Algorithmen und Datenstrukturen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Tanya Braun (Übungen)

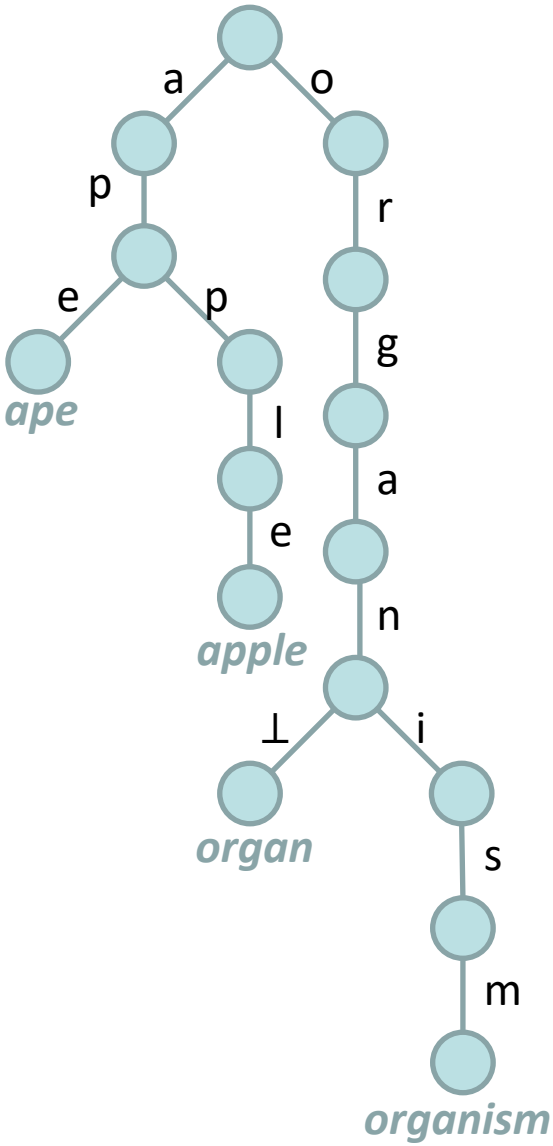
sowie viele Tutoren

Danksagung

Die nachfolgenden Präsentationen wurden mit einigen Änderungen übernommen aus:

- „Algorithmen und Datenstrukturen“
  gehalten von Sven Groppe an der UzL

Idee: Ausnutzung von gemeinsamen Präfixen



Trie

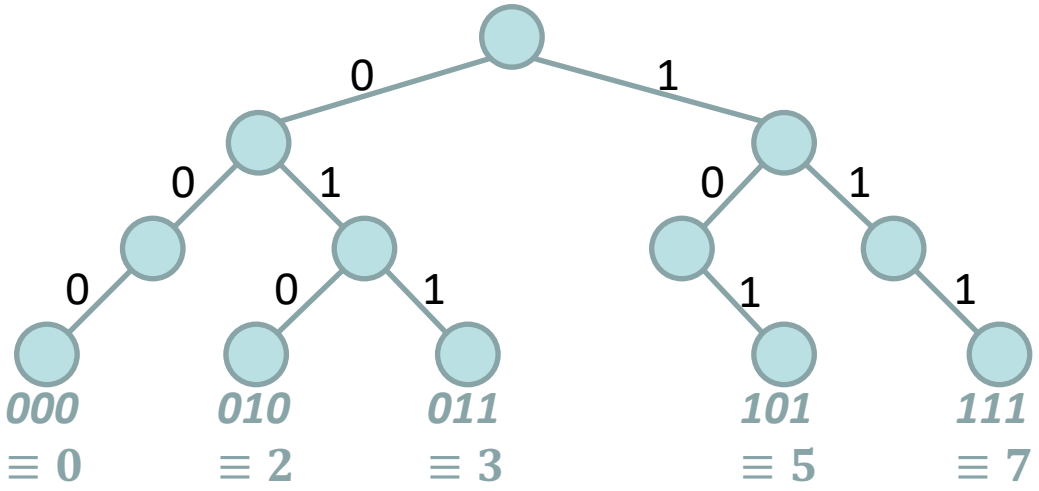
- Repräsentation von Mengen von Zeichenketten
- Name stammt nach *Edward Fredkin* von re**TRIE**val
  - Anwendungen von Tries finden sich im Bereich des *Information Retrieval*
    - Informationsrückgewinnung aus bestehenden komplexen Daten (Beispiel Internet-Suchmaschine)
  - auch Radix-Baum oder Suffix-Baum genannt
- Relativ kompakte Speicherung von Daten insbesondere mit gemeinsamen Präfixen

Edward Fredkin, Trie Memory,
Communications of the ACM 3 (9), S. 490–499, **1960**

Trie

- Voraussetzung
  - Daten darstellbar als Folge von Elementen aus einem (endlichen) Alphabet
    - Beispiele
      - Zeichenkette „Otto“ besteht aus Zeichen ‚O‘, ‚t‘, ‚t‘ und ‚o‘ (Alphabet ist alle Zeichen)
      - Zahl 7 ist darstellbar als Folge von Bits 111 (Alphabet ist {0, 1})
  - Unter den Elementen aus dem Alphabet besteht eine Ordnung

Trie – Beispiel mit Bits als Alphabet



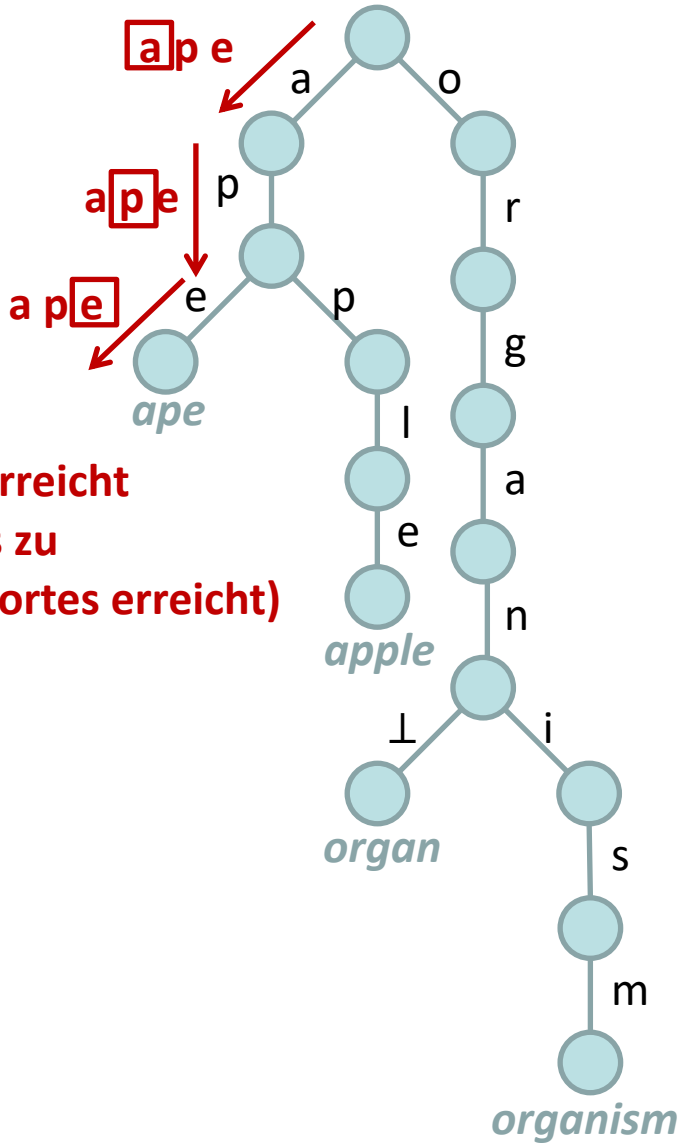
Trie - Definition

- Sei Σ das Alphabet der zu speichernden Wörter inklusive $\perp$ für „leeres Zeichen“

- Dann ist ein Trie ein Baum
  - Jeder Knoten hat 0 bis maximal $|\Sigma|$ Kinder
  - Jede Kante ist durch genau einen Bezeichner aus Σ gekennzeichnet
  - Die Kanten zu den Kindern eines Knotens haben *unterschiedliche* Bezeichner
    - d.h. es gibt *keine* Kanten eines Knotens mit demselben Bezeichner
    - in der Regel werden die Kanten sortiert dargestellt und gespeichert
  - $\perp$ steht nur über einen Blattknoten und auch nur wenn der Elternknoten des Blattknotens weitere Kindsknoten besitzt

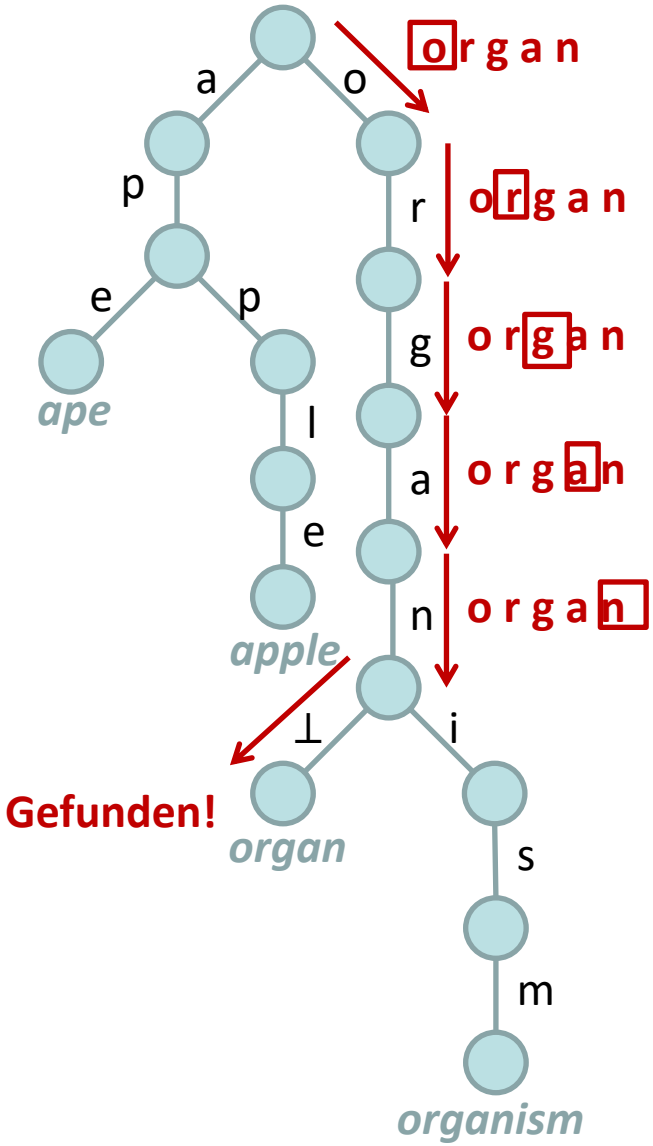
- Der Wert eines Blattknotens ergibt sich aus der Konkatenation der Kantenbezeichner von der Wurzel zum Blattknoten

Beispiel: Suche nach **ape**

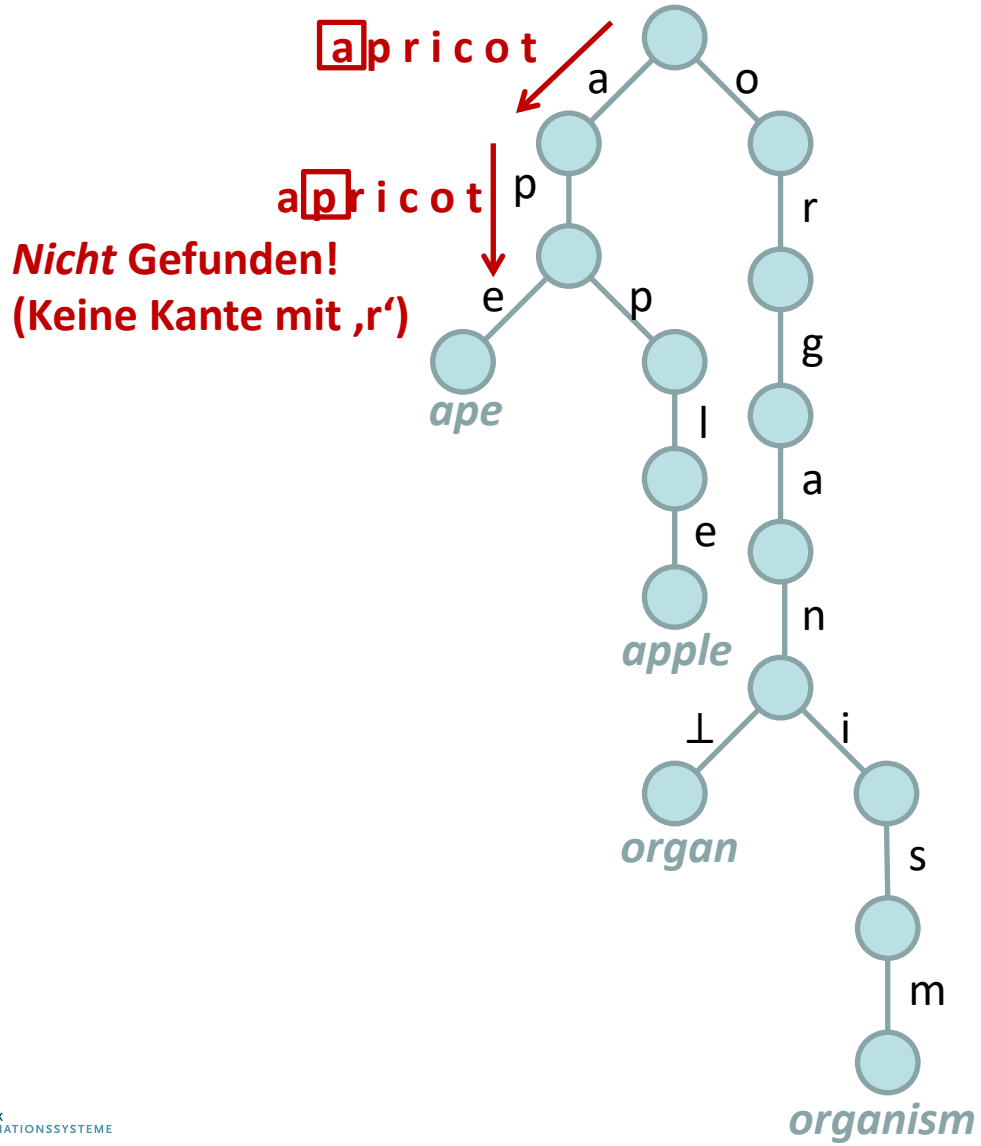


Gefunden!
(Blattknoten erreicht und Ende des zu suchendes Wortes erreicht)

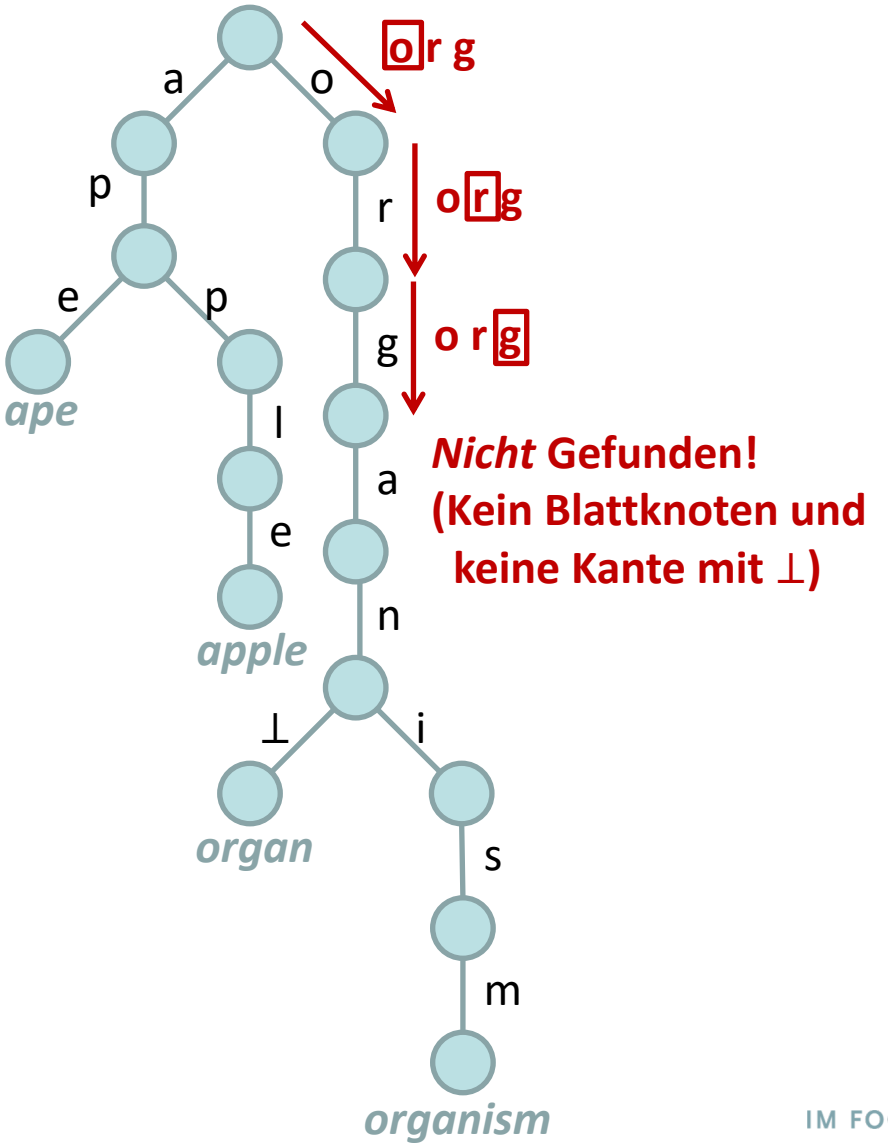
Beispiel: Suche nach **organ**



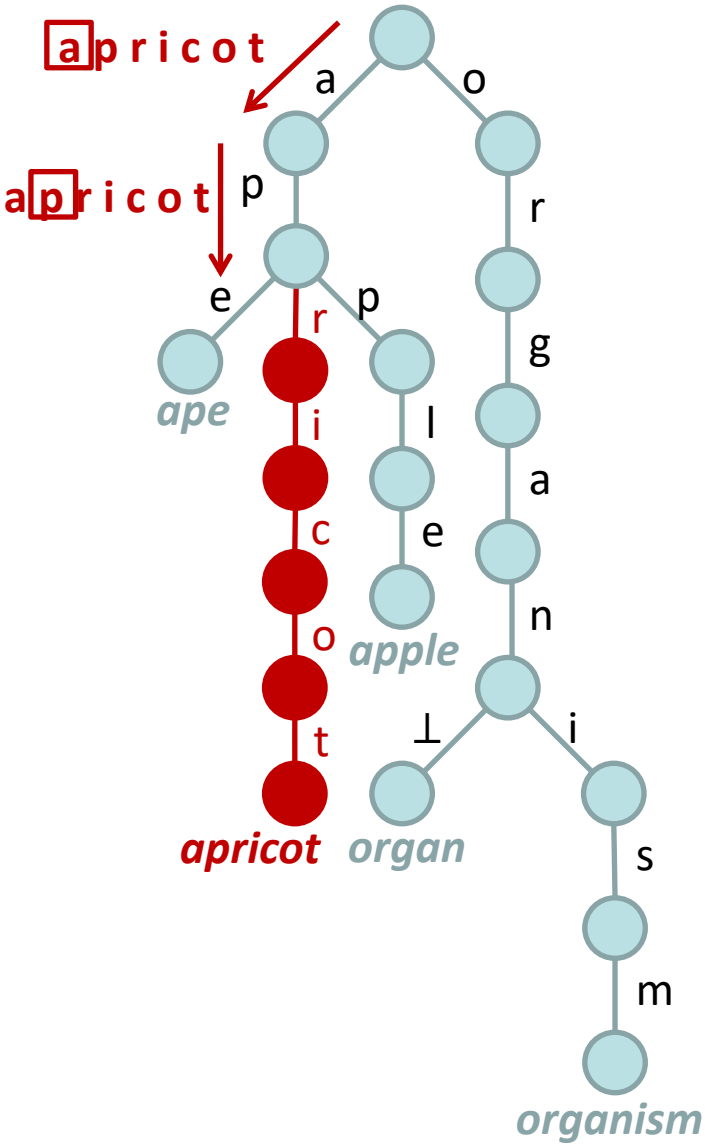
Beispiel: Suche nach **apricot**



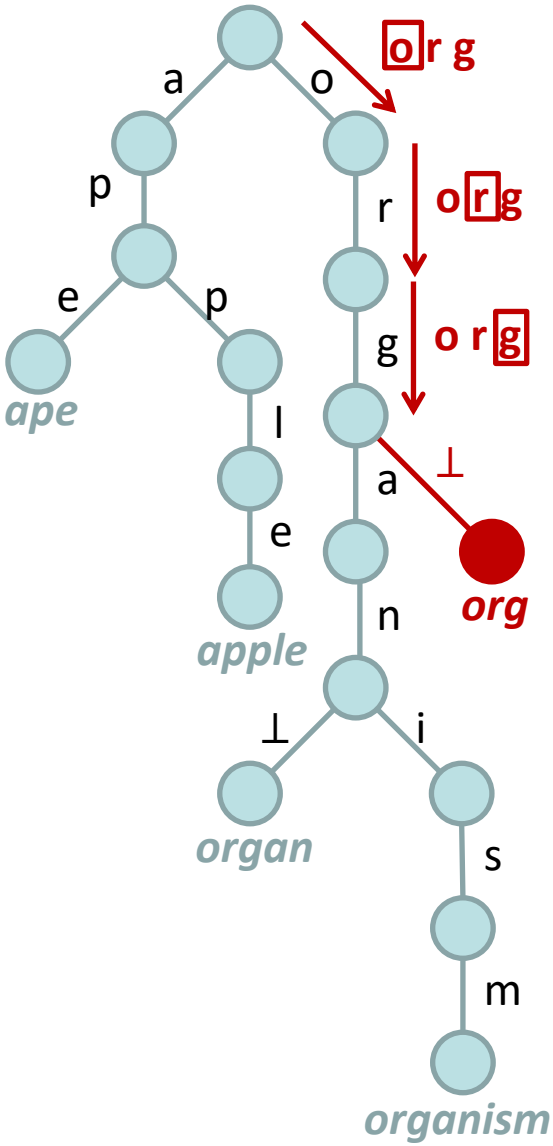
Beispiel: Suche nach **org**



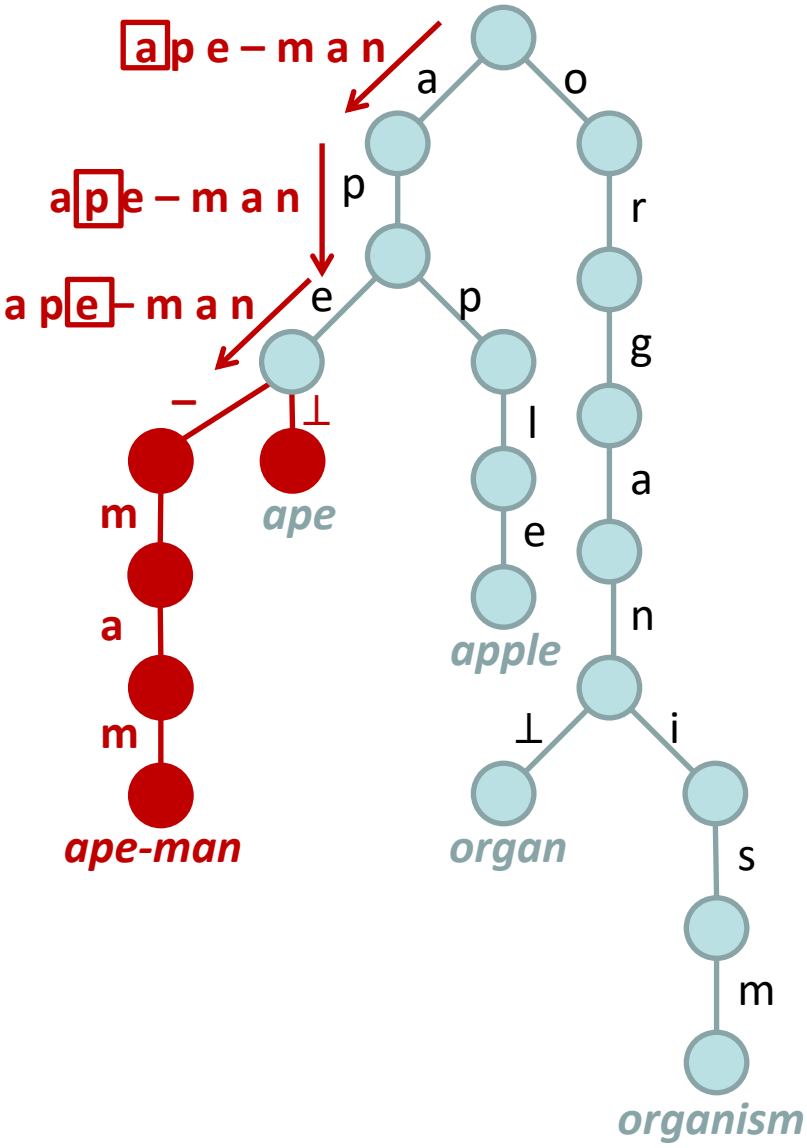
Einfügen: Beispiel **apricot**



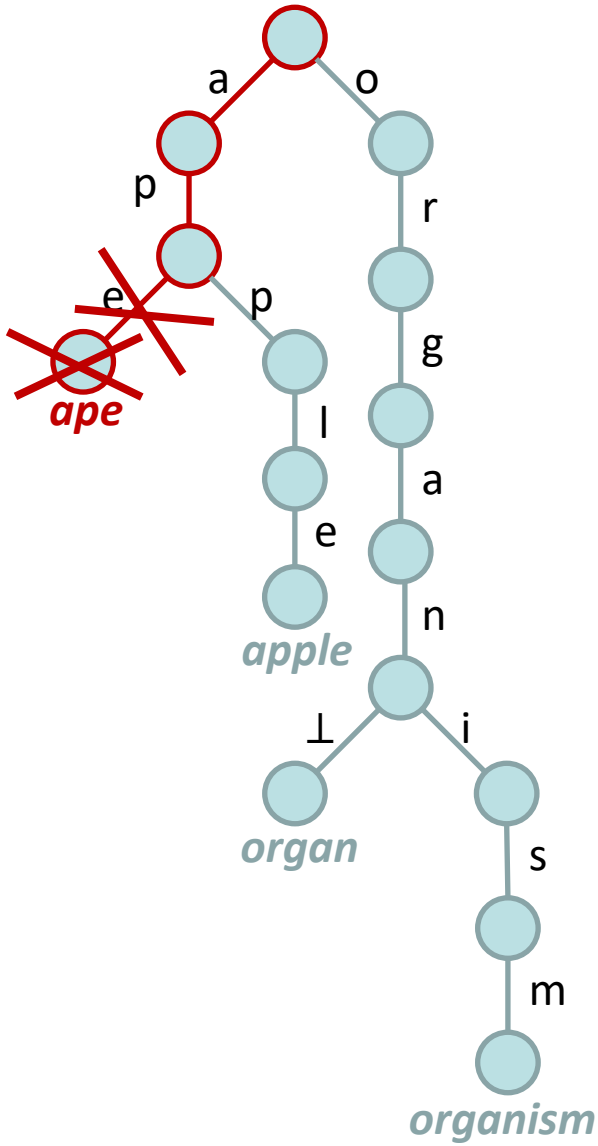
Beispiel: Einfügen von **org**



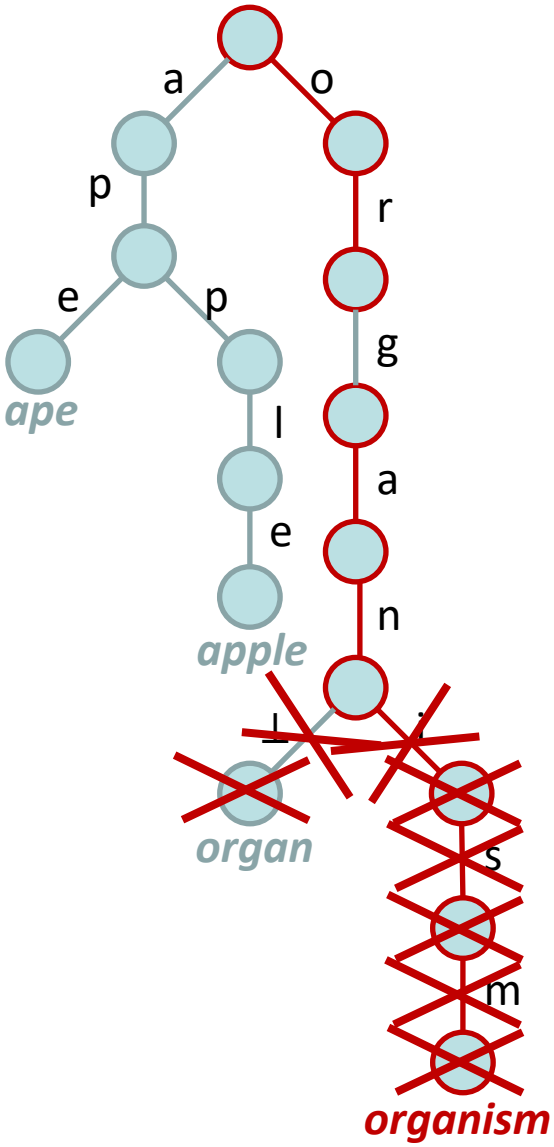
Beispiel: Einfügen von **ape-man**



Löschen von ***ape***

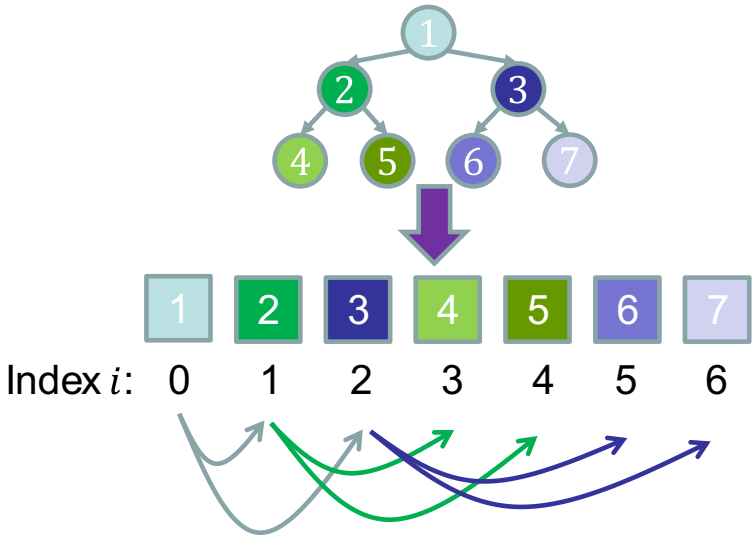


Löschen von ***organism***



Implementierungsmöglichkeiten

- Feld



- Zeigerstruktur
  - Kinder in Liste
  - Kinder in Feld
    - Größe Kinderanzahl
    - Größe $|\Sigma|$

Komplexität der Basisoperationen (Suchen, Einfügen, Löschen)

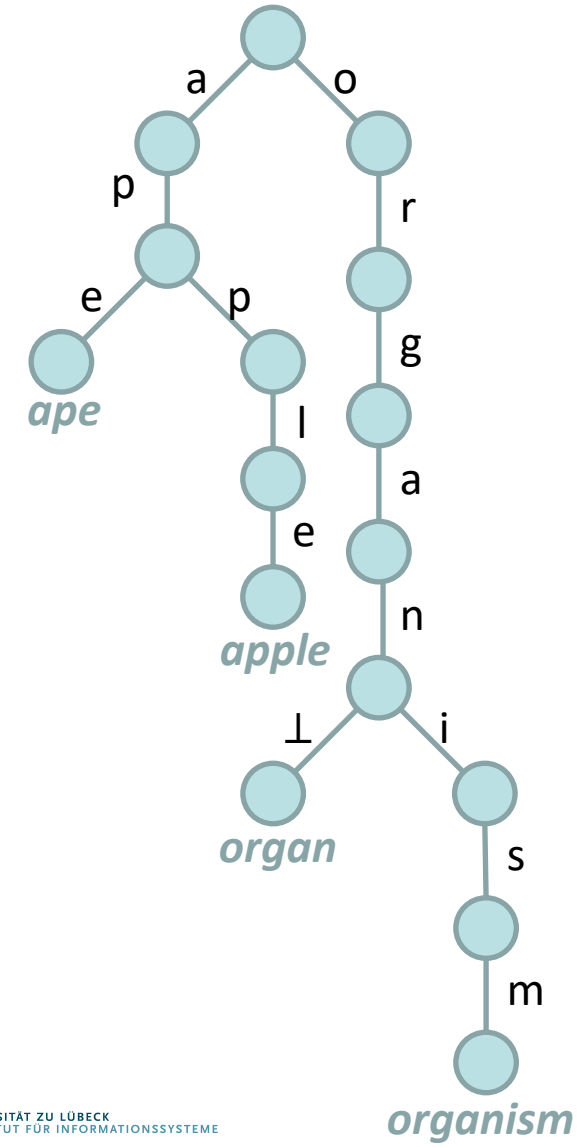
- Alle Basisoperationen hängen ab
  - von der Tiefe t des Tries
    - t ist gleich der (maximalen) Länge der eingefügten Wörter
  - sowie der Kosten f(|Σ|) pro Knoten für diese Operation
    - Abhängig von Speicherstruktur des Tries, siehe vorherige Folie O(t·f(|Σ|))
- Bei geeigneter Implementation oder kleinem |Σ|: O(t)

Patricia Tries

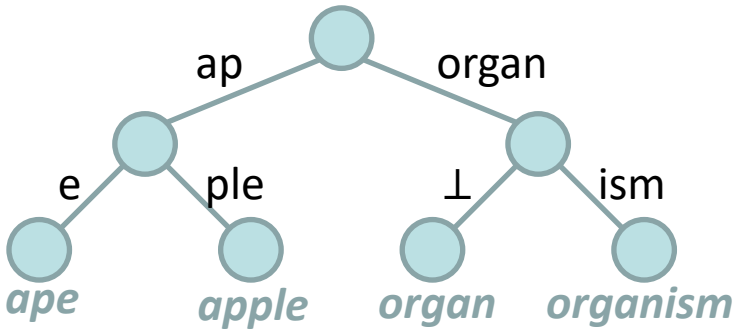
- In Tries haben viele Knoten nur 1 Kind und es bilden sich oft lange “Ketten” mit solchen Knoten
- **Idee:** Diese lange “Ketten” zusammenfassen
  - **Konsequenz:** Kanten sind nicht nur mit einem Zeichen, sondern mit Teilwörtern beschriftet

Donald R. Morrison, PATRICIA - Practical Algorithm to Retrieve Information Coded in Alphanumeric, *Journal of the ACM*, 15(4):514-534, October **1968**

Trie:



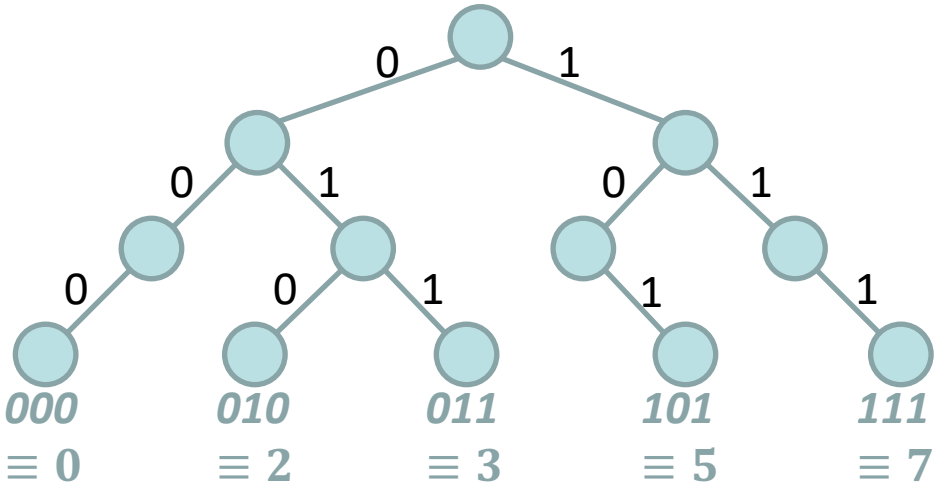
Patricia Trie:



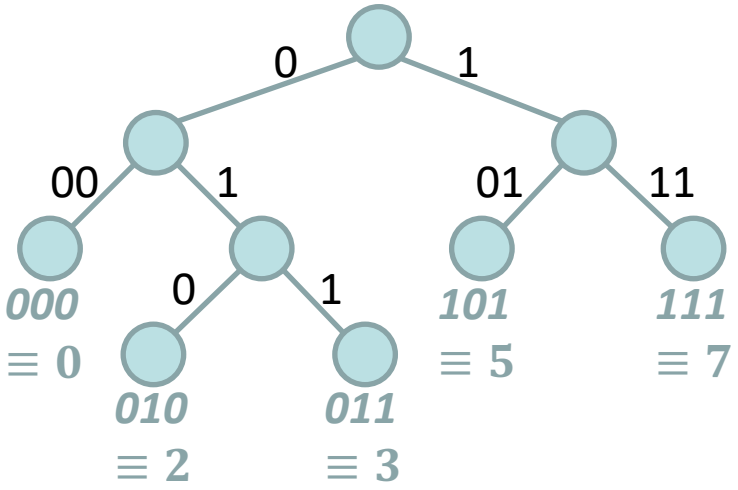
Bedingungen:

- Ausgehende Kanten eines Knotens starten *niemals* mit demselben Zeichen!
- Knoten haben 0 oder 2 bis $|\Sigma|$ viele Kinder, *niemals* hat ein Knoten 1 Kind!

Trie (binär):



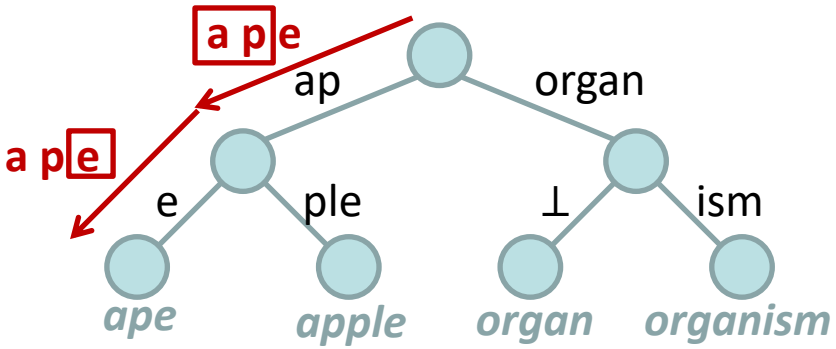
Patricia Trie:



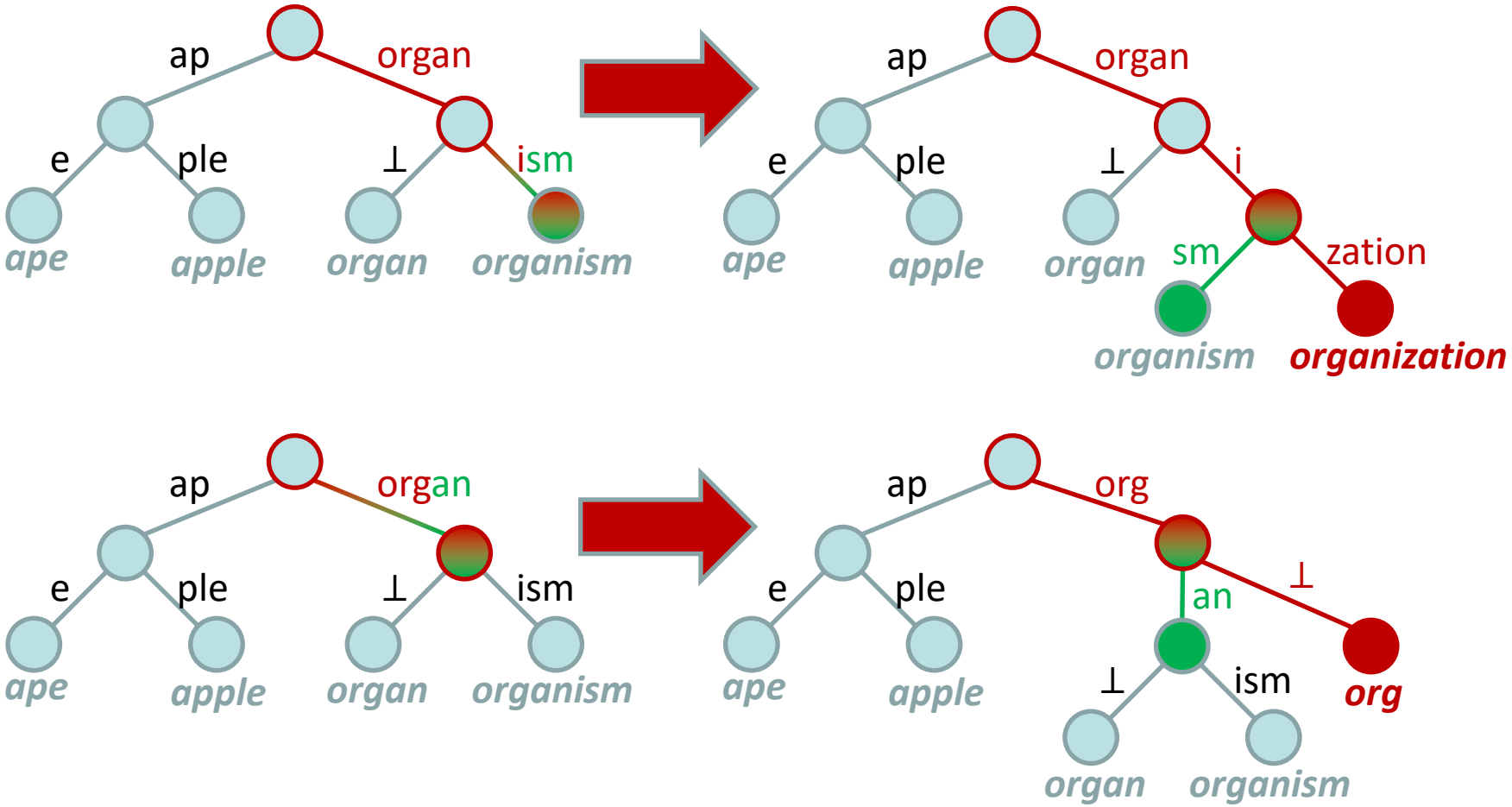
Konsequenzen für die Basisoperationen

- Suchen
  - Anstatt Vergleich von Zeichen nun Vergleich von Teilzeichenketten, sonst keine wesentliche Änderung gegenüber Suchen in Tries
- Einfügen
  - **Neuer Sonderfall:** Aufteilen eines Knotens, „falls Bezeichner der eingehenden Kante (echte) Teilzeichenkette des einzufügenden Wortes ist“
- Löschen
  - **Neuer Sonderfall:** Nach Löschen überprüfen, ob Knoten vereinigt werden können

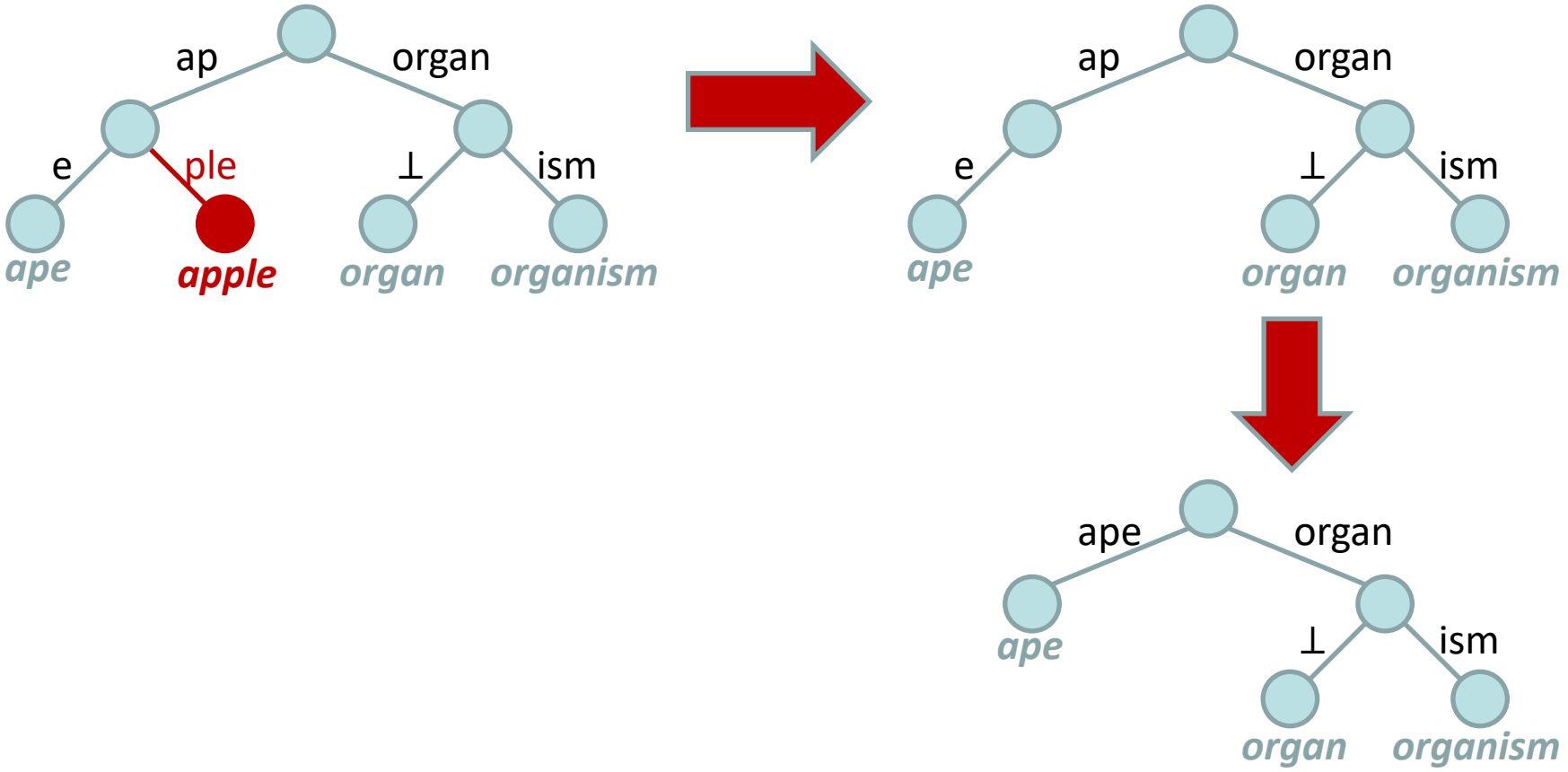
Beispiel Suchen nach ***ape***



Sonderfall beim Einfügen von ***organization*** bzw. ***org***



Beispiel für neuen Sonderfall beim Löschen von *apple*



Komplexität

- Ergebnisse für Basisoperationen korrelieren asymptotisch mit denen der Tries
  - D.h. bei geeigneter Implementierung $O(t)$ mit t Tiefe des Patricia Tries, welches mit der maximalen Länge der eingefügten Wörter korreliert
- Da Patricia Tries i.d.R. weniger Knoten haben, sind in der Praxis
  - Patricia Tries *schneller* und
  - haben *geringeren Speicherplatzverbrauch*

Zusammenfassung

- Trie
  - n-äres Alphabet
    - Speichern von Zeichenketten
  - binäres Alphabet
    - Speichern von z.B. Zahlen
- Patricia Trie
  - kompaktere Darstellung
  - n-äres Alphabet
  - binäres Alphabet