Automatensynthese

Automatensynthese

Minimales Ziel:

Wir wollen einen Becher mit Kaffee aus dem Automaten erhalten.

Maximales Ziel:

Wir wollen verstehen, wie der Automat funktioniert, um planmäßig (und beliebig oft) aus dem Automaten einen Becher mit Kaffee erhalten zu können.

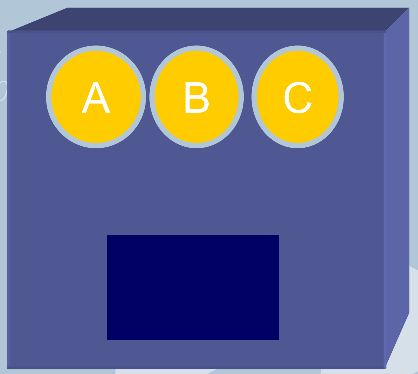
Kann man die Funktionsweise des Automaten erlernen?

Automatensynthese

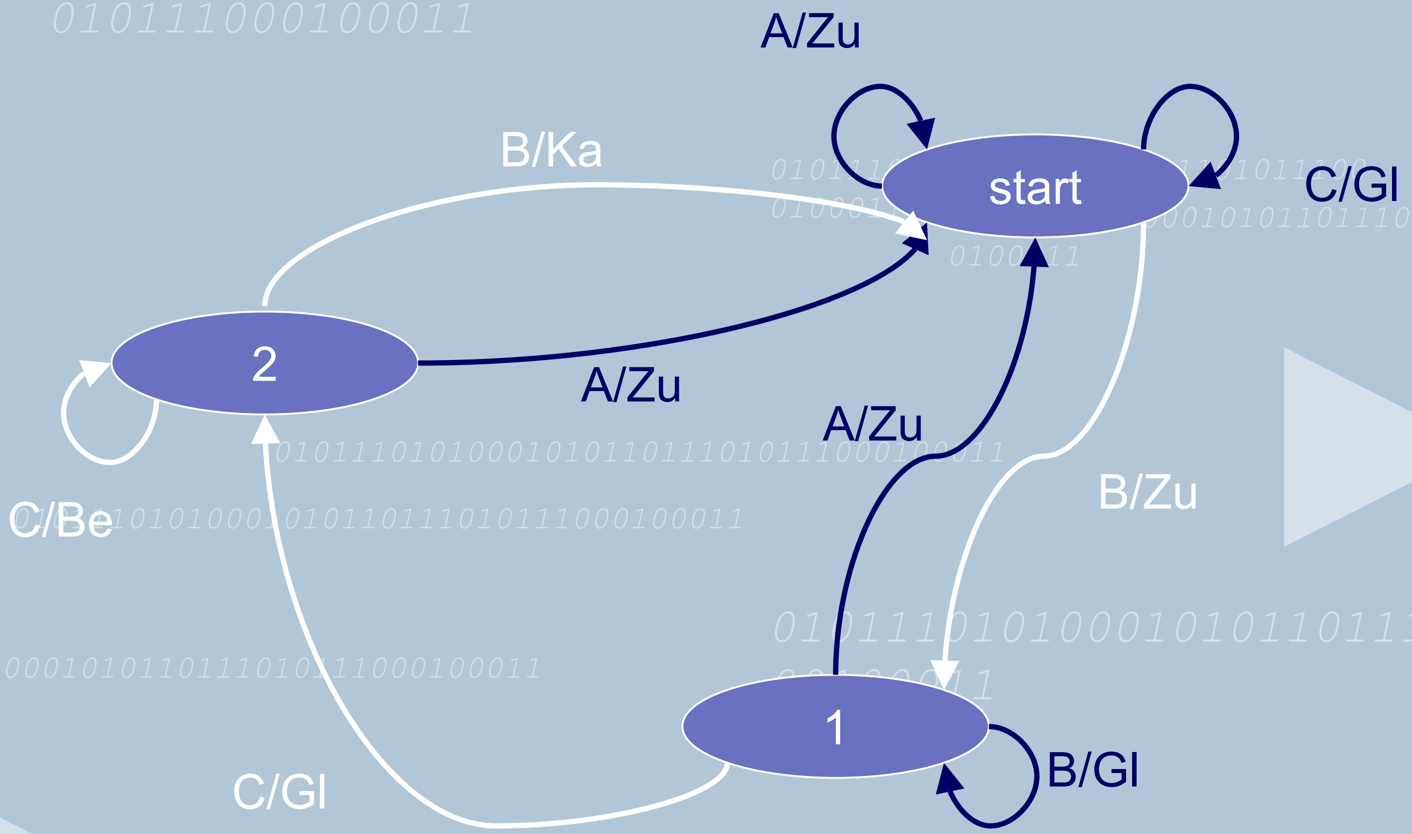
Es gibt 3 Knöpfe: A, B und C.

Wir sind 4 Ausgaben des Systems möglich:

Ertönen einer Glocke	Gl
Becher	Be
Zucker	Zu
Kaffee	Ka



Automatensynthese



Automatensynthese

Klasse zu lernender Objekte

endlich deterministische Automaten A (mit Ausgabe)

Informationen

über das Verhalten eines unbekannten Automaten A werden Interaktionsfolgen vorgelegt

Lernziel

es soll ein Automat B gelernt werden, der sich genauso wie A verhält

Automatensynthese

Vorgehensweise:

- Minimierung von DFA
    - mittels Äquivalenzklassenbildung
- Leichte Änderung am Äquivalenzbegriff
- Minimierungsalgorithmus für DFA liefert einen Lernalgorithmus für DFA aus deren Verhalten

Automatensynthese

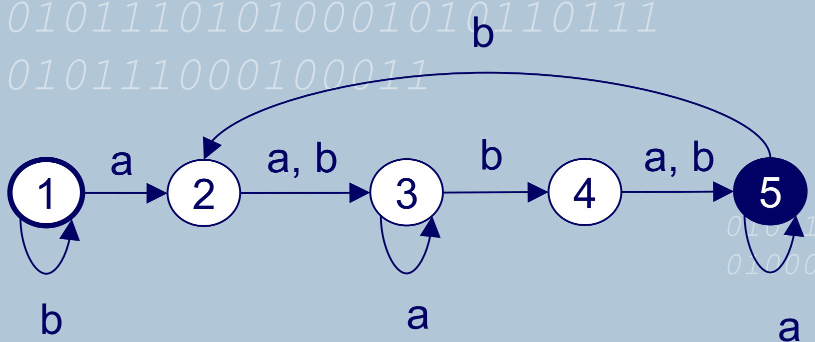
Zur Erinnerung: endl. det. Automaten (ohne Ausgabe)

$A = [\, X, Z, \delta, \{z_0\}, F \,]$

- Eingabealphabet X
- Zustandsmenge Z
- Zustandsüberführungsfunktion $\delta: Z \times X \rightarrow Z$
- Anfangszustand $z_0 \in Z$
- Endzustände $F \subseteq Z$

Wie in Schöning / GdI IV eingeführt

Automatensynthese



- $X = \{ a, b \}$
- $Z = \{ 1, 2, 3, 4, 5 \}$
- $z_0 = 1$
- $F = \{ 5 \}$

	a	b
1	2	1
2	3	3
3	3	4
4	5	5
5	5	2

Automatensynthese

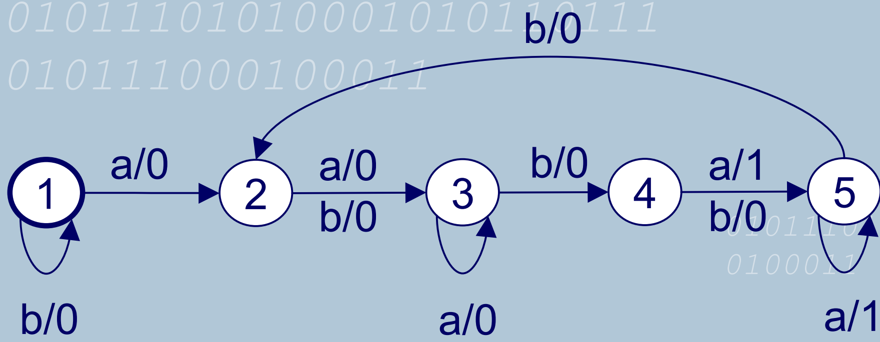
End. det. Automaten (mit Ausgabe)

$A = [\, X, Y, Z, \delta, \lambda, \{z_0\} \,]$

- Eingabe-/Ausgabealphabet X, Y
- Zustandsmenge Z
- Zustandsüberführungsfunktion $\delta: Z \times X \to Z$
- **Ausgabefunktion $\lambda: Z \times X \to Y$**
- Anfangszustand $z_0 \in Z$

Moore/Mealy-Automat

Automatensynthese



- $X = \{ a, b \}$
- $Y = \{ 0, 1 \}$
- $Z = \{ 1, 2, 3, 4, 5 \}$
- $z_0 = 1$

	a	b
1	2/0	1/0
2	3/0	3/0
3	3/0	4/0
4	5/1	5/0
5	5/1	2/0

... beschreibt dasselbe Akzeptierungsverhalten; letztes Bit der Ausgabe ist relevant

Automatensynthese

End. det. Automaten (mit Ausgabe)

$A = [\, X, Y, Z, \delta, \lambda, \{z_0\} \,]$

- Eingabe-/Ausgabealphabet X, Y
- Zustandsmenge Z
- Zustandsüberführungsfunktion $\delta: Z \times X \rightarrow Z$
- Ausgabefunktion $\lambda: Z \times X \rightarrow Y$
- Anfangszustand $z_0 \in Z$

Verhalten von A: $V(A) = \{ (w, \lambda^*(z_0,w)) \mid w \in X^+ \}$

Automatensynthese

Fortsetzung von δ und λ

sei $A = [\, X, Y, Z, \delta, \lambda, \{z_0\} \,]$

$\lambda^*(z_0,x) = \lambda(z_0,x)$, falls $x \in X$
$\lambda^*(z_0,wx) = \lambda^*(z_0,w)\ \lambda(\delta^*(z_0,w),x)$, falls $w \in X^+$, $x \in X$

$\delta^*(z_0,x) = \delta(z_0,x)$, falls $x \in X$
$\delta^*(z_0,wx) = \delta(\delta^*(z_0,w),x)$, falls $w \in X^+$, $x \in X$

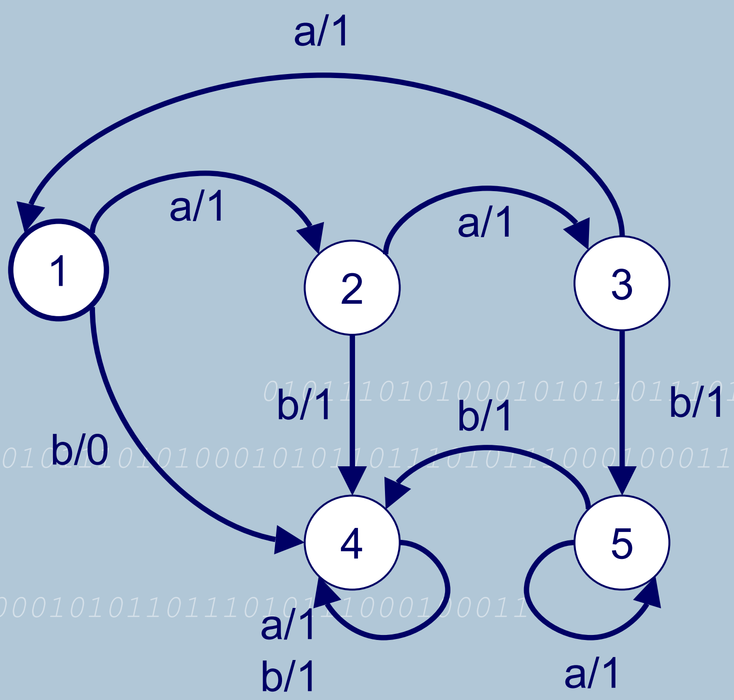
Automatensynthese

Grundlagen

- Es sei A ein endl. det. Automat. Dann gibt es einen (/* bis auf Isomorphie */) eindeutig bestimmten minimalen endl. det. Automaten A’ mit V(A’) = V(A).
- Es gibt einen effizienten Algorithmus zur Bestimmung von A’.

Automatensynthese

Ist folgender endl. det. Automat A minimal?



minimal, falls gilt:

es gibt keinen äquivalenten Automaten endl. det. Automat B mit weniger Zuständen

Automatensynthese

es sei $A = [X, Y, Z, \delta, \lambda, \{z_0\}]$

- zwei Zustände z, z' heißen äquivalent, falls für alle $w \in X^+$ gilt: $\lambda^*(z,w) = \lambda^*(z',w)$
- zwei Zustände z, z' heißen k-äquivalent, falls für alle $w \in X^+$ mit $|w| = k$ gilt: $\lambda^*(z,w) = \lambda^*(z',w)$

Beobachtung:

Zustände z und z' sind k+1-äquivalent
gdw.
es gilt für alle $x \in X$:

- $\lambda(z,x) = \lambda(z',x)$
- die Zustände $\delta(z,x)$ und $\delta(z',x)$ sind k-äquivalent

Automatensynthese

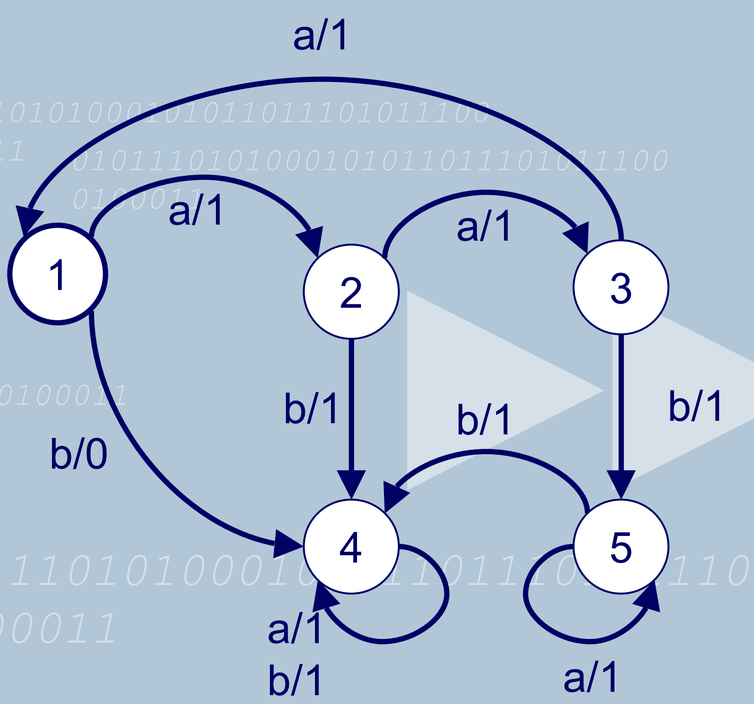
Minimierungsalgorithmus

1-äquivalent: { 1 } { 2, 3, 4, 5}

2-äquivalent: { 1 } { 2, 4, 5 } { 3 }

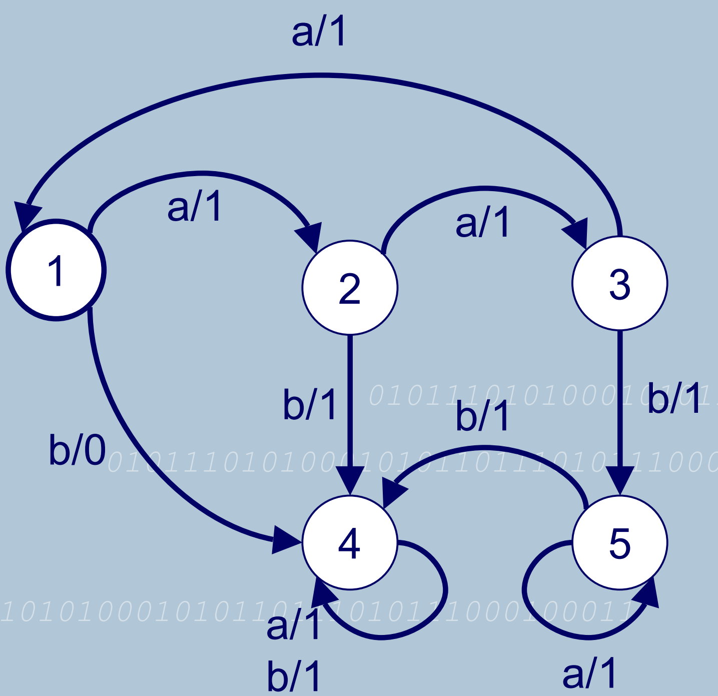
3-äquivalent: { 1 } { 2 } { 4, 5 } { 3 }

4-äquivalent: { 1 } { 2 } { 4, 5 } { 3 }

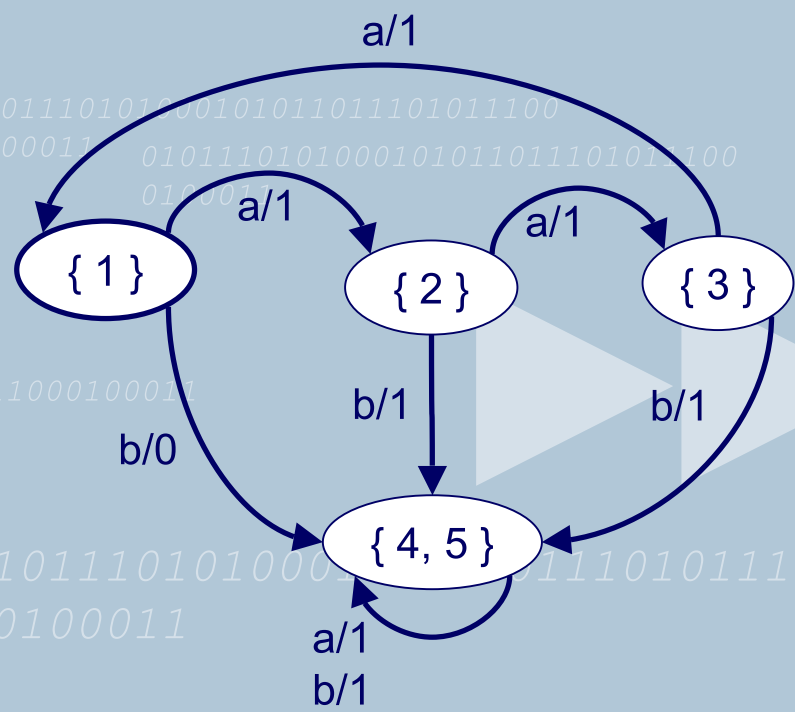


geht in Zeit $O(mn^2)$

Automatensynthese



zugehöriger minimaler Automat



äquivalente Zustände: { 1 } { 2 } { 4, 5 } { 3 }

Automatensynthese

Klasse zu lernender Objekte

endlich deterministische Automaten A (mit Ausgabe)

Informationen

über das Verhalten eines unbekannten Automaten A werden Interaktionsfolgen vorgelegt

Lernziel

es soll ein Automat B gelernt werden, der sich genauso wie A verhält

Automatensynthese

Interaktionsfolge für A

unendliche Folge $\{V_i\}$ endlicher Mengen, so daß für alle i gilt:

- $V_i \subseteq V(A)$
- V_i ist anfangsstückvollständig (/* d.h. falls $(u,\lambda^*(u)) \in V_i$, so $(u',\lambda^*(u')) \in V_i$ für alle Anfangsstücke u' von u */)
- $V_i \subseteq V_{i+1}$
- $V_1 \cup V_2 \cup \ldots = V(A)$

Automatensynthese

Lernverfahren (Idee)

wir betrachten V(A) als Automaten (/* V(A) ist äquivalent zu A */)

- V(A) ist unendlich
- V(A) ist zu jedem endlichen Zeitpunkt nur unvollständig beschrieben

<u>Ansatz:</u> Wir minimieren V(A).

<u>Beobachtung:</u>

Das Ergebnis ist ein Automat B mit den Eigenschaften:

- B ist äquivalent zu V(A) (und damit auch zu A).
- B ist minimal.

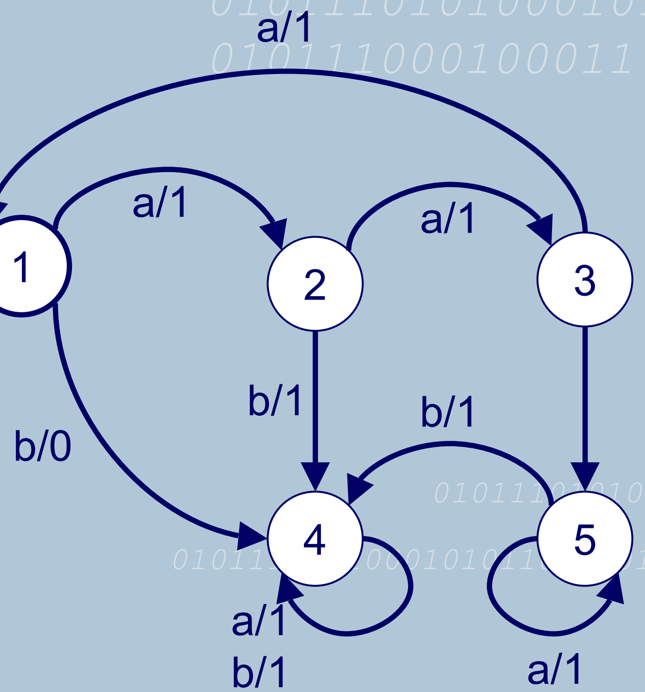
Automatensynthese

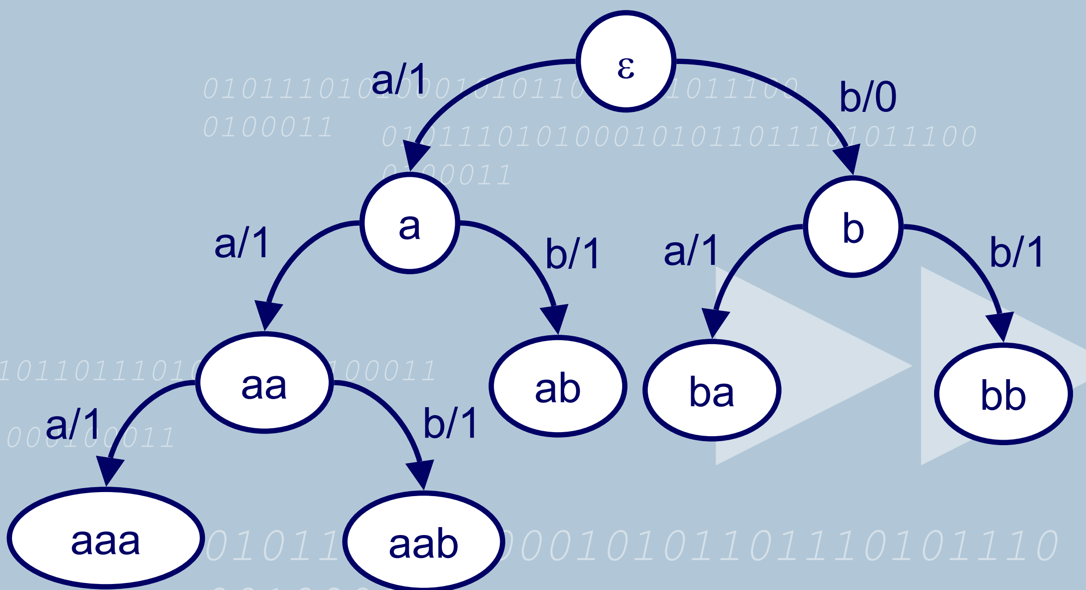
Wir betrachten V(A) als Automaten.

$V(A) = [\, X, Y, X^*, \delta^\infty, \lambda^\infty, \{ \varepsilon \} \,]$ mit:

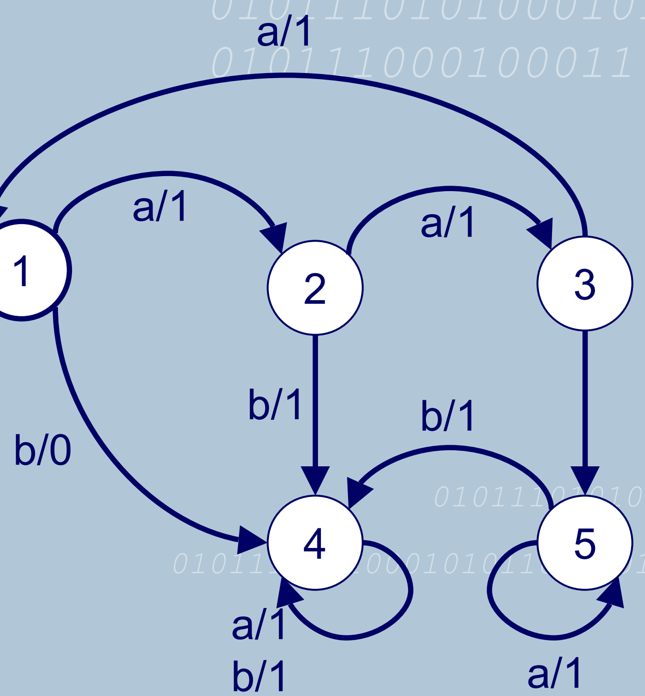
- $\delta^\infty(z,x) = zx$
- $\lambda^\infty(z,x) = y$, falls $y \in Y$ und $(zx,vy) \in V(A)$ für ein $v \in Y^*$

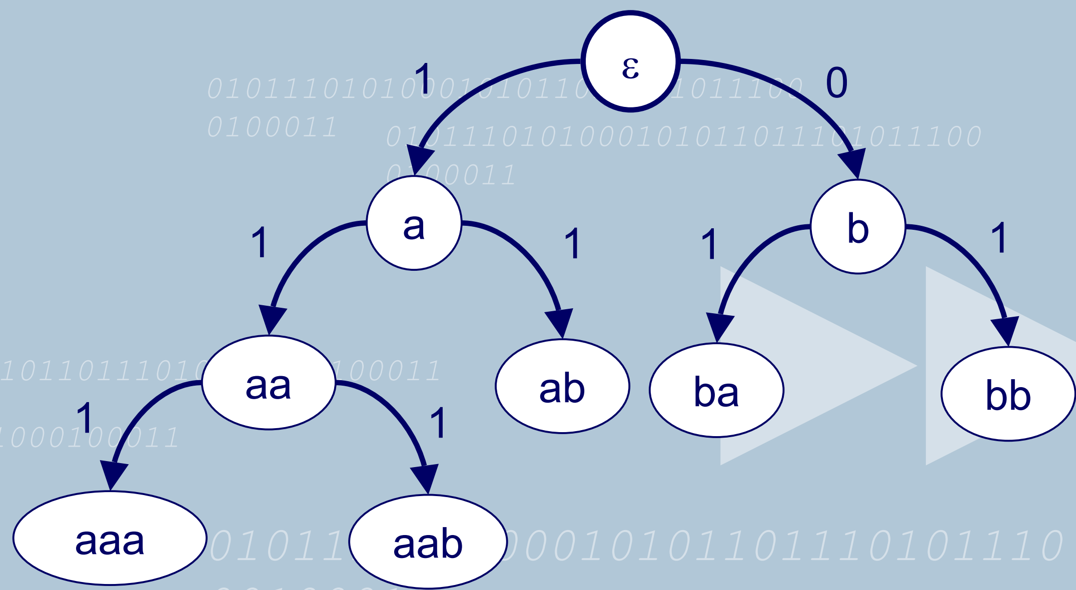
Automatensynthese





Automatensynthese





Automatensynthese

Grobstruktur des Lernalgorithmus:

betrachten $V(A) = [\, X, Y, X^*, \delta^\infty, \lambda^\infty, \{ \varepsilon \} \,]$

und $V_1 \subseteq V_2 \subseteq V_3 \subseteq V_4 \subseteq V_5 \subseteq \ldots V(A)$.

~ bezeichnet die Zustandsäquivalenz auf X^*.

$V_{i/\sim}$ bezeichnet den nach ~ „faktorisierten“ Automaten

Schritt 1:

Bilde $V_{1/\sim}$. Gib $H_1 = V_{1/\sim}$ aus.

Schritt n + 1:

Teste, ob H_n konsistent ist, d.h. $V_{n+1} \subseteq V(H_n)$.

Falls ja, gib $H_{n+1} = H_n$ aus.

Sonst bestimme $V_{n+1/\sim}$ und gib $H_{n+1} = V_{n+1/\sim}$ aus.

Automatensynthese

Bestimmung von $V_{i/\sim}$:

Man betrachtet V_i als (partiellen) Automaten und versucht, V_i zu minimieren, d.h. $V_{i/\sim}$ zu konstruieren.

... aber aufgrund fehlender Information ist noch zu klären, wie man die Relation ~ bzw. eine Approximation davon berechnet …

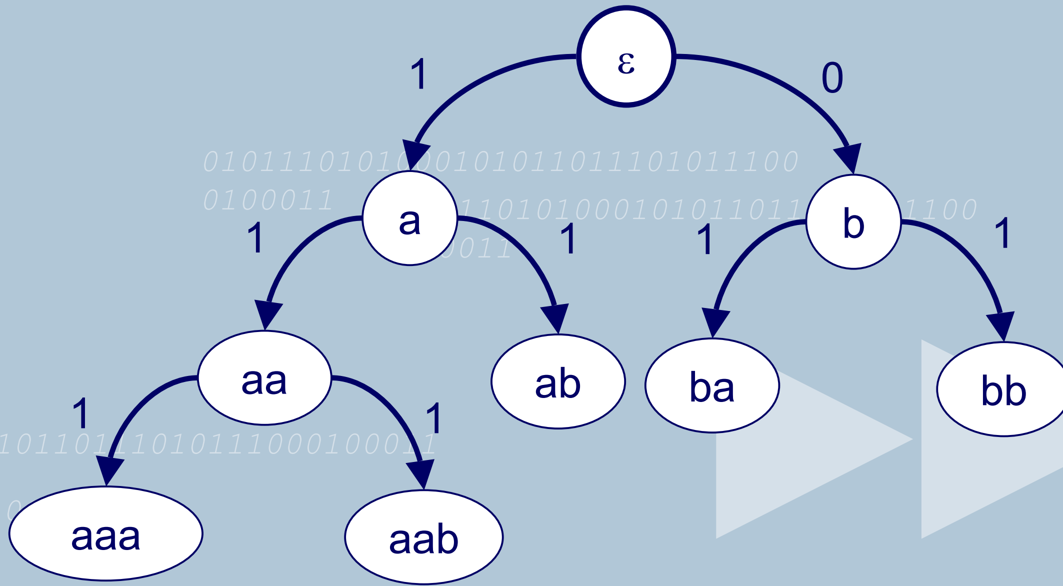
Automatensynthese

1-äquivalent:

{ ε } { a, b, aa }

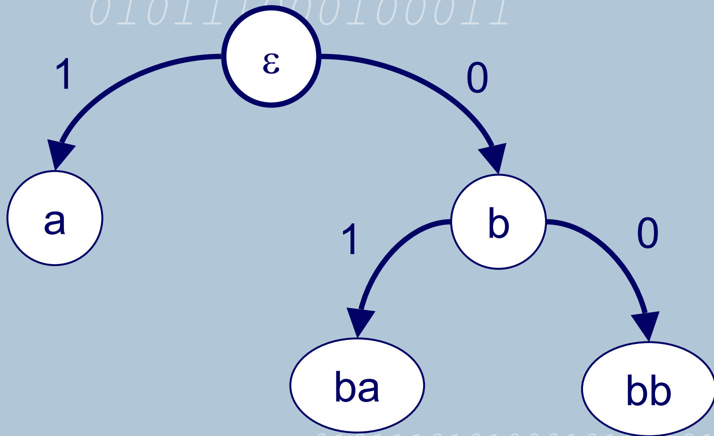
2-äquivalent:

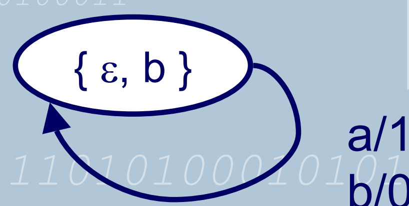
{ε } {a, ??? }



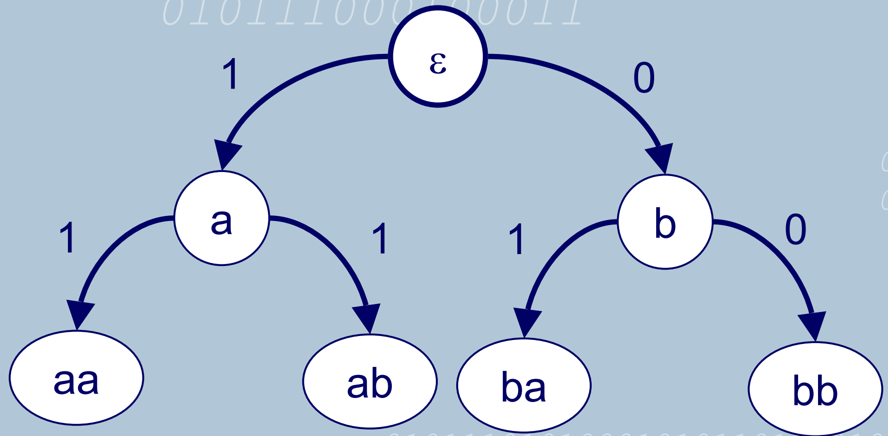
Idee: Zustände gelten als äquivalent, solange nichts dagegen spricht

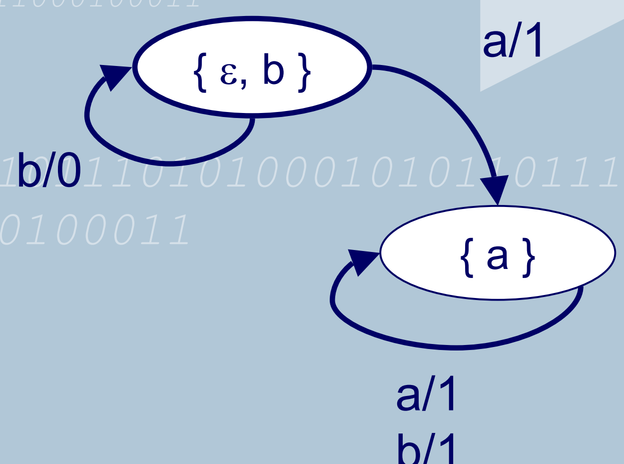
Automatensynthese



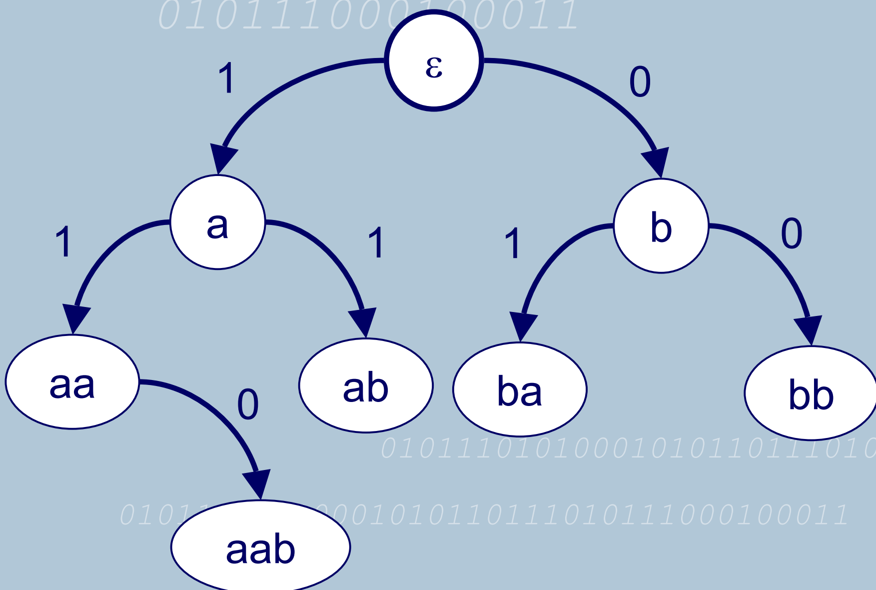


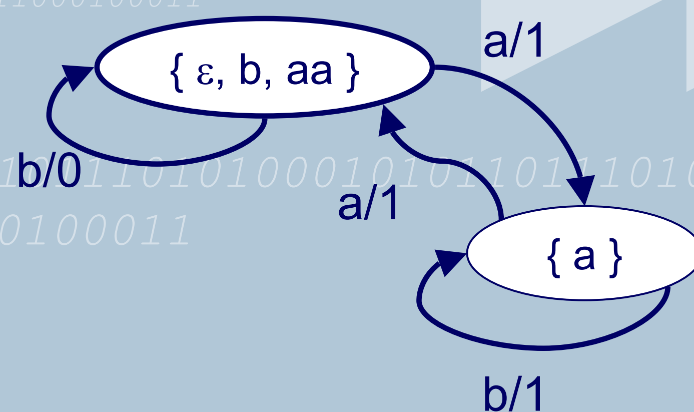
Automatensynthese





Automatensynthese





Automatensynthese

Bestimmung von $V_{i/\sim}$:

es gibt offenbar verschiedene Möglichkeiten, die Zustände von V_i zusammenzufassen und nicht definierte Werte im hypothetischen Automaten $V_{i/\sim}$ festzulegen (Heuristiken, ...)

Festlegungen führen zu unterschiedlichen Lernverfahren

Grundkonzept: nachweisbare Inäquivalenz

technische Voraussetzung: lexikographische Ordnung

Automatensynthese

Bestimmung von $V_{i/\sim}$:

<u>Bezeichnung:</u>

„$u_1 \nabla u_2$“, falls aus dem mit V_i verfügbaren Wissen folgt, daß die mit u_1 und u_2 bezeichneten Zustände nicht äquivalent sein können

d.h., die in u_1 und u_2 beginnenden Teilbäume unterscheiden sich, wenn man sie „übereinander legt“

Automatensynthese

Bestimmung von $V_{i/\sim}$ (/* Bestimmung der Zustandsmenge */)

sei V_i gegeben

```
k = 1;
R = { u | es gibt x ∈ X, v ∈ Y⁺: (ux,v) ∈ V_i }
while ( R ≠ ∅ )
        r_k = min { r | r ∈ R };
        R  = R \ { r_k };
        z_k = { r_k } ;
        while ( v = min { u ∈ R |  ∀ r ∈ z_k : ¬( u ∇ r ) }  existiert)
                R = R \ { v };
                z_k = z_k ∪ { v };
        k = k+1;
```

Automatensynthese

Bestimmung von $V_{i/\sim}$ (/* Bestimmung von δ und λ */)

sei V_i gegeben, sei $z_1,\ldots,z_n$ die bestimmte Zustandsmenge

- $\delta(z_r,x) = z_k$, falls es ein $u \in z_r$ mit $\delta^\infty(u,x) \in z_k$ gibt
- $\delta(z_r,x) = z_r$, sonst
- $\lambda(z_r,x) = y$, falls es ein $u \in z_r$ gibt mit $\lambda^\infty(u,x) = y$
- $\lambda(z_r,x) = y'$, sonst

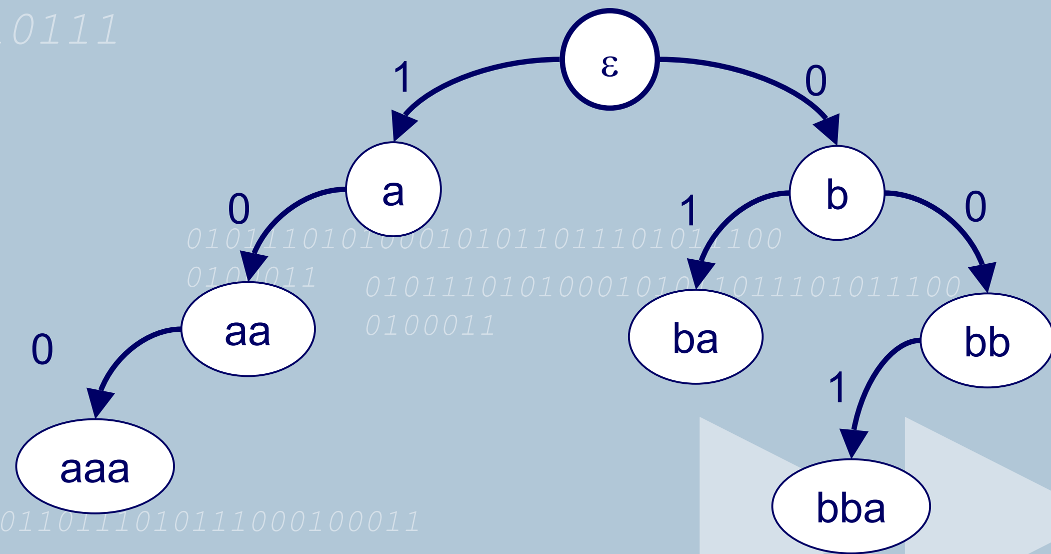
Bemerkung:

z_i ist Anfangszustand gdw. $\varepsilon \in z_i$

y‘ ist ausgezeichnetes Zeichen in Y

Automatensynthese

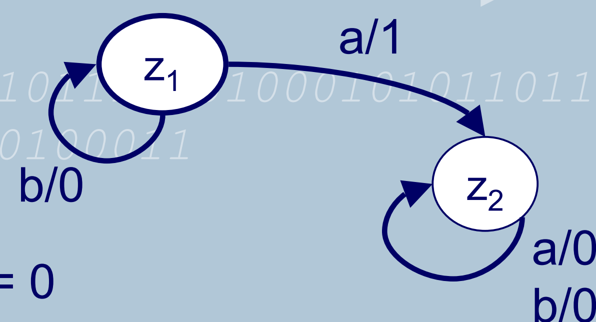
$V_i = \{$ (a,1), (b,0),
(aa,10),
(ba,01), (bb,00),
(aaa,100), (bba,001) $\}$



$R_i = \{ \varepsilon, a, b, aa, bb \}$

$z_1 = \{ \varepsilon, b, bb \}$

$z_2 = \{ a, aa \}$



Anmerkung: $y' = 0$

Automatensynthese

Satz: Die Klasse aller endlichen deterministischen Automaten ist im Limes aus ihrem Verhalten erlernbar.

Verifikation:

1. In jedem Schritt wird eine Hypothese berechnet
2. Die Folge der Hypothesen konvergiert
3. Konvergenz gegen korrekte Hypothese

Beobachtung:

~ bezeichnet die Zustandsäquivalenz auf X^*.

$V(A)_{/\sim}$ bezeichnet den nach ~ „faktorisierten" Automaten

Es gibt eine endliche Teilmenge V von V(A), so daß gilt: $V(A)_{/\sim} = V_{/\sim}$.

Außerdem gilt für alle V' mit: $V' \supseteq V$: $V'_{/\sim} = V_{/\sim}$.

Automatensynthese

Vorgehensweise:

- Minimierung von DFA
  - mittels Äquivalenzklassenbildung
- Leichte Änderung am Äquivalenzbegriff
- Minimierungsalgorithmus für DFA liefert einen Lernalgorithmus für DFA aus deren Verhalten