Relational Reinforcement Learning

Stefano Albrecht, Mark Sollweck

Seminar aus maschinellem Lernen

TECHNISCHE UNIVERSITÄT DARMSTADT

Vortragsstruktur

TECHNISCHE UNIVERSITÄT DARMSTADT

Teil 1:

- *Relational Reinforcement Learning*
  S. Džeroski, L. De Readt, K. Driessens

Teil 2:

- *Reinforcement Learning in Relational Domains: A Policy-Language Approach*
  A. Fern, S. Yoon, R. Givan

Vortragsstruktur von Teil 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reinforcement Learning
- Relational Reinforcement Learning

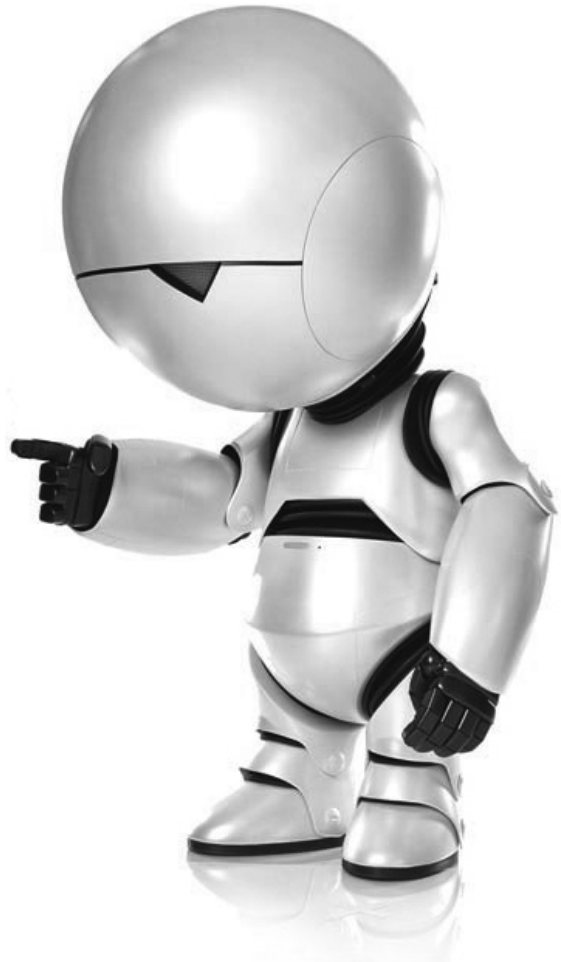
Reinforcement Learning
Inhalt

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reinforcement Learning
  - → Beispiel: Roboter
  - → Beispiel: Hund
  - → Definition
  - → Q-Learning
  - → P-Learning
  - → Nachteile der Algorithmen

Reinforcement Learning
Beispiel: Roboter

TECHNISCHE UNIVERSITÄT DARMSTADT



Reinforcement Learning
Beispiel: Hund

TECHNISCHE UNIVERSITÄT DARMSTADT

Reinforcement Learning
Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

Gegeben:

- Zustandsmenge S
- Aktionsmenge A
- Übergangsfunktion $\delta : S \times A \rightarrow S$ (unbekannt)
- Reward-Funktion $r : S \times A \rightarrow \mathbb{R}$ (unbekannt)

Gesucht ist eine Policy $\pi : S \rightarrow A$, die den kumulativen Reward

$$V^{\pi}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

mit $0 \leq \gamma < 1$ für alle $s_t \in S$ maximiert (fortan bezeichnet als π^*).

Reinforcement Learning
Q-Learning (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Die optimale Policy π^* lässt sich schreiben als:

- $\pi^*(s) = argmax_a(r(s,a) + \gamma V^{\pi^*}(\delta(s,a)))$

Problem: r und δ sind unbekannt

Reinforcement Learning
Q-Learning (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Die Q-Funktion für Policy π ist definiert als:

- $Q^{\pi}(s, a) = r(s, a) + \gamma V^{\pi}(\delta(s, a))$

Damit lässt sich π^* schreiben als:

- $\pi^*(s) = argmax_a Q^*(s, a)$
- Q^* ist die Q-Funktion der optimalen Policy π^*

Lösung: Approximiere Q^* (möglich ohne Kenntnis von r und δ)

Reinforcement Learning
Q-Learning (3)

TECHNISCHE UNIVERSITÄT DARMSTADT

for each s, a **do**
  initialize table entry $\hat{Q}(s, a) = 0$
do forever
  i := 0
  generate random state s_0
  while not goal(s_i) **do**
    select action a_i and execute it
    receive immediate reward $r_i = r(s_i, a_i)$
    observe new state s_{i+1}
    i := i + 1
  endwhile
  for j = i - 1 to 0 **do**
    update $\hat{Q}(s_j, a_j) := r_j + \gamma max_{a'} \hat{Q}(s_{j+1}, a')$

Reinforcement Learning
P-Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

Die P-Funktion für Policy π ist definiert als:

- $P^{\pi}(s, a) = 1$, falls $a = argmax_{a'} Q^{\pi}(s, a')$, sonst 0

Aus der Approx. $\hat{Q}$ von Q^* lässt sich eine Approx. $\hat{P}$ von P^* ableiten.

Vorteile der P-Funktion gegenüber Q-Funktion:

- Geringere Komplexität
- Generalisiert oftmals besser

Reinforcement Learning
Nachteile der Algorithmen

TECHNISCHE UNIVERSITÄT DARMSTADT

Drei Nachteile der Q/P-Algorithmen:

- Undurchführbar für große Zustandsräume
- Gelernte Ziele generalisieren nicht auf ähnliche Ziele
- Manipulation der Lernumgebung erfordert neues Lernen der Ziele

Insbesondere die letzten beiden Probleme (*Generalisierungs-Problem*) werden durch Hinzufügen von relationalen Lernmechanismen behandelt.

$\Rightarrow$ Relational Reinforcement Learning

Relational Reinforcement Learning
Inhalt

TECHNISCHE UNIVERSITÄT DARMSTADT

- Relational Reinforcement Learning
  - → Definition
  - → Q-Learning
  - → P-Learning
  - → Beispiel: 3-Blocks-World

Relational Reinforcement Learning
Definition

TECHNISCHE UNIVERSITÄT DARMSTADT

Gegeben:

- Zustandsmenge S (Zustand ist Menge von Fakten)
- Aktionsmenge A (Menge von Fakten)
- Übergangsfunktion $\delta : S \times A \rightarrow S$
- Reward-Funktion $r : S \times A \rightarrow \mathbb{R}$
- Hintergrundwissen (Menge von Prädikaten)

Gesucht ist eine Policy $\pi : S \rightarrow A$, die den kumulativen Reward maximiert.

$\Rightarrow$ Repräsentiert als Q/P-Funktion (Q/P-Entscheidungsbaum)

Relational Reinforcement Learning
Q-Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

Initialize $\hat{Q}_0$ to assign 0 to all (s, a) pairs, set e := 0
do forever
  e := e + 1, i := 0
  generate random state s_0
  while not goal(s_i) **do**
    select action a_i and execute it
    receive immediate reward $r_i = r(s_i, a_i)$
    observe new state s_{i+1}
    i := i + 1
  endwhile
  for j = i - 1 to 0 **do**
    generate example $x = (s_j, a_j, \hat{q}_j := r_j + \gamma max_{a'} \hat{Q}_e(s_{j+1}, a'))$
    replace example $(s_j, a_j, \hat{q}_{old})$, if existing
update $\hat{Q}_e$ using a tree generator to produce $\hat{Q}_{e+1}$ from the examples

Relational Reinforcement Learning
P-Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

P-Funktion wird hier nicht direkt von der Q-Funktion abgeleitet:

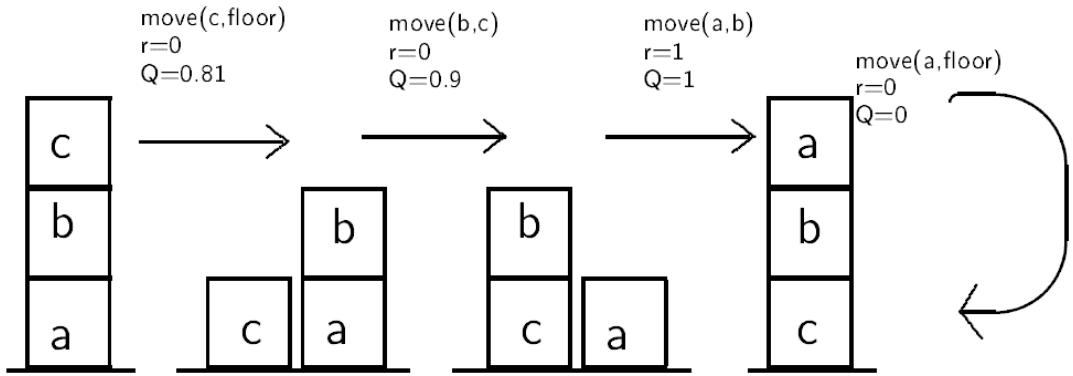
$\Rightarrow$ Darstellung zu komplex

Stattdessen wird sie mit folgendem zusätzlichen Code separat gelernt:

for j = i - 1 to 0 **do**
  for all actions a_k possible in state s_j **do**
    if state action pair (s_j, a_k) is optimal according to $\hat{Q}_{e+1}$
      then generate example (s_j, a_k, c) where $c = 1$
      else generate example (s_j, a_k, c) where $c = 0$
update $\hat{P}_e$ using a tree generator to produce $\hat{P}_{e+1}$ from examples (s_j, a_k, c)

Relational Reinforcement Learning
Beispiel: 3-Blocks-World (1)

TECHNISCHE UNIVERSITÄT DARMSTADT



Example 1	Example 2	Example 3	Example 4
qvalue(0.81).	qvalue(0.9).	qvalue(1.0).	qvalue(0.0).
move(c,floor).	move(b,c).	move(a,b).	move(a,floor).
goal(on(a,b)).	goal(on(a,b)).	goal(on(a,b)).	goal(on(a,b)).
clear(c).	clear(b).	clear(a).	clear(a).
on(c,b).	clear(c).	clear(b).	on(a,b).
on(b,a).	on(b,a).	on(b,c).	on(b,c).
on(a,floor).	on(a,floor).	on(a,floor).	on(c,floor).
	on(c,floor).	on(c,floor).	

Relational Reinforcement Learning
Beispiel: 3-Blocks-World (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Logischer Entscheidungsbaum in Prolog:

```
qvalue(0) :- goal_on(A,B), action_move(C,D), on(A,B), !.
qvalue(1) :- goal_on(A,B), action_move(C,D), clear(A), !.
qvalue(0.9) :- goal_on(A,B), action_move(C,D), clear(D), !.
qvalue(0.81).
```

Example 1	Example 2	Example 3	Example 4
qvalue(0.81).	qvalue(0.9).	qvalue(1.0).	qvalue(0.0).
move(c,floor).	move(b,c).	move(a,b).	move(a,floor).
goal(on(a,b)).	goal(on(a,b)).	goal(on(a,b)).	goal(on(a,b)).
clear(c).	clear(b).	clear(a).	clear(a).
on(c,b).	clear(c).	clear(b).	on(a,b).
on(b,a).	on(b,a).	on(b,c).	on(b,c).
on(a,floor).	on(a,floor).	on(a,floor).	on(c,floor).
	on(c,floor).	on(c,floor).	

Vortragsstruktur von Teil 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Markov Decision Process
- Relational Markov Decision Process

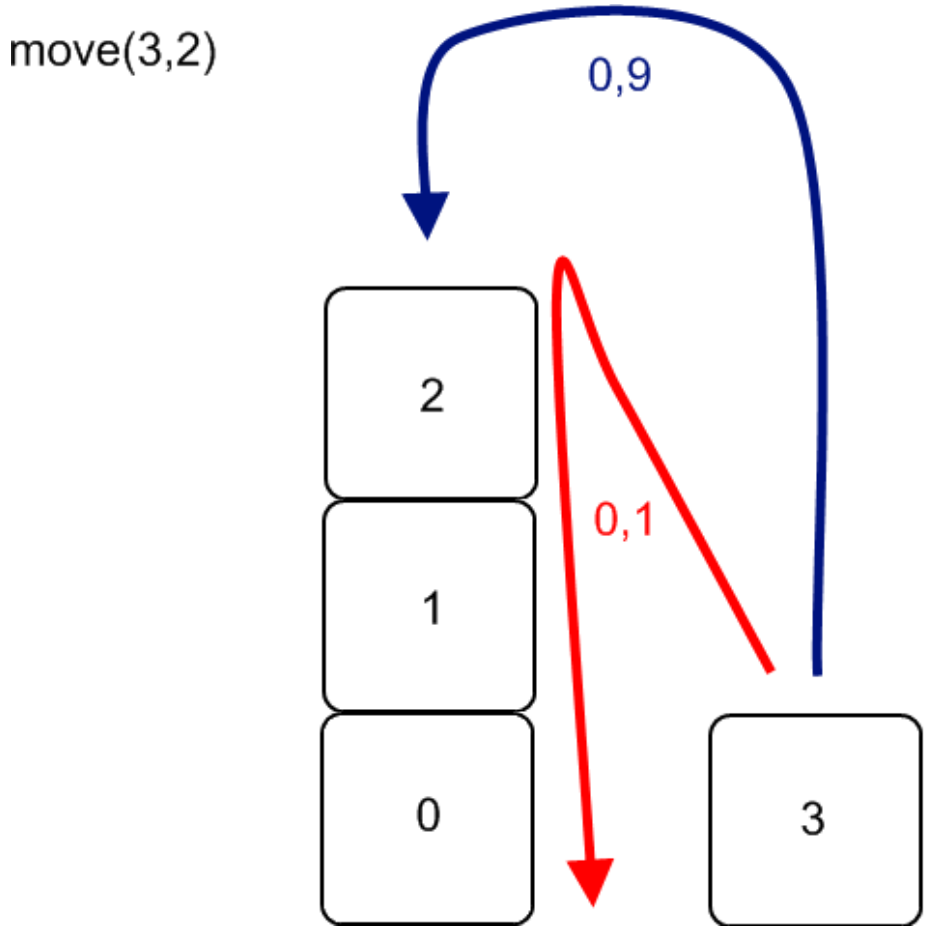
Markov Decision Process
Inhalt

TECHNISCHE UNIVERSITÄT DARMSTADT

- Markov Decision Process
  - → Beispiel
  - → Definition
  - → Policy Iteration

Markov Decision Process
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT



Markov Decision Process
Definition (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Gegeben:

- Zustandsmenge S
- Aktionsmenge A
- Übergangsfunktion $T : S \times A \rightarrow S$ *(probabilistisch)*
- Reward-Funktion $R : S \times A \rightarrow \mathbb{R}$
- Initialzustands-Funktion I *(probabilistisch)*

Annahme: T und R sind bekannt

Markov Decision Process
Definition (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wert-Funktion:

  $\Rightarrow V^{\pi}(s) = E[R(s, \pi(s)) + \gamma V^{\pi}(T(s, \pi(s)))]$

- Q-Funktion:

  $\Rightarrow Q(s, a)^{\pi} = R(s, a) + \gamma E[V^{\pi}(T(s, a))]$

- Gesucht ist eine optimale Policy $\pi : S \rightarrow A$

  $\Rightarrow \pi^{*} = argmax_{\pi} \overline{V}(\pi)$ mit $\overline{V}(\pi) = E[V^{\pi}(I)]$

Markov Decision Process
Policy Iteration (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Ausgehend von Anfangspolicy π_0 werden folgende Schritte iterativ ausgeführt:

- 1.) Policy-Evaluation: Berechne für jeden Zustand $s \in S$

  $\Rightarrow V^{\pi}(s) = E[R(s, \pi(s)) + \gamma V^{\pi}(T(s, \pi(s)))]$

- 2.) Policy-Verbesserung:

  $\Rightarrow \hat{\pi}(s) = argmax_{a \in A}(R(s, a) + \gamma E[V^{\pi}(T(s, a))])$

Markov Decision Process
Policy Iteration (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Nachteile:

- Wert-Funktionen sind schwer zu lernen
- Exakte Policy-Iteration ist nicht handhabbar für große Zustandsmengen, welche besonders bei relationalem Planen vorkommen
- Keine Generalisierbarkeit

Relational Markov Decision Process
Inhalt

TECHNISCHE UNIVERSITÄT DARMSTADT

- Relational Markov Decision Process
  - → Definition
  - → Approximate Policy Iteration
  - → Decision List Policies
  - → Bootstrapping
  - → Zusammenfassung
  - → Evaluierung

Relational Markov Decision Process Definition (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Ein relationaler MDP ist gegeben durch:

- Menge von Zuständen S *(relational)*
- Menge von Aktionen A *(relational)*
- Übergangsfunktion $T : S \times A \rightarrow S$ *(probabilistisch)*
- Reward-Funktion $R : S \times A \rightarrow \mathbb{R}$
- Problemgenerator I *(probabilistisch)*

$\Rightarrow$ Gesucht ist eine optimale Policy π^*

S und A sind definiert durch eine endliche Menge an Objekten (O), Prädikaten (P) und Aktionstypen (Y).

Relational Markov Decision Process
Definition (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Prädikatenmenge P ist gegeben durch $P = W \cup G$

- W ist Menge an Weltprädikaten
- G die dazugehörige Menge an Zielprädikaten

Beispiel: $W = \{on, clear, on - table\} \rightarrow G = \{gon, gclear, gon - table\}$

Ein MDP-Zustand ist eine Menge von wahren Weltfakten und Zielbedingungen.

Beispiel: $S = \{on - table(a), on(a, b), clear(b), gclear(b)\}$

Relational Markov Decision Process
Approximate Policy Iteration (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Schritt: Verbesserte Trajektorien generieren
   - Gegeben ist Anfangspolicy π
   - Trajektorie der verbesserten Policy $\hat{\pi}$ generieren:
     $I \overset{rnd}{\rightarrow} S_0 \overset{\hat{\pi}(S_0)}{\rightarrow} S_1 \overset{\hat{\pi}(S_1)}{\rightarrow} \ldots \overset{\hat{\pi}(S_{h-1})}{\rightarrow} S_h$
     wobei $\hat{\pi}$ eine Approximation der verbesserten Policy $\pi' = PI^{\pi}$ ist

2. Schritt: Policy Lernen
   - Es soll eine neue Policy gelernt werden, die den Trajektorien der verbesserten Policy entspricht

Vorteil:

$\Rightarrow$ Policy wird direkt gelernt, ohne die Wertfunktion V^{π} zu lernen

Relational Markov Decision Process
Approximate Policy Iteration (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

API $(n, w, h, M, \pi_0, \gamma)$
$\pi \leftarrow \pi_0$;
loop
  $T \leftarrow$ **Improved-Trajectories**(n, w, h, M, π);
  $\pi \leftarrow$ **Learn-Policy**(T);
until satisfied with π;
Return π;

Relational Markov Decision Process
Approximate Policy Iteration (3)

TECHNISCHE UNIVERSITÄT DARMSTADT

Improved-Trajectories(n, w, h, M, π)
// training set size n, sampling width w
// horizon h, MDP M, current policy π
$T \leftarrow \emptyset$;
repeat n times
  $t \leftarrow$ **nil**;
  $s \leftarrow$ state drawn from I;
  for $i = 1$ to h
    $\left\langle \hat{Q}(s, a_1), ..., \hat{Q}(s, a_m) \right\rangle \leftarrow$ **Policy-Rollout**(π, s, w, h, H);
    $t \leftarrow t * \left\langle s, \pi(s), \hat{Q}(s, a_1), ..., \hat{Q}(s, a_m) \right\rangle$;
    $a \leftarrow$ action maximizing $\hat{Q}(s, a)$;
    $s \leftarrow$ state sampled from $T(s, a)$;
  $T \leftarrow T \cup t$;
Return T;

Relational Markov Decision Process
Approximate Policy Iteration (4)

TECHNISCHE UNIVERSITÄT DARMSTADT

Policy-Rollout(s, w, h, M, π)
// policy π, state s, sampling width w
// horizon h, cost estimator H
for each action $a_i \in A$
  $\hat{Q}(s, a_i) \leftarrow 0$;
  repeat w times
    $R \leftarrow R(s, a_i)$;
    $s' \leftarrow$ a state sampled from $T(s, a_i)$;
    for $i = 1$ to $h - 1$
      $R \leftarrow R + \gamma^i R(s', \pi(s'))$;
      $s' \leftarrow$ a state sampled from $T(s', \pi(s'))$;
    $\hat{Q}(s, a_i) \leftarrow \hat{Q}(s, a_i) + R$;
  $\hat{Q}(s, a_i) \leftarrow \frac{\hat{Q}(s, a_i)}{w}$;
Return $\left\langle \hat{Q}(s, a_1), ..., \hat{Q}(s, a_m) \right\rangle$;

Relational Markov Decision Process
Decision List Policies (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Repräsentation der Policy π mit Hilfe von Decision-Lists:

Sei $X = (x_1, x_2, ..., x_k)$ eine Liste von Variablen, dann ist die Syntax eines Klassenausdrucks gegeben durch:

$C[X] ::= C_0 \mid x_i \mid \textbf{a-thing} \mid \neg C[X] \mid (R\ C[X]) \mid (min\ R)$

$R ::= R_0 \mid R^{-1} \mid R^*$

Wobei:

- C_0: Primitive Klasse (1-stelliges Prädikat)
- R_0: Primitive Relation (2-stelliges Prädikat)
- x_i: Variable aus X
- **a-thing**: Alle Objekte im aktuellen Zustand
- $d(C[X])$: Tiefe von $C[X]$ sei 1 für Primitive, Variablen, **a-thing** und $(min\ R)$, sonst $d(C[X]) + 1$

Relational Markov Decision Process
Decision List Policies (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Die Semantik eines Klassenausdrucks ist gegeben durch einen MDP-Zustand s und eine Variablenzuweisung $O = (o_1, o_2, ..., o_n)$.

$(C_0)^{s,0}$: Menge an Objekten, für die das Prädikat C_0 in s wahr ist
(z.B. *on-table* ergibt alle Objekte, die sich im Zustand s auf dem Tisch befinden)

$(R_0)^{s,0}$: Menge an Tupeln von Objekten, für welche die Relation R_0 in s wahr ist.

- $(\neg C[X])^{s,O} = \{o \mid o \notin C[X]^{s,O}\}$
- $(R\ C[X])^{s,O} = \{o \mid \exists o' \in C[X]^{s,O}\ s.t.\ (o', o) \in R^{s,O}\}$
- $(min\ R)^{s,O} = \{o \mid \exists o'\ s.t.\ (o, o') \in R^{s,O}, \nexists o'\ s.t.\ (o', o) \in R^{s,O}\}$
- $(R^*)^{s,O} = \mathbf{ID} \cup \{(o_1, o_v) \mid \exists o_2, ..., o_{v-1}\ s.t.\ (o_i, o_{i+1}) \in R^{s,O}\ for\ 1 \leq i < v\}$
- $(R^{-1})^{s,O} = \{(o, o') \mid (o', o) \in R^{s,O}\}$

Relational Markov Decision Process
Decision List Policies (3)

TECHNISCHE UNIVERSITÄT DARMSTADT

Eine Policy π kann nun als eine Liste L von Regeln der Form

$$R = a(x_1, ..., x_k) : L_1, L_2, ..., L_m$$

dargestellt werden, mit

- a ist ein k-stelliger Aktionstyp
- L_i sind Literale der Form $x_i \in C[X]$
- x_i sind Variablen

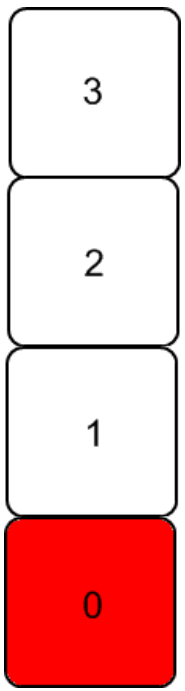
Gegeben ein Zustand s und eine Menge an Objekten $O = (o_1, ..., o_n)$ ist ein Literal $x_i \in C[X]$ wahr $\leftrightarrow o_i \in C[X]^{s,O}$. Eine Regel R erlaubt eine Aktion a, wenn jedes Literal wahr ist.

Relational Markov Decision Process
Decision List Policies (4)

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel: Gesucht sei eine Policy $\pi[L]$, die alle roten Blöcke freilegt.

L: **putdown**(x_i) : $x_1 \in$ **holding**
pickup(x_1) : $x_1 \in$ **clear**, $x_1 \in$ (**on***(**on red**))
mit (**on***(**on red**)) = $\{(2,1),(3,2),(3,1)\}$



Relational Markov Decision Process
Decision List Policies (5)

TECHNISCHE UNIVERSITÄT DARMSTADT

Learn-Decision-List(D, d, l, b)
// training set D, concept depth d, rule length l, beam width b
$L \leftarrow$ **nil**;
while (D is not empty)
  $R \leftarrow$ **Learn-Rule**(D, d, l, b);
  $D \leftarrow D - \{d \in D \mid R \text{ covers } d\}$;
  $L \leftarrow$ **Extend-List**(L, R);
Return L;

Relational Markov Decision Process
Decision List Policies (6)

TECHNISCHE UNIVERSITÄT DARMSTADT

Learn-Rule(D, d, l, b)
// training set D, concept depth d, rule length l, beam width b
for each action type a
  $R_a \leftarrow$ **Beam-Search**(D, d, l, w, a);
Return $argmax_a$**Hvalue**(R_a, D);

Relational Markov Decision Process
Decision List Policies (7)

TECHNISCHE UNIVERSITÄT DARMSTADT

Beam-Search(D, d, l, w, a)
// training set D, concept depth d, rule length l, beam width b, action type a
$k \leftarrow$ arity of a;
$X \leftarrow (x_1, ..., x_k)$;
$L \leftarrow \{(x \in C) \mid x \in X, C \in C_d[X]\}$;
$B_0 \leftarrow \{a(X) : \textbf{nil}\}$; $i \leftarrow 1$;
loop
  $G = B_{i-1} \cup \{R \in R_{d,l} \mid R = \textbf{Add-Literal}(R', l), R' \in B_{i-1}, l \in L\}$;
  $B_i \leftarrow \textbf{Beam-Select}(G, w, D)$;
  $i \leftarrow i + 1$;
until $B_{i-1} = B_i$;
Return $argmax_{R \in B_i}$**Hvalue**(R, D);

Relational Markov Decision Process
Bootstrapping (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Die Wahl der Anfangspolicy π_0 ist entscheidend, damit bei großen Zustandsräumen mit seltenem Reward (wie beim zielbasierten Planen) Policy-Iteration funktioniert.

Zufällige Policy π_0:

$\Rightarrow$ Geringe Wahrscheinlichkeit einen Zielzustand in vertretbarer Zeit zu erreichen

Lösung: **Random-Walk**

Relational Markov Decision Process
Bootstrapping (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Gegeben sei ein MDP $M = \langle S, A, T, R, I \rangle$:

- Zustand $s = (w, g)$, wobei w Menge an Weltfakten und g Menge an Zielfakten.
- Gegeben eine Menge an Zielprädikaten G, sei $s|_G$ derjenige Zustand (w, g'), in dem g' alle Zielfakten aus g enthält, die Anwendungen der Prädikate aus G sind.

Ein n-Schritt Random-Walk $RW_n(M, G)$ ist gegeben durch:

- 1.) Generiere mit I einen zufälligen Zustand $s_0 = (w_0, g_0)$.
- 2.) Führe n zufällige Aktionen durch. Der resultierende Zustand ist $s_n = (w_n, g_0)$.
- 3.) Sei g die zur Menge w_n gehörigen Zielfakten $\Rightarrow$ **Return** $(w_0, g)|_G$

Relational Markov Decision Process
Bootstrapping (3)

TECHNISCHE UNIVERSITÄT DARMSTADT

Eigenschaften:

- Für kleine n sind Probleme leicht zu lösen
- Sobald gelernte Policy gewisse Erfolgsrate erreicht: Erhöhe n, um schwereres, aber dennoch lösbares Problem zu erzeugen
- Durch festlegen der Zielprädikate G kann die Policy π in eine bestimmte "Richtung" gelenkt werden

Ziel:

- Gute Performance bei langen Random-Walks ($n = N$)
- Dadurch soll die gelernte Policy π bereits genügend Wissen aus der Domäne gesammelt haben, um "echte" Probleme lösen zu können

Relational Markov Decision Process
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

LRW-API(N, G, n, w, h, M, π_0, γ)
// max random-walk length N, goal predicates G
// training set size n, sampling width w, horizon h
// MDP M, initial policy π_0, discount factor γ
$\pi \leftarrow \pi_0$; $n \leftarrow 1$;
loop
  if $\widehat{SR}_\pi(n) > \tau$
    // Find harder n-step distribution for π
    $n \leftarrow$ least $i \in [n, N]$
  $M' = M[RW_n(M, G)]$;
  $T \leftarrow$ **Improved-Trajectories**(n, w, h, M', π);
  $\pi \leftarrow$ **Learn-Policy**(T)
until satisfied with π
Return π;

Relational Markov Decision Process
Evaluierung

TECHNISCHE UNIVERSITÄT DARMSTADT

		π_*		FF	
Domain	Size	SR	AL	SR	AL
Blocks	(20)	1	54	0.81	60
	(50)	1	151	0.28	158
Freecell	(4,2,2,4)	0.36	15	1	10
	(4,13,4,8)	0	—	0.47	112
Logistics	(1,2,2,6)	0.87	6	1	6
	(3,10,2,30)	0	—	1	158
Elevator	(60,30)	1	112	1	98
Schedule	(50)	1	175	1	212
Briefcase	(10)	1	30	1	29
	(50)	1	162	0	—
Gripper	(50)	1	149	1	149

Quellen

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Relational Reinforcement Learning*
  S. Džeroski, L. De Readt, K. Driessens
- *Reinforcement Learning in Relational Domains: A Policy-Language Approach*
  A. Fern, S. Yoon, R. Givan
- *Artificial Intelligence: A Modern Approach*
  S. Russell, P. Norvig
- *Machine Learning*
  T. M. Mitchell