Einführung in die
Künstliche Intelligenz
SS 18
Prof. Dr. J. Fürnkranz, Prof. Dr. K. Kersting

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiellösung für das 3. Übungsblatt

Aufgabe 1 Constraint Satisfaction Problem

Es sollte klar sein, dass folgende Formalisierung eine Möglichkeit und nicht die einzig Richtige darstellt.

a) Wertebereiche der Variablen:

$$\begin{aligned} domain(A) = domain(C) &= \{0,\ldots,9\} \\ domain(B) &= \{1,\ldots,9\} \\ domain(U) &= \{0,1\} \end{aligned}$$

Constraints:

$$\begin{aligned} A+B &= C+10\cdot U \\ B &= U \\ A\neq B,\ A\neq C,\ B\neq C & \end{aligned}$$

b) `C selected`

```
C = 0 consistent
  B selected
    B = 1 consistent
      A selected
        A = 0 not consistent (A != C constraint violated)
        A = 1 not consistent (A != B constraint violated)
        A = 2 consistent
          U selected
            U = 0 not consistent (B = U constraint violated)
            U = 1 not consistent (A+B=C+10*U constraint violated)
          U removed
        A = 3 consistent
        ... (identischer Verlauf wie in A=2, mit den gleichen Constraint Verletzungen)
        A = 9 consistent
          U selected
            U = 0 not consistent
            U = 1 consistent
            result <- C=0,B=1,A=9,U=1
```

c) Forward Checking erweitert Backtracking dahingehend, daß wenn immer eine neue Variable X einen Wert erhält, es überprüft wird, ob es für jeden Wert einer durch einen Constraint mit X verbundene Variable Y noch einen passenden Wert von X gibt. Falls nicht, wird der entsprechende Wert aus Y gelöscht.

In der folgenden Darstellung des Algorithmus-Ablaufs werden nun zusätzlich die durch Forward-Checking veränderten Wertebereiche dargestellt.

```
C selected
  C = 0 consistent
    FC: domain(A) = {1,...,9}
    B selected
```

```
B = 1 consistent
  FC: domain(A) = {2,...,9}, domain(U) = {1}
  A selected
    A = 2 consistent (Beachte: die Zuweisung A = 0,1 kommt hier nicht mehr vor)
      FC: domain(U)={}, backtrack
    A = 3 consistent
    ... (identischer Verlauf wie in A=2)
    A = 9 consistent
      FC: nothing to change
      U selected
        U = 1 consistent  (U=0 kommt nicht mehr vor)
        result <- C=0,B=1,A=9,U=1
```

In diesem Beispiel ist der Suchraum der noch nicht belegten Variablen nach der Festlegung $C = 0$ und $B = 1$:

- $A \in \{2, \ldots, 9\}$, da $A \neq B$ und $A \neq C$
- $U \in \{1\}$, da $U = B$

Es müssen also die Varianten mit $A = 0$ und $A = 1$ sowie die Varianten mit $U = 9$ für alle Werte von A nicht mehr getestet werden, die Backtracking-Suche mit Forward Checking konvergiert also schneller.

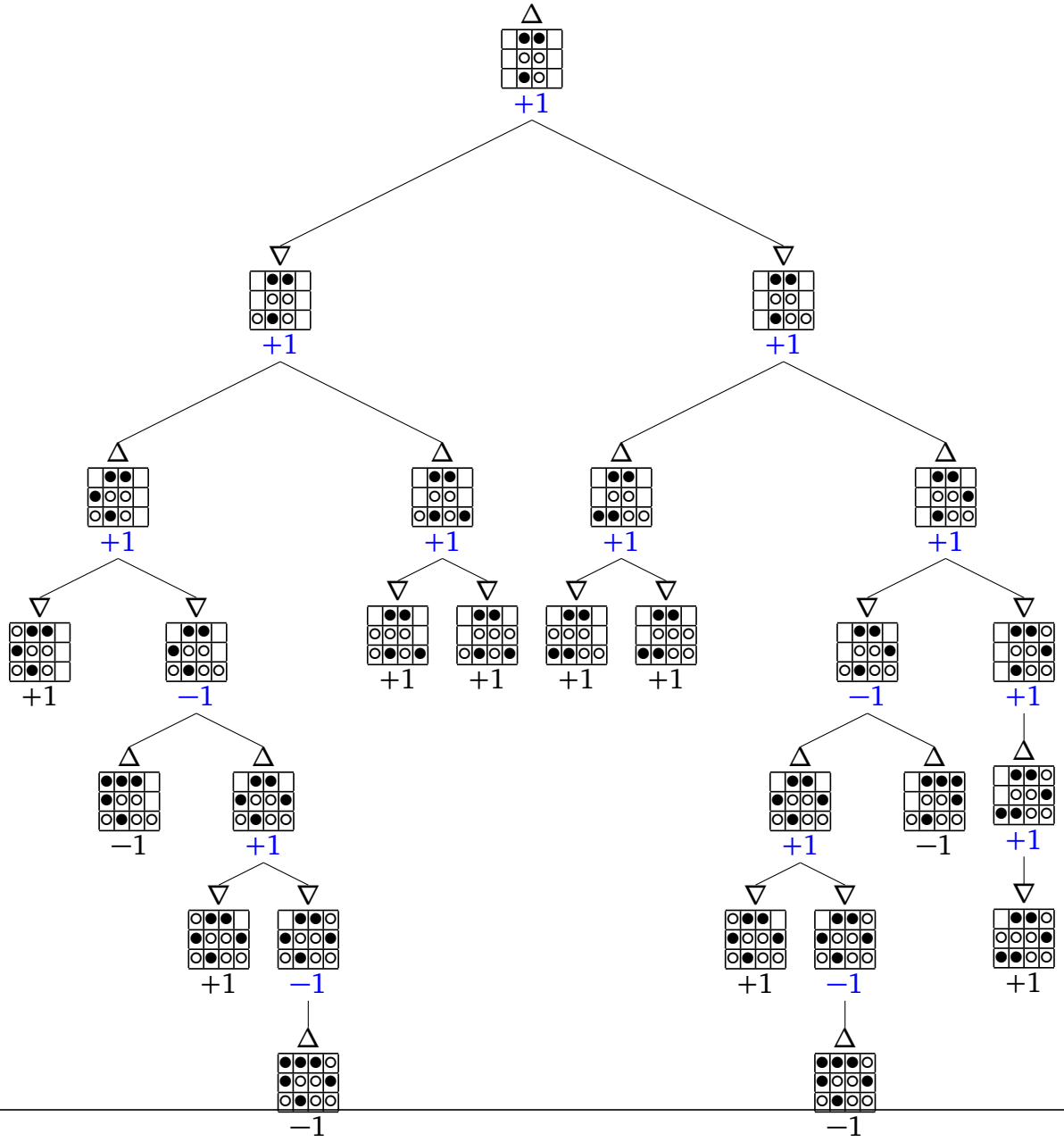
Anmerkung: Die Festlegung von $B = 1$ und $C = 0$ hat auch Konsequenzen auf das erste Constraint, das man nun als $A + 1 = 10 \cdot U$ schreiben könnte. Einfaches Forward-Checking überprüft nur Arc Consistency mit der zuletzt zugewiesenen Variable (also nur Costraints, die auch B involvieren). Erweitert man Forward Checking jedoch auf den AC-3 Algorithmus (→ Vorlesung) dann würde auch für jede Variable, deren Wertebereich verändert wurde (also nach der Zuweisung $B = 1$ wären das A und U) Arc Consistency überprüft. AC-3 könnte also sofort nach den Zuweisungen $C = 0$ und $B = 1$ erkennen, daß $U = 1$ und somit auch $A = 9$ sein muß. Es würde also praktisch ohne Suche konvergieren.

Aufgabe 2 Minimax, Alpha-Beta Suche

a) siehe b)

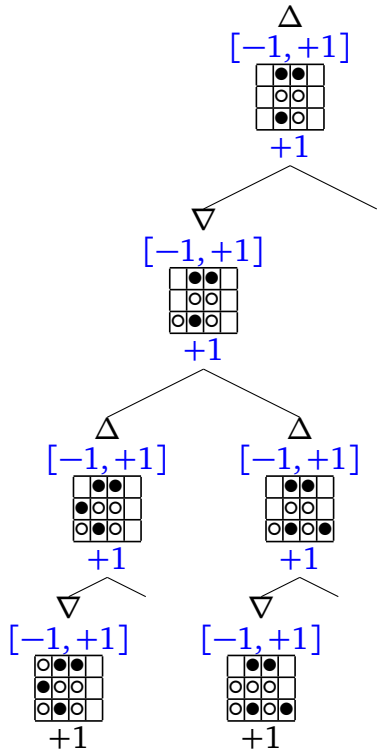
b) Der Minimax-Wert eines Zustandes ist unterhalb des entsprechenden Knoten dargestellt. Die blauen Werte entsprechen dabei Bewertungen, die im Laufe des Algorithmus ermittelt werden.

Minimax Algorithmus:

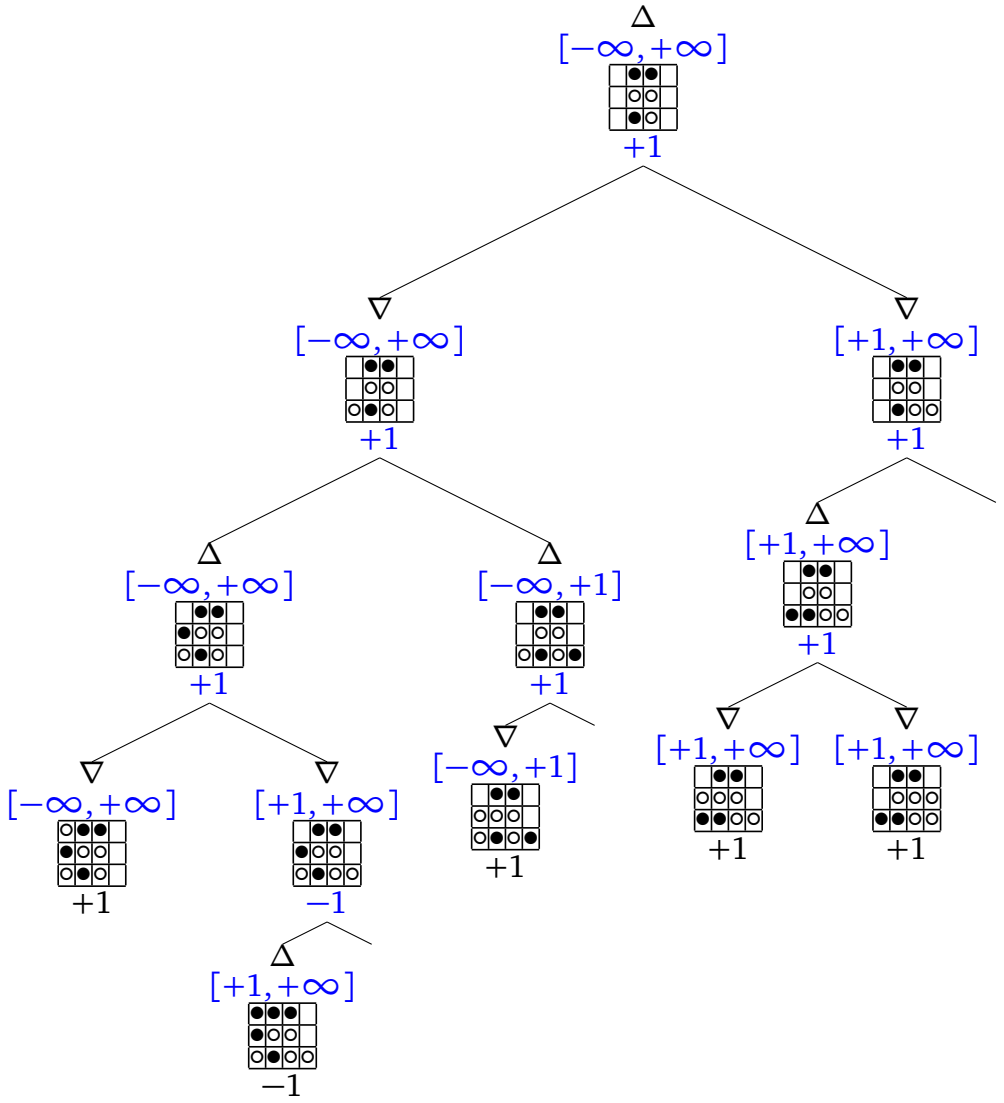

+1
+1
+1
+1
+1
+1
+1
+1
+1
+1
+1
+1
+1
−1
+1
−1
−1
+1
+1
−1
+1
−1
+1
+1
−1
+1
−1
−1

c) Für c) und d) stellt der Wert unterhalb eines Knotens nun den Rückgabewert der MaxValue bzw. MinValue Methode dar. Das heisst, entweder handelt es sich um den Minimax-Wert oder es wurde geprunt und stellt die letzte Bewertung dar. Oberhalb des Knotens sind die Schranken $[\alpha, \beta]$ beschrieben, mit denen der Knoten besucht wird.

AlphaBeta-Algorithmus mit den Schranken $[-1, +1]$ und Gewinne für Weiß bzw. für Schwarz wurden mit $+1$ bzw. -1 bewertet:



d) AlphaBeta-Algorithmus mit den Schranken $[-\infty, +\infty]$ und Gewinne für Weiß bzw. für Schwarz wurden mit $+1$ bzw. -1 bewertet:


△
[−∞,+∞]
+1
▽
[−∞,+∞]
+1
▽
[+1,+∞]
+1
△
[−∞,+∞]
+1
△
[−∞,+1]
+1
△
[+1,+∞]
+1
▽
[−∞,+∞]
+1
▽
[+1,+∞]
−1
▽
[−∞,+1]
+1
▽
[+1,+∞]
+1
▽
[+1,+∞]
+1
△
[+1,+∞]
−1

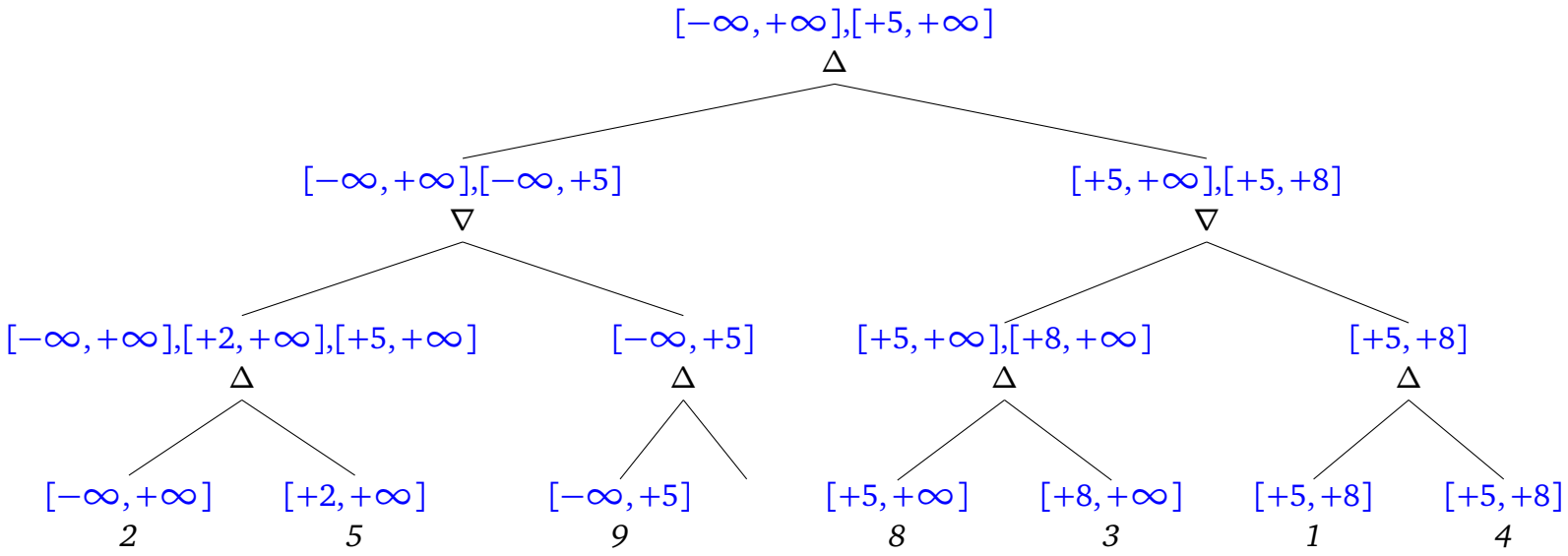
In Teilaufgabe c) konnten mehr Zustände geprunt werden, da im Gegensatz zu d) die Ausgangs-Schranken bezüglich der Bewertungsfunktion optimal waren. Es lässt sich keine größere untere Schranke und keine kleinere obere Schranke angeben. Im Allgemeinen führen schärfere Schranken zu einem gleichwertigen oder erhöhtem Pruning.

In diesem konkreten Fall wiederholte sich auf ähnliche Weise folgende Situation: nach der Evaluation des vierten Knotens, das von einem Max-Knoten aufgerufen wurde, war mithilfe der β-Schranke klar, dass es sich um den maximalen Wert handelt (hier war es sogar der maximal mögliche Wert, und nicht nur der maximale Wert, den der Min-Spieler „noch erlaubt“). Somit ist eine weitere Evaluierung der Nachbarknoten unnötig. In Teilaufgabe d) ist diese Information nicht bekannt, so dass die Nachbarknoten weiter nach potentiell besseren Bewertungen durchsucht werden.

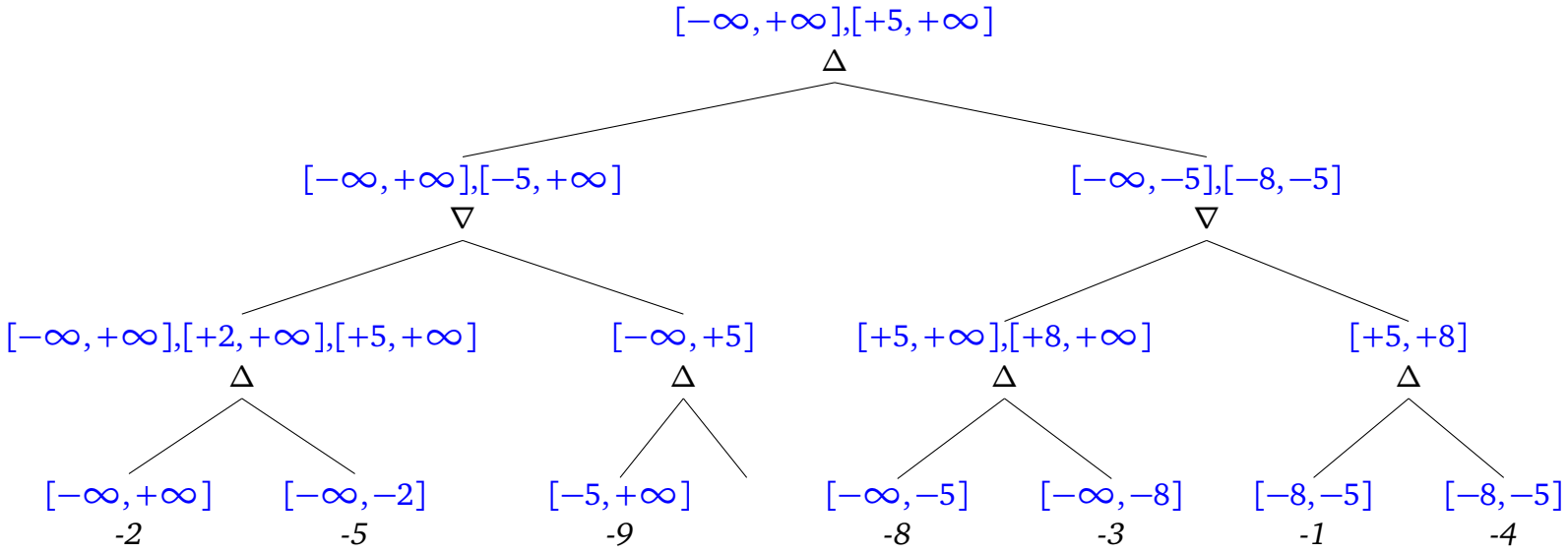
Aufgabe 3 NegaMax Formulierung, Minimal Window

In den folgenden Bäumen sind zu jedem besuchten Knoten die Anfangs-Schranken und eventuelle Anpassungen (der lokalen Schranken) angegeben. Die Werte sind dabei von links nach rechts zu lesen.

a) Alpha-Beta Algorithmus:



b) Alpha-Beta Algorithmus in der NegaMax-Formulierung:



c) Alpha-Beta mit Minimal Window Search:

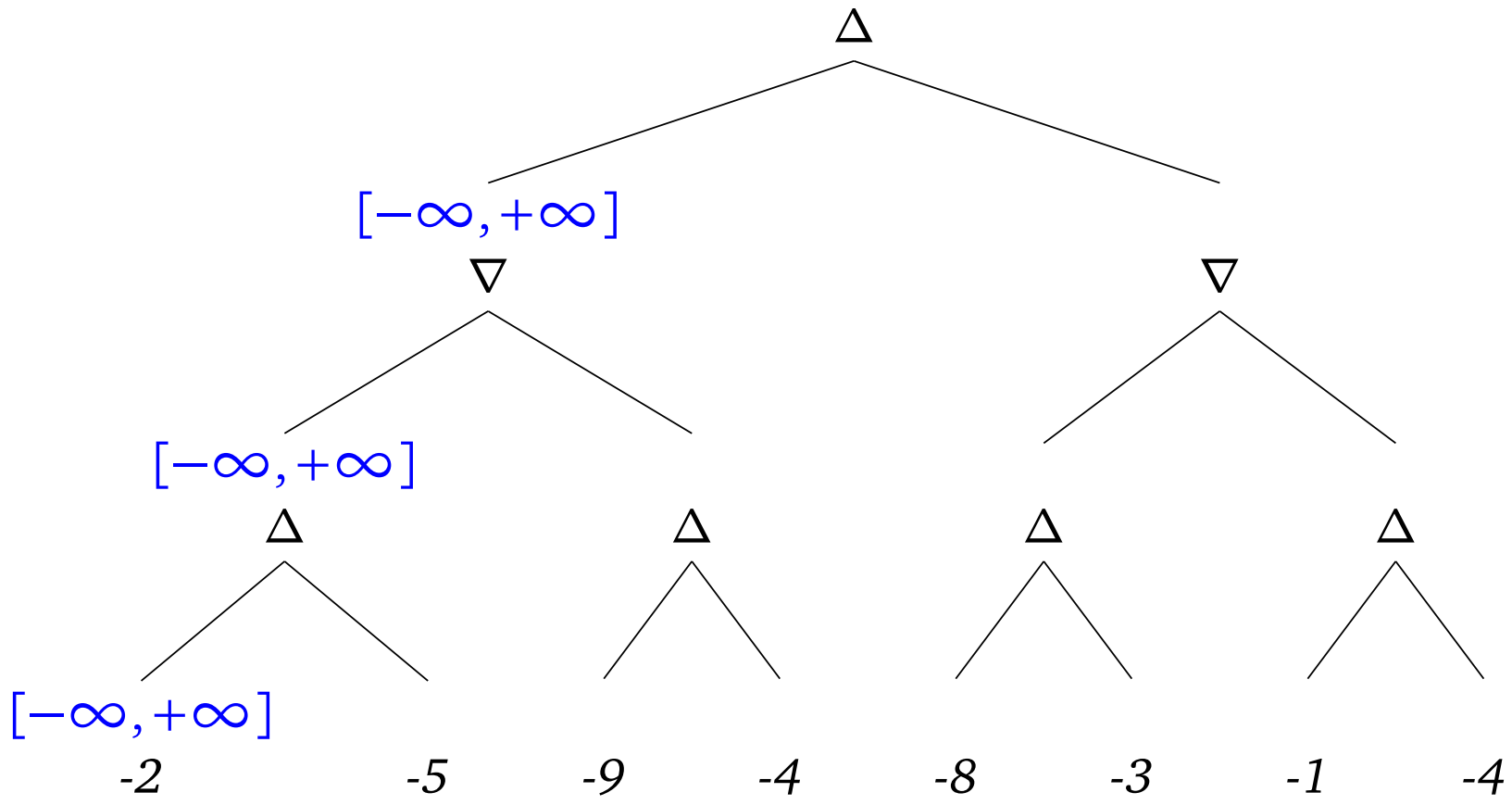
Siehe Aufgabe d), wobei alle Vorzeichenwechsel und Wechsel zwischen α und β-Werten nicht durchgeführt werden müssen, dafür aber abwechselnd maximiert (ungerade Suchtiefen) und minimiert (gerade Suchtiefen) werden muß.

d) Alpha-Beta in der NegaMax-Formulierung mit Minimal Window Search:

Abarbeitung des Algorithmus NegaScout (Folie 75):

Die Blattwerte werden in der NegaMax-Formulierung aus der Sicht des Ziehenden dargestellt (im konventionellen Setting immer aus der Sicht des Max-Spielers).

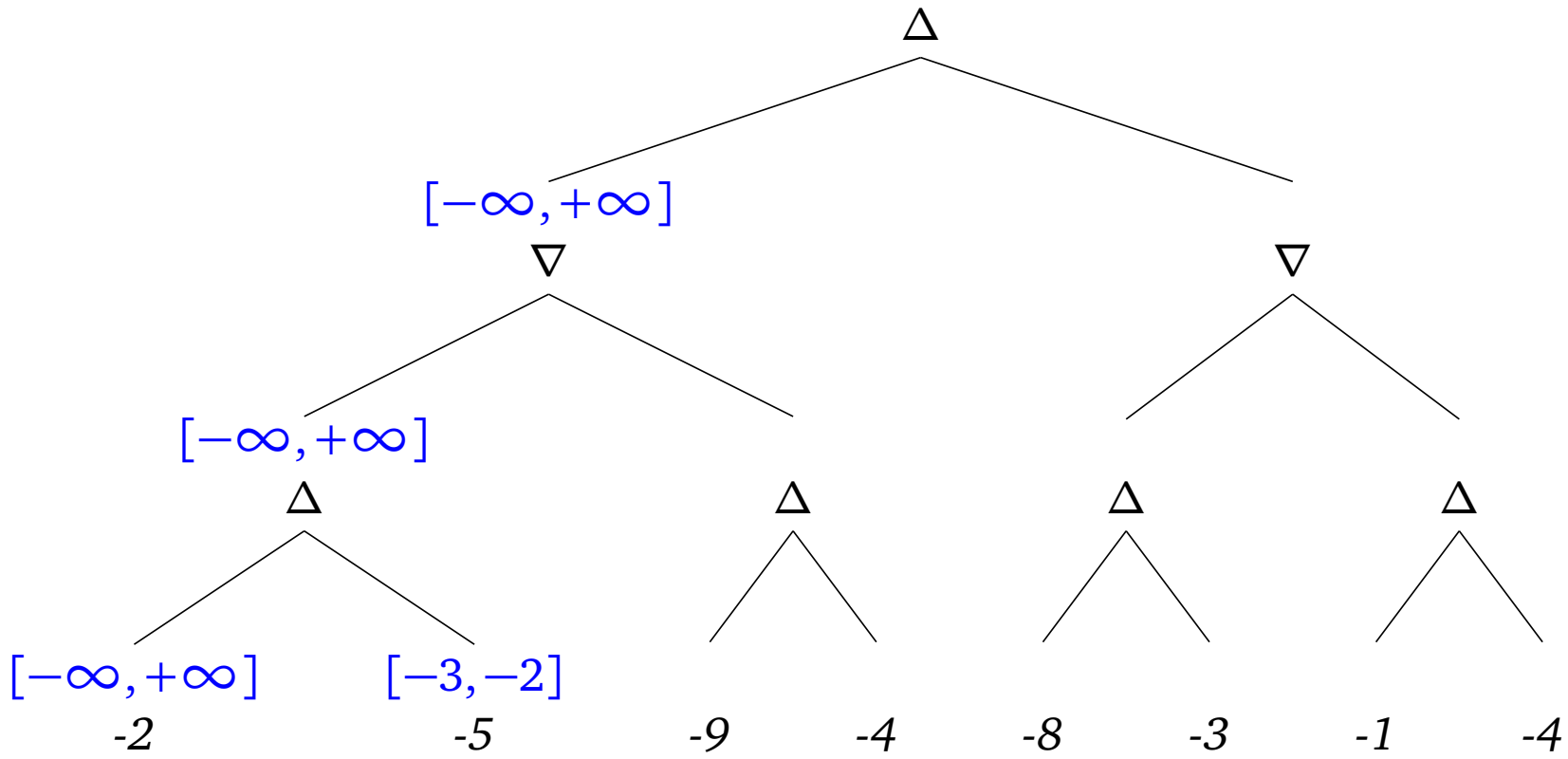
Aufruf mit $[-\infty, +\infty]$, die Werte werden bis zum Knoten -2 links unten durchgereicht (wobei bei der jeweils nächsten Ebene $\beta = -\alpha$ und $\alpha = -\beta$ gesetzt wird, was sich aber nicht bemerkbar macht.



An dieser Stelle wird jetzt Wert -2 an die aufrufende Instanz von NegaScout zurückgeliert, und somit die Variable t auf $-(-2)$ also auf $+2$ gesetzt.

t is nun größer als a (wurde mit dem Wert von α, also mit $-\infty$ initialisiert), und kleiner als β, wir sind aber noch beim ersten Zug, d.h. $i = 1$, daher passiert noch keine Re-Search, die untere Schranke a wird auf das Maximum des alten Wertes von a $(-\infty)$ und des Wertes von t, also auf 2 gesetzt. a is somit nach wie vor kleiner als β (∞) und der nächste Zug muß durchsucht werden.

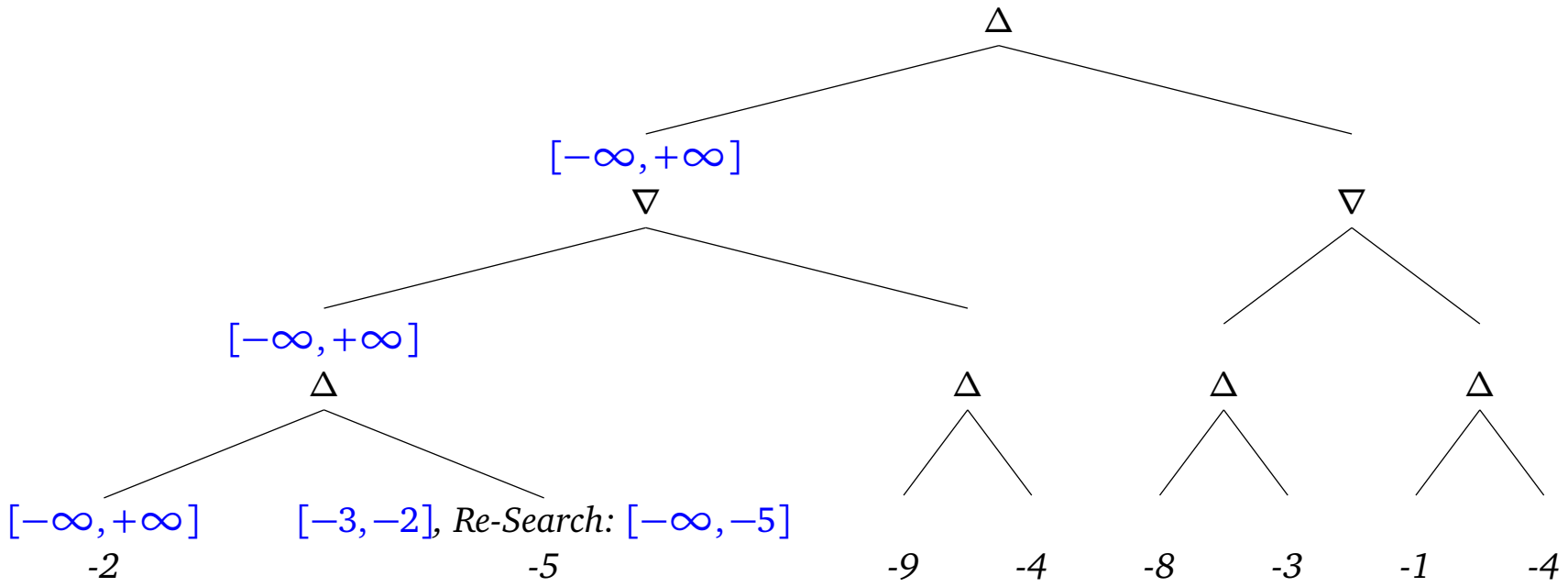
Abweichend von der Aufgabe b) wird jetzt jedoch nicht das Interval $[a, \beta]$ zum Durchsuchen des nächsten Zugs verwendet, sondern das Intervall $[a, a+1]$, also $[2,3]$. Im rekursiven Aufruf von NegaScout wird das dann zu $[-3,-2]$.



Der Aufruf liefert als Ergebnis -5 zurück, t wird also auf den Wert 5 gesetzt. Nun liegt ein sogenanntes *Fail-High* vor (alle Bedingungen der IF-Abfrage sind erfüllt: $t > a = 2$, $t < \beta$, $i = 2 > 1$).

NegaScout hatte ja oben angenommen, daß es keinen besseren Wert als 2 gibt, und daher mit $\beta = 3$ gesucht. Diese Annahme hat sich nun als falsch herausgestellt, und dieser Knoten (im allgemeinen Fall der gesamte Unterbaum mit diesem Knoten als Wurzel) muß nochmals durchsucht werden, um den genauen Wert zu bestimmen. In diesem Fall wird der Baum nochmals mit dem Intervall $[5, \infty]$ durchsucht, da

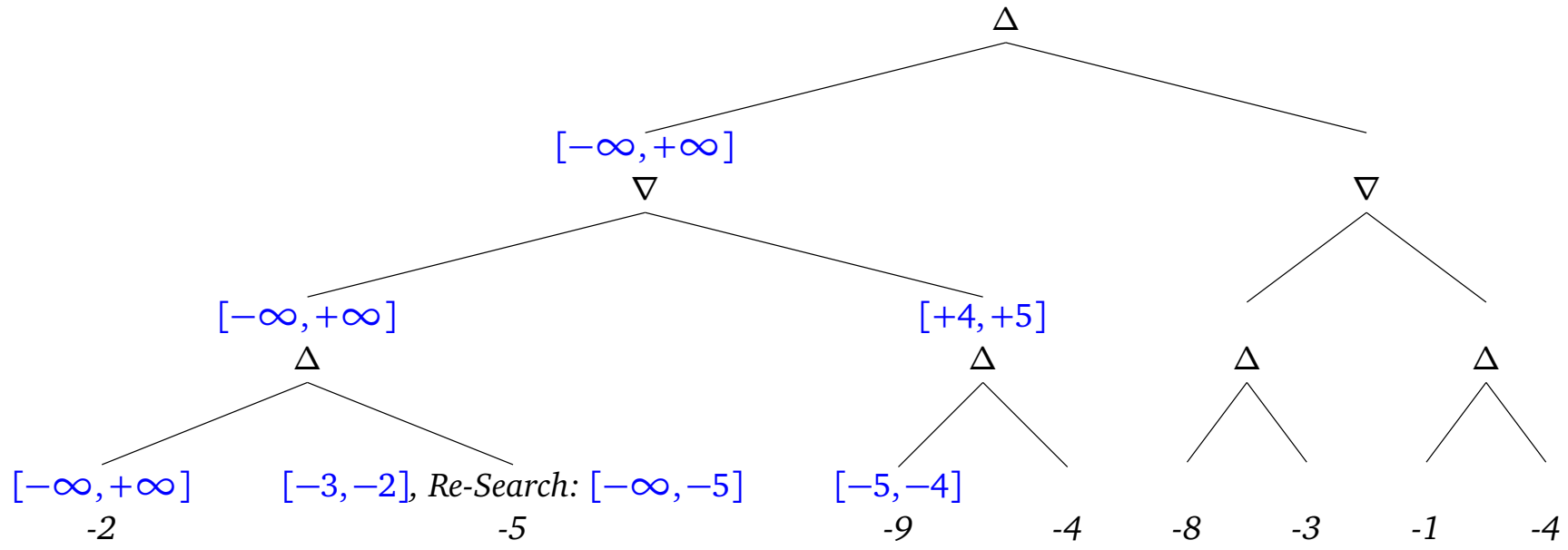
wir ja das Fail-High mit einem Wert von $t = 5$ angezeigt bekommen haben, d.h. wir wissen daß der korrekte Wert ≥ 5 ist.[1] Im rekursiven Aufruf von NegaScout wird das dann zu $[-\infty, -5]$.



Durch diese abermalige Suche (Re-Search) wird nun festgestellt, daß der Wert genau -5 ist, was im Max-Knoten zu $+5$ wird. Da es keine weiteren Züge mehr gibt, wird dieser Wert $a = 5$ an den aufrufenden Min-Knoten zurückgemeldet.

Dieser erhält also das Resultat $t = -5$ als Ergebnis der Suche seines linken Teilbaums, und durchsucht nun den rechten Teilbaum (analog zu oben) mit dem Minimal Window $[-5, -4]$, was im rekursiven NegaScout-Aufruf zu $[+4, +5]$ wird. Diese Werte werden (wiederum invertiert) an das Blatt weitergegeben.

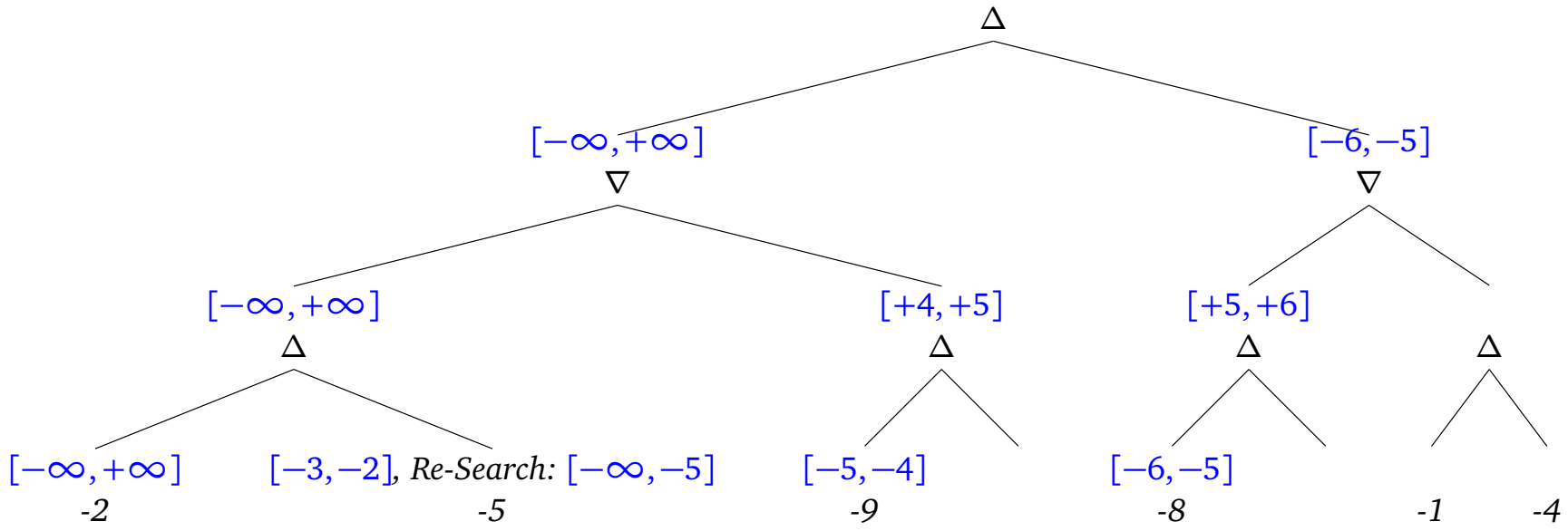
[1] In diesem Fall wäre es einfach zu sehen, daß der korrekte Wert $= 5$ ist, da ja der gesamte neu zu dursuchenden Unterbaum nur aus einem Blattknoten besteht. Im allgemeinen Fall weiß man aber nur, daß es einen Wert grösser als $a + 1$ gegeben hat, weiß aber nicht, ob dieser Wert schon der beste ist, da andere geprunt worden sein könnten.



Der Max-Knoten erhält also den Wert $t = -(-9) = +9$. Der Wert ist zwar größer als a $(-\infty)$, aber er ist auch größer als β (5). Das bedeutet, daß dieser Zug "zu gut" ist, d.h., Min kann im Knoten darüber durch das Spielen des anderen Zuges dafür sorgen, dass Max nicht Gelegenheit erhält, den Zug mit 9 Punkten zu spielen. Weil $t > \beta$ passiert auch keine Re-Search und a wird auf das Maximum von a und t gesetzt (also auf $t = 9$), wodurch danach gilt $a > \beta$. Somit werden keine weiteren Züge durchsucht, sondern sofort $a = 9$ als Ergebnis an den Min-Knoten weitergeleitet (*cut-off*).

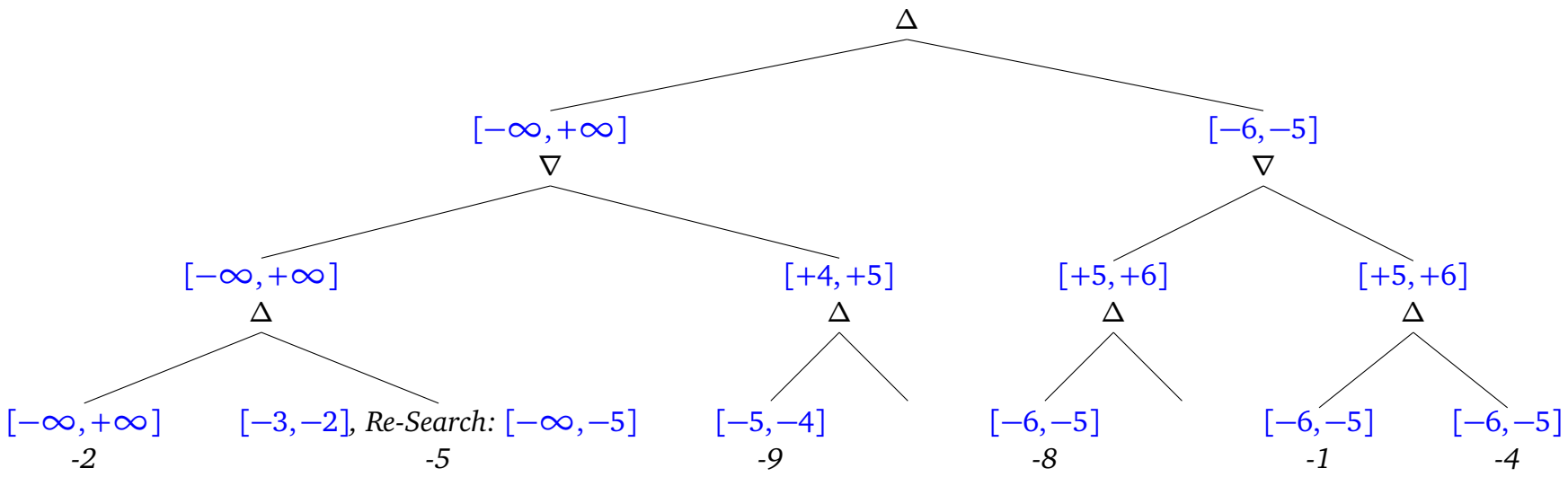
Der Min Knoten erhält also $t = -9$ als Ergebnis, bildet das Maximum von $a = -5$ (aus der Suche im linken Teilbaum) und $t = -9$. t ist hier kleiner als a, d.h. das Re-Search wird nicht ausgeführt und der Wert von a bleibt unverändert. Da es keine weiteren Züge mehr gibt, wird nun -5 als Resultat an den Wurzelknoten weitergegeben, der also einen Wert von $t = -(-5) = +5$ erhält.

Der Wurzelknoten setzt nun a von $-\infty$ auf 5. Das ist immer noch kleiner als das β in diesem Knoten (∞), weshalb kein cut-off passieren kann. Alpha-Beta würde dann den rechten Zweig mit $[+5,+\infty]$ durchsuchen, aber die Minimal Window Search setzt die obere Schranke b auf $a+1$ und ruft den nächsten Knoten mit $[-6,-5]$ auf. Dieser Wert wird wieder bis an das Blatt durchgegeben.



Hier wird nun der Wert -8 an den darüber liegenden Max-Knoten zurückgegeben, der erhält also den Wert $t = -(-8) = +8$, der über seinem β-Wert von $+6$ liegt. Es passiert also ein cut-off, der bei einer normalen Alpha-Beta-Suche nicht passiert wäre, und der rechte Zweig wird nicht weiter durchsucht. Als Ergebnis wird $+8$ nach oben durchgereicht, was dann als $t = -8$ im Min-Knoten ankommt. Dieses t ist kleiner als das a, welches in diesem Knoten noch -6 ist, der a-Wert bleibt also unverändert, und der rechte Zweig wird wiederum mit $[-6,-5]$ bzw. $[+5,+6]$ durchsucht.[2]

[2] Reguläres Alpha-Beta hätte hier das Intervall von $[+5,+\infty]$ auf $[+5,+8]$ verkleinert (siehe Aufgaben a) und b)), was hier aber nicht notwendig ist, da es ohnehin schon auf $[+5,+6]$ ist.

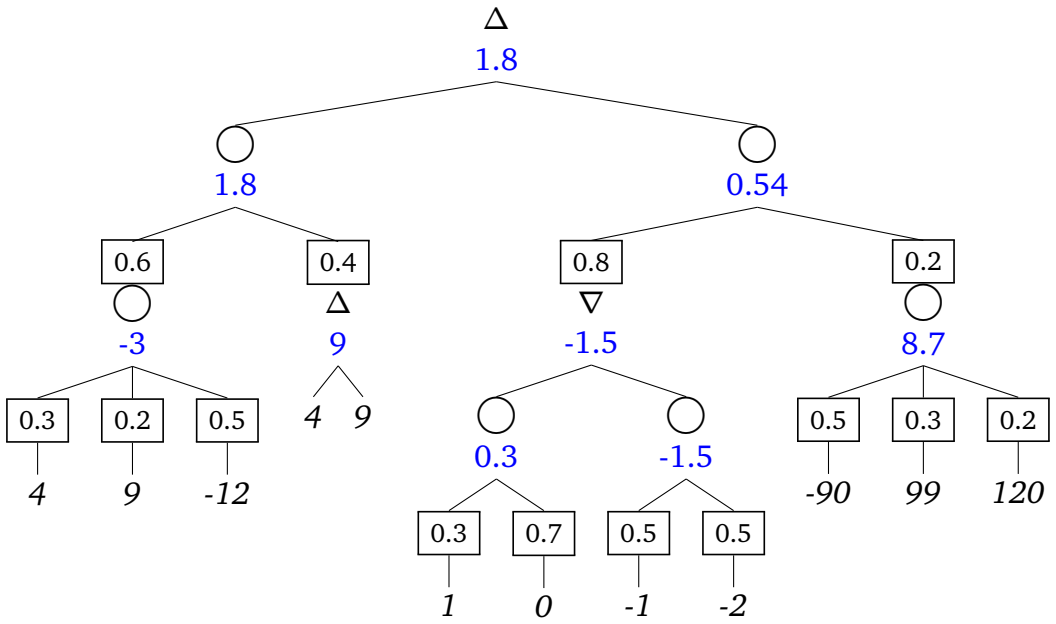


Der Max-Knoten erhält nun $t = +1$ als negierten Wert aus dem Min-Knoten. Da $t < \alpha$ ist, passiert nichts und der letzte Knoten wird ebenfalls mit $[-6,-5]$ durchsucht. Dieser retourniert $t = +4$, was ebenfall kleiner als α ist. Der Wert a ist also unverändert 5 (aufgrund der Initialisierung $a = \alpha = +5$), was als Ergebnis an den Min-Knoten weitergeben wird, der es als -5 erhält. Das bedeutet, dass in diesem Zweig keine Variante zu finden ist, die für den Max-Spieler ein besseres Ergebnis als +5 liefert, daß also nach wie vor die untere Schranke +5 aus dem linken Teilbaum der Wurzel gelten soll. Der Min-Knoten bildet jetzt das Maximum von $t = -5$ und dem α-Wert $a = -6$. Der Wert von -6 zeigt, an, daß Max im linken Teilbaum des Min-Knotens einen Wert von 6 oder besser hat (wir hatten ja oben schon +8 gefunden, und dann einen Ast geprunt). Der Min-Spieler würde hier seinen rechten Zug wählen, wodurch sich das Prunen des Astes neben dem +8-Knoten nachträglich als korrekt herausgestellt hat.

Der Min-Knoten retourniert nun -5 an den aufrufenden Max-Knoten, der es als $t = +5$ erhält. Es wurde also im rechten Teilbaum keine Variante gefunden, die einen Wert besser als 5 liefern würde. Daher würde Max an der Wurzel den linken Teilbaum wählen, bei dem er einen Wert von 5 erwarten kann.

Aufgabe 4 Expectiminimax

a)


Δ
1.8
1.8
0.54
0.6
0.4
0.8
0.2
-3
Δ
9
∇
-1.5
8.7
0.3
0.2
0.5
4
9
0.3
-1.5
0.5
0.3
0.2
4
9
-12
-90
99
120
0.3
0.7
0.5
0.5
1
0
-1
-2