Zusammenfassung zur Vorlesung Basismodul Computerlinguistik

Daciuk-Algorithmus

Konstruktion eines minimalen Trie anhand einer sortierten Wortliste

25.11.2021

Inhaltsverzeichnis

Minimierung eines vorhandenen Tries

Ausgang: Lexikon

Trie (Präfixbaum, Lexikonautomat): ein azyklischer DEA

- geeignet für die Darstellung und Speicherung von Lexika

Wann sind 2 Zustände äquivalent?

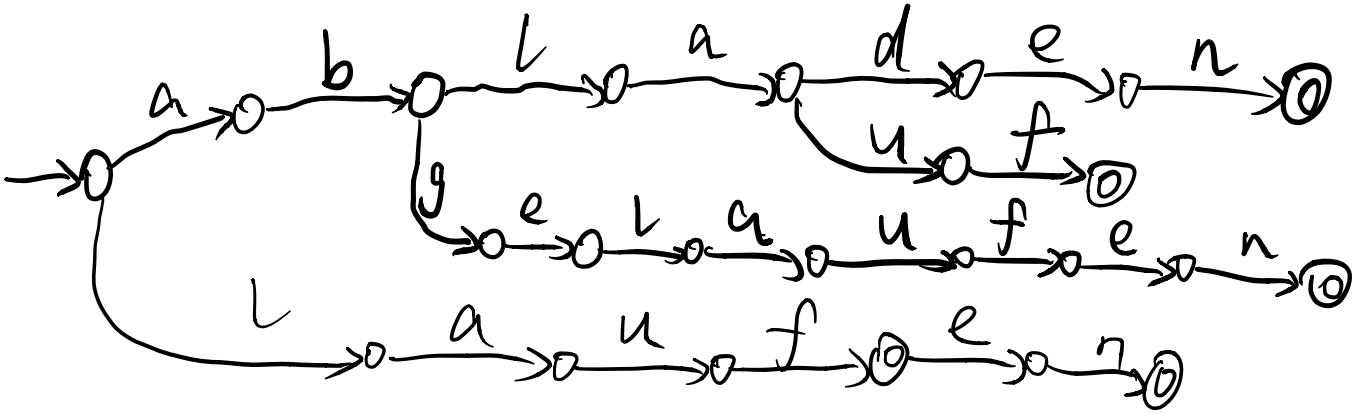
Wenn sie

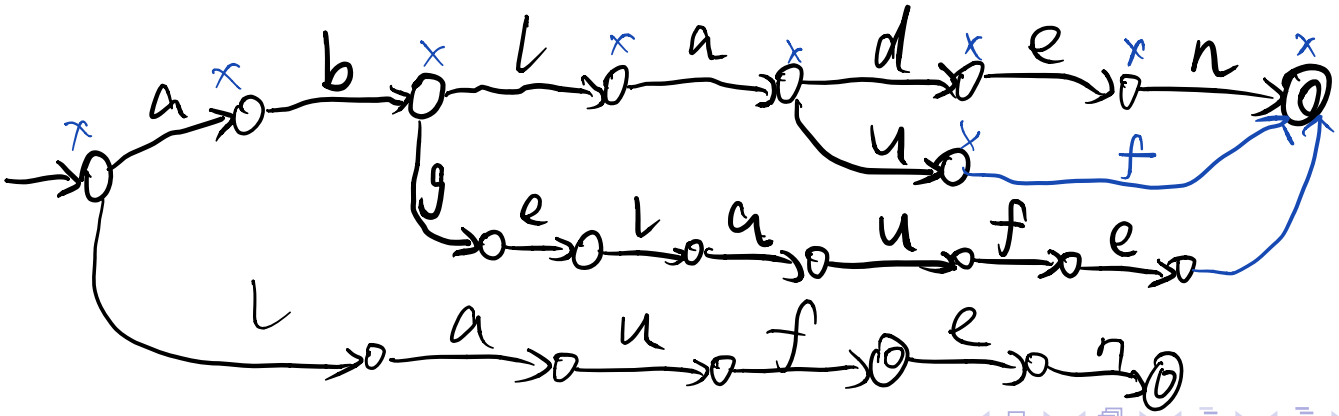
- beide final oder nicht-final sind
- die gleichen ausgehenden Übergänge haben (Anzahl, label)
- zu äquivalenten Zuständen gehen (für die wiederum die obigen Bedingungen gelten).

Achtung: Die Konstruktion eines minimierten Tries muss anhand einer **sortierten** Wortliste durchgeführt werden.

Beispiel

Gegeben sei ein Trie für das Lexikon mit den sortieren Wörtern **abladen, ablauf, abgelaufen, lauf, laufen**, berechne den minimalen DEA des Tries.





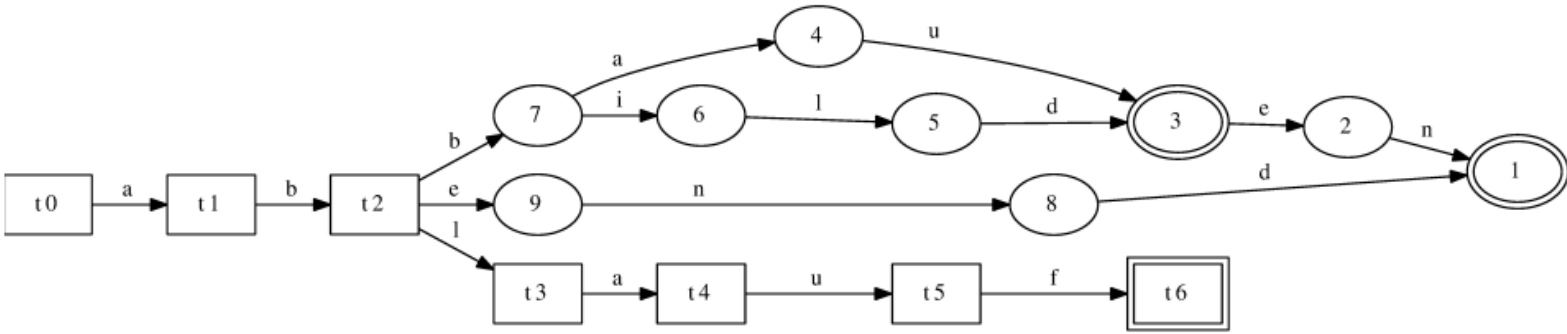
Daciuk-Algorithmus

Ausgangspunkt: **Lexikon**

Daciuk-Algorithmus

Online-Minimierung: Vereinigung von Trie-Erstellung und Minimierung.

- Nur nicht mehr veränderte Zustände werden gespeichert, d.h., solche Zustände können keine neuen Übergänge mehr erhalten.
- Eine **sortierte** Liste muss geliefert werden.
- Temporäre Zustände: sich zum späterern Zeitpunkt noch möglich ändern können, stets die Zustände, die den Pfad für das letzt gelesene Wort repräsentieren.



Pseudocode

Pseudocode für Daciuk-Algorithmu zur Online-Minimierung

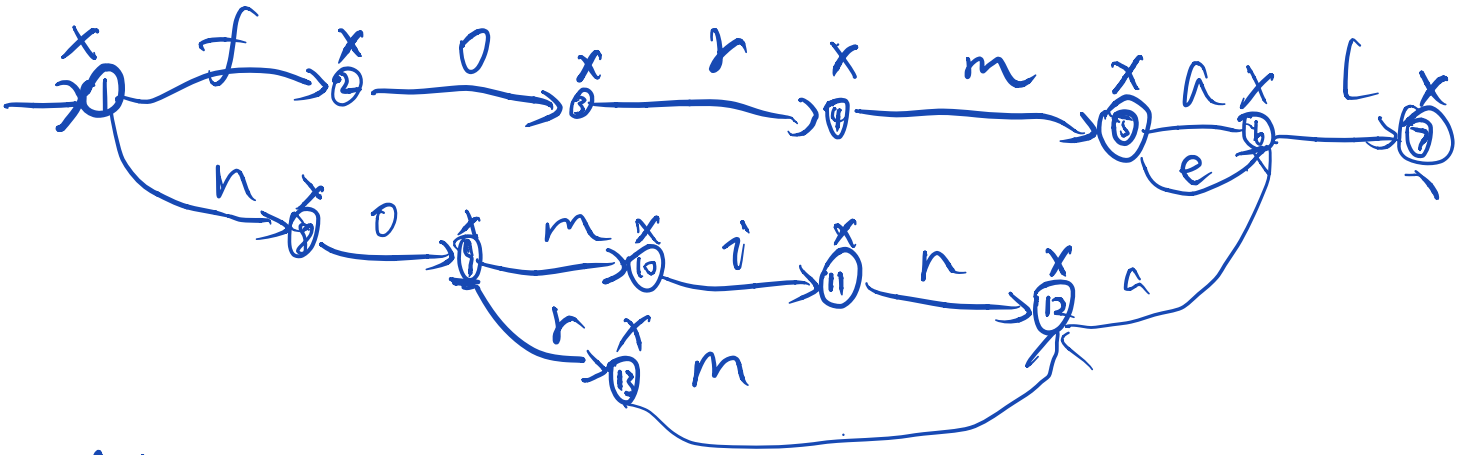
```
1  Register:= {}
2  while there is another word {
3     Word:= next word in lexicographic order;
4     CommonPrefix:= common_prefix(Word);
5     SplitState:=delta_string(q0,CommonPrefix);
6     CurrentSuffix:=
7        Word[length(CommonPrefix)...length(Word)-1];
8     if( has_children(SplitState) ){
9        replace_or_register(SplitState);
10    }
11    add_suffix(SplitState,CurrentSuffix);
12 }
13 replace_or_register(q0);
14
15 replace_or_register(State) {
16    Child:=last_child(State);
17    if( has_children(Child) ) {
18       replace_or_register(Child);
19    }
20    if(exists q in Register and q equals Child ) {
21       last_child(State):=q: (q in Register and q equals Child);
22       delete(Child);
23    } else {
24       Register := Register + {Child};
25    }
26 }
```

[1]

[1] Perez. *Skript zum Kurs Finite-State-Technologien beim Indizieren und Suchen.*

Beispiel

Gegeben sei das Lexikon mit den sortieren Wörtern **form, formal, formel, nominal, normal**. Berechne den minimalen DEA für das Lexikon nach dem Daciuk-Algorithmus.



Implementierung des Daciuk-Algorithmus

https://www.cip.ifi.lmu.de/~nie/p1/06_daciuk.html
https://www.cip.ifi.lmu.de/~nie/p1/06_daciuk_oop.html