Algorithmen und Datenstrukturen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Felix Kuhr (Übungen)

sowie viele Tutoren

AVL-Bäume

- Ein binärer Suchbaum heißt **AVL-Baum**, falls für die beiden Teilbäume *T1* und *T2* der Wurzel gilt:
  - $|h(T1) - h(T2)| <= 1$
  - *T1* und *T2* sind ihrerseits AVL-Bäume.
- Der Wert $|h(T1) - h(T2)|$ wird als **Balancefaktor** (BF) eines Knotens bezeichnet. Er kann in einem AVL-Baum nur die Werte -1, 0 oder 1 (dargestellt durch -, = und +) annehmen.
- Jeder AVL-Baum ist ein binärer Suchbaum.
- Strukturverletzungen durch Einfügen oder Entfernen von Schlüsseln erfordern **Rebalancierungsoperationen**.

G. M. Adelson-Velskii, E. M. Landis; In: Doklady Akademii Nauk SSSR 146, S. 263-266, **1962**; Englische Übersetzung in Soviet Math 3, S. 1259-1263

Danksagung

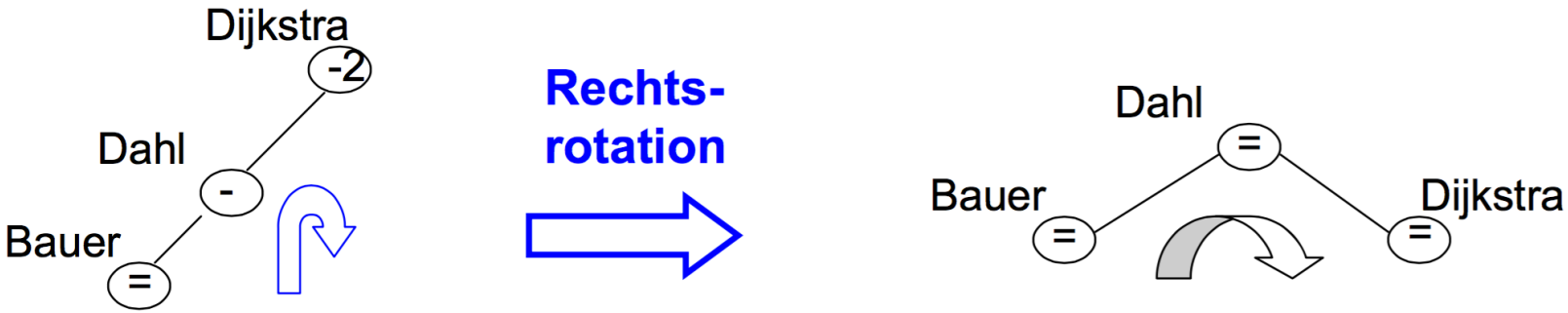
Die AVL-Präsentationen wurden übernommen aus:

- „Informatik II“ (Kapitel: Balancierte Bäume) gehalten von Martin Wirsing an der LMU http://www.pst.ifi.lmu.de/lehre/SS06/infoII/

Einfügen in AVL-Baum

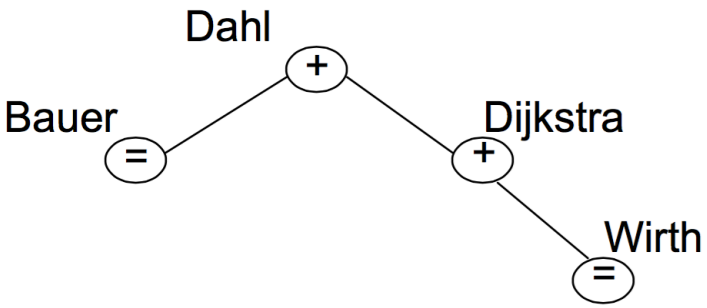
Einfügen von Dahl: Neuberechnung des Balancierungsfaktors, AVL Kriterium erfüllt

Einfügen von Bauer: Verletzung des AVL Kriteriums



Nach Rechtsrotation: AVL Kriterium wieder erfüllt

Einfügen in AVL-Baum

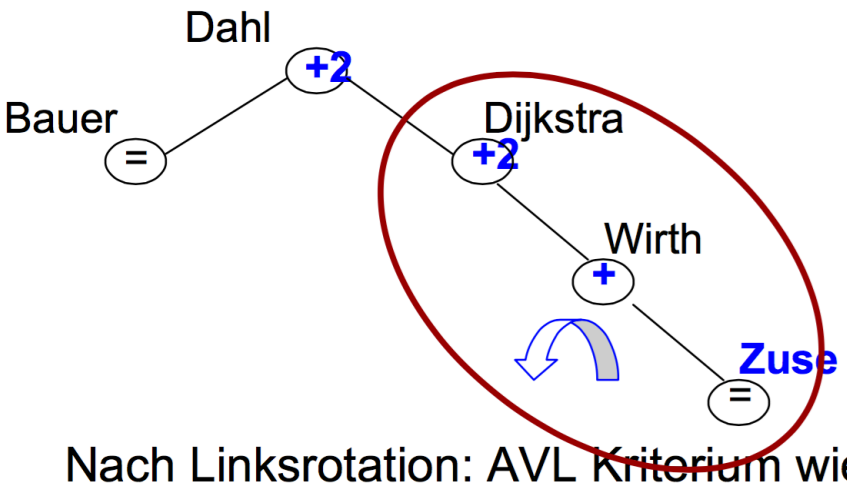


Einfügen von Wirth:

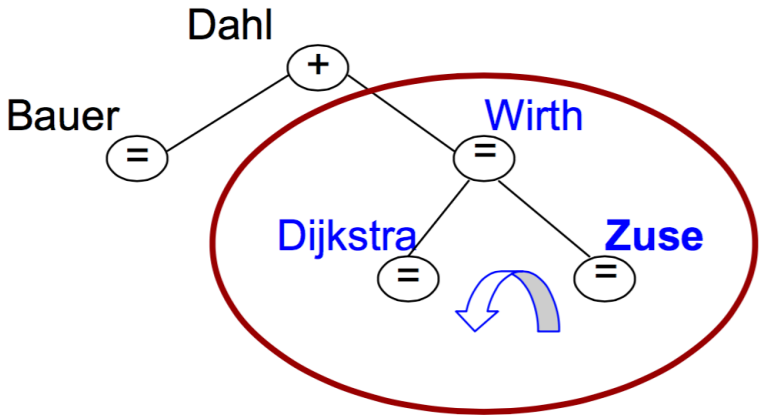
Nach Neuberechnung des Balancierungsfaktors ist

AVL-Kriterium weiter erfüllt

Einfügen von Zuse:

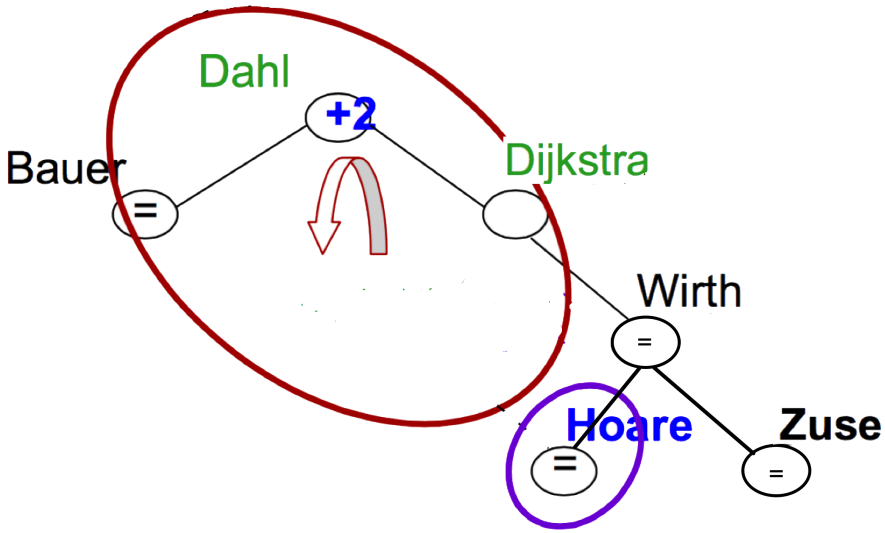


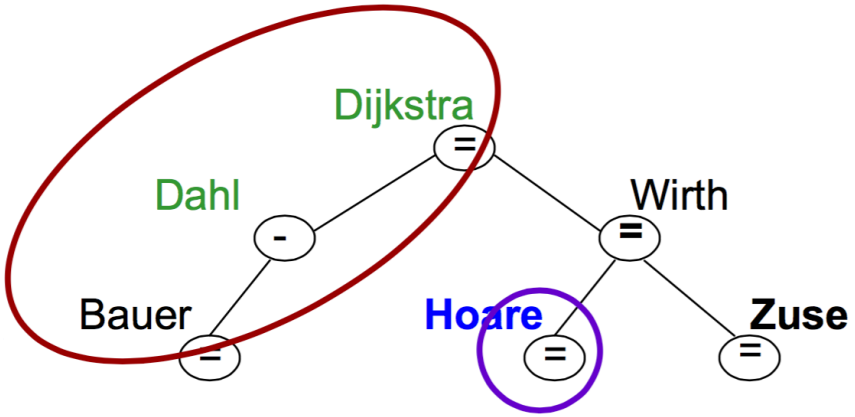
Links-rotation



Nach Linksrotation: AVL Kriterium wieder erfüllt

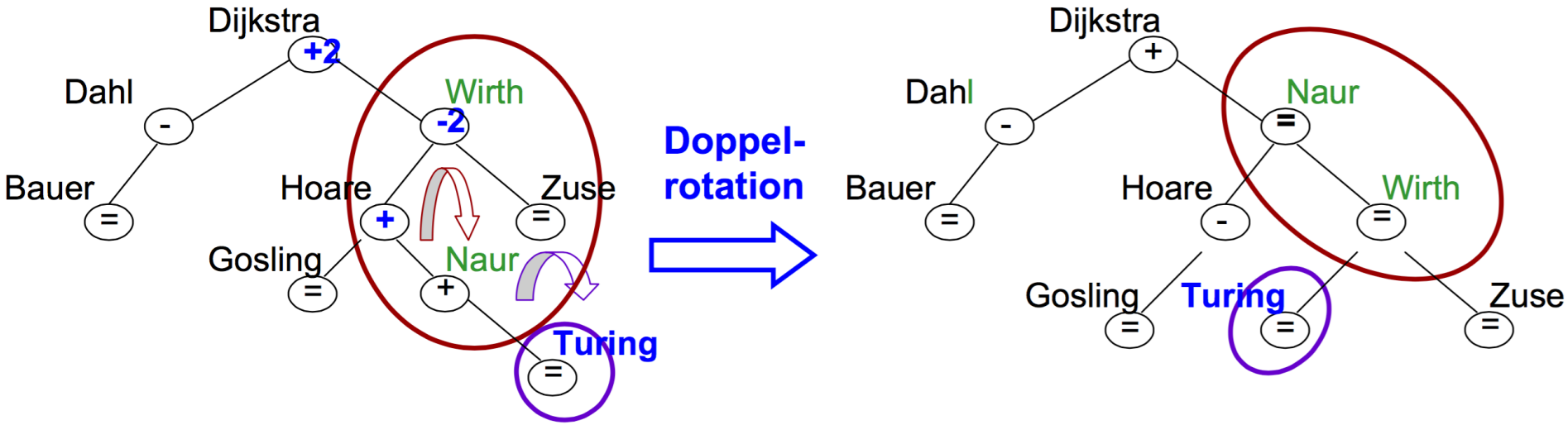
Einfügen in AVL-Baum





Einfügen in AVL-Baum

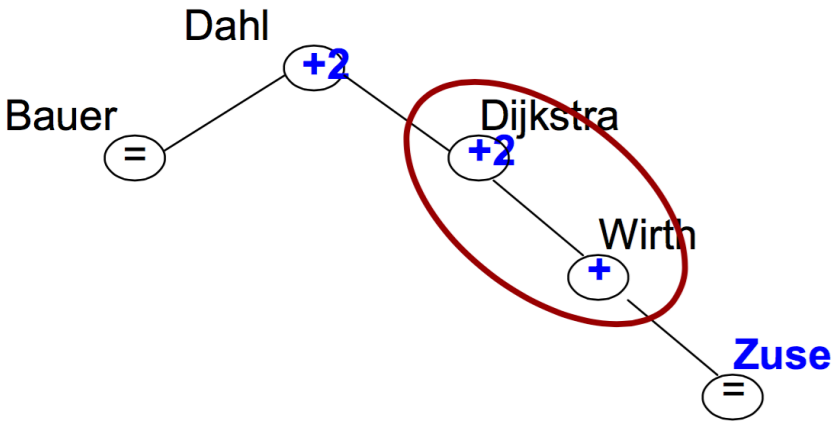
Einfügen von Turing: Noch einmal Doppelrotation

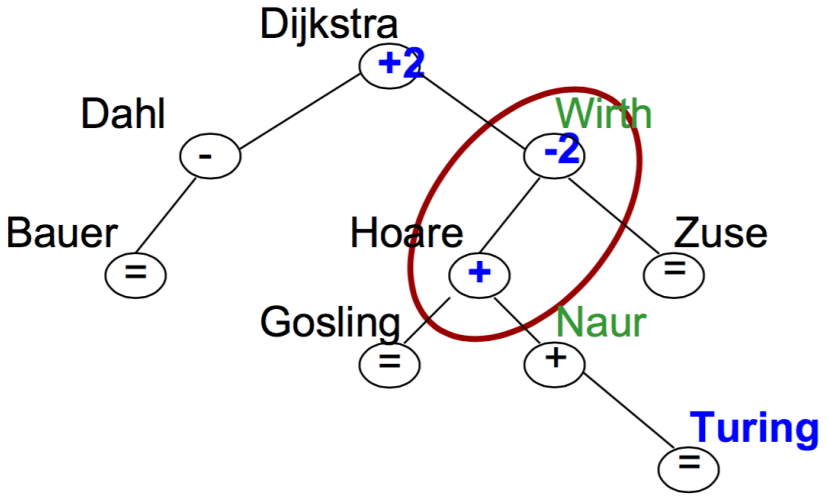


- Doppelrotation stellt AVL Kriterium wieder her
- Sind die vorgestellten Rotationen ausreichend?

Anwendungsstelle der Rotation

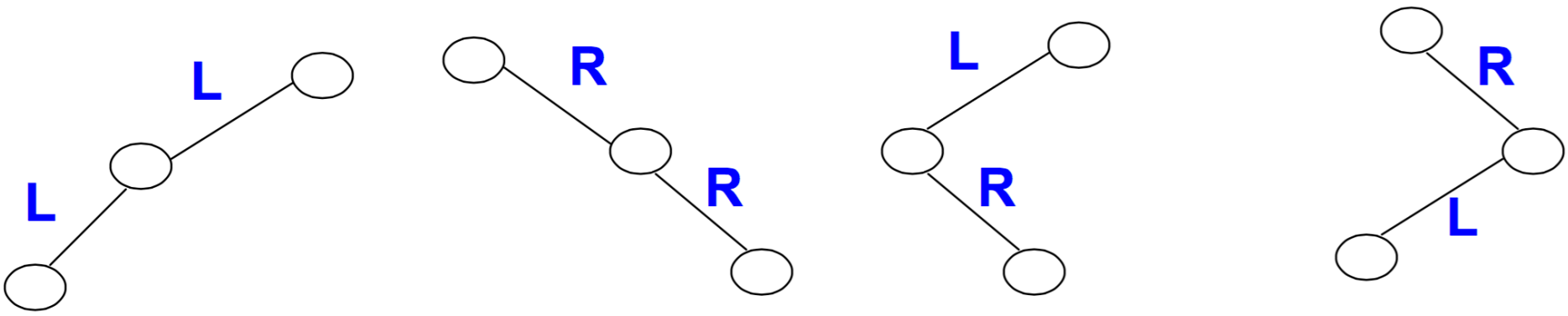
- Veränderungen der Balancierungsfaktoren geschehen ausschließlich auf dem **Pfad von der Wurzel zur Einfügeposition**
- Ausgangspunkt der Rotation ist immer der „**tiefste“ Elternknoten mit BF = ±2** (dieser Knoten hatte vorher BF = ±1)
- Der (auf dem Pfad) **darunter liegende Knoten** hat **BF = ±1**





Rotationstypen

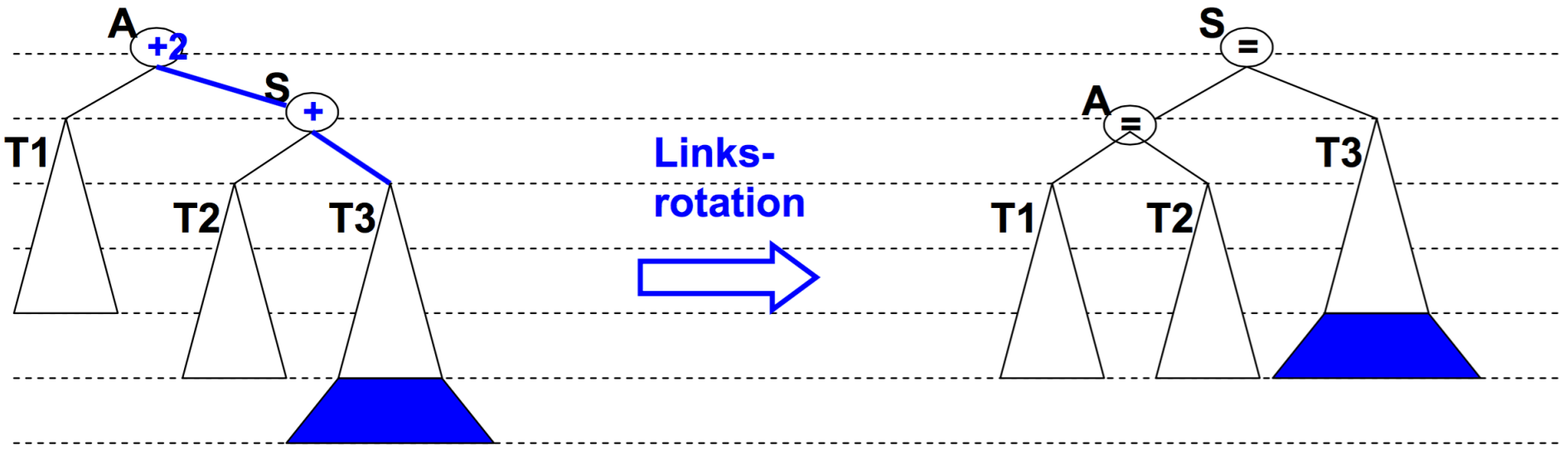
- Betrachte ausgehend vom tiefsten Knoten mit BF = 2 den Pfad zur Einfügeposition:
  - **RR: Rechts-Rechts** Linksrotation
  - **LL: Links-Links** Rechtsrotation
  - **RL: Rechts-Links** Doppelrotation „rechts“
  - **LR: Links-Rechts** Doppelrotation „links“



- Rotation ist immer eindeutig bestimmt
- Jetzt genauere Betrachtungen der einzelnen Typen

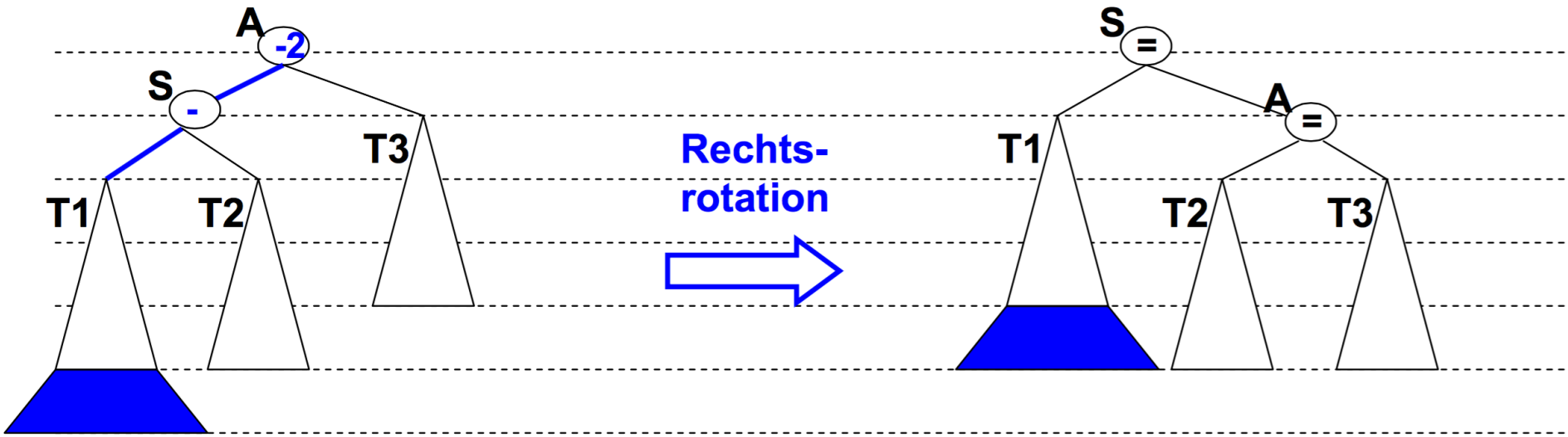
Typ RR: Linksrotation

- Wir bezeichnen den „tiefsten“ Knoten mit Strukturverletzung mit A, dessen Kind mit S und den Enkelknoten mit B
- Bei der Linksrotation hat S den BF „+“ und A den BF „+2“



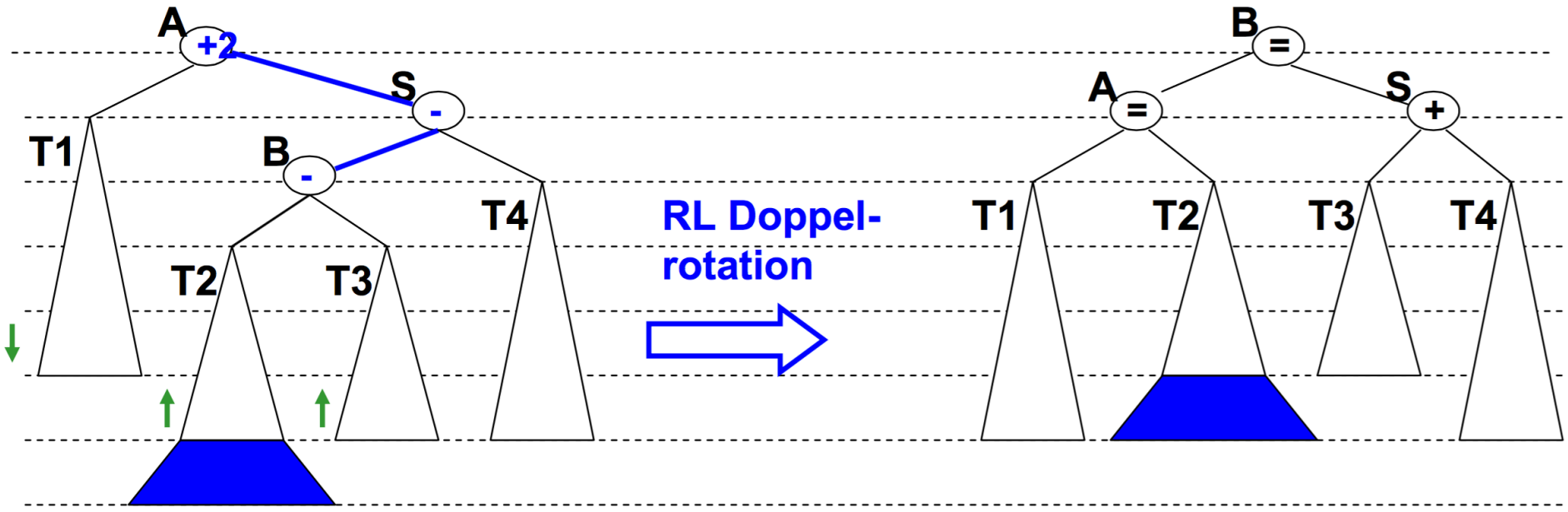
Typ LL: Rechtsrotation

- Bei der Rechtsrotation hat S den BF „-“ und A den BF „-2“



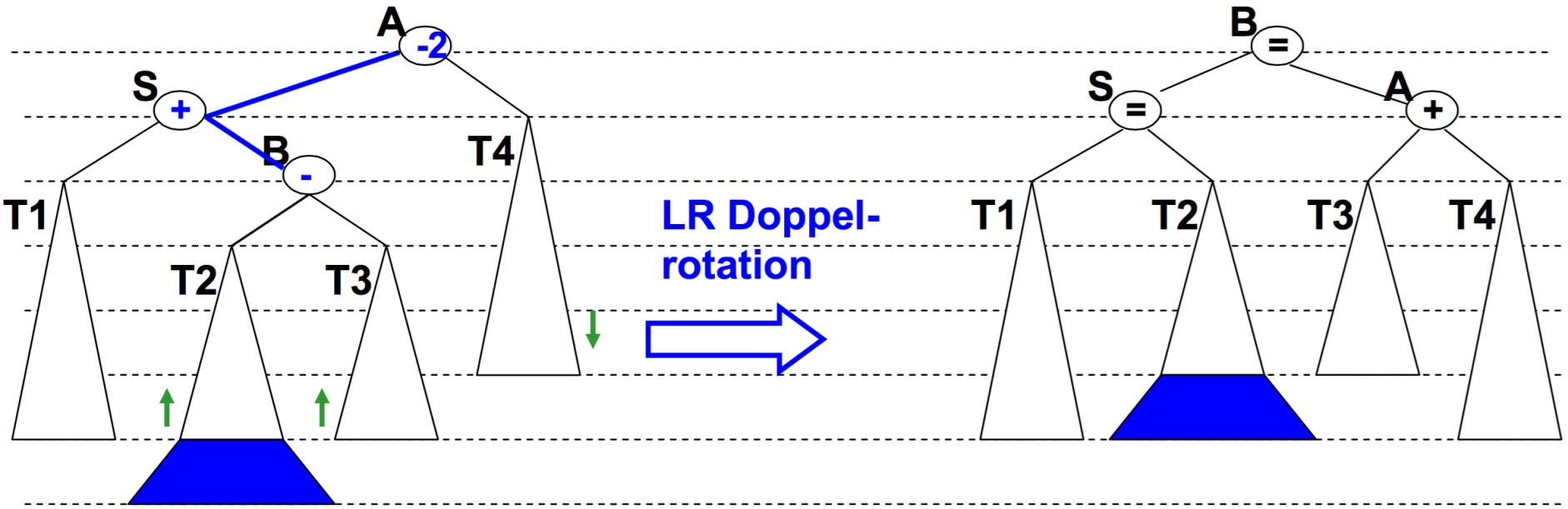
Typ RL: Doppelrotation

- Bei der RL-Doppelrotation hat A den BF „+2“, S den BF „- “, B den BF „+“ oder „-“
- Wir wählen „-“ für den BF von B.



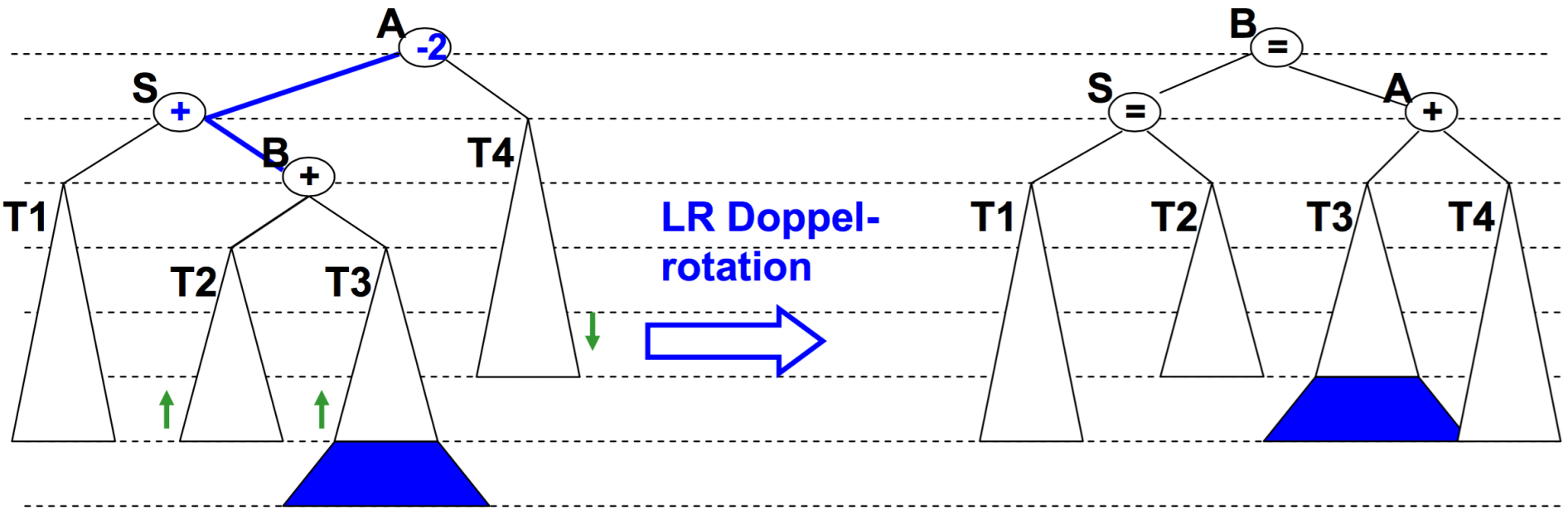
Typ LR: Doppelrotation

- Bei der LR-Doppelrotation hat A den BF „-2“, S den BF „+“, B den BF „+“ oder „-“
- Wir wählen „-“ für den BF von B.



Typ LR: Doppelrotation

- Variante der LR-Doppelrotation mit Balancefactor „+“ für B



Löschen von Knoten in AVL-Bäumen

- Löschen erfolgt wie bei Suchbäumen und ...
- kann zu Strukturverletzungen führen (wie beim Einfügen), ...
- … die durch Rotationen ausgeglichen werden
  - Es genügt nicht immer eine einzige Rotation oder Doppelrotation (Restrukturierung)
  - Im schlechtesten Fall:
    - auf dem Suchpfad bottom-up vom zu entfernenden Schlüssel bis zur Wurzel
    - auf jedem Level Rotation bzw. Doppelrotation
    - Aufwand O(log n)

Vergleich

AVL-Baum:

- search(k): O(log n)
- insert(e): O(log n)
- delete(k): O(log n)

Restrukturierungen:

- insert(e): max. 1
- delete(k): max. log n

Rot-Schwarz-Baum:

- search(k): O(log n)
- insert(e): O(log n)
- delete(k): O(log n)

Restrukturierungen:

- insert(e): max. 1
- delete(k): max. 2

Splay-Baum:

- search: O(log n) amort.
- insert: O(log n)
- delete: O(log n)