Hinweis: EA, die neben ihrer Rolle als Akzeptoren auch Ausgaben erzeugen, heißen Finite State Machines (FSM)

Nichtdeterministische EA (NFA)

Verwendung

a) als Beweishilfsmittel für die Äquivalenz reguläre Grammatik → DFA

b) als kompakte Darstellung für bestimmte Sprachen.

Erinnerung: Beim NFA ist es zugelassen, daß von einem Zustand $z \in Z$ mehrere Kanten mit der gleichen Beschriftung zu unterschiedlichen Folgezuständen gehen:

Funktionsweise: Wird im Zustand z das Zeichen a gelesen, kann der Automat sowohl nach z_1 als auch nach z_2 gehen. Ein Ein Wort $x \in A^*$ gilt als akzeptiert, wenn der Automat eine Zustandsfolge durchlaufen kann, die auf einem Endzustand endet.

Es kann durchaus Zustandsfolgen geben (beim gleichen vorgelegten Wort), die nicht auf Endzustände führen.

Ausschlaggebend ist, daß es mindestens eine akzeptierende Zustandsfolge gibt; dementsprechend lassen wir auch Mengen von Startzuständen zu.

Definition: Ein NFA ist ein 5-Tupel

$$M = (Z, A, \delta, S, E).$$

Bedeutung wie beim DFA, nur ist nun $S \subseteq Z$ Menge und die Funktionalität von δ ist nun $Z \times A \to P(Z)$ (Potenzmenge)

weil gleichbeschriftete Kanten im Prinzip auf jede Teilmenge von Folge zustanden aus der gesamten Zustandsmenge führen kann.

Auch hier Verallgemeinerung der Übergangsfunktion δ für Wörter. Zunächst war beim DFA die induktive Definition von δ

$$\hat{\delta}(z, \varepsilon) = z$$

$$\hat{\delta}(z, ax) = \hat{\delta}(\delta(z, a), x) \text{ mit } z \in Z, x \in A^*$$

beim NFA ist dagegen (wegen der vielen möglichen Pfade)

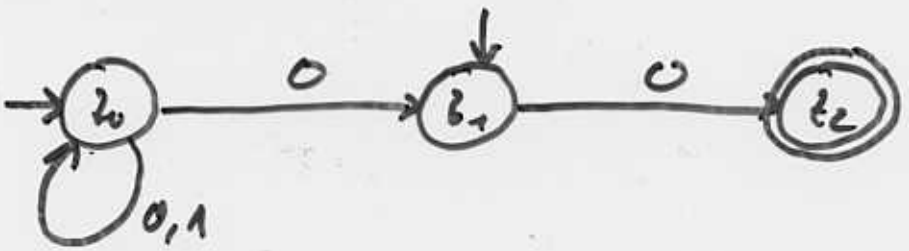
$$\hat{\delta}(z, \varepsilon) = z' \quad \text{wo} \quad z' \subseteq Z$$

$$\hat{\delta}(z, ax) = \bigcup_{t \in z'} \hat{\delta}(\delta(a, a), x)$$

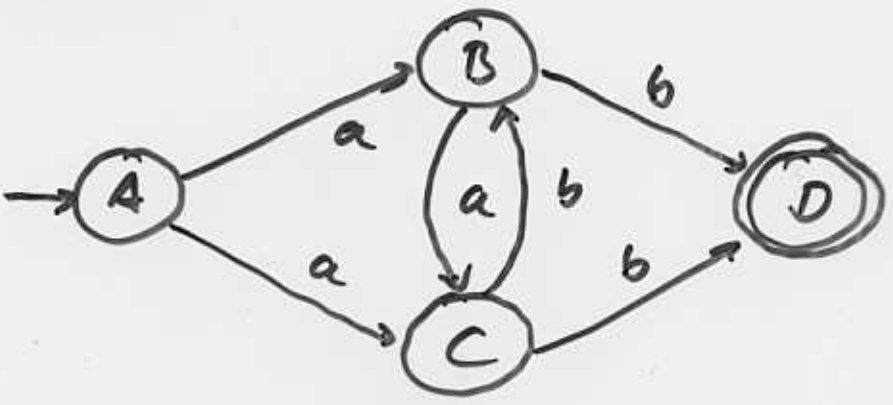
und die vom NFA Akzeptierte Sprache ist $L(M) = \{ x \in A^* \mid \hat{\delta}(S, x) \cap E \neq \emptyset \}$

Beispiele für NFA

a) Automat, der alle Wörter über $\{0,1\}$ akzeptiert, die mit 00 enden, oder das Wort $x = 0$



b) Automat



$$L = a((b?(ab)^* \mid a?(ba)^*))b$$

Jeder NFA kann in einen DFA als "Ersatzakzeptor" umgeformt werden.
"Trick" bei der Umformung: Von einem Zustand gehen nun nicht mehr mehrere Kanten a zu vielen einzelnen Zielzuständen, sondern eine Kante a geht zu einem Knoten, der alle Zielzustände umfaßt. Damit ist die Struktur des DFA wieder erreicht.
Damit hat der aus einem NFA M mit n Zuständen folgende DFA M' alle Elemente der Potenzmenge (= alle möglichen Teilmengen der Zustände vom NFA) als Zustände (also 2^n).
Damit wird $M' = (\mathfrak{Z}, A, \delta', z_0', E)$ wo

$\mathfrak{Z} = P(Z)$ Potenzmenge

$$\delta'(z', a) = \bigcup_{z \in z'} \delta(z, a) = \hat{\delta}(z', a)$$

mit $z' \in \mathfrak{Z}$

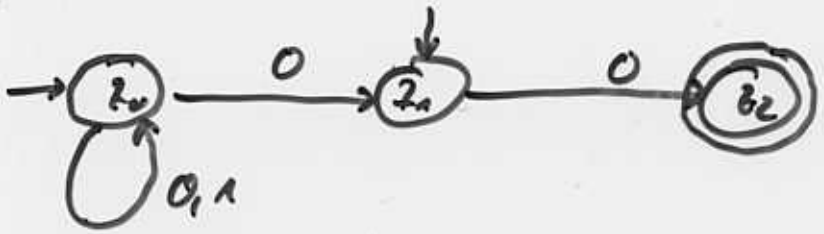
$z_0 = S$

$$E' = \{ z' \subseteq Z \mid z' \cap E \neq \emptyset \}$$

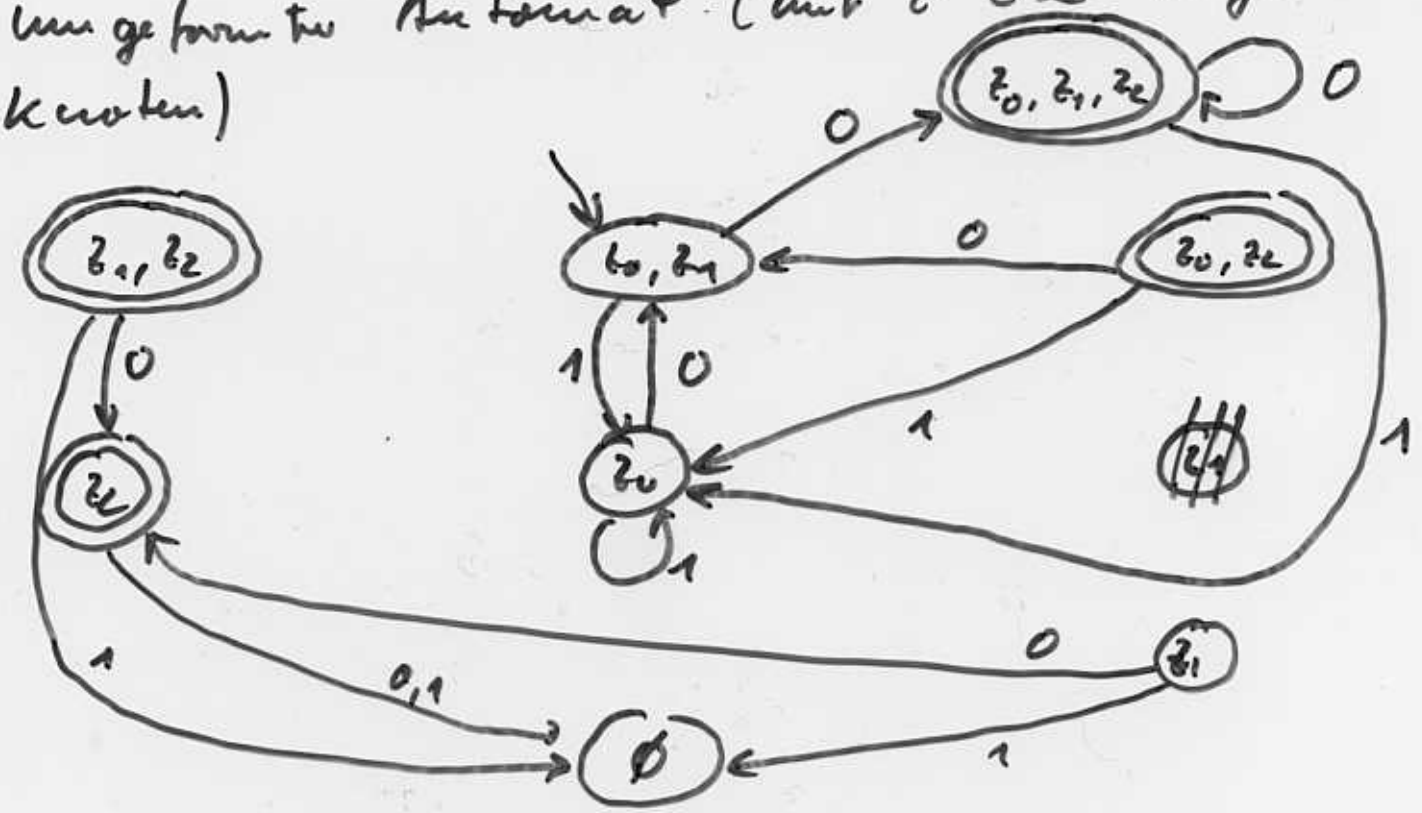
und es läßt sich zeigen, daß jede von einem NFA erzeugte Sprache auch von einem DFA akzeptiert wird.

Beispiele für diese Umwandlung:

Automat und Reg a)

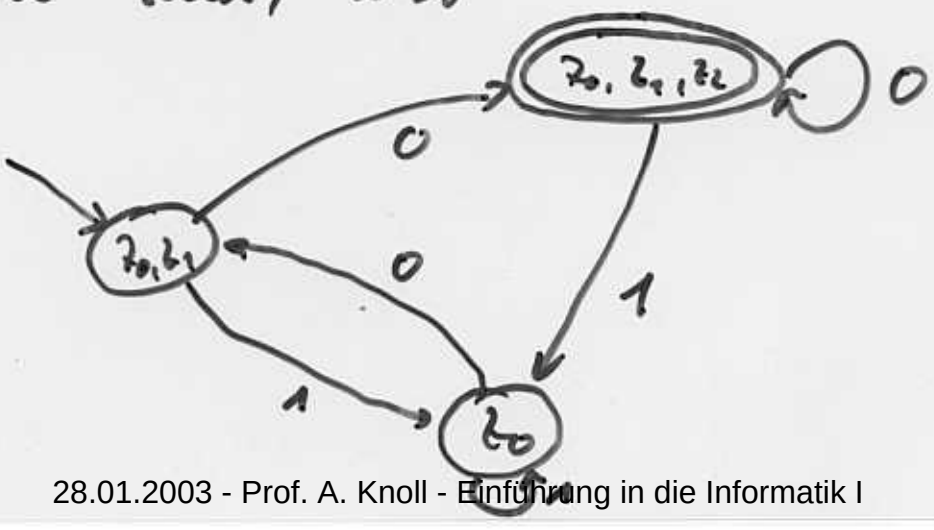


umgeformter Automat (mit $t' \in \mathbb{Z}$ in jedem Knoten)

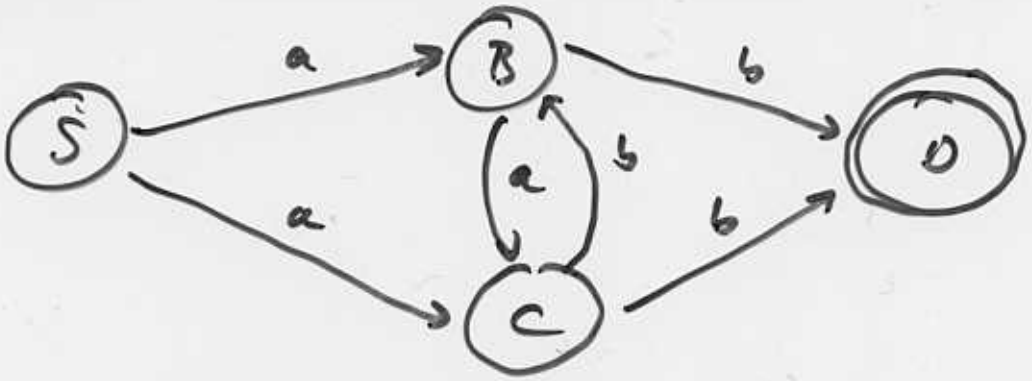


"Fehlerzustand"

Interessant ist nur der Automat, dessen Endzustände vom Startzustand aus erreichbar sind, also



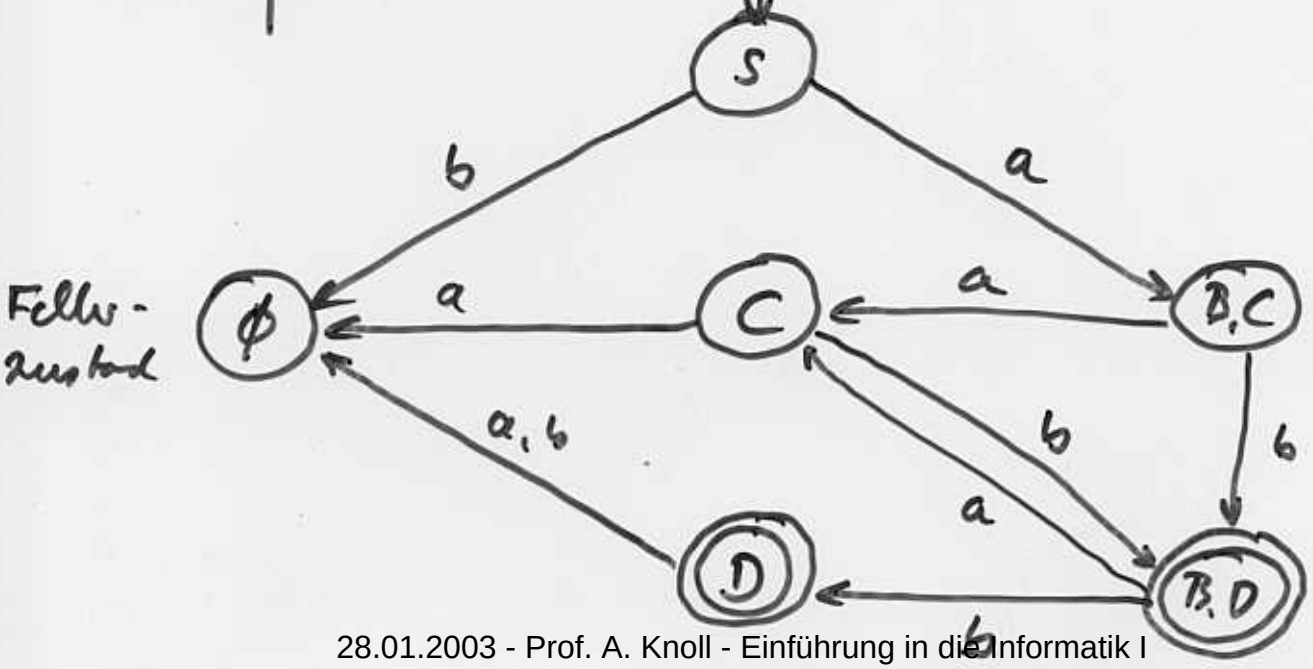
Umwandlung des Automaten und b) durch systematische Erzeugung nur der tatsächlich benötigten Zustandsmenge:



Entwicklung der Übergangsfunktion des Ersatzakzeptors wie folgt:

δ_d	a	b
S	B, C	$\emptyset$
C	$\emptyset$	B, D
D	$\emptyset$	$\emptyset$
B	C	D
B, C	C	B, D
B, D	C	D

B ist in beiden Zuständen B, C und B, D enthalten



Kontextfreie Sprachen und Kellerautomaten

- Reguläre Sprachen haben großes Anwendungsfeld, aber ihre Ausdrucksfähigkeit bzw. "Berechnungsmächtigkeit" ist begrenzt.
- Schon sehr einfache "Zählaufgaben" können sie nicht wahrnehmen.

  Beispiel: $L = \{a^n b^n \mid n \geq 0\}$ kann von einem endlichen Automaten nicht erkannt werden, dazu müßte er im Prinzip n (also ggf. unbeschränkt viele) Zustände
- In arithmetischen Ausdrücken bzw. in Programmiersprachen treten überall balancierte Klammerausdrücke auf, z. B. sei a = "(" und b = ")" oder ; { }; BEGIN...END.

  Um mit solchen Ausdrücken umgehen zu können, muß der Akzeptor zählen können, wieviele öffnende Klammern vorliegen, um n zu bestimmen, danach die zwischen den Klammern stehenden Ausdrücke verarbeiten und dann bestimmen, ob n schließende Klammern folgen.

→ Der Automat muß also Zwischeninformation speichern können, bis sie bei der Analyse

von weiteren Zeilen auch verwendet werden kann.

Sprachen, die diese Strukturen beschreiben und Automaten, die ~~diese~~ Wörter dieser Sprachen akzeptieren, sind die kontext-freien Sprachen und die Kellerautomaten (PDA: push-down-automata).