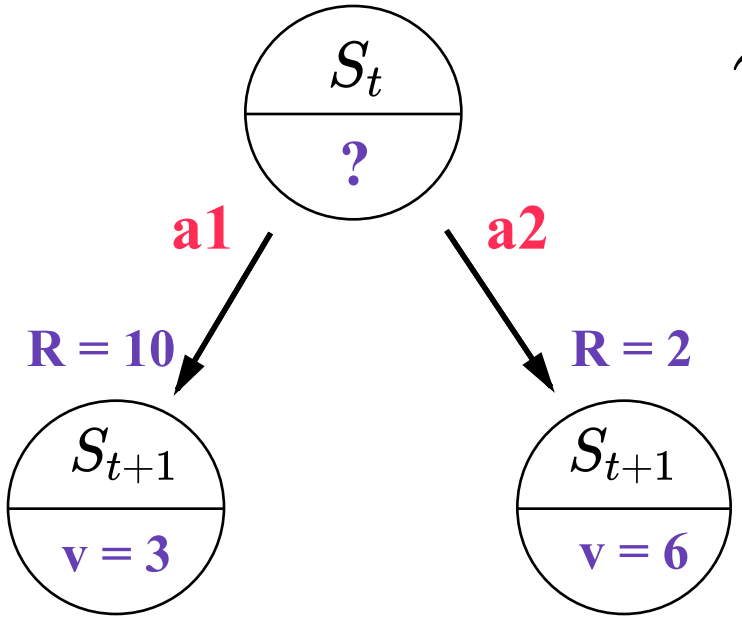
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s_{t+1}} P(s_{t+1}|a, s)[R_{t+1} + \gamma v_\pi(s_{t+1})]$$

- Unser Ziel: Optimale Policy
- Bellman optimality equation:

$$v_*(s) = \max_a \mathbb{E}\{R_{t+1} + \gamma v_*(s_{t+1})|S_t = s\}$$

- Nicht linear durch max
- Kern von RL = Finden einer Lösung für das Gleichungssystem

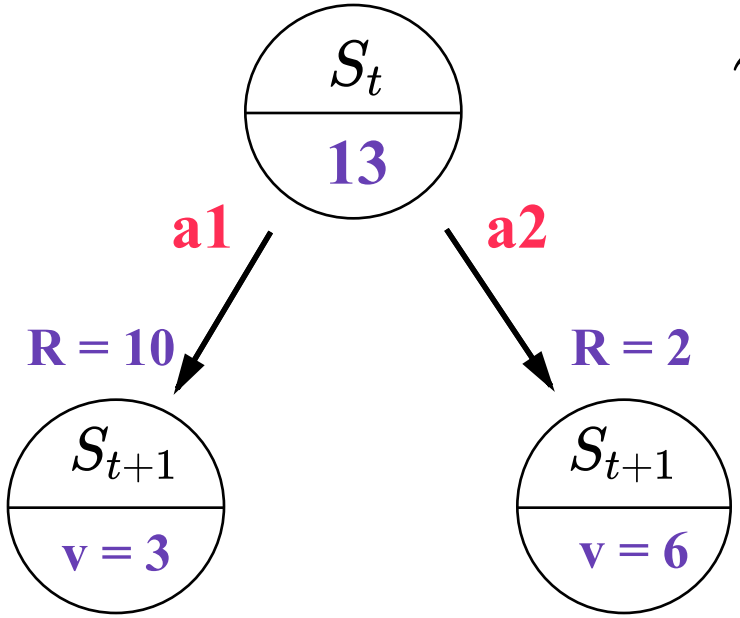
BELLMAN OPTIMALITY EQUATION (BEISPIEL)



$$\gamma = 1$$

$$v_*(s) = \max_a \mathbb{E}\{R_{t+1} + \gamma v_*(s_{t+1}) | S_t = s\}$$

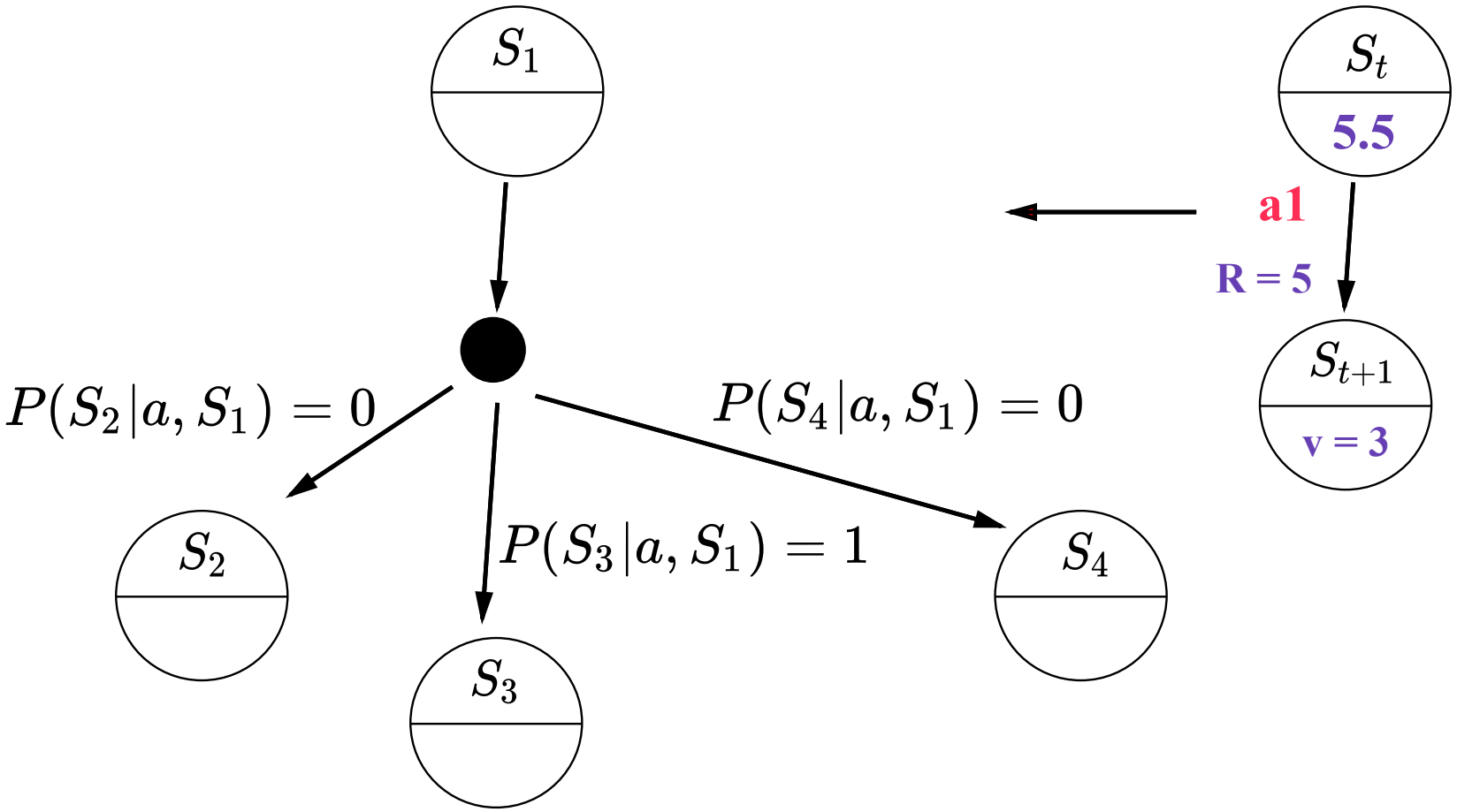
BELLMAN OPTIMALITY EQUATION (BEISPIEL)



$$\gamma = 1$$

$$v_*(s) = \max_a \mathbb{E}\{R_{t+1} + \gamma v_*(s_{t+1}) | S_t = s\}$$

AKTION BESTIMMT NICHT ZWANGSLÄUFIG ZIELZUSTAND



$$v_*(s) = \max_a \sum_{s_{t+1}} P(s_{t+1}|a, s)[R_{t+1} + \gamma v_*(s_{t+1})]$$

LÖSUNGSSTRATEGIEN

$$v_*(s) = \max_a \sum_{s_{t+1}} P(s_{t+1}|a, s)[R_{t+1} + \gamma v_*(s_{t+1})]$$

- Model der Umgebung = $P(S_{t+1}|a, S)$

- Model gegeben: Dynamic Programing (iterativ)
- Ohne Model (Realität): Z. B. Monte Carlo (Schätzung + iterativ)

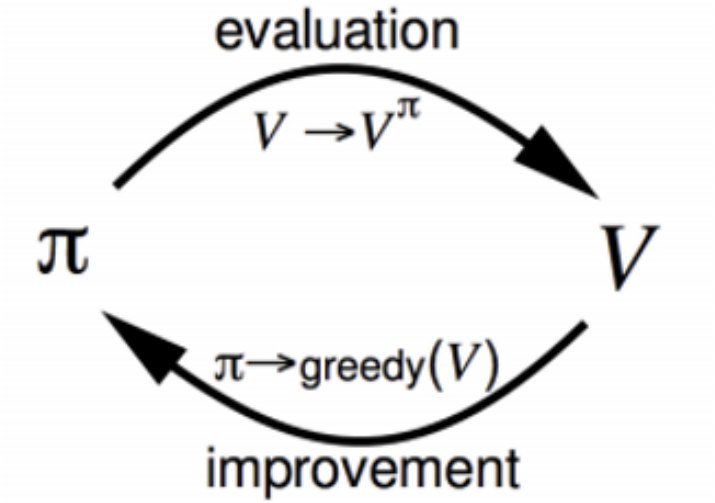
LÖSUNGSSTRATEGIEN

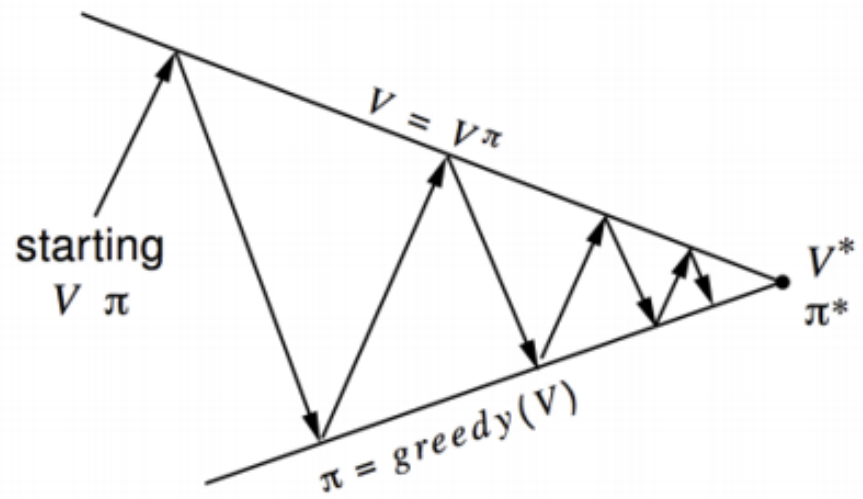
$$v_*(s) = \max_a \sum_{s_{t+1}} P(s_{t+1}|a, s)[R_{t+1} + \gamma v_*(s_{t+1})]$$

- Model der Umgebung = $P(S_{t+1}|a, S)$
- Model gegeben: Dynamic Programing (iterativ)
- Ohne Model (Realität): Z. B. Monte Carlo (Schätzung + iterativ)

DYNAMIC PROGRAMMING: GRUNDLAGEN

- Klasse von Algorithmen, die für ein MDP mit gegebenem Model eine optimale Policy finden
- Strategie: General Policy Iteration (GPI)





DYNAMIC PROGRAMMING: POLICY ITERATION (BEISPIEL)

- Start: $v_\pi(s)$ und $\pi(a|s)$ zufällig initialisieren

DYNAMIC PROGRAMMING: POLICY ITERATION

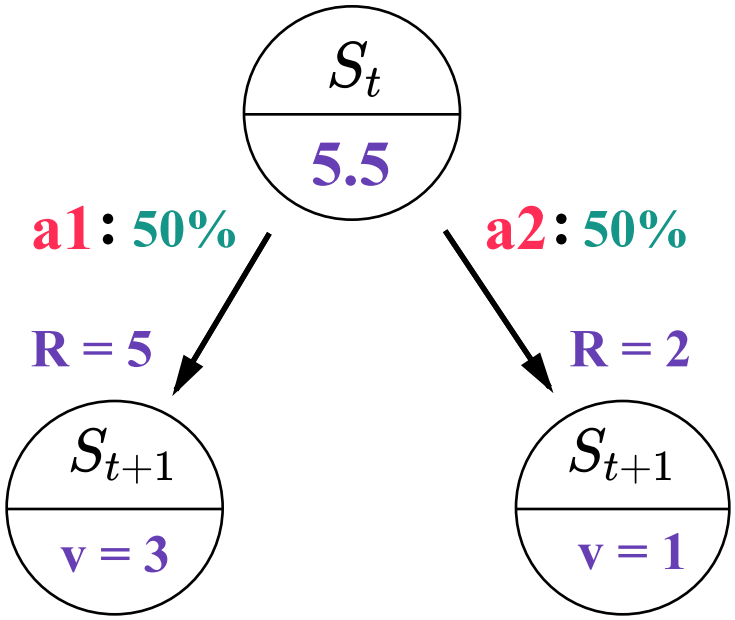
- Start: $v_\pi(s)$ und $\pi(a|s)$ zufällig initialisieren
- Policy Evaluation:
  - $k :=$ Aktueller Iterationsschritt
  - $v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s_{t+1}} P(s_{t+1}|a,s)[R_{t+1} + \gamma v_k(s_{t+1})]$
  - $k \to \infty \implies v_k \to v_\pi$
  - $|v_k(s) - v_{k+1}(s)| < Schwelle$

DYNAMIC PROGRAMMING: POLICY ITERATION

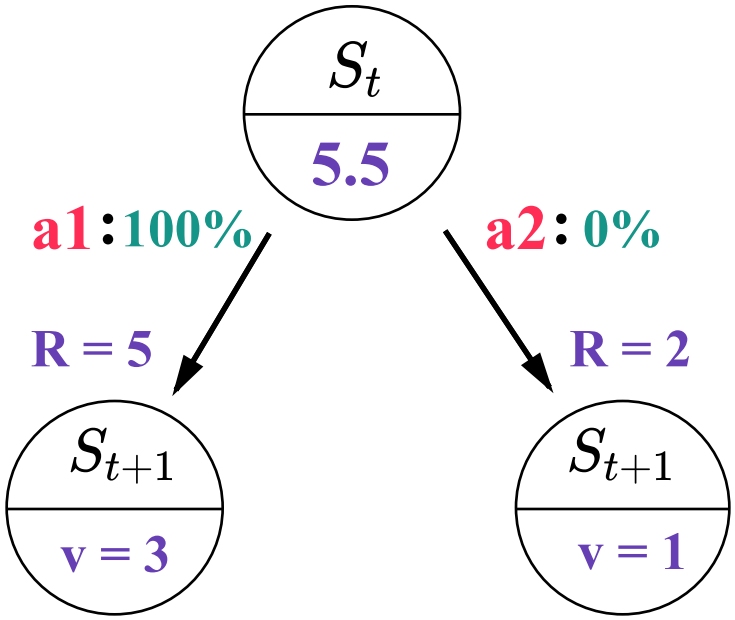
- Start: $v_\pi(s)$ und $\pi(a|S)$ zufällig initialisieren
- Policy Evaluation:
  - $$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s_{t+1}} P(s_{t+1}|a, s)[R_{t+1} + \gamma v_k(s_{t+1})]$$
- Policy Improvement:
  - Verbesserte Policy greedy bestimmen

$$\pi_0 \rightarrow v_{\pi 0} \rightarrow \pi_1 \rightarrow v_{\pi 1} \rightarrow \ \ldots \rightarrow v_* \rightarrow \pi_*$$

DYNAMIC PROGRAMMING: POLICY GREEDY BESTIMMEN



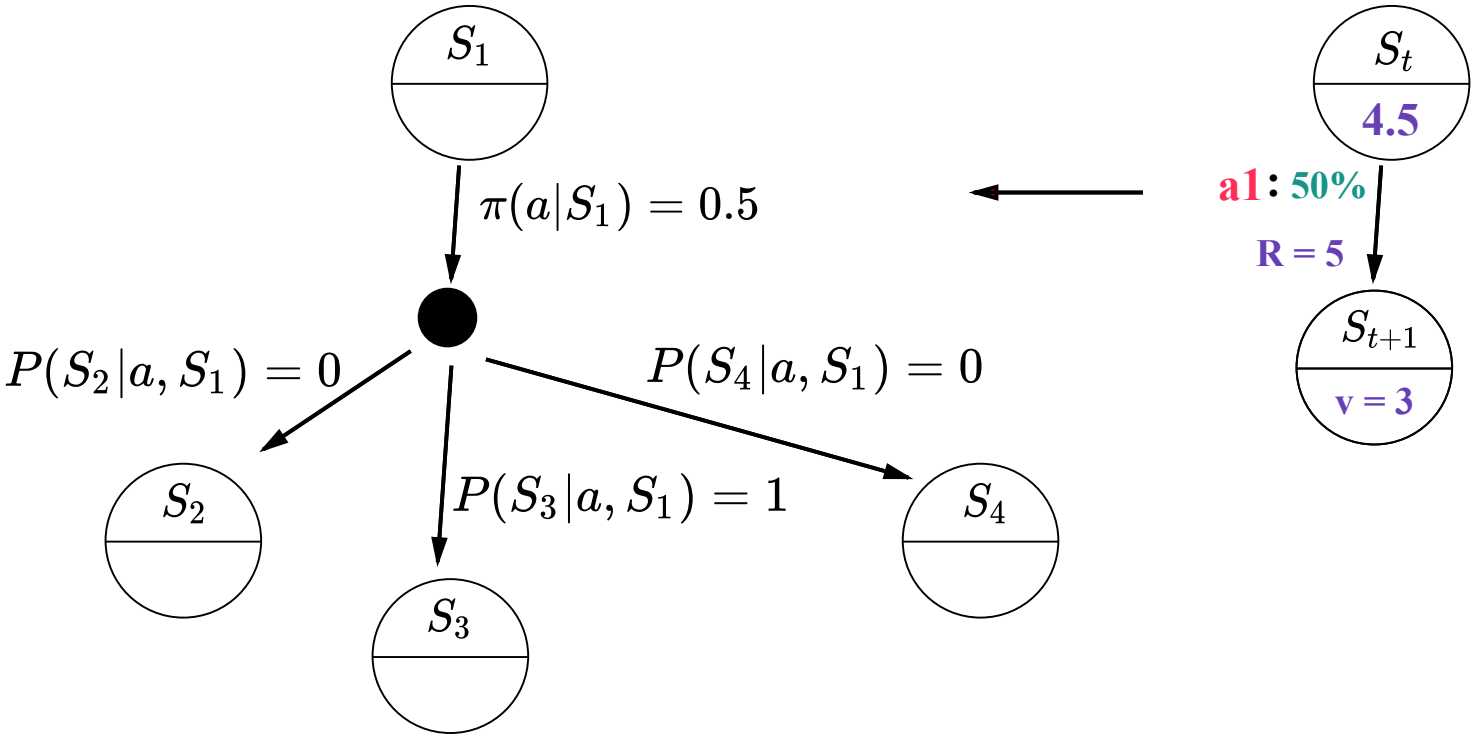
DYNAMIC PROGRAMMING: POLICY GREEDY BESTIMMEN



AGENDA

- Formelles Framework
- Bellman Gleichungen
- Lösung mit Model
- Lösung ohne Model (Monte Carlo Methoden)
- Lösung ohne Model (Temporal Difference L.)
- Lösung ohne Model mit Funktionsapproximation
- Ausblick

LÖSUNGSSTRATEGIEN OHNE BEKANNTES MODEL



- P ist nicht bekannt/zu kompliziert zu bestimmen

LÖSUNGSSTRATEGIEN OHNE BEKANNTES MODEL

- Monte Carlo Methoden
  - On-Policy Algorithmus
  - Off-Policy Algorithmus

- Temporal Difference Learning
  - On-Policy Algorithmus (SARSA)
  - Off-Policy Algorithmus (Q-Learning)
  - Mischung: Expected SARSA

MONTE CARLO METHODEN

- Model P ist nicht bekannt
- Lernt von echter Erfahrung (Unterschied zu DP)
- Schätzung der Value function
- Schätzung hängt NICHT von anderen Schätzungen ab
  => Kein Bootstrapping
- Grundprinzip: General Policy Iteration (GPI)
- Update erst nach voller Episode (Unterschied zu TD)

DEMO: STAB BALANCIEREN (TEIL 1)

MONTE CARLO METHODEN

Value function und Policy zufällig initialisieren

Wiederhole:

1. Episode aufnehmen
2. Aufsummierten Reward für jeden State bestimmen

   => Value function updaten (Mittelwert)
3. Policy updaten

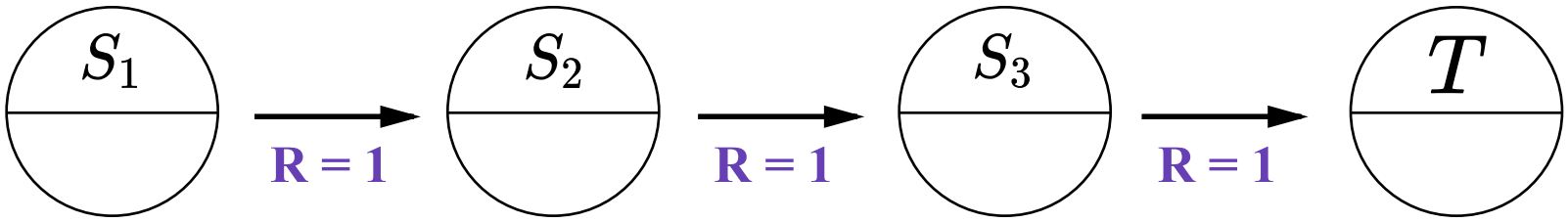
MONTE CARLO METHODEN

Value function und Policy zufällig initialisieren

Wiederhole:

1. Episode aufnehmen (nach Policy)
2. Aufsummierten Reward für jeden State bestimmen

   => Value function updaten (Mittelwert)
3. Policy updaten



MONTE CARLO METHODEN

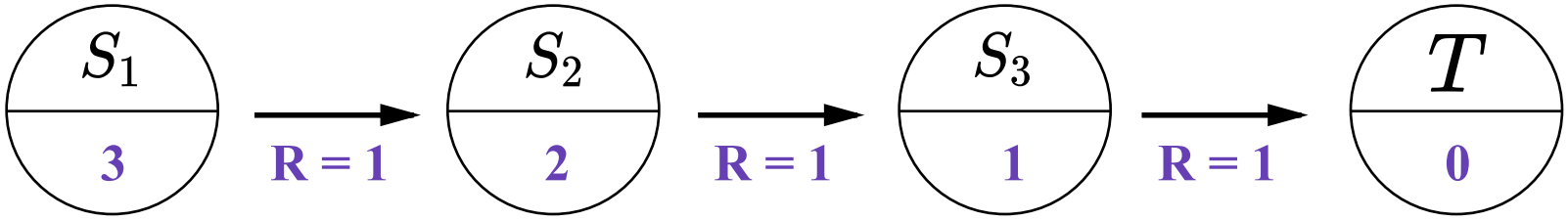
Value function und Policy zufällig initialisieren

Wiederhole:

1. Episode aufnehmen
2. Aufsummierten Reward für jeden State bestimmen

   => Value function updaten (Mittelwert)
3. Policy updaten

$$q_\pi(s, a) = \text{mean}(\text{Return}(\text{s,a}))$$



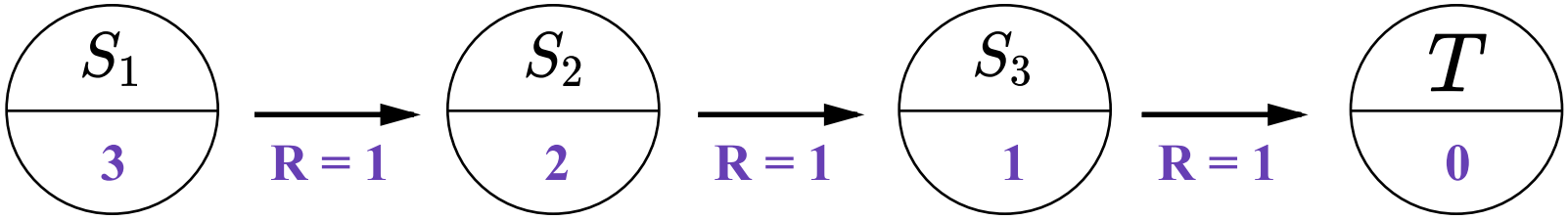
MONTE CARLO METHODEN

Value function und Policy zufällig initialisieren

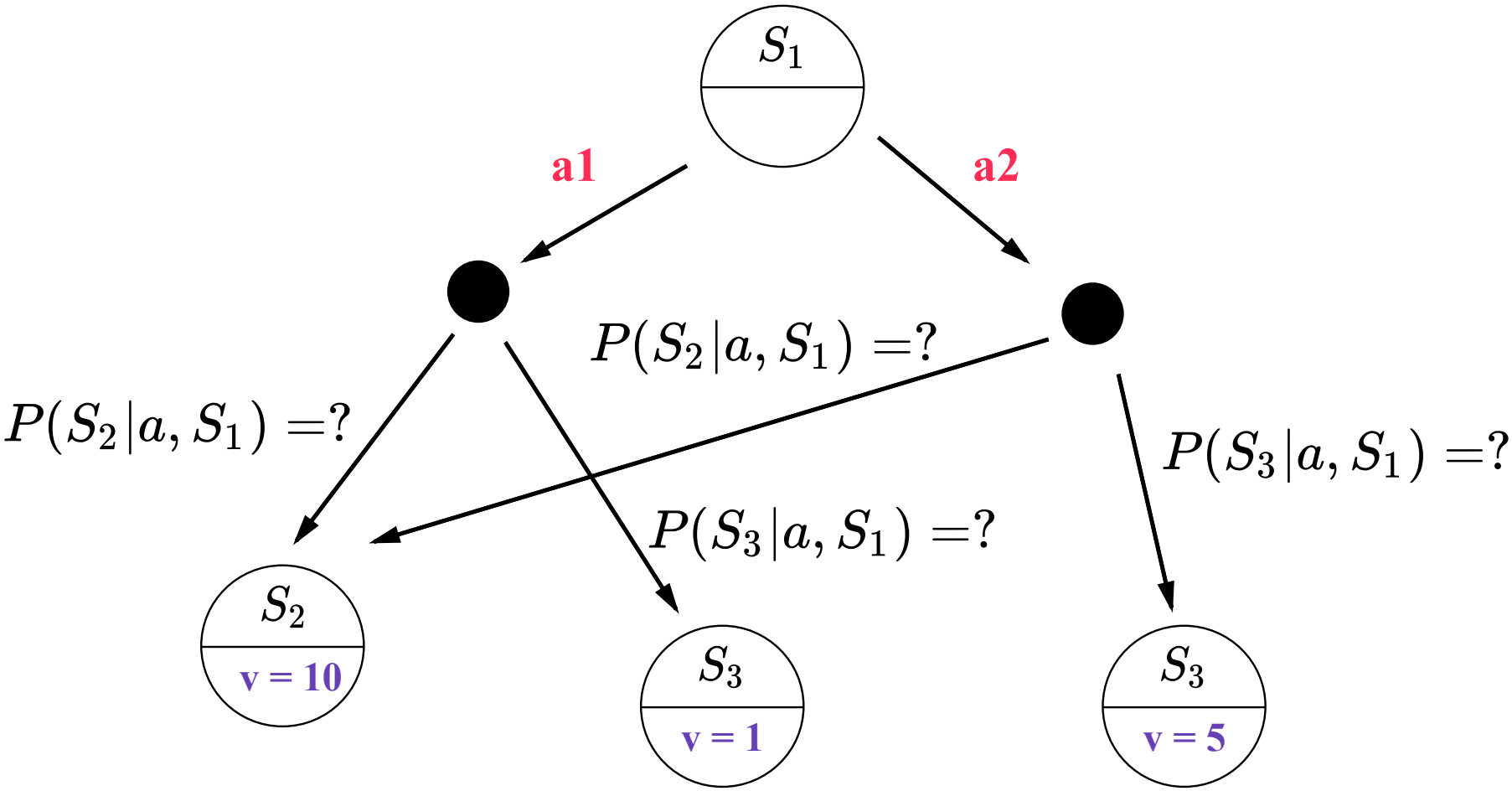
Wiederhole:

1. Episode aufnehmen
2. Aufsummierten Reward für jeden State bestimmen

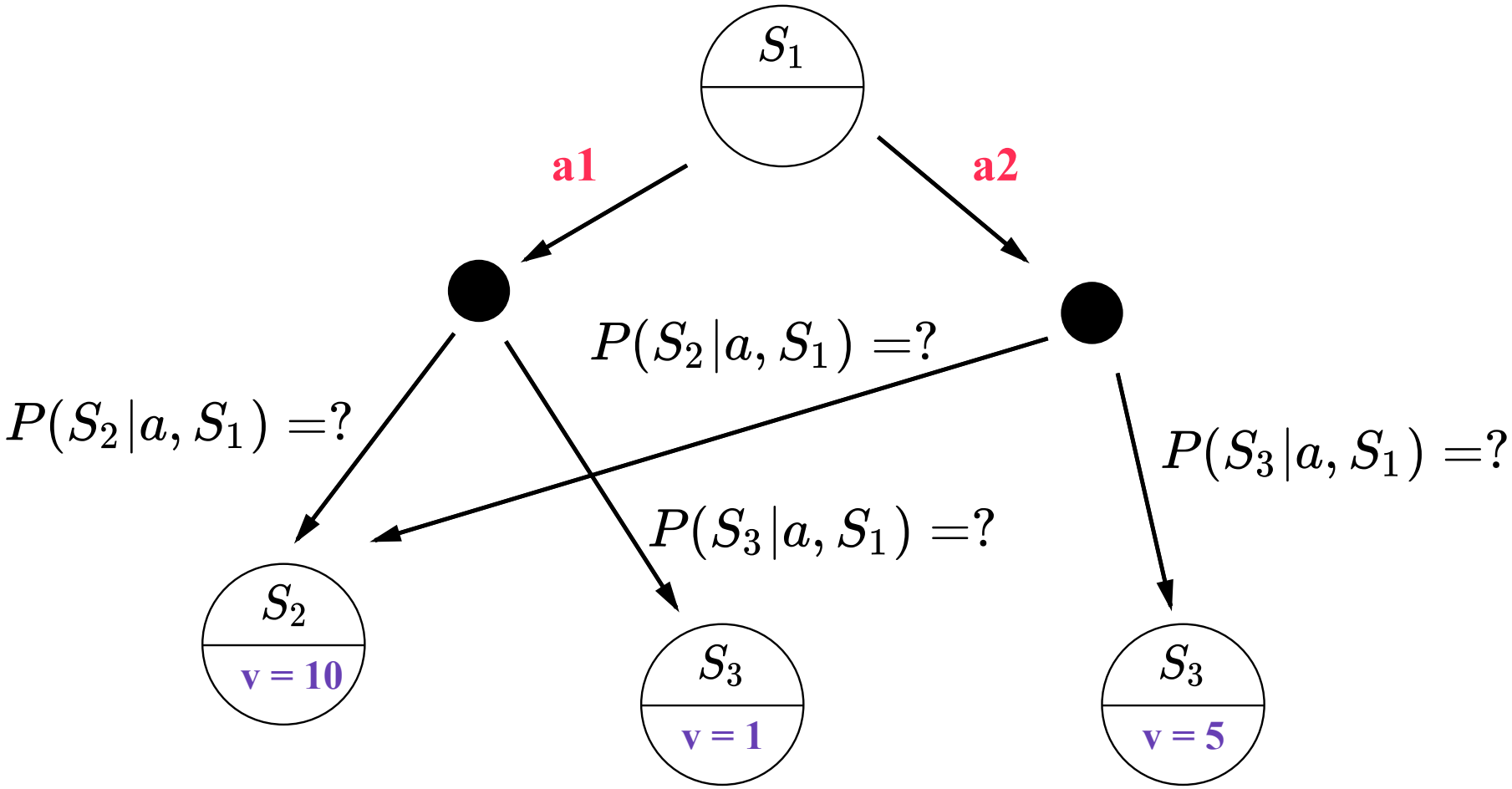
   => Value function updaten (Mittelwert)
3. Policy updaten (Greedy => PROBLEM)



WELCHE AKTION SOLL ZUKÜNFTIG AUSGEWÄHLT WERDEN?



WELCHE AKTION SOLL ZUKÜNFTIG AUSGEWÄHLT WERDEN?



➡ Einführung einer neuen State-Value function notwendig

NEUE STATE-ACTION VALUE FUNCTION

- Value Function:

$$v_\pi(s):\ S \to \mathbb{R}$$

- State Value Function:

$$q_\pi(s,a):\ S \times A \to \mathbb{R}$$

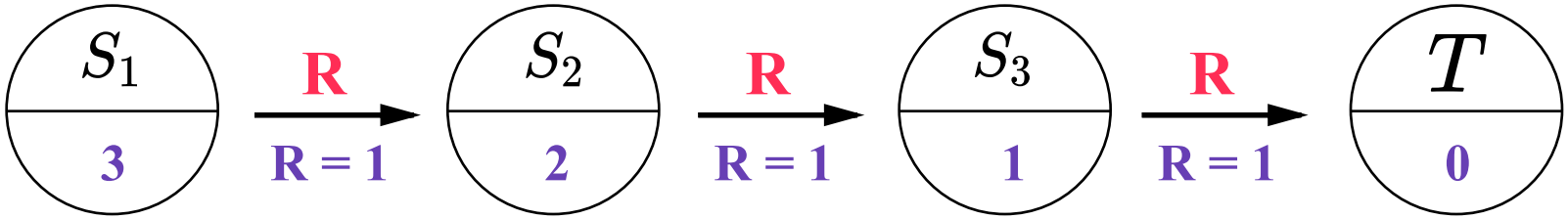
- Bellman equations:

$$q_\pi(s,a) = \mathbb{E}\{R_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1}) | S_t = s, A_t = a\}$$

$$q_*(s,a) = \mathbb{E}\{R_{t+1} + \gamma \max_{a_{t+1}}(q_*(s_{t+1}, a_{t+1})) | S_t = s, A_t = a\}$$

NEUE STATE-ACTION VALUE FUNCTION

- State + Action: Zukünftiger Reward



- Bestimmung der greedy Policy ist nun trivial
- Greedy Policy führt zu einem neuen Problem

MONTE CARLO METHODE: ON-POLICY LEARNING

Value function und Policy zufällig initialisieren

Wiederhole:

1. Episode aufnehmen
2. Aufsummierten Reward für jeden State bestimmen

   => Value function updaten (Mittelwert)
3. Policy updaten (Greedy => PROBLEM)

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, & \text{wenn a} = \text{A}^*. \\ \frac{\epsilon}{|A(s)|}, & \text{wenn a} \neq \text{A}^*. \end{cases}$$

DEMO: STAB BALANCIEREN (TEIL 2)

MONTE CARLO METHODE: OFF-POLICY LEARNING

Value function und zwei Policies A und B aufstellen

Wiederhole:

1. Episode aufnehmen mit Policy B
2. Aufsummierten Reward für jeden State bestimmen

   => Value function updaten (Importance Sampling)
3. Policy A greedy updaten

Problem: Schätzung von einem Erwartungswert q(s,a) der auf einer Verteilung/Policy A beruht mit Hilfe von Samples aus einer Verteilung B.

Lösung: Importance Sampling (verschiedene Gewichtung der Samples) zur Verringerung der Varianz des Schätzwerts

AGENDA

- Formelles Framework
- Bellman Gleichungen
- Lösung mit Model
- Lösung ohne Model (Monte Carlo Methoden)
- Lösung ohne Model (Temporal Difference L.)
- Lösung ohne Model mit Funktionsapproximation
- Ausblick

TEMPORAL DIFFERENCE LEARNING

- Model P ist nicht bekannt
- Lernt von echter Erfahrung (wie bei Monte Carlo Methoden)
- Schätzung der State-Value function
- Schätzung hängt von anderen Schätzungen ab (Bootstrapping)
- Grundprinzip: General Policy Iteration (GPI)
- Update kann nach beliebigen Zeitschritten erfolgen (Unterschied zu Monte Carlo Methoden)

TEMPORAL DIFFERENCE LEARNING: ON-POLICY (SARSA)

Value function und Policy zufällig initialisieren

Wiederhole für jede Episode:

Wiederhole bis Ende der Episode:

1. Führe Aktion aus => Erhalte neuen Zustand
2. Update State-Value function
3. Update Policy (e-greedy)

TEMPORAL DIFFERENCE LEARNING: ON-POLICY (SARSA)

Value function und Policy zufällig initialisieren

Wiederhole für jede Episode:

Wiederhole bis Ende der Episode:

1. Führe Aktion aus => Erhalte neuen Zustand
2. Update State-Value function
3. Update Policy (e-greedy)

Updateregel:

$$q_{\pi,neu}(s,a) = q_{\pi,alt}(s,a) + \alpha[R_{t+1} + \gamma q_{\pi,alt}(s_{t+1}, a_{t+1})]$$

TEMPORAL DIFFERENCE LEARNING: OFF-POLICY (Q-LEARNING)

Value function und zwei Policies A und B zufällig initialisieren

Wiederhole für jede Episode:

Wiederhole bis Ende der Episode:

1. Führe Aktion aus mit Policy A => Erhalte neuen Zustand
2. Update State-Value function
3. Update Policy B (greedy)

Updateregel:

$$q_{B,neu}(s,a) = q_{B,alt}(s,a) + \alpha[R_{t+1} + \gamma \max_{a_{t+1}} q_{B,alt}(s_{t+1}, a_{t+1})]$$

AGENDA

- Formelles Framework
- Bellman Gleichungen
- Lösung mit Model
- Lösung ohne Model (Monte Carlo Methoden)
- Lösung ohne Model (Temporal Difference L.)
- Lösung ohne Model mit Funktionsapproximation
- Ausblick

DEMO: CARTPOLE OHNE ROUNDING

DEMO: CARTPOLE OHNE ROUNDING

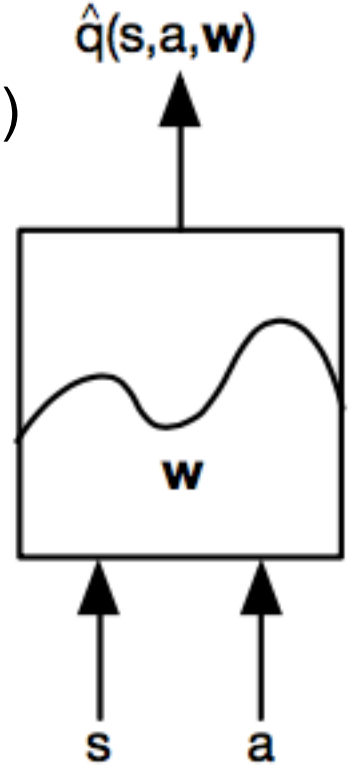
Problem: Tabelle wird zu groß bzw. Training dauert zu lange

STATE-ACTION	VALUE
1.0192929292, 1	10
1.0919292923, 1	15
1.0909123123, 1	16
1.0919292929, 1	3
1.0919292929, 0	4
1.0929292929, 1	2
…	…

FUNCTION APPROXIMATION

- $q_\pi(s,a) \approx \hat{q}_\pi(s,a)$
- Kontinuierliche Funktion (basiert nicht auf Tabelle)
- Liefert Values für vorher unbekannte States
- Muss “Vergessen” können

  => Updates sind nicht stationär
- Muss “on the fly” lernen können



GRUNDLAGEN FUNCTION APPROXIMATION

- Kostenfuntkion: $J(w) = \mathbb{E}\{(q(s,a) - \hat{q}_\pi(s,a,w))^2\}$
- Kostenfunktion soll minimiert werden => Gradient Descent Algo.
- q(s,a) ist unbekannt und wird durch ein Target ersetzt
- MC: $\Delta w = -\alpha(G_t - \hat{q}_\pi(s,a,w))\nabla J(w)$
- TD: $\Delta w = -\alpha(R_{t+1} + \gamma\hat{q}_\pi(s_{t+1}, a_{t+1}, w) - \hat{q}_\pi(s,a,w))\nabla J(w)$

WELCHE VERÄNDERUNGEN ERGEBEN SICH?

Value function und Policy zufällig initialisieren

Wiederhole für jede Episode:

Wiederhole bis Ende der Episode:

1. Führe Aktion aus => Erhalte neuen Zustand
2. Update State-Value function
3. Update Policy (e-greedy)

Updateregel:

$$q_{\pi,neu}(s,a) = q_{\pi,alt}(s,a) + \alpha[R_{t+1} + \gamma q_{\pi,alt}(s_{t+1}, a_{t+1})]$$

$$\Delta w = -\alpha(R_{t+1} + \gamma \hat{q}_\pi(s_{t+1}, a_{t+1}, w) - \hat{q}_\pi(s,a,w))\nabla J(w)$$

FUNCTIONSAPPROXIMATION KONVERGENZ

Konvergenztabelle für Policy Evaluation Algorithmus

On/Off	Algorithmus	Tabular	Linear Approx.	Nicht-linear Approx.
On	Monte Carlo	Ja	Ja	Ja
	TD(0)	Ja	Ja	Nein
	TD(lamda)	Ja	Ja	Nein
Off	Monte Carlo	Ja	Ja	Ja
	TD(0)	Ja	Nein	Nein
	TD(lambda)	Ja	Nein	Nein

FUNCTIONSAPPROXIMATION KONVERGENZ

Konvergenztabelle für Control Algorithmus

On/Off	Algorithmus	Tabular	Linear Approx.	Nicht-linear Approx.
On	Monte Carlo	Ja	(Ja)	Nein
	TD(0)	Ja	(Ja)	Nein
	TD(lamda)	Ja	Ja	Nein
Off	Monte Carlo	Ja	Ja	Nein
	TD(0)	Ja	Nein	Nein
	TD(lambda)	Ja	Nein	Nein

FUNCTIONSAPPROXIMATION KONVERGENZ

Konvergenztabelle für Control Algorithmus

On/Off	Algorithmus	Tabular	Linear Approx.	Nicht-linear Approx.
On	Monte Carlo	Ja	(Ja)	Nein
	TD(0)	Ja	(Ja)	Nein
	TD(lamda)	Ja	Ja	Nein
Off	Monte Carlo	Ja	Ja	Nein
	TD(0)	Ja	Nein	Nein
	TD(lambda)	Ja	Nein	Nein

DEMO: CARTPOLE (DEEP Q-LEARNING)