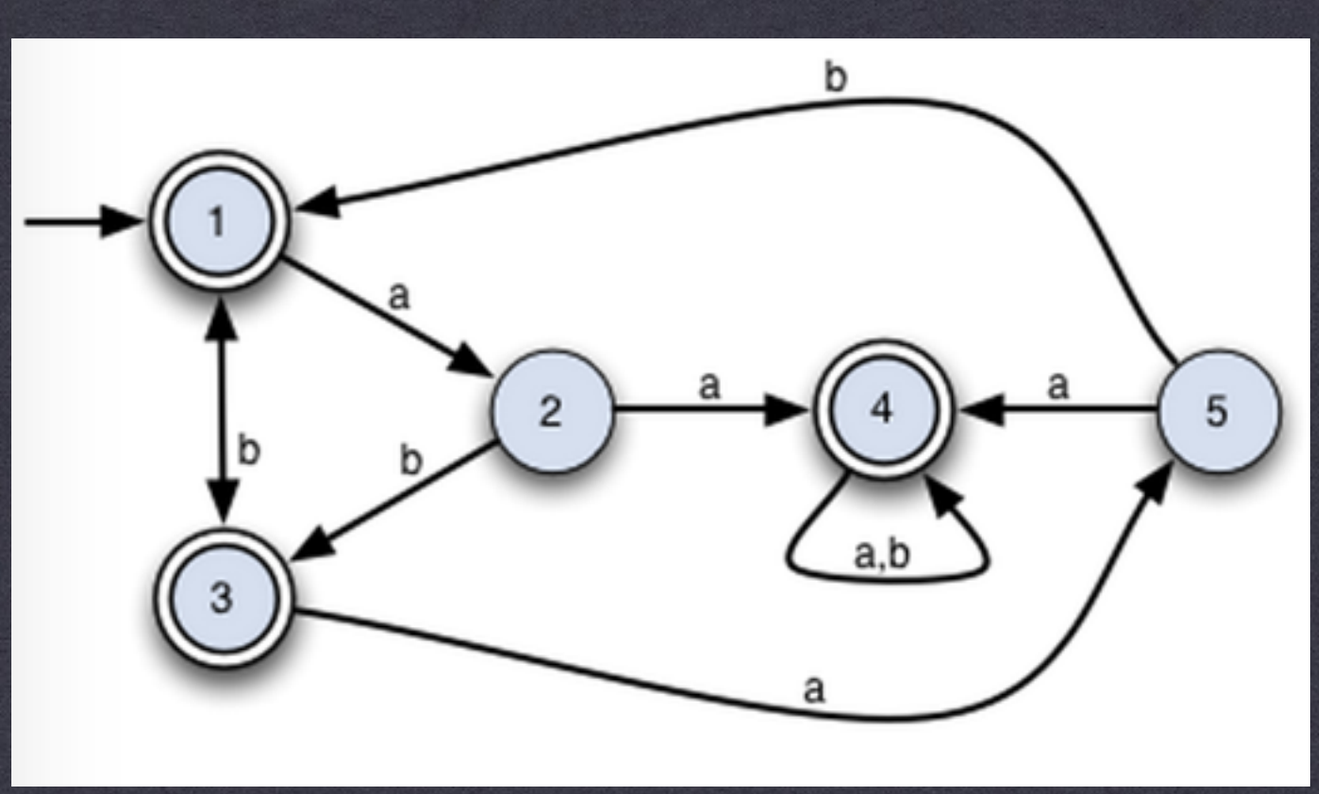
VORTRAG 09.03.2015

BY LOUIS RETTER

Der Hopcroft Minimierungsalgorithmus



Einleitung

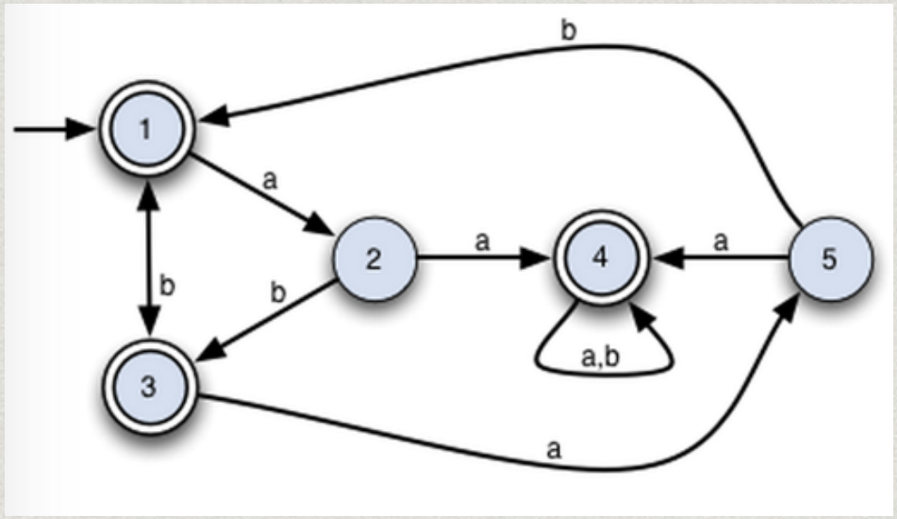
Im Vortrag geht es darum einen Algorithmus beschreiben um einen deterministischen endlichen Automaten in O(n*log(n)*k) zu minimieren, wobei n die Anzahl an Zuständen und k die Größe des Alphabets ist.

Inhaltsverzeichnis

Wichtige Begriffe

- DEA (deterministischer endlicher Automat)
- äquivalente Zustände
- Minimierung eines DEA
- O-Notation

DEA



- A = (∑, Q, f, q0, F)
  - ∑ : eine endliche Menge an Eingabesymbolen
  - Q : eine endliche Menge an Zuständen
  - f : die Transitionsfunktion
  - q0: der Anfangszustand
  - F: endliche Menge an finalen Zuständen

- Akzeptanz des Automaten :
  - f(q0, w) = q, mit q ∈ F

äquivalente Zustände

- Zwei Zustände sind äquivalent, falls für jedes Eingabewort w = w_0…w_n beide in einem Endzustand oder nicht enden.

- q1 = q2, mit f(q1, w) = q1’ und f(q2, w) = q2’ :
  falls q1’ ∈ F <=> q2’ ∈ F,
  falls q1’ !∈ F <=> q2’ !∈ F.

Minimieren

- DEA so verändern dass man mit der kleinstmöglichen Anzahl an Zuständen auskommt
- akzeptiert die selbe Sprache
- minimaler Automat ist wohldefiniert
- Äquivalenz zweier Automaten testen
- Idee : fast alle äquivalente Zustände zusammen

O-Notation

- Im weiteren Verlauf des Vortrags wird die O-Notation benutzt um das Laufzeitverhalten eines Algorithmus zu beschreiben.
- Hier bezeichnet K die Anzahl an Elementen im Alphabet und n die Anzahl an Zuständen.

Was hat die Minimierung eines Automaten für einen Zweck ?

- weniger Speicher verbrauch
- übersichtlicher Automat

  => das Minimieren hat keinen Nutzen für das Lösen des Wortproblems für Regulare Sprachen !

Nutzen des von John Hopcroft eingeführten Algorithmus

- Verbesserte Laufzeit gegenüber herkömmlichen Algorithmen :
  - O(n*log(n)*k) im Gegensatz zu O(k*n^2)

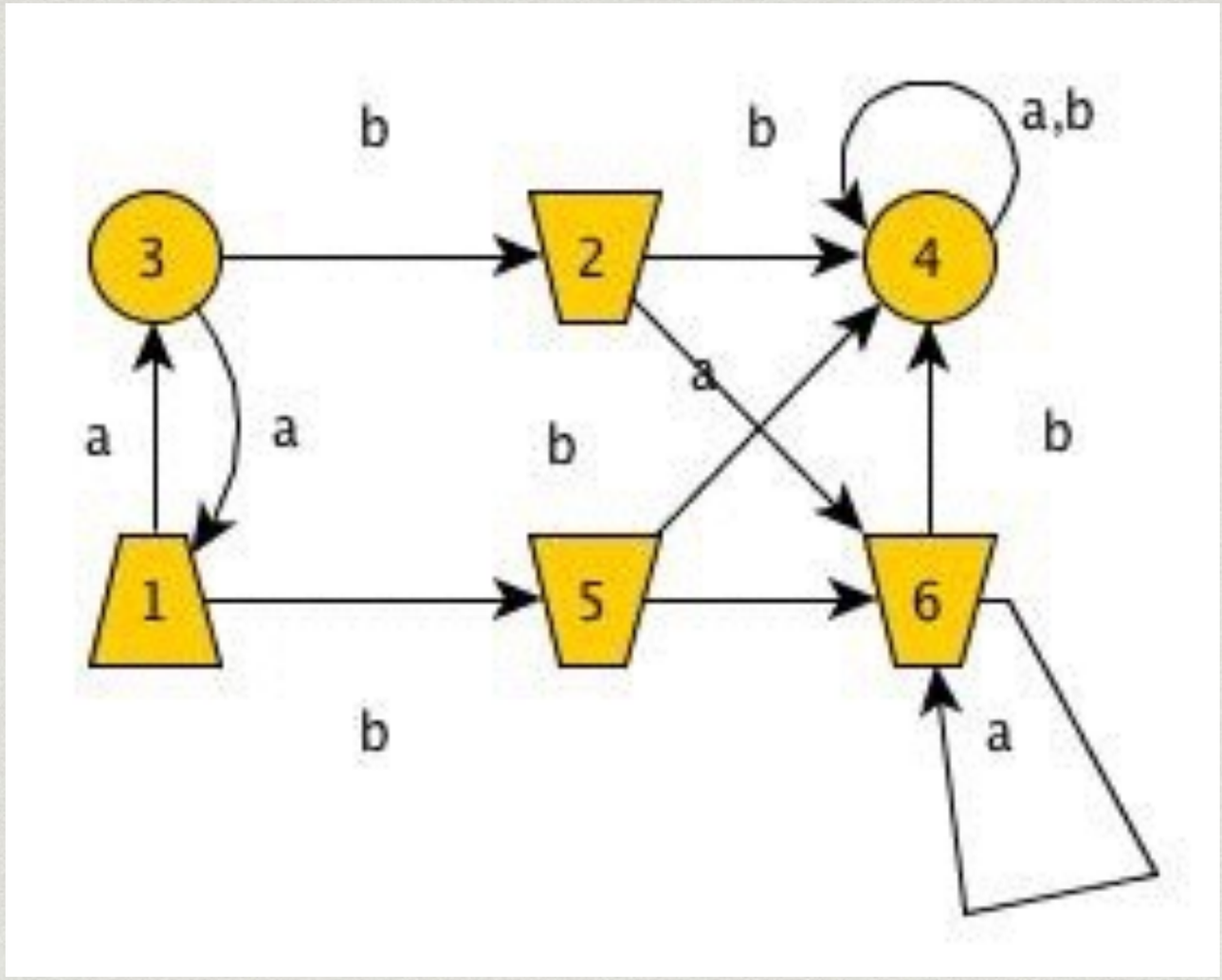
Funktionsweise des Algorithmus

- Formale Definition des Algorithmus :

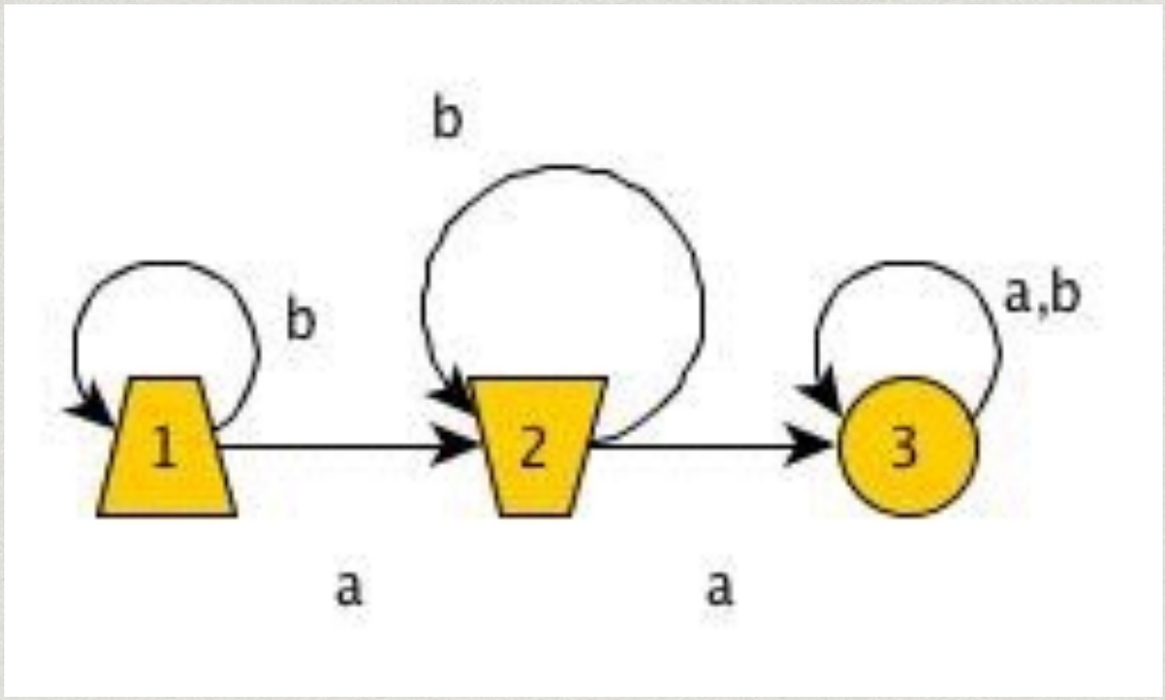
Aufteilen der Zustände in zwei Anfangs Mengen :
P := {F, Q \ F};
W := {F};

- Wähle eine Menge A aus der Menge W, für jedes e Symbol aus ∑ erstelle die Menge X, wobei es eine Transition von einem Element aus X nach W gibt.
- Ersetze jede Menge Y aus P durch die Mengen (X ∩ Y) und (Y \ X), falls diese nicht leer sind.
- Nun ersetze Y in W durch die kleinere Menge (X ∩ Y) oder (Y \ X), falls Y nicht in W, sonst ersetze Y in W durch (X ∩ Y) und (Y \ X).
- Diese letzten 3 Schritte muss man jetzt so lange wiederholen bis W leer ist.
- Man kann nun alle Zustände in den Teilmengen von P zu neuen Zuständen zusammenfügen.
  z.b : P = { {q0, q1, q2}, {q3, q4}, { qf}} => {q0', q1', q2'}

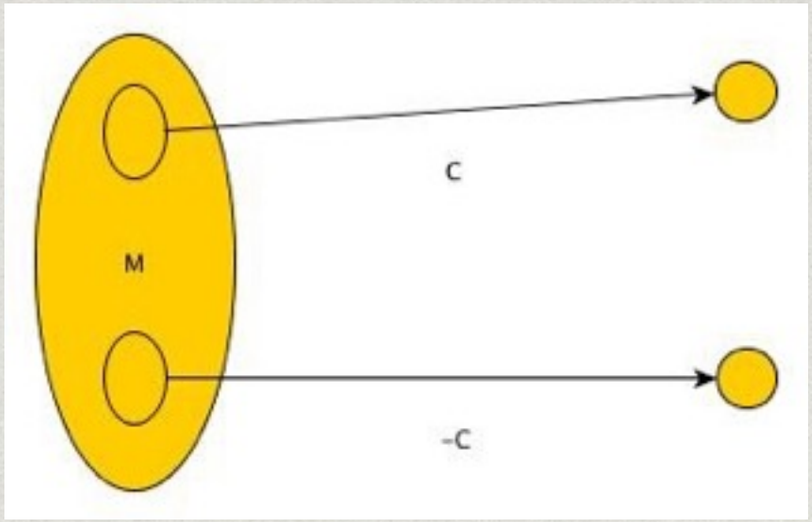
Beispiel



Minimaler Automat :



Korrektheit



- Um die Korrektheit des Algorithmus zu beweisen muss man zwei Teilprobleme beweisen :
  “=>” zwei nicht äquivalente Zustände sind am Ende des Algorithmus in verschiedenen Mengen.

  “<=“ zwei äquivalente Zustände sind in der selben Menge.

Laufzeit

- Woher kommt die Laufzeit O(n* log(n)*k) ?
- Sobald man eine Menge für ein Eingabesymbol geteilt hat muss man die neu entstanden Mengen nicht mehr auf dieses Symbol testen
- Man arbeitet nur mit einer der Beiden entstandenen Teilmengen weiter

Markierungsalgorithmus

- erstelle eine Tabelle der Größe n^2
- markiere genau die Zustandspaare bei denen einer ein Endzustand ist
- markiere nun die Zustandspaare (q1, q2) für die gilt, (f(q1, a), f(q2, a)) ist schon markiert, wobei a ein Eingabesymbol ist.
- wiederhole dies bis keine Veränderung mehr stattfindet
- füge die nicht markierten Zustände zusammen

Vergleich der beiden Vorgestellten Algorithmen

	Hopcroft	Mark-Algo
Laufzeit	O(n*log(n)*k)	O(n^2)
Speicherverbrauch	O(n)	O(n^2)

Woher kommt dieser Unterschied in der Laufzeit ?

References :

- http://en.wikipedia.org/wiki/DFA_minimization
- “An n log n algorithm for minimizing states in a finite automaton”, by John Hopcroft.

VIELEN DANK

by Louis Retter