Hochschule für Angewandte
Wissenschaften Hamburg
Hamburg University of Applied Sciences

Intelligente Systeme

– Spielen –

Prof. Dr. Michael Neitzke

Sp1: Spielen

- Zwei-Personen-Nullsummenspiele
- Lösung durch Suche
- Minimax-Verfahren
- Alpa-Beta Pruning

Sp1: Lernziele

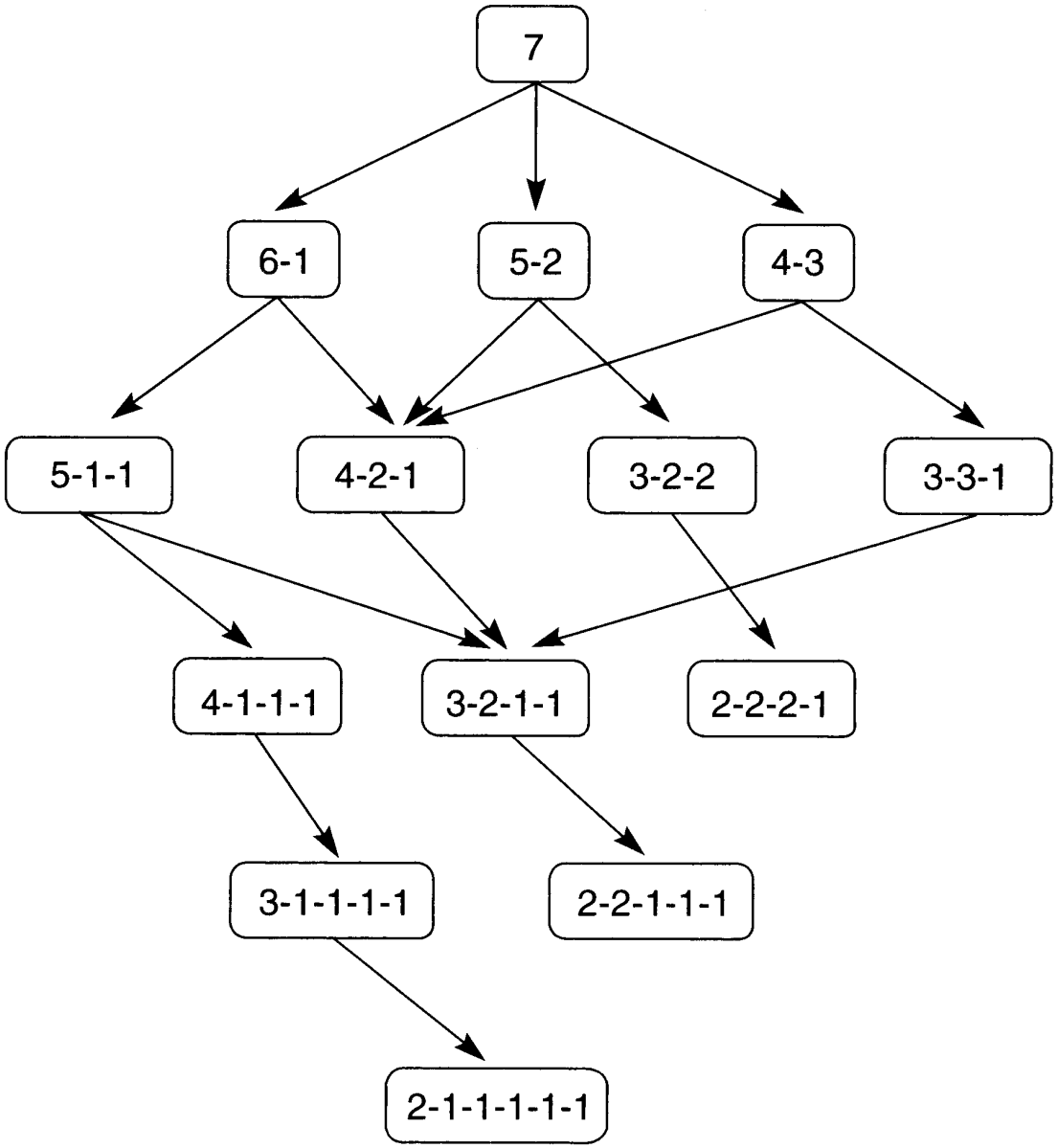
- V1: Bezug zwischen Suche und Zwei-Personen Nullsummen-Spielen erläutern können
- V2: Einsatz von Heuristiken bei Spielen erläutern können
- V3: Minimax-Verfahren erläutern können
- V4: Alpha-Beta Pruning erläutern können

- A1: Minimax-Verfahren anwenden können (--> Teil 2 der Klausur)
- A2: Alpha-Beta Pruning anwenden können (--> Teil 2 der Klausur)

Was ist ein Zwei-Personen-Nullsummenspiel?

- Zwei-Personen: klar
- Nullsummenspiel: Summe der Gewinne und Verluste beider Gegner ist Null
  - Gegenbeispiel: Fußball-Bundesliga

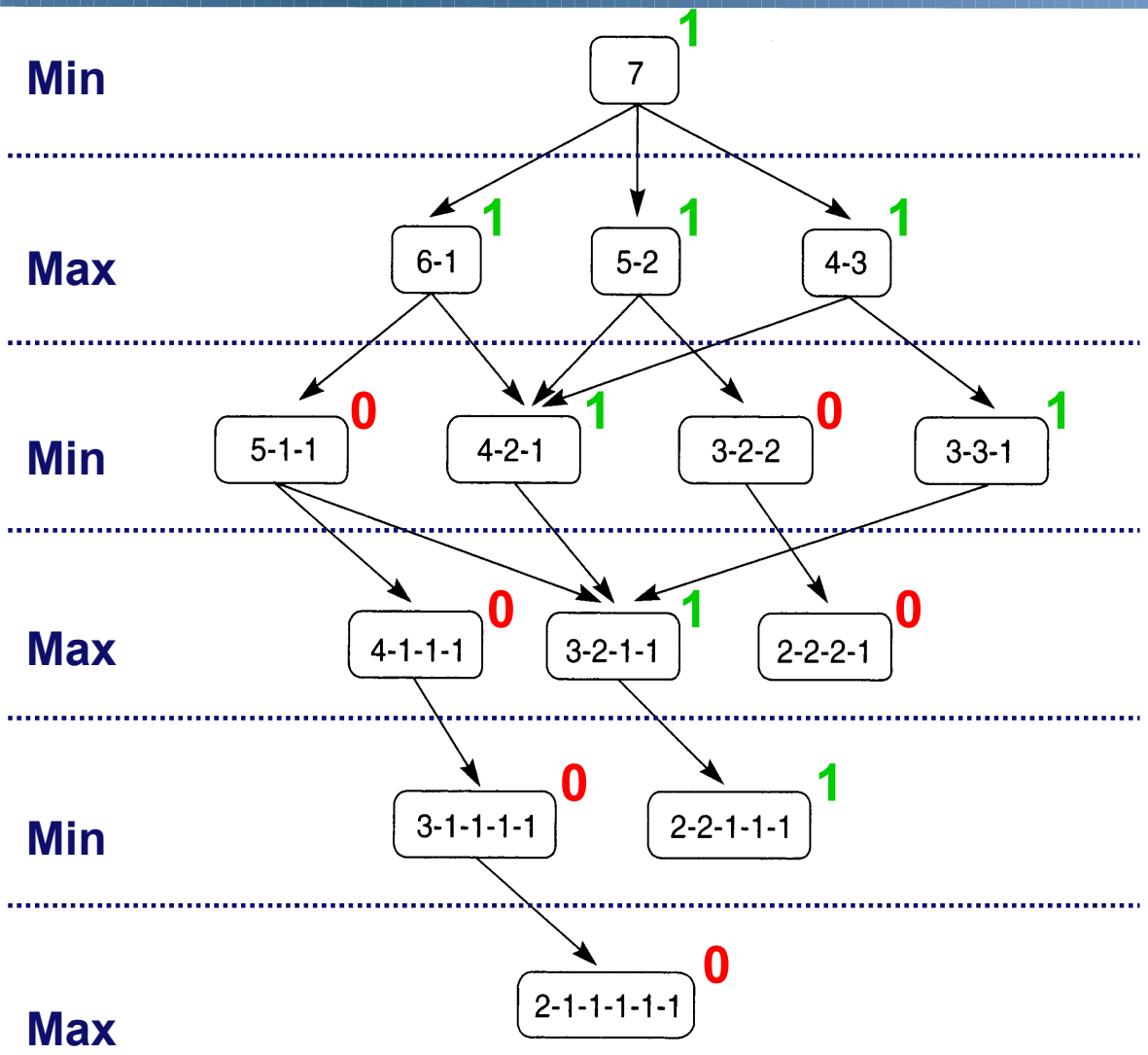
Zustandsraum Nim



Erläuterungen

- Im Spiel „Nim“ wird ein Häufchen mit Streichhölzern abwechselnd in kleinere Häufchen aufgeteilt. Am Anfang gibt es nur ein Häufchen, hier im Beispiel besteht es aus sieben Streichhölzern. Der Spieler, der anfängt, muss es in zwei unterschiedlich große Häufchen aufteilen, was im Beispiel aufgrund der ungeraden Anzahl von Streichhölzern ja auch gar nicht anders möglich ist. Danach ist der Gegner dran, usw.. Wenn es für einen Spieler keine weitere Möglichkeit des Aufteilens mehr gibt, so hat er verloren.

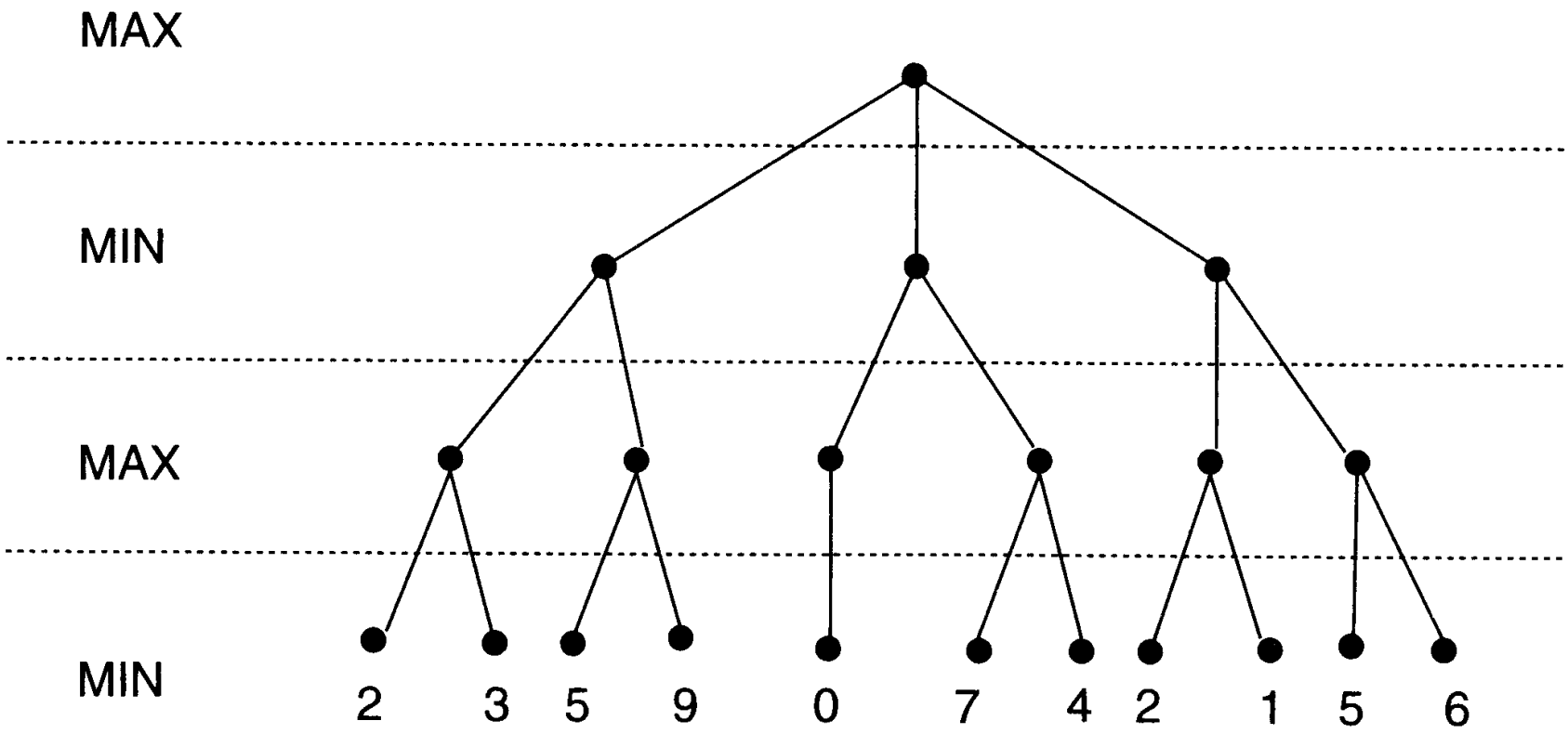
Minimax-Verfahren



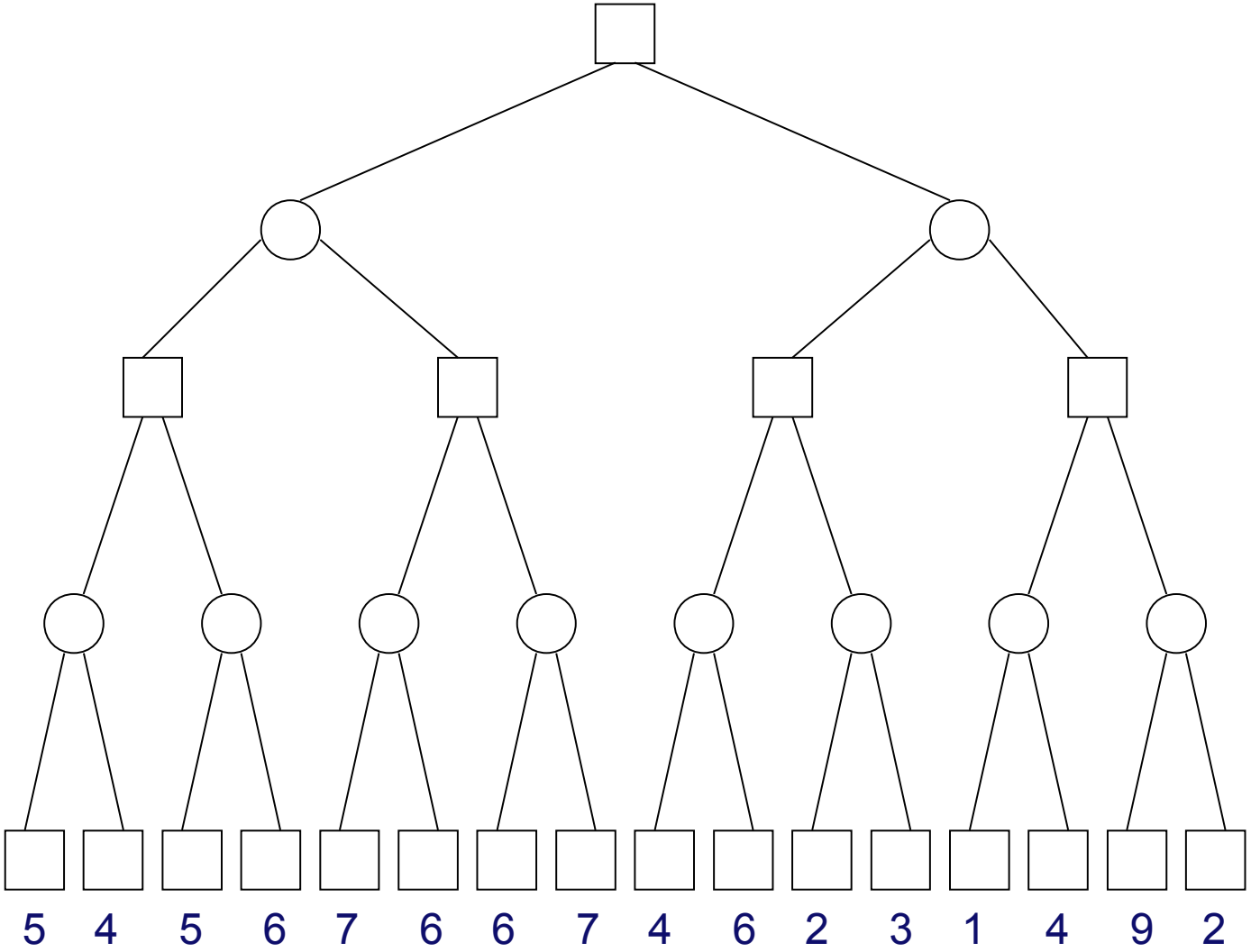
Erläuterungen

- Typischerweise ist Max die Sicht des Spielecomputers
- In diesem Beispiel beginnt also der Gegner
  - (Damit er sich nicht beschweren kann, falls er verlieren sollte. :-))

Minimax mit Vorausschau von drei Zügen



Übung: Minimax



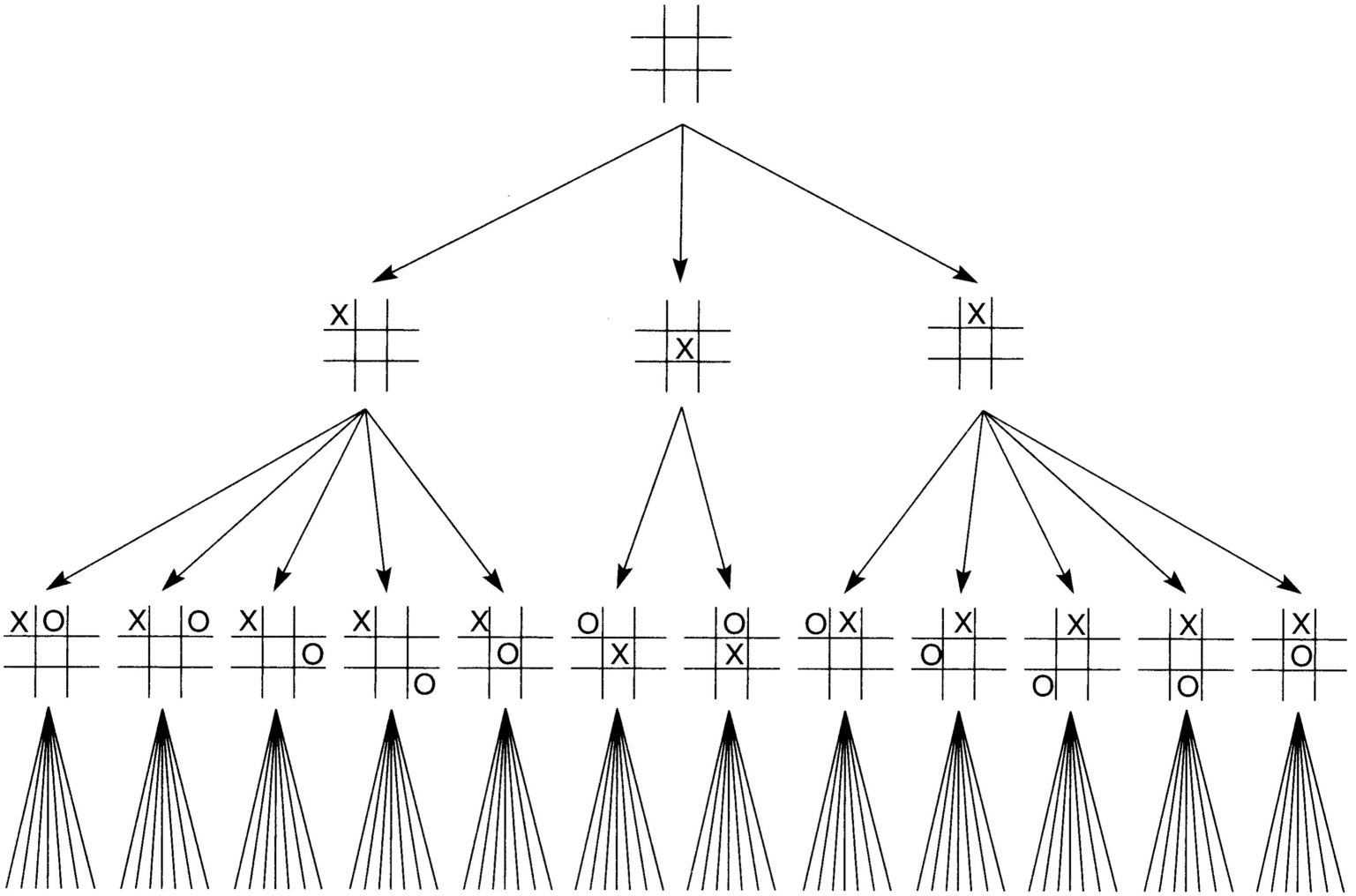
Diskutieren Sie!

- Wie könnte man zu einer Bewertung von Zuständen kommen?

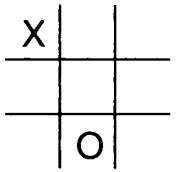
Lösung

- Mit Heuristiken
- Durch Einschätzung von Experten
- Durch statistische Auswertung von Spielen

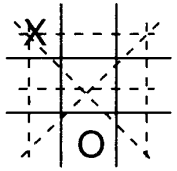
Beispiel: Tic-Tac-Toe



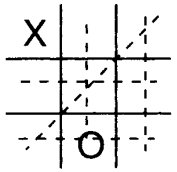
Heuristik



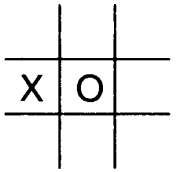
X hat 6 mögliche Gewinnwege:



O hat 5 Gewinn-möglichkeiten:

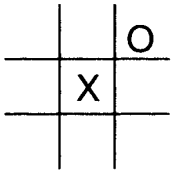


E(n) = 6 – 5 = 1



X hat 4 mögliche Gewinnwege;
O hat 6 Gewinnmöglichkeiten

E(n) = 4 – 6 = –2



X hat 5 mögliche Gewinnwege;
O hat 4 Gewinnmöglichkeiten

E(n) = 5 – 4 = 1

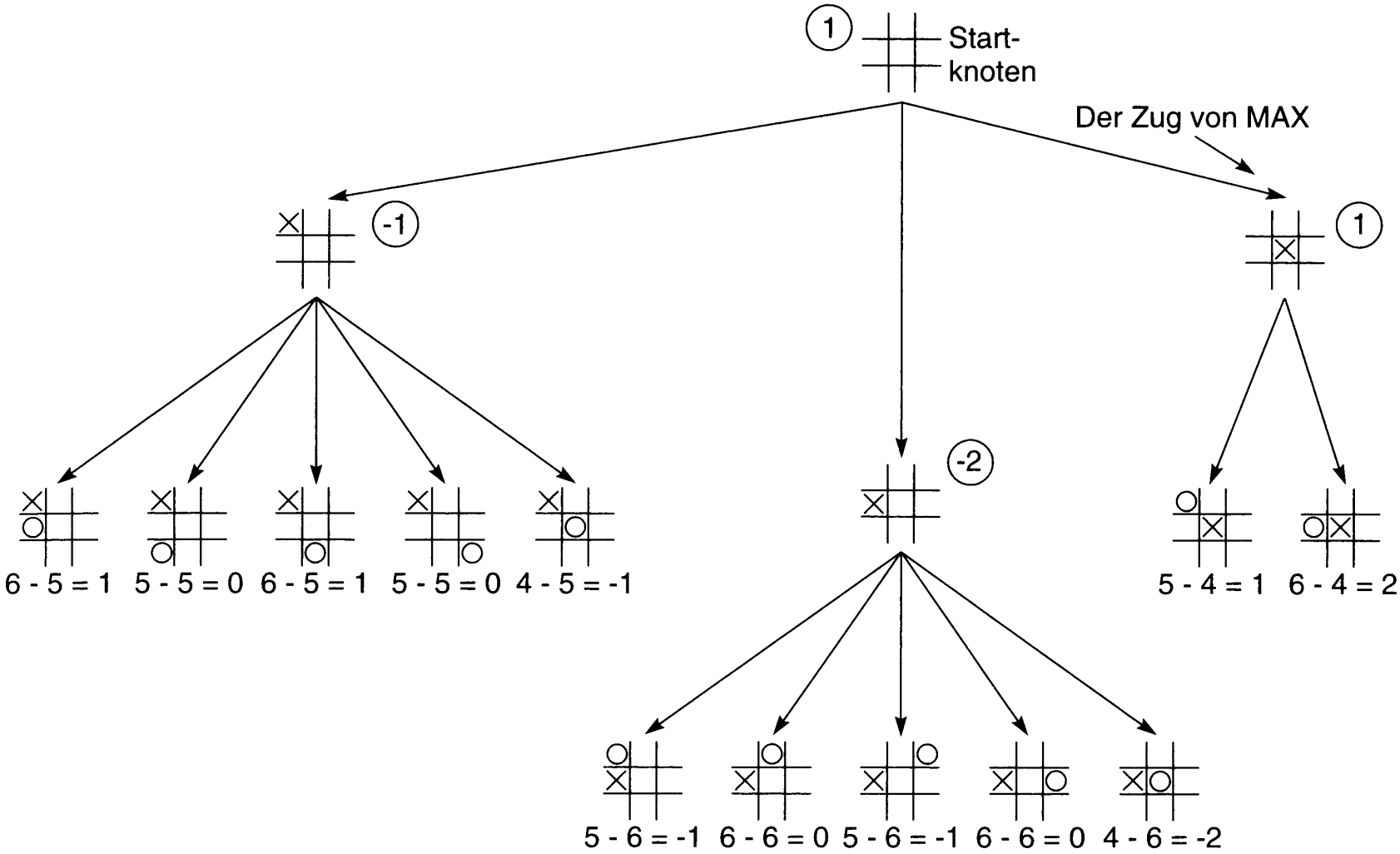
Die Heuristik lautet E(n) = M(n) – O(n)

wobei M(n) die Summe der Gewinnmöglichkeiten ist,

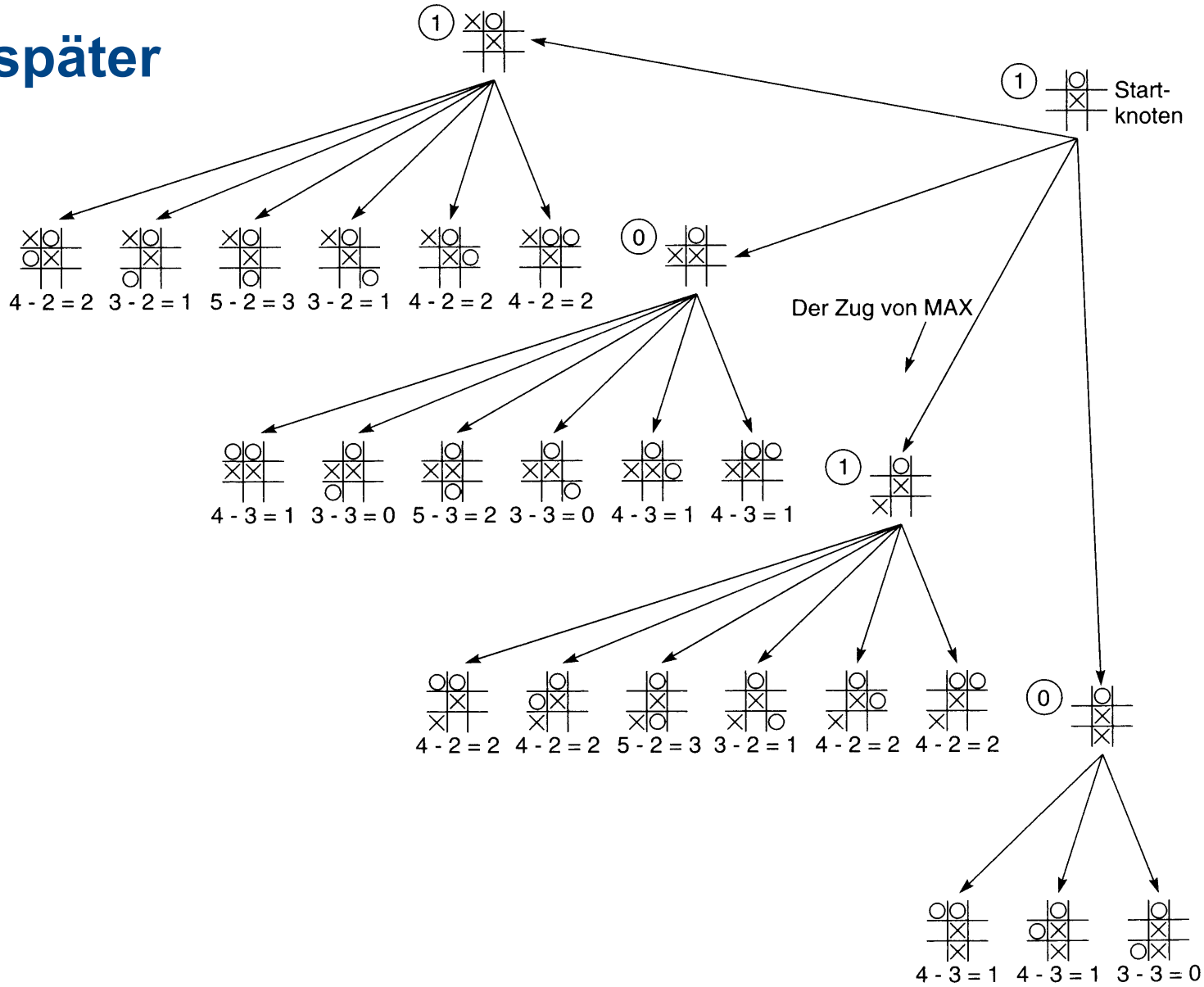
O(n) die Summe der Gewinnmöglichkeiten des Gegners ist,

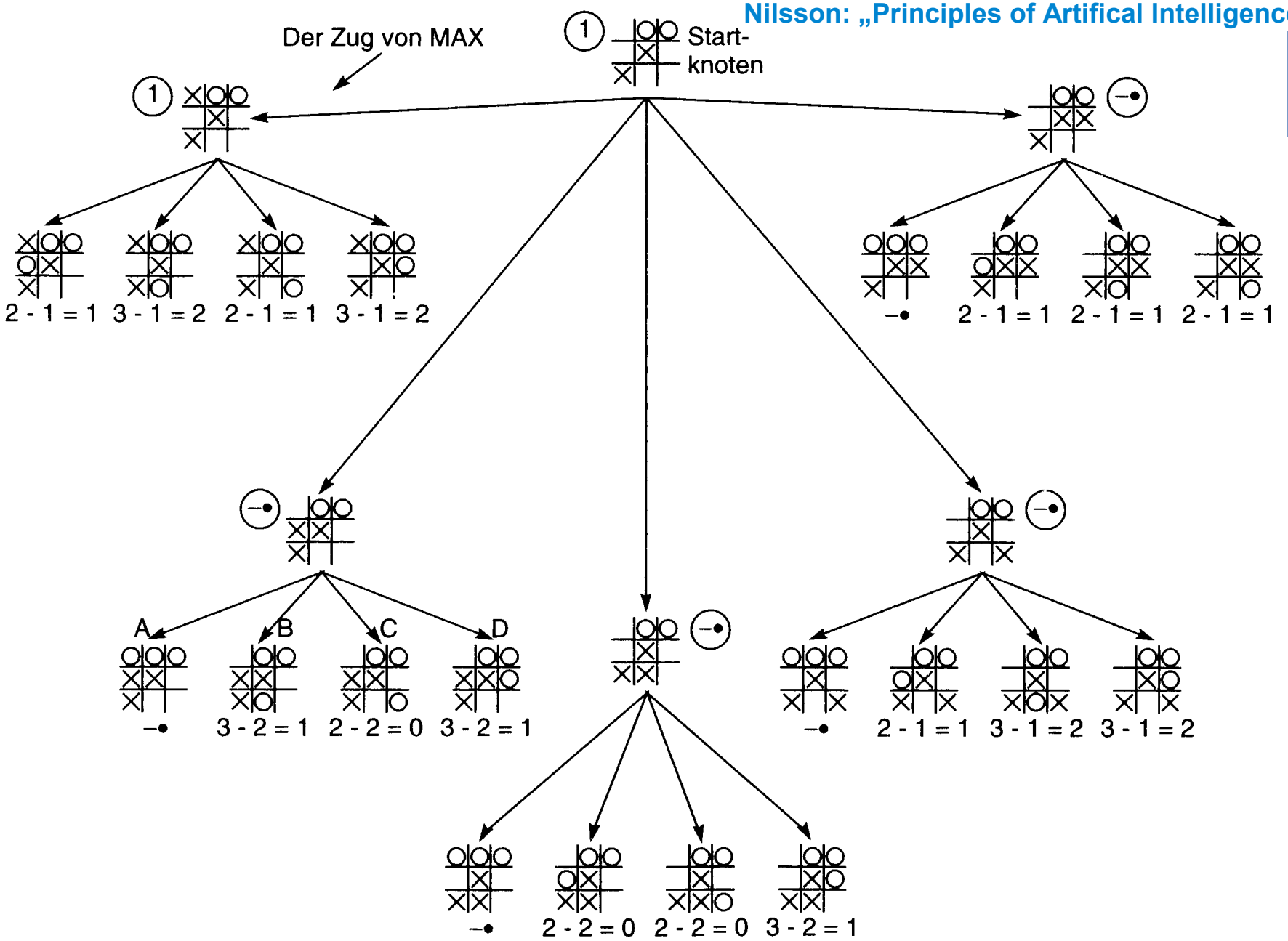
E(n) die Gesamtbewertung des Zustands n ist.

Tic-Tac-Toe mit Vorausschau von zwei Zügen



2 Züge später

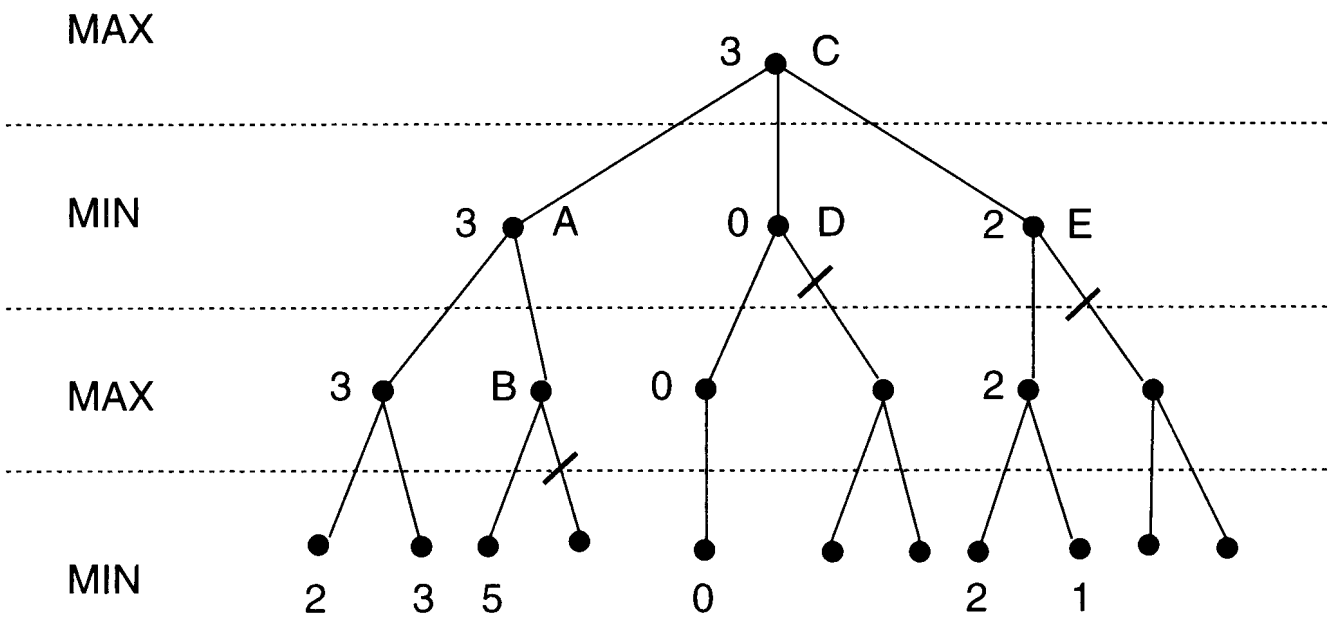



Der Zug von MAX
1
Start-
knoten
2 - 1 = 1
3 - 1 = 2
2 - 1 = 1
3 - 1 = 2
2 - 1 = 1
2 - 1 = 1
2 - 1 = 1
A
B
C
D
3 - 2 = 1
2 - 2 = 0
3 - 2 = 1
2 - 1 = 1
3 - 1 = 2
3 - 1 = 2
2 - 2 = 0
2 - 2 = 0
3 - 2 = 1

Wie kann man Rechenzeit sparen?

Nehmen wir an es gibt einige Taschen voll mit Gegenständen mit mehr oder weniger Wert und Sie erhalten einen Gegenstand aus einer der Taschen. Sie dürfen die Tasche wählen, und der Gegner wählt einen Gegenstand daraus. Sobald sie eine Tasche gewählt haben, wird ihr Gegner natürlich den Gegenstand mit dem geringsten Wert daraus wählen. Sie wählen daher die Tasche, deren geringstwertiger Gegenstand immer noch mehr Wert besitzt, als der jeweils geringstwertige der anderen Taschen.
Wendet man den Minimax-Algorithmus an, durchsucht man einfach alle Taschen vollständig und kennt damit den Wert eines jeden Gegenstandes, und somit kann man genau die richtige Tasche für sich wählen. Aber dieses Durchsuchen dauert eben ziemlich lange.
Also geht man folgendermaßen vor: Man durchsucht die erste Tasche vollständig. Nehmen wir an, darin befinden sich die Schlüssel für einen Neuwagen und ein 20-Euro-Schein. Würden wir diese Tasche wählen, bekämen wir vom Gegner wohl den Geldschein zugeteilt, da dies der Gegenstand mit dem geringsten Wert in dieser Tasche ist.
Wenn wir die zweite Tasche durchsuchen und darin als erstes einen 50-Euro-Schein finden, steigt die Hoffnung, dass noch mehr wertvolle Gegenstände darin enthalten sind, und wir nehmen den zweiten Gegenstand daraus. Angenommen dabei handelt es sich um einen Apfel, dann wird uns der Gegner aus dieser Tasche wohl lieber diesen Apfel als den Geldschein geben. Dieser Apfel ist aber wertloser als der 20-Euro-Schein, den wir aus der ersten Tasche bekämen. Also werden wir nie die zweite Taschen wählen, und durchsuchen sie nicht weiter. Die restlichen Gegenstände darin interessieren nicht mehr, diese können natürlich noch wertloser sein, als der Apfel. Wie wertlos aber diese Tasche genau ist, spielt keine Rolle, ausschlaggebend ist, dass sie wertloser als die erste Tasche ist, nicht, wie viel wertloser sie ist. Auch wenn die restlichen Gegenstände der Tasche großen Wert besitzen, wir bekämen doch nur höchstens diesen Apfel, also legen wir die Tasche beiseite.

Alpha-Beta-Pruning



Diese Beschreibung stammt aus [Lug 01]. Sie stellt eine alternative Vorgehensweise zu der in der Vorlesung behandelten Vorgehensweise dar und wird hier nicht näher erläutert.

A hat $\beta = 3$ (A wird nicht größer als 3 werden)

B ist β-beschnitten, da $5 > 3$

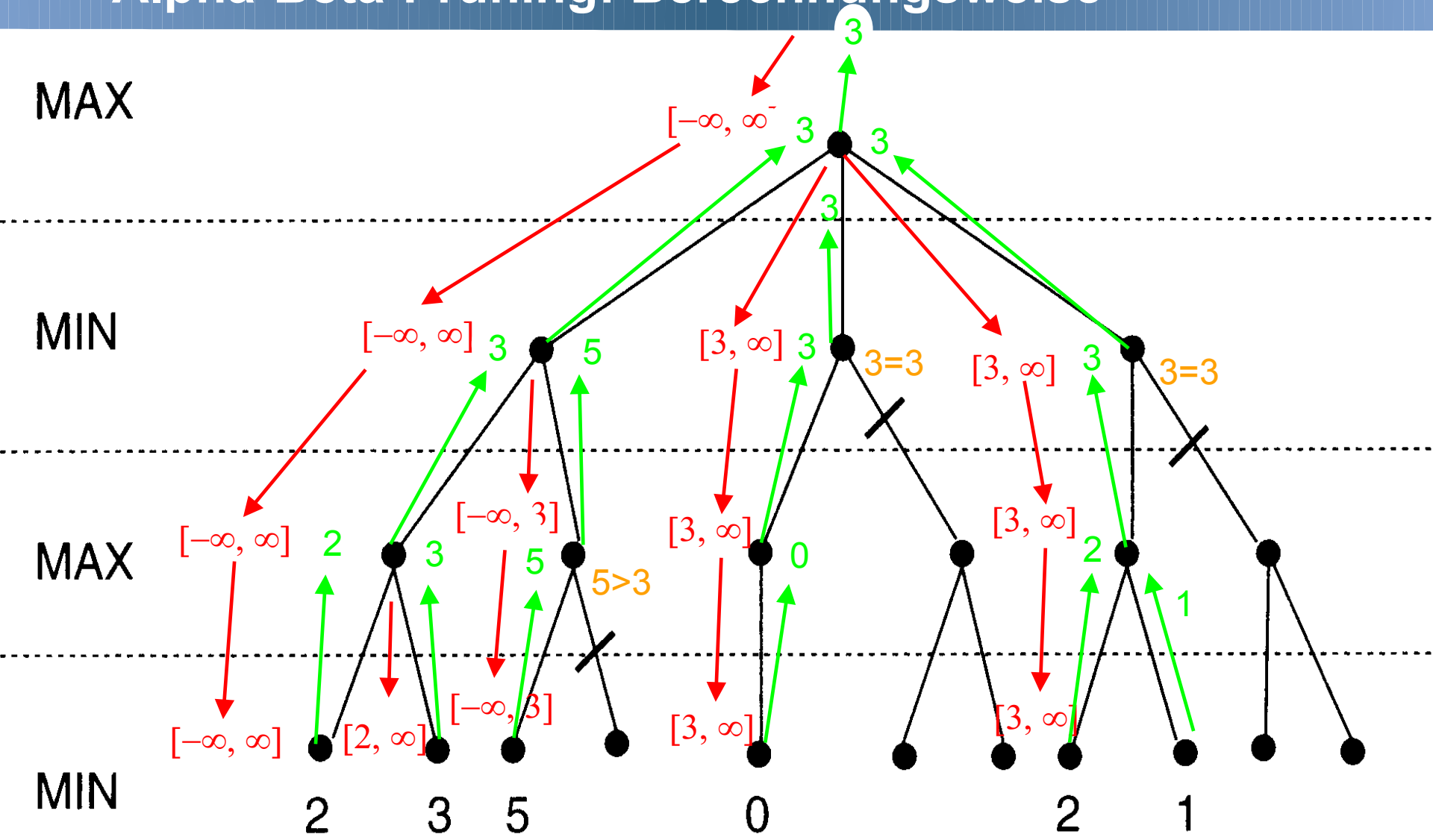
C hat $\alpha = 3$ (C wird nicht kleiner als 3 werden)

D ist α-beschnitten, da $0 < 3$

E ist α-beschnitten, da $2 < 3$

C ist 3

Alpha-Beta-Pruning: Berechnungsweise



Alpha-Beta Algorithmus

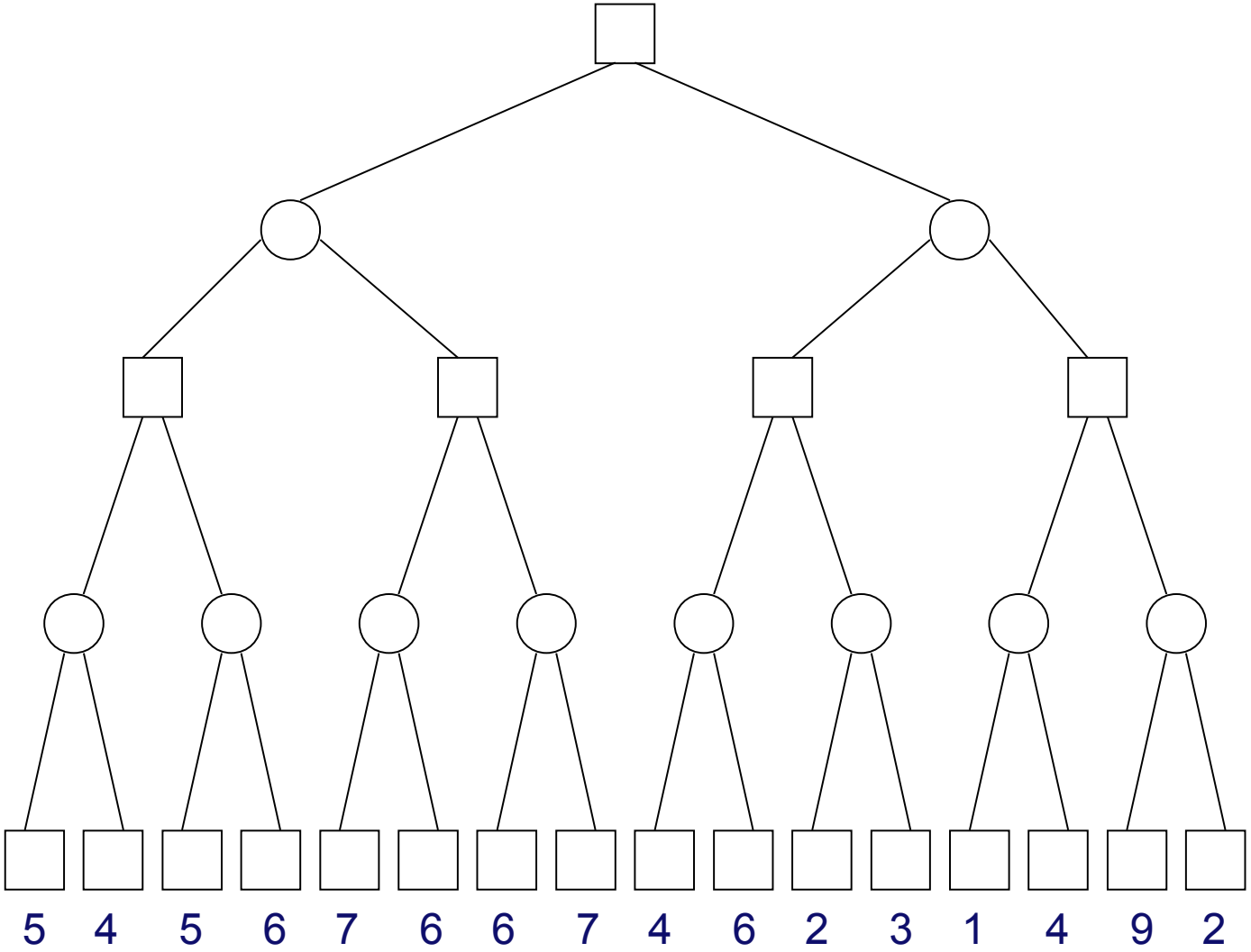
```
function alphabeta_max (node n, int alpha, int beta): int
  if (leafnode(n)) return eval(n);
  forall (m ∈ succ(n)) do
    alpha := max(alpha,alphabeta_min(m,alpha,beta);
    if (alpha >= beta) return alpha;
  end forall;
  return alpha; /*kein vorzeitiger Abbruch*/
end function.

function alphabeta_min (node n, int alpha, int beta): int
  if (leafnode(n)) return eval(n);
  forall (m ∈ succ(n)) do
    beta := min(beta,alphabeta_max(m,alpha,beta);
    if (alpha >= beta) return beta;
  end forall;
  return beta; /*kein vorzeitiger Abbruch*/
end function.
```

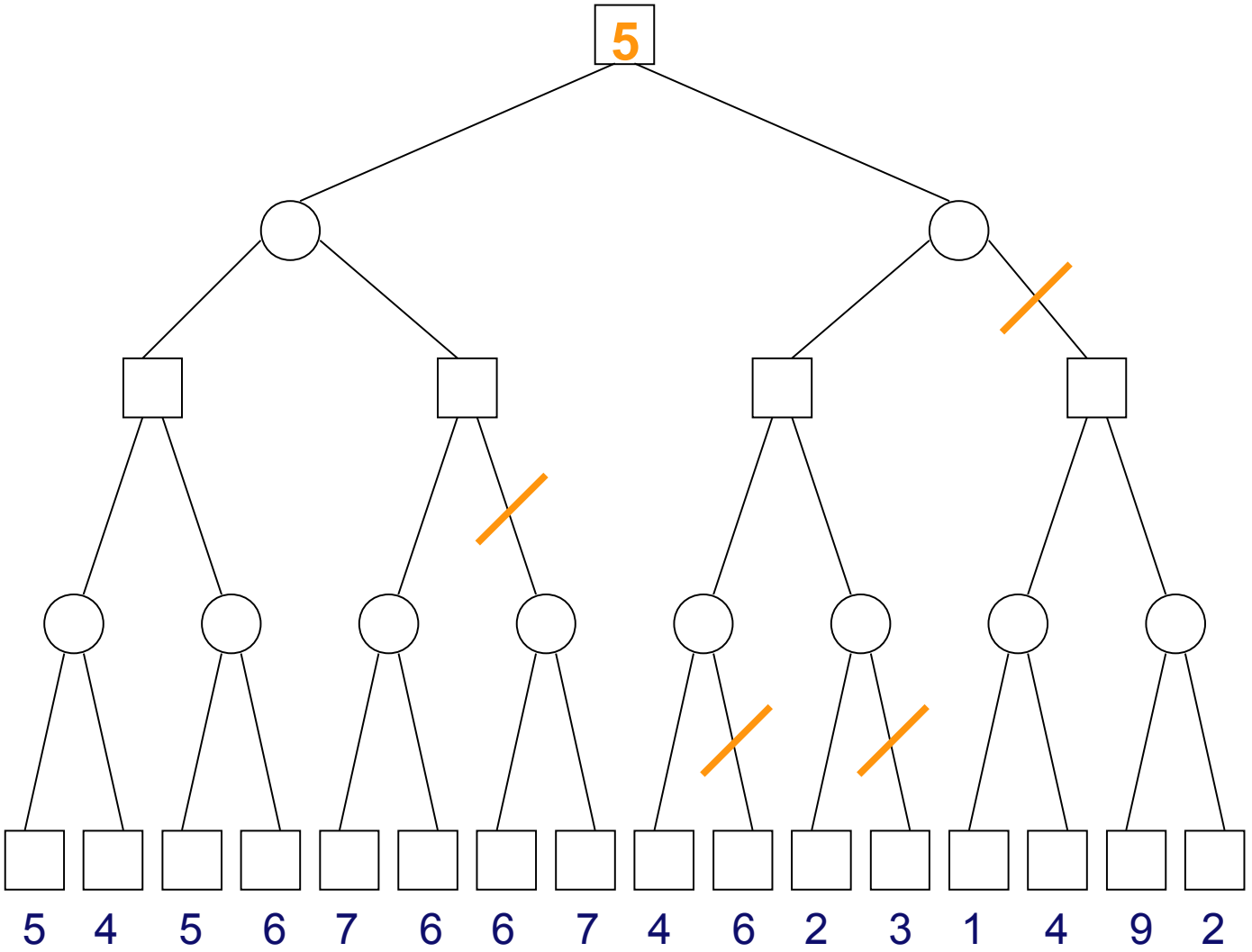
Erläuterungen

- Wenn es um die Berechnung des günstigsten Zuges für Max, geht, so beginnt der Algorithmus mit alphabeta_max und den folgenden Parametern
  - n: Der aktuelle Zustand, für den der günstigste Zug für Max gesucht wird
  - alpha: - ∞
  - beta: ∞
- So lange kein Blattknoten vorliegt, rufen sich alphabeta_max und alphabeta_min immer abwechselnd für den ersten Nachfolge-Knoten auf, und zwar mit dem Intervall [-∞,∞]. Für den ersten Blattknoten, der erreicht wird, nämlich der von unten links, wird dann der Wert zurückgegeben. Wenn der Rückgabewert bei einem Max-Knoten eintrifft, so wird hier der alpha-Wert aktualisiert, sonst der beta-Wert.
- Durch die Aktualisierung der Intervallgrenzen wird das Intervall immer kleiner. Wenn aufgrund der Aktualisierung aber die linke Intervallgrenze größer oder gleich der zweiten werden sollte, so erfolgt ein Abbruch und damit ggf. keine weitere Untersuchung der nächsten Geschwisterzweige. Alphabeta_max gibt immer alpha zurück, alphabeta_min immer beta.

Übung: Alpha-Beta Pruning



Lösung



Sp1: Lernziele

- V1: Bezug zwischen Suche und Zwei-Personen Nullsummen-Spielen erläutern können
- V2: Einsatz von Heuristiken bei Spielen erläutern können
- V3: Minimax-Verfahren erläutern können
- V4: Alpha-Beta Pruning erläutern können

- A1: Minimax-Verfahren anwenden können (--> Teil 2 der Klausur)
- A2: Alpha-Beta Pruning anwenden können (--> Teil 2 der Klausur)