Programmierkurs Python II

Michaela Regneri
FR 4.7 Allgemeine Linguistik (Computerlinguistik)
Universität des Saarlandes

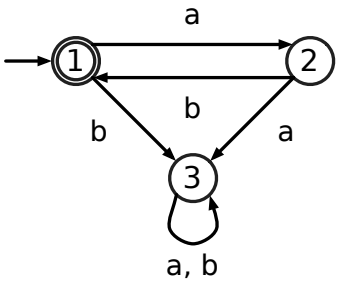
Sommersemester 2012

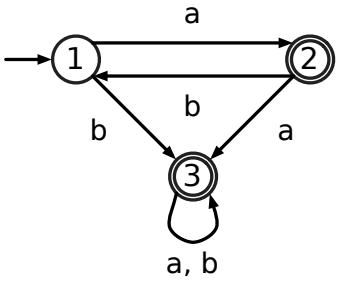
Übersicht

- Weitere Abschlusseigenschaften
  - Komplement
  - Schnitt
- Determinisierung (NEA ⇒ DEA)
- Minimierung

Komplementautomat

- Gegeben: ein DFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$
- Gesucht ist ein Automat, der das Komplement der Sprache L(M) beschreibt.
- Lösung: Vertausche die Endzustände
  - $M' = \langle Q, \Sigma, \delta, q_0, Q \backslash F \rangle$
- Voraussetzung: der Automat muss vollständig sein
  - $\delta(q,a)$ ist für alle $a \in \Sigma$, $q \in Q$ definiert





Schnitt

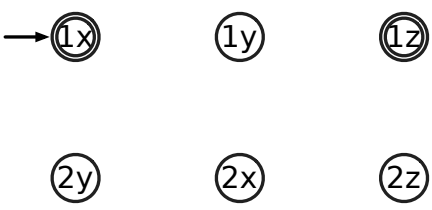
- Gegeben sind zwei Automaten M_1 und M_2
- Gesucht wird ein Automat M_3, der $L(M_1) \cap L(M_2)$ akzeptiert.
- Idee: die Automaten werden parallel laufengelassen
  - wenn beide Automaten nach dem Lesen der Eingabe in einem Endzustand stehen ⇒ das Wort liegt im Schnitt von L_1 und L_2.

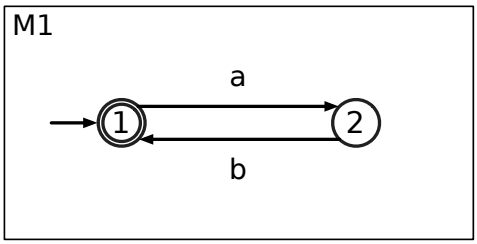
Schnitt (Produktautomat)

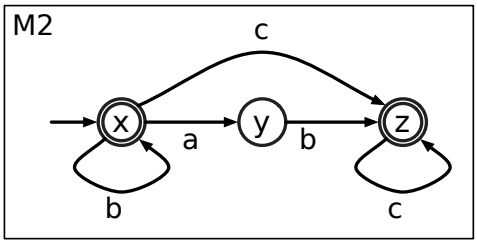
- Gegeben zwei Sprachen L_1 und L_2, die jeweils von den Automaten M_1 und M_2 akzeptiert werden.
  - $M_1 = \langle Q_1, \Sigma, \delta_1, s_1, F_1 \rangle$
  - $M_2 = \langle Q_2, \Sigma, \delta_2, s_2, F_2 \rangle$
- Produktautomat $M = \langle Q, \Sigma, \delta, s, F \rangle$
  - $Q = Q_1 \times Q_2$
  - $F = F_1 \times F_2$
  - $s = \langle s_1, s_2 \rangle$
  - $\delta(\langle q_1, q_2 \rangle, a) = \langle \delta_1(q_1, a), \delta_2(q_2, a) \rangle$

Schnitt (Produktautomat)

- Alle Zustände paaren
- Endzustände: alle Paare aus zwei Endzuständen
- Startzustand: das Paar aus den zwei Startzuständen

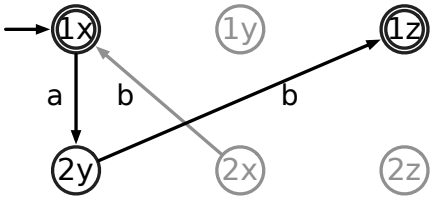


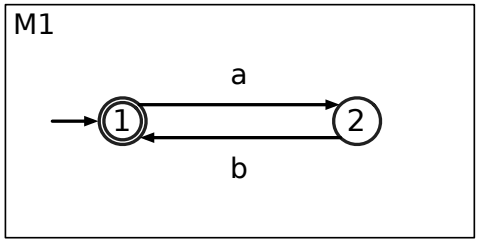


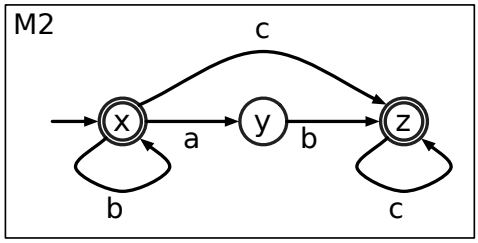


Schnitt (Produktautomat)

- $\delta(\langle p_1, q_1\rangle, a) = \langle p_2, q_2\rangle$ gdw.
  - $\delta_1(p_1, a) = p_2$ und
  - $\delta_2(q_1, a) = q_2$
- Nicht erreichbare Zustände können gelöscht werden.







Schnitt über Abschlusseigenschaften

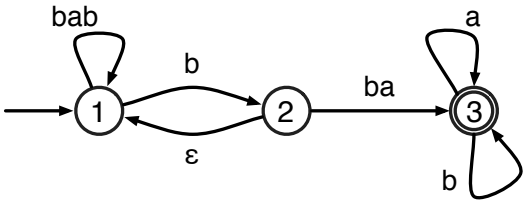
- $L(M_1) \cap L(M_2) = \Sigma^* - ((\Sigma^* - L(M_1)) \cup (\Sigma^* - L(M_2)))$
- Vereinige die Komplementautomaten und bilde davon den Komplementautomaten
- Notwendiger Zwischenschritt: Determinisierung

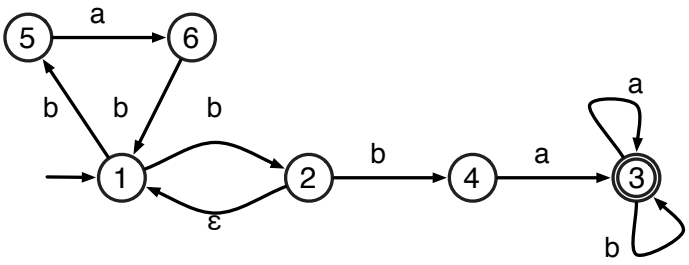
Äquivalenz von DEA und NEA

- Für jeden nichtdeterministischen Automanten M gibt es einen deterministischen Automaten M’ mit L(M) = L(M’)
- Der Beweis ist gleichzeitig der Korrektheitsbeweis für den Konstruktionsalgorithmus.
- Der resultierende Automat
  - ist deterministisch
  - und beschreibt die gleiche Sprache wie der NEA
- Wir beschränken uns hier auf die Beschreibung des Algorithmus

Determinisierung

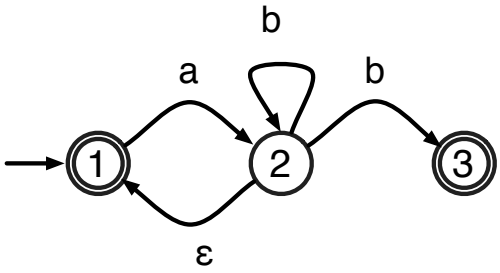
- Vorverarbeitungsschritt: Mehrsymbolkanten entfernen





Determinisierung

- ε-Abschluss:
  - $E(s) = \{ s' \mid \langle s, \varepsilon \rangle \vdash^* \langle s', \varepsilon \rangle \}$
- Beispiel:
  - $E(q_1) = \{q_1\}$
  - $E(q_2) = \{q_1, q_2\}$
  - $E(q_3) = \{q_3\}$



Determinisierung

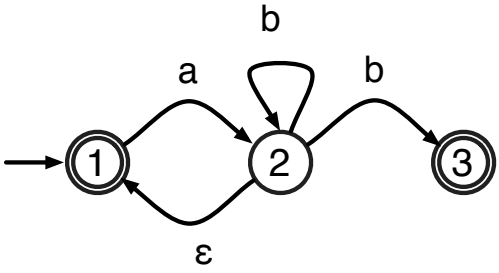
- Die Zustände des determinierten Automaten sind Mengen von Zuständen des ursprünglichen Automaten
- Idee: Die Zustandsmengen geben an, in welchem Zustand sich der ursprüngliche Automat nach Lesen der Eingabe befinden kann.

Determinisierung

- NEA $M = \langle Q, \Sigma, \Delta, q0, F \rangle$
- DEA $M' = \langle Q', \Sigma, \delta', q0', F' \rangle$ mit
  - $Q' = 2^Q$
  - $q_0' = E(q_0)$
  - $F' = \{ K \subseteq Q' \mid K \cap F \neq \emptyset \}$
  - Für alle $K \subseteq Q'$ und $a \in \Sigma$:
    - $\delta'(K, a) = \bigcup \{ E(p) \mid p \in Q$ und für ein $q \in K: \Delta(q, a, p) \}$

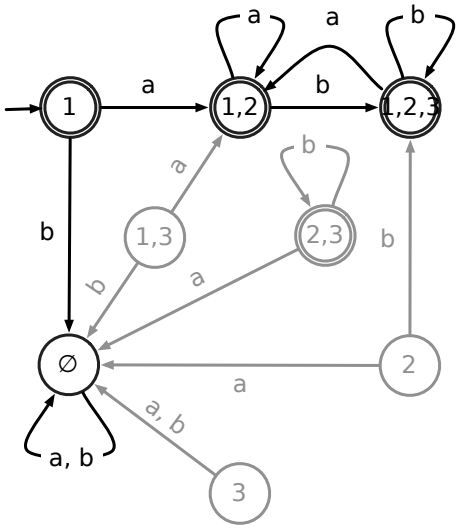
Determinisierung

	1	2	3	1,2	1,3	2,3	1,2,3	∅
1				a				b
2							b	a
3								a,b
1,2				a			b	
1,3				a				b
2,3						b		a
1,2,3				a			b	
∅								a,b



Determinisierung

	1	2	3	1,2	1,3	2,3	1,2,3	∅
1				a				b
2							b	a
3								a,b
1,2				a			b	
1,3				a				b
2,3						b		a
1,2,3				a			b	
∅								a,b

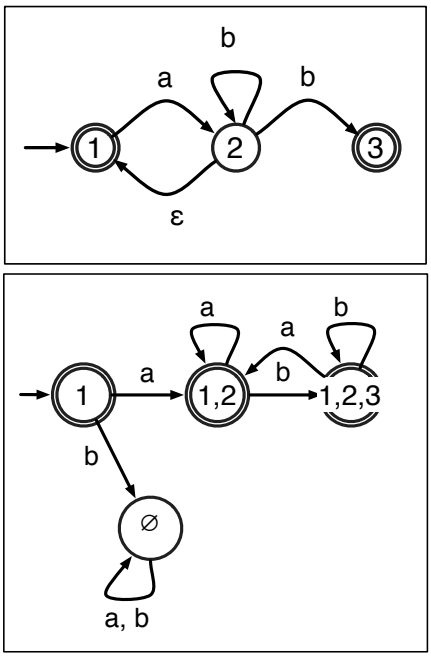


Determinisierung (kürzer)

- Algorithmus zur Teilmengen-Konstruktion (*Subset construction*)
- Generiere nur die Zustände, die überhaupt besucht werden können
- Prinzip:
  - Ersetze Zustände durch ihren ε-Abschluss
  - Beginne beim Startzustand
  - Generiere „on the fly“ die Zielzustände
  - Verfahre mit den generierten Zielzuständen entsprechend

Determinisierung (kürzer)

- Startzustand: {1}
- δ'({1}, a) = {1,2} (= Endz.)
- δ'({1,2}, a) = {1,2}
  δ'({1,2}, b) = {1,2,3} (= Endz.)
- δ'({1,2,3}, a) = {1,2}
  δ'({1,2,3}, b) = {1,2,3}
- δ'({2,3}, b) = {1,2,3}
- ggf. Senke hinzufügen
  (nicht zwingend notwendig)

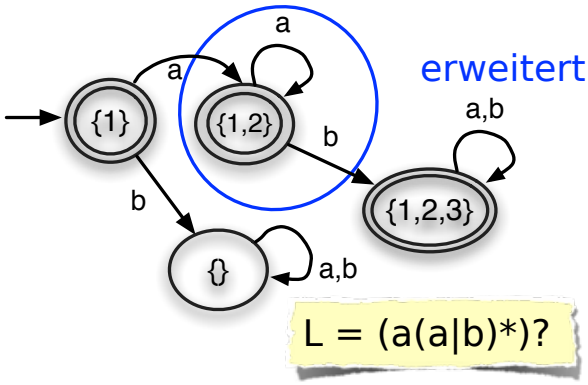


Determinisierung - Algo-Skizze

- (entferne Mehrsymbol-Kanten)
- entferne Epsilon-Übergänge (siehe letzte Übung)
- alter Startzustand = neuer Startzustand
- initialisiere eine Agenda mit Startzustand
- so lange es unbesuchte Zustände gibt:
  - füge einen Übergang hinzu vom aktuellen Zustand zu einer "Zustandsmenge" (=Zustand mit Mengen-Label)
  - werfe den neu generierten Zielszustand auf die Agenda
- Endzustände = alle Zustände mit mindestens einem Endzustand als Element

Minimierung eines DEA

- Ein DEA ist minimal, wenn es keinen DEA mit weniger Zuständen gibt, der die gleiche Sprache beschreibt
- Problem: finde zu einem beliebigen DEA einen äquivalenten, minimalen DEA



erweitert die Sprache nicht

Minimierung eines DEA

- Ein DEA kann minimiert werden, wenn er äquivalente Zustände enthält
- Zwei Zustände sind äquivalent, wenn sie die gleichen rechten Sprachen haben
- Die rechte Sprache eines Zustandes q_i in einem DEA $M=(Q,\Sigma,\delta,q0,F)$ ist $L(Q,\Sigma,\delta,q_i,F)$ (alle Worte, die ab qi gelesen werden können)
- Äquivalenzklassen eines Automaten M sind Mengen, die jeweils äquivalente Zustände (in M) enthalten
- Ein minimaler Automat hat als Zustände die Äquivalenzklassen des ursprünglichen Automaten

Minimierung eines DEA

- Sei `EqClass(q)` eine Funktion, die die Äquivalenzklasse eines Zustandes q zurückgibt
- Ein minimaler Automat M‘ äquivalent zu M=(Q,Σ,δ,q0,F)
- Q = {K | K=`EqClass(q)` for q ∈ Q}
- q0‘ = `EqClass(q0)`
- F = {K | K=`EqClass(q)` for q ∈ F}
- δ‘(ki,σ)={T | T=`EqClass(`q_j`)`, δ(q_i,σ)=q_j for some q_i ∈ k_i}
- Kritischer Punkt: Berechnung der Äquivalenzklassen

Äquivalenzklassen im DEA

- Idee: führe Äquivalenzklassen auf Nicht-Äquivalenz zurück
- „Lokale“ Nicht-Äquivalenz ist einfach

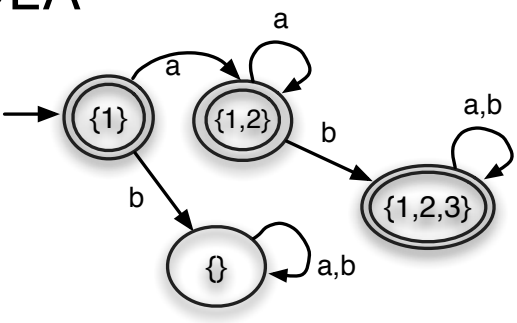
```
LocalEq(qi,qj):

if final(qi) != final(qj): return false
if out_symbols(qi) != out_symbols(qj):
   return False
return True
```

Äquivalenzklassen im DEA

- Als Hilfs-Struktur: speichere eine Tabelle, die für 2 Zustände sagt, ob sie äquivalent sind
- Initialisiere alle Felder mit „True“; dann setze sie ggf. nacheinander auf „False“
- Nimm eine Sortierung über den Zuständen an (z.B. jeder kriegt eine Nr.), um Dopplungen zu vermeiden
- Rekursiv:
  - Überprüfe lokale Äquivalenz von Zustandspaaren q_i und q_j
  - nicht äquivalent: markiere alle Vorgänger von q_i als nicht äquivalent mit allen Vorgängern von q_j

Äquivalenzklassen im DEA

```
Prop(qi,qj):
   for ini in in_edges(qi)
      for inj in in_edges(qj)
         if EQ[ini.src][inj.src] == 1
           EQ[ini.src][inj.src] = 0
           Prop(ini.src, inj.src)
```




```
EquivalenceClasses(Q,Σ,δ,q0,F):
   for qi,qj in Q, qi< qj:
      EQ[qi][qj] = 1
   for qi in Q:
      for qj<qi in Q:
         if EQ[qj][qi] And Not LocalEq(qi,qj):
            EQ[qj][qi] = 0
            Prop(qi,qj)
```

Äquivalenzklassen im DEA - Ergebnis

	{}	{1}	{1,2}	{1,2,3}
{}	1	0	0	0
{1}	0	1	0	0
{1,2}	0	0	1	1
{1,2,3}	0	0	1	1

Bsp.: Äquivalenz-Klasse für {1,2}

Minimierung eines DEA nach Brzozowski

- Beginnend bei einem DEA M, minimiserung durch Umkehrung ➔ Determinisierung ➔ Umkehrung ➔ Determinisierung
- Das Ergebnis der letzten Determinisierung ist ein minimaler Automat
- Umkehrung eines Automaten:
- Startzustand ➔ Endzustand
- Endzustände ➔ Startzustand
- Übergänge werden umgedreht

Minimierung eines DEA nach Brzozowski

- Nach dem Umkehren eines DEA gibt es nur noch einen Endzustand
- Gäbe es nach der 2. Umkehrung noch Äquivalente Zustände, müssten 2 Kanten mit gleichem Symbol auf den Endzustand zeigen
- Das kann aber nicht sein: Die Kanten wurden umgedreht, und der Automat war vorher deterministisch
- Die Determinisierung ändert das Minimalitäts-Kriterium nicht

umkehren
determinisieren

Zusammenfassung

- Mehr Automaten-Algorithmen
- Abschlusseigenschaften:
  - Komplement
  - Schnitt
- Determinisierung:
  - Potenzautomat
  - Teilmengen-Konstruktion
- Minimierung
  - Äquivalenz-Klassen
  - Brzozowski