Universität zu Köln

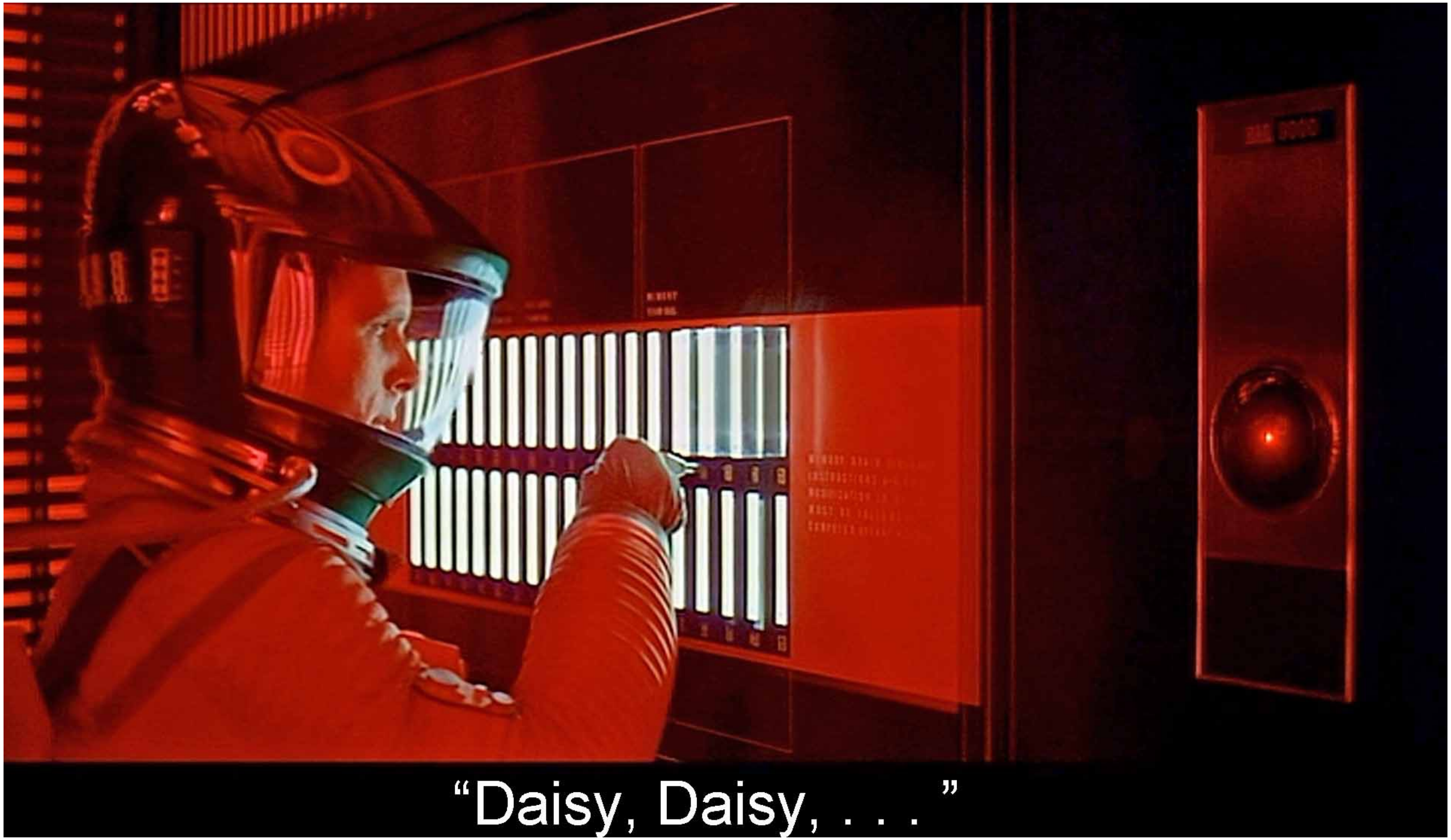
Basisinformationstechnologie II

Sommersemester 2023. 9: Künstliche Intelligenz: Geschichte, Künstliche Neuronale Netze, Monte Carlo Simulationen

Teilweise basierend auf Jan Wieners' Folien

Institut für Digital Humanities, Historisch-Kulturwissenschaftliche Informationsverarbeitung | Prof. Dr. Øyvind Eide | Slavina Stoyanova

1968: der Tod des Rechners



Stanley Kubrick, 2001: A Space Odyssey, 1968

KI: eine lange Geschichte

- Interesse an Wellen
- Angst vor dem Rechner – und Liebe
  - Roboter
  - der militärisch-industrielle Komplex
- Overselling
  - Der Verkauf von Produkten durch aufgeblähte und übertriebene Behauptungen

Kognitive Wissenschaft

- Technologie und Modelle des Gehirns
- Von der Technologie bis zum Gehirn
  - Uhrwerke
  - Telefonzentralen
  - Rechner
  - neurale Netzwerke
- Von dem Hirn zur Technologie
  - was ist Intelligenz?
  - menschliche bzw. maschinelle Intelligenz

Programmiersprache

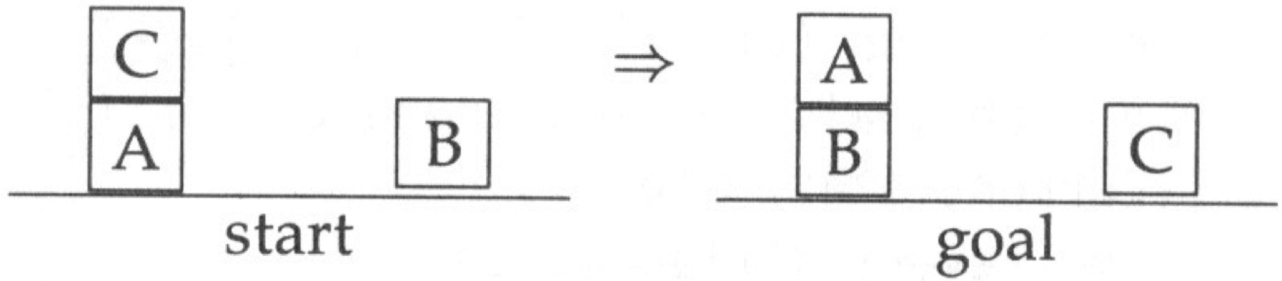
- LISP (1958)
  - KI-Programmierung
  - Fokus auf Text-Verwendungen
  - LISt Processor
    - (Long Infinite String of Parentheses)
- Fortran (1957)
  - Ingenieurwissenschaft
  - Fokus auf Kalkulationen
  - FORmula TRANslation

GPS: General Problem Solver

- Newell und Simon 1957
- Ein Programm für die Lösung aller Probleme
  - sie müssen jedoch erst formuliert werden
- Wichtig für die Entwicklung der Programmierung
  - Separation zwischen den Regeln und der allgemeinen Problemlösungsmaschine
- jedoch nicht besonders intelligent

GPS: Arbeitsschritte

1. Das Problem in vagen Begriffen zu beschreiben
2. Das Problem algorithmisch zu spezifizieren
3. Das Problem in einer Programmiersprache zu implementieren
4. Das Programm auf repräsentative Beispiele auszuprobieren
5. Das Programm zu debuggen und analysieren
    - Wiederholung von dem Prozess




```
> (gps '((c on a) (a on table) (b on table)
         (space on c) (space on b) (space on table))
       '((c on table) (a on b)))
((START)
 (EXECUTING (MOVE C FROM A TO B))
 (EXECUTING (MOVE C FROM B TO TABLE))
 (EXECUTING (MOVE A FROM TABLE TO C))
 (EXECUTING (MOVE A FROM C TO B)))
```

Eliza

- Joseph Weizenbaum, 1966
- MAD-Slip auf einem IBM 7094
- Gespräch mit einer Maschine
- Mustererkennung
- Regelbasierte Transformationen

Eliza

```
(((?* ?x) I want (?* ?y))
 (What would it mean if you got ?y)
 (Why do you want ?y)
 (Suppose you got ?y soon))

> I want to test this program
Why do you want to test this program
```

https://github.com/emacs-mirror/emacs/blob/master/lisp/play/doctor.el

Wohin finden wir die Intelligenz?

```
(defparameter *eliza-rules*
 '(((((?* ?x) hello (?* ?y))
    (How do you do.  Please state your problem.))
   (((?* ?x) I want (?* ?y))
    (What would it mean if you got ?y)
    (Why do you want ?y) (Suppose you got ?y soon))
   (((?* ?x) if (?* ?y))
    (Do you really think its likely that ?y) (Do you wish that ?y)
    (What do you think about ?y) (Really-- if ?y))
   (((?* ?x) no (?* ?y))
    (Why not?) (You are being a bit negative)
    (Are you saying "NO" just to be negative?))
   (((?* ?x) I was (?* ?y))
    (Were you really?) (Perhaps I already knew you were ?y)
    (Why do you tell me you were ?y now?))
   (((?* ?x) I feel (?* ?y))
    (Do you often feel ?y ?))
   (((?* ?x) I felt (?* ?y))
    (What other feelings do you have?))))
```

ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine

JOSEPH WEIZENBAUM
Massachusetts Institute of Technology, Cambridge, Mass.*

ELIZA is a program operating within the MAC time-sharing system at MIT which makes certain kinds of natural language conversation between man and computer possible. Input sentences are analyzed on the basis of decomposition rules which are triggered by key words appearing in the input text. Responses are generated by reassembly rules associated with selected decomposition rules. The fundamental technical problems with which ELIZA is concerned are: (1) the identification of key words, (2) the discovery of minimal context, (3) the choice of appropriate transformations, (4) generation of responses in the absence of key words, and (5) the provision of an editing capability for ELIZA "scripts". A discussion of some psychological issues relevant to the ELIZA approach as well as of future developments concludes the paper.

Introduction

It is said that to explain is to explain away. This maxim is nowhere so well fulfilled as in the area of computer programming, especially in what is called heuristic programming and artificial intelligence. For in those realms machines are made to behave in wondrous ways, often sufficient to dazzle even the most experienced observer. But once a particular program is unmasked, once its inner workings are explained in language sufficiently plain to induce understanding, its magic crumbles away; it stands revealed as a mere collection of procedures, each quite comprehensible. The observer says to himself "I could have written that". With that thought he moves the program in question from the shelf marked "intelligent", to that reserved for curios, fit to be discussed only with people less enlightened than he.

The object of this paper is to cause just such a re-evaluation of the program about to be "explained". Few programs ever needed it more.

ELIZA Program

ELIZA is a program which makes natural language conversation with a computer possible. Its present implementation is on the MAC time-sharing system at MIT. It is written in MAD-SLIP [4] for the IBM 7094. Its name was chosen to emphasize that it may be incrementally improved by its users, since its language abilities may be continually improved by a "teacher". Like the Eliza of Pygmalion fame, it can be made to appear even more civilized, the relation of appearance to reality, however, remaining in the domain of the playwright.

For the present purpose it is sufficient to characterize the MAC system as one which permits an individual to operate a full scale computer from a remotely located typewriter. The individual operator has the illusion that he is the sole user of the computer complex, while in fact others may be "time-sharing" the system with him. What is important here is that the computer can read messages typed on the typewriter and respond by writing on the same instrument. The time between the computer's receipt of a message and the appearance of its response is a function of the program controlling the dialogue and of such MAC system parameters as the number of users currently corresponding with the system. These latter parameters generally contribute so little to the overall response time that conversational interaction with the computer need never involve truly intolerable delays.

When in conversation with ELIZA, the user types in some statement or set of statements in natural language using normal punctuation and sentence structures. Only the question mark may not be used, because it is interpreted as a line delete character by the MAC system. The user's statement is terminated by a double carriage return which serves to turn control over to ELIZA. ELIZA then analyzes the user's statement and generates some response which it types out. Control is then again with the user.

A typical conversation is the following:

Men are all alike.
IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED

Work reported herein was supported (in part) by Project MAC, an MIT research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01).

* Department of Electrical Engineering.

Computational Linguistics

A. G. OETTINGER, Editor

ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine

JOSEPH WEIZENBAUM
Massachusetts Institute of Technology, Cambridge, Mass.*

ELIZA is a program operating within the MAC time-sharing system at MIT which makes certain kinds of natural language conversation between man and computer possible. Input sentences are analyzed on the basis of decomposition rules which are triggered by key words appearing in the input text. Responses are generated by reassembly rules associated with selected decomposition rules. The fundamental technical problems with which ELIZA is concerned are: (1) the identification of

The object of this paper is to cause just such a re-evaluation of the program about to be "explained". Few programs ever needed it more.

ELIZA Program

ELIZA is a program which makes natural language conversation with a computer possible. Its present implementation is on the MAC time-sharing system at MIT. It is written in MAD-SLIP [4] for the IBM 7094. Its name was chosen to emphasize that it may be incrementally improved by its users, since its language abilities may be continually improved by a "teacher". Like the Eliza of Pygmalion fame, it can be made to appear even more civilized, the relation of appearance to reality, however, remaining in the domain of the playwright.

For the present purpose it is sufficient to characterize the MAC system as one which permits an individual to operate a full scale computer from a remotely located typewriter. The individual operator has the illusion that he is the sole user of the computer complex, while in fact others

Darstellung von Wissen

- 1960er
  - Suchtechniken
  - Maschinengestütztes Beweisen
    - Erzeugung und Überprüfung von mathematischen Beweisen logischer Theoreme
  - auf Schlussfolgemechanismen basiert
  - mit den richtigen Suchtechniken wären alle Probleme lösbar
- Ab den 1970er
  - Realisierung: es geht nicht
  - skaliert nicht
- Von Schlussfolgerung bis Wissen

Darstellung von Wissen

- 1980er
- Schlussfolgerung
  - nicht so zentral
- Erwerb und Darstellung von Wissen
  - Kernproblem
- Die Rolle der Expertin / des Experten
- Klarheit der Ausdrücke
  - Ausdrucksfähigkeit bzw. Effizienz

Darstellung von Wissen

- Ab Ende der 1980er
  - intractability
  - Ausdrucksfähigkeit *und* Effizienz nicht möglich?
- 1990er Jahre
  - Übergang zur Wissensdarstellung
    - (knowledge representation)
  - (heute: Ontologien, semantic Web, …)

Expertensysteme

- Mycin 1974
  - medizinische Diagnose
- Protokollanalyse

  ≈ User Stories
- Regelbasiert
- In Lisp-Programm definiert

```
Rule 52:
  If
    1) THE SITE OF THE CULTURE IS BLOOD
    2) THE GRAM OF THE ORGANISM IS NEG
    3) THE MORPHOLOGY OF THE ORGANISM IS ROD
    4) THE BURN OF THE PATIENT IS SERIOUS
  Then there is weakly suggestive evidence (0.4) that
    1) THE IDENTITY OF THE ORGANISM IS PSEUDOMONAS
```

```
(defrule 52
  if (site culture is blood)
     (gram organism is neg)
     (morphology organism is rod)
     (burn patient is serious)
  then .4
     (identity organism is pseudomonas))
```

Ungewissheit

- Wahrscheinlichkeitsfaktoren
  - fuzzy logic
- Kombination von ungewissen Fakten
  - Beispiel: Elvis lebt (in Zeitung)
  - Abhängige bzw. unabhängige Fakten
- Wo ist das Können
- Was ist das Können

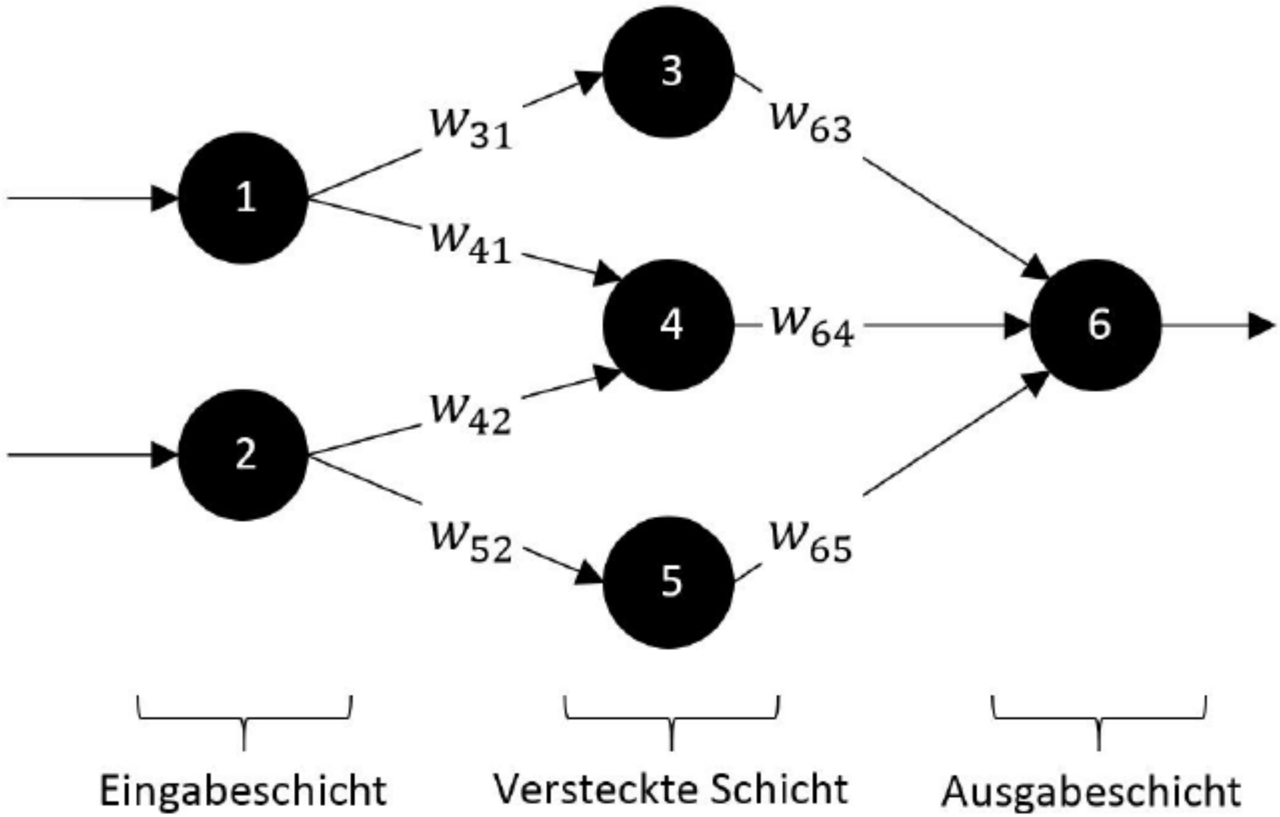
Sprachverarbeitung
(Natural language processing, NLP)

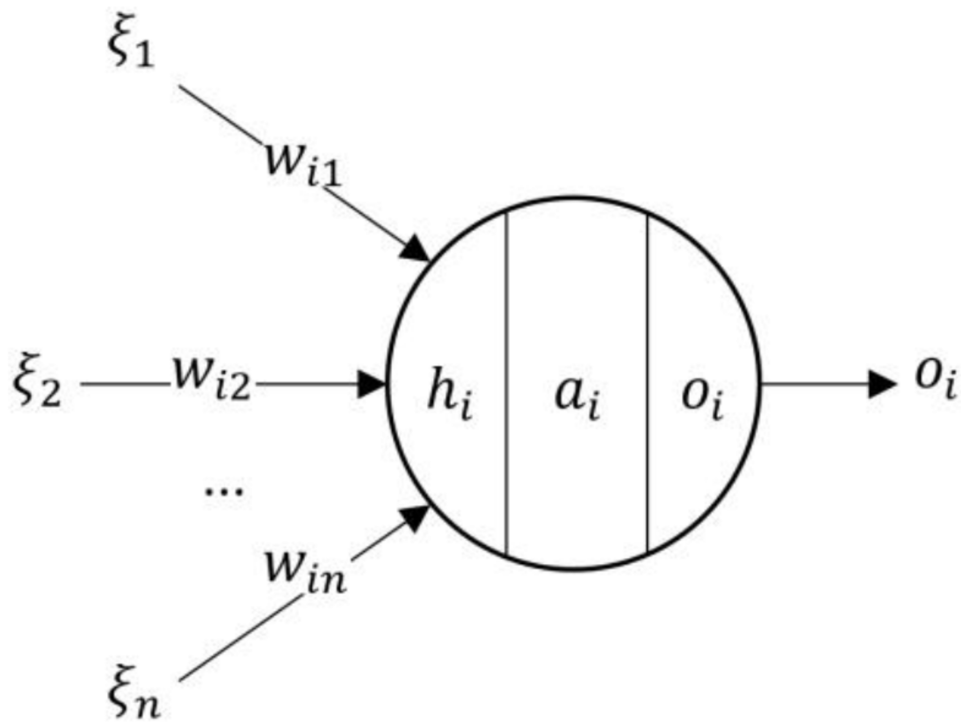
- Künstliche bzw. natürliche Sprache
  - Ambivalenz
- Sprache für Parsen
- Maschinenübersetzung
- Phrasenstrukturgrammatiken
- Maschinelles Lernen
- Statistikbasiert

The grounding problem

- Wie können KI-Systeme mit der externen Welt verbunden sein?
- Wie verbinden sich Menschen mit der externen Welt?
- Roboter
  - Industrie
  - Rasenmäher
  - Menschen-ähnliche
- Kontext
  - …is a spurious concept (Hirst)

Ein künstliches neuronales Netz





Einfaches Modell eines künstlichen Neurons *i* mit:

- hi:Summe der gewichteten Eingabeimpulse $wi1 \ldots win$, die das Neuron von verknüpften externen Einheiten oder von Sensoren ($\xi 1 \ldots \xi n$) erhält
- ai: Aktivierungsmaß
- $g(hi)$: Aktivierungsfunktion
- oi: Ausgabe des Neurons

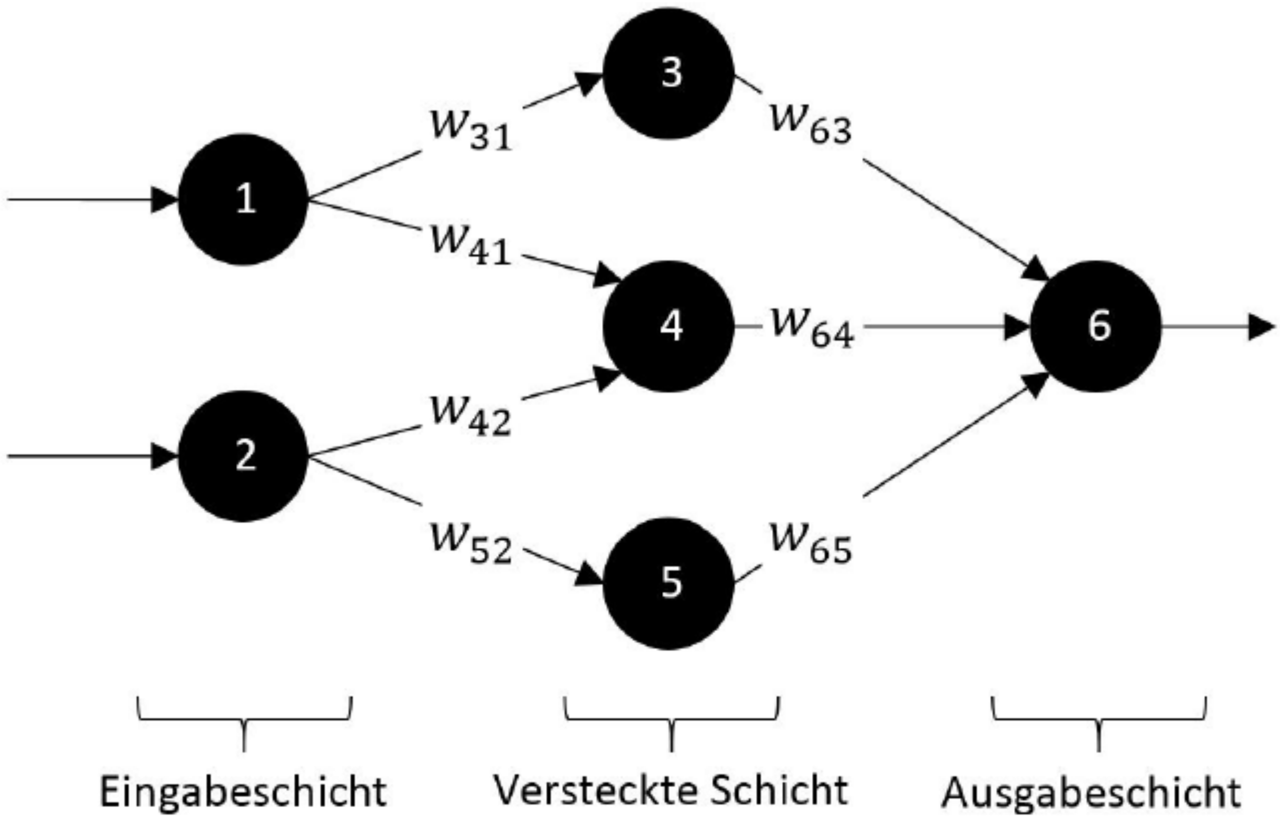
Verarbeitung der Eingabeimpulse $wi_1 \ldots wi_n$ erfolgt in zwei Schritten:

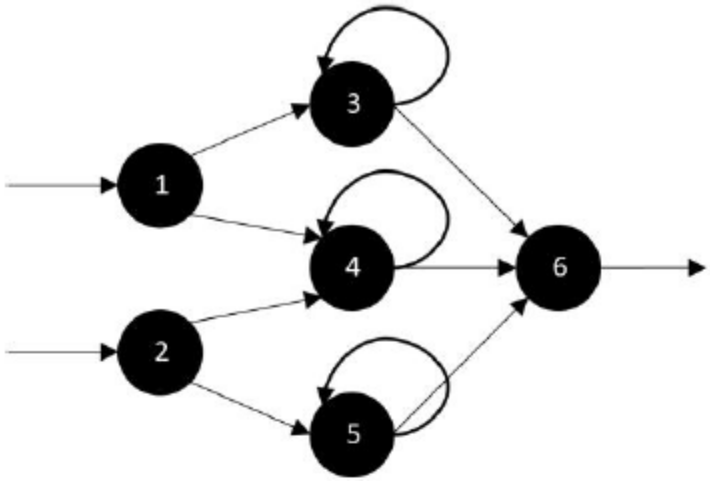
- h_i: gewichtete Summe der Eingabeimpulse bestimmen, die an dem Neuron anliegen:

$$h_i = \sum_{j=1}^{n} w_{ij} o_j$$

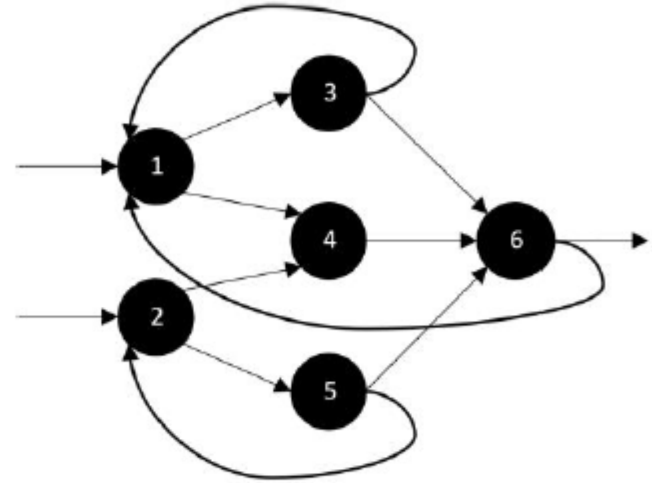
- Bestimmung des Aktivierungsmaßes (in den meisten Fällen: Ausgabe) des Neurons:

$$a_i = g(h_i) = g\left(\sum_{j=1}^{n} w_{ij} o_j\right)$$

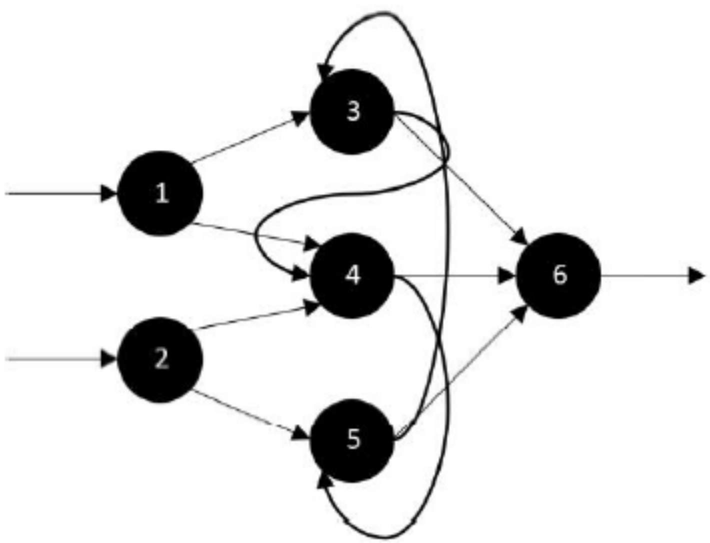

1
2
3
4
5
6
w_{31}
w_{41}
w_{42}
w_{52}
w_{63}
w_{64}
w_{65}
Eingabeschicht
Versteckte Schicht
Ausgabeschicht



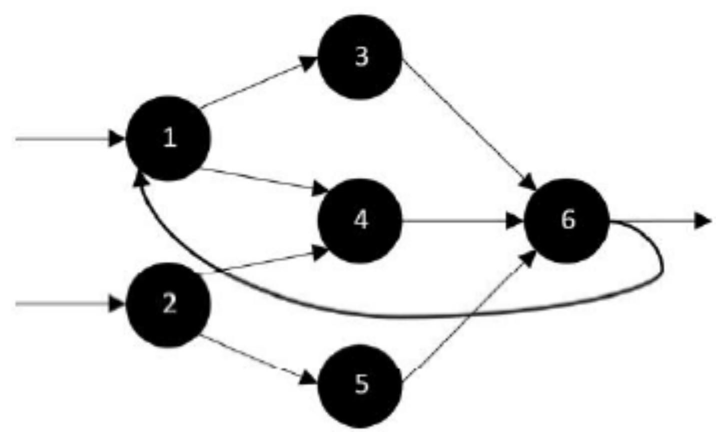
a) Netz mit direkt rückgekoppelten Neuronen in der versteckten Schicht.



b) Neuronales Netz mit indirekten Rückkopplungen.



c) Lateral rückgekoppeltes Netz.



d) Extern rekurrentes Netz.

Abbildung 36: Arten der Rückkopplung in einem Künstlichen neuronalen Netz.

Verstärkendes Lernen
Q-Learning

Endlicher Markov-Entscheidungsprozess

→ Verlauf von Entscheidungen, bei dem die Belohnung des Agenten einzig von dem Zustand der Umwelt und der Aktion des Agenten abhängt. Definiert als Tupel (S, A, P, R, γ) mit:

- $S = \{s^1, s^2, \dots, s^n\}$: Zustandsraum, d.h. die möglichen Zustände der Umwelt; $s_t \in S$ repräsentiert den Zustand der Agentenumwelt zum Zeitpunkt t.

Endlicher Markov-Entscheidungsprozess

→ Verlauf von Entscheidungen, bei dem die Belohnung des Agenten einzig von dem Zustand der Umwelt und der Aktion des Agenten abhängt. Definiert als Tupel (S, A, P, R, γ):

- $S = \{s^1, s^2, \dots, s^n\}$: Zustandsraum, d.h. die möglichen Zustände der Umwelt; $s_t \in S$ repräsentiert den Zustand der Agentenumwelt zum Zeitpunkt t.

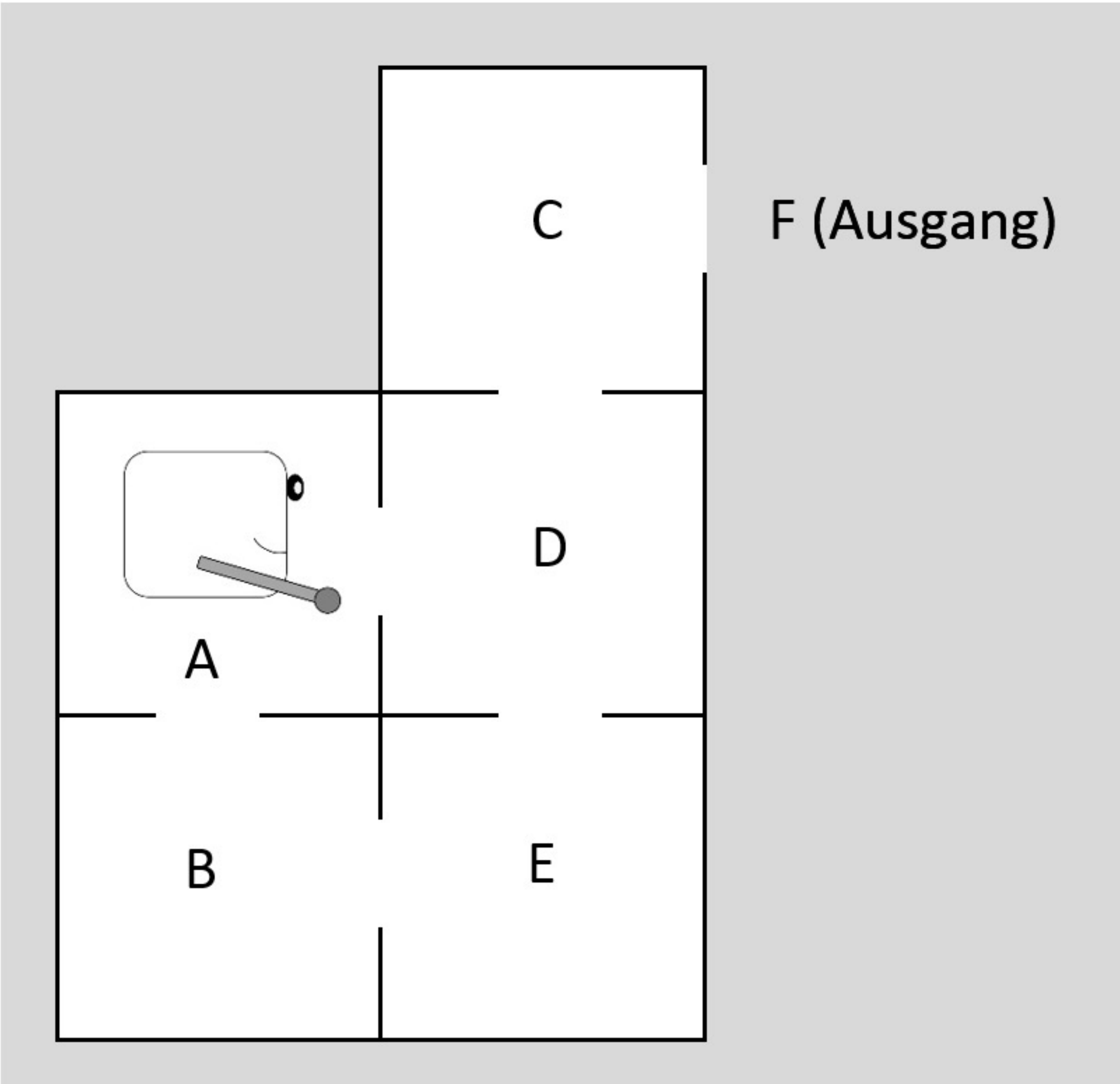
- A: Menge der Entscheidungen bzw. Zugmöglichkeiten, die dem Agenten zu jedem Umweltzustand $A(s)$ zur Verfügung stehen.

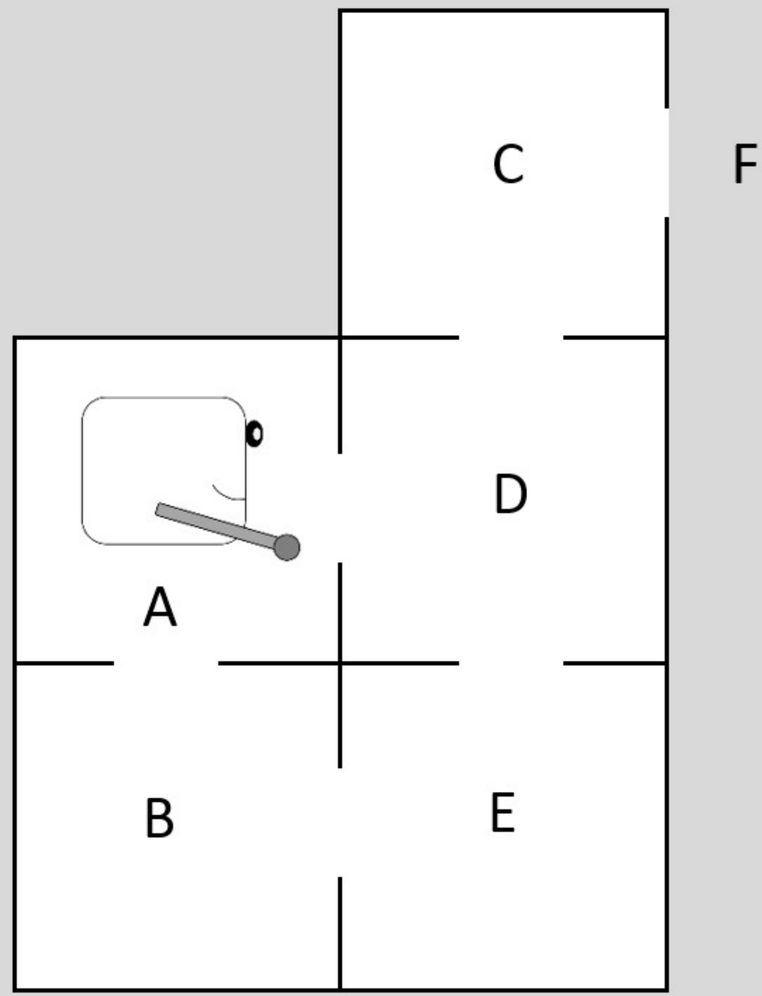
Endlicher Markov-Entscheidungsprozess

(S, A, P, R, γ):

- $a_t \in A(s_t)$: von dem Agenten ausgeführte Aktion zum Zeitpunkt t.
- Übergangsfunktion $P(s, a, s')$: führt die Umwelt in ihrem Zustand s aufgrund der Handlung a des Agenten zu einem neuen möglichen Nachfolgezustand s'.
- $R(s, a, s')$: Belohnungsfunktion, Feedback, das der Agent aufgrund seiner Handlung erhält.
- Diskontierungsfaktor γ mit $0 \leq \gamma \leq 1$ lässt sich dazu verwenden, um das Lernverhalten des Agenten feinzujustieren. Dient dazu, künftige Belohnungen gegenüber zeitnahen Belohnungen abzuschwächen.

$$\pi^*(s) \to a^*|s$$


C
F (Ausgang)
D
A
B
E



Zustände / Räume

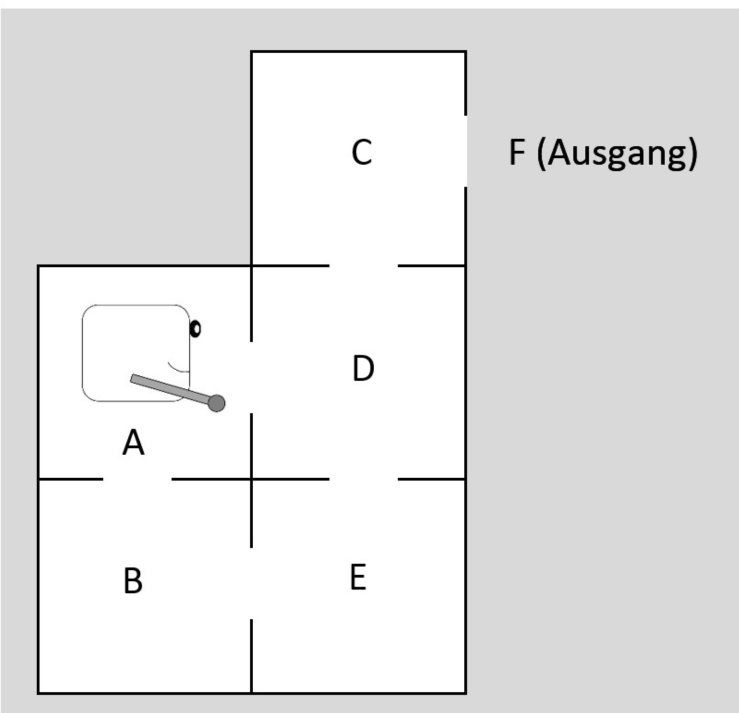
$$R = \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & 0 & - & 0 & - & - \\ 0 & - & - & - & 0 & - \\ - & - & - & 0 & - & 0 \\ 0 & - & 0 & - & 0 & - \\ - & 0 & - & 0 & - & - \\ - & - & 100 & - & - & 100 \end{bmatrix} \end{array} \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}$$

Aktionen / Zustandsübergänge

Zustände / Räume

$$R = \begin{array}{c c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} & \\ \begin{bmatrix} - & 0 & - & 0 & - & - \\ 0 & - & - & - & 0 & - \\ - & - & - & 0 & - & 0 \\ 0 & - & 0 & - & 0 & - \\ - & 0 & - & 0 & - & - \\ - & - & 100 & - & - & 100 \end{bmatrix} & \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} \end{array}$$

Aktionen / Zustandsübergänge

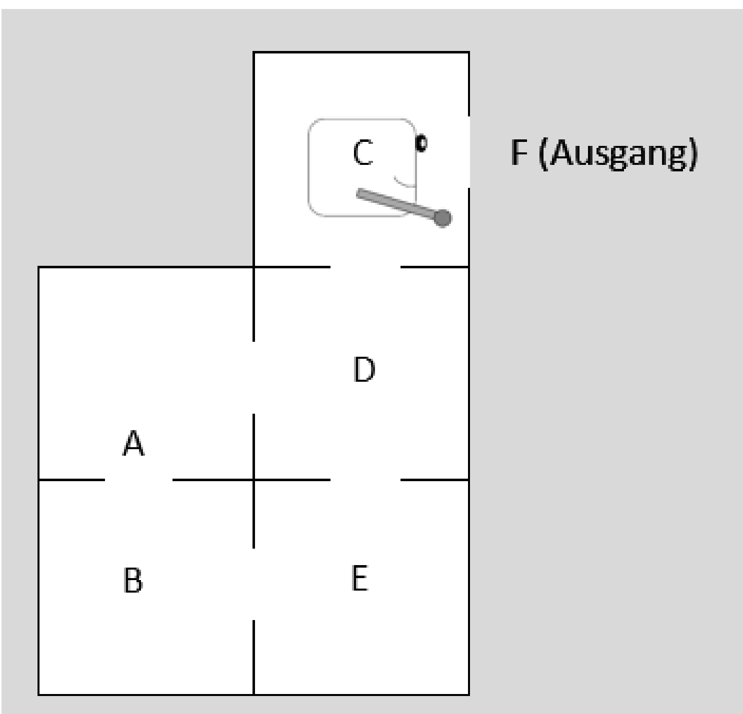


$$Q = \begin{array}{c c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} & \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} \end{array}$$

Zustände / Räume

$$R = \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & 0 & - & 0 & - & - \\ 0 & - & - & - & 0 & - \\ - & - & - & 0 & - & 0 \\ 0 & - & 0 & - & 0 & - \\ - & 0 & - & 0 & - & - \\ - & - & 100 & - & - & 100 \end{bmatrix} \end{array} \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}$$

Aktionen / Zustandsübergänge



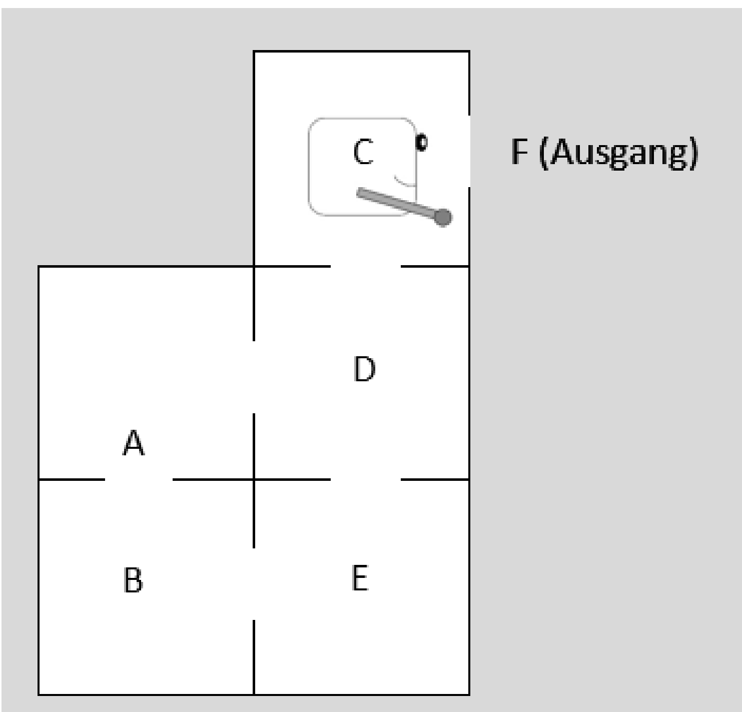
$$Q = \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}$$

$$
\begin{aligned}
&Q(C,F)\\
&= R(C,F) + 0.5\times Max[Q(F,C), Q(F,F)]\\
&= 100 + 0.5\times Max[0,0] = 100 + 0.5\times 0\\
&= 100.
\end{aligned}
$$

Zustände / Räume

$$
R = \begin{array}{c}
\begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\left[\begin{array}{cccccc}
- & 0 & - & 0 & - & - \\
0 & - & - & - & 0 & - \\
- & - & - & 0 & - & 0 \\
0 & - & \boxed{0} & - & 0 & - \\
- & 0 & - & 0 & - & - \\
- & - & \boxed{100} & - & - & 100
\end{array}\right]
\end{array}
\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
$$

Aktionen / Zustandsübergänge



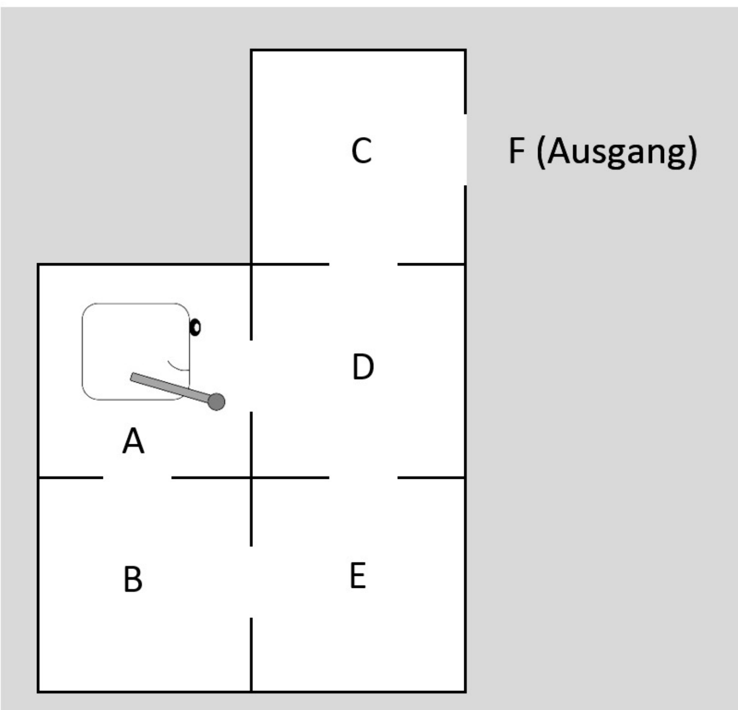
$$
Q = \begin{array}{c}
\begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\left[\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 100 & 0 & 0 & 0
\end{array}\right]
\end{array}
\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
$$

Zustände / Räume

$$R = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{cccccc} A & B & C & D & E & F \end{array}$$

$$R = \left[\begin{array}{cccccc} - & 0 & - & 0 & - & - \\ 0 & - & - & - & 0 & - \\ - & - & - & 0 & - & 0 \\ 0 & - & \boxed{0} & - & 0 & - \\ - & 0 & - & 0 & - & - \\ - & - & \boxed{100} & - & - & 100 \end{array}\right] \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array}$$

Aktionen / Zustandsübergänge



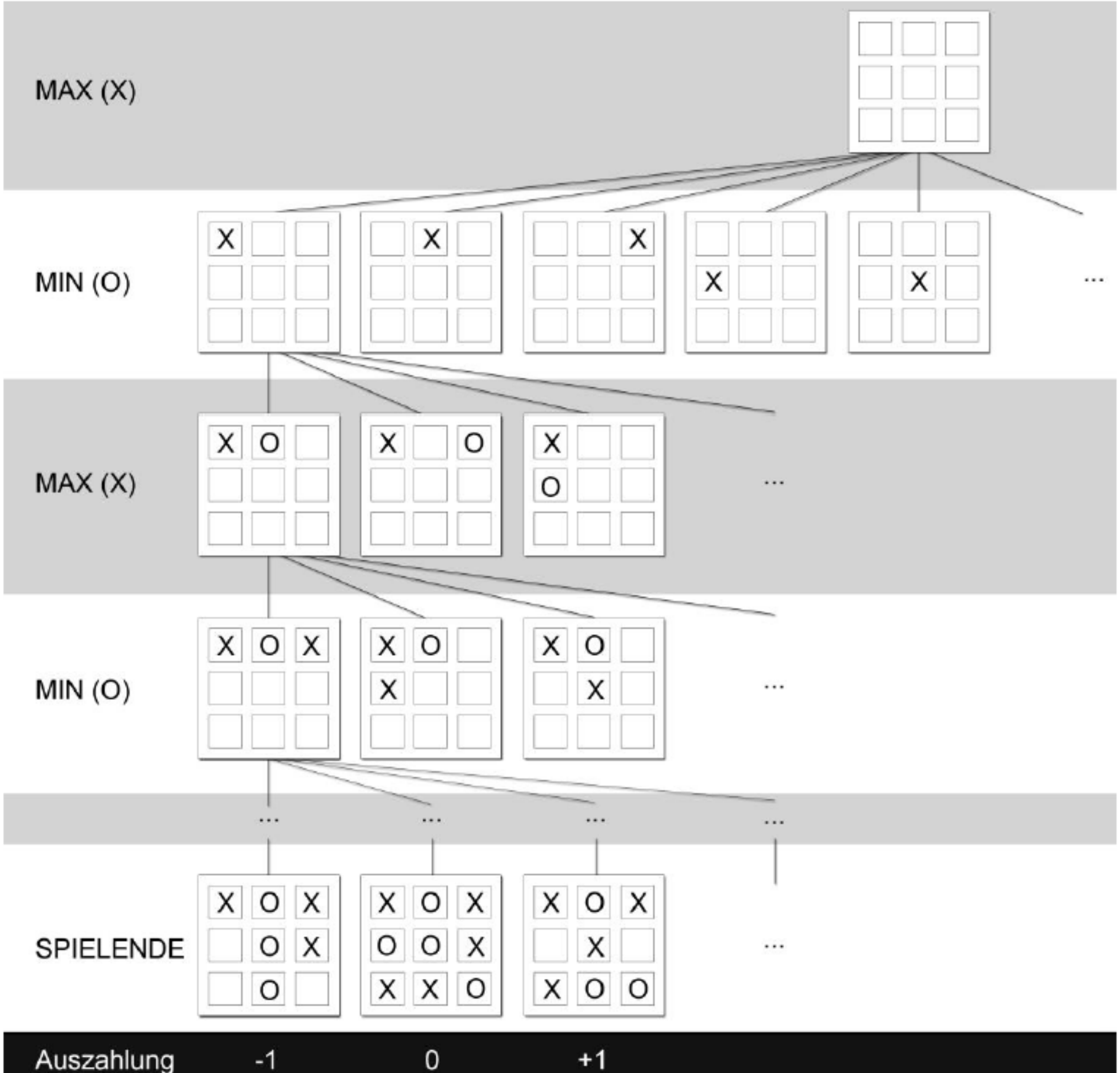
$$\begin{array}{cccccc} A & B & C & D & E & F \end{array}$$

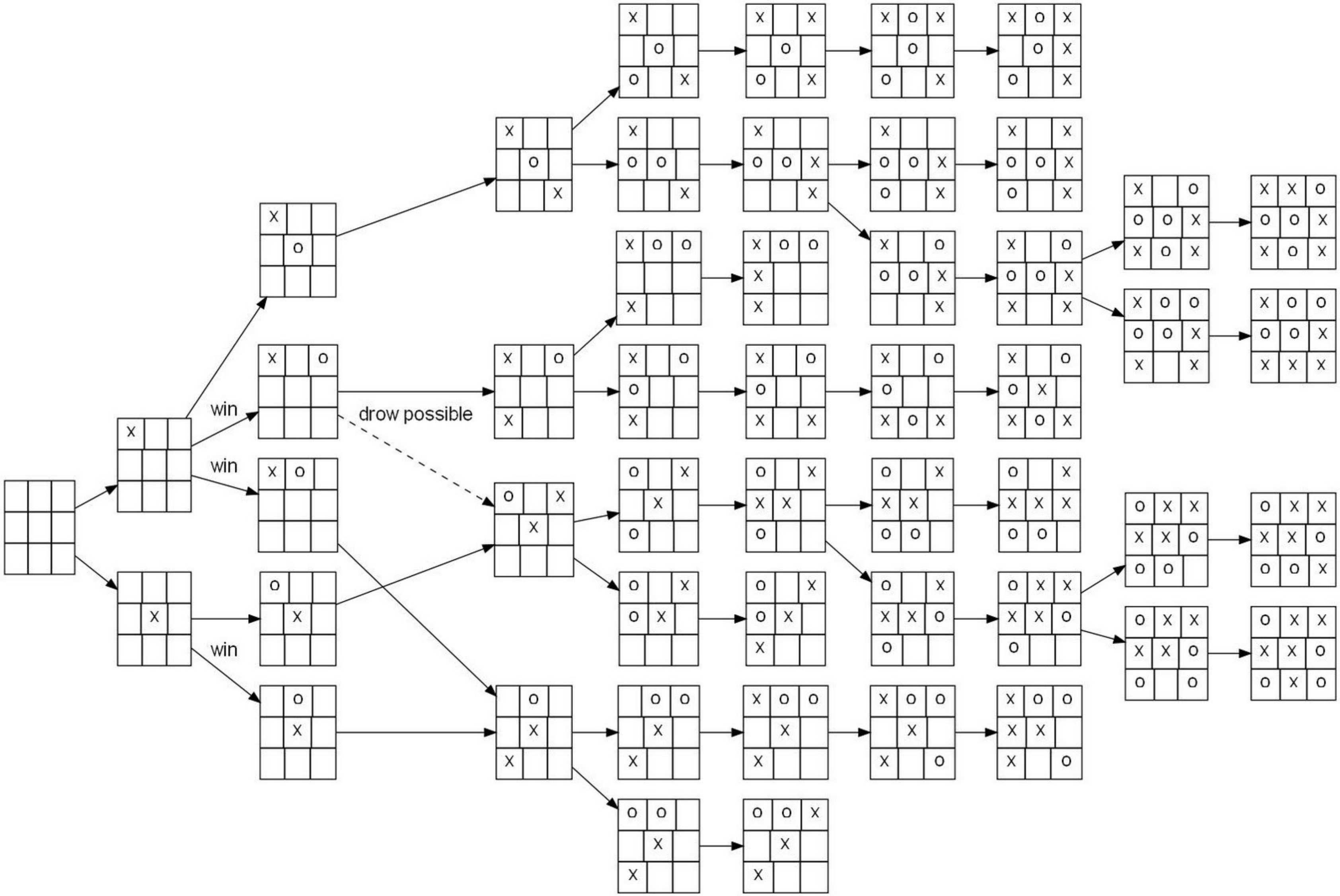
$$Q = \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \end{array}\right] \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array}$$

Spiele

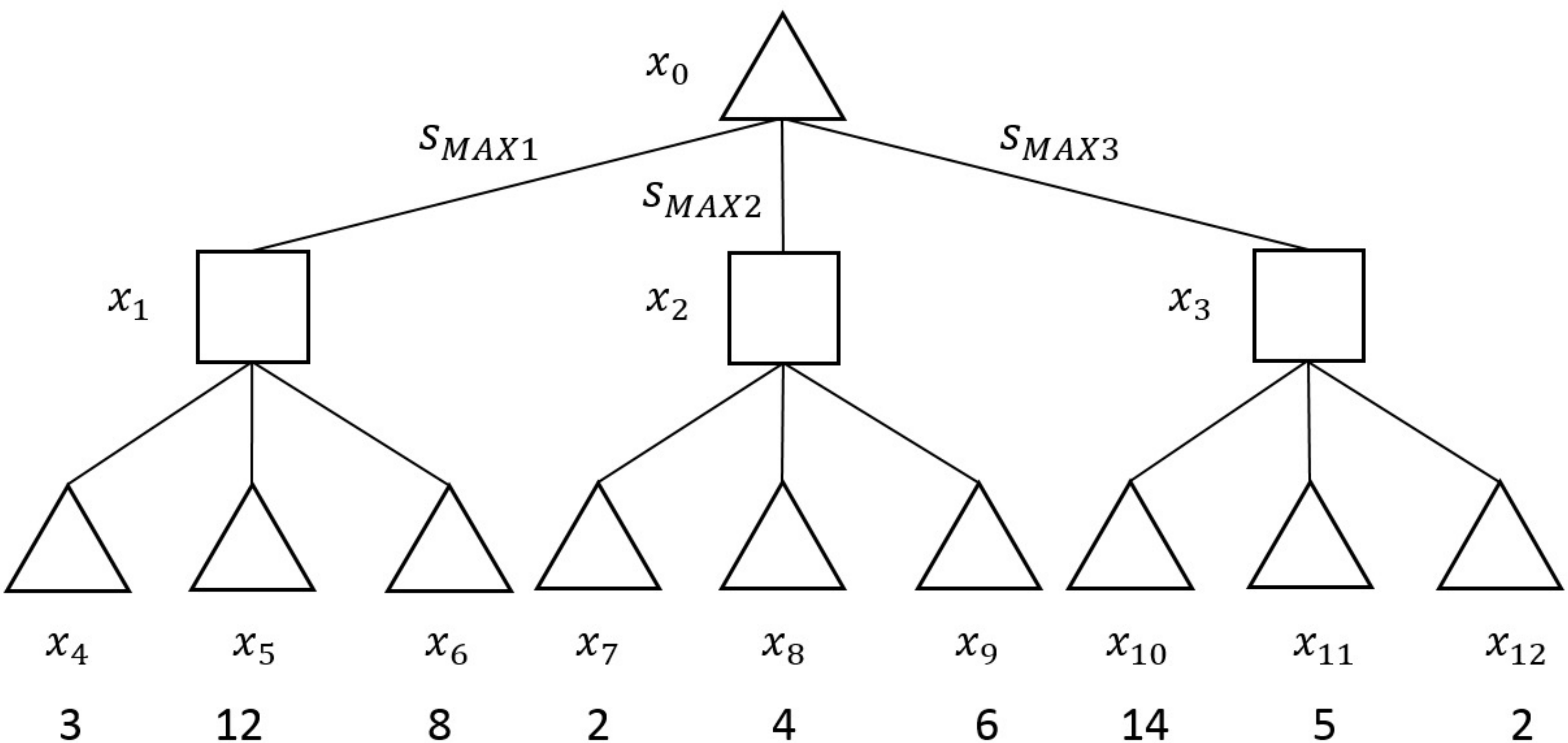
- Spielen als suchen
- Vom Sieger zum Sieger
  - Othello (1980er)
  - Schack (1997)
  - Go (2016)
- Können Rechner alle Spiele spielen?

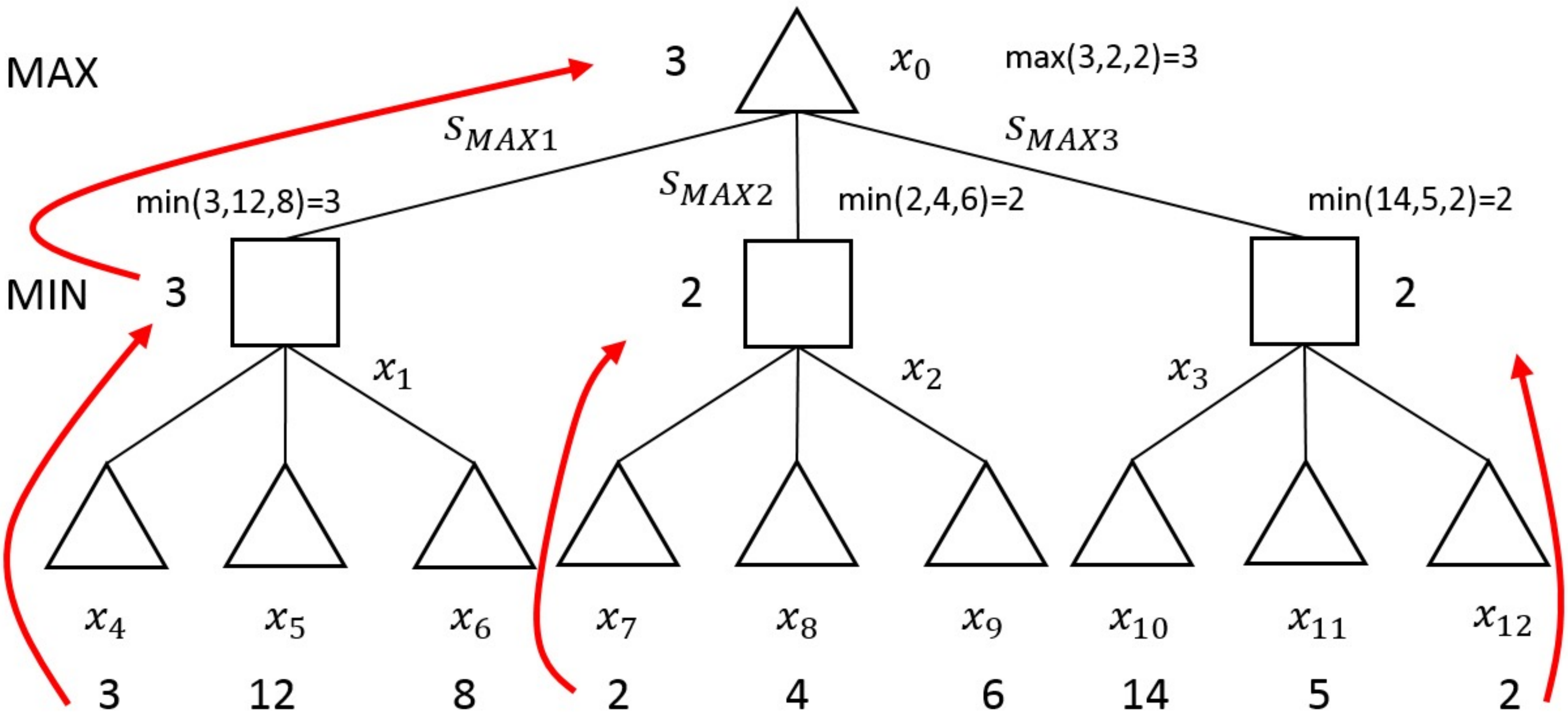
Monte Carlo Spielbaumsuche

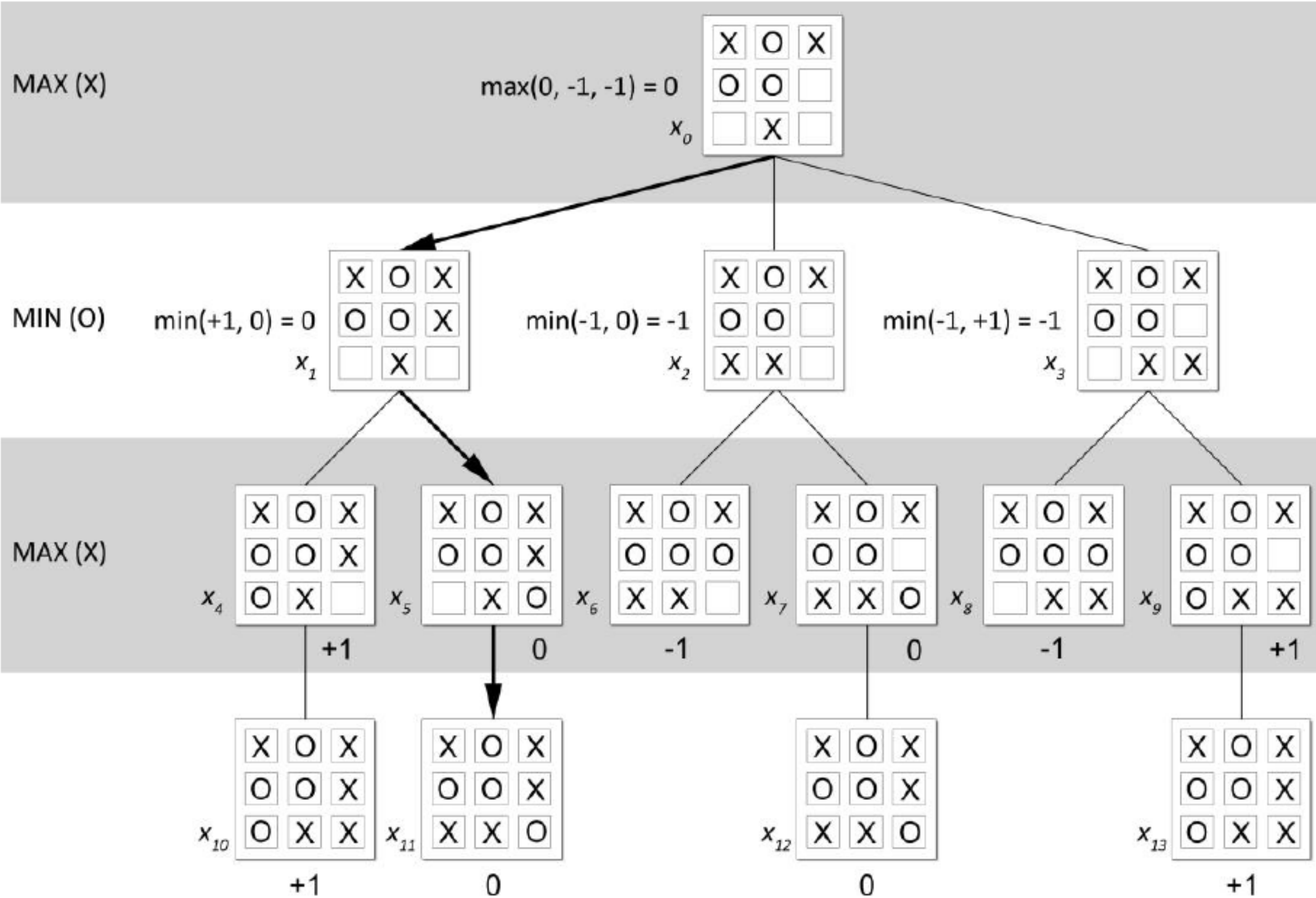

MAX (X)
MIN (O)
MAX (X)
MIN (O)
SPIELENDE
Auszahlung
-1
0
+1

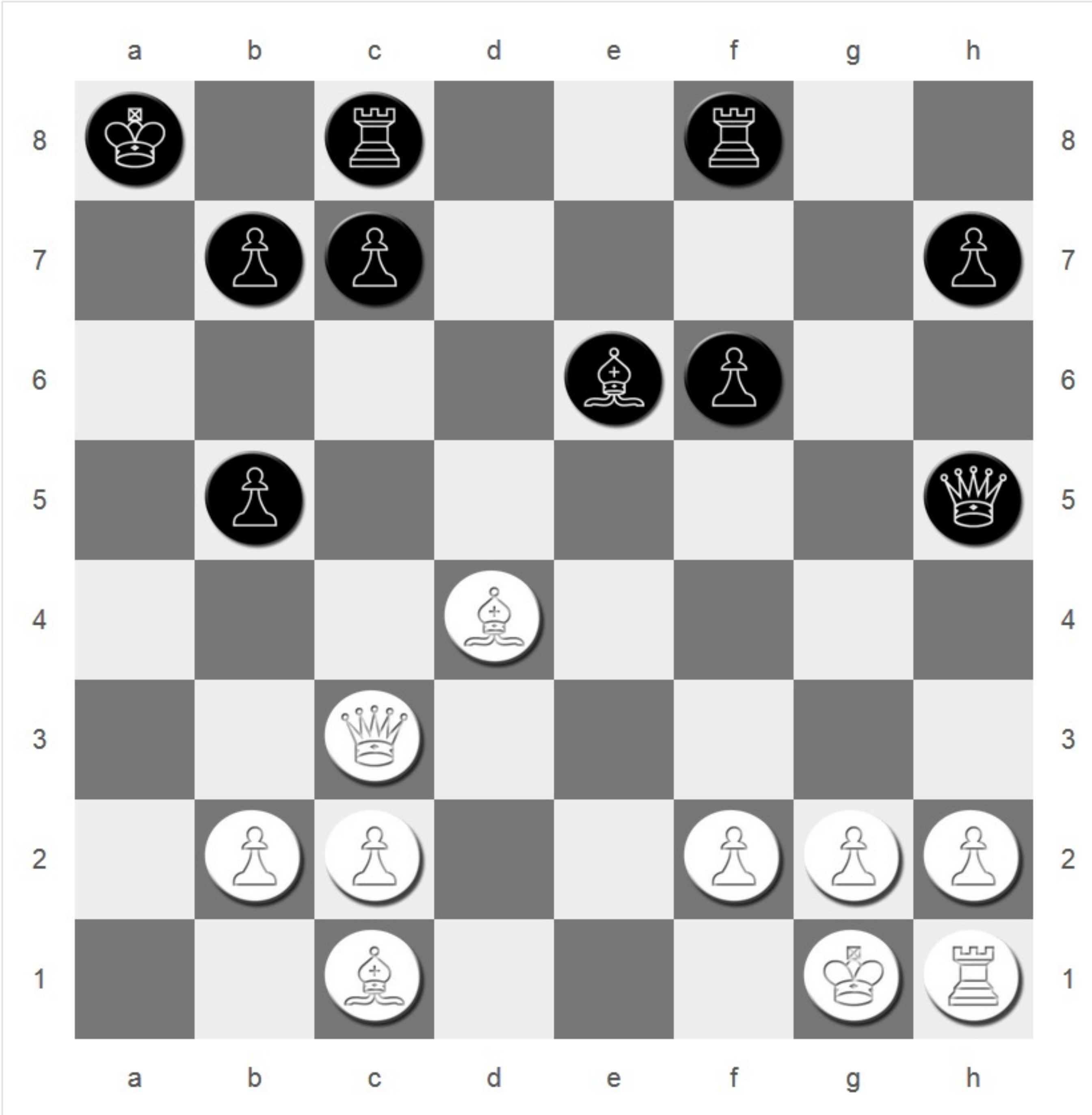


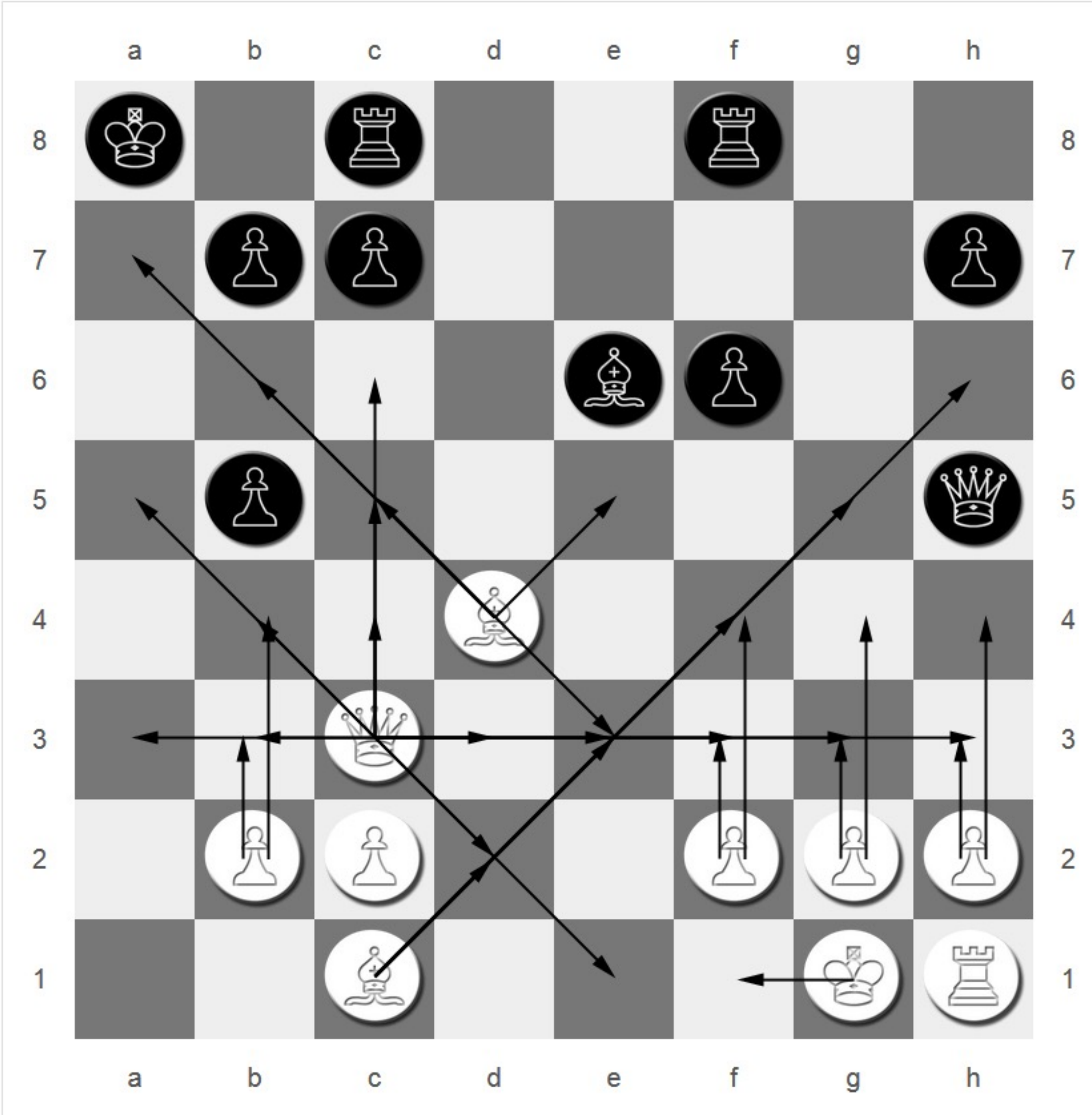
Quelle: Juul, Jesper: „255,168 ways of playing Tic Tac Toe" (http://www.jesperjuul.net/ludologist/255168-ways-of-playing-tic-tac-toe)


x_0
s_{MAX1}
s_{MAX2}
s_{MAX3}
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
x_{10}
x_{11}
x_{12}
3
12
8
2
4
6
14
5
2


MAX
MIN
3
x_0
max(3,2,2)=3
s_{MAX1}
s_{MAX2}
s_{MAX3}
min(3,12,8)=3
min(2,4,6)=2
min(14,5,2)=2
3
2
2
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
x_{10}
x_{11}
x_{12}
3
12
8
2
4
6
14
5
2


MAX (X)
max(0, -1, -1) = 0
x_0
MIN (O)
min(+1, 0) = 0
x_1
min(-1, 0) = -1
x_2
min(-1, +1) = -1
x_3
MAX (X)
x_4
x_5
x_6
x_7
x_8
x_9
+1
0
-1
0
-1
+1
x_{10}
x_{11}
x_{12}
x_{13}
+1
0
0
+1


a b c d e f g h
8 7 6 5 4 3 2 1


a b c d e f g h
8 7 6 5 4 3 2 1

Was ist es, zu spielen?

We now know that Lee Sedol lost to AlphaGo (1-4). [...] This is an important achievement for Artificial Intelligence (AI). It is, however, yet another example of how computational AI achievements like this messes with the way we talk about what humans do and what machines do, such that machines look at least as good as humans, if not better, at doing things humans do. There are stark and strong differences between what Lee Sedol did and what AlphaGo did. [...]

Lee Sedol can explain to anybody who asks what Go is, how it is played, how he plays Go, including that there are mysterious hardly utterable aspects to his way of playing, how he felt about each game against AlphaGo, and he can teach others how to play Go, comment usefully on their efforts, and promote the game. AlphaGo can do none of these things. [...]

As I said, this is no doubt an important achievement for AI, but it hardly scratches the surface of Human intelligence and of what being Human is. What would be nice is if we Humans, took a little more care to keep this clear and out front when we talk about these Human vs Machines contests. [...]

Tim Smithers, Humanist, 16 Mar 2016. Others disagreed.


DOUBLE
BLUE

Mehrarmiges-Banditen-Problem: UCB1 (Auer et al. 1995)

$$UCB1 = \overline{X}_j + \sqrt{\frac{2\ln(n)}{n_j}}$$

$\overline{X}_j$: durchschnittliche Auszahlung des Automatenarmes j; n_j: Anzahl, wie häufig der Arm j gespielt wurde
n : Gesamtzahl der bereits gespielten Spiele.

Browne et al. 2012: Belohnungsterm $\overline{X}_j$: Exploitation
$\sqrt{\frac{2\ln(n)}{n_j}}$: Exploration bislang vernachlässigter Strategiewahlen.

Upper Confidence Bounds applied to Trees (Kocsis und Szepesvári 2006)

UCT wählt den Kindknoten k des aktuell betrachteten Knotens p aus, der den folgenden Ausdruck maximiert:

$$k \in argmax_{i \in I}\left(v_i + 2 \times C\sqrt{\frac{2 \times \ln(n_p)}{n_i}}\right)$$

n_p : Besuchshäufigkeit des Elternknotens p
n_i : Besuchsfrequenz des Kindknotens i
v_i : Wert des Knotens i (durchschnittliche Gewinnhäufigkeit des betrachteten Kindknotens
C im Intervall [0,1] dient dem Zweck, den Grad der Erkundung (*exploration*) noch nicht betrachteter Strategien feinzujustieren

Ethik

- Angst vor dem Rechner
- Überwachung
- Bots und Destabilisierung
- Filterblasen
- Lernbasis
- Wer macht Entscheidungen?
- Was sind die Grundlagen für Entscheidungen?
- Autonomie
  - Autos
- Verantwortung
- Relevant für uns? Für Sie?

/