

# **Forward Model Learning für Prädiktionsbasierte Suche in Unbekannten Umgebungen**

**Anwendungen in Autonomous Game-Playing und Motion Control**

**Alexander Dockhorn**

a.dockhorn@qmul.ac.uk

Queen Mary University of London

School of Electronic Engineering and Computer Science  
Game AI Research Group

## Motivation

Computational Intelligence bezieht sich auf die Fähigkeit eines Computers, eine bestimmte Aufgabe aus Daten oder experimentellen Beobachtungen zu lernen.

# Motivation

Computational Intelligence bezieht sich auf die Fähigkeit eines Computers, eine bestimmte Aufgabe aus Daten oder experimentellen Beobachtungen zu lernen.

- Steuerung von selbstfahrenden Autos und Einparkhilfen



[1]

---

[1] [https://commons.wikimedia.org/wiki/File:Tesla\\_Autopilot\\_Engaged\\_in\\_Model\\_X.jpg](https://commons.wikimedia.org/wiki/File:Tesla_Autopilot_Engaged_in_Model_X.jpg)

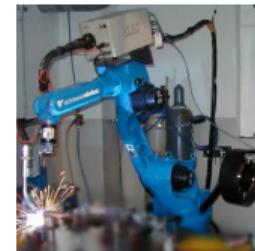
# Motivation

Computational Intelligence bezieht sich auf die Fähigkeit eines Computers, eine bestimmte Aufgabe aus Daten oder experimentellen Beobachtungen zu lernen.

- Steuerung von selbstfahrenden Autos und Einparkhilfen
- Automatisieren einer Produktionspipeline



[1]



[2]

---

[1] [https://commons.wikimedia.org/wiki/File:Tesla\\_Autopilot\\_Engaged\\_in\\_Model\\_X.jpg](https://commons.wikimedia.org/wiki/File:Tesla_Autopilot_Engaged_in_Model_X.jpg)

[2] <https://commons.wikimedia.org/wiki/File:Industrieroboter.jpg>

# Motivation

Computational Intelligence bezieht sich auf die Fähigkeit eines Computers, eine bestimmte Aufgabe aus Daten oder experimentellen Beobachtungen zu lernen.

- Steuerung von selbstfahrenden Autos und Einparkhilfen
- Automatisieren einer Produktionspipeline

## Mögliche Probleme:

- Lernumgebungen sind aufwändig einzurichten und auszuwerten
- Ausfall des Algorithmus hat erhebliche Kosten (z. B. Autounfall)

# Motivation

Computational Intelligence bezieht sich auf die Fähigkeit eines Computers, eine bestimmte Aufgabe aus Daten oder experimentellen Beobachtungen zu lernen.

- Steuerung von selbstfahrenden Autos und Einparkhilfen
- Automatisieren einer Produktionspipeline

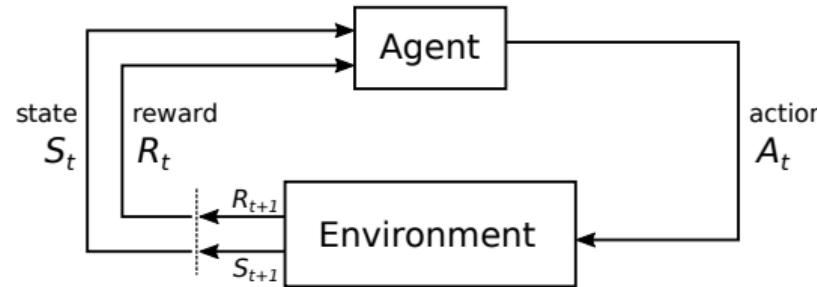
## Mögliche Probleme:

- Lernumgebungen sind aufwändig einzurichten und auszuwerten
- Ausfall des Algorithmus hat erhebliche Kosten (z. B. Autounfall)

Spiele können Simulationen von Aufgaben der realen Welt darstellen

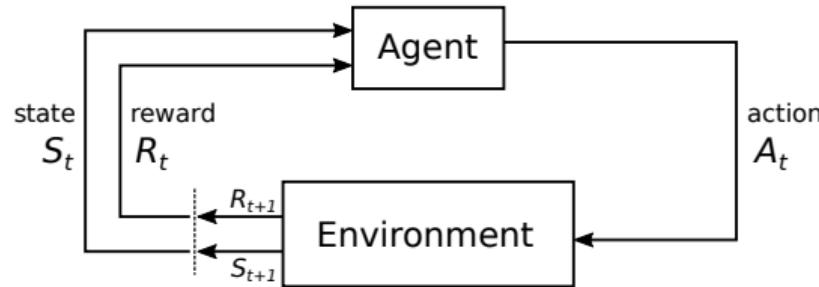
- quantifizierbares Ziel, unterschiedliche Schwierigkeit und große Datensätze
- digitale Spiele sind für Computer voll zugänglich

# Agent-Environment Interface



Ein allgemeines Framework für das Lernen anhand von Interaktion um ein Ziel zu erreichen:

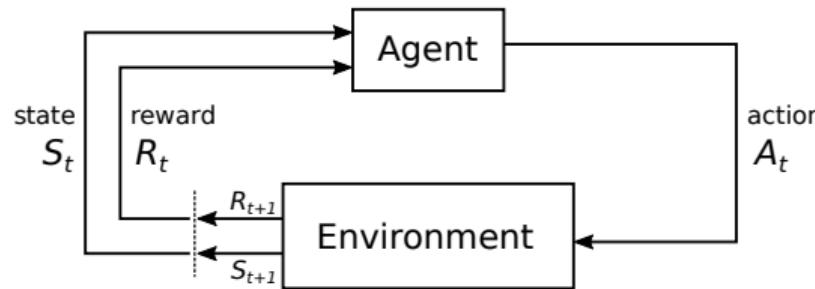
# Agent-Environment Interface



Ein allgemeines Framework für das Lernen anhand von Interaktion um ein Ziel zu erreichen:

- **Agent:** der Lernende und Entscheidungsträger

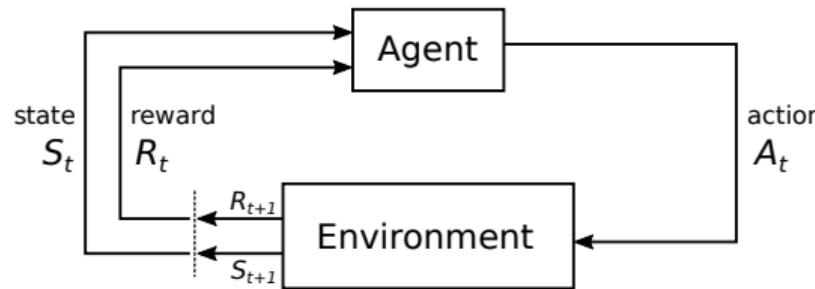
# Agent-Environment Interface



Ein allgemeines Framework für das Lernen anhand von Interaktion um ein Ziel zu erreichen:

- **Agent:** der Lernende und Entscheidungsträger
- **Environment:** Elemente mit denen der Agent interagiert, z. B. ein Spiel

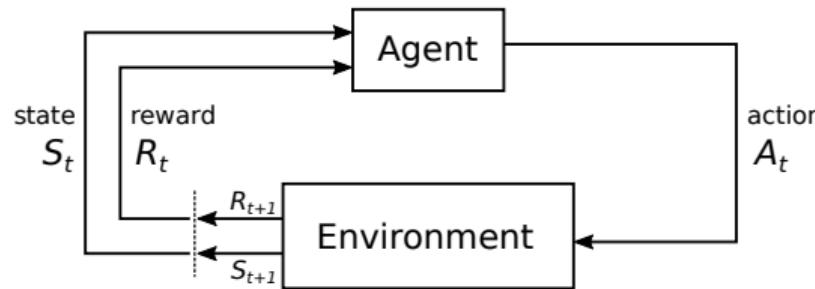
# Agent-Environment Interface



Ein allgemeines Framework für das Lernen anhand von Interaktion um ein Ziel zu erreichen:

- **Agent:** der Lernende und Entscheidungsträger
- **Environment:** Elemente mit denen der Agent interagiert, z. B. ein Spiel
- **Actions:** Agent und Umgebung stehen in ständiger Wechselwirkung zueinander.
  - der Agent führt eine Aktion aus
  - die Umwelt reagiert auf diese Aktionen

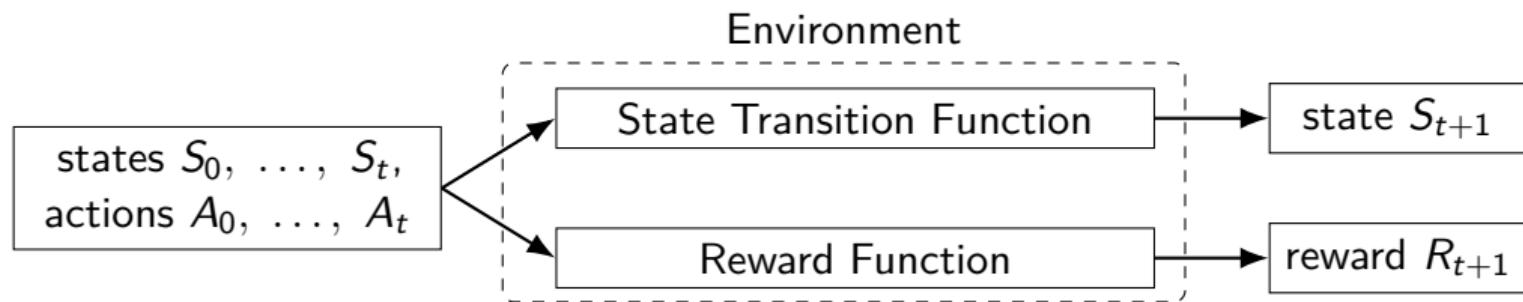
# Agent-Environment Interface



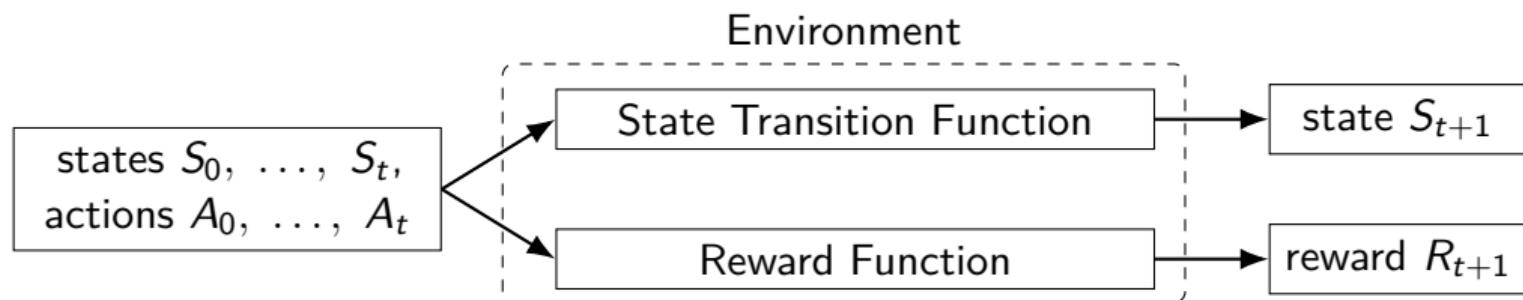
Ein allgemeines Framework für das Lernen anhand von Interaktion um ein Ziel zu erreichen:

- **Agent:** der Lernende und Entscheidungsträger
- **Environment:** Elemente mit denen der Agent interagiert, z. B. ein Spiel
- **Actions:** Agent und Umgebung stehen in ständiger Wechselwirkung zueinander.
  - der Agent führt eine Aktion aus
  - die Umwelt reagiert auf diese Aktionen
- **Reward:** numerische Werte, welche der Agent über die Zeit zu maximieren versucht.

# Komponenten des Environment-Modells



# Komponenten des Environment-Modells



Zustand  $S_t \in \mathcal{S}$  kann durch mehrere Sensoren wahrgenommen werden ( $S_t^{(1)}, S_t^{(2)}, \dots, S_t^{(n)}$ ).

# Methoden der Künstlichen Intelligenz in Spielen

# Methoden der Künstlichen Intelligenz in Spielen

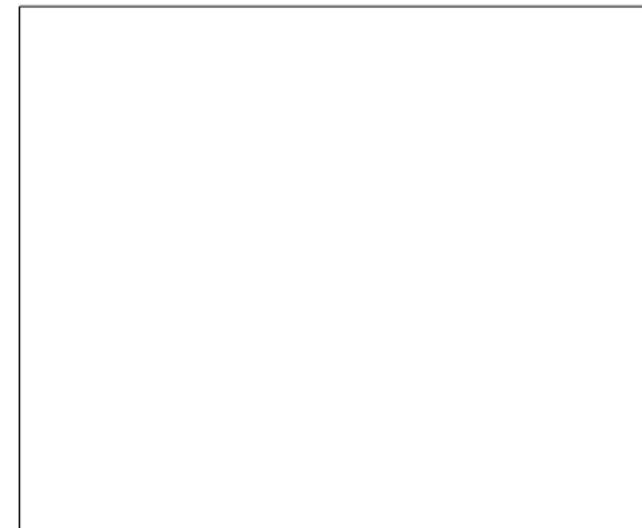
zwei typische Lernansätze:

- lerne die besten Aktionen
- lerne zukünftige Zustände

# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

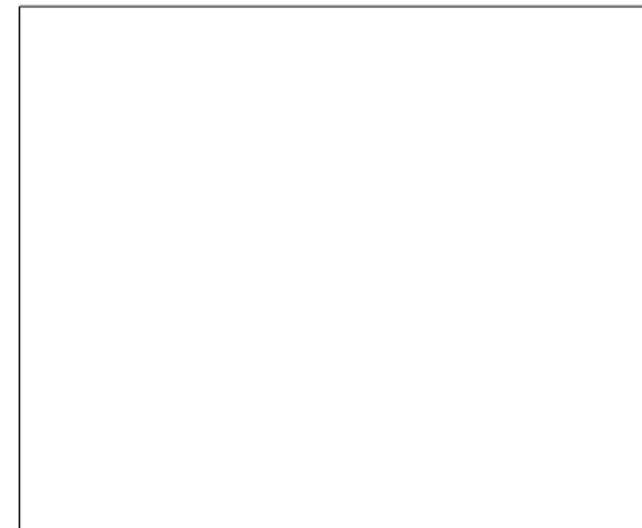
- lerne die besten Aktionen
- lerne zukünftige Zustände



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

- lerne die besten Aktionen
- lerne zukünftige Zustände

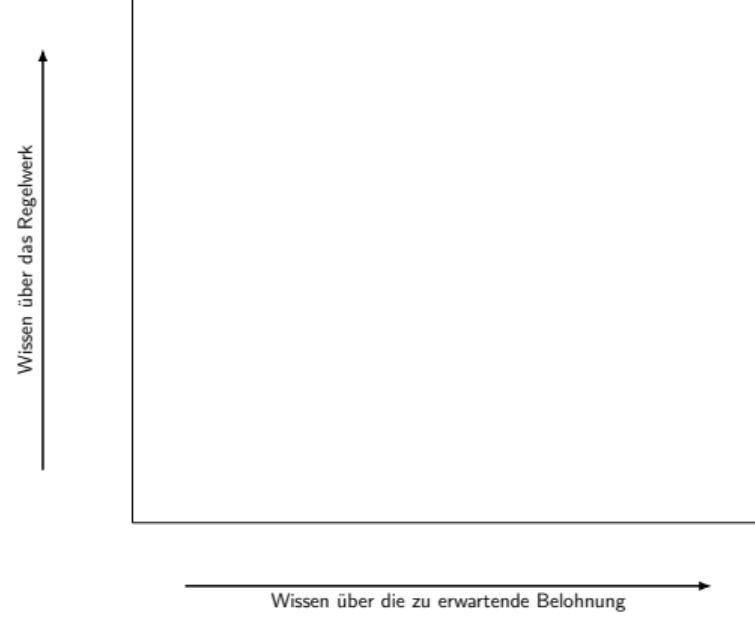


→ Wissen über die zu erwartende Belohnung

# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

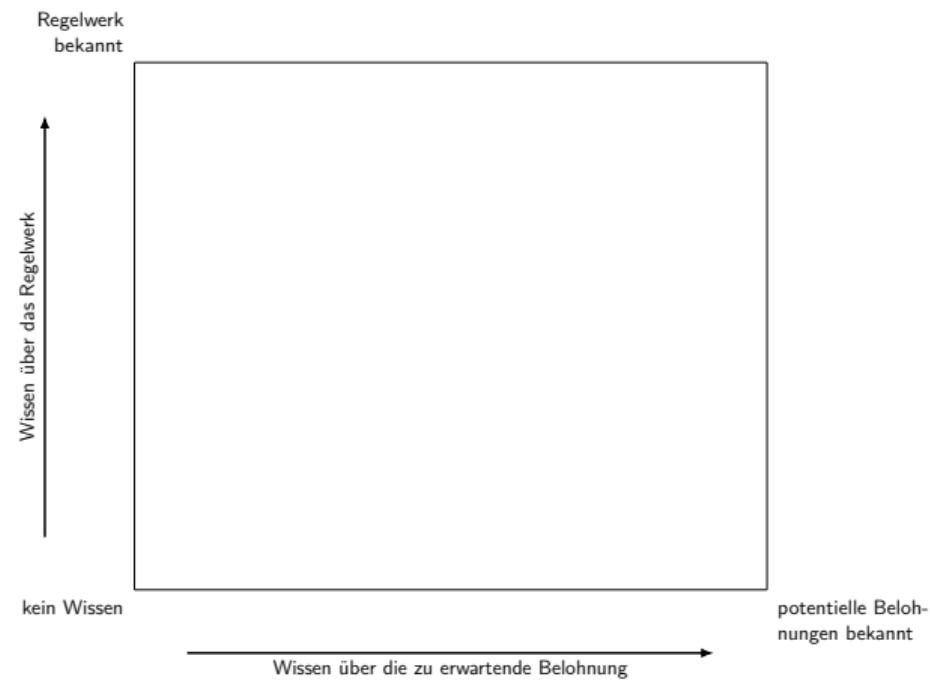
- lerne die besten Aktionen
- lerne zukünftige Zustände



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

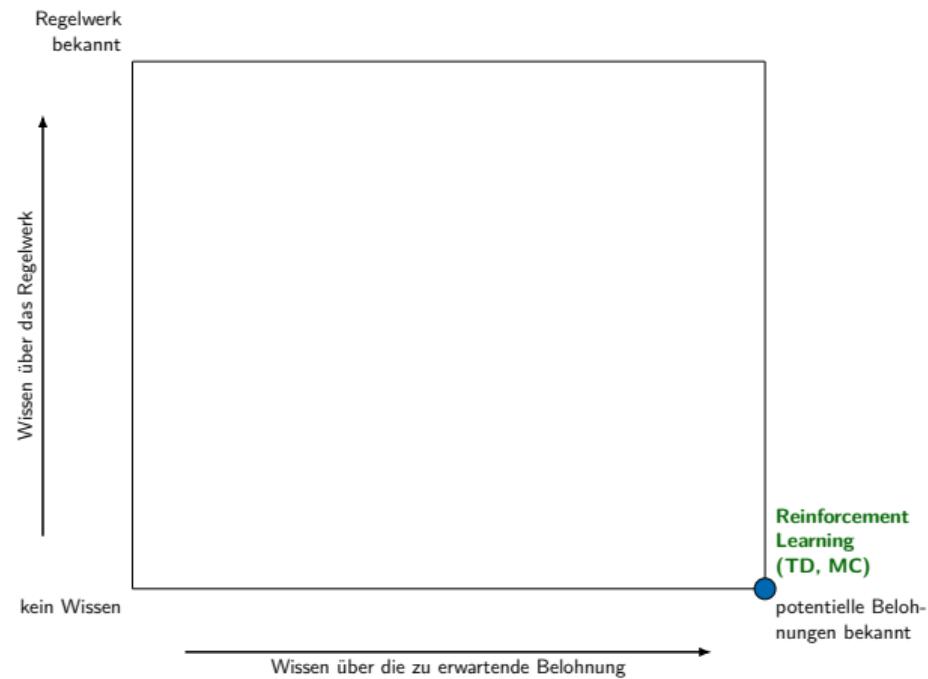
- lerne die besten Aktionen
- lerne zukünftige Zustände



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

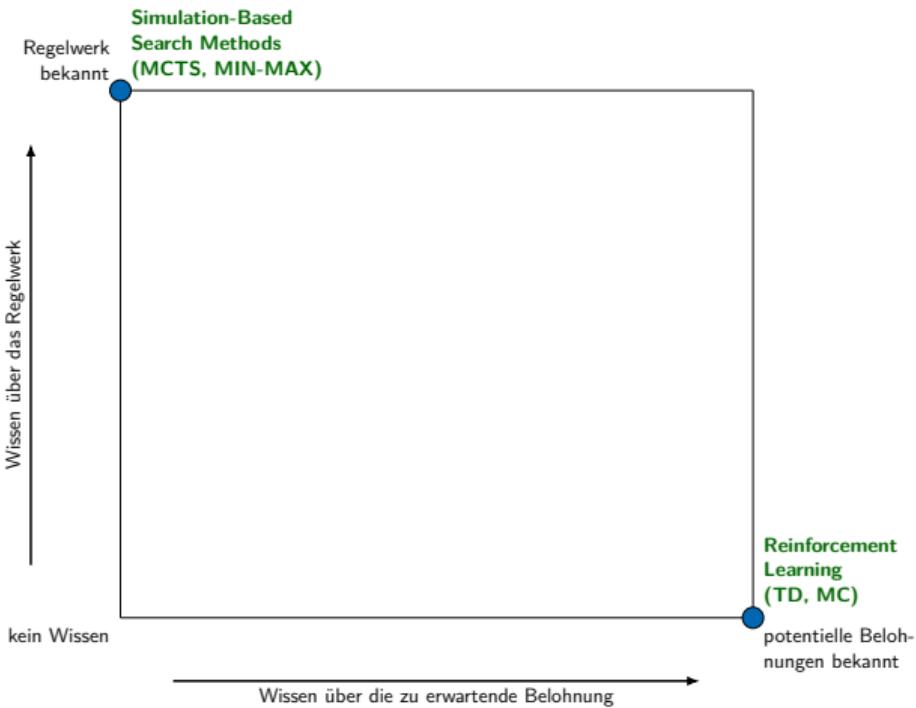
- lerne die besten Aktionen
- lerne zukünftige Zustände



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

- lerne die besten Aktionen
- lerne zukünftige Zustände



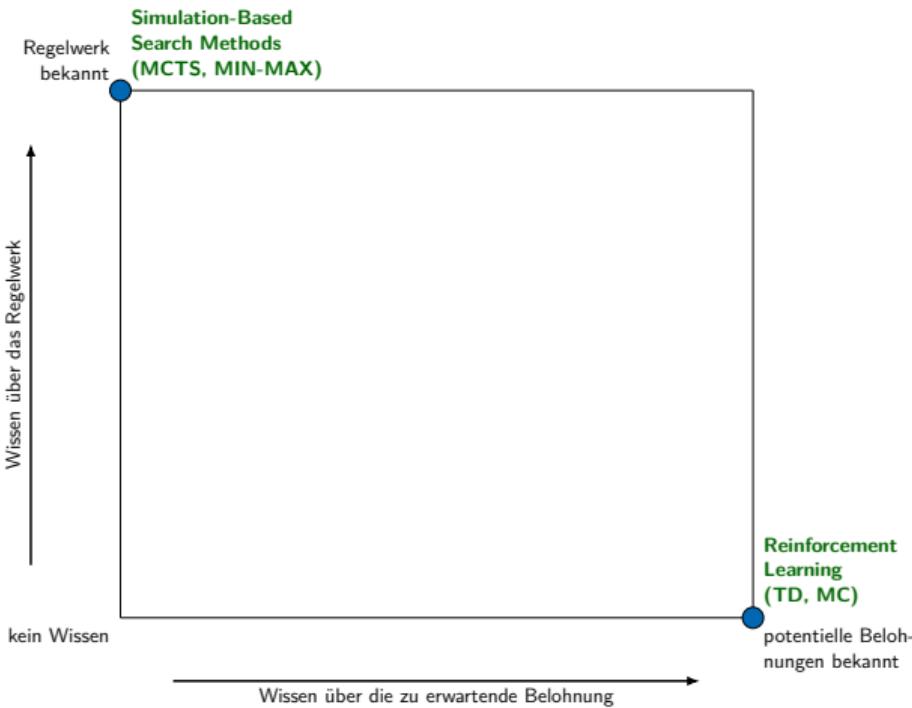
# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

- lerne die besten Aktionen
- lerne zukünftige Zustände

Reinforcement Learning

- Leistung hängt von der verfügbaren Trainingszeit ab



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

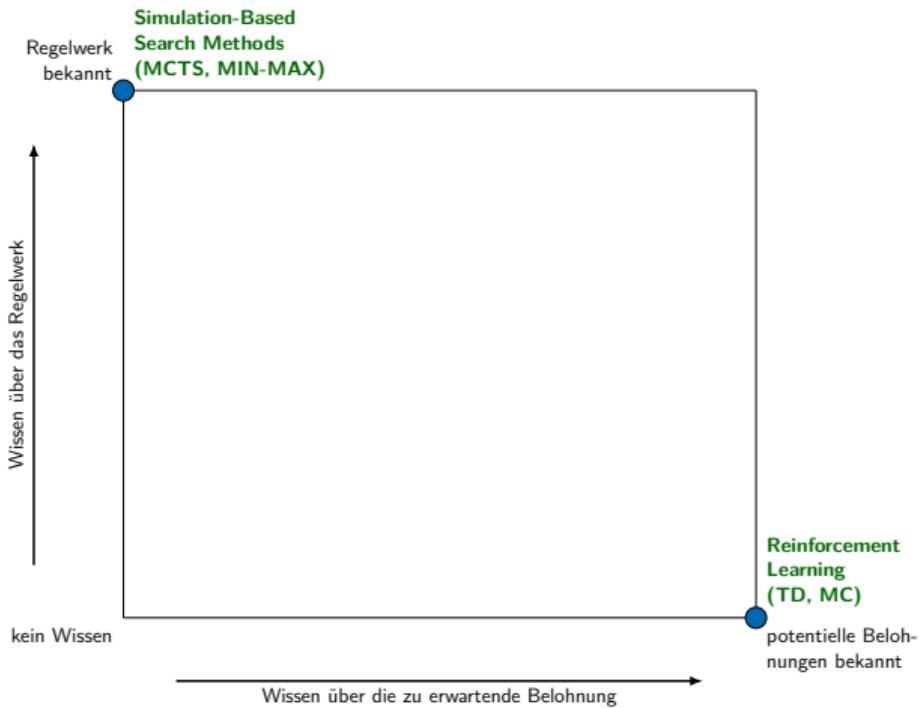
- lerne die besten Aktionen
- lerne zukünftige Zustände

Reinforcement Learning

- Leistung hängt von der verfügbaren Trainingszeit ab

Simulation-based Search

- Leistung hängt von der verfügbaren Rechenzeit während der Auswertung ab



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

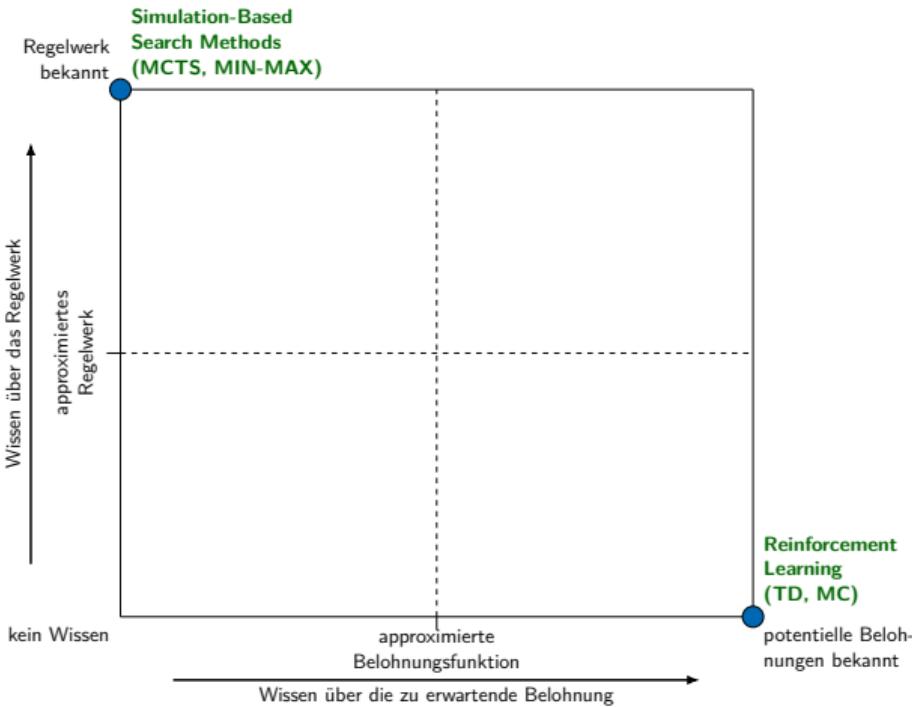
- lerne die besten Aktionen
- lerne zukünftige Zustände

Reinforcement Learning

- Leistung hängt von der verfügbaren Trainingszeit ab

Simulation-based Search

- Leistung hängt von der verfügbaren Rechenzeit während der Auswertung ab



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

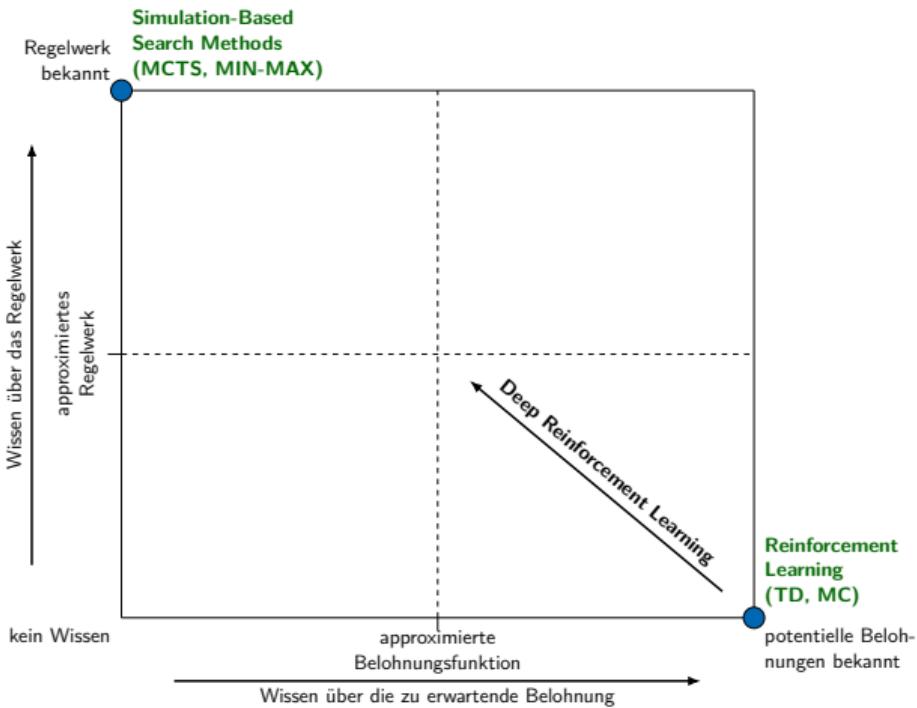
- lerne die besten Aktionen
- lerne zukünftige Zustände

Reinforcement Learning

- Leistung hängt von der verfügbaren Trainingszeit ab

Simulation-based Search

- Leistung hängt von der verfügbaren Rechenzeit während der Auswertung ab



# Methoden der Künstlichen Intelligenz in Spielen

zwei typische Lernansätze:

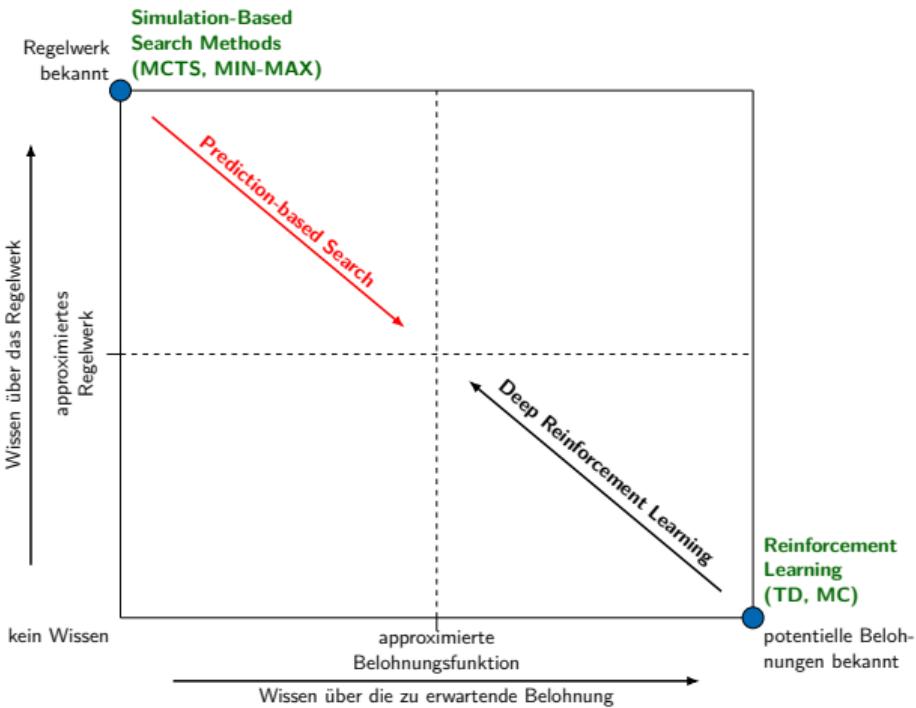
- lerne die besten Aktionen
- lerne zukünftige Zustände

Reinforcement Learning

- Leistung hängt von der verfügbaren Trainingszeit ab

Simulation-based Search

- Leistung hängt von der verfügbaren Rechenzeit während der Auswertung ab



# Simulation-Based Search Algorithmen

# Simulation-Based Search Algorithmen

## Eingabe:

- aktueller Zustand
- Umgebungsmodell

## Ausgabe:

- bestbewertete Aktion

# Simulation-Based Search Algorithmen

aktueller Zustand

## Eingabe:

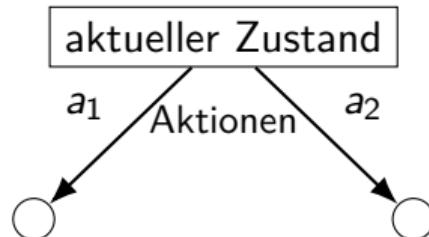
- aktueller Zustand
- Umgebungsmodell

## Ausgabe:

- bestbewertete Aktion

# Simulation-Based Search Algorithmen

Simulation durch  
Umgebungsmodell



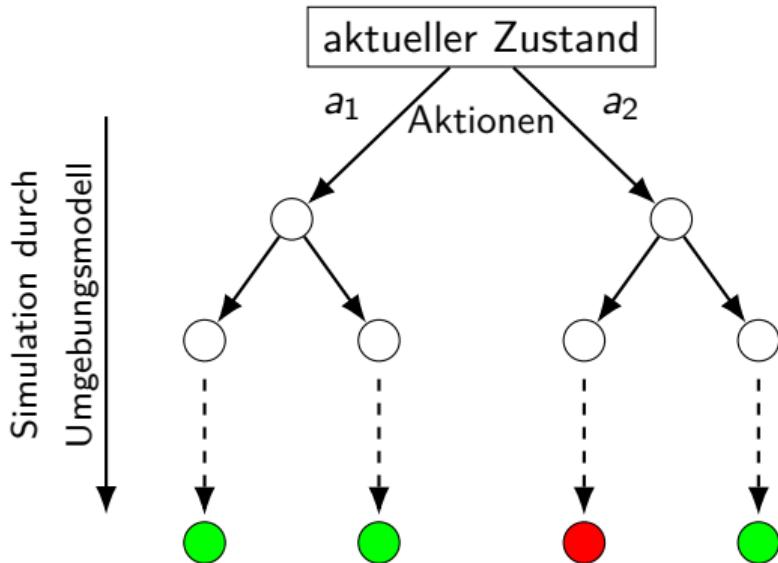
## Eingabe:

- aktueller Zustand
- Umgebungsmodell

## Ausgabe:

- bestbewertete Aktion

# Simulation-Based Search Algorithmen



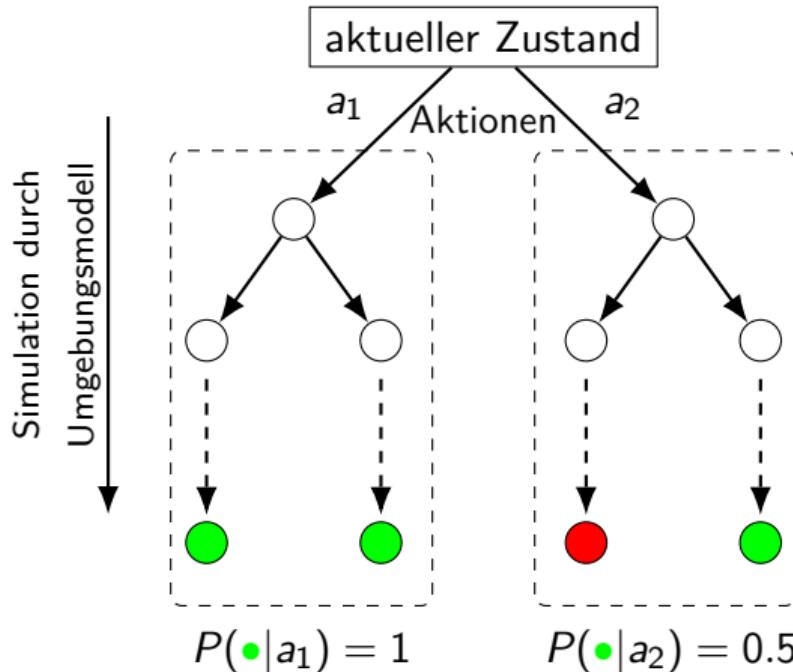
## Eingabe:

- aktueller Zustand
- Umgebungsmodell

## Ausgabe:

- bestbewertete Aktion

# Simulation-Based Search Algorithmen



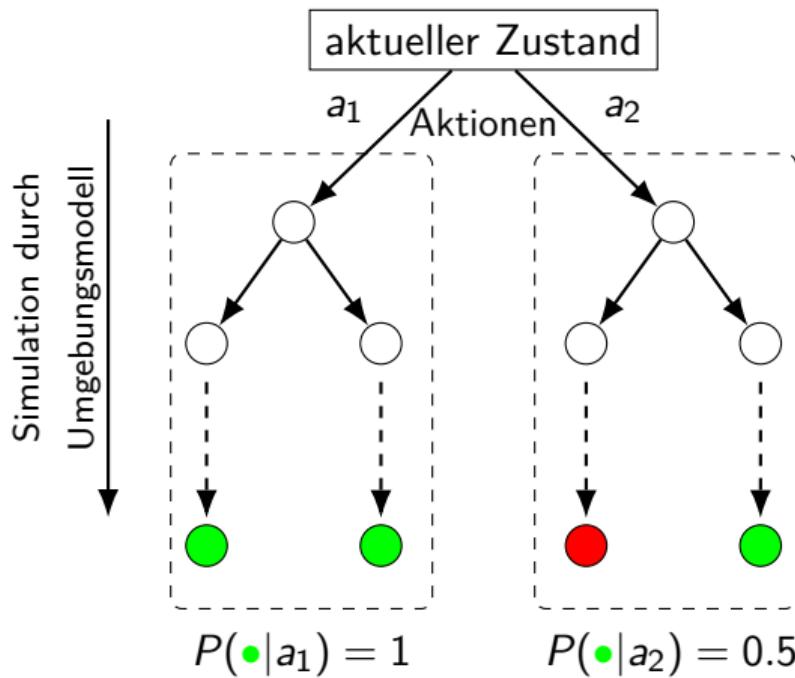
## Eingabe:

- aktueller Zustand
- Umgebungsmodell

## Ausgabe:

- bestbewertete Aktion

# Simulation-Based Search Algorithmen



## Eingabe:

- aktueller Zustand
- Umgebungsmodell

## Ausgabe:

- bestbewertete Aktion

Wie sollen Umgebungen behandelt werden, in denen...

- das Umgebungsmodell unbekannt ist?
- der Zustand nur teilweise beobachtet wird?

# Forward Model Learning

# Forward Model Learning

**Ziel:** Vorhersagen zukünftiger Zustände einer unbekannten Umgebung

# Forward Model Learning

**Ziel:** Vorhersagen zukünftiger Zustände einer unbekannten Umgebung

**Definition: Forward Model**

- Ein Forward Model  $fm$  bildet den Zustand  $S_t$  der Umgebung und die Aktion  $A_t$  des Agenten zum Zeitpunkt  $t$  auf den kommenden Zustand  $S_{t+1}$  der Umgebung ab:

$$fm : (\mathcal{S} \times \mathcal{A}) \rightarrow \mathcal{S} \quad (S_t, A_t) \longmapsto S_{t+1}$$

# Forward Model Learning

**Ziel:** Vorhersagen zukünftiger Zustände einer unbekannten Umgebung

**Definition: Forward Model**

- Ein Forward Model  $fm$  bildet den Zustand  $S_t$  der Umgebung und die Aktion  $A_t$  des Agenten zum Zeitpunkt  $t$  auf den kommenden Zustand  $S_{t+1}$  der Umgebung ab:

$$fm : (\mathcal{S} \times \mathcal{A}) \rightarrow \mathcal{S} \quad (S_t, A_t) \longmapsto S_{t+1}$$

- Diese Definition gilt nur für Umgebungsmodelle, die die Markov-Eigenschaft erfüllen!

# Forward Model Learning

**Ziel:** Vorhersagen zukünftiger Zustände einer unbekannten Umgebung

## Definition: Forward Model

- Ein Forward Model  $fm$  bildet den Zustand  $S_t$  der Umgebung und die Aktion  $A_t$  des Agenten zum Zeitpunkt  $t$  auf den kommenden Zustand  $S_{t+1}$  der Umgebung ab:

$$fm : (\mathcal{S} \times \mathcal{A}) \rightarrow \mathcal{S} \quad (S_t, A_t) \longmapsto S_{t+1}$$

- Diese Definition gilt nur für Umgebungsmodelle, die die Markov-Eigenschaft erfüllen!

## Problem-Kategorisierung:

- beobachtete Interaktionen als Trainingsbeispiele
- Modellbildung als Klassifikations- oder Regressionsproblem

# Decomposed Forward Models<sup>[1]</sup>

## Modellannahmen:

- Sensorwerte werden unabhängig voneinander modelliert

$$\forall i, j \in 1..n : i \neq j \Rightarrow S_{t+1}^{(i)} \perp\!\!\!\perp S_{t+1}^{(j)} \mid S_t, A_t$$

---

[1] Dockhorn, A., Tippelt, T., & Kruse, R. (2018). Model Decomposition for Forward Model Approximation

# Decomposed Forward Models<sup>[1]</sup>

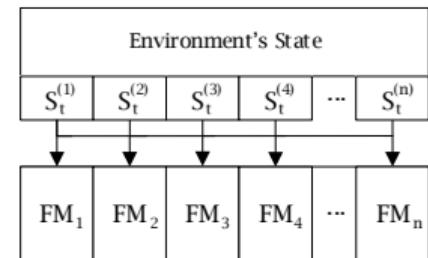
## Modellannahmen:

- Sensorwerte werden unabhängig voneinander modelliert

$$\forall i, j \in 1..n : i \neq j \Rightarrow S_{t+1}^{(i)} \perp\!\!\!\perp S_{t+1}^{(j)} \mid S_t, A_t$$

Lerne ein Teilmodell für jeden beobachtbaren Sensorwert

$$fm_i : (S_t, A_t) \longmapsto S_{t+1}^{(i)} \quad fm_{\delta i} : (S_t, A_t) \longmapsto S_{t+1}^{(i)} - S_t$$



[1] Dockhorn, A., Tippelt, T., & Kruse, R. (2018). Model Decomposition for Forward Model Approximation

# Decomposed Forward Models<sup>[1]</sup>

## Modellannahmen:

- Sensorwerte werden unabhängig voneinander modelliert

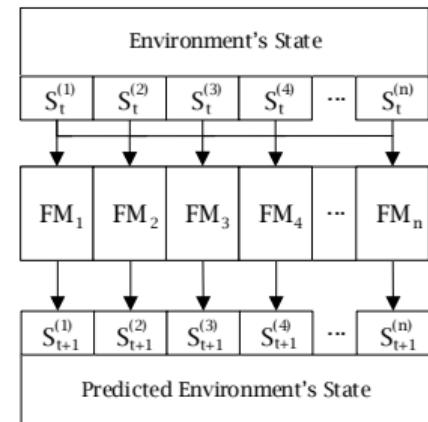
$$\forall i, j \in 1..n : i \neq j \Rightarrow S_{t+1}^{(i)} \perp\!\!\!\perp S_{t+1}^{(j)} \mid S_t, A_t$$

Lerne ein Teilmodell für jeden beobachtbaren Sensorwert

$$fm_i : (S_t, A_t) \mapsto S_{t+1}^{(i)} \quad fm_{\delta i} : (S_t, A_t) \mapsto S_{t+1}^{(i)} - S_t$$

Aggregiere das Ergebnis der einzelnen Sensorwertvorhersagen

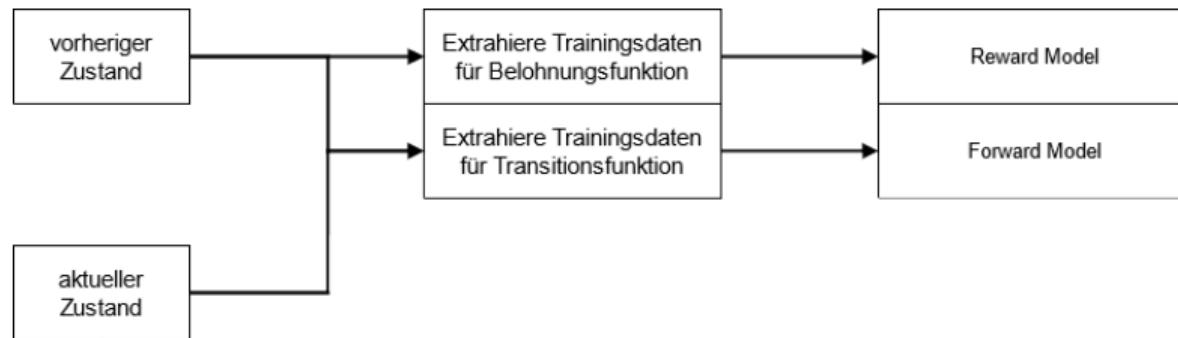
$$\begin{aligned} fm(S_t, A_t) &= (fm_1(S_t, A_t), fm_2(S_t, A_t), \dots, fm_n(S_t, A_t)) \\ &= (S_{t+1}^{(1)}, S_{t+1}^{(2)}, \dots, S_{t+1}^{(n)}) = S_{t+1} \end{aligned}$$



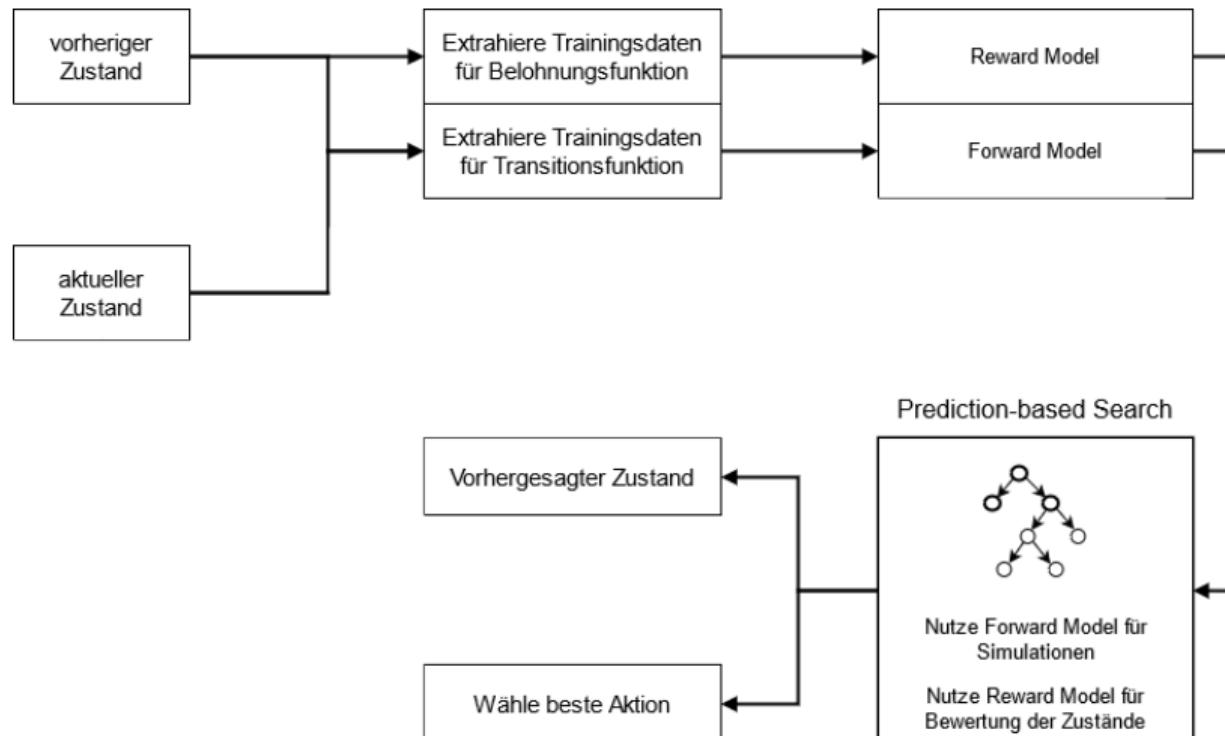
[1] Dockhorn, A., Tippelt, T., & Kruse, R. (2018). Model Decomposition for Forward Model Approximation

# Prediction-Based Search

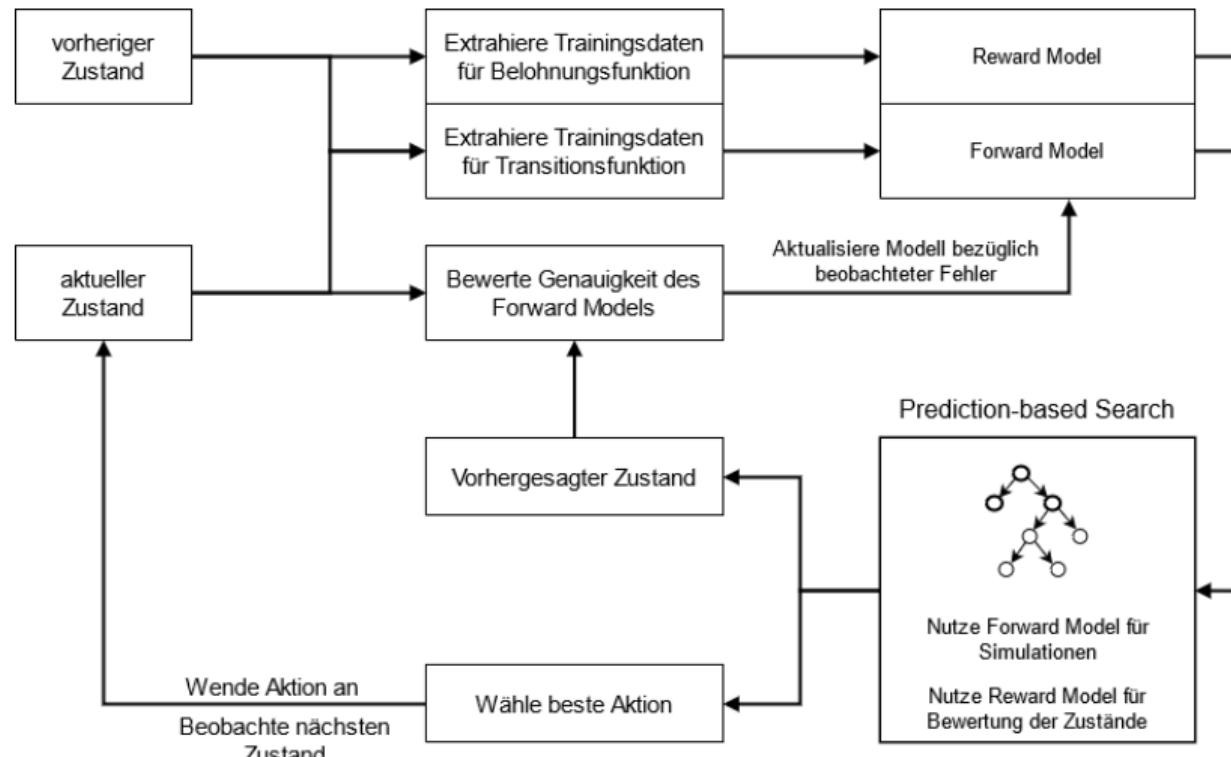
# Prediction-Based Search



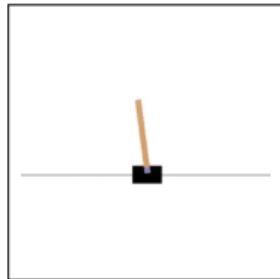
# Prediction-Based Search



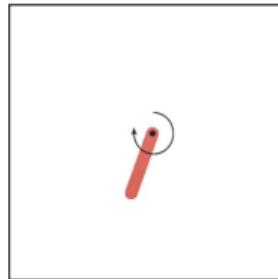
# Prediction-Based Search



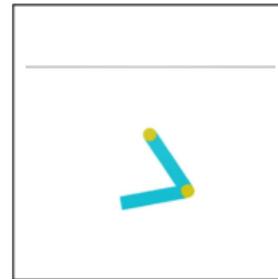
## Anwendung: Continuous Motion Control<sup>[1,2]</sup>



Cart Pole



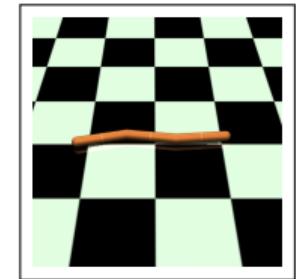
Pendulum



Acrobot



Lunar Lander



Swimmer

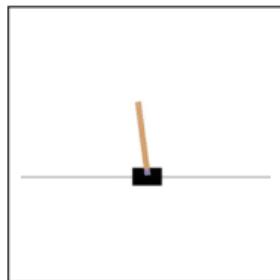
Evaluierung anhand von 5 Motion Control Umgebungen

---

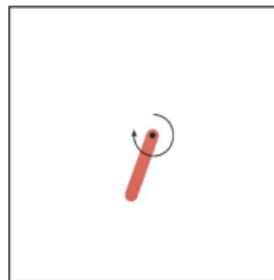
[1] Dockhorn, A., & Kruse, R. (2020). Forward Model Learning for Motion Control Tasks. In: IEEE 10th International Conference on Intelligent Systems

[2] Dockhorn, A., & Kruse, R. (2020). Balancing Exploration And Exploitation in Forward Model Learning. In: Advances in Intelligent Systems Research and Innovation

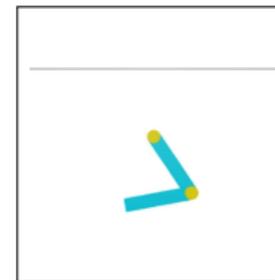
# Anwendung: Continuous Motion Control<sup>[1,2]</sup>



Cart Pole



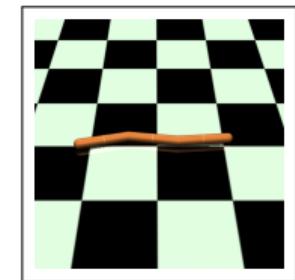
Pendulum



Acrobot



Lunar Lander



Swimmer

Evaluierung anhand von 5 Motion Control Umgebungen

Vergleich mit Deep Reinforcement Learning Methoden:

- schnelleres Training bei gleicher oder besserer Leistung
- verbleibende Trainingszeit steigert die Robustheit

[1] Dockhorn, A., & Kruse, R. (2020). Forward Model Learning for Motion Control Tasks. In: IEEE 10th International Conference on Intelligent Systems

[2] Dockhorn, A., & Kruse, R. (2020). Balancing Exploration And Exploitation in Forward Model Learning. In: Advances in Intelligent Systems Research and Innovation

# Modellieren Lokaler Abhangigkeiten

## Modellannahmen:

- strukturierte Darstellung des Zustands
- Ähnlichkeits- oder Abstandsfunktion für Sensorwerte
- Semantik ist unabhängig vom Sensor-Index

# Modellieren Lokaler Abhangigkeiten

## Modellannahmen:

- strukturierte Darstellung des Zustands
- Ähnlichkeits- oder Abstandsfunktion für Sensorwerte
- Semantik ist unabhängig vom Sensor-Index



Sokoban Game-State

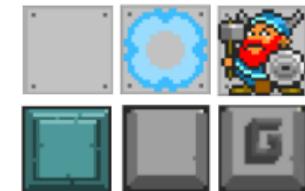
# Modellieren Lokaler Abhangigkeiten

## Modellannahmen:

- strukturierte Darstellung des Zustands
- Ähnlichkeits- oder Abstandsfunktion für Sensorwerte
- Semantik ist unabhängig vom Sensor-Index



Sokoban Game-State



Mögliche Kacheln/Tiles

# Modellieren Lokaler Abhängigkeiten

## Modellannahmen:

- strukturierte Darstellung des Zustands
- Ähnlichkeits- oder Abstandsfunktion für Sensorwerte
- Semantik ist unabhängig vom Sensor-Index

## Tile-based Representation (in Spielen):

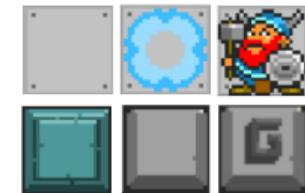
- Zustand als Matrix  $T$  der Größe  $n \times m$

$$T = \begin{bmatrix} T(1, 1) & \dots & T(1, m) \\ \vdots & \ddots & \vdots \\ T(n, 1) & \dots & T(n, m) \end{bmatrix}$$

- $T(x, y)$  spezifiziert das Tile an der Position  $(x, y)$



Sokoban Game-State



Mögliche Kacheln/Tiles

## Local Transition Function

Zerlege das Forward Model in ein Teilmodell pro Tile:

## Local Transition Function

Zerlege das Forward Model in ein Teilmodell pro Tile:

$$_{x,y}^{fm} : \left( N(x,y)_t, A_t \right) \longmapsto T(x,y)_{t+1}$$

- $N(x,y)_t$  beschreibt die lokale Nachbarschaft des Tiles  $T(x,y)$  zum Zeitpunkt  $t$
- enthält jedes Tile mit einem Abstand kleiner einem Schwellenwert

## Local Transition Function

Zerlege das Forward Model in ein Teilmodell pro Tile:

$$fm_{x,y} : \left( N(x,y)_t, A_t \right) \longmapsto T(x,y)_{t+1}$$

- $N(x,y)_t$  beschreibt die lokale Nachbarschaft des Tiles  $T(x,y)$  zum Zeitpunkt  $t$
- enthält jedes Tile mit einem Abstand kleiner einem Schwellenwert



Game-State von Sokoban

# Local Transition Function

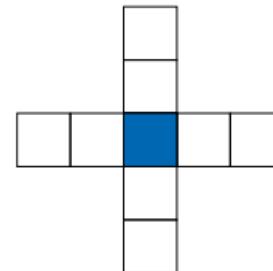
Zerlege das Forward Model in ein Teilmodell pro Tile:

$$fm_{x,y} : (N(x,y)_t, A_t) \longmapsto T(x,y)_{t+1}$$

- $N(x,y)_t$  beschreibt die lokale Nachbarschaft des Tiles  $T(x,y)$  zum Zeitpunkt  $t$
- enthält jedes Tile mit einem Abstand kleiner einem Schwellenwert



Game-State von Sokoban



Lokale Nachbarschaft

# Local Transition Function

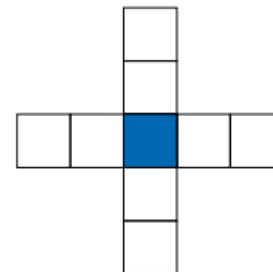
Zerlege das Forward Model in ein Teilmodell pro Tile:

$$fm_{x,y} : \left( N(x,y)_t, A_t \right) \longmapsto T(x,y)_{t+1}$$

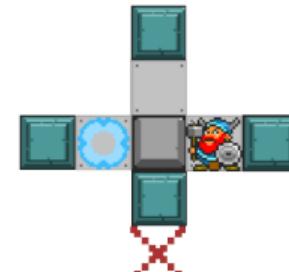
- $N(x,y)_t$  beschreibt die lokale Nachbarschaft des Tiles  $T(x,y)$  zum Zeitpunkt  $t$
- enthält jedes Tile mit einem Abstand kleiner einem Schwellenwert



Game-State von Sokoban



Lokale Nachbarschaft



Extrahiertes Muster

# Local Transition Function

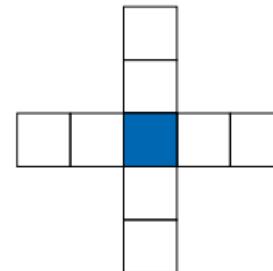
Zerlege das Forward Model in ein Teilmodell pro Tile:

$$fm_{x,y} : \left( N(x,y)_t, A_t \right) \longmapsto T(x,y)_{t+1}$$

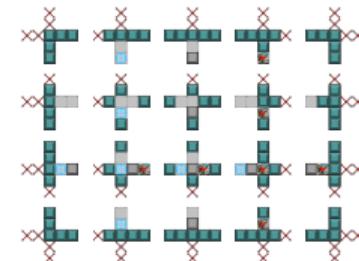
- $N(x,y)_t$  beschreibt die lokale Nachbarschaft des Tiles  $T(x,y)$  zum Zeitpunkt  $t$
- enthält jedes Tile mit einem Abstand kleiner einem Schwellenwert



Game-State von Sokoban



Lokale Nachbarschaft



Muster pro Tile

## Local Forward Model

Vorhersage des nächsten Zustands durch die separate  
Vorhersage jedes Tiles

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1, 1), A_t) & \dots & fm_{1,m}(N(1, m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n, 1), A_t) & \dots & fm_{n,m}(N(n, m), A_t) \end{bmatrix}$$

## Local Forward Model

Vorhersage des nächsten Zustands durch die separate  
Vorhersage jedes Tiles

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1, 1), A_t) & \dots & fm_{1,m}(N(1, m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n, 1), A_t) & \dots & fm_{n,m}(N(n, m), A_t) \end{bmatrix}$$

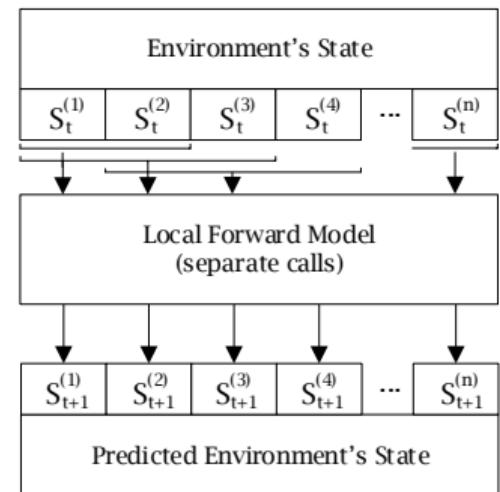
Bei gleicher Semantik unabhängig von der Position ⇒  
lerne nur ein Modell.

# Local Forward Model

Vorhersage des nächsten Zustands durch die separate Vorhersage jedes Tiles

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1,1), A_t) & \dots & fm_{1,m}(N(1, m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n,1), A_t) & \dots & fm_{n,m}(N(n, m), A_t) \end{bmatrix}$$

Bei gleicher Semantik unabhängig von der Position  $\Rightarrow$   
 lerne nur ein Modell.



# Local Forward Model

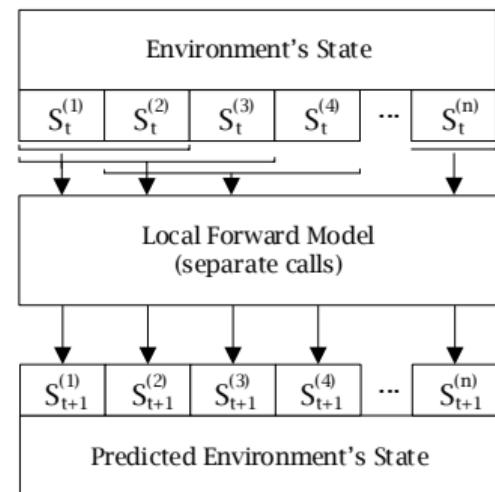
Vorhersage des nächsten Zustands durch die separate Vorhersage jedes Tiles

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1, 1), A_t) & \dots & fm_{1,m}(N(1, m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n, 1), A_t) & \dots & fm_{n,m}(N(n, m), A_t) \end{bmatrix}$$

Bei gleicher Semantik unabhängig von der Position  $\Rightarrow$   
 lerne nur ein Modell.

**Vorteil:** höhere Stichprobeneffizienz

- jeder Zustandsübergang besteht aus einem beobachteten Muster pro Tile (insgesamt:  $n \times m$ )



# Objektbasierte Zustandsrepräsentation

## Modellannahmen:

- Spielkomponenten werden als unabhängig agierende Einheiten betrachtet
- ähnlich Objekte zeigen ein ähnliches Verhalten

# Objektbasierte Zustandsrepräsentation

## Modellannahmen:

- Spielkomponenten werden als unabhängig agierende Einheiten betrachtet
- ähnlich Objekte zeigen ein ähnliches Verhalten

## Object-based Representation

- der Zustand besteht aus mehreren Entitäten, von denen jeweils mehrere Attribute beobachtet werden können

# Objektbasierte Zustandsrepräsentation

## Modellannahmen:

- Spielkomponenten werden als unabhängig agierende Einheiten betrachtet
- ähnlich Objekte zeigen ein ähnliches Verhalten

## Object-based Representation

- der Zustand besteht aus mehreren Entitäten, von denen jeweils mehrere Attribute beobachtet werden können

$$\begin{aligned}\mathcal{S} &= (S^{(1)}, S^{(2)}, \dots, S^{(n)}) \\ &= (\underbrace{S^{(1,1)}, \dots, S^{(1,i)}}, \underbrace{S^{(2,1)}, \dots, S^{(2,j)}}, \dots, \underbrace{S^{(m,1)}, \dots, S^{(m,k)}})_{\text{Objekt } 1} \quad \text{Objekt } 2 \quad \text{Objekt } m\end{aligned}$$

# Objektbasierte Zustandsrepräsentation

## Modellannahmen:

- Spielkomponenten werden als unabhängig agierende Einheiten betrachtet
- ähnlich Objekte zeigen ein ähnliches Verhalten

## Object-based Representation

- der Zustand besteht aus mehreren Entitäten, von denen jeweils mehrere Attribute beobachtet werden können

$$\begin{aligned}\mathcal{S} &= (S^{(1)}, S^{(2)}, \dots, S^{(n)}) \\ &= (\underbrace{S^{(1,1)}, \dots, S^{(1,i)}}, \underbrace{S^{(2,1)}, \dots, S^{(2,j)}}, \dots, \underbrace{S^{(m,1)}, \dots, S^{(m,k)}})_{\text{Objekt } 1} \quad \text{Objekt } 2 \quad \text{Objekt } m\end{aligned}$$

Erstellen eines Teilmodells für jedes Objekt oder jeden Objekttyp.

## Object-based Forward Model

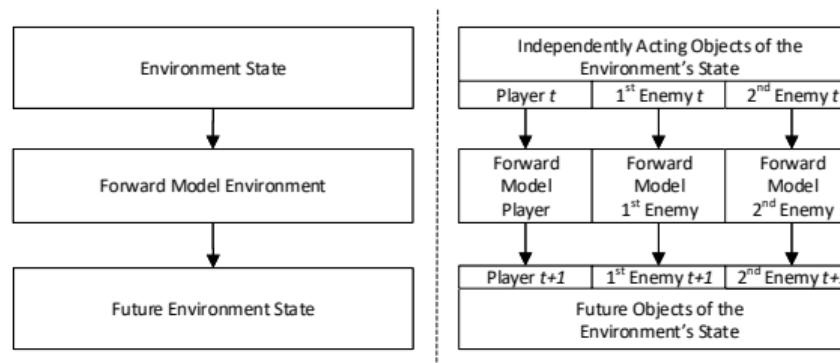
Aggregiere die Vorhersage für jedes Objekt und die zugehörigen Sensorwerte:

$$fm(S_t, A_t) = (f_m^1(S_t, A_t), \dots, f_m^n(S_t, A_t))$$

# Object-based Forward Model

Aggregiere die Vorhersage für jedes Objekt und die zugehörigen Sensorwerte:

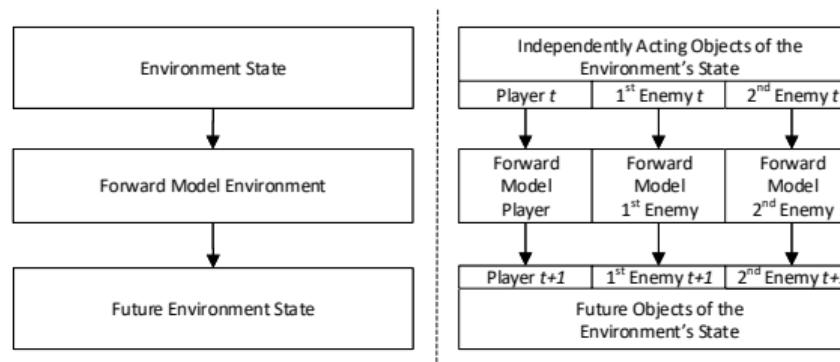
$$fm(S_t, A_t) = (f_m^1(S_t, A_t), \dots, f_m^n(S_t, A_t))$$



# Object-based Forward Model

Aggregiere die Vorhersage für jedes Objekt und die zugehörigen Sensorwerte:

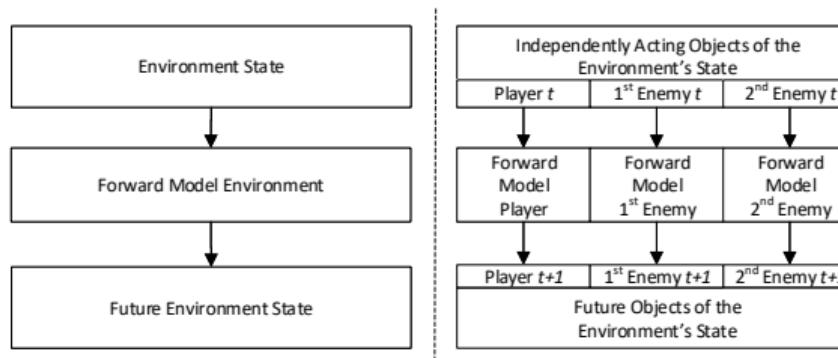
$$\begin{aligned}
 fm(S_t, A_t) &= (fm_1(S_t, A_t), \dots, fm_n(S_t, A_t)) \\
 &= ((fm_{1,1}((S_t^{(1,1)}, \dots, S_t^{(1,k)}), A_t), \dots, fm_{m,k}((S_t^{(m,1)}, \dots, S_t^{(m,k')}), A_t)))
 \end{aligned}$$



# Object-based Forward Model

Aggregiere die Vorhersage für jedes Objekt und die zugehörigen Sensorwerte:

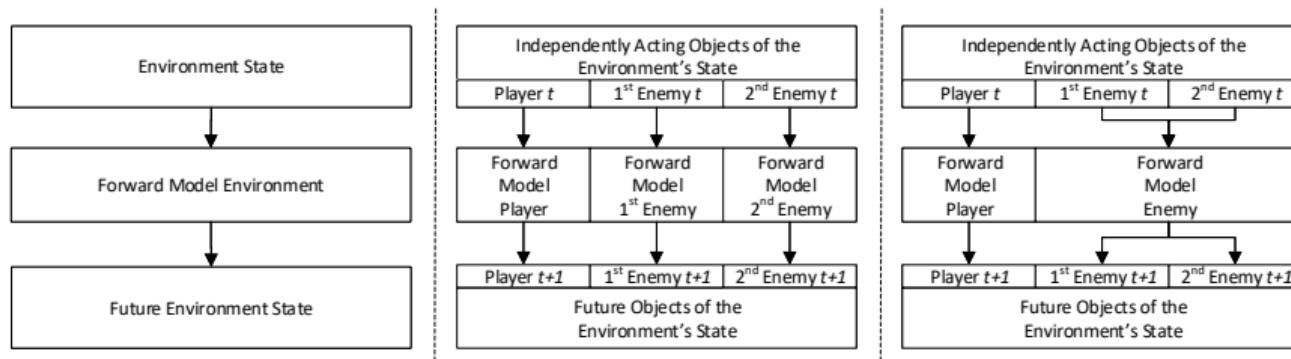
$$\begin{aligned}
 fm(S_t, A_t) &= (fm_{1,1}(S_t^{(1,1)}, \dots, S_t^{(1,k)}), \dots, fm_{m,k}(S_t^{(m,1)}, \dots, S_t^{(m,k')}), A_t) \\
 &= ((fm_{1,1}((S_t^{(1,1)}, \dots, S_t^{(1,k)}), A_t), \dots, fm_{m,k}((S_t^{(m,1)}, \dots, S_t^{(m,k')}), A_t))) \\
 &= (S_{t+1}^{(1)}, S_{t+1}^{(2)}, \dots, S_{t+1}^{(n)}) = S_{t+1}
 \end{aligned}$$



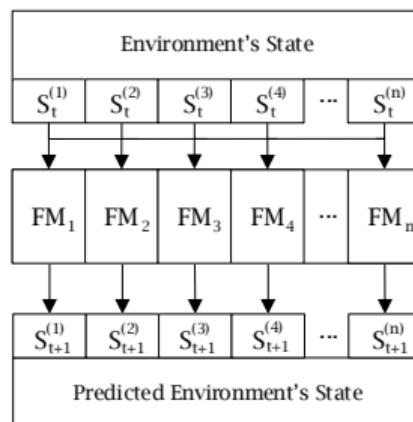
# Object-based Forward Model

Aggregiere die Vorhersage für jedes Objekt und die zugehörigen Sensorwerte:

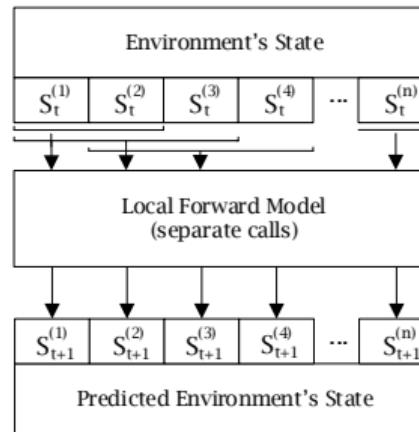
$$\begin{aligned}
 fm(S_t, A_t) &= (fm_1(S_t, A_t), \dots, fm_n(S_t, A_t)) \\
 &= ((fm_{1,1}((S_t^{(1,1)}, \dots, S_t^{(1,k)}), A_t), \dots, fm_{m,k}((S_t^{(m,1)}, \dots, S_t^{(m,k')}), A_t))) \\
 &= (S_{t+1}^{(1)}, S_{t+1}^{(2)}, \dots, S_{t+1}^{(n)}) = S_{t+1}
 \end{aligned}$$



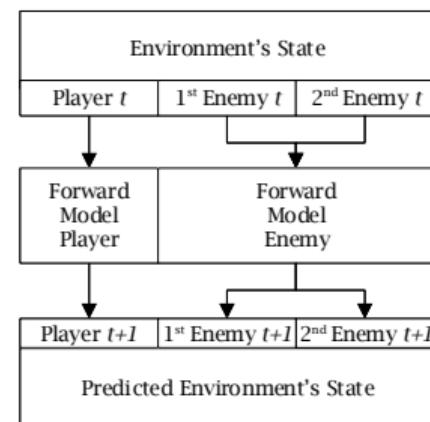
# Vergleich Entwickelter Modelle



Decomposed FM



Local FM



Object-based FM

# Evaluierung: Lernen Unbekannter Spielmodelle

# Evaluierung: Lernen Unbekannter Spielmodelle

**Ziel:** Lerne ein unbekanntes Spiel zu spielen

- lerne ein Modell anhand vorheriger Interaktionen
- nutze Prädiktionsbasierte Suche

# Evaluierung: Lernen Unbekannter Spielmodelle

**Ziel:** Lerne ein unbekanntes Spiel zu spielen

- lerne ein Modell anhand vorheriger Interaktionen
- nutze Prädiktionsbasierte Suche

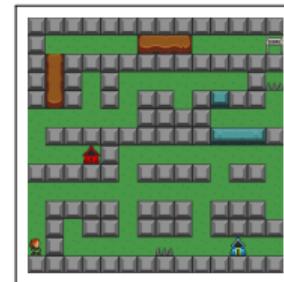
Auswertung basierend auf 30 Spielen des General Video Game AI (GVGAI) Frameworks



Escape



Vortex



Labyrinth



Bait

# Evaluierung: Lernen Unbekannter Spielmodelle

**Ziel:** Lerne ein unbekanntes Spiel zu spielen

- lerne ein Modell anhand vorheriger Interaktionen
- nutze Prädiktionsbasierte Suche

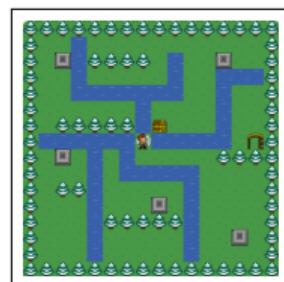
Auswertung basierend auf 30 Spielen des General Video Game AI (GVGAI) Frameworks

Unterschiedliche Spieleigenschaften, z.B.:

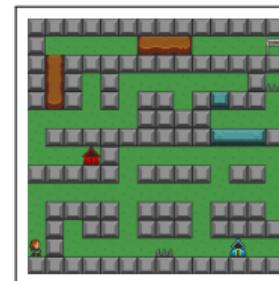
- Belohnungsstil (dicht/spärlich)
- Art der Siegbedingung
- Determinismus vs. Nicht-Determinismus



Escape



Vortex



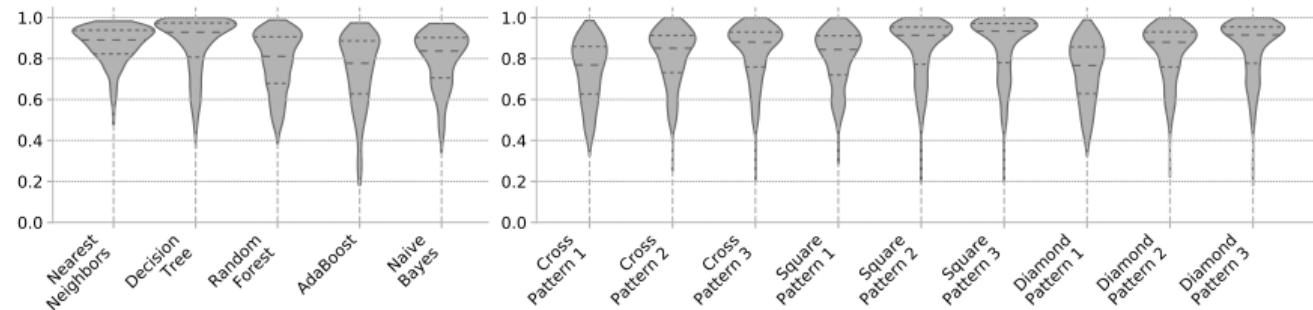
Labyrinth



Bait

# Vorhersagegenauigkeit Entwickelter Modelle

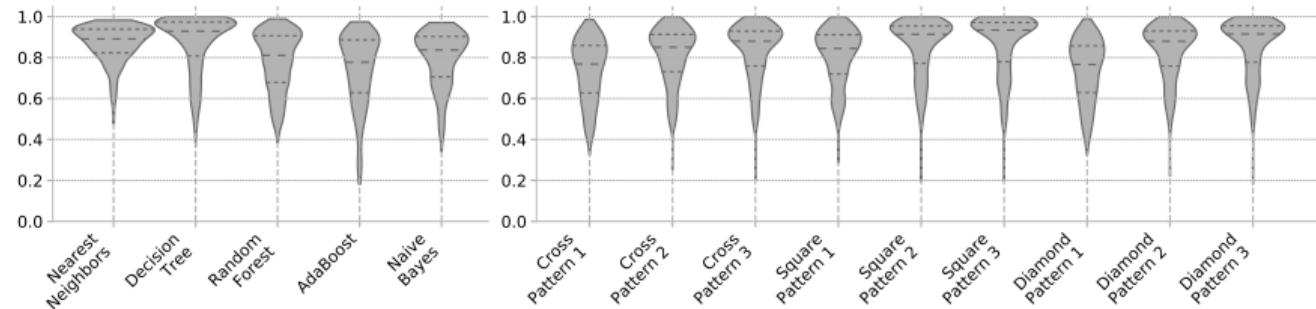
## Modellbewertung und Parameteroptimierung



[1] Dockhorn, A., Doell C., Hewelt, M & Kruse, R.(2017) A Decision Heuristic for Monte Carlo Tree Search Doppelkopf Agents. In: 2017 IEEE Symposium Series on Computational Intelligence

# Vorhersagegenauigkeit Entwickelter Modelle

## Modellbewertung und Parameteroptimierung



⇒ weitere Performanzverbesserungen durch den Einsatz von Ensemble Verfahren

Trade-off zwischen Vorhersagegenauigkeit und Simulationsanzahl

- komplexe Vorhersagemodelle reduzieren die Anzahl möglicher Simulationen
- Leistung hängt von Simulationsgüte und Anzahl ab<sup>[1]</sup>

[1] Dockhorn, A., Doell C., Hewelt, M & Kruse, R.(2017) A Decision Heuristic for Monte Carlo Tree Search Doppelkopf Agents. In: 2017 IEEE Symposium Series on Computational Intelligence

# Ergebnisse General Game Learning

State-of-the-Art

Forward Model Learning

nach 5 Minuten Trainingszeit:

- können zahlreiche Spiele gewonnen werden
- gleichen die gelernten Modelle bis auf wenige Ausnahmen dem Originalmodell

---

[1] Dockhorn, A., Saxton, C., & Kruse, R. (2020). Association Rule Mining for Unknown Video Games. In: M.-J. Lesot & C. Marsala (Eds.), Fuzzy Approaches for Soft Computing and Approximate Reasoning: Theories and Applications

# Ergebnisse General Game Learning

State-of-the-Art

Forward Model Learning

nach 5 Minuten Trainingszeit:

- können zahlreiche Spiele gewonnen werden
- gleichen die gelernten Modelle bis auf wenige Ausnahmen dem Originalmodell

Lernen von Siegbedingungen durch Assoziationsregeln und Hypothesengenerierung<sup>[1]</sup>

---

[1] Dockhorn, A., Saxton, C., & Kruse, R. (2020). Association Rule Mining for Unknown Video Games. In: M.-J. Lesot & C. Marsala (Eds.), Fuzzy Approaches for Soft Computing and Approximate Reasoning: Theories and Applications

## General Strategy Game Learning

Weitere Herausforderungen entstehen durch die gesteigerte Komplexität der Umgebung:

- Die Leistung der Suchalgorithmen leidet unter der steigenden Breite und Tiefe des Suchbaums.

# General Strategy Game Learning

Weitere Herausforderungen entstehen durch die gesteigerte Komplexität der Umgebung:

- Die Leistung der Suchalgorithmen leidet unter den steigenden Breite und Tiefe des Suchbaums.

Stratega - General Strategy Game-Playing<sup>[1,2]</sup>



[1] Dockhorn, A., Hurtado-Grueso, J., Jeurissen, D., & Perez-Liebana, D. (2020). Stratega - A General Strategy Games Framework. In: AIIDE-20 Workshop on Artificial Intelligence for Strategy Games

[2] Perez-Liebana, D., Dockhorn, A., Hurtado-Grueso, J., Jeurissen, D., & (2020). The Design Of "Stratega": A General Strategy Games Framework

# General Strategy Game Learning

Weitere Herausforderungen entstehen durch die gesteigerte Komplexität der Umgebung:

- Die Leistung der Suchalgorithmen leidet unter der steigenden Breite und Tiefe des Suchbaums.

## Stratega - General Strategy Game-Playing<sup>[1,2]</sup>

- Rundenbasierte und Echtzeitstrategiespiele
- Kontrollieren von mehreren Einheiten mit jeweils mehreren Aktionen pro Zug



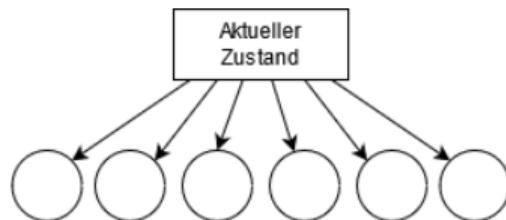
[1] Dockhorn, A., Hurtado-Grueso, J., Jeurissen, D., & Perez-Liebana, D. (2020). Stratega - A General Strategy Games Framework. In: AIIDE-20 Workshop on Artificial Intelligence for Strategy Games

[2] Perez-Liebana, D., Dockhorn, A., Hurtado-Grueso, J., Jeurissen, D., & (2020). The Design Of "Stratega": A General Strategy Games Framework

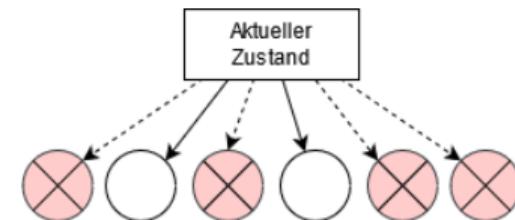
# Action Abstractions

Methoden zum Abstrahieren der Aktionsmenge:

- Trainieren einer Policy<sup>[1]</sup>



Action Abstraction  
Action Filtering  
Dynamic Policies



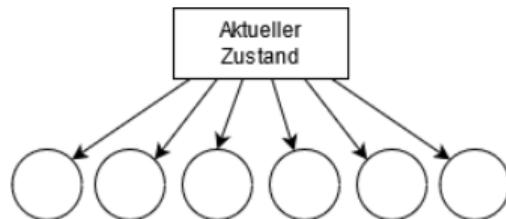
[1] Dockhorn, A., Doell, C., Hewelt, M., & Kruse, R. (2017). A Decision Heuristic for Monte Carlo Tree Search Doppelkopf Agents. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI).

[2] Dockhorn, A., Hurtado-Grueso, J., Jeurissen, D., Xu, L., & Perez-Liebana, D. (2021). Portfolio Search and Optimization for General Strategy Game-Playing. (submitted)

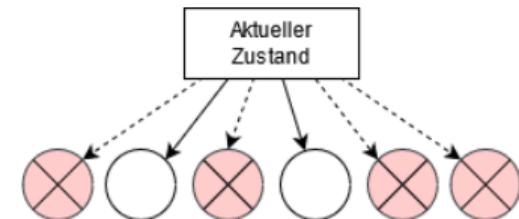
# Action Abstractions

Methoden zum Abstrahieren der Aktionsmenge:

- Trainieren einer Policy<sup>[1]</sup>
- Optimierung von Portfolio-basierten Suchverfahren<sup>[2]</sup>



Action Abstraction  
Action Filtering  
Dynamic Policies



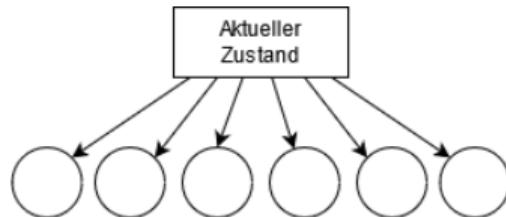
[1] Dockhorn, A., Doell, C., Hewelt, M., & Kruse, R. (2017). A Decision Heuristic for Monte Carlo Tree Search Doppelkopf Agents. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI).

[2] Dockhorn, A., Hurtado-Grueso, J., Jeurissen, D., Xu, L., & Perez-Liebana, D. (2021). Portfolio Search and Optimization for General Strategy Game-Playing. (submitted)

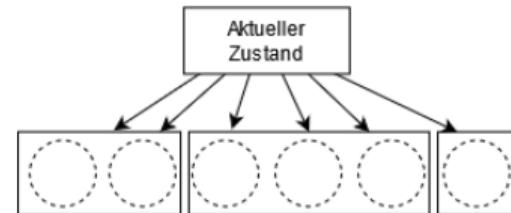
# Game State Abstractions

Methoden zum Abstrahieren der Zustandsmenge:

- (Fuzzy) Cluster-Analyse zum Erstellen von Meta-Zuständen<sup>[1]</sup>



Game State Abstraction  
Game State Clustering  
Search on Meta-States



[1] Dockhorn, A. & Kruse, R. (2020). Predicting Cards Using a Fuzzy Multiset Clustering of Decks. In: International Journal of Computational Intelligence Systems (IJCIS).

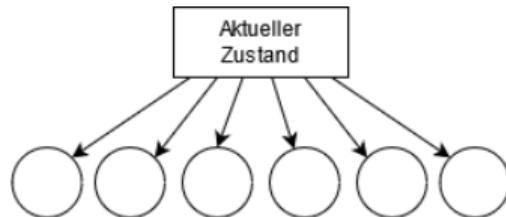
[2] Dockhorn, A. & Perez-Liebana, D. (2021). Learning State Abstractions for Complex Strategy Games. (working title)

[3] Xu, L., Dockhorn, A. & Perez-Liebana, D. (2021). Abstractions in Strategy Games. (working title)

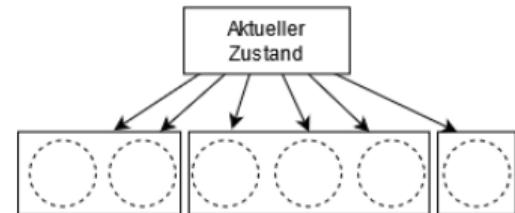
# Game State Abstractions

Methoden zum Abstrahieren der Zustandsmenge:

- (Fuzzy) Cluster-Analyse zum Erstellen von Meta-Zuständen<sup>[1]</sup>
- Abstraktion als Optimierungsproblem<sup>[2]</sup>



Game State Abstraction  
Game State Clustering  
Search on Meta-States



[1] Dockhorn, A. & Kruse, R. (2020). Predicting Cards Using a Fuzzy Multiset Clustering of Decks. In: International Journal of Computational Intelligence Systems (IJCIS).

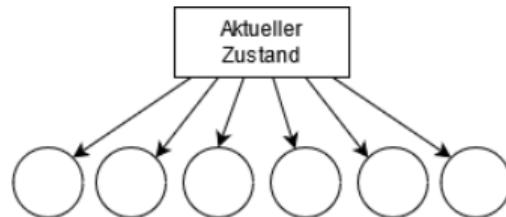
[2] Dockhorn, A. & Perez-Liebana, D. (2021). Learning State Abstractions for Complex Strategy Games. (working title)

[3] Xu, L., Dockhorn, A. & Perez-Liebana, D. (2021). Abstractions in Strategy Games. (working title)

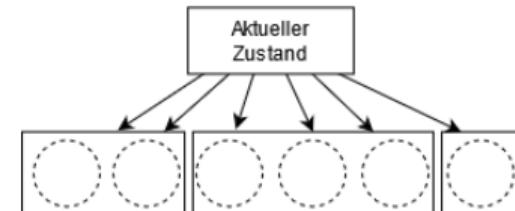
# Game State Abstractions

Methoden zum Abstrahieren der Zustandsmenge:

- (Fuzzy) Cluster-Analyse zum Erstellen von Meta-Zuständen<sup>[1]</sup>
- Abstraktion als Optimierungsproblem<sup>[2]</sup>
- Entwicklung von Suchverfahren in abstrahierten Zustandsräumen<sup>[3]</sup>



Game State Abstraction  
Game State Clustering  
Search on Meta-States



[1] Dockhorn, A. & Kruse, R. (2020). Predicting Cards Using a Fuzzy Multiset Clustering of Decks. In: International Journal of Computational Intelligence Systems (IJCIS).

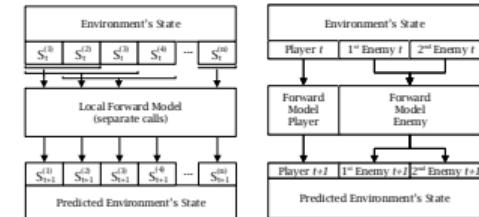
[2] Dockhorn, A. & Perez-Liebana, D. (2021). Learning State Abstractions for Complex Strategy Games. (working title)

[3] Xu, L., Dockhorn, A. & Perez-Liebana, D. (2021). Abstractions in Strategy Games. (working title)

# Zusammenfassung

Methoden zum effizienten Lernen von Umgebungsmodellen

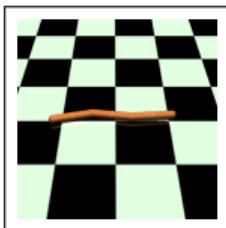
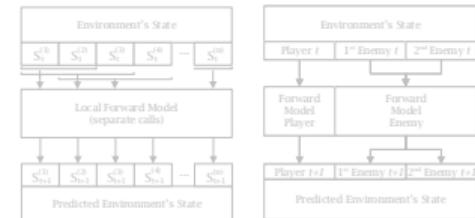
- Unabhängigkeitsannahmen reduzieren den Modellraum und die benötigte Trainingszeit
- Ermöglicht eine Prädiktionsbasierte Suche



# Zusammenfassung

Methoden zum effizienten Lernen von Umgebungsmodellen

- Unabhängigkeitsannahmen reduzieren den Modellraum und die benötigte Trainingszeit
- Ermöglicht eine Prädiktionsbasierte Suche



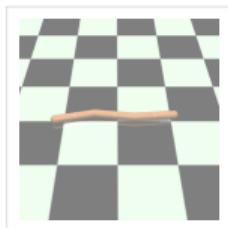
Anwendungen in General Game Learning und Motion Control

- Vorhersage ermöglicht Interpretation des Ergebnisses
- Modelle sind übertragbar auf Problemvarianten

# Zusammenfassung

Methoden zum effizienten Lernen von Umgebungsmodellen

- Unabhängigkeitsannahmen reduzieren den Modellraum und die benötigte Trainingszeit
- Ermöglicht eine Prädiktionsbasierte Suche

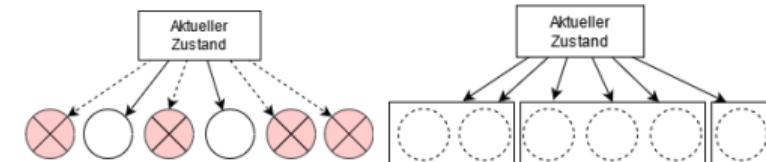
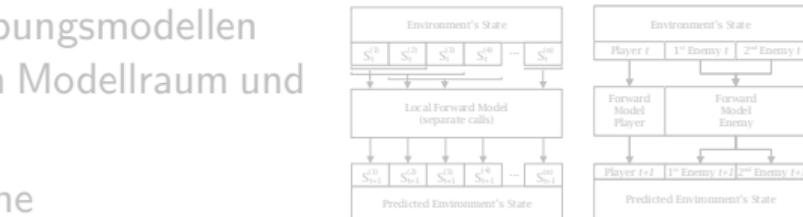


Anwendungen in General Game Learning und Motion Control

- Vorhersage ermöglicht Interpretation des Ergebnisses
- Modelle sind Übertragung auf Problemvarianten

Erweiterungen für komplexere Umgebungen

- Abstrahieren des Aktionsraums
- Abstrahieren des Zustandsraums



# Alexander Dockhorn

a.dockhorn@qmul.ac.uk

Queen Mary University of London

School of Electronic Engineering and Computer Science  
Game AI Research Group