

Tic-Tac-Toe like a Pro

Vortrag zum Thema Spielbäume

Joschka Heinrich

		X
X	O	
		O

Proseminar Theoretische Informatik

Fakultät Informatik, TU Dresden

15.02.2018

Tic-Tac-Toe like a Bro

Vortrag zum Thema Spielbäume

Joschka Heinrich

		X
X	O	
		O

Proseminar Theoretische Informatik

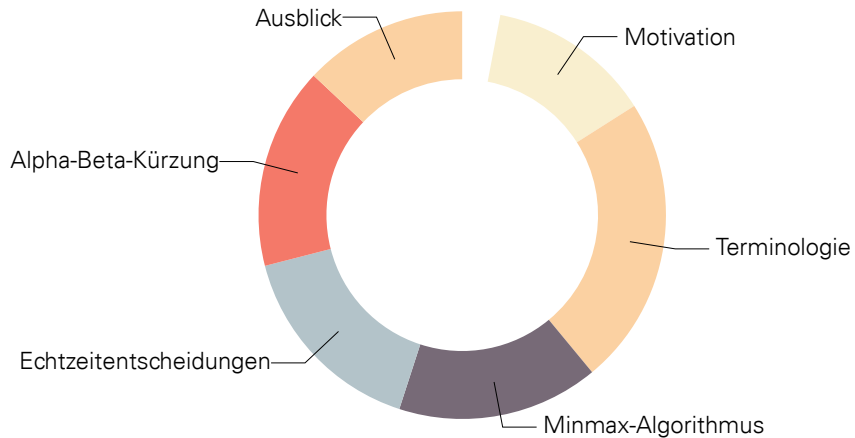
Fakultät Informatik, TU Dresden

15.02.2018

Wie wähle ich in einem Spiel den günstigsten Zug?

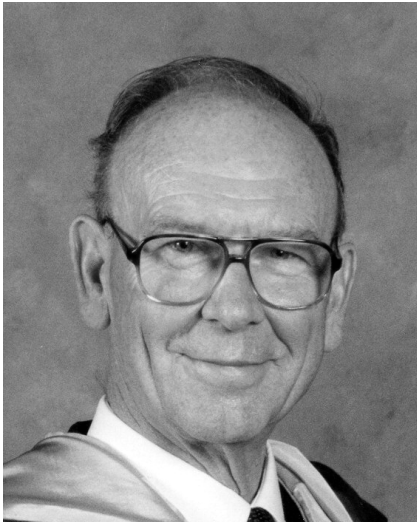
Agenda

Tic-Tac-Toe like a Pro



Donald Michies „Menace“

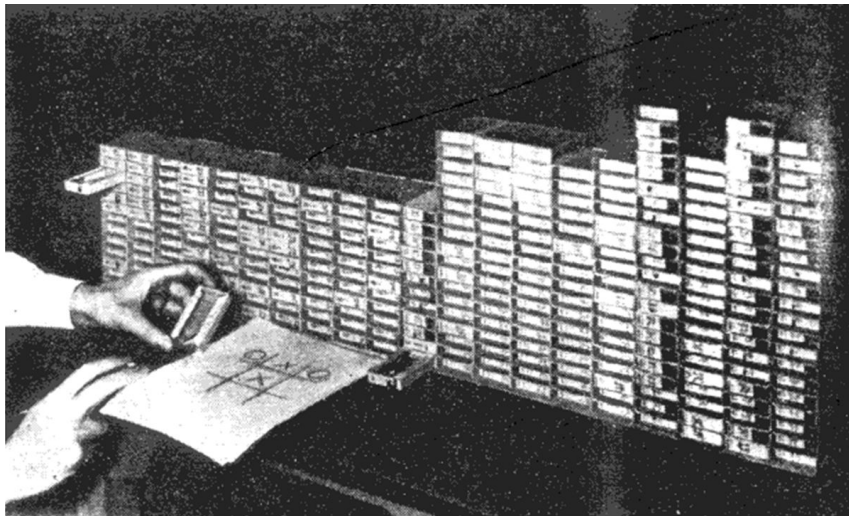
Motivation



_ Donald Michies, 2003 ¹

Donald Michies „Menace“

Motivation



_ Menace, 1963²

Lernziele dieses Vortrages

Motivation

— formale Betrachtung von **Spiele**n verstehen

Lernziele dieses Vortrages

Motivation

- formale Betrachtung von **Spiele**n verstehen
- **Minmax-Algorithmus** nachvollziehen können

Lernziele dieses Vortrages

Motivation

- formale Betrachtung von **Spiele**n verstehen
- **Minmax-Algorithmus** nachvollziehen können
- gute und schlechte **Heuristiken** unterscheiden und anwenden können

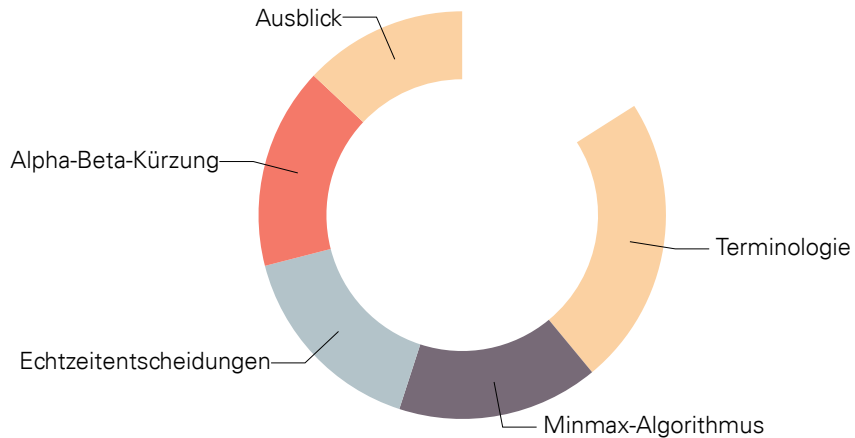
Lernziele dieses Vortrages

Motivation

- formale Betrachtung von **Spiele**n verstehen
- **Minmax-Algorithmus** nachvollziehen können
- gute und schlechte **Heuristiken** unterscheiden und anwenden können
- Vorteile der **Alpha-Beta-Kürzung** verstehen

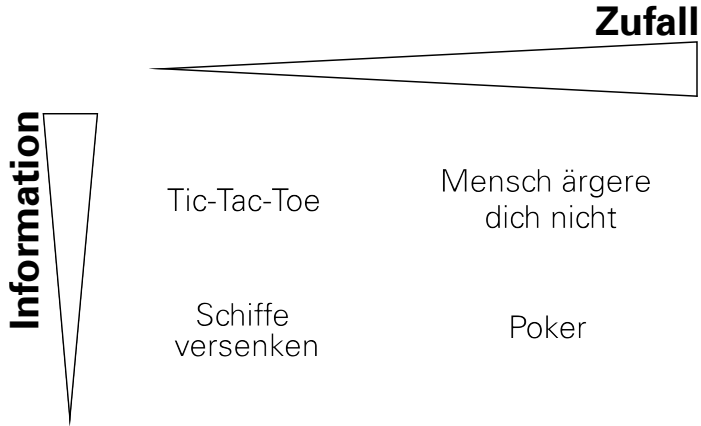
Terminologie

Spiel, Konfiguration, Spielbaum



Zufall und Information

Terminologie



Spiel, Spielzug, Konfiguration

Terminologie

— zwei Spieler: **Max** und **Min**

Spiel, Spielzug, Konfiguration

Terminologie

- zwei Spieler: **Max** und **Min**
- Nullsummenspiel

Spiel, Spielzug, Konfiguration

Terminologie

- zwei Spieler: **Max** und **Min**
- Nullsummenspiel
- K Menge aller **zulässigen Konfigurationen**

Spiel, Spielzug, Konfiguration

Terminologie

- zwei Spieler: **Max** und **Min**
- Nullsummenspiel
- K Menge aller **zulässigen Konfigurationen**
- Spiel $S := (R, k_0, F)$

Spiel, Spielzug, Konfiguration

Terminologie

- zwei Spieler: **Max** und **Min**
- Nullsummenspiel
- K Menge aller **zulässigen Konfigurationen**
- Spiel $S := (R, k_0, F)$
 - R Menge **legaler Spielzüge**

Spiel, Spielzug, Konfiguration

Terminologie

- zwei Spieler: **Max** und **Min**
- Nullsummenspiel
- K Menge aller **zulässigen Konfigurationen**
- Spiel $S := (R, k_0, F)$
 - R Menge **legaler Spielzüge**
 - $k_0 \in K$ **Anfangskonfiguration**

Spiel, Spielzug, Konfiguration

Terminologie

- zwei Spieler: **Max** und **Min**
- Nullsummenspiel
- K Menge aller **zulässigen Konfigurationen**
- Spiel $S := (R, k_0, F)$
 - R Menge **legaler Spielzüge**
 - $k_0 \in K$ **Anfangskonfiguration**
 - $F \subset K$ Menge der **Endkonfigurationen**

Spielzüge als Funktion, Folgekonfiguration

Terminologie

— $R : K^2, r$ legal

Spielzüge als Funktion, Folgekonfiguration

Terminologie

- $R : K^2, r$ legal
 - wenn $v = r(u)$, dann heißt v **Kindkonfiguration** von u (bezüglich r)
 - analog: u **Elternkonfiguration** von v (bezüglich r)

Spielzüge als Funktion, Folgekonfiguration

Terminologie

- $R : K^2, r$ legal
 - wenn $v = r(u)$, dann heißt v **Kindkonfiguration** von u (bezüglich r)
 - analog: u **Elternkonfiguration** von v (bezüglich r)
- $R_k := \{r \in R \mid r \text{ anwendbar auf } k\}$

Spielzüge als Funktion, Folgekonfiguration

Terminologie

- $R : K^2, r$ legal
 - wenn $v = r(u)$, dann heißt v **Kindkonfiguration** von u (bezüglich r)
 - analog: u **Elternkonfiguration** von v (bezüglich r)
- $R_k := \{r \in R \mid r \text{ anwendbar auf } k\}$
- **Menge der Kindkonfigurationen** $N(k) := \{r(k) \mid r \in R_k\} \subseteq K$

Spielzüge als Funktion, Folgekonfiguration

Terminologie

- $R : K^2, r$ legal
 - wenn $v = r(u)$, dann heißt v **Kindkonfiguration** von u (bezüglich r)
 - analog: u **Elternkonfiguration** von v (bezüglich r)
- $R_k := \{r \in R \mid r \text{ anwendbar auf } k\}$
- **Menge der Kindkonfigurationen** $N(k) := \{r(k) \mid r \in R_k\} \subseteq K$
 - $N(k_f) = \emptyset, k_f \in F$

Spielzüge als Funktion, Folgekonfiguration

Terminologie

- $R : K^2$, r legal
 - wenn $v = r(u)$, dann heißt v **Kindkonfiguration** von u (bezüglich r)
 - analog: u **Elternkonfiguration** von v (bezüglich r)
- $R_k := \{r \in R \mid r \text{ anwendbar auf } k\}$
- **Menge der Kindkonfigurationen** $N(k) := \{r(k) \mid r \in R_k\} \subseteq K$
 - $N(k_f) = \emptyset$, $k_f \in F$
- K induktiv über Kindkonfiguration definierbar:

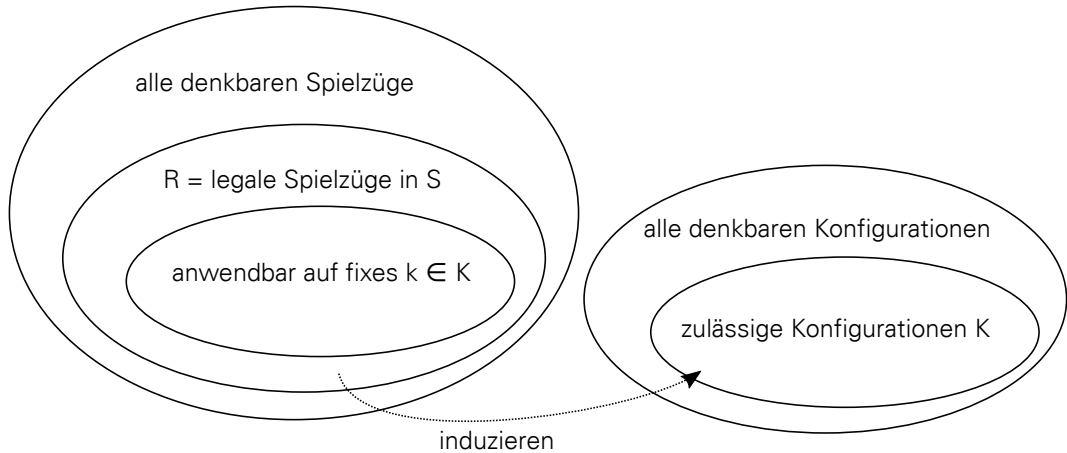
Spielzüge als Funktion, Folgekonfiguration

Terminologie

- $R : K^2$, r legal
 - wenn $v = r(u)$, dann heißt v **Kindkonfiguration** von u (bezüglich r)
 - analog: u **Elternkonfiguration** von v (bezüglich r)
- $R_k := \{r \in R \mid r \text{ anwendbar auf } k\}$
- **Menge der Kindkonfigurationen** $N(k) := \{r(k) \mid r \in R_k\} \subseteq K$
 - $N(k_f) = \emptyset$, $k_f \in F$
- K induktiv über Kindkonfiguration definierbar:
 - (1) $k_0 \in K$
 - (2) $\forall k \in K : N(k) \subset K$

mögliche, legale und anwendbare Spielzüge

Terminologie



— **Spielbaum** ist gerichteter Graph $G_S := (V, E)$

Spielbaum

Terminologie

— **Spielbaum** ist gerichteter Graph $G_S := (V, E)$

— Knoten $V = K$

Spielbaum

Terminologie

- **Spielbaum** ist gerichteter Graph $G_S := (V, E)$
 - Knoten $V = K$
 - Kanten $E = \bigcup_{u \in K} \{(u, v) \mid v \in N(u)\}$

Spielbaum

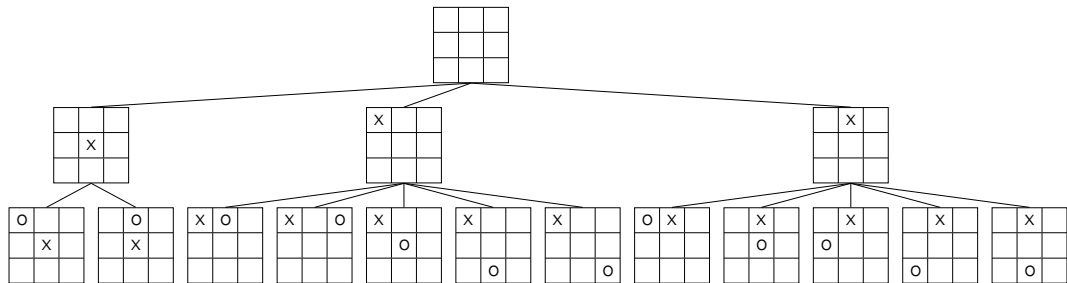
Terminologie

- **Spielbaum** ist gerichteter Graph $G_S := (V, E)$
 - Knoten $V = K$
 - Kanten $E = \bigcup_{u \in K} \{(u, v) \mid v \in N(u)\}$
- G_S ist Baum mit Wurzel k_0 , Blättern F

- **Spielbaum** ist gerichteter Graph $G_S := (V, E)$
 - Knoten $V = K$
 - Kanten $E = \bigcup_{u \in K} \{(u, v) \mid v \in N(u)\}$
- G_S ist Baum mit Wurzel k_0 , Blättern F
- Spielverlauf mit l Zügen ist Pfad p subsequenter Kindkonfigurationen
 - $p = (k_0, k_1, \dots, k_l)$
- zu große Komplexität

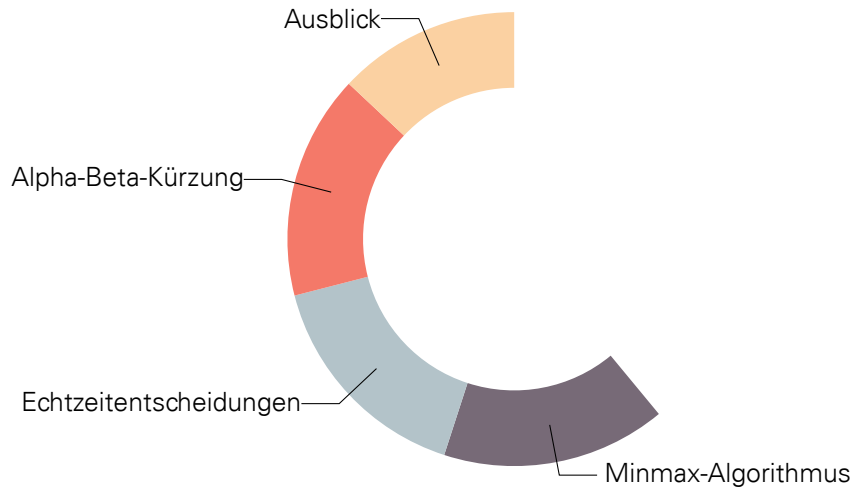
Tic-Tac-Toe-Spielbaum der Tiefe 2

Terminologie



Minmax

Geschichte, Algorithmus, Komplexität



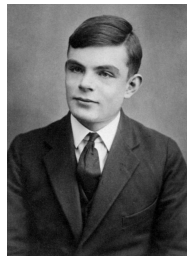
Geschichte des Minmax-Algorithmus

Minmax

- _ populär als Suchalgorithmus für Spielstrategie in Schachprogrammen
- _ 1912 erstmals von Ernst Zermelo erwähnt
- _ 1940er: Bedeutung von Minmax betont (Shannon, Turing)



_ Claude Shannon, 1963³



_ Alan Turing, 1928⁴



Voraussetzungen für den Algorithmus

Minmax

— zwei Typen von Knoten



Voraussetzungen für den Algorithmus

Minmax

- zwei Typen von Knoten
 - Max ist am Zug: **Max-Knoten** 
 - Min ist am Zug: **Min-Knoten** 
 - Min/Max alternierend in jedem Halbzug

Voraussetzungen für den Algorithmus

Minmax

- zwei Typen von Knoten
 - Max ist am Zug: **Max-Knoten** 
 - Min ist am Zug: **Min-Knoten** 
 - Min/Max alternierend in jedem Halbzug
- jedem Knoten im Baum wird ein Minmax-Wert zugeordnet
 - entspricht Max' **Nutzen dieser Konfiguration**

Voraussetzungen für den Algorithmus

Minmax

- zwei Typen von Knoten
 - Max ist am Zug: **Max-Knoten** \triangle
 - Min ist am Zug: **Min-Knoten** ∇
 - Min/Max alternierend in jedem Halbzug
- jedem Knoten im Baum wird ein Minmax-Wert zugeordnet
 - entspricht Max' **Nutzen dieser Konfiguration**
- Minmax-Wert des Knotens k ergibt sich aus Minmax-Werten der Kinder $N(k)$
 - **Tiefensuche** (*depth first*)

Voraussetzungen für den Algorithmus

Minmax

- zwei Typen von Knoten
 - Max ist am Zug: **Max-Knoten** \triangle
 - Min ist am Zug: **Min-Knoten** ∇
 - Min/Max alternierend in jedem Halbzug
- jedem Knoten im Baum wird ein Minmax-Wert zugeordnet
 - entspricht Max' **Nutzen dieser Konfiguration**
- Minmax-Wert des Knotens k ergibt sich aus Minmax-Werten der Kinder $N(k)$
 - **Tiefensuche** (*depth first*)
- Gewinnfunktion $g : F \rightarrow \mathbb{N}$, die Endkonfigurationen bewertet
(aus Sicht von Max)

Der Minmax-Algorithmus

Minmax

$$\text{Minmax}(u) = \begin{cases} g(k) & \text{wenn } k \in F \\ \max_{v \in N(u)} \text{Minmax}(v) & \text{wenn Max in } u \text{ am Zug} \\ \min_{v \in N(u)} \text{Minmax}(v) & \text{wenn Min in } u \text{ am Zug} \end{cases}$$

Der Minmax-Algorithmus

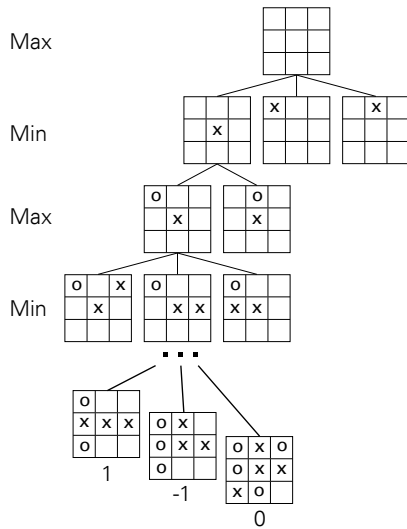
Minmax

$$\text{Minmax}(u) = \begin{cases} g(k) & \text{wenn } k \in F \\ \max_{v \in N(u)} \text{Minmax}(v) & \text{wenn Max in } u \text{ am Zug} \\ \min_{v \in N(u)} \text{Minmax}(v) & \text{wenn Min in } u \text{ am Zug} \end{cases}$$

- Max-Knoten: Max wählt besten Zug
→ **höchsten** Wert annehmen
- Min-Knoten: bester Zug für Min ist schlechtester für Max
→ **kleinsten** Wert für Max annehmen

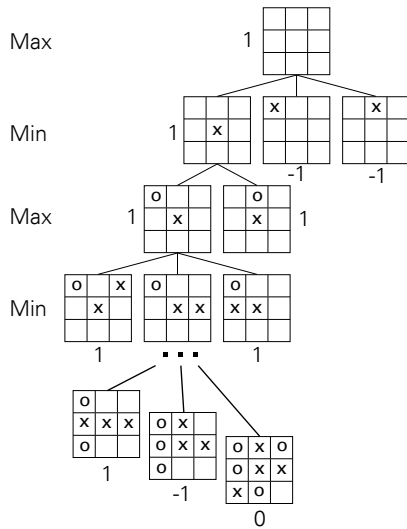
Minimax bei Tic-Tac-Toe

Minimax



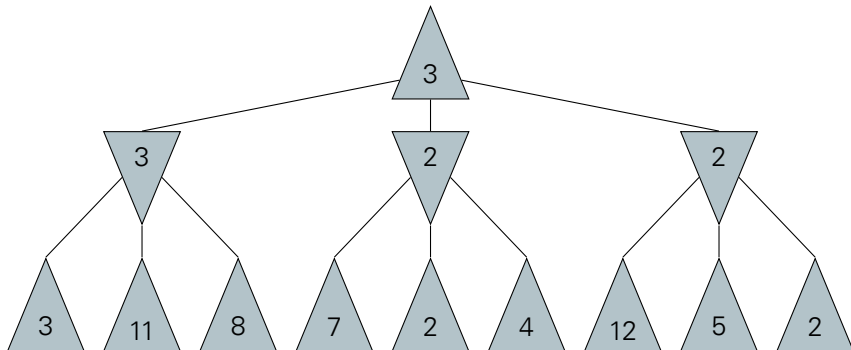
Minimax bei Tic-Tac-Toe

Minimax



Abstrakter Minmax-Bäume

Minmax



Komplexität

Minmax

— m **maximale Tiefe** des Spielbaumes

— b **Verzweigungsgrad**

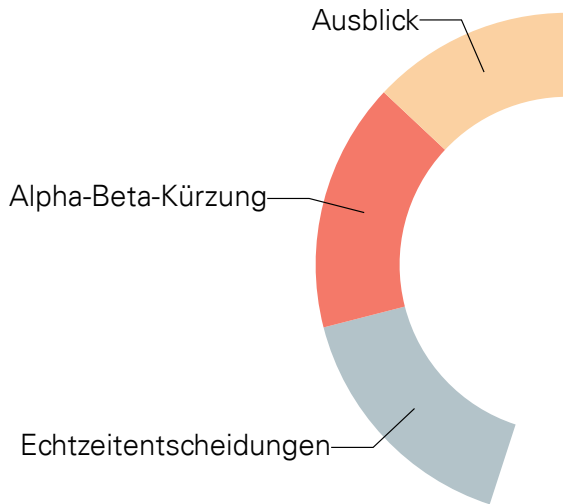
Komplexität

Minmax

- m **maximale Tiefe** des Spielbaumes
- b **Verzweigungsgrad**
- Zeitkomplexität: $\mathcal{O}(b^m)$
- Speicherkomplexität: $\mathcal{O}(bm)$, bzw. $\mathcal{O}(m)$

Unvollständige Echtzeitentscheidungen

Heuristik, Expansionskriterium



Verbesserung Minmax

Unvollständige Echtzeitentscheidung

— neu: **Kriterium**, wann Knoten expandiert werden soll

Verbesserung Minmax

Unvollständige Echtzeentscheidung

- neu: **Kriterium**, wann Knoten expandiert werden soll
- **Heuristik**, um nicht-Endkonfiguration zu bewerten

Verbesserung Minmax

Unvollständige Echtzeitentscheidung

- neu: **Kriterium**, wann Knoten expandiert werden soll
- **Heuristik**, um nicht-Endkonfiguration zu bewerten
- Stärke des Agenten hängt ab von:
 - (a) Bewertung des Nutzens einer Knotenexpansion
 - (b) Genauigkeit der Heuristik

Heuristiken

Unvollständige Echtzeitentscheidung

- Heuristik schätzt Nutzen für Spieler in bestimmter Konfiguration
 - vergleichbar mit A^* -Suche

Heuristiken

Unvollständige Echtzeitentscheidung

- __ Heuristik schätzt Nutzen für Spieler in bestimmter Konfiguration
 - __ vergleichbar mit A^* -Suche
- __ aus Merkmalen der Konfiguration zusammengesetzt

Heuristiken

Unvollständige Echtzeitentscheidung

- __ Heuristik schätzt Nutzen für Spieler in bestimmter Konfiguration
 - __ vergleichbar mit A^* -Suche
- __ aus Merkmalen der Konfiguration zusammengesetzt
- __ Anforderungen:
 - __ muss gleiche Ordnung wie Gewinnfunktion erzeugen

Heuristiken

Unvollständige Echtzeitentscheidung

- Heuristik schätzt Nutzen für Spieler in bestimmter Konfiguration
 - vergleichbar mit A^* -Suche
- aus Merkmalen der Konfiguration zusammengesetzt
- Anforderungen:
 - muss gleiche Ordnung wie Gewinnfunktion erzeugen
 - muss effizient sein

Heuristiken

Unvollständige Echtzeitentscheidung

- Heuristik schätzt Nutzen für Spieler in bestimmter Konfiguration
 - vergleichbar mit A^* -Suche
- aus Merkmalen der Konfiguration zusammengesetzt
- Anforderungen:
 - muss gleiche Ordnung wie Gewinnfunktion erzeugen
 - muss effizient sein
 - muss für nicht-Endkonfigurationen Wert nahe der *tatsächlichen* Chance liefern

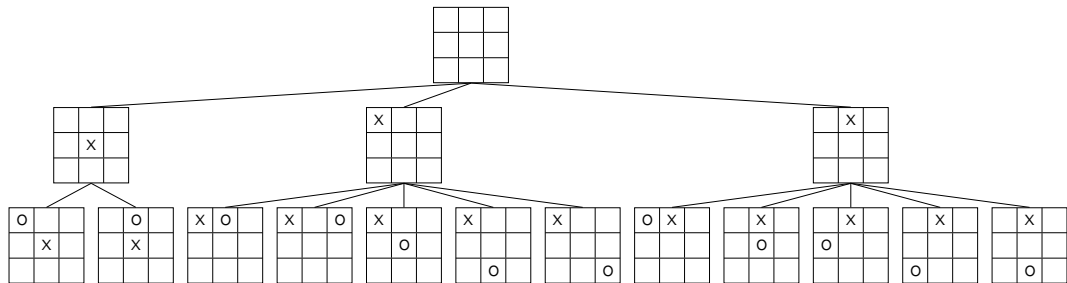
Eine Heuristik für TicTacToe

Unvollständige Echtzeitentscheidung

- $f(k) = A_X(k) - A_O(k)$
 - $A_{X/O}(k)$ „Wie viele Möglichkeiten zur Vervollständigung von Linien gibt es?“

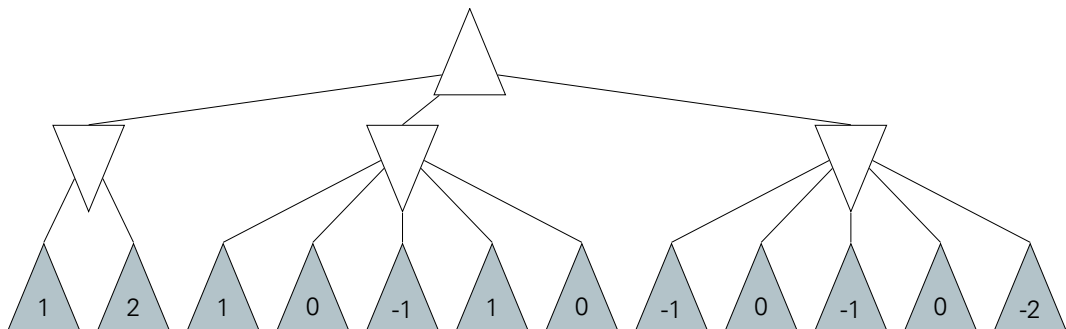
Noch einmal Tic-Tac-Toe

Terminologie



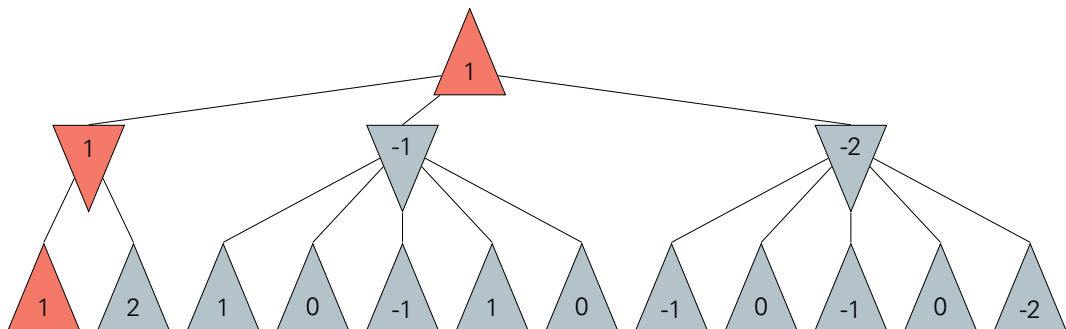
Noch einmal Tic-Tac-Toe

Terminologie



Noch einmal Tic-Tac-Toe

Terminologie



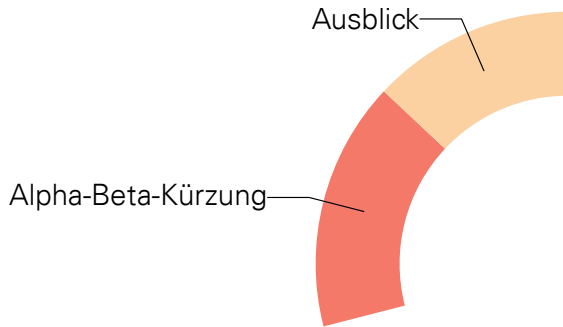
Expansionskriterium

Unvollständige Echtzeitentscheidung

- trivial: immer terminieren, wenn Endkonfiguration erreicht ist
- Metaschluss: Wann lohnt es sich, eine Berechnung anzustellen?
- Beispiele
 - feste Tiefe l
 - **Ruhesuche:** Bewertung ändert sich nicht mehr stark

α, β

Geschichte, Algorithmus, Komplexität



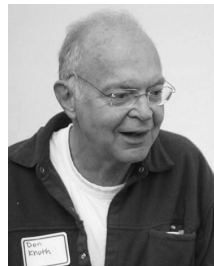
Geschichte der Alpha-Beta-Kürzung (auch α - β -Pruning, -Suche)

α, β

- _ Optimierung von Minmax
- _ 1956 in Dartmouth vorgestellt (McCarthy)
- _ 1961 erstmals in Schachprogramm eingesetzt
- _ 1975 verfeinert (Knuth, Moore)



_ John McCarthy, 1967 ⁵



_ Donald E. Knuth, 2005 ⁶

Erweiterung von Minmax

α, β

— einen Alpha- und einen Beta-Wert für jeden Knoten

Erweiterung von Minmax

α, β

- einen Alpha- und einen Beta-Wert für jeden Knoten
- **untere Schranke** α für Maxknoten
 - bisher größte Bewertung im Pfad zu k

Erweiterung von Minmax

α, β

- __ einen Alpha- und einen Beta-Wert für jeden Knoten
- __ **untere Schranke** α für Maxknoten
 - __ bisher größte Bewertung im Pfad zu k
- __ **obere Schranke** β für Minknoten
 - __ bisher kleinste Bewertung im Pfad zu k

Erweiterung von Minmax

α, β

- einen Alpha- und einen Beta-Wert für jeden Knoten
- **untere Schranke** α für Maxknoten
 - bisher größte Bewertung im Pfad zu k
- **obere Schranke** β für Minknoten
 - bisher kleinste Bewertung im Pfad zu k
- α und β werden nach jeder untersuchten Konfiguration aktualisiert
 - initial $\alpha = -\infty, \beta = +\infty$
 - α wird an einem Max-Knoten maximiert
 - β wird an einem Min-Knoten minimiert

Kürzung

α, β

- **Kürzung** der Kindknoten,
d.h. die Rekursion *vor* Erreichen der Endkonfiguration terminiert

Kürzung

α, β

- **Kürzung** der Kindknoten,
d.h. die Rekursion *vor* Erreichen der Endkonfiguration terminiert
- an Max-Knoten **α -Cut**
 - es wird der größte Wert unter den Kindern gesucht
 - sobald β eines Kindes $\leq \alpha$

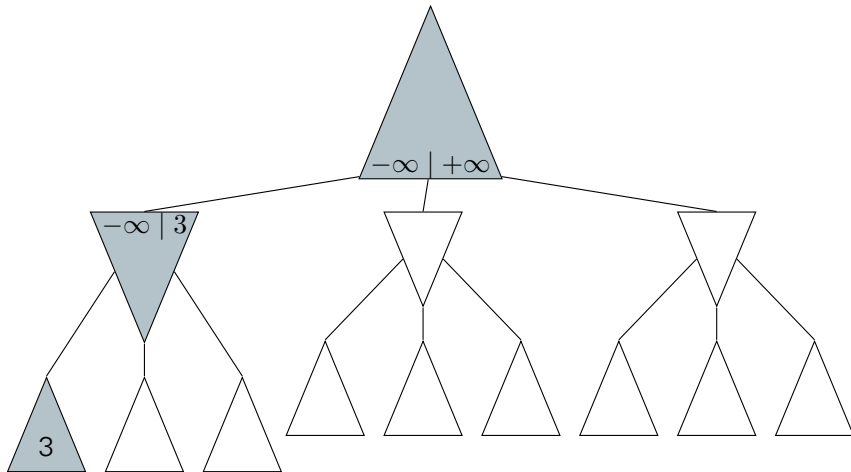
Kürzung

α, β

- **Kürzung** der Kindknoten,
d.h. die Rekursion *vor* Erreichen der Endkonfiguration terminiert
- an Max-Knoten **α -Cut**
 - es wird der größte Wert unter den Kindern gesucht
 - sobald β eines Kindes $\leq \alpha$
- an Min-Knoten **β -Cut**
 - es wird der kleinste Wert unter den Kindern gesucht
 - sobald α eines Kindes $\geq \beta$

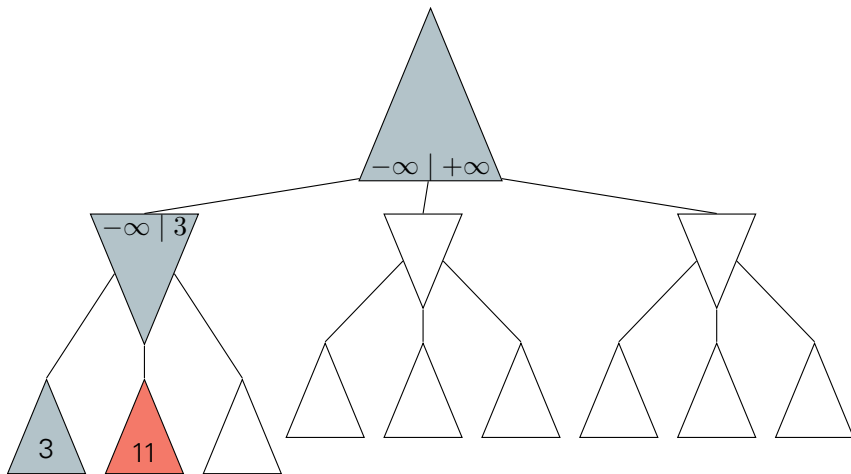
Ein α, β Beispiel

α, β



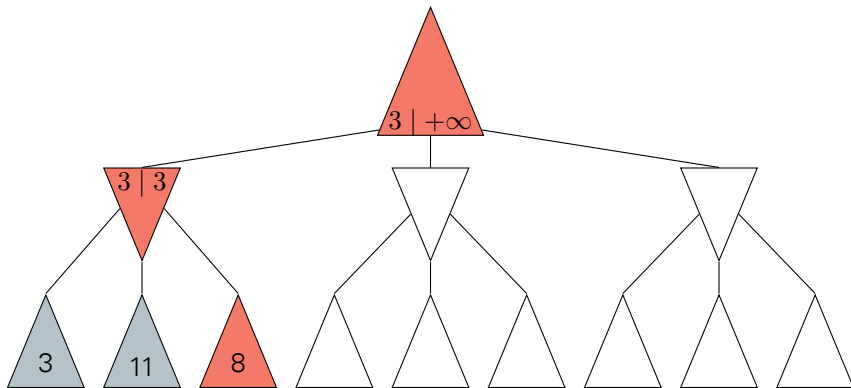
Ein α, β Beispiel

α, β



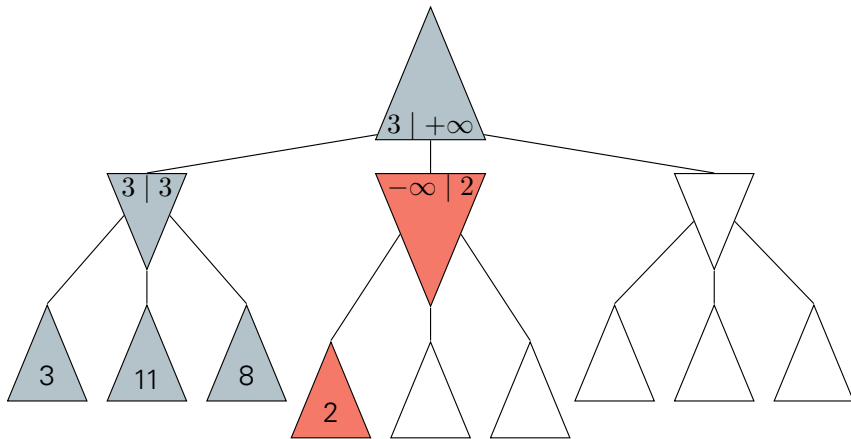
Ein α, β Beispiel

α, β



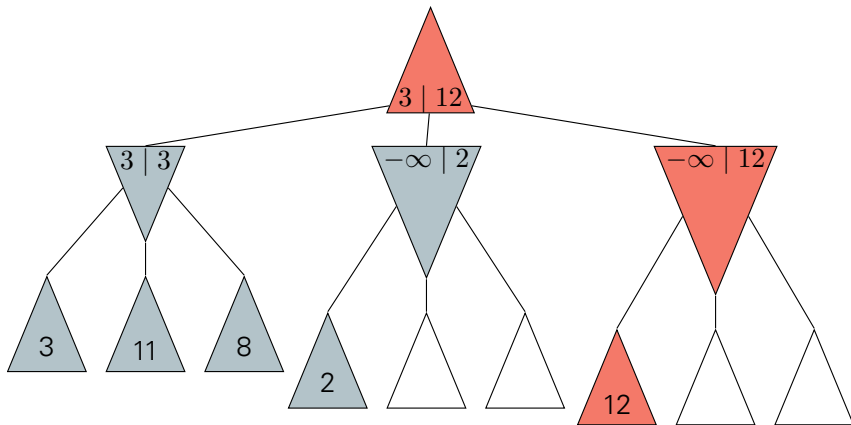
Ein α, β Beispiel

α, β



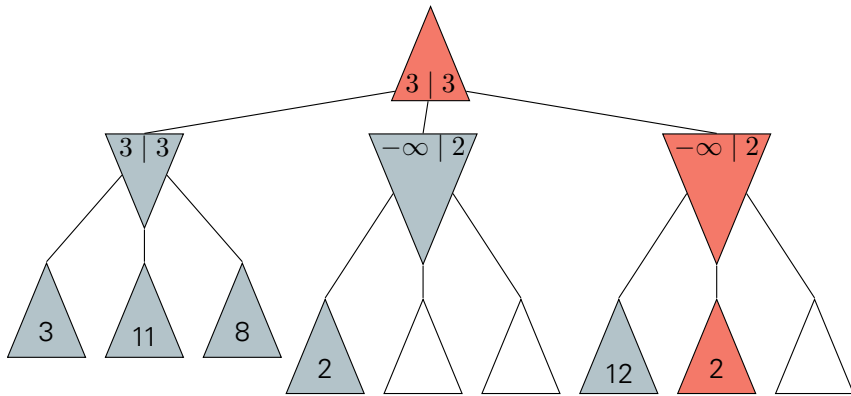
Ein α, β Beispiel

α, β



Ein α, β Beispiel

α, β



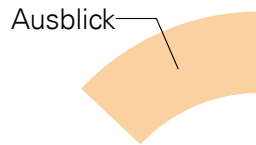
Komplexität

α, β

- Minmax: $\mathcal{O}(b^m)$
- Alpha-Beta-Kürzung: $\mathcal{O}(b^{\frac{m}{2}})$
 - bei optimaler Sortierung (**Zugreihenfolge**)
 - theoretisches Limit, wenn immer sofort gekürzt werden kann
 - bei zufälliger Sortierung der Knoten: $\mathcal{O}(b^{\frac{3}{4}m})$

Ausblick

Zugreihenfolge



Zugreihenfolge

Ausblick

- _ vorsortieren und Killerzüge finden ("Killerzugheuristik")
- _ dynamisch: zuerst Züge probieren, die vorher gut waren
 - _ Situationen können wieder auftreten → Hashtabelle anlegen

Verbesserung mit iterativer Tiefensuche

Ausblick

- Suchtiefe mit jedem Durchlauf erhöhen

Verbesserung mit iterativer Tiefensuche

Ausblick

- Suchtiefe mit jedem Durchlauf erhöhen
- hilft bei Wahl einer günstigen Zugreihenfolge
- nutzt begrenzte Zeit best möglich aus

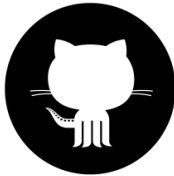
Rückblick

zur Motivation

- formale Betrachtung von **Spiele**n verstehen
- **Minmax-Algorithmus** nachvollziehen können
- gute und schlechte **Heuristiken** unterscheiden und anwenden können
- Vorteile der **Alpha-Beta-Kürzung** verstehen

joschka.heinrich@tu-dresden.de

B40E 67C7 FF62 C860 7854 A778 6FB9 666F 1147 A401



<https://github.com/foobar0112/tic>

Sachquellen

Anhang

- _ Klüppelholz: „Entwurfs- und Analysemethoden für Algorithmen – Skript zur Vorlesung SS 2016“
- _ Russel, Norvig: „Künstliche Intelligenz – Ein moderner Ansatz“, 3., aktualisierte Auflage, 2012
- _ <http://chalkdustmagazine.com/features/menace-machine-educable-noughts-crosses-engine>
- _ <https://de.wikipedia.org/wiki/Tic-Tac-Toe>
- _ <https://en.wikipedia.org/wiki/Tic-tac-toe>
- _ <https://de.wikipedia.org/wiki/Minimax-Algorithmus>
- _ <https://en.wikipedia.org/wiki/Minimax>
- _ <https://de.wikipedia.org/wiki/Alpha-Beta-Suche>
- _ https://en.wikipedia.org/wiki/Alpha-beta_pruning

Bildquellen

Anhang

- [1] Donald Michies <http://www.aiai.ed.ac.uk/~dm/donald-michie-2003.jpg>
- [2] Menace http://images.slideplayer.nl/9/2257151/slides/slide_6.jpg
- [3] Claude Shannon https://upload.wikimedia.org/wikipedia/commons/9/99/ClaudeShannon_MF03807.jpg
- [4] Alan Turing https://upload.wikimedia.org/wikipedia/commons/a/a1/Alan_Turing_Aged_16.jpg
- [5] John McCarthy http://images.computerhistory.org/fellows/2-4a.stanford_university.mccarthy-john.c1967.1062302006.stanford_university.src.jpg
- [6] Donald E. Knuth <https://upload.wikimedia.org/wikipedia/commons/4/4f/KnuthAtOpenContentAlliance.jpg>

Suchbaum \neq Spielbaum

Anhang

- **Suchbaum** ist gerichteter Graph (V, E, l, w) , Baum
 - Tiefe $l \in \mathbb{N}_{>0}$
 - Knoten entsprechen Pfaden der maximalen Länge l , ausgehend von w im *Spielbaum*
 - Kanten entsprechen anwendbarem Zug

Min spielt nicht optimal

Anhang

- __ Minmax ist nur beste Strategie, wenn Min-Spieler auch optimal spielt
- __ wenn nicht: dann noch besser für Max
- __ aber: es gibt ggf. noch bessere Strategien

Spiel mit mehr als 2 Spielern

Anhang

- __ nicht Min-/Max-Halbzüge, sondern A,B,C,...-Züge
- __ nicht ein Minmax-Wert sondern Vektor mit Max-Werte jedes Spielers
- __ Warum kein Vektor bei 2 Spielern?
 - __ → Nullsummenspiel: Gewinn des einen ist Verlust des anderen
- __ Problem: Spielstrategie mit Allianzen

Nicht-Nullsummenspiel

Anhang

__ jeder Spieler hat eigene Nutzenfunktion (die allgemein bekannt ist)

Spiele mit Zufall

Anhang

- dritter Typ Knoten: Zufallsknoten
- Erwartungswert statt Minmax-Wert pro Knoten