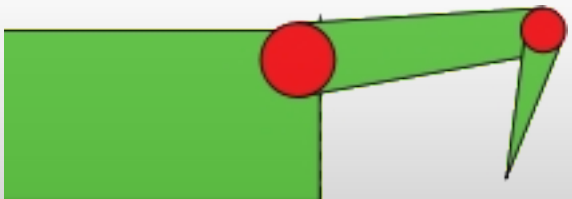
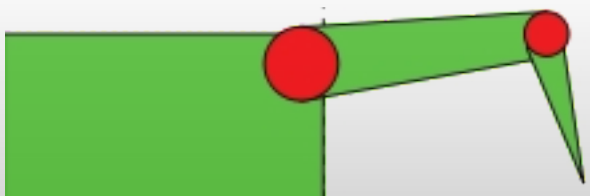
Reinforcement Learning für Lauf- und Krabbelbewegung von Robotern

SE Biologisch motivierte Lernverfahren



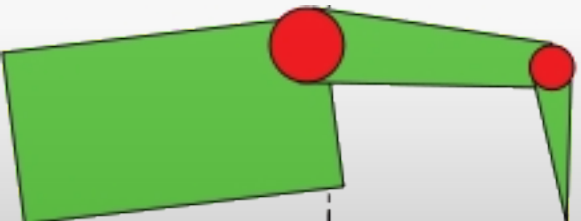
Reinforcement Learning

- lernen aus Interaktion mit der Umwelt
- dynamisches Programm

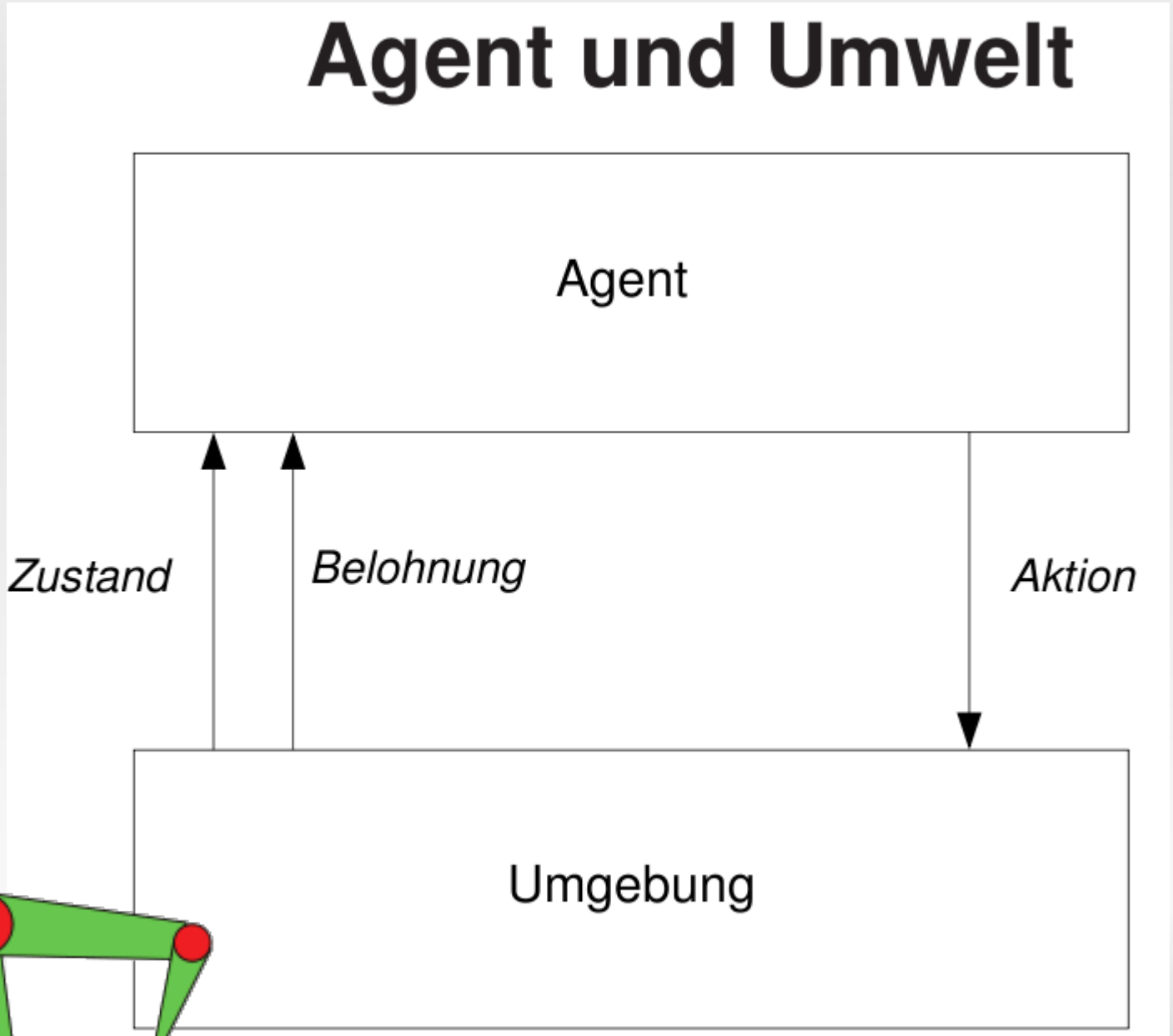


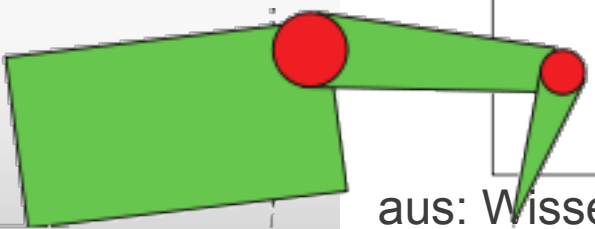
Reinforcement Learning

- lernen aus Interaktion mit der Umwelt
- dynamisches Programm
- das Ziel ist bekannt
- lernen durch Belohnung



Reinforcement Learning

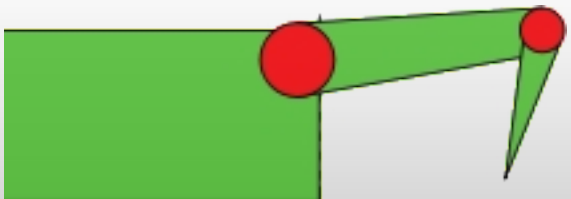




aus: Wissensarchitekturen — HS Bonn-Rhein-Sieg, WS 09/10

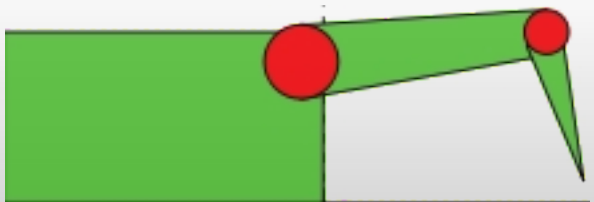
Reinforcement Learning

- Zustände s_i
- Aktionen a_i
- Belohnungen r_i



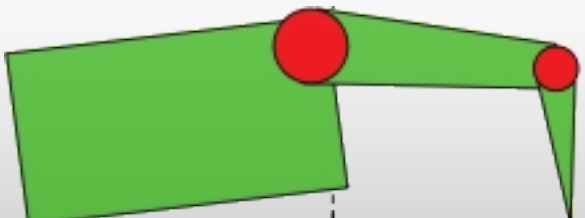
Reinforcement Learning

- Zustandsübergange sind deterministisch oder stochastisch
- Zustandsübergange sind vorher bekannt oder müssen gelernt werden
- Belohnungen können deterministisch oder stochastisch sein



Wahl von Aktionen

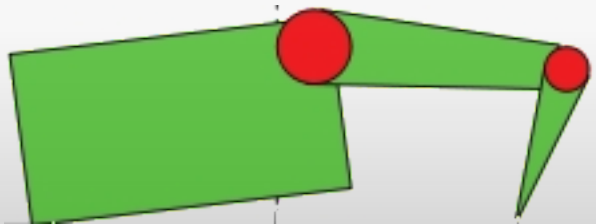
- Agent wählt Aktionen anhand einer Policy π
- $\pi(s,a) = Pr\{a=a_t \mid s = s_t\}$
- Policy kann eine Tabelle oder komplexe Funktion(neuronale Netze, Entscheidungsbaum) sein



Entscheidungsprozesse

- Markov Decision Process

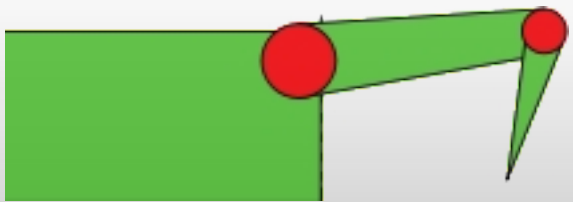
- die Wahrscheinlichkeit für das Erreichen eines bestimmten Nachfolgezustandes ist nur vom aktuellen Zustand und der gewählten Aktion abhängig



Entscheidungsprozesse

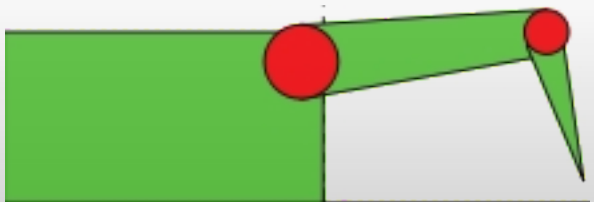
- Entscheidungen werden auf Grund des aktuellen Zustandes getroffen

- wird bei Backgammon oder Schach verwendet



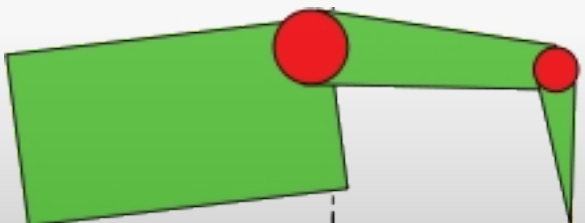
Belohnung

- Verusch der Maximierung der Belohnungen
  - r_i ist die Belohnung für die Aktion a_i im Zustand s_i.



Belohnung

- bei episodischen Aufgaben (Endzustände) einfach
  - → Summe aller folgenden Belohnungen
  - $r_0 + r_1 + r_2 + \ldots$

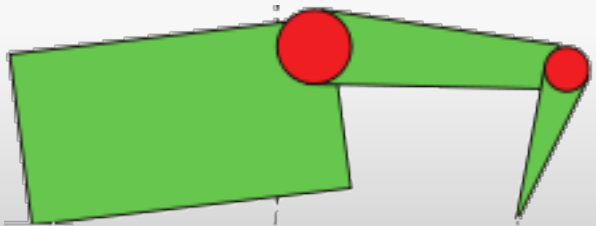


Belohnung

- bei kontinuierlichen Aufgaben schwierig

  --> Belohnung würde ins unendliche wachsen

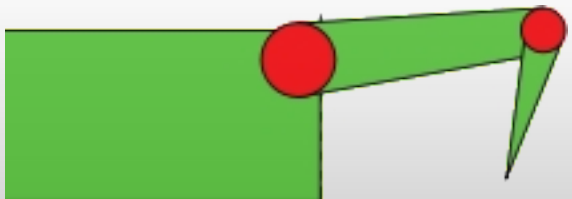
- Ziel: Maximiere die Summe der erhaltenen Belohnungen
  Max: $r_0 + yr_1 + y^2r_2 + \ldots$ mit $0 \leq y < 1$
- Diskontierungsfaktor y



Wertefunktion

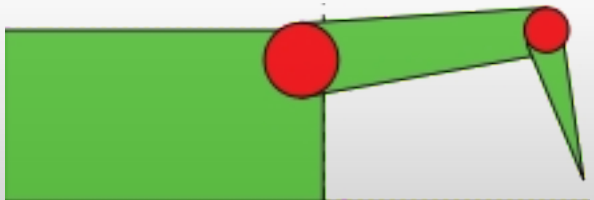
- V(s) wird benutzt um den Wert einer Aktion oder eines Zustandes abzuschätzen

- Benutzt die Belohnung als Abschätzung



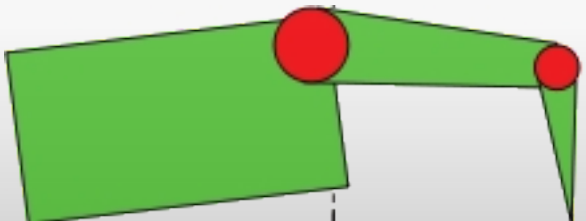
Laufen

- Anforderungen
  - Stabilität
  - Flüssiger Bewegungsablauf
  - Kontinuierliches Laufen
  - Geschwindigkeit
  - sanftes Wechseln der Laufgeschwindigkeiten



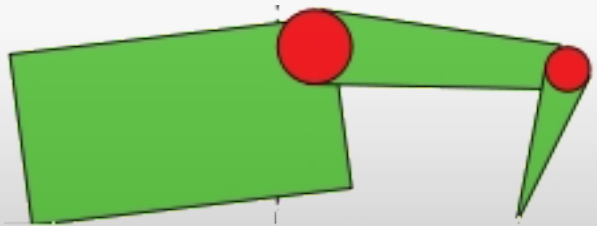
Laufbeschreibung

- Mögliche Phasen
  - Fuß verläßt Boden
  - Fuß am höchsten Punkt
  - Fuß berührt Boden
  - Beide Füße am Boden
- Darstellung von Bewegungen z. B. durch polynomiale, sinusförmige oder Spline-Funktionen differenzierbar, zweite Ableitung stetig



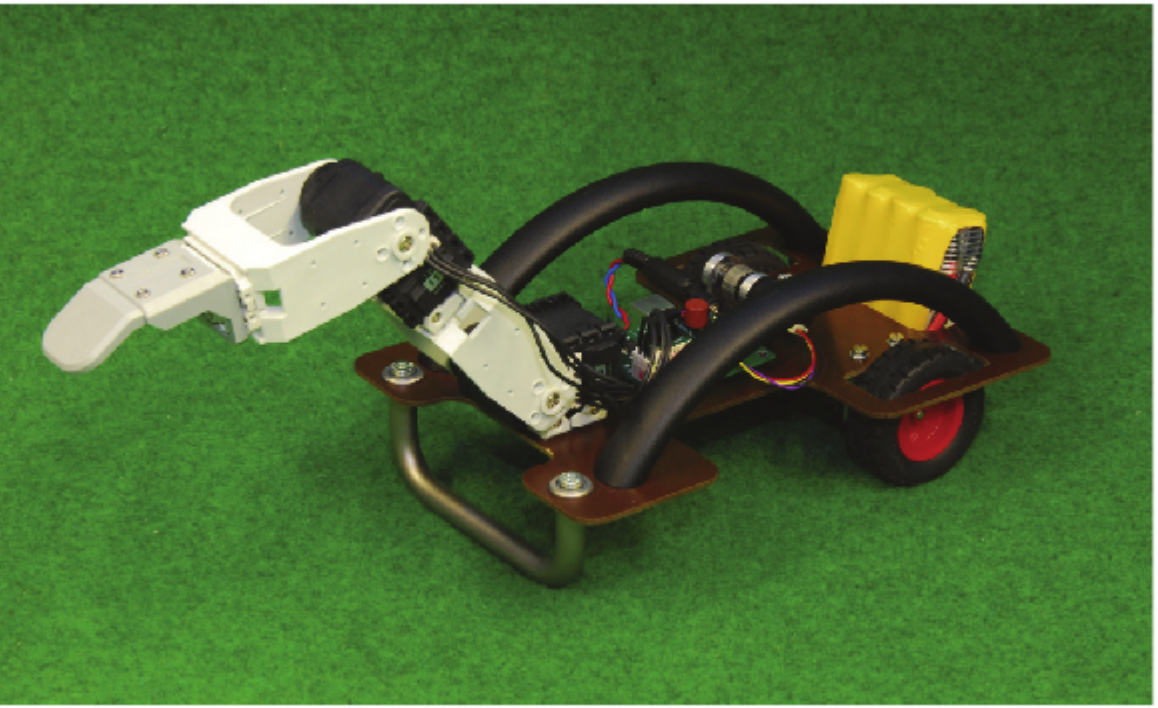
Warum Reinforcement Learning

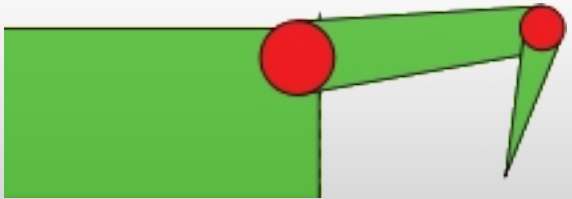
- Viele mögliche Zustände → komplexes Modell
- Nicht modellierbare Umwelteinflüsse
- Zu großer Zustands-/Aktionsraum → wenig Platz im Roboter → alles müsste online geschehen



Beispiel: The Crawler

- Arbeitet mit einem Markov Decision Process (MDP)





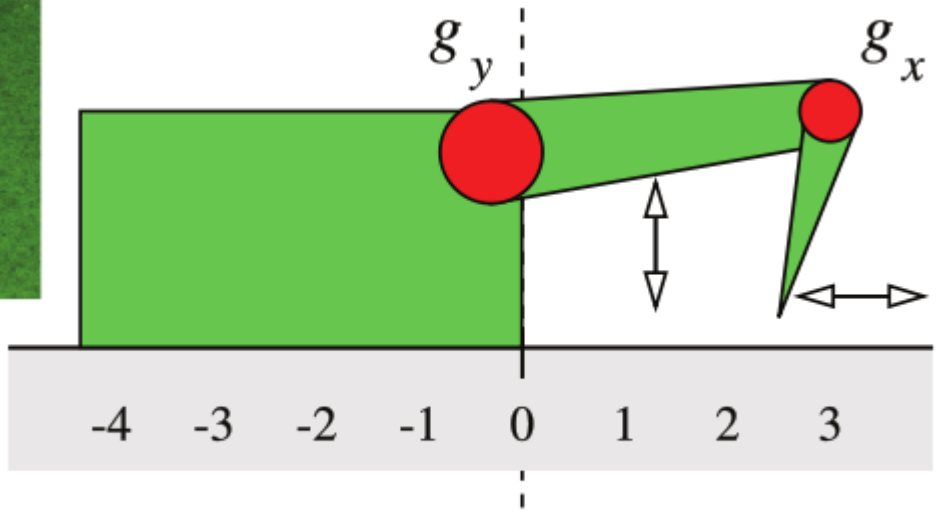
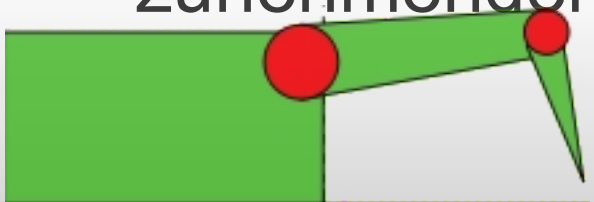


Figure 1: The robot with its two joints g_x and g_y.

Beispiel: The Crawler

- Roboter kennt seine Umwelt nicht
- Tradofff:
  - long-term optimization vs. short-term optimization
- Lösung: epsilon-Greedy-exploration
  - Zufällige Wahl einer Aktion mit einer Wkt. ε
  - ε meist am Anfang hoch und nimmt mit zunehmender Zeit ab



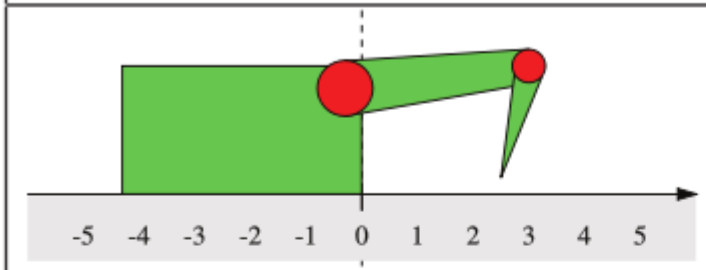
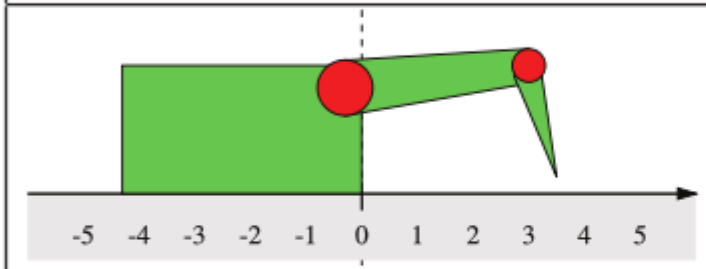
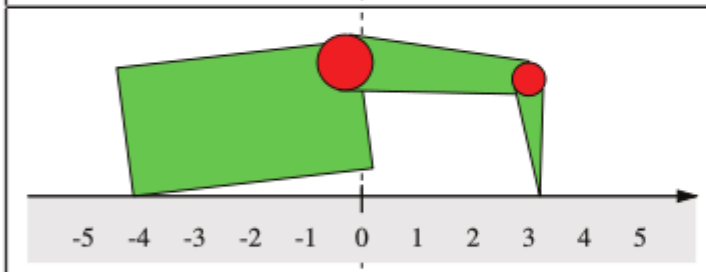
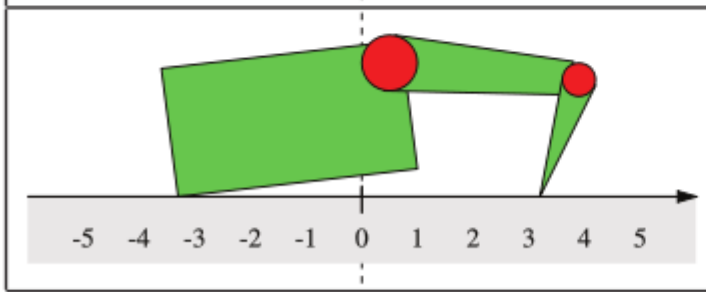
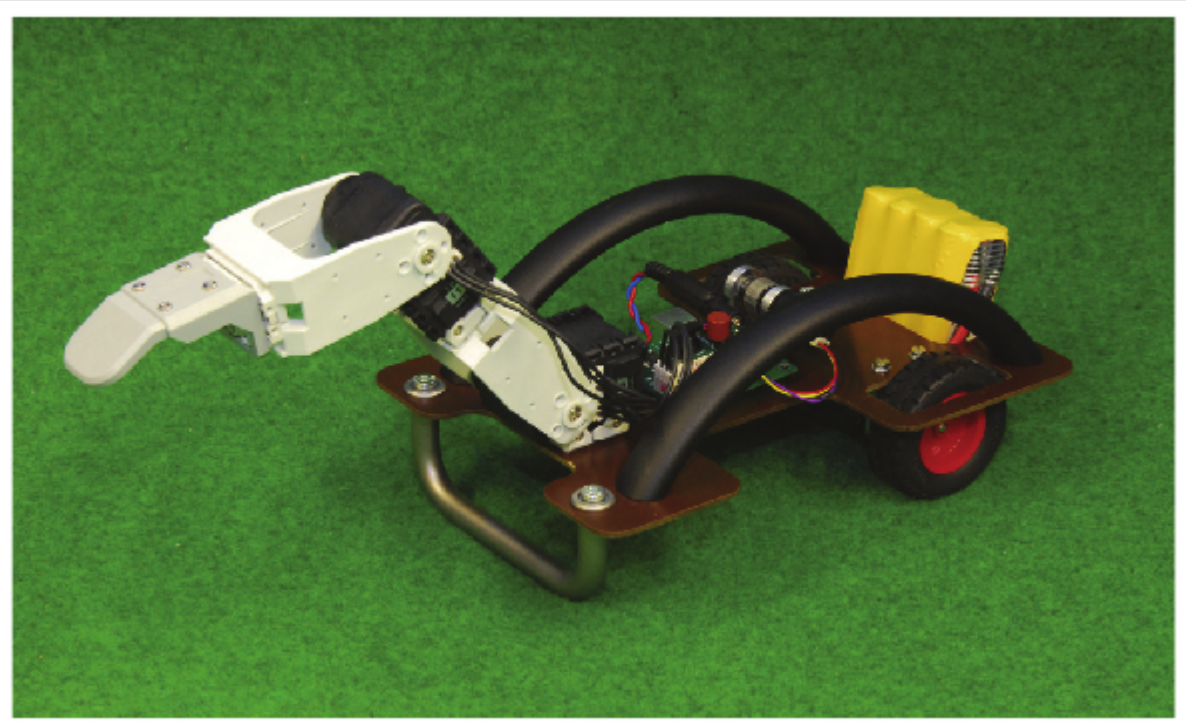
robot	time t	state g_y	state g_x	reward x	action a_t
	0	up	left	0	right
	1	up	right	0	down
	2	down	right	0	left
	3	down	left	1	up

Table 1: Four steps of a simple cyclic forward walking policy.

Algorithm 1 VALUEITERATION ON ROBOT

1: Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}$
2: Initialize $\mathcal{R}^a_{ss'}$ arbitrarily, e.g., $r(s,a) = 0$, for all $r \in \mathcal{R}^a_{ss'}$
3: $state \leftarrow (g_x = 1, g_y = 1)$
4: **loop**
5: $\quad \xi \leftarrow \text{rand}(0..1)$
6: $\quad$ **if** $\xi < \varepsilon$ **then**
7: $\quad\quad a \leftarrow \text{rand}(\mathcal{A}(state))$
8: $\quad$ **else**
9: $\quad\quad a \leftarrow \text{argmax}_a \mathcal{R}^a_{ss'} + \gamma V(s')$
10: $\quad$ **end if**
11: $\quad successorState \leftarrow \delta(state, a)$
12: $\quad$ observe $r(state, a)$ and update $\mathcal{R}^a_{ss'}$
13: $\quad$ **for all** $s \in \mathcal{S}$ **do**
14: $\quad\quad V(s) \leftarrow \max_{a \in \mathcal{A}(s)} r(s,a) + \gamma V(\delta(s,a))$
15: $\quad\quad \pi(s) \leftarrow \text{argmax}_{a \in \mathcal{A}(s)} r(s,a) + \gamma V(\delta(s,a))$
16: $\quad$ **end for**
17: $\quad state \leftarrow successorState$
18: **end loop**

- http://www.youtube.com/watch?v=qsvVmUuhCZQ



Quellen

- The Crawler, A Class Room Demonstrator for Reinforcement Learning, Michel Tokic, Wolfgang Ertel and Joachim Fessler
- Erlernen einer Bewegungssteuerung für ein autonomes Prallluftschift mittels Reinforcement Learning,Jonas Sternisko
- Reinforcement Learning für Laufroboter, Markus Schneider