Automatentheorie „Berechnungsmodell für logische Sprachen“

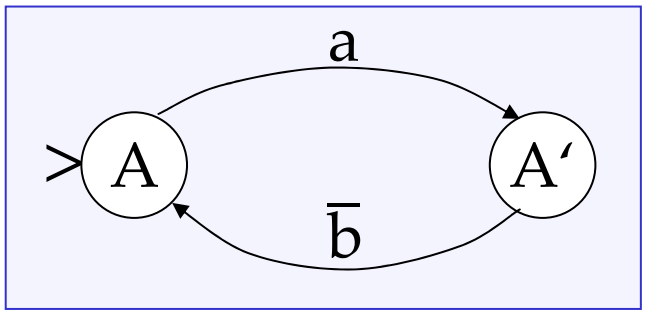
Thorsten Haupt

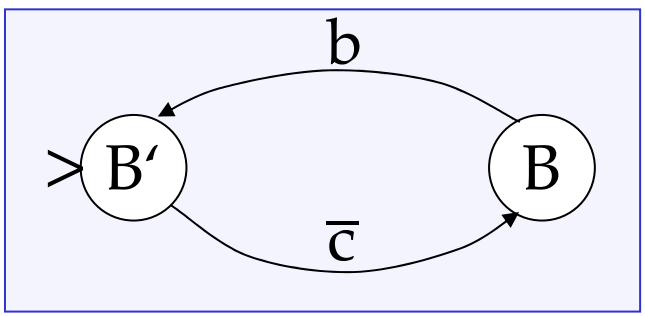
Betreut durch Tim Priesnitz

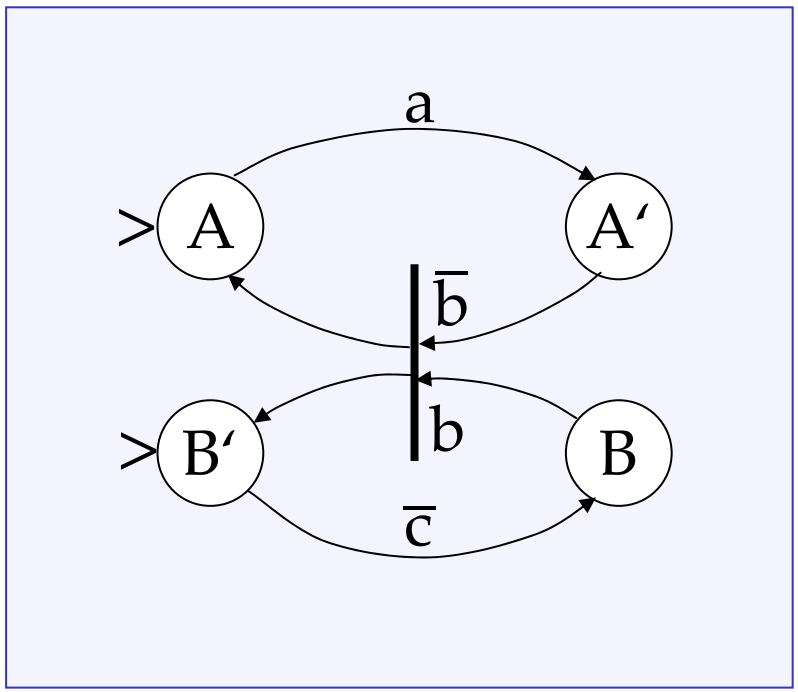
Proseminar Theorie kommunizierender Systeme

Programming Systems Lab – Prof. Gert Smolka

Automaten für Nebenläufigkeit







$A = a.A'$ $\quad B = b.B'$

$A' = \overline{b}.A$ $\quad B' = \overline{c}.B$

CCS

Übersicht

- Einführung
- Abschluss- & Spracheigenschaften
- Sprachäquivalenzen
- Erweiterung

Theorie der endlichen Automaten

- eingeführt für
  Modellierung von Neuronen-Netzen,
  Schaltkreisen, Parsern, ...

  [Kleene, 1956]
- Berechnungsmodell für logische Sprachen
  Klassiker: schwache Monadische Logik 2. Ordnung
  mit einem Nachfolger

  [Büchi, 1960]
- Anwendungen: Verifikation, Compilerbau, ...

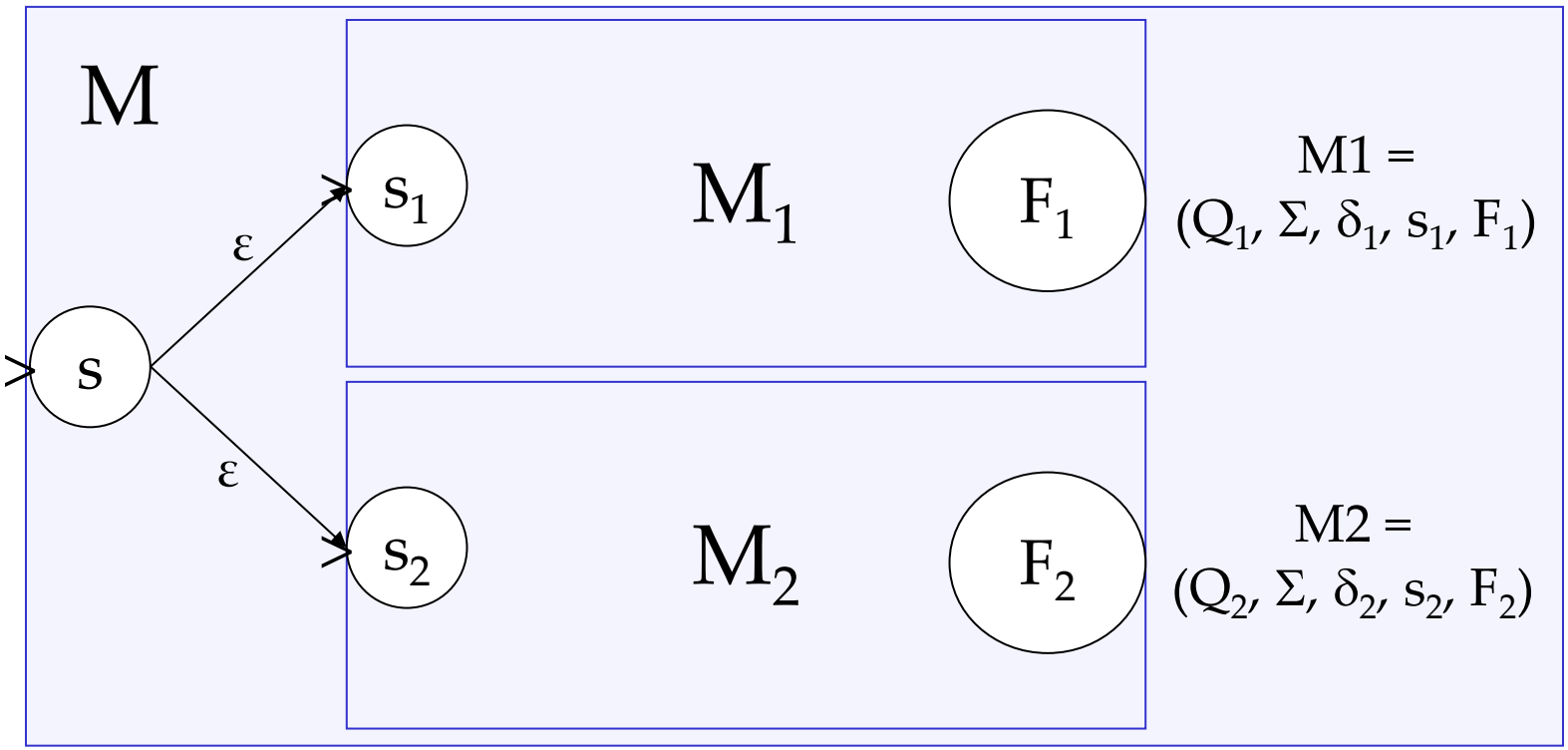
Definition endlicher Automaten

- Q : endliche Menge von Zuständen
- Σ : endliche Menge von Eingabesymbolen
- $\delta : Q \times \Sigma \rightharpoonup \{Q \text{ Übergangsfunktion}$
- $s \subseteq Q$: Menge von Startzuständen
- $F \subseteq Q$: Menge von Endzuständen

Abschluss- & Spracheigenschaften

		NEA	DEA
Vereinigung	$L_1 \cup L_2$	$O(n)$	$O(n^2)$
Komplement	$\overline{L}_1$	$O(2^n)$	$O(n)$
Schnitt	$L_1 \cap L_2$	$O(n^2)$	$O(n^2)$
Leerheit	$L(A) \stackrel{?}{=} \varnothing$	$O(n \log n)$	$O(n \log n)$
Universalität	$L(A) \stackrel{?}{=} \Sigma^*$	PSPACE	$O(n)$

Vereinigung von Automaten



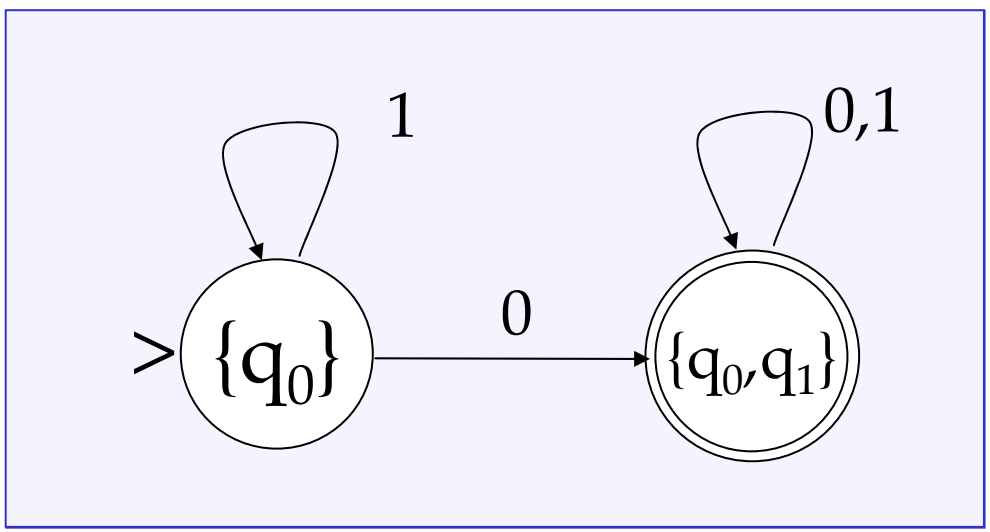
$$M = (Q_1 \cup Q_2 \cup \{s\}, \Sigma, \delta_1 \cup \delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}, s, F_1 \cup F_2)$$

Determinisierung von Automaten

Jede von einem NEA akzeptierbare Sprache ist auch durch einen DEA akzeptierbar.

[Rabin,Scott, 1959]

Idee: Teilmengenkonstruktion

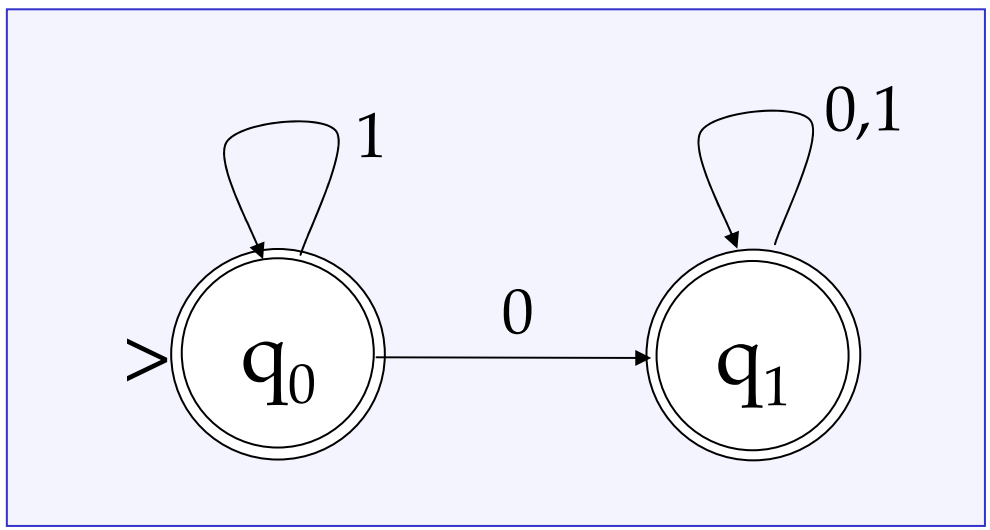


	0	1
$\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_1\}$	$\{q_1\}$	$\{q_1\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_1\}$

Komplementbildung

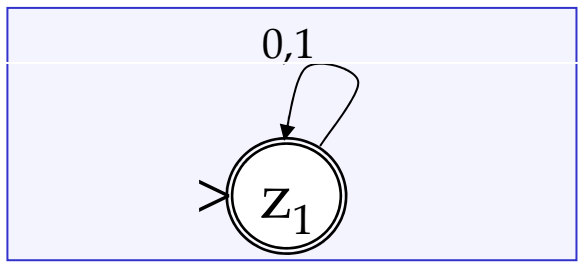
Idee:

1. Automat determinisieren √
2. Automat vervollständigen √
3. Endzustände und Nichtendzustände tauschen √

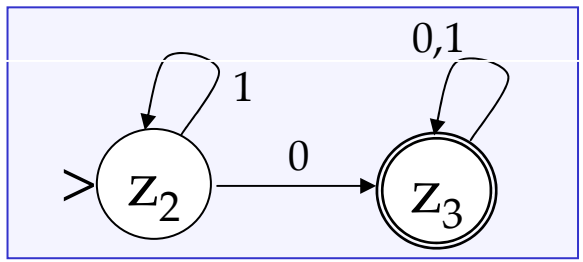


Schnitt

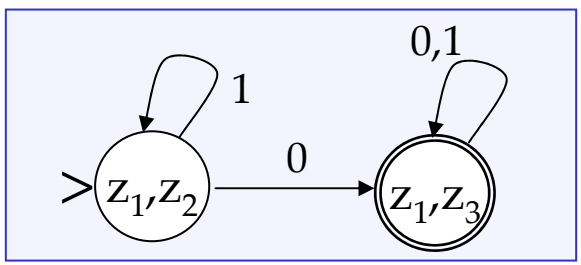
Idee: Produktautomat



$(Q_1, \Sigma, \delta_1, s_1, F_1)$



$(Q_2, \Sigma, \delta_2, s_2, F_2)$



$(Q_1 \times Q_2, \Sigma, \delta, (s_1,s_2), F_1 \times F_2)$

mit $\delta((q, p), a) = (\delta_1(q, a), \delta_2(p, a))$

Leerheit

Sprache ist leer $\Leftrightarrow$ existiert kein Pfad
von Start- nach Endzustand

Implementierung: Dijkstra Algorithmus [1959]
O(n log n)

Universalität

Idee: Zurückführung auf Leerheitsproblem

1. Determinisieren
2. Vervollständigen
3. Komplement bilden
4. Komplement leer => Sprache universell

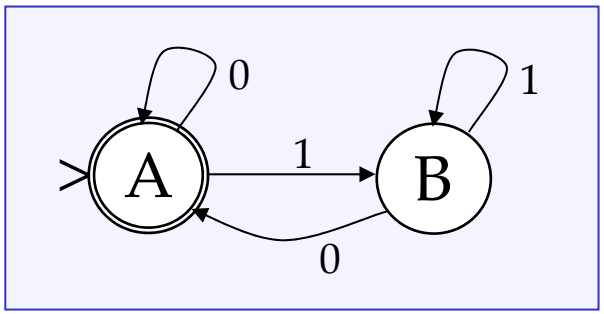
- determinisieren sehr teuer
  => determinisieren „on-the-fly“
- raten des Pfades auf dem Zustandsgraph
- Terminierung: Übergangszähler (Schranke : $2^{|Q|}$)

=> Reduzierung der Komplexität auf PSPACE

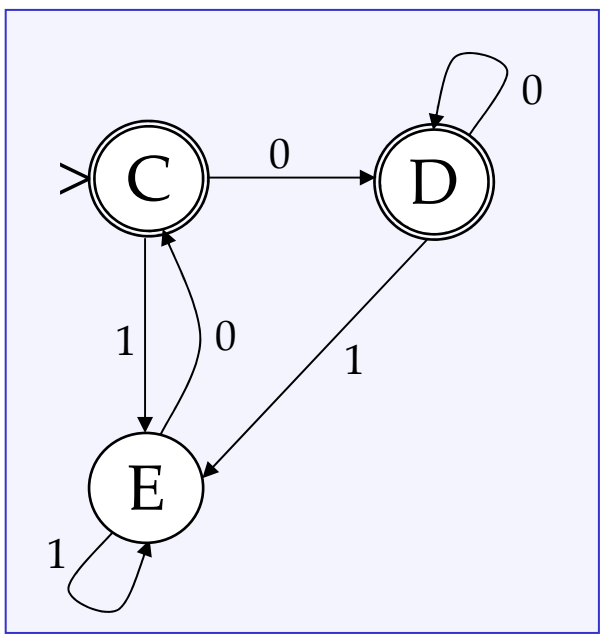
Sprachäquivalenzen

	NEA	DEA
Bisimulation	PSPACE	$O(n \log(n))$
klassische	PSPACE	$O(n^2)$

Klassische Sprachäquivalenz



Automat 1

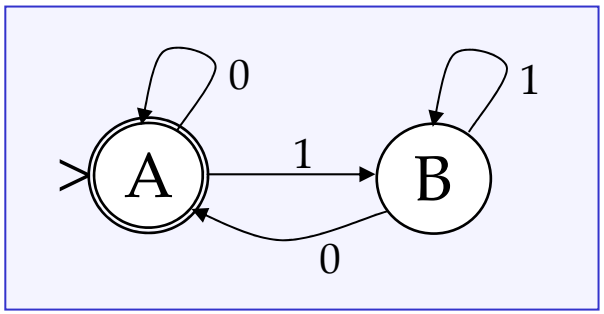


Automat 2

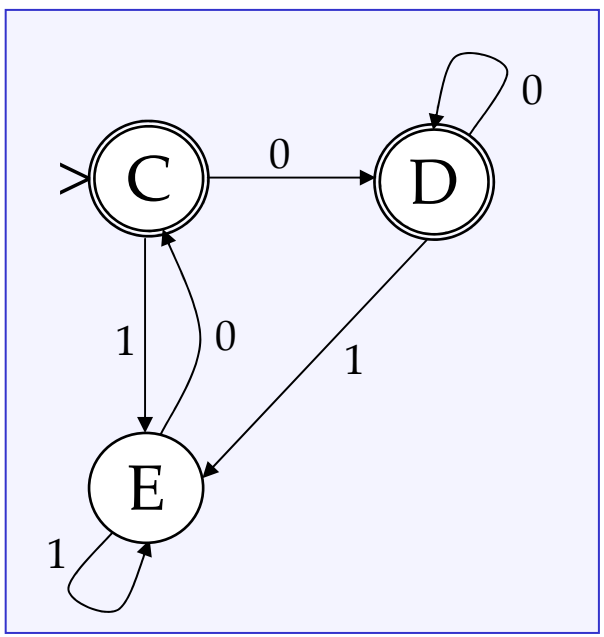
	A	B	C	D
B	X			
C	○	X		
D		X		
E	X		X	X

A und C äquivalent
=> Automat 1 & 2 sind äquivalent

Minimierung



Automat 1

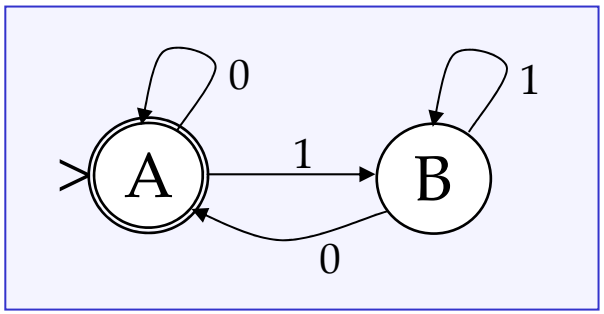


Automat 2

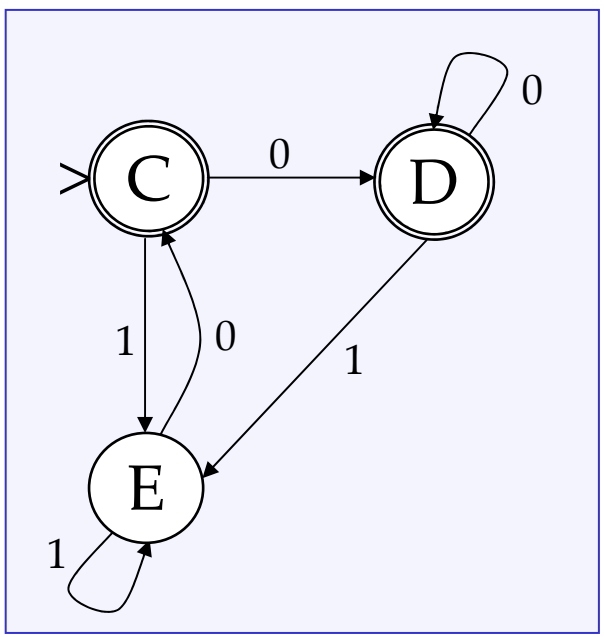
	C	D
D		
E	X	X

C und D äquivalent
=> Automat 1 Minimalautomat

Bisimulation



Automat 1



Automat 2

{A}{B}{C}{D}{E}

{A}{B}{C}{D}{E}

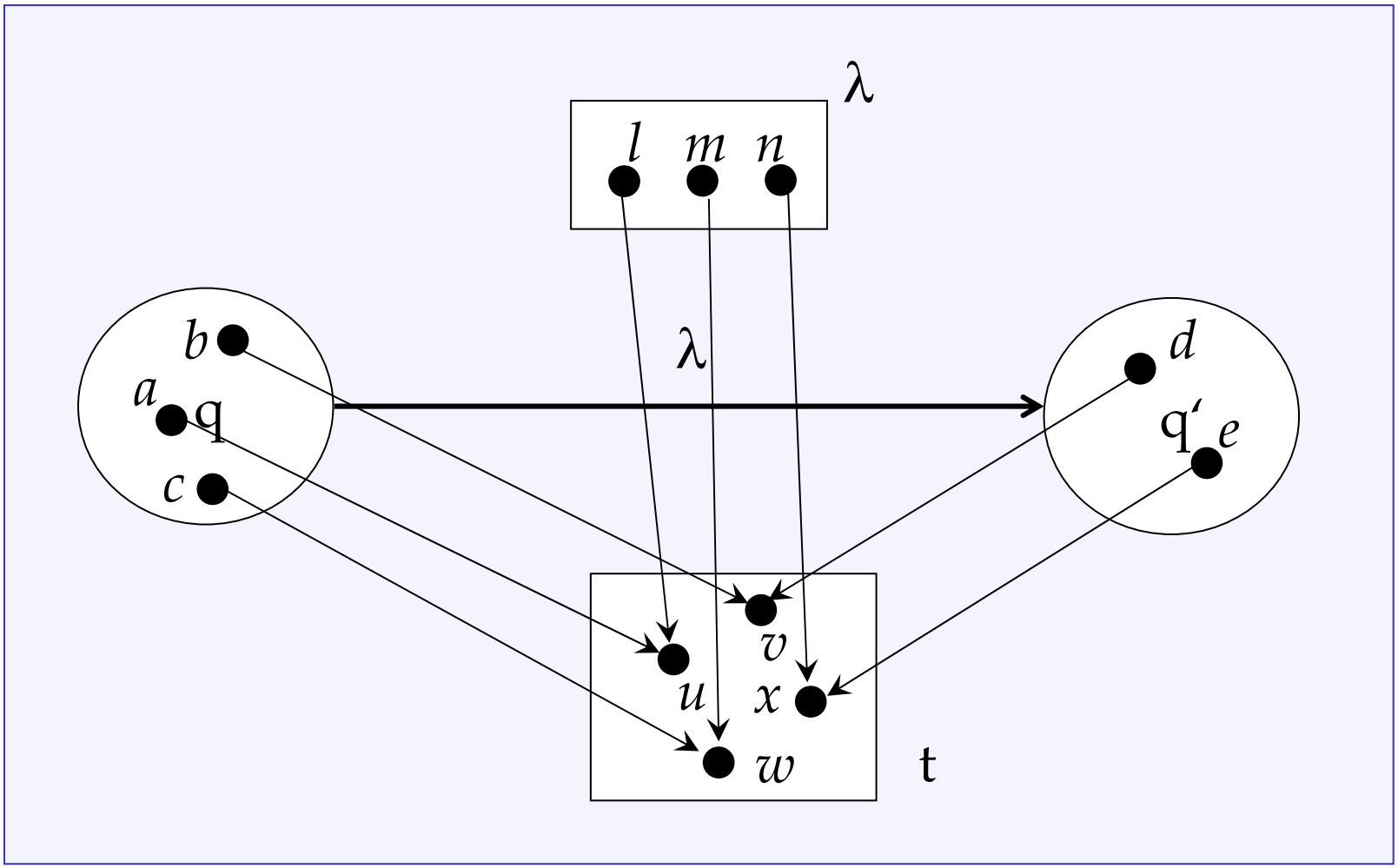
B	X			
C	X	X		
D	X	X	X	
E	X	X	X	X
	A	B	C	D

{A}{C}

{E} - 0

Automat 1 & 2 sind NICHT äquivalent

Motivation HD-Automaten



Referenzen

- Milner, R., Communicating and Mobile Systems: the π – Calculus, Cambridge University Press, 1999
- Hopcroft, J.E., Motwani, R. und Ullman, J.D., Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, 2. Auflage, Addison Wesley, 2002
- Schöning, U., Theoretische Informatik kurzgefasst, Spectrum, 2001
- Montanari, U. und Pistore, M., History-Dependent Automata, ENTCS, Vol. 10, 1998
- Hermanns, H., Finkbeiner, B., Podelski, A., Verification, Vorlesung, Universität des Saarlandes, SS 2003