Automatentheorie und Formale Sprachen

Skript zur Vorlesung im SS 2000 von Prof. Dr. Franz Baader

geLaTeXt von

Claus Richterich
Claus@Richterich.net

Diego Biurrun
diego@biurrun.de

Stefan Jacobs
Stefan.Jacobs@post.rwth-aachen.de

Stefan Schiffer
dr.stf@web.de

Thomas Deselaers
Thomas@Deselaers.de

Tran Huy Nguyen
TranHuy.Nguyen@gmx.net

Vorschau

ATFS: Teil der Theoretischen Informatik, beschäftigt sich mit Wörtern und Mengen von Wörtern (formale Sprachen).

Wichtige Fragestellungen

- Wie beschreibt man die (meist unendlichen) Mengen mit endlichem Aufwand?
  - Automaten oder Maschinen, die genau die Elemente der Menge akzeptieren (vgl. BuK, endliche Automaten, Turingmaschinen).
  - Grammatiken, die die Elemente der Menge generieren (vgl. Programmierung, kontextfreie Grammatiken).
  - Ausdrücke, welche beschreiben, wie man die Sprache aus Basissprachen mit Hilfe von Operationen ($\cap$, $\cup$, $^-$) erzeugen kann.

    Abhängig vom verwendeten Automaten- bzw Grammatiktyp bekommt man verschiedene Klassen von Sprachen.

Klasse	Automatentyp	Grammatiktyp
Typ0	Turingmaschine	allgemeine Chomsky-grammatik
Typ1	linear bandbeschränkte Turingmaschine	kontextsensitive Grammatik
Typ2	Kellerautomat	kontextfreie Grammatik
Typ3	endlicher Automat	einseitig lineare Grammatik

    Typ2 und Typ3 sind die wichtigsten Typen. Z.B. können Typ2 Sprachen weitgehend die Syntax von Programmiersprachen definieren.

- Welche Problemstellungen sind für eine Sprachklasse entscheidbar? Wenn ja, mit welchem Aufwand?
  - Wortproblem: Gegeben eine Beschreibung der Sprache L (Automat, Grammatik) und gegeben Wort w. Gilt $w \in L$?

    Anwendungsbeispiele:

    - Programmiersprache, deren Syntax durch eine kontextfreie Grammatik gegeben ist. Entscheide, ob ein Programm syntaktisch korrekt ist.
    - Suchpattern für Textdateien sind häufig reguläre Ausdrücke (beschreiben Typ3 Sprachen). Suche die Dateien ($\hat{=}$ Wörter), welche das Pattern enthalten, d.h. die zur beschriebenen Sprache gehören.
  - Leerheitsproblem: Gegeben eine Beschreibung einer Sprache L. Gilt $L \neq \emptyset$?

    Anwendungsbeispiel:
    - Wenn ein Suchpattern die leere Sprache beschreibt, so muß man erst gar nicht danach suchen.
  - Äquivalenzproblem: Beschreiben zwei verschiedene Beschreibungen dieselbe Sprache?

    Anwendungsbeispiel:
    - Zwei Lehrbücher über eine Programmiersprache beschreiben die Syntax mit verschiedenen Grammatiken. Sind diese wirklich äquivalent?

- Welche Abschlußeigenschaften hat eine Sprachklasse? z.B. Abschluß unter $\cap, \cup, \bar{}$

  L_1, L_2 drin $\Rightarrow$ auch $L_1 \cap L_2$, $L_1 \cup L_2$, $\bar{L_1}$ drin?

  Anwendungsbeispiele:
  - Suchpattern: Suche nach Dateien, in denen ein Pattern nicht vorkommt.
  - Reduziere das Äquivalenzproblem auf das Leerheitsproblem. $L_1 = L_2$ gdw $(L_1 \cap \bar{L_2}) \cup (\bar{L_1} \cap L_2) = \emptyset$.

Inhaltsverzeichnis

Teil I

Endliche Automaten und reguläre Sprachen

0 Einführung

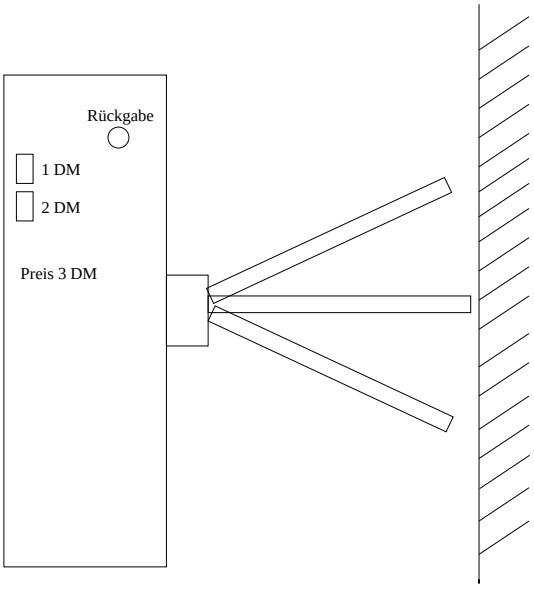
Endliche Automaten sind gekennzeichnet durch:

- endliche Menge von (internen) Zuständen
- Übergänge zwischen Zuständen abhängig von der internen Struktur des Automaten und von der Eingabe

Im Prinzip sind Rechner auch endliche Automaten: Sie haben endlichen Speicherplatz und daher nur eine endliche Menge von Konfigurationen (Speicherplatzbelegungen).

Aber wegen der extrem großen Anzahl von Zuständen sind endliche Automaten kein geeignetes Beschreibungsmittel für Rechner. Man abstrahiert daher von der Speicherplatzbeschränkung und nimmt potentiell unendlichen Speicherplatz an (z.B. bei Turingmaschinen das unendliche Band).

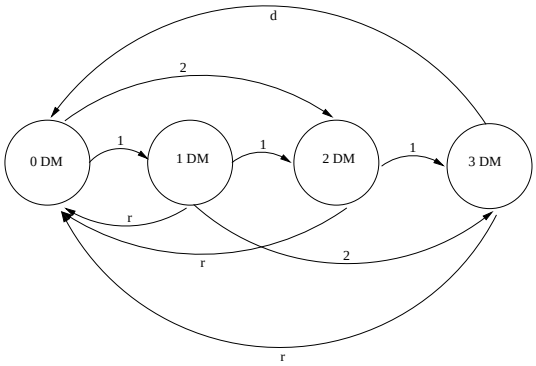
Beispiel für endlichen Automaten, bei dem diese Beschreibung adäquat ist:



Eingaben: 1,2,r,d (Drehkreuz dreht sich)

Zustände: 0 DM, 1 DM, 2 DM, 3 DM

Zustandsübergänge:



Welche Eingabefolgen führen vom Zustand 0 DM zum Zustand 3 DM?

Teilweise interessiert man sich auch für Automaten mit Ausgabe (z.B. Rückgabe des Geldes, Freigabe der Sperre). Wir werden uns hier nicht damit beschäftigen.

Definition 0.1 (Grundbegriffe)

Alphabet Σ

endliche Menge von Buchstaben a,b,c, ...

Wort w über dem Alphabet Σ

endliche Folge von Buchstaben $w = a_1a_2a_3a_4 \dots a_n$ mit $a_i \in \Sigma$

Länge $\mid w \mid$ des Wortes w

$w = a_1 \dots a_n$ $|w| = n$ z.B. $|aba| = 3$

Anzahl der Vorkommen $\mid w \mid_x$ des Buchstabens x in w

z.B. $|aba|_a = 2$

leeres Wort ε

hat Länge 0

Σ^*

Menge aller Wörter

Σ^+

$= \Sigma^* \setminus \{\varepsilon\}$ (alle nichtleeren Wörter)

formale Sprache L

Menge von Wörtern

$L \subseteq \Sigma^*$ ist formale Sprache über Σ.

z.B.:

- Menge der Wörter über dem Alphabet $\{a \dots z\}$, die korrekte Wörter der deutschen Sprache sind.
- Menge der Wörter über dem Alphabet $\{a \dots z\}$, die das Wort "Theorie" enthalten.

Definition 0.2 (Operationen auf Sprachen und Wörtern)

Boolesche Operationen

$L_1 \cup L_2$

$L_1 \cap L_2$

$\bar{L_1} := \Sigma^* \setminus L_1$

$L_1 \setminus L_2 := L_1 \cap \bar{L_2}$

Konkatenation

$u \cdot v := uv$ z.B. $abb \cdot ab = abbab$

$L_1 \cdot L_2 := \{uv \mid u \in L_1 \text{ und } v \in L_2\}$

Konkatenationspunkt "$\cdot$" wird meist weggelassen.

Kleene Stern iterierte Konkatenation

$L^0 = \{\varepsilon\}$

$L^{n+1} = L^n \cdot L$

$L^* = \bigcup_{n \geq 0} L^n$

$L^+ = \bigcup_{n \geq 1} L^n$

Präfix, Suffix, Infix

u ist Präfix von v, wenn $v = uw$ für ein $w \in \Sigma^*$

u ist Suffix von v, wenn $v = wu$ für ein $w \in \Sigma^*$

u ist Infix von v, wenn $v = w_1 u w_2$ für $w_1, w_2 \in \Sigma^*$

1 Nichtdeterministische endliche Automaten

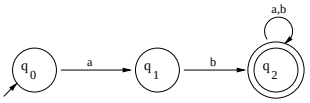
Definition 1.1 Ein Transitionssystem ist von der Form $A = (Q, \Sigma, I, \Delta, F)$, wobei:

- Q Menge von Zuständen (beliebig groß, auch unendlich)
- Σ endliches Alphabet
- $I \subseteq Q$ Anfangszustände
- $\Delta \subseteq Q \times \Sigma \times Q$
- $F \subseteq Q$ Endzustände

endlicher Automat: endlich viele Zustände, nur ein Anfangszustand

Definition 1.2 Das Transitionssystem $A = (Q, \Sigma, I, \Delta, F)$ heißt nichtdeterministischer endlicher Automat (NEA), wenn $|Q| < \infty$ und $|I| = 1$ anstelle von $(Q, \Sigma, \{q_0\}, \Delta, F)$ schreiben wir $(Q, \Sigma, q_0, \Delta, F)$.

Beispiel 1.3 $A = (Q, \Sigma, q_0, \Delta, F)$ mit $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $F = \{q_2\}$, $\Delta = \{(q_0, a, q_1), (q_1, b, q_2), (q_2, a, q_2), (q_2, b, q_2)\}$ wird graphisch dargestellt durch



Definition 1.4 Ein Pfad in einem Transitionssystem ist eine Folge

$$\pi = (p_0, a_1, p_1)(p_1, a_2, p_2) \dots (p_{n-1}, a_n, p_n)$$

mit $(p_i, a_{i+1}, p_{i+1}) \in \Delta$ für $i = 0, \dots, n-1$. π ist ein Pfad von p nach q gdw $p_0 = p$ und $p_n = q$. Die Beschriftung von π ist das Wort $\beta(\pi) = a_1 \dots a_n$.

Länge von π: n

Spezialfall: $n = 0$ leerer Pfad mit Beschriftung ε

Notation:

$p \xrightarrow{w}_A q$**:** Es gibt einen Pfad von p nach q in A mit Beschriftung w.

$Q_1 \xrightarrow{w}_A Q_2$: $Q_1, Q_2 \subseteq Q$ und es gibt $q_1 \in Q_1$ und $q_2 \in Q_2$ mit $q_1 \xrightarrow{w}_A q_2$.

Beispiel: Für den NEA aus Beispiel 1.3 gilt:

$q_0 \xrightarrow{abw}_A q_2$ für alle $w \in \{a, b\}^*$

Definition 1.5 Das Transitionssystem $A = (Q, \Sigma, q_0, \Delta, F)$ akzeptiert das Wort $w \in \Sigma^*$ gdw $I \xrightarrow{w}_A F$. Die von A akzeptierte Sprache (erkannte Sprache) ist

$$L(A) := \{w \in \Sigma^* \mid A \text{ akzeptiert } w\}$$

Beispiel: NEA aus Beispiel 1.3

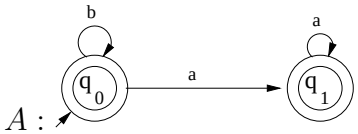
$L(A) = \{abw \mid w \in \{a, b\}^*\}$

Definition 1.6 Eine Sprache $L \subseteq \Sigma^*$ heißt erkennbar gdw es einen NEA A gibt mit $L(A) = L$.

Äquivalenz von Transitionssystemen:

A_1 und A_2 heißen äquivalent gdw $L(A_1) = L(A_2)$, d.h. sie akzeptieren dieselbe Sprache.

Beispiel 1.7 $L = \{w \in \{a, b\}^* \mid ab$ ist nicht Infix von $w\}$ ist erkennbar:



akzeptiert L

Für manche Konstruktionen ist es günstiger, auch Übergänge mit Wörtern zuzulassen.

Definition 1.8

- Ein NEA mit Wortübergängen hat die Form $A = (Q, \Sigma, q_0, \Delta, F)$, wobei Q, Σ, q_0, F wie beim NEA und $\Delta \subseteq Q \times \Sigma^* \times Q$ eine endliche Menge von Übergängen.
- Ein ε-NEA ist ein NEA mit Wortübergängen, wobei für alle $(q, w, q') \in \Delta$ gilt:

  $|w| \leq 1$, d.h. $w \in \Sigma$ oder $w = \varepsilon$

Pfade, Pfadbeschriftungen und akzeptierte Sprachen werden wie bei NEAs definiert.

Z.B. hat der Pfad $(q_0, ab, q_1)(q_1, \varepsilon, q_2)(q_2, bb, q_3)$ die Beschriftung $ab \cdot \varepsilon \cdot bb = abbb$

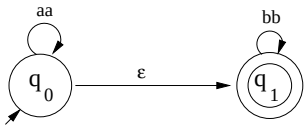
Satz 1.9 Zu jedem NEA mit Wortübergängen kann man effektiv (d.h. man kann diesen NEA berechnen) einen äquivalenten NEA konstruieren.

Wir zeigen den Satz durch Umweg über ε-NEA:

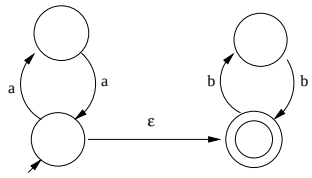
Lemma 1.10 Zu jedem NEA mit Wortübergängen kann man effektiv einen äquivalenten ε-NEA konstruieren.

Beweis: $p \overset{a_1 \ldots a_n}{\longrightarrow} q$ für $n > 1$ wird ersetzt durch $p \overset{a_1}{\to} p_1 \overset{a_2}{\to} p_2 \to \ldots p_{n-1} \overset{a_n}{\to} q$, wobei $p_1, \ldots, p_{n-1}$ jeweils neue Zustände sind. Man sieht leicht, daß dies einen äquivalenten ε-NEA liefert.

Beispiel 1.11



wird zu



Lemma 1.12 Zu jedem ε-NEA kann man effektiv einen äquivalenten NEA konstruieren.

Beweis: Der ε-NEA $A = (Q, \Sigma, q_0, \Delta, F)$ sei gegeben. Wir konstruieren daraus einen NEA $A' = (Q, \Sigma, q_0, \Delta', F')$, wobei

- $\Delta' := \{(p, a, q) \in Q \times \Sigma \times Q \mid p \overset{a}{\longrightarrow}_A q\}$
- $F' := \begin{cases} F \cup \{q_0\} & \text{falls} \quad q_0 \overset{\varepsilon}{\longrightarrow}_A F \\ F & \text{sonst} \end{cases}$

zu zeigen:

1. A' kann effektiv bestimmt werden.
2. $L(A) = L(A')$

zu 1. Δ' und F' können effektiv bestimmt werden.

- $p \xrightarrow{a}_A q$ gdw $\exists p', q' \in Q$ mit $p \xrightarrow{\varepsilon}_A p'$, $(p', a, q') \in \Delta$, $q' \xrightarrow{\varepsilon}_A q$.
  Es bleibt zu zeigen: $p \xrightarrow{\varepsilon}_A p'$ kann effektiv entschieden werden. Dies ist ein Erreichbarkeitsproblem in einem endlichen Graphen, mit Aufwand $O(|Q| + |\Delta|)$ entscheidbar (siehe Vorlesung Datenstrukturen und Algorithmen).
- $q_0 \xrightarrow{\varepsilon} F$ ist ebenfalls ein Erreichbarkeitsproblem.

zu 2.

- $L(A) \subseteq L(A')$:

  Sei $w \in L(A), w = a_1 \dots a_n$. Sei $\pi = (p_0, a_1, p_1)(p_1, a_2, p_2) \dots (p_{n-1}, a_n, p_n)$ ein Pfad mit $p_0 = q_0$ und $p_n \in F$ und $a_i \in \Sigma \cup \{\varepsilon\}$, wobei $w = a_1 \dots a_n$. Seien $i_1 \dots i_m$ Indizes mit $a_{ij} \neq \varepsilon$. Offenbar gilt $w = a_{i1} \dots a_{im}$.

  1.Fall: Es gibt keinen solchen Index, d.h. $w = \varepsilon$.
  Es gilt also $q_0 = p_0 \xrightarrow{\varepsilon}_A p_n \in F$. Damit ist $q_0 \in F'$ und somit $\varepsilon \in L(A')$.

  2.Fall: $m > 0$. Nach Definition von Δ' ist $(p_0, a_{i1}, p_{i1})(p_{i1}, a_{i2}, p_{i2}) \dots (p_{i_{m-1}}, a_{im}, p_n)$ ein Pfad in A'.

- $L(A') \subseteq L(A)$:

  Es sei $w = a_1 \dots a_n \in L(A')$ mit $a_i \in \Sigma$. Sei $(p_0, a_1, p_1) \dots (p_{n-1}, a_n, p_n)$, wobei $p_0 = q_0$ und $p_n \in F'$.

  Nach Definition von Δ' gilt: $p_i \xrightarrow{a_{i+1}}_A p_{i+1}$. Also gibt es in A einen Pfad von $p_0 = q_0$ nach p_n mit Beschriftung w.

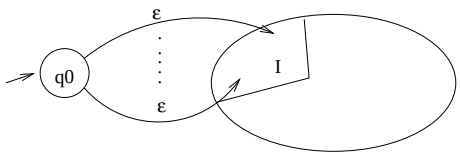
  1.Fall: $p_n \in F$ Dann ist $w \in L(A)$

  2.Fall: $p_n \in F' \setminus F$, d.h. $p_n = q_0$ und $p_n = q_0 \xrightarrow{\varepsilon}_A \hat{q} \in F$ Damit gibt es einen Pfad mit Beschriftung w von q_0 nach $\hat{q}$. Damit folgt wieder $w \in L(A)$.

□

Anmerkung: Zu jedem endlichen Transitionssystem kann man effektiv einen äquivalenten ε-NEA konstruieren.

Gegeben sei das endliche Transitionssystem $A = (Q, \Sigma, I, \Delta, F)$. Wir definieren daraus den ε-NEA $A' = (Q \cup \{q_0\}, \Sigma, q_0, \Delta', F)$,

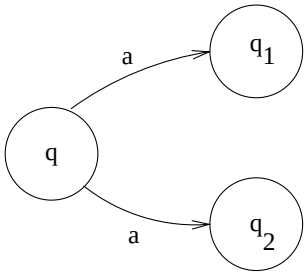


wobei $q_0 \notin Q$ und $\Delta' := \{(q_0, \varepsilon, q) \mid q \in I\} \cup \Delta$.

A' ist offenbar äquivalent zu A und ein ε-NEA.

2 Deterministische endliche Automaten

Was heißt bei NEAs nichtdeterministisch?

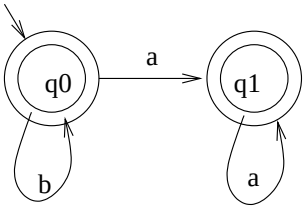


Für ein gegebenes Paar $(q, a) \in Q \times \Sigma$ kann es mehr als ein q_i geben, mit $(q, a, q_i) \in \Delta$.

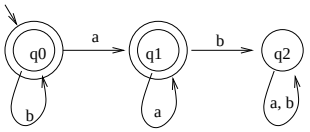
Definition 2.1 Ein NEA $A = (Q, \Sigma, q_0, \Delta, F)$ heißt deterministischer endlicher Automat (DEA), falls es für alle $(q, a) \in Q \times \Sigma$ genau ein $q' \in Q$ gibt mit $(q, a, q') \in \Delta$.
Anstelle der Übergangsrelation Δ schreibt man die Übergangsfunktion
$\delta : Q \times \Sigma \to Q$ mit $\delta(q, a) = q'$ gdw $(q, a, q') \in \Delta$. DEAs werden damit in der Form $(Q, \Sigma, q_0, \delta, F)$ geschrieben.

Beachte: Zusätzlich zur Eindeutigkeit (es gibt höchstens ein q' mit $(q, a, q') \in \Delta$) fordern wir auch die Existenz des Übergangs (d.h. es gibt mindestens ein q' mit $(q, a, q') \in \Delta$). In der Literatur wird manchmal bei DEAs nur Eindeutigkeit gefordert. Das macht bzgl. der definierten Sprachklasse keinen Unterschied.

Beispiel 2.2

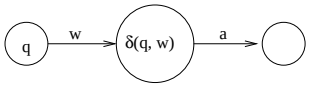


Dieser Automat erfüllt zwar die Eindeutigkeitsforderung, aber zu (q_1, b) existiert kein Übergang! Ein äquivalenter DEA entsteht durch Hinzunahme eines "Papierkorbzustandes" q_2:



Definition 2.3 Die kanonische Fortsetzung $\delta^* : Q \times \Sigma^* \to Q$ von $\delta : Q \times \Sigma \to Q$ wird induktiv definiert:

- $\delta^*(q, \varepsilon) := q$
- $\delta^*(q, wa) := \delta(\delta^*(q, w), a)$



Der Einfachheit halber schreiben wir im folgenden auch für $w \in \Sigma^*$ anstatt $\delta^*(q, w)$ einfacher $\delta(q, w)$.

Beachte: Für einen DEA $A = (Q, \Sigma, q_0, \delta, F)$ gilt:

- $\delta(q, w) = q'$ gdw q' der eindeutige Zustand mit $q \xrightarrow{w}_A q'$ ist.
- $L(A) := \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$
- $\delta(q, uv) = \delta(\delta(q, u), v)$

Übung: Beweise die einzelnen Aussagen!

Satz 2.4 (Satz von Rabin/Scott) Zu jedem NEA kann man effektiv einen äquivalenten DEA konstruieren.

Das bedeutet, daß die Einschränkung "deterministisch" keine echte ist.

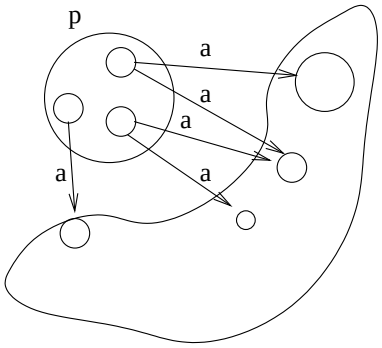
Beweisidee: Sei $A = (Q, \Sigma, q_0, \Delta, F)$ ein NEA. Für $w \in \Sigma^*$ gilt: $w \in L(A)$ gdw $\{q \in Q \mid q_0 \xrightarrow{w}_A q\} \cap F \neq \emptyset$ mit $\{q \in Q \mid q_0 \xrightarrow{w}_A q\} \subseteq 2^Q = \{P \mid P \subseteq Q\}$

Baue einen Automaten mit Zustandsmenge 2^Q und Übergangsfunktion δ, für die gilt:

$$\delta(\{q_0\}, w) = \{q \in Q \mid q_0 \xrightarrow{w}_A q\}$$

Beweis: Man verwende die folgende Potenzmengenkonstruktion: Sei $A = (Q, \Sigma, q_0, \Delta, F)$ ein NEA. Der DEA $A' := (2^Q, \Sigma, \{q_0\}, \delta, F')$ ist wie folgt definiert:

- $\delta(P \in 2^Q, a \in \Sigma) := \bigcup_{p \in P} \{p' \mid (p, a, p') \in \Delta\}$



- $F' := \{P \in 2^Q \mid P \cap F \neq \emptyset\}$

Behauptung: $q' \in \delta(\{q\}, w)$ gdw $q \xrightarrow{w}_A q'$. Daraus folgt unmittelbar $L(A) = L(A')$, da:

$w \in L(A)$ gdw $\exists q \in F : q_0 \xrightarrow{w}_A q$

gdw $\exists q \in F : q \in \delta(\{q_0\}, w)$

gdw $\delta(\{q_0\}, w) \cap F \neq \emptyset$

gdw $\delta(\{q_0\}, w) \in F'$

gdw $w \in L(A')$

Beweis der Behauptung: Induktion über |w|

- $|w| = 0$: $\delta(\{q\}, \varepsilon) = \{q\}$ (nach Definition 2.3) Es gilt: $q \xrightarrow{\varepsilon}_A q$ und q ist der einzige Zustand mit dieser Eigenschaft (es gibt kein $q' \neq q$ mit $q \xrightarrow{\varepsilon} q'$).

- $|w| = n + 1$: Für Wörter der Länge n gelte die Behauptung bereits.
  $w = ua$ mit $a \in \Sigma, u \in \Sigma^*, |u| = n$.

  $\delta(\{q\}, ua) \overset{\text{Def 2.3}}{=} \delta(\delta(\{q\}, u), a) \overset{\text{Def v. } \delta}{=} \bigcup_{q' \in \delta(\{q\},u)} \{q'' \mid (q', a, q'') \in \Delta\}$

  Zu zeigen: Ein Zustand q'' gehört zu $\bigcup_{q' \in \delta(\{q\},u)} \{q'' \mid (q', a, q'') \in \Delta\}$ gdw $q \xrightarrow{ua}_A q''$.

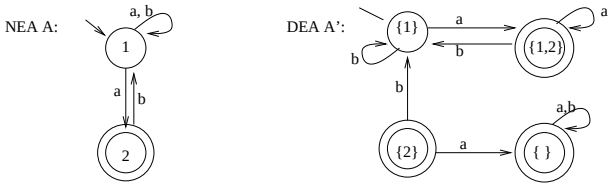
  $q \xrightarrow{ua}_A q''$ gdw $\exists q' : q \xrightarrow{u}_A q'$ und $(q', a, q'') \in \Delta$

  gdw $q' \in \delta(\{q\}, u)$

  gdw $q'' \in \bigcup_{q' \in \delta(\{q\},u)} \{q'' \mid (q', a, q'') \in \Delta\}$

□

Beispiel 2.5



Problem: Potenzmengenkonstruktion vergrößert die Zustandsmenge exponentiell.

Im Allgemeinen kann man das nicht vermeiden. In vielen Fällen kann man aber kleinere äquivalente DEAs konstruieren.

Allgemein: Wie konstruiere ich zu einem gegebenen DEA einen DEA mit minimaler Zustandszahl?

Definition 2.6 Ein Zustand q eines DEA $A = (Q, \Sigma, q_0, \delta, F)$ heißt erreichbar, falls es ein $w \in \Sigma^*$ gibt, mit $\delta(q_0, w) = q$. Sonst heißt q unerreichbar. Für die akzeptierte Sprache sind nur die erreichbaren Zustände relevant.

NEA $\overset{\text{Satz 2.4}}{\longrightarrow}_A$ DEA (kann exponentiell groß werden)

Minimieren von DEA: 2 Schritte

1. Schritt: Eliminieren unerreichbarer Zustände.

Da für die akzeptierte Sprache nur erreichbare Zustände relevant sind, kann man unerreichbare einfach weglassen. Dies liefert den äquivalenten Automaten:

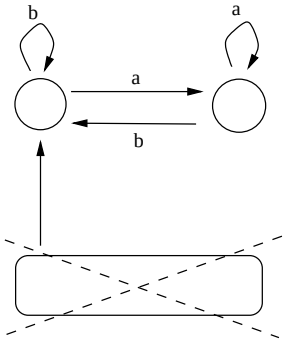
$A_0 := (Q_0, \Sigma, q_0, \delta_0, F_0)$ mit

$Q_0 := \{q \in Q \mid q\text{erreichbar}\}$,

$\delta_0 := \delta \mid_{Q_0 \times \Sigma}$ (Beachte: Für $q \in Q_0$ und $a \in \Sigma$ ist auch $\delta(q, a) \in Q_0$.),

$F_0 := F \cap Q_0$.

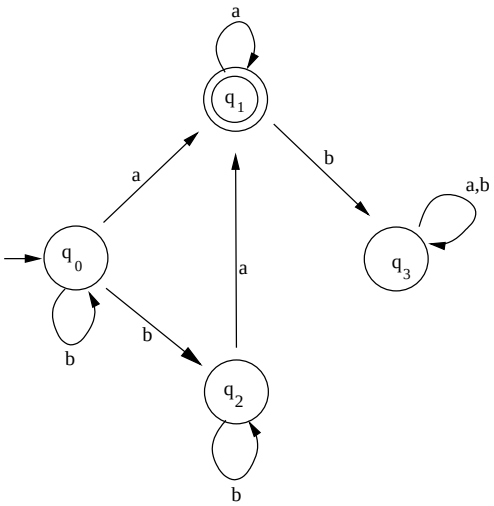
Beispiel: Im Automaten A' von Beispiel 2.5 sind die Zustände $\{2\}$, $\emptyset$ unerreichbar. Durch Weglassen erhält man



Bei der Potenzmengenkonstruktion kann man sich auf die Berechnung der erreichbaren Zustände beschränken. Beginne mit $\{q_0\}$ und konstruiere daraus die erreichbaren Mengen.

2. Schritt: Zusammenfassen äquivalenter Zustände.

Beispiel 2.7



Hier sind alle Zustände erreichbar. Dieser Automat ist nicht minimal, da er dieselbe Sprache erkennt, wie der DEA aus Beispiel 2.2. Dies kommt daher, daß q_0 und q_2 äquivalent sind.

Definition 2.8 Es sei $A = (Q, \Sigma, q_0, \delta, F)$ ein DEA. Für $q \in Q$ sei $A_q := (Q, \Sigma, q, \delta, F)$. Zwei Zustände $q, q' \in Q$ heißen A-äquivalent ($q \sim_A q'$) gdw $L(A_q) = L(A_{q'})$.

Im Beispiel 2.7 gilt: $q_0 \sim_A q_2$.

Lemma 2.9

1. $\sim_A$ ist eine Äquivalenzrelation auf Q, d.h. reflexiv, transitiv, symmetrisch.
2. $\sim_A$ ist verträglich mit δ, d.h. $q \sim_A q' \Rightarrow \forall a \in \Sigma : \delta(q, a) \sim_A \delta(q', a)$.
3. $\sim_A$ kann effektiv bestimmt werden.

Beweis:

1. Klar, da "=" reflexiv, transitiv und symmetrisch ist.
2. $q \sim_A q'$ gdw $L(A_q) = L(A_{q'})$
   gdw $\delta(q, w) \in F \Leftrightarrow \delta(q', w) \in F \Rightarrow \forall a \in \Sigma, \forall v \in \Sigma^* : \delta(q, av) \in F \Leftrightarrow \delta(q', av) \in F$
   gdw $\forall a \in \Sigma, \forall v \in \Sigma^* : \delta(\delta(q, a), v) \in F \Leftrightarrow \delta(\delta(q', a), v) \in F$
   gdw $\forall a \in \Sigma : L(A_{\delta(q,a)}) = L(A_{\delta(q',a)})$
   gdw $\forall a \in \Sigma : \delta(q, a) \sim_A \delta(q', a)$
3. Wir definieren Approximationen $\sim_k$ (k natürliche Zahl) von $\sim_A$:
   (a) $q \sim_0 q'$ gdw $q \in F \Leftrightarrow q' \in F$
   (b) $q \sim_{k+1} q'$ gdw $q \sim_k q'$ und $\forall a \in \Sigma : \delta(q, a) \sim_k \delta(q', a)$

Es gilt: $Q \times Q \supseteq \sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \ldots \overset{!}{\supseteq} \sim_A$. Da Q endlich ist, gibt es ein k mit $\sim_k = \sim_{k+1}$. Man zeigt leicht, daß dann $\sim_k = \sim_A$.

Die $\sim_A$-Äquivalenzklasse eines Zustandes $q \in Q$ bezeichnen wir mit $\tilde{q} := \{q' \mid q \sim_A q'\}$.

Beispiel: Automat aus Beispiel 2.7

$\sim_0$: zwei Klassen: $F = \{q_1\}$, $Q \setminus F = \{q_0, q_2, q_3\}$

$\sim_1$: drei Klassen: $\{q_1\}$, $\{q_0, q_2\}$, $\{q_3\}$

$\sim_1 = \sim_2 = \sim_A$

Definition 2.10 Der Quotientenautomat $\tilde{A} := (\tilde{Q}, \Sigma, \tilde{q_0}, \tilde{\delta}, \tilde{F})$ ist definiert durch:

- $\tilde{Q} = \{\tilde{q} \mid q \in Q\}$
  $\sim_A$-Äquivalenzklasse q nach Definition von $\sim_A$
- $\tilde{F} = \{\tilde{q} \mid q \in F\}$ repräsentantenunabhängig! nach Definition von $\sim_A$
- $\tilde{\delta}(\tilde{q}, a) = \widetilde{\delta(q, a)}$
- $\tilde{q_0} = \{q \mid q \sim_A q_0\}$
- $\tilde{q} = \{q, q', q'', \ldots\}$

Sei $\underbrace{q \in F}_{\varepsilon \in L(A,q)} \wedge \underbrace{q' \notin F}_{\varepsilon \notin L(A,q)}$

$\Rightarrow q \not\sim_A q'$

Lemma 2.11 $\tilde{A}$ ist äquivalent zu A.

Beweis: $w \in L(A)$ gdw $\delta(q_0, w) \in F$

gdw $\widetilde{\delta(q_0, w)} \in \tilde{F}$

gdw $\underbrace{\tilde{\delta}(\tilde{q_0}, w) \in \tilde{F}}_{\text{Beh: } \widetilde{\delta(q_0,w)} = \tilde{\delta}(\tilde{q},w)}$

gdw $w \in L(\tilde{A})$

□

siehe Beispiel 2.7

vgl. Beispiel 2.2

Definition 2.12 Für einen DEA A bezeichne $A_{red} = \tilde{A_0}$ den reduzierten Automaten, den man aus A durch eliminieren unerreichbarer Zustände (A_0) und zusammenfassen äquivalenter Zustände von A_0 erhält.

Wir werden zeigen:

- A_{red} kann nicht noch kleiner gemacht werden, d.h. er ist ein Automat minimaler Zustandszahl der $L(A)$ erkennt.
- A_{red} hängt nur von $L(A)$ und nicht von A ab, d.h. gilt $L(A) = L(B)$, so gilt $\underbrace{A_{red} \cong B_{red}}_{\text{isomorph}}$, wobei isomorph gleich bis auf Zustandsumbenennung bedeutet.

  Wir konstruieren zur Sprache L einen "kanonischen" Automaten A_L und zeigen, daß $A_{red} \cong A_L$ für alle Automaten A mit $L = L(A)$.

Definition 2.13 (Nerode Rechtskongruenz) Es sei $L \subseteq \Sigma^*$ eine beliebige Sprache. Für $u, v \in \Sigma^*$ definieren wir $u \cong_L v$ (u und v sind kongruent L) gdw $\forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L$.

Beispiel 2.14 Vergleiche: 1.7 2.2 2.7

$L = \{w \in \{a,b\}^* \mid ab \text{ ist nicht Infix von } w\}$

- $\varepsilon \cong_L b : \forall w \quad \varepsilon w \in L$
  gdw $w \in L$
  gdw w enthält ab nicht
  gdw bw enthält ab nicht
  gdw $bw \in L$
- $\varepsilon \ncong_L a : \varepsilon b \in L,$ aber $ab \notin L$

Lemma 2.15 (Eigenschaften von $\cong_L$)

1. $\cong_L$ ist Äquivalenzrelation.
2. $\cong_L$ ist rechtskongruent, d.h zusätzlich zu 1. gilt $u \cong_L v \Rightarrow uw \cong_L vw$ für alle $w \in \Sigma^*$.
3. L ist Vereinigung von $\cong_L$-Klassen:
   $L = \bigcup_{u \in L}[u]$, wobei $[u] := \{v \mid u \cong_L v\}$
4. Ist $L = L(A)$ für einen DEA, so gilt:
   Anzahl der $\cong_L$-Klassen, Index von $\cong_L$ $\quad \leq \quad$ Zustandsanzahl von A.

Beweis:

1. Klar, da $\Leftrightarrow$ reflexiv, transitiv und symmetrisch ist.
2. Damit $uw \cong_L vw$ gilt, muß für alle $w' \in \Sigma^*$ gelten:
   $u \underbrace{ww'}_{\in\Sigma^*} \in L \quad$ gdw $\quad v \underbrace{ww'}_{\in\Sigma^*} \in L$
   Da $ww' \in \Sigma^*$ ist, folgt das aus $u \cong_L v$
3. $L = \bigcup_{u \in L}[u]$
   "$\subseteq$" trivial, da $u \in [u]$
   "$\supseteq$" Zu zeigen: Ist $u \in L$ und $v \in [u]$, so auch $v \in L$.
   Wegen $u \cong_L v$ folgt $u\varepsilon \in L$ gdw $v\varepsilon \in L$.

4. Es sei $A = (Q, \Sigma, q_0, \delta, F)$ ein DEA mit $L = L(A)$.

   Behauptung: $\delta(q_0, u) = \delta(q_0, v) \Rightarrow u \cong_L v$

   denn: $\forall w : uw \in L$
   gdw $\delta(q_0, uw) \in F$
   gdw $\delta(\delta(q_0, u), w) \in F$
   $\delta(q_0, u) \overset{\Uparrow}{=} \delta(q_0, v)$
   gdw $\delta(\delta(q_0, v) \overset{\Downarrow}{,} w) \in F$
   gdw $\delta(q_0, vw) \in F$
   gdw $vw \in L$
   d.h. Behauptung schon gezeigt

   Annahme: Es gibt mehr $\cong_L$-Klassen als Zustände in Q.
   $|Q| = n$. Dann gibt es mindestens $n + 1$ Wörter
   $u_1 \dots u_{n+1}$ mit $u_i \ncong_L u_j$ für $i \neq j$

   Betrachte $\delta(q_0, u_1), \dots \delta(q_0, u_{n+1})$, da $|Q| = n$ gibt es $i \neq j$ mit
   $\delta(q_0, u_1) = \delta(q_0, u_j) \Rightarrow$ Behauptung $u_i \cong_L u_j$ Widerspruch! $\Rightarrow$
   Behauptung!

□

Beispiel 2.14 (Fortsetzung):

$\cong_L$ hat drei Klassen.

$L = \cup \begin{cases} [\varepsilon] = \{b\}^* \rightarrow & \text{Um } L \text{ zu verlassen, muß } w\ ab \text{ enthalten.} \\ [a] = \{b^*\}\{a\}^+ \rightarrow & \text{Um } L \text{ zu verlassen, muß } w\ ab \text{ enthalten, oder mit } b \text{ beginnen.} \end{cases}$

$[ab] = \underbrace{\Sigma^*\{ab\}\Sigma^*}_{\bar{L}} \rightarrow$ nie in L

Man kann die $\cong_L$-Klassen als Zustände eines Automaten für L auffassen.

Definition 2.16 Zu $L \subseteq \Sigma^*$ ist das Transitionssystem $A_L = (Q', \Sigma, q_0', \delta', F')$ definiert durch:

- $Q' := \{[u] \mid u \in \Sigma^*\}$
- $q_0' := [\varepsilon]$
- $\delta'([u], a) = [ua]$ repräsentantenunabhängig wegen Lemma 2.15

Lemma 2.17 Hat $\cong_L$ endlichen Index, so ist A_L DEA mit $L = L(A_L)$.

Beweis: Da A_L dann nur endlich viele Zustände hat, ist A_L offenbar ein DEA.

$$L(A_L) = \{u \mid \delta'(q_0', u) \in F'\}$$
$$= \{u \mid \delta([\varepsilon], u) \in F'\}$$
$$= \underbrace{\{u \mid [u] \in F'\}}$$

da $\delta'([u], v) = [uv]$ mit Ind. |v|

$$= \{u \mid u \in L\} = L$$

□

Satz 2.18 (Satz von Nerode) Eine Sprache L ist erkennbar gdw $\cong_L$ endlichen Index hat.

Beweis:

"$\Rightarrow$" Ist L erkennbar, so gibt es einen DEA A mit $L = L(A)$. Mit Lemma 2.15 ist der Index $n \cong_L \leq$ Anzahl der Zustände von A, also endlich.

"$\Leftarrow$" Hat $\cong_L$ endlichen Index, so ist mit Lemma 2.17 A_L ein DEA mit $L = L(A_L)$. Also ist L erkennbar.

□

Beispiel 2.19 (nicht erkennbare Sprache) $L = \{a^n b^n \mid n \geq 0\}$ ist nicht erkennbar, denn für $n \neq m$ gilt $a^n \ncong_L a^m$. Es gilt ja $a^n b^n \in L$ und $a^m b^n \notin L$. Zusammenhang zwischen reduzierten Automaten und der Nerode-Rechtskongruenz.

Definition 2.20 Zwei DEAs $A = (Q, \Sigma, q_0, \delta, F)$ und $A' = (Q', \Sigma, q_0', \delta', F')$ sind isomorph $(A \cong A')$ gdw es eine Bijektion $f : Q \to Q'$ gibt mit

- $f(q_0) = q_0'$
- $F' = f(F) = \{f(q) \mid q \in F\}$
- $f(\delta(q, a)) = \delta'(f(q), a)$

Lemma 2.21 $A \cong A' \Rightarrow L(A) = L(A')$

Beweis: Man zeigt leicht durch Induktion über $|w|$, daß $f(\delta(q, w)) = \delta'(f(q), w)$
$w \in L(A)$ gdw $\delta(q_0, w) \in F$

$$
\begin{array}{lll}
f(F) = F' & \text{gdw} & f(\delta(q_0, w)) \in F \\
& \text{gdw} & \delta'(f(q), w) \in F' \\
& \text{gdw} & \delta'(q_0', w) \in F' \\
& \text{gdw} & w \in L(A')
\end{array}
$$

□

Satz 2.22 Es sei L eine erkennbare Sprache. Dann gilt:

1. A_L hat minimale Zustandszahl unter allen DEAs, welche L erkennen.
2. Ist A ein DEA mit $L(A) = L$, so ist A_{red} isomorph zu A_L.

Beweis:

1. Mit Satz 2.18 hat $\cong_L$ endlichen Index. Daher ist mit Lemma 2.17 A_L ein DEA für L. Mit Lemma 2.15.4 hat jeder DEA, der L erkennt, mindestens so viele Zustände, wie $\cong_L$ Klassen hat. Die Zustandszahl von A_L ist ja genau die Anzahl der $\cong_L$-Klassen.

2. Sei $L = L(A)$. Zu zeigen: $A_{red} \cong A_L$. Es sei $A_{red} = (Q, \Sigma, q_0, \delta, F)$ und $A_L = (Q', \Sigma, q_0', \delta, F')$. Wir definieren eine Funktion $f : Q \to Q'$ und zeigen, daß sie ein Isomorphismus ist. Für jedes $q \in Q$ existiert mindestens ein Wort $w_q \in \Sigma^*$ mit $\delta(q_0, w_q) = q$, da in A_{red} alle Zustände erreichbar sind.

   O.E. sei $w_{q_0} = \varepsilon$.

   Wir definieren:

   $$f(q) := [w_q]$$

   i f ist injektiv: Wir müssen dazu zeigen, daß $p \neq q \Rightarrow \underbrace{[w_p]}_{=f(p)} \neq \underbrace{[w_q]}_{=f(q)}$

   Da A_{red} reduziert ist, sind verschiedene Zustände nicht äquivalent. Es gibt daher mindestens ein Wort $w \in \Sigma^*$ mit
   $\delta(p, w) \in F \not\Leftrightarrow \delta(q, w) \in F$.
   Das heißt aber

   $$\delta(q_0, w_p w) \in F \not\Leftrightarrow \delta(q_0, w_q w) \in F$$

   Daraus folgt

   $$w_p w \in L \not\Leftrightarrow w_q w \in L$$

   Daher gilt

   $$w_p \not\cong_L w_q$$

   d.h.

   $$[w_q] \neq [w_p]$$

ii f ist surjektiv: Folgt aus Injektivität und $|Q| \underbrace{\geq}_{\text{(1) des Satzes}} |Q'|$, da Q, Q' endlich sind.

iii $f(q_0) = q_0'$
Da $w_{q_0} = \varepsilon$ und $q_0' = [\varepsilon] = f(q_0)$.

iv $f(F) = F'$
$q \in F$ gdw $\underbrace{\delta(q_0, w_q)}_{q} \in F$
gdw $w_q \in L$ gdw $\underbrace{[w_q]}_{=f(q)} \in F'$

v $f(\delta(q,a)) = \delta'(f(q), a)$: Es sei $\delta(q,a) =: p$. Dann ist $f(\delta(q,a)) = [w_p]$. Außerdem ist $\delta'(f(q), a) = \delta'([w_q], a) = [w_q a]$. Es bleibt also zu zeigen, daß $[w_p] = [w_q a]$, d.h. $\forall a : w_p w \in L \Leftrightarrow w_q a w \in L$. Dies ist offensichtlich dann der Fall, wenn
$\delta(q_0, w_p) = \delta(q_0, w_q a)$.
Es gilt aber:
$\delta(q_0, w_q a) = \delta(\delta(q_0, w_q), a) = \delta(q, a) = p = \delta(q_0, w_p)$

□

Korollar 2.23 Es seien A, A' DEAs. Dann gilt:

$$L(A) = L(A') \Leftrightarrow A_{red} \cong A'_{red}$$

Im Prinzip kann man damit das Äquivalenzproblem entscheiden: Reduziere A und A' und teste dann A_{red} und A'_{red} auf Isomorphie. Es gibt aber effizientere Verfahren dafür.

3 Nachweis der Nichterkennbarkeit

Nachweis der Erkennbarkeit:

Man gibt einen NEA an und beweise, daß er die Sprache akzeptiert.

Nachweis der Nichterkennbarkeit:

Schwieriger, da man zeigen muß, daß es keinen Automaten für die Sprache geben kann. Problem: es gibt unendlich viele Automaten! Satz 2.18 kann zum Nachweis verwendet werden.

Hilfsmittel zum Nachweis der Nichterkennbarkeit ist das

Lemma 3.1 (Pumping Lemma (einfache Version)) Es sei L eine erkennbare Sprache, dann gibt es eine natürliche Zahl $n_0 \geq 1$, so daß gilt:

Jedes Wort $w \in L$ mit $|w| \geq n_0$ läßt sich zerlegen in $w = xyz$ mit

- $y \neq \varepsilon$
- $xy^k z \in L$ für alle $k \geq 0$

Beweis: Es sei $A = (Q, \Sigma, q_0, \Delta, F)$ ein NEA mit $L = L(A)$. Wir wählen $n_0 := |Q|$.

Für jedes Wort $w = a_1 \dots a_m \in L$ existiert ein Pfad $(p_0, a_1, p_1)(p_1, a_2, p_2) \dots (p_{m-1}, a_m, p_m)$ in A mit $p_0 = q_0$ und $p_m \in F$.

Ist $m \geq n_0$, so können die $m+1$ Zustände $p_0, p_1 \dots p_m$ nicht alle verschieden sein, d.h. es gibt $i < j$ mit $p_i = p_j$.

Für $x = a_1 \dots a_i$, $y = a_{i+1} \dots a_j$ und $z = a_{j+1} \dots a_m$ gilt daher:

- $y \neq \varepsilon$ (da $i < j$)
- $q_0 = p_0 \xrightarrow{x} p_i \xrightarrow{y} p_j = p_i \xrightarrow{z} p_m \in F$

Folglich gilt für alle $h \geq 0$: $\quad xy^h z \in L(A) = L$

□

Wir verwenden das Lemma um zu zeigen, daß $\{a^n b^n \mid n \geq 0\}$ nicht erkennbar ist.

Beispiel 3.2 $\{a^n b^n \mid n > 0\}$ ist nicht erkennbar!

Beweis: Angenommen L wäre erkennbar, dann gibt es eine Zahl n_0 mit den in Lemma 3.1 beschriebenen Eigenschaften.

Es sei n so, daß $2n \geq n_0$ ist, d.h. $|a^n b^n| \geq n_0$. Da $a^n b^n \in L$ ist, gibt es x, y, z mit $a^n b^n = xyz$ und

- $y \neq \varepsilon$
- $xy^k z \in L \quad$ für alle $k \geq 0$

1. Fall: y liegt ganz in a^n, d.h. $x = a^{n_1}$, $y = a^{n_2}$, $z = a^{n_3} b^n$ mit $n_2 > 0$ und $n_1 + n_2 + n_3 = n$.

   Dann ist aber $xy^0 z = xz = a^{n_1 + n_3} b^n \notin L$ da $n_1 + n_3 < n$.

2. Fall: y liegt ganz in b^n, d.h. ... (kann symmetrisch behandelt werden).

3. Fall: y enthält a's und b's, d.h. $x = a^{n_1}$, $y = a^{n_2}b^{n_3}$, $z = b^{n_4}$ mit $n_2 \neq 0 \neq n_3$.

Dann ist aber $xyyz = a^{n_1}a^{n_2}b^{n_3}a^{n_2}b^{n_3}b^{n_4} \notin L$

Als eine weitere Konsequenz aus Lemma 3.1 erhalten wir:

Satz 3.3 Für erkennbare Sprachen (gegeben durch NEA oder DEA) ist das Leerheitsproblem "$L \neq \emptyset$" entscheidbar.

Beweis: Da L erkennbar ist, gibt es ein n_0 mit den Eigenschaften aus Lemma 3.1. Dieses n ist die Zustandsanzahl eines Automaten für L.

Behauptung: $L \neq \emptyset$ gdw $\exists w \in L$ mit $|w| < n_0$, denn

$\Leftarrow$ trivial

$\Rightarrow$ Es sei $L \neq \emptyset$. Wir wählen ein Wort w minimaler Länge in L. Wäre $|w| \geq n_0$, so könnte man w zerlegen in $w = xyz$ mit $y \neq \varepsilon$ und $xz = xy^0z \in L$. Dies Widerspricht der Minimalität von w. Damit ist $|w| < n_0$. (q.e.d. Behauptung)

Es gibt nur endlich viele Wörter der Länge $< n_0$ (da Alphabet endlich). Man muß nur für jedes dieser Wörter überprüfen, ob es zu L gehört. (q.e.d. Satz)

□

Mit Lemma 3.1 kann man dies nicht für alle nichterkennbaren Sprachen nachweisen.

Beispiel 3.4 Ist $L = \{a^nb^m \mid n \neq m\}$ erkennbar ?

Versucht man die Nichterkennbarkeit mit Lemma 3.1 zu zeigen, so scheitert man, da das Lemma für L nicht zutrifft:

Wähle $n_0 = 3$. Es sei nun $w \in L$ mit $|w| \geq 3$ d.h. $w = a^nb^m$ mit $n \neq m$ und $n + m \geq 3$.

1. Fall: $n > m$, d.h. $n := m + i$ für ein $i \geq 1$

1. $i > 1$, d.h. $n - 1 > m$
   Für $x := \varepsilon, y := a$ und $z := a^{n-1}b^m$ gilt für alle $k \geq 0$:
   $xy^kz = a^ka^{n-1}b^m \in L$, da $k + n - 1 \geq n - 1 > m$

2. $i = 1$, d.h. $n = m + 1$
   Wegen $n + m \geq 3$ ist $n = m + 1 \geq 2$
   Für $x := \varepsilon$, $y := a^2$, $z = a^{n-2}b^m$ gilt:
   (a) $xy^0z \in L$, da $n - 2 = m - 1 \neq m$
   $a^{n-2}b^m$
   (b) $xy^kz \in L$ für $k \geq 1$, da $(n - 2) + (k - 2) \geq n > m$

2. Fall: $n < m$ symmetrisch

Trotzdem ist $L = \{a^nb^m \mid n \neq m\}$ nicht erkennbar.

Lemma 3.5 (Pumping Lemma (verschärfte Version)) Es sei L erkennbar. Dann gibt es eine natürliche Zahl $n_0 \geq 1$, so daß gilt: Für alle Wörter u, v, w mit $uvw \in L$ und $|v| \geq n_0$ gibt es eine Zerlegung $v = xyz$ mit

- $y \neq \varepsilon$
- $uxy^kzw \in L$ für alle $k \geq 0$.

Beweis: Wir wählen $n_0 = |Q|$, wobei Q die Zustände eines Automaten A für L sind.

Ist $uvw \in L$, so gibt es Zustände $p, q, f \in Q$ mit $q_0 \xrightarrow{u}_A p \xrightarrow{v}_A q \xrightarrow{w}_A f \in F$.

Auf dem Pfad von p nach q liegen $|v| + 1 > n_0$ Zustände, also müssen wieder zwei davon gleich sein.

Jetzt kann man wie im Beweis nach Lemma 3.1 weitermachen.

□

Vorteil: Man kann das Teilwort in dem gepumpt werden soll geeignet positionieren.

Beispiel 3.4 (Fortsetzung):
$L = \{a^nb^m \mid n \neq m\}$ ist nicht erkennbar.

Beweis: Angenommen doch, dann gibt es ein n_0, das die in Lemma 3.5 geforderten Eigenschaften hat.

Wir betrachten die Wörter $u := \varepsilon$, $v = a^{n_0}$, $w = b^{n_0!+n_0}$. Offenbar ist $uvw = a^{n_0}b^{n_0!+n_0} \in L$,

Mit Lemma 3.5 gibt es eine Zerlegung $v = xyz$ mit $y \neq \varepsilon$ und $uxy^kzw \in L$. Für alle $k \geq 0$.

Es sei $x = a^{n_1}$, $y = a^{n_2}$, $z = a^{n_3}$ mit $n_1 + n_2 + n_3 = n_0$ und $n_2 > 0$.

Offenbar existiert ein l mit $n_2 \cdot l = n_0!$ (da $1 \leq n_2 \leq n_0$).

Es ist nun aber $n_1 + (l+1) \cdot n_2 + n_3 = n_0 + n_0!$. Damit ist
$uxy^{l+1}zw = a^{n_0+n_0!}b^{n_0+n_0!} \notin L$.

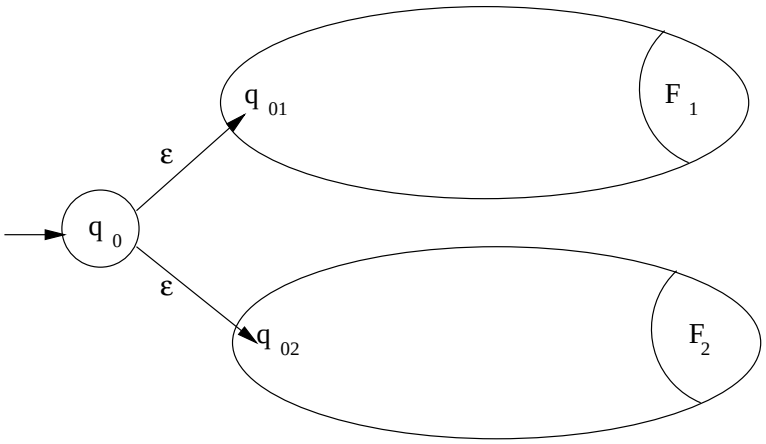
4 Abschlußeigenschaften und Entscheidungsprobleme

Satz 4.1 Sind L_1, L_2 erkennbar, so auch $L_1 \cup L_2$, $\bar{L_1}$, $L_1 \cap L_2$, $L_1 \setminus L_2$, $L_1 \cdot L_2$, L_1^*.

Beweis: Es sei $A_i = (Q_i, \Sigma, q_{0i}, \Delta_i, F_i)$ ein NEA mit $L_i = L(A_i)$.

O.E. sei $Q_1 \cap Q_2 = \emptyset$.

1. Der folgende ε-NEA akzeptiert $L_1 \cup L_2$:



   $A = \{Q_1 \cup Q_2 \cup q_0, \Sigma, q_0, \Delta, F_1 \cup F_2\}$ wobei $q_0 \notin Q_1 \cup Q_2$ und
   $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q_0, \varepsilon, q_{01}), (q_0, \varepsilon, q_{02})\}$

   Mit Lemma 1.12 gibt es zu A einen äquivalenten NEA.

2. Einen DEA für $\bar{L_1}$ erhält man wie folgt:

   - Verwende Potenzmengenkonstruktion um einen DEA $A = (Q, \Sigma, q_0, \delta, F)$ für L_1 zu konstruieren. $\bar{A} = (Q, \Sigma, q_0, \delta, Q \setminus F)$ ist ein DEA für $\bar{L_1}$.

$w \notin L_1$ gdw $w \notin L(A_1)$
gdw $w \notin L(A)$
gdw $\delta(q_0, w) \notin F$
gdw $\delta(q_0, w) \in Q \setminus F$
gdw $w \in L(\bar{A})$

Beachte: Man darf dies nicht direkt mit einem NEA für L_1 machen.

3. $L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$ zeigt Abschluß unter Durchschnitt wegen Abschluß unter $\cup, \bar{}$.

   Da die Potenzmengenkonstruktion sehr aufwendig sein kann, zeigen wir Abschluß unter $\cap$ noch direkt:

   Wir konstruieren dazu den Produktautomaten zu A_1 und A_2:

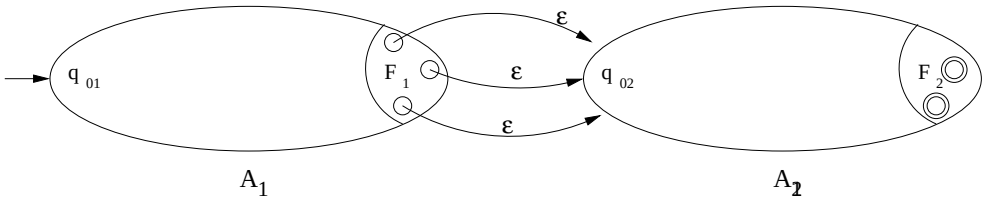
   $A := (Q_1 \times Q_2, \Sigma, (q_{01}, q_{02}), \Delta, F_1 \times F_2)$ mit

   $\Delta := \{((q_1, q_2), a, (q_1', q_2')) \mid (q_1, a, q_1') \in \Delta_1 \text{und} (q_2, a, q_2') \in \Delta_2\}$

   Ein Übergang in A ist also genau dann möglich, wenn die entsprechenden Übergänge in A_1 und A_2 möglich sind. Daraus ergibt sich leicht $L(A) = L(A_1) \cap L(A_2)$.

4. $L_1 \setminus L_2 = L_1 \cap \bar{L}_2$

5. $L_1 \cdot L_2$

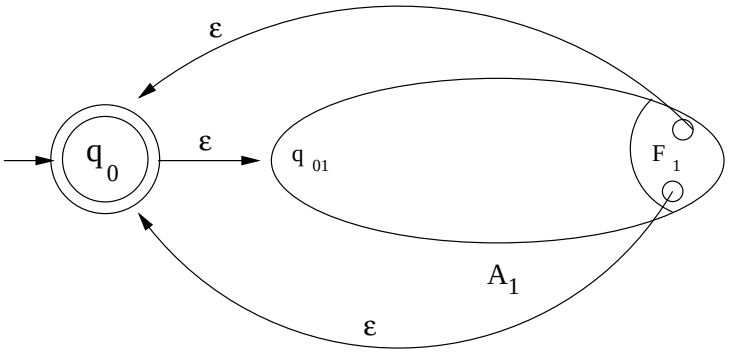


   Der folgende ε-NEA akzeptiert $L_1 \cdot L_2$:

   $A := (Q_1 \cup Q_2, \Sigma, q_{01}, \Delta, F_2)$, wobei

   $\Delta := \Delta_1 \cup \Delta_2 \cup \{(f, \varepsilon, q_{02}) \mid f \in F_1\}$

6. L_1^*:



Der folgende ε-NEA akzeptiert $L_1{}^*$: $A := (Q_1 \cup \{q_0\}, \Sigma, q_0, \Delta, \{q_0\})$, wobei $\Delta := \Delta_1 \cup \{(q_0, \varepsilon, q_{01})\} \cup \{(f, \varepsilon, q_0) \mid f \in F_1\}$ und $q_0 \notin Q_1$.

□

Beachte: Alle angegebenen Konstruktionen sind effektiv. Die Automaten für $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \cdot L_2$, $L_1{}^*$ sind polynomiell in der Größe von A_1 und A_2. Beim Komplement kann die Konstruktion exponentiell werden, wenn man von einem NEA ausgeht.

Man kann Abschlußeigenschaften nicht nur dazu verwenden, Erkennbarkeit zu zeigen, sondern umgekehrt auch dazu, Nichterkennbarkeit zu zeigen.

Beispiel 4.2 $L := \{a^n b^m \mid n \neq m\}$ ist nicht erkennbar (vgl. Beispiel 3.4).

Anstatt dies mit Lemma 3.5 direkt zu zeigen, kann man verwenden, daß $L' = \{a^n b^n \mid n \geq 0\}$ nicht erkennbar ist.

Wäre L erkennbar, so auch $\bar{L} \cap \underbrace{\{a\}^* \cdot \{b\}^*}_{\text{erkennbar}} = L'$.

Da wir schon wissen, daß L' nicht erkennbar ist, kann also L auch nicht erkennbar sein!

Entscheidungsprobleme:

- Leerheitsproblem: $L \neq \emptyset$?
- Wortproblem: $w \in L$?
- Äquivalenzproblem: $L_1 = L_2$?

Die Sprachen sind dabei gegeben durch einen NEA oder einen DEA. Für die Entscheidbarkeit ist es egal, ob man einen NEA oder einen DEA gegeben hat, aufgrund der Konstruktionseffizienz.

Für die Komplexität des Verfahrens kann dies einen Unterschied machen.

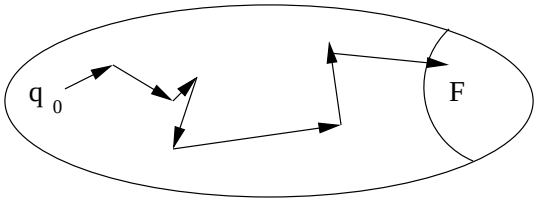
Leerheitsproblem:

Gegeben: erkennbare Sprache L
Frage: Ist $L \neq \emptyset$?

Entscheidbarkeit ist bereits gezeigt (Satz 3.3). Dieses Verfahren ist aber exponentiell, da es für $|\Sigma| > 1$ exponentiell viele Wörter mit Länge $< n_0$ gibt.

Satz 4.3 Es sei $A = (Q, \Sigma, q_0, \Delta, F)$ ein NEA. Dann kann man $''L(A) \neq \emptyset''$ in Zeit $O(\underbrace{|Q| + |\Delta|}_{\text{Größe des Automaten}})$ entscheiden.

Beweis:



Also kann man das als ein Erreichbarkeitsproblem in einem Graphen betrachten, und zwar folgendermaßen:

Man fasse den Automaten A als gerichteten Graphen $G = (Q, E)$ auf, mit $E := \{(q_1, q_2) \mid (q_1, a, q_2) \in \Delta \text{ für ein } a \in \Sigma\}$. Es gilt:

$L(A) \neq \emptyset$ gdw in der von q_0 aus erreichbaren Knotenmenge ein Endzustand liegt.

□

Die von q_0 aus erreichbaren Knoten kann man mit Aufwand $O(|Q| + |E|)$ berechnen (vgl. Vorlesung Algorithmen und Datenstrukturen). Insbesondere ist das Leerheitsproblem mit polynomiellem Aufwand lösbar (sogar mit linearem!).

Wortproblem:

Gegeben: Eine erkennbare Sprache L und ein Wort $w \in \Sigma^*$.
Frage: Gilt $w \in L$?

Ist $L = L(A)$ für einen DEA $A = (Q, \Sigma, q_0, \delta, F)$, so kann man einfach, beginnend mit q_0, durch Anwenden von δ berechnen, zu welchen Zustand man mit w kommt. Dann ist $w \in L$ gdw dieser Zustand zu F gehört. Nimmt man Σ als konstant an, so benötigt die Anwendung von δ konstante Zeit. Dies liefert:

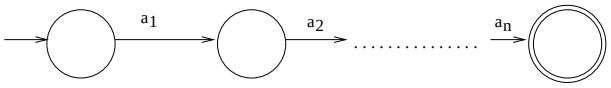
Satz 4.4 Sei $A = (Q, \Sigma, q_0, \delta, F)$ ein DEA und $w \in \Sigma^*$. Dann kann man $''w \in L(A)''$ in Zeit $O(|w|)$ entscheiden.

Bei NEAs ist das nicht so einfach, da es für ein gegebenes w viele Pfade geben kann, die mit w beschriftet sind.

Satz 4.5 Es sei $A = (Q, \Sigma, q_0, \delta, F)$ ein NEA und $w \in \Sigma^*$. Dann kann man $''w \in L(A)''$ in Zeit $O(|w| \cdot (|Q| + |\Delta|))$ entscheiden.

Beweis: Wir konstruieren einen Automaten A_w, der die Sprache $\{w\}$ akzeptiert:

Für $w = a_1 \dots a_n$ ist A_w:

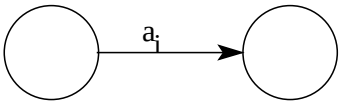


Anzahl der Zustände: $|w| + 1$

Offenbar ist $w \in L(A)$ gdw $L(A) \cap L(A_w) \neq \emptyset$.

Der Produktautomat zu A und A_w:

- Zustände: $(|w| + 1) \cdot |Q|$
- Übergänge: Für jeden Übergang

  in A_w gibt es maximal $|\Delta|$ viele Übergänge in A.

Also maximal $|w| \cdot |\Delta|$ im Produktautomaten. Nach Satz 4.3 ist daher der Aufwand zum Testen von "$L(A) \cap L(A_w) \neq \emptyset$?"

$O(|Q| \cdot (|w| + 1) + |w| \cdot |\Delta|) = O(|w| \cdot (|Q| + |\Delta|))$.

□

Äquivalenzproblem:

Gegeben: zwei erkennbare Sprachen L_1 und L_2
Frage: Gilt $L_1 = L_2$?

Wie bereits erwähnt kann man das Äquivalenzproblem auf das Leerheitsproblem reduzieren:

$L_1 = L_2$ gdw $(L_1 \cap \bar{L_2}) \cup (\bar{L_1} \cap L_2) = \emptyset$.

Satz 4.6 Das Äquivalenzproblem ist für erkennbare Sprachen entscheidbar. Sind die Sprachen durch DEAs gegeben, so ist dies in polynomieller Zeit möglich.

Bei NEAs: Die Potenzmengenkonstruktion kann exponentiellen Aufwand haben. Es geht wahrscheinlich nicht besser: Das Äquivalenzproblem für NEAs ist [1].

[1] PSPACE ist vermutlich schlimmer als NP!

Wenn im folgenden von “ist vermutlich” gesprochen wird, ist damit gemeint, daß diese Aussagen derzeit vermutet werden, aber noch nicht bewiesen sind. Da allerdings diese Aussagen auch nicht widerlegt werden können, ist gerade dieser Bereich der theoretischen Informatik von großem Interesse.

5 Reguläre Ausdrücke und Sprachen

Verschiedene äquivalente Charakterisierungen der erkennbaren Sprachen:

1. $L = L(A)$ für einen NEA A
2. $L = L(A)$ für einen ε-NEA A
3. $L = L(A)$ für einen NEA mit Wortübergängen
4. $L = L(A)$ für ein endliches Transitionssystem
5. $L = L(A)$ für einen DEA A
6. Die Nerode Rechtskongruenz $\cong_L$ hat endlichen Index.

Im weiteren betrachten wir eine weitere Charakterisierung, die durch reguläre Ausdrücke.

Definition 5.1 (Syntax regulärer Ausdrücke) Es sei Σ ein endliches Alphabet. Die Menge Reg_Σ der reguläre Ausdrücke über Σ ist induktiv definiert:

- $\emptyset, \varepsilon, a$ für $a \in \Sigma$ sind Elemente von Reg_Σ.
- Sind $r, s \in Reg_\Sigma$, so auch $(r + s), (r \cdot s), r^* \in Reg_\Sigma$.

Beispiel 5.2 $((a \cdot b^*) + \emptyset^*)^* \in Reg_\Sigma$ für $\Sigma = \{a, b\}$

Notation:

Um Klammern zu sparen, lassen wir Außenklammern weg und vereinbaren, daß $*$ stärker bindet als $\cdot$, und daß $\cdot$ stärker bindet als $+$. Außerdem lassen wir $\cdot$ meist weg. Zum Beispiel wird der Ausdruck aus obigem Beispiel damit zu $(a \cdot b^* + \emptyset^*)^* = (ab^* + \emptyset^*)^*$.

Jedem regulären Ausdruck wird genau eine formale Sprache zugeordnet:

Definition 5.3 (Semantik regulärer Ausdrücke) Die durch den regulären Ausdruck r definierte Sprache $L(r)$ ist induktiv definiert:

- $L(\emptyset) := \emptyset$, $L(\varepsilon) := \{\varepsilon\}$, $L(a) := \{a\}$
- $L(r + s) = L(r) \cup L(s)$
  $L(r \cdot s) = L(r) \cdot L(s)$
  $L(r^*) = L(r)^*$

Eine Sprache $L \subseteq \Sigma^*$ heißt regulär gdw es ein $r \in Reg_\Sigma$ gibt mit $L = L(r)$.

Beispiel 5.4

- $(a + b)^*ab(a + b)^*$ definiert die Sprache aller Wörter über $\{a, b\}$, die ab als Infix enthalten.
- $L(ab^* + b) = \{ab^i \mid i \geq 0\} \cup \{b\}$

Bemerkung: Statt $L(r)$ schreiben wir häufig einfach r. Dies ermöglicht zu schreiben:

- $abb \in ab^* + b$ statt $abb \in L(ab^* + b)$
- $(ab)^*a = a(ba)^*$ statt $L((ab)^*a) = L(a(ba)^*)$

Satz 5.5 (Satz von Kleene) Für eine Sprache $L \subseteq \Sigma^*$ sind äquivalent:

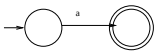
1. L ist regulär.
2. L ist erkennbar.

Beweis:

$''1 \rightarrow 2''$: Induktion über den Aufbau der regulären Ausdrücke:

Verankerung:

- $L(\emptyset) = \emptyset$:  ist ein NEA für $\emptyset$.
- $L(\varepsilon) = \{\varepsilon\}$:  ist ein NEA für $\{\varepsilon\}$.

- $L(a) = \{a\}$:  ist ein NEA für $\{a\}$.

Schritt: Weiß man bereits, daß $L(r)$ und $L(s)$ erkennbar sind, so folgt mit Satz 4.1 (Abschlußeigenschaften), daß auch

$L(r + s) = L(r) \cup L(s)$

$L(r \cdot s) = L(r) \cdot L(s)$

$L(r^*) = L(r)^*$

erkennbar sind.

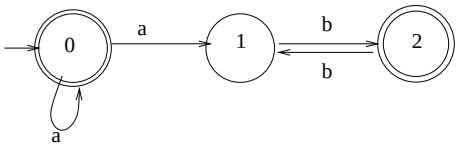
$''2 \to 1'':$ Sei $A = (Q, \Sigma, q_0, \Delta, F)$ ein NEA mit $L = L(A)$. Wie in Definition 2.8 sei für $q \in Q$ definiert:

$A_q := (Q, \Sigma, q, \Delta, F)$ und $L_q := L(A_q)$

Der Zusammenhang zwischen den L_q kann durch ein Gleichungssystem beschrieben werden, dessen Lösungen eindeutig bestimmte reguläre Sprachen sind.

□

Beispiel 5.6 Sei A =



$L_0 = \{a\}L_0 \cup \{a\}L_1$

$L_1 = \{b\}L_2$

$L_2 = \{b\}L_1 \cup \underbrace{\{\varepsilon\}}_{2 \in F}$

NEA $A \quad \longrightarrow \quad$ regulärer Ausdruck r mit $L(r) = L(A)$.

Allgemeiner: Für $p, q \in Q$ sei $A_{p,q} := \{a \in \Sigma \mid (p, a, q) \in \Delta\}$ und

$B_p := \begin{cases} \{\varepsilon\} & \text{falls } p \in F \\ \emptyset & \text{falls } p \notin F \end{cases}$.

Damit erfüllen die Sprachen L_p die folgenden Gleichungen:

$L_p = (\bigcup_{q \in Q} A_{p,q} \cdot L_q) \cup B_p$.

Behauptung: Das Gleichungssystem

$(*)$ $\qquad X_p = (\bigcup_{q \in Q} A_{p,q} \cdot X_q) \cup B_p \qquad (p \in Q)$

hat genau eine Lösung, die aus regulären Sprachen besteht.

Da die Sprachen L_p $(*)$ lösen folgt, daß sie regulär sind.

Wir zeigen, wie man einzelne Gleichungen der Form

$(**)$ $\qquad X = A \cdot X \cup B$

lösen kann.

Daraus ergibt sich dann die Lösung von $(*)$ durch Auflösen nach einer Variablen und Einsetzen in den Rest.

Lemma 5.7 (Arden) Es seien $A, B \subseteq \Sigma^*$ und $\varepsilon \notin A$.

Die Gleichung $X = AX \cup B$ hat als eindeutige Lösung $X = A^*B$.

Beachte: Sind A, B regulär, so auch A^*B.

Aus dem Lemma folgt für Gleichung $(*)$, die wir als Gleichung der Form $(**)$ auffassen: Wähle ein $p \in Q$ und betrachte

$$X_p = \underbrace{A_{p,p}X_p}_{A \subseteq \Sigma \Rightarrow \varepsilon \notin A} \cup \underbrace{((\bigcup_{p \neq q} A_{p,q}X_q) \cup B_p)}_{B}$$

Mit Arden hat diese Gleichung die eindeutige Lösung

$$X_p = A_{p,p}^* \cdot \underbrace{(\bigcup_{p \neq q} A_{p,q}X_q \cup B_p)}_{\text{enthält } X_p \text{ nicht!}}.$$

Setzt man diese Lösung in die restlichen Gleichungen ein, so erhält man ein System mit einer Variablen weniger. Mit Induktion kann man daher annehmen, daß dieses System eine eindeutige Lösung mit regulären Sprachen hat. Die Lösung für X_p ist damit auch eindeutig und regulär, da $A_{p,p}$, $A_{p,q}$, B_p regulär sind und die Lösung für X_p nur reguläre Operationen $(\cup, \cdot, *)$ enthält.

Beispiel 5.6 (Fortsetzung):

$$X_0 = \{a\}X_0 \cup \{a\}X_1$$
$$X_1 = \{b\}X_2$$
$$X_2 = \{b\}X_1 \cup \{\varepsilon\}$$

Auflösen nach X_0:

$X_0 = \{a\}^*\{a\}X_1$ Einsetzen in X_0 ändert nichts.

Auflösen nach X_1:

$X_1 = \emptyset \cdot X_1 \cup \{b\}X_2$

Arden: $X_1 = \underbrace{\emptyset^*}_{\{\varepsilon\}}\{b\}X_2 = \{b\}X_2$

Einsetzen liefert

$X_2 = \underbrace{\{b\}\{b\}}_{\{bb\}} X_2 \cup \{\varepsilon\}$

Arden: $X_2 = \{bb\}^*\{\varepsilon\} = \{bb\}^*$

Damit ist $X_1 = \{b\}\{bb\}^*$ und $X_0 = \{a\}^*\{a\}\{b\}\{bb\}^* = L(A)$.

Als regulärer Ausdruck:

$a^*ab(bb)^*$

Beweis von Lemma 5.7:

(1) A^*B ist Lösung:

$$\begin{aligned} AA^*B \cup B &= (AA^* \cup \{\varepsilon\})B \\ &= A^*B \end{aligned}$$

(2) Eindeutigkeit: Sei L eine Lösung, d.h. $L = AL \cup B$.
Zu zeigen: $L = A^*B$

(2.1) $A^*B \subseteq L$
Wir zeigen durch Induktion über n: $A^nB \subseteq L$

- $A^0B = B \quad \subseteq \quad AL \cup B = L$
- Gelte $A^nB \subseteq L$. Es folgt
  $A^{n+1}B = A\underbrace{A^nB}_{\in L} \quad \subseteq \quad AL \quad \subseteq \quad AL \cup B = L$

Daraus folgt $A^*B \subseteq L$.

(2.2) $L \subseteq A^*B$.
Angenommen, dies gilt nicht. Es sei $w \in L \setminus A^*B$ von minimaler Länge, $w \in L = AL \cup B$, d.h. $w \in AL$ oder $w \in B$.

1. $w \in B \Rightarrow w \in A^*B$ (Widerspruch!)
2. $w \in AL$, d.h. $w = uv$ mit $u \in A$ und $v \in L$
   Da $\varepsilon \notin A$ ist, folgt $|v| < |w|$. Wegen der Minimalität von w folgt:
   $v \in A^*B \Rightarrow w \in A^*B$ (Widerspruch!)

□

Teil II

Grammatiken, kontextfreie Sprachen und Kellerautomaten

Klassen formaler Sprachen, die allgemeiner sind als reguläre Sprachen.
Grammatiken, die Sprachen erzeugen.

6 Die Chomskyhierarchie

Wie erzeugen Grammatiken Sprachen?

Beginne mit dem Startsymbol S und wende Regeln an, die ein Wort durch ein anderes ersetzen können. Die Sprache enthält die Wörter, die man von S durch Regelanwendung erreichen kann.

Beispiel 6.1 Regeln:

$$S \to aSb \quad (1)$$

$$S \to \varepsilon \quad (2)$$

Startsymbol: S

$$S \xrightarrow{1} aSb \xrightarrow{1} aaSbb \xrightarrow{1} aaaSbbb \xrightarrow{2} aaabbb$$

Das Symbol S ist ein Hilfssymbol (Nichtterminalsymbol). Man ist nur an den erzeugten Wörtern interessiert, die keine Hilfssymbole mehr enthalten (Terminalwörter). Im Beispiel werden als Terminalwörter genau $a^n b^n$ mit $n \geq 0$ erzeugt.

Definition 6.2 Eine Grammatik ist von der Form $G = (N, \Sigma, P, S)$ wobei:

- N und Σ disjunkte, endliche Alphabete (N: Nichtterminalsymbole, Σ: Terminalsysmbole)
- $S \in N$ Startsysmbol
- $P \subseteq (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ eine (endliche) Menge von Ersetzungsregeln (Produktionen)

Eine Produktion $(u, v) \in P$ schreibt man gewöhnlich als $u \to v$.

Beispiel 6.3 $G = (N, \Sigma, P, S)$ mit $N = \{S, B\}$, $\Sigma = \{a, b, c\}$,
$P = \{S \to aSBc, S \to abc, cB \to Bc, bB \to bb\}$

Definition 6.4 Es sei $G = (N, \Sigma, P, S)$ eine Grammatik und $x, y \in (N \cup \Sigma)^*$.

1. $x \underset{G}{\vdash} y$ gdw $\exists x_1, x_2 \in (N \cup \Sigma)^* : \exists u \to v \in P$ mit $x = x_1 u x_2$ und $y = x_1 v x_2$
2. $x \overset{n}{\underset{G}{\vdash}} y$ gdw
   $\exists x_0, \ldots, x_n \in (N \cup \Sigma)^* : x_0 = x \wedge x_n = y \quad x_i \underset{G}{\vdash} x_{i+1} (0 \leq i < n)$

   $n = 0 \quad x \overset{0}{\underset{G}{\vdash}}$ gdw $x = y$
3. $x \overset{*}{\underset{G}{\vdash}} y$ gdw $\exists n \geq$ mit $x \overset{n}{\underset{G}{\vdash}} y$
4. Die durch G erzeugte Sprache: $L(G) := \{w \in \Sigma^* \mid S \overset{*}{\underset{G}{\vdash}} w\}$

Man ist nur an den von S aus ableitbaren Terminalwörtern, d.h. Wörtern aus Σ^* interessiert.

Beispiel 6.3 (Fortsetzung):

$S \underset{G}{\vdash} abc$, d.h. $abc \in L(G)$

$S \underset{G}{\vdash} aSBc \underset{G}{\vdash} aaSBcBc \underset{G}{\vdash} aaaSBcBcBcBc \underset{G}{\vdash} aaaabcBcBcBc \overset{*}{\underset{G}{\vdash}}$
$aaaabBBBcccc \overset{*}{\underset{G}{\vdash}} a^4b^4c^4$

Es gilt: $L(G) = \{a^n b^n c^n \mid n \geq 1\}$

Beweis:

"$\supseteq$" $n = 1 : abc \in L(G)$

$S \overset{n-1}{\underset{G}{\vdash}} a^{n-1}S(Bc)^{n-1} \underset{G}{\vdash} a^n bc(Bc)^{n-1} \overset{*}{\underset{G}{\vdash}} a^n bB^{n-1}c^n \overset{*}{\underset{G}{\vdash}} a^n b^n c^n$

"$\subseteq$" Es gelte $S \overset{*}{\underset{G}{\vdash}} w$ mit $w \in \Sigma^*$. Wird sofort $S \to abc$ angewandt, so ist $w = abc \in \{a^n b^n c^n \mid n \geq 1\}$. Sonst betrachten wir die Stelle in der Ableitung $S \overset{*}{\underset{G}{\vdash}} w$, an der S das letzte Mal auftritt.

$S \overset{*}{\underset{G}{\vdash}} a^{n-1} S u$ mit $u \in \{c, B\}^*$ und $|u|_B = |u|_c = n - 1$ (nur $S \to aSBc$ und $CB \to Bc$ angewandt)

$a^{n-1} S u \underset{G}{\vdash} a^n bcu \overset{*}{\underset{G}{\vdash}} w$ (in den letzten Schritten nur noch $cB \to Bc$ und $bB \to bb$ angewandt)

Dadurch bleibt die Anzahl der b, B gleich n und die Anzahl der c auch gleich n. Um die B zu b zu machen, müssen sie auf ein b treffen und damit links von den c's stehen.

Beispiel 6.5 $G = (N, \Sigma, P, S)$ mit $N = \{S, B\}$, $\Sigma = \{a, b\}$, $P = \{S \to bS, S \to abB, B \to aB, B \to bB, B \to \varepsilon\}$

$$L(G) = \Sigma^* ab \Sigma^*$$

Die Beispiele 6.5, 6.3 und 6.1 gehören zu verschiedenen Stufen der Chomskyhierarchie.

Definition 6.6 Es sei $G = (N, \Sigma, P, S)$ eine Grammatik.

- Jede Grammatik G ist eine Typ0 Grammatik.
- G heißt Grammatik vom Typ1 (kontextsensitiv), falls jede Produktion von G die Form
  - $u_1 A u_2 \to u_1 w u_2$ mit $A \in N \quad u_1, u_2, w \in (N \cup \Sigma)^*$ und $|w| \geq 1$, oder
  - $S \to \varepsilon$ hat.

  Ist $S \to \varepsilon \in P$, so kommt S nicht auf der rechten Seite einer Produktion vor.
- G heißt Grammatik vom Typ2 (kontextfrei), falls jede Regel die Form $A \to w$ hat mit $A \in N$ und $w \in (N \cup \Sigma)^*$.
- G heißt Grammatik vom Typ3 (rechtslinear), falls jede Regel von der Form $A \to uB$ oder $A \to u$ ist mit $A, B \in N, u \in \Sigma^*$.

  siehe auch Beispiel 6.5 (Typ3), 6.1 (Typ2; $S \to aSb$) und 6.3 (fast Typ2; $cB \to Bc$ kontextsensitiv!)

Definition 6.7 Für $i = 0, 1, 2, 3$ ist die Klasse der Typ$i$ Sprachen definiert als:

$$\mathcal{L}_i = \{L(G) \mid G \text{ ist Typ}i \text{ Grammatik}\}$$

Wir werden sehen $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$. Nach Definition der Grammatiktypen gilt offenbar $\mathcal{L}_3 \subseteq \mathcal{L}_2$ und $\mathcal{L}_1 \subseteq \mathcal{L}_0$.

7 Rechtslineare Grammatiken und reguläre Sprachen

Satz 7.1 $\mathcal{L}_3 = \{L \mid L \text{ ist regulär}\}$

Beweis:

"$\subseteq$" Es sei $L \in \mathcal{L}_3$, d.h. $L = L(G)$ für eine Typ3 Grammatik $G = (N, \Sigma, P, S)$
Es gilt $w \in L(G)$ gdw

(*) es gibt eine Ableitung

$$S = B_0 \underset{G}{\vdash} w_1 B_1 \underset{G}{\vdash} w_1 w_2 B_2 \underset{G}{\vdash} \ldots \underset{G}{\vdash} w_1 w_2 \ldots w_{n-1} B_{n-1} \underset{G}{\vdash} w_1 w_2 \ldots w_n$$

für die Produktionen $B_{i-1} \to w_1 B_i \in P \quad (i = 1, \ldots, n), \quad B_{n-1} \to w_n$.
Wir definieren einen NEA mit Wortübergängen für $L(G)$:

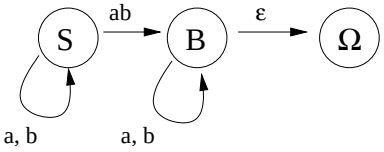
$$A = (N \cup \Omega, \Sigma, S, \Delta, \{\Omega\}) \quad \Omega \notin N \text{ mit}$$

$$\Delta = \{(A, w, B) \mid A \to wB \in P\} \cup \{(A, w, \Omega) \mid A \to w \in P\}$$

Ableitungen der Form (*) entsprechen Pfaden

$$(S, w_1, B_1)(B_1, w_2, B_2) \ldots (B_{n-2}, w_{n-1}, B_{n-1})(B_{n-1}, w_n, \Omega)$$

vergleiche Beispiel 6.5

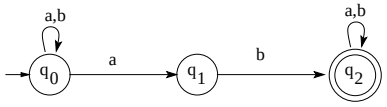


"$\supseteq$" Es sei $L = L(A)$ für einen NEA $A = (Q, \Sigma, q_0, \Delta, F)$. Wir definieren daraus eine Typ3-Grammatik $G = (N, \Sigma, P, S)$ mit:
$N := Q$, $S := q_0$, $P := \{p \to aq \mid (p, a, q) \in \Delta\} \cup \{p \to \varepsilon \mid p \in F\}$
Ein Pfad in A der Form $(q_0, a_1, q_1)(q_1, a_2, q_2) \ldots (q_{n-1}, a_n, q_n), q_n \in F$ entspricht genau der Ableitung

$$q_0 \underset{G}{\vdash} a_1 q_1 \underset{G}{\vdash} a_1 a_2 q_2 \underset{G}{\vdash} \ldots \underset{G}{\vdash} a_1 \ldots a_n q_n \underset{G}{\vdash} a_1 \ldots a_n$$

□

Beispiel 7.2



liefert die Produktionen
$P = \{q_0 \to aq_0, q_0 \to bq_0, q_0 \to aq_1, q_1 \to bq_2, q_2 \to aq_2, q_2 \to bq_2, q_2 \to \varepsilon\}$

Korollar 7.3 $\mathcal{L}_3 \subset \mathcal{L}_2$

Beweis: Wir wissen bereits $\mathcal{L}_3 \subseteq \mathcal{L}_2$. Mit Beispiel 6.1 ist
$L := \{a^n b^n \mid n \geq 0\} \in \mathcal{L}_2$. Im Teil I der Vorlesung haben wir gezeigt, daß L nicht regulär/erkennbar ist. Mit Satz 7.1 folgt $L \notin \mathcal{L}_3$.

□

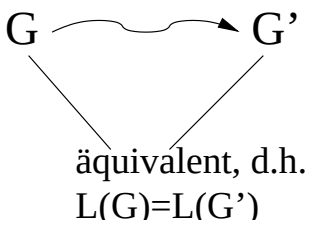
Beispiel 7.4 Ein weiteres Beispiel für eine Sprache in $\mathcal{L}_2 \setminus \mathcal{L}_3$: $\{a^n b^m \mid n \neq m\}$ (nicht regulär).
Man kann diese Sprache mit der folgenden kontextfreien Grammatik erzeugen:
$S \to aS^{\geq}$, $S \to S^{\leq}b$
$S^{\geq} \to aS^{\geq}b$, $S^{\geq} \to aS^{\geq}$, $S^{\geq} \to \varepsilon$
$S^{\leq} \to aS^{\leq}b$, $S^{\leq} \to S^{\leq}b$, $S^{\leq} \to \varepsilon$
Es gilt:

- $S^{\geq} \overset{*}{\underset{G}{\vdash}} w \in \{a, b\}^* \Rightarrow w = a^n b^m$ mit $n \geq m$
- $S^{\leq} \overset{*}{\underset{G}{\vdash}} w \in \{a, b\}^* \Rightarrow w = a^n b^m$ mit $n \leq m$

Daraus ergibt sich:
$S^{\geq} \overset{*}{\underset{G}{\vdash}} w \in \{a, b\}^* \Rightarrow w = a^n b^m$ mit $n > m$ oder $n < m$,
d.h.

$$L(G) = \{a^n b^m \mid n \neq m\}$$

8 Normalformen kontextfreier Grammatiken



Wir zeigen zunächst, daß man "überflüssige" Symbole entfernen kann, ohne die erzeugte Sprache zu ändern.

Definition 8.1 Es sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik.

1. $A \in N$ heißt terminierend, falls es ein $w \in \Sigma^*$ gibt, mit $A \overset{*}{\underset{G}{\vdash}} w$.
2. $A \in N$ heißt erreichbar, falls es $u, v \in (\Sigma \cup N)^*$ gibt, mit $S \overset{*}{\underset{G}{\vdash}} uAv$.
3. G heißt reduziert, falls alle Elemente von N erreichbar und terminierend sind.

Beachte: In einer Ableitung $S \overset{*}{\underset{G}{\vdash}} w \in \Sigma^*$ können nur erreichbare und terminierende Symbole $A \in N$ vorkommen.

Lemma 8.2 Zu einer kontextfreien Grammatik $G = (N, \Sigma, P, S)$ kann man effektiv die Menge aller erreichbaren Symbole bestimmen.

Beweis: Wir definieren:
$E_0 := \{S\}$
$E_{i+1} := E_i \cup \{A \mid \exists B \in E_i$ und Regel $B \to u_1 A u_2 \in P\}$
Es gilt: $E_0 \subseteq E_1 \subseteq E_2 \subseteq \ldots \subseteq N$
Da N endlich ist, gibt es ein k mit $E_k = E_{k+1}$ und damit $E_k = \bigcup_{i \geq 0} E_i$.
Behauptung: $E_k = \{A \in N \mid A$ ist erreichbar$\}$
denn:

"$\subseteq$" Offenbar enthalten die E_i nur erreichbare Symbole.

"$\supseteq$" Zeige durch Induktion über i:
$S \overset{i}{\underset{G}{\vdash}} uAv \Rightarrow A \in E_i$

□

Beispiel 8.3 $S \to aS$, $S \to SB$, $S \to SS$, $S \to \varepsilon$, $A \to ASB$, $A \to c$, $B \to Cb$
$E_0 = \{S\} \subseteq E_1 = \{S, B\} \subseteq E_2 = \{S, B, C\} = E_3$
Damit ist A das einzige nicht erreichbare Symbol.

Lemma 8.4 Zu einer kontextfreien Grammatik $G = (N, \Sigma, P, S)$ kann man effektiv die Menge der terminierenden Symbole berechnen.

Beweis:

$T_1 := \{A \in N \mid \exists w \in \Sigma^* : A \to w \in P\}$

$T_{i+1} = T_i \cup \{A \in N \mid \exists w \in (\Sigma \cup T_i)^* : A \to w \in P\}$

$$A \underset{G}{\vdash} w_1 B_1 w_2 \cdots w_n B_n w_{n+1}$$

mit $w_i \in \Sigma^*$ für $i = 1, 2, \ldots n+1$ und $B_i \in T_i$.

Wegen $T_1 \subseteq T_2 \subseteq \cdots \subseteq N$ gibt es ein k mit $T_k = T_{k+1} = \bigcup_{i \geq 1} T_i$.

Behauptung: $T_k = \{A \in N \mid A \text{ ist terminierend}\}$

denn:

"$\subseteq$" Durch Induktion über i zeigen wir T_i besteht nur aus terminierenden Symbolen.

$i = 1$**:** klar nach Definition

$i \to i+1$**:** Es sei $A \in T_{i+1} \setminus T_i$. Dann gibt es eine Produktion
$A \to w_1 B_1 w_2 B_2 \cdots w_n B_n w_{n+1}$ mit $w_i \in \Sigma^*$ und $B_i \in T_i$.
Mit Induktionsvorraussetzung ist B_i terminierend, d.h. $u_i \in \Sigma^*$ mit $B_i \overset{*}{\underset{G}{\vdash}} u_i$
Damit gilt: $A \overset{*}{\underset{G}{\vdash}} \underbrace{w_1 u_1 w_2 u_2 \cdots w_n u_n w_{n+1}}_{\in \Sigma^*}$.
Also ist A terminierend.

"$\supseteq$" Zeige durch Induktion über i:

$$A \overset{\leq i}{\underset{G}{\vdash}} w \in \Sigma^* \Rightarrow A \in T_i$$

$i = 1$**:** $A \overset{\leq 1}{\underset{G}{\vdash}} w \in \Sigma^* \Rightarrow A \to w \in P \Rightarrow A \in T_1$

$i \to i+1$: $A \underset{G}{\vdash} u_1 B_1 u_2 B_2 \cdots u_n B_n u_{n+1} \overset{\leq i}{\underset{G}{\vdash}} w$ mit $u_j \in \Sigma^*, B_j \in N$ und
$w = u_1 w_1 u_2 w_2 \cdots u_n w_n u_{n+1}$ mit $w_j \in \Sigma^*$ und $B_j \overset{\leq i}{\underset{G}{\vdash}} w_j$
$\Rightarrow$ mit Induktion $B_j \in T_i$
$\Rightarrow A \in T_{i+1}$

□

Satz 8.5 Das Leerheitsproblem für kontextfreie Sprachen ist entscheidbar.

Beweis: Es gilt: $L(G) \neq \emptyset$ gdw $\exists w \in \Sigma^* : S \overset{*}{\underset{G}{\vdash}} w$
gdw S ist terminierend.

□

Lemma 8.2 und 8.4 zeigen, daß man kontextfreie Grammatiken effektiv reduzieren kann.

Beachte: Man muß das Entfernen der nicht-erreichbaren und nicht-terminierenden Symbole in der richtigen Reihenfolge durchführen. Das Problem ist, daß durch Entfernen nicht-terminierender Symbole vorher erreichbare Symbole nicht-erreichbar werden.

$S \to AB$ damit sind A und B erreichbar.

$A \to a$ damit ist A terminierend, B nicht.

$\Rightarrow$ entferne $S \to AB$, weil nicht-terminierend

$\Rightarrow$ entferne A da unerreichbar

Satz 8.6 Zu jeder kontextfreien Grammatik G mit $L(G) \neq \emptyset$ kann man effektiv eine äquivalente reduzierte kontextfreie Grammatik konstruieren.

Beweis: Zu $G = (N, \Sigma, P, S)$ definieren wir $G' = (N', \Sigma, P', S)$ mit
$N' := \{A \in N \mid A \text{ ist terminierend in } G\}$ und
$P' := \{A \to w \in P \mid A \in N', w \in (\Sigma \cup N')^*\}$

Da $L(G) \neq \emptyset$ ist, gilt natürlich $S \in N'$.

Im zweiten Schritt definieren wir $G'' = (N'', \Sigma, P'', S)$ mit
$N'' := \{A \in N' \mid A \text{ ist erreichbar in } G'\}$ und $P'' := \{A \to w \in P' \mid A \in N''\}$

Beachte: $S \in N''$ und ist $A \in N''$ und $A \to w \in P'$, so ist jedes Nichtterminalsymbol in w auch in N''.

Man sieht leicht, daß G'' reduziert und äquivalent zu G ist. $S \overset{*}{\underset{G''}{\vdash}} w \in \Sigma^*$ kann nur erreichbare und terminierende Symbole enthalten.

reduziert $A \overset{*}{\underset{G'}{\vdash}} w \in \Sigma^*$ und $A \in N'' \Rightarrow A \overset{*}{\underset{G''}{\vdash}} w$. Da A erreichbar ist, sind alle Symbole in der Ableitung $A \overset{*}{\underset{G''}{\vdash}} w$ erreichbar.

□

Als nächstes eliminieren wir Produktionen der Form $A \to \varepsilon$.

Lemma 8.7 Zu jeder kontextfreien Grammatik kann man effektiv eine kontextfreie Grammatik G' konstruieren mit der Eigenschaft

$$L(G') = L(G) \setminus \{\varepsilon\}$$

Beweis:

1. Finde alle $A \in N$ mit $A \overset{*}{\underset{G}{\vdash}} \varepsilon$.

   $N_1 := \{A \in N \mid A \to \varepsilon \in P\}$

   $N_{i+1} := N_i \cup \{A \in N \mid A \to B_1 \dots B_n \in P \text{ und } B_1 \dots B_n \in N_i\}$

   Es gibt ein k mit $N_k = N_{k+1} = \bigcup_{i \geq 1} N_i$

   Für dieses k gilt: $A \in N_k$ gdw $A \overset{*}{\underset{G}{\vdash}} \varepsilon$.

2. Eliminiere in G alle Regeln der Form $A \to \varepsilon$.

   Um dies auszugleichen, nimmt man für alle Regeln $A \to u_1 B_1 u_2 B_2 \cdots u_n B_n u_{n+1}$ mit $B_i \in N_k$ und $u_i \in (\Sigma \cup (N \setminus N_k))^*$ die Regeln $A \to u_1 \beta_1 u_2 \beta_2 \cdots u_n \beta_n u_{n+1}$ hinzu, wobei:

   - $\beta_i \in \{B_i, \varepsilon\}$
   - $u_1 \beta_1 u_2 \beta_2 \cdots u_n \beta_n u_{n+1} \neq \varepsilon$

Man zeigt leicht: Die erhaltene Grammatik G' enthält keine ε-Produktionen und $L(G') = L(G) \setminus \{\varepsilon\}$.

□

Beispiel:

$S \to aS$, $S \to SS$, $S \to bA$,

$A \to BB$,

$B \to CC$, $B \to aAbC$,

$C \to \varepsilon$,

$N_1 = \{C\} \subset N_2 = \{C, B\} \subset N_3 = \{C, B, A\} = N_4$

G'**:** $S \to aS$, $S \to SS$, $S \to ba$, $S \to b$,

$A \to BB$, $A \to B$,

$B \to CC$, $B \to C$,

$B \to aAbC$, $B \to abC$, $B \to aAb$, $B \to ab$

Die Ableitung $S \underset{G}{\vdash} bA \underset{G}{\vdash} bBB \underset{G}{\vdash} bCCB \underset{G}{\vdash} bCCCC \overset{4}{\underset{G}{\vdash}} b$ kann in G' direkt abgeleitet werden:

$S \underset{G'}{\vdash} b$

Definition 8.8 Eine kontextfreie Grammatik heißt ε-frei, falls gilt:

1. Sie enthält keine Produktion $A \to \varepsilon$ für $A \neq S$.
2. Ist $S \to \varepsilon$ enthalten, so kommt S nicht auf einer rechten Seite vor.

Satz 8.9 Zu einer kontextfreien Grammatik G kann man effektiv eine äquivalente ε-freie kontextfreie Grammatik G'' konstruieren.

Beweis: Konstruiere G' wie im Beweis von Lemma 8.7. Ist $\varepsilon \notin L(G)$ (d.h. $S \not\overset{*}{\underset{G}{\vdash}} \varepsilon$, also $S \notin N_k$), so $G'' := G'$ und wir sind fertig. Sonst erweitere G' um ein neues Startsymbol S_0 und die Produktionen $S_0 \to \varepsilon$ und $S_0 \to S$.

□

Korollar 8.10 $\mathcal{L}_2 \subseteq \mathcal{L}_1$

Beweis: ε-freie kontextfreie Grammatiken sind auch kontextsensitiv.

□

Satz 8.11 Zu jeder kontextfreien Grammatik kann man effektiv eine äquivalente kontextfreie Grammatik konstruieren, die keine Produktionen der Form $A \to B$ für $A, B \in N$ enthält.

Beweisskizze:

1. Bestimme zu jedem $A \in N$

   $N(A) := \{B \in N \mid A \overset{*}{\underset{G}{\vdash}} B\}$ (effektiv berechenbar)

2. $P' = \{A \to w \mid B \to w \in P, B \in N(A), w \notin N\}$

□

Beispiel: $P = \{S \to A, A \to B, B \to aA, B \to b\}$

$N(S) = \{S, A, B\}$

$N(A) = \{A, B\}$, $N(B) = \{B\}$

$(S \to A, A \to B)$ fallen weg

$P' = \{B \to aA, A \to aA, S \to aA, B \to b, A \to b, S \to b\}$.

Chomsky-Normalform: nur Produktionen der Form

- $A \to a$ $A \in N$, $a \in \Sigma$
- $A \to BC$ $A, B, C \in N$
- $S \to \varepsilon$ Wobei s nicht auf der rechten Seite einer Produktion vorkommt

Satz 8.12 Jede kontextfreie Grammatik läßt sich effektiv umformen in eine äquivalente kontextfreie Grammatik in Chomsky-Normalform.

Beweis:

1. Konstruiere eine äquivalente ε-freie Grammatik ohne Produktionen der Form $A \to B$. (Reihenfolge! Erst $A \to \varepsilon$ raus und dann $A \to B$.)
2. Führe für jedes $a \in \Sigma$ ein neues Nichtterminalsymbol X_a und die Produktion $X_a \to a$ ein.
3. Ersetze in jeder Produktion $A \to w$ mit $w \notin \Sigma$ und $|w| > 1$ alle Terminalsymbole a durch die zugehörigen X_a's.

4. Produktionen der Form $A \to B_1 \cdots B_n$ für $n > 2$ werden ersetzt durch $A \to B_1 C_1, C_1 \to B_2 C_2, \cdots, C_{n-2} \to B_{n-1} B_n$ wobei die C_i jeweils neue Nichtterminalsymbole sind.

□

Aus dem Satz ergibt sich, daß das Wortproblem ("$w \in L(G)$?") für kontextfreie Grammatiken entscheidbar ist.

Bei endlichen Automaten/rechtslinearen Grammatiken:
Pfad im Automaten/Ableitung in der Grammatik ist von der Länge $|w|$. Davon gibt es nur endlich viele. Man kann sie also alle durchprobieren.

Bei allgemeinen kontextfreien Grammatiken:
Durch die Länge des Worts kann die Länge der Ableitung nicht beschränkt werden.

Bei kontextfreien Grammatiken in Chomsky-Normalform:

- Produktionen der Form $A \to BC$ verlängern um 1.
  $\Rightarrow$ Sie werden $|w| - 1$ mal angewandt.
- Produktionen der Form $A \to a$ erzeugen 1 Terminalsymbol.
  $\Rightarrow$ Sie werden $|w|$ mal angewandt.

Es folgt: $w \in L(G) \setminus \{\varepsilon\}$ wird durch Ableitung der Länge $2|w| - 1$ erzeugt.

Da es im allgemeinen $\geq 2^n$ Ableitungen der Länge n geben kann, liefert dies ein exponentielles Verfahren.

Ein besseres Verfahren (kubisch: n^3) liefert die folgende Überlegung:

Definition 8.13 Es sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik in Chomsky-Normalform und $w = a_1 \cdots a_n \in \Sigma^+$.
Wir definieren:

- $w_{ij} := a_i \cdots a_j$
- $N_{ij} := \{A \in N \mid A \overset{*}{\underset{G}{\vdash}} w_{ij}\}$

Damit gilt:

1. $S \in N_{1n} \Leftrightarrow w \in L(G)$
2. $A \in N_{ii} \Leftrightarrow A \overset{*}{\underset{G}{\vdash}} a_i \Leftrightarrow A \to a_i \in P$ (mit Chomsky-Normalform)

3.

$$\begin{aligned} A \in N_{ij}(i<j) &\Leftrightarrow A \overset{*}{\underset{G}{\vdash}} a_i \cdots a_j \\ &\Leftrightarrow \exists A \to BC \in P \text{ und } i \le k < j \text{ mit } B \overset{*}{\underset{G}{\vdash}} a_i \cdots a_k, C \overset{*}{\underset{G}{\vdash}} a_{k+1} \cdots a_j \\ &\Leftrightarrow \exists A \to BC \in P \text{ und } i \le k < j \text{ mit } B \in N_{ik} \text{ und } C \in N_{k+1j} \end{aligned}$$

Das liefert das folgende iterative Verfahren zur Berechnung der N_{ij} (und damit von N_{1n}):

Algorithmus 8.14 (CYK-Algorithmus (Cocke Younger Kasami))

```
for i:=1 to n do N_ii := {A | A → a_j ∈ P} od;
for d:=1 to n-1 do      //wachsende Differenz
   for i:=1 to n-d do
      j:=i+d;            //d=j-i
      N_ij := ∅;
      for k:=i to j-1 do
         N_ij := N_ij ∪ {A | A → BC ∈ P, B ∈ N_ik, C ∈ N_k+1j};
                                  kleinere Differenz
      od
   od
od
```

Satz 8.15 Für eine gegebene kontextfreie Grammatik in Chomsky-Normalform entscheidet Algorithmus 8.14 die Frage "$w \in L(G)$?" in der Zeit $O(|w|^3)$.

Beweis: Drei geschachtelte Schleifen, die jeweils $\le |w| = n$ Schritte machen.

$\Rightarrow |w|^3$ Schritte in der innersten Schleife.

□

Beachte: Die Größe von G ist dabei als konstant angenommen.

Beispiel: Sei $P = \{S \to SA \mid a,\ A \to BS,\ B \to BB \mid BS \mid b \mid c\}$ eine Grammatik in Chomsky Normalform.

Frage: Ist $w = abacba$ durch P erzeugbar?

i \ j	**1**	**2**	**3**	**4**	**5**	**6**
1	S	$\emptyset$	S	$\emptyset$	$\emptyset$	**S**
2		B	A, B	B	B	A, B
3			S	$\emptyset$	$\emptyset$	S
4				B	B	A, B
5					B	A, B
6						S
w	**a**	**b**	**a**	**c**	**b**	**a**

$\Rightarrow S \in N_{1,6} = \{S\} \Rightarrow w \in L(G)$

Greibach-Normalform:

Die Greibach-Normalform enthält nur Produktionen der Form:

$A \to aw (A \in N, a \in \Sigma, w \in N^*)$

$S \to \varepsilon$, wobei s nicht auf der rechten Seite vorkommt.

Satz 8.16 Jede kontextfreie Grammatik läßt sich effektiv umformen in eine Grammatik in Greibach-Normalform.

(ohne Beweis)

9 Abschlußeigenschaften kontextfreier Sprachen

Satz 9.1 Die Klasse $\mathcal{L}_2$ der kontextfreien Sprachen ist abgeschlossen unter Vereinigung, Konkatenation und Stern.

Beweis: Es sei $L_1 = L(G_1)$ und $L_2 = L(G_2)$ für kontextfreie Grammatiken $G_i = (N_i, \Sigma, P_i, S_i)$. Ohne Beschränkung der Allgemeinheit sei $N_1 \cap N_2 = \emptyset$.

1. $G := (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{\mathrm{S} \to S_1 \mid S_2\}, S)$, mit S neues Startsymbol und $S \notin N_1 \cup N_2$ ist eine kontextfreie Grammatik für $L_1 \cup L_2$.
2. $G' := (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$ ist eine kontextfreie Grammatik für $L_1 L_2$.
3. $G'' := (N_1 \cup \{S\}, \Sigma, P_1 \cup \{S \to \varepsilon \mid SS_1\}, S)$ ist eine kontextfreie Grammatik für L_1^*.

□

Der Abschluß unter $\cap, ^-$ gilt nicht!

Eine geeignete Methode, um zu zeigen, daß eine Sprache nicht kontextfrei ist, ist das zugehörige Pumping Lemma. Dazu definieren wir jedoch erst Ableitungsbäume, die das Pumping Lemma vereinfachen werden.

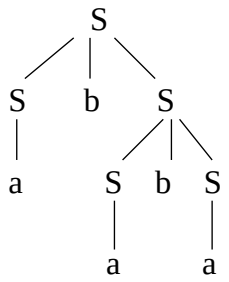
Beispiel 9.2 Sei $P = \{S \to SbS \mid a\}$.

Drei Ableitungen von $w = ababa$ sind zum Beispiel:

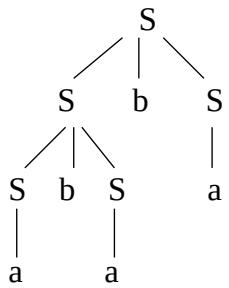
1. $S \vdash SbS \vdash abS \vdash abSbS \vdash ababS \vdash ababa$
2. $S \vdash SbS \vdash abS \vdash abSbS \vdash abSba \vdash ababa$
3. $S \vdash SbS \vdash Sba \vdash SbSba \vdash Sbaba \vdash ababa$

Die zugehörigen Ableitungsbäume sind dann:

1.



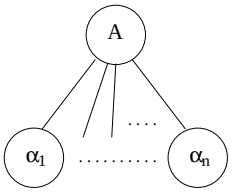
2. Sieht genauso aus wie der obige Baum!
3.



- Ein Ableitungsbaum steht für mehrere Ableitungen.

- Es existieren u.U. mehrere Bäume für ein Wort.

Allgemein: Knoten sind mit Elementen von $\Sigma \cup N$ beschriftet,



falls $A \to \alpha_1 \dots \alpha_n \in P$.

Ein Ableitungsbaum, dessen Wurzel mit A und dessen Blätter (von links nach rechts gelesen) mit $\alpha_1 \dots \alpha_n$ beschriftet sind, entspricht der Ableitung $A \overset{*}{\underset{G}{\vdash}} \alpha_1 \dots \alpha_n$.

Lemma 9.3 (Pumping Lemma für kontextfreie Sprachen) Sei G eine kontextfreie Grammatik, die ε-frei ist und keine Übergänge der Form $A \to B$ erlaubt. Es sei m die Anzahl der Nichtterminalsymbole und k die Länge der längsten rechten Seite einer Produktion.

Es sei $n_0 := k^{m+1}$.

Dann gibt es für jedes $z \in L(G)$ mit $|z| > n_0$ eine Zerlegung $z = uvwxy$ mit:

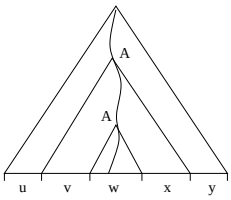
- $vx \neq \varepsilon$ und $|vwx| \leq n_0$
- $uv^i wx^i y \in L(G)$ für alle $i \geq 0$

Beweis:

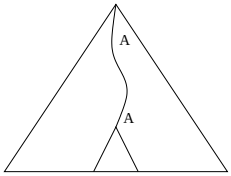
1. Ein Baum der Tiefe $\leq t$ und der Verzweigung $\leq k$ hat maximal k^t viele Blätter (s. Datenstrukturen und Algorithmen).

   Da der Ableitungsbaum für z eine Blattanzahl $|z| > k^{m+1}$ hat, ist die maximale Tiefe $> m + 1$.

2. Auf dem längsten Pfad im Ableitungsbaum stehen also mindestens $m + 1$ Nichtterminalsymbole. Es gibt aber nur m verschiedene, das heißt, eines taucht mindestens zweimal auf (zum Beispiel hier das A):



Ohne Einschränkung gebe es in



keine weiteren Wiederholungen von Nichtterminalsymbolen.

3. Es gilt:

   $S \overset{*}{\underset{G}{\vdash}} uAy$

   $A \overset{*}{\underset{G}{\vdash}} vAx$

   $A \overset{*}{\underset{G}{\vdash}} w$

   Daraus folgt:

   $S \overset{*}{\underset{G}{\vdash}} uAy \overset{*}{\underset{G}{\vdash}} uv^iAx^iy \overset{*}{\underset{G}{\vdash}} uv^iwx^iy$

4. $vx \neq \varepsilon$: Da G ε-frei ist, wäre sonst $A \overset{+}{\underset{G}{\vdash}} vAx$ nur bei Anwesenheit von Produktionen der Form $A \to B$ möglich.

5. $|vwx| \leq n_0 = k^{m+1}$: Wäre vwx länger, so müßte es noch weitere Wiederholungen enthalten.

□

Satz 9.4 $\mathcal{L}_2 \subset \mathcal{L}_1$

Beweis: Wir zeigen, daß $L := \{a^nb^nc^n \mid n \geq 1\} \in \mathcal{L}_1 \setminus \mathcal{L}_2$.

1. $\underline{L \notin \mathcal{L}_2}$:

   Angenommen, L ist kontextfrei. Dann gibt es eine ε-freie kontextfreie Grammatik ohne Produktionen der Form $A \to B$ für L. Es sei nun $n_0 = k^{m+1}$ die zugehörige Zahl aus Lemma 9.3. Wir betrachten $z := a^{n_0}b^{n_0}c^{n_0} \in L$. Mit Lemma 9.3 gibt es eine Zerlegung $z = uvwxy$ mit $vx \neq \varepsilon$ und $uv^iwx^iy \in L$.

   $\underline{\text{1. Fall}}$: v enthält verschiedene Buchstaben. Dann ist aber $uv^2wx^2y \notin a^*b^*c^* \supseteq L$ (Widerspruch!).

   $\underline{\text{2. Fall}}$: x enthält verschiedene Buchstaben führt analog zum Widerspruch.

   $\underline{\text{3. Fall}}$: v enthält lauter gleiche Buchstaben und x auch. Dann gibt es einen Buchstaben aus $\{a, b, c\}$, der in vx nicht vorkommt. Daher kommt dieser in uv^0wx^0y weiterhin n_0-mal vor, aber $|uv^0wx^0y| < 3n_0$. Damit folgt $uv^0wx^0y \notin L$ (Widerspruch!).

2. Es bleibt zu zeigen, daß die Sprache kontextsensitiv ist. Das Beispiel 6.3 liefert "fast" eine kontextsensitive Grammatik für L:

   $cB \to Bc$ ist nicht kontextsensitiv.

   Daher modifizieren wir G wie folgt:

   - Ersetze überall c durch das Nichtterminalsymbol C und führe die Produktion $C \to c$ ein.
     $S \to aSBC \mid abC$
     $CB \to BC$ nicht kontextsensitiv
     $bB \to bb$
     $C \to c$
   - Ersetze $CB \to BC$ durch die kontextsensitive Produktion
     $CB \to A_1B$
     $A_1B \to A_1A_2$
     $A_1A_2 \to BA_2$
     $BA_2 \to BC$
     Diese können nur dazu verwendet werden $CB \to BC$ zu simulieren.

□

$\underline{\text{Beachte:}}$ Auf diese Art kann man zeigen, daß man jede nicht-kürzende Produktion $u \to v$ mit $|u| \leq |v|$ durch kontextsensitive Produktionen ersetzen kann, ohne die Sprache zu ändern.

Korollar 9.5 $\mathcal{L}_2$ ist nicht unter Durchschnitt und Komplement abgeschlossen.

Beweis:

1. $\{a^n b^n c^m \mid n \geq 1, m \geq 1\} = \{a^n b^n \mid n \geq 1\} \cdot c^+ \in \mathcal{L}_2$ (Abschluß $\mathcal{L}_2$ unter $\cdot$)
   $\{a^m b^n c^n \mid n \geq 1, m \geq 1\} \in \mathcal{L}_2$
   $\{a^n b^n c^m \mid n \geq 1, m \geq 1\} \cap \{a^m b^n c^n \mid n \geq 1, m \geq 1\}$
   $= \{a^n b^n c^n \mid n \geq 1\} \notin \mathcal{L}_2$

2. $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
   $\mathcal{L}_2$ ist abgeschlossen unter $\cup$.
   Wäre $\mathcal{L}_2$ abgeschlossen unter $^-$, so auch unter $\cap$.

□

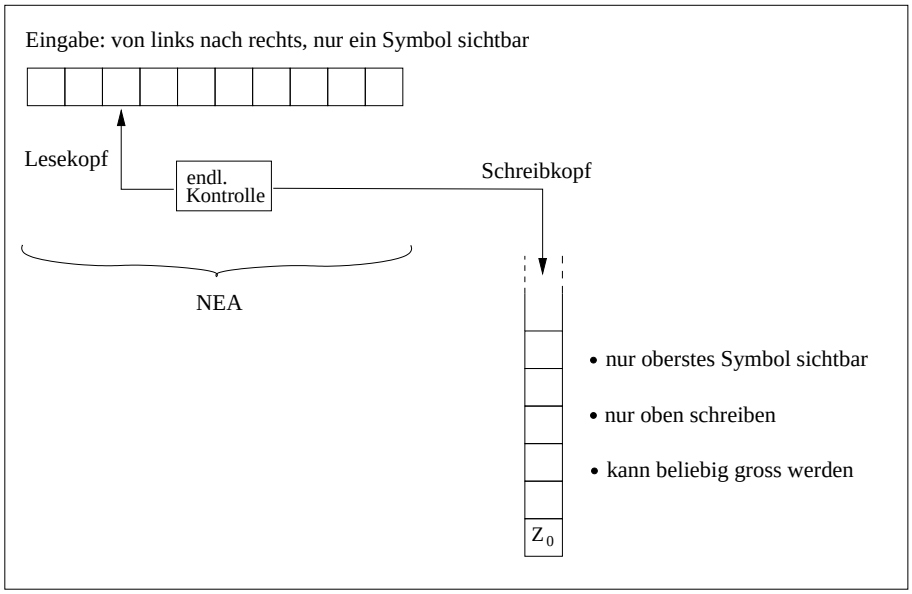
Beachte: Man kann daher das Äquivalenzproblem nicht mehr so einfach auf das Leerheitsproblem reduzieren.

$L_1 = L_2$ gdw $\underbrace{(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)}_{\text{muß nicht mehr kf sein}} = \emptyset$

Wir werden sehen: Das Äquivalenzproblem für kontextfreie Sprachen ist unentscheidbar.

10 Killer(au)tomaten

Automatenmodell, das genau die kontextfreien Sprachen akzeptiert. Endliche Automaten genügen dafür nicht. Wir erweitern endliche Automaten um eine zusätzliche (potentiell unendliche) Speicherkomponente (Keller).



Definition 10.1 Ein Kellerautomat (pushdown automaton, PDA) hat die Form $A = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta, F)$, wobei:

- Q endliche Zustandsmenge
- Σ Eingabealphabet
- Γ Kelleralphabet
- $q_0 \in Q$ Anfangszustand
- $Z_0 \in \Gamma$ Kellerstartsymbol
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times \Gamma^* \times Q$ endliche Übergangsrelation
- $F \subseteq Q$ Endzustandsmenge

Anschaulich bedeutet:

$(q, a, Z, \gamma, q') \in \Delta$: Im Zustand q mit aktuellem Eingabesymbol a und oberstem Kellersymbol Z darf der Automat Z durch γ ersetzen, in den Zustand q' gehen und zum nächsten Eingabesymbol gehen.

$(q, \varepsilon, Z, \gamma, q') \in \Delta$: Wie oben, nur ist das Eingabesymbol irrelevant und es wird nicht zum nächsten Eingabesymbol übergegangen.

Den aktuellen Zustand (Konfiguration) einer Kellerautomatenberechnung beschreibt man durch:

- den noch zu lesenden Rest $w \in \Sigma^*$ des Eingabebandes (Lesekopf steht auf erstem Symbol von w)
- den Zustand $q \in Q$
- den Kellerinhalt $\gamma \in \Gamma^*$ (Der Schreiblesekopf steht auf dem ersten Symbol von γ.)

Definition 10.2 Eine Konfiguration von A ist von der Form

$$\kappa = (q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*.$$

Konfigurationsübergänge:

- $(q, aw, Z\gamma) \vdash (q', w, \beta\gamma)$ falls $(q, a, Z, \beta, q') \in \Delta$
- $(q, w, Z\gamma) \vdash (q', w, \beta\gamma)$ falls $(q, \varepsilon, Z, \beta, q') \in \Delta$
- $\kappa \overset{*}{\vdash} \kappa'$ gdw $\exists n \geq 0, \kappa_0, \ldots, \kappa_n$ mit $\kappa = \kappa_0, \kappa' = \kappa_n, \kappa_i \vdash \kappa_{i+1}$ $(0 \leq i < n)$

Der Automat A akzeptiert das Wort $w \in \Sigma^*$ gdw $(q_0, w, Z_0) \overset{*}{\vdash} (q, \varepsilon, \gamma)$ mit $q \in F, \gamma \in \Gamma^*$.

Die von A akzeptierte Sprache ist $L(A) := \{w \in \Sigma^* \mid A \text{ akzeptiert } w\}$.

Beispiel 10.3 ein PDA für $\{a^n b^n \mid n \geq 1\}$

$Q = \{q_0, q_1, f\}$, $\Gamma = \{Z, Z_0\}$, $\Sigma = \{a, b\}$, $F = \{f\}$

Δ:	q_0	a	Z_0	ZZ_0	q_0	(erstes a, speichere Z)
	q_0	a	Z	ZZ	q_0	(weiteres a, speichere Z)
	q_0	b	Z	ε	q_1	(erstes b, lösche Z)
	q_1	b	Z	ε	q_1	(weiteres b, lösche Z)
	q_1	ε	Z_0	ε	f	

Konfigurationsübergänge:

1. $(q_0, aabb, Z_0) \vdash (q_0, abb, ZZ_0) \vdash (q_0, bb, ZZZ_0) \vdash (q_1, b, ZZ_0) \vdash (q_1, \varepsilon, Z_0)$ $\vdash \underbrace{(f, \varepsilon, \varepsilon)}_{\text{akzeptiert}}$

2. $(q_0, aab, Z_0) \overset{*}{\vdash} (q_0, b, ZZZ_0) \vdash (q_1, \varepsilon, ZZ_0)$, wobei $q_1 \notin F \Rightarrow$ nicht akzeptiert

3. $(q_0, abb, Z_0) \vdash (q_0, bb, ZZ_0) \vdash (q_1, b, Z_0) \vdash (f, b, \varepsilon)$, wobei $b \neq \varepsilon \Rightarrow$ nicht akzeptiert

Der Kellerautomat aus dem Beispiel ist deterministisch, d.h. es ist stets höchstens ein Konfigurationsübergang möglich.

Definition 10.4 Der PDA $A = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta, F)$ heißt deterministisch, falls die folgenden Bedingungen erfüllt sind:

1. $\forall q \in Q \quad \forall a \in \Sigma \cup \{\varepsilon\} \quad \forall Z \in \Gamma$ existiert höchstens ein Übergang der Form $(q, a, Z, \ldots, \ldots) \in \Delta$.

2. Existiert ein Tupel $(q, \varepsilon, Z, \ldots, \ldots) \in \Delta$, so existiert kein Tupel der Form $(q, a, Z, \ldots, \ldots) \in \Delta$ mit $a \in \Sigma$.

Beispiel 10.5 $L := \{w\overleftarrow{w} \mid w \in \{a, b\}^*\}$ (mit $a_1 \overleftarrow{\ldots} a_n := a_n \ldots a_1$) wird von einem nichtdeterministischen PDA akzeptiert.

Idee: Der PDA legt die erste Worthälfte im Keller ab (automatisch in umgekehrter Reihenfolge) und vergleicht den Kellerinhalt dann mit der zweiten Worthälfte.

Nichtdeterminismus ist nötig, da man raten muß, wann die erste Hälfte gelesen ist.

$Q = \{q_0, q_1, q_2, f\}, \Gamma = \{a, b, Z_0\}, \Sigma = \{a, b\}, F = \{f\}$

Δ:	q_0	ε	Z_0	ε	f	(akzeptiert $\varepsilon = \varepsilon \cdot \Sigma$)
	q_0	a	Z_0	aZ_0	q_1	$\leftarrow$ lese und speichere die erste Worthälfte
	q_0	b	Z_0	bZ_0	q_1	
	q_1	a	a	aa	q_1	
	q_1	a	b	ab	q_1	
	q_1	b	a	ba	q_1	
	q_1	b	b	bb	q_1	
	q_1	a	a	ε	q_2	$\leftarrow$ lese zweite Worthälfte, Vergleich mit erster
	q_1	b	b	ε	q_2	
	q_2	b	b	ε	q_2	
	q_2	a	a	ε	q_2	
	q_2	ε	Z_0	ε	f	$\leftarrow$ Übergang zum Endzustand

Ein anderer Akzeptanzbegriff für Kellerautomaten:

Definition 10.6 Es sei $A = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta)$ ein PDA ohne Endzustandsmenge. Wir sagen: A akzeptiert w mit leerem Keller, falls $(q_0, w, Z_0) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$ für beliebige $q \in Q$.

$N(A) := \{w \in \Sigma^* \mid A \text{ akzeptiert } w \text{ mit leerem Keller}\}$

Beispiel: Für die PDAs in Beispiel 10.3 und 10.5 gilt:

$L(A) = N(A)$. Das muß aber nicht immer so sein!

Satz 10.7 Für PDAs ist Akzeptanz mit Endzuständen äquivalent zu Akzeptanz mit leerem Keller, d.h. für eine formale Sprache L sind äquivalent:

1. $L = L(A)$ für einen PDA A
2. $L = N(B)$ für einen PDA B

Beweis:

"1 → 2": B arbeitet im Prinzip wie A. Beim Erreichen eines Endzustandes leert er den Keller. Zusätzlich muß verhindert werden, daß der Keller leer wird, ohne daß ein Endzustand erreicht war.

A wird erweitert um:

- neuen Anfangszustand q_0'
- neuen Zustand q_1 zum Leeren des Kellers
- neues Kellerstartsymbol X_0
- Übergänge:

q_0'	ε	X_0	Z_0X_0	q_0	
f	ε	Z	ε	q_1	für $Z \in \Gamma \cup \{X_0\}, f \in F$
q_1	ε	Z	ε	q_1	für $Z \in \Gamma \cup \{X_0\}$

Da X_0 in den Übergängen von A nicht vorkommt, kann es nur entfernt werden, wenn ein Endzustand erreicht wurde.

"2 → 1": A arbeitet im Prinzip wie B. Zusätzlich muß A feststellen, wann der Keller leer ist, und dann in einen Endzustand übergehen. Erkennen des leeren Kellers durch ein neues Kellerstartsymbol.

B wird ergänzt um:

- neuen Anfangszustand q_0'
- neuen Zustand f, der einziger Endzustand ist
- neues Kellerstartsymbol X_0
- Übergänge:

q_0'	ε	X_0	Z_0X_0	q_0	
q	ε	X_0	ε	f	für alle $q \in Q$

□

Satz 10.8 Für eine formale Sprache L sind äquivalent:

1. $L = L(G)$ für eine kontextfreie Grammatik G
2. $L = N(A)$ für einen PDA A

Beweis:

"1 → 2": Es sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik mit $L = L(G)$. Der zugehörige PDA simuliert Linksableitungen von G, d.h. Ableitungen bei denen stets das am weitesten links stehende Nichtterminalsymbol ersetzt wird.

Beachte: Jedes Wort in $L(G)$ kann durch eine Linksableitung erzeugt werden.

Wir definieren $A = (\{q\}, \Sigma, \underbrace{\Sigma \cup N}_{P}, q, S, \Delta)$ mit

$$\Delta := \underbrace{\{(q, \varepsilon, A, \gamma, q) \mid A \to \gamma \in P\}}_{(*)} \cup \underbrace{\{(q, a, a, \varepsilon, q) \mid a \in \Sigma\}}_{(**)}$$

$(*)$: Anwenden einer Produktion.
$(**)$: Entfernen von Terminalsymbolen aus dem Keller, wenn sie mit der Eingabe übereinstimmen.

Beispiel: $\{S \to \varepsilon, S \to aSa, S \to bSb\}$ liefert Übergänge:

q	ε	S	ε	q
q	ε	S	aSa	q
q	ε	S	bSb	q
q	a	a	ε	q
q	b	b	ε	q

Die Ableitung $S \vdash aSa \vdash abSba \vdash abba$ entspricht der Konfigurationfolge $(q, abba, S) \vdash (q, abba, aSa) \vdash (q, bba, Sa) \vdash (q, bba, bSba) \vdash (q, ba, Sba) \vdash (q, ba, ba) \vdash (q, a, a) \vdash (q, \varepsilon, \varepsilon)$

Behauptung: Für $u, v \in \Sigma^*$ und $\alpha \in \{\varepsilon\} \cup N(\Sigma \cup N)^*$ gilt $S \overset{*}{\underset{G}{\vdash}} u\alpha$ gdw $(q, uv, S) \overset{*}{\underset{A}{\vdash}} (q, v, \alpha)$ Linksableitung

Beachte: Für $a = \varepsilon = v$ folgt

$$\underbrace{S \overset{*}{\underset{G}{\vdash}} u}_{u \in L(G)} \Leftrightarrow \underbrace{(q, u, S) \overset{*}{\underset{A}{\vdash}} (q, \varepsilon, \varepsilon)}_{u \in N(A)}$$

Beweis der Behauptung:

"⇐" Induktion über die Anzahl der Übergänge mit Transitionen der Form $(q, \varepsilon, A, \gamma, q)$

"⇒" Induktion über die Länge der Linksableitung

"2 → 1": Es sei $A = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta)$ ein PDA mit $L = N(A)$ und zugehöriger Grammatik G.

Nichtterminalsymbole sind Tripel $[p, Z, q] \in Q \times \Gamma \times Q$.

Idee: Es soll gelten: $[p, Z, q] \overset{*}{\underset{G}{\vdash}} u \in \Sigma^*$ gdw A kommt vom Zustand p in den Zustand q durch Lesen von u auf dem Eingabeband und Löschen von Z aus dem Keller (ohne die Symbole unter Z anzutasten).

Wie realisiert man das durch Produktionen?

PDA-Übergang: $(p, a, Z, X_1 \dots X_n, p') \in \Delta$

Hier wird a auf dem Eingabeband verbraucht und Z durch $X_1 \dots X_n$ ersetzt. Um den Kellerinhalt unter Z zu erreichen, müssen $X_1 \dots X_n$ entfernt werden. Löschen von X_i kann durch ein Nichtterminalsymbol $[p_{i-1}, x_i, p_i]$ realisiert werden.

Formale Definition:

$G = (N, \Sigma, P, S)$ mit $N := \{S\} \cup \{[p, Z, q] \mid p, q \in Q, Z \in \Gamma\}$

$P := \{S \to [q_0, Z_0, q] \mid q \in Q\} \cup \{[p, Z, q] \to a[p_0 X_1 p_1] \dots [p_{n-1} X_n p_n] \mid (p, a, Z, X_1 \dots X_n, p_0) \in \Delta, p, q, p_0, \dots p_n \in Q, p_n = q\}$

Beispiel: (vgl. Beispiel 10.3)

$(q_0, ab, Z_0) \vdash (q_0, b, ZZ_0) \vdash (q_1, \varepsilon, Z_0) \vdash (f, \varepsilon, \varepsilon)$

entspricht der Ableitung:

$S \underset{G}{\vdash} [q_0, Z_0, f] \underset{G}{\vdash} a[q_0, Z, q_1][q_1, Z_0, f] \underset{G}{\vdash} ab[q_1, Z_0, f] \underset{G}{\vdash} ab$

Behauptung: Für alle $p, q \in Q, u \in \Sigma^*, Z \in \Gamma$ gilt:

$(p, u, Z) \overset{*}{\underset{A}{\vdash}} (q, \varepsilon, \varepsilon)$ gdw $[p, Z, q] \overset{*}{\underset{G}{\vdash}} u$

Für $p = q_0, z = z_0$: $\underbrace{(q_0, u, z_0) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)}_{u \in N(A)}$ gdw $\underbrace{S \vdash [q_0, Z_0, q] \overset{*}{\vdash} u}_{u \in L(G)}$

□

Eigenschaften von kontextfreien Sprachen mit Hilfe von Kellerautomaten zeigen:

Korollar 10.9 Es sei L kontextfrei und R regulär. Dann ist $L \cap R$ kontextfrei.

Beweis: Es sei $L = L(A)$ für einen PDA $A = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta, F)$ und $R = L(A')$ für einen DEA $A' = (Q', \Sigma, q_0', \delta', F')$.

Produktautomat:

$B := (Q \times Q', \Sigma, \Gamma, (q_0, q_0'), Z_0, \Delta', F \times F')$

$\Delta' := \{((p, p'), a, Z, \gamma, (q, q')) \mid a \in \Sigma, (p, a, Z, \gamma, q) \in \Delta$ und $\delta(p', a) = q'\} \cup \{((p, p'), \varepsilon, Z, \gamma, (q, p')) \mid (p, \varepsilon, Z, \gamma, q) \in \Delta\}$

Man zeigt leicht durch Induktion:

$((p, p'), uv, \gamma) \overset{k}{\underset{B}{\vdash}} ((q, q'), v, \beta)$ gdw $(p, uv, \gamma) \overset{k}{\underset{A}{\vdash}} (q, v, \beta)$ und $p' \overset{u}{\underset{A'}{\to}} q'$

Daraus folgt:

$((q_0, q_0'), u, Z_0) \overset{*}{\underset{B}{\vdash}} ((f, f'), \varepsilon, \beta)$ gdw $\underbrace{(q_0, u, Z_0) \overset{*}{\underset{A}{\vdash}} (f, \varepsilon, \beta)}_{u \in L(A)}$ und $\underbrace{q_0' \overset{u}{\underset{A'}{\to}} f'}_{u \in L(A')}$

□

Bemerkung 10.10 Ist A in dem Beweis deterministisch, so auch der konstruierte Automat B, da wir o.E. einen deterministischen endlichen Automaten A' für R gewählt hatten.

L wird von einem PDA akzeptiert $\Leftrightarrow$ L ist kontextfrei.

Beispiel 10.11 Die Sprache $L = \{ww \mid w \in \{a, b\}^*\}$ ist nicht kontextfrei.

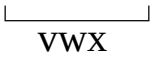
Beweis: Angenommen, L ist kontextfrei, dann auch

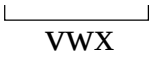
$$L' := L \cap a^+b^+a^+b^+ = \{a^ib^ja^ib^j \mid i,j \geq 1\}$$

Es sei nun n die Zahl aus dem Pumping Lemma für L'. Wir betrachten $z := a^nb^na^nb^n \in L'$.

Es gibt eine Zerlegung $z = uvwxy$ mit $|vwx| \leq n$ und $vx \neq \varepsilon$, so daß $uwy \in L'$.

Da $|vwx| \leq n$ ist, kann vwx höchstens 2 aufeinanderfolgende a^n/b^n-Blöcke berühren.

1. Fall 

2.Fall 

3.Fall 

In allen drei Fällen kann uwy kein Element von L' sein.

Also ist L nicht kontextfrei.

□

Bei endlichen Automaten gibt es die Potenzmengenkonstruktion, mit deren Hilfe man aus einem NEA einen DEA errechnen kann. Für Kellerautomaten ist so eine Konstruktion nicht möglich.

Definition 10.12 Eine Sprache heißt deterministisch kontextfrei (dkf), falls sie von einem deterministischen Kellerautomaten akzeptiert wird.

Ziel: $L = \{w\overleftarrow{w} \mid w \in \{a,b\}^*\}$ ist kontextfrei aber nicht deterministisch kontextfrei.

Wir wissen bereits: L ist kontextfrei (Beispiel 10.5).

Definition 10.13 Für $L \subseteq \Sigma^*$ sei
$\text{Min}(L) := \{w \in L \mid$ kein echtes Präfix von w liegt in $L\}$.

Lemma 10.14 Ist L deterministisch kontextfrei, so auch $\text{Min}(L)$.

Beweis: Es sei $A = (Q, \Sigma, \Gamma, q_0, z_0, \Delta, F)$ ein DPDA mit $L = L(A)$. Wir ändern A so ab, daß gilt: Wurde das erste Mal ein Endzustand erreicht, so kann danach keiner mehr erreicht werden:

$$A' = (Q', \Sigma, \Gamma, q_0, Z_0, \Delta', F)$$

mit

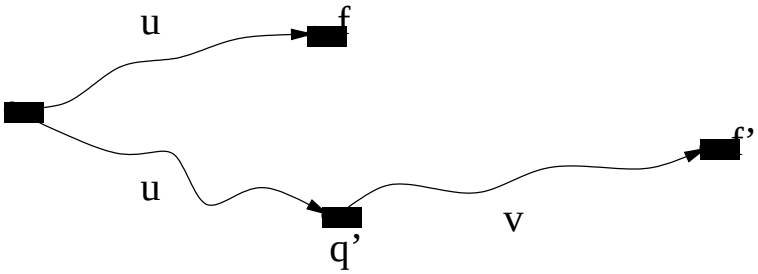
$Q' = Q \cup \{\hat{q}\}$ $\quad \hat{q}$ Papierkorbzustand

$$\begin{aligned}\Delta' :=& (\Delta \setminus (F \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times \Gamma^* \times Q)) \\ & \cup \{(q, a, Z, Z, \hat{q}) \mid q \in F, a \in \Sigma, Z \in \Gamma\} \\ & \cup \{(\hat{q}, a, Z, Z, \hat{q}) \mid a \in \Sigma, Z \in \Gamma\}\end{aligned}$$

Man sieht leicht: A' ist deterministisch und $L(A') = \text{Min}(L)$.

□

Beachte: Bei einem nichtdeterministischen PDA funktioniert diese Konstruktion nicht:



Satz 10.15 $L := \{w\overleftarrow{w} \mid w \in \{a, b\}^*\}$ ist nicht deterministisch kontextfrei.

Beweis: Wäre L deterministisch kontextfrei, so wäre mit Bemerkung 10.10 und Lemma 10.14 auch $L' := \text{Min}(L) \cap (ab)^+(ba)^+(ab)^+(ba)^+$ deterministisch kontextfrei.

Es ist $L' = \{(ab)^i(ba)^j(ab)^j(ba)^i \mid i > j > 0\}$ (für $i \leq j$ wäre $(ab)^i(ba)^i \in L$ echtes Präfix).

Da L' kontextfrei ist, gilt das Pumping Lemma. Sei n Zahl aus dem Pumping Lemma. Betrachte:

$$z = (ab)^{n+1}(ba)^n(ab)^n(ba)^{n+1}$$

Damit hat z eine Zerlegung $z = uvwxy$ mit $|vwx| \leq n$, $vx \neq \varepsilon$ und $uv^iwx^iy \in L'$ für alle $i \geq 0$.

1. Fall: vwx im Mittelteil $(ba)^n(ab)^n$

Aufpumpen liefert entweder ein Wort $\notin (ab)^+(ba)^+(ab)^+(ba)^+$ oder $\notin \text{Min}(L)$.

2. Fall: vwx im linken Teil $(ab)^{n+1}(ba)^n$ oder im rechten Teil $(ab)^n(ba)^{n+1}$ liefert sogar Wort $\notin L$.

□

Deterministisch kontextfreie Sprachen sind im Compilerbau interessant, da dafür das Wortproblem linear entscheidbar ist (im Unterschied zu kubisch beim CYK-Algorithmus).

(Siehe dazu Vorlesung Compilerbau oder Buch von J. Wegener - Theoretische Informatik S.199-216.)

Satz 10.16 Ist L dkf, so auch $\overline{L}$.

(ohne Beweis)

Beachte: Bei DPDAs kann man nicht einfach F durch $Q \setminus F$ ersetzen wie bei DEAs.

Grund: Ein DPDA hat zwei Möglichkeiten ein Wort w nicht zu akzeptieren:

1. w ist ganz gelesen und der Automat ist nicht in Endzustand.
2. w wird nicht ganz gelesen.

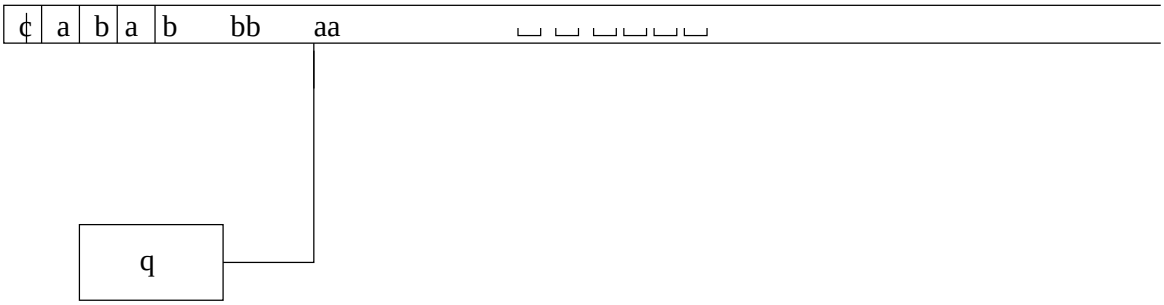
Man muß zunächst einen DPDA konstruieren, bei dem der 2. Fall nicht eintritt, dann kann man F durch $Q \setminus F$ ersetzen.

(Wegener S.193-195)

Teil III

Turingmaschinen sowie Typ0 und Typ1 Sprachen

11 Turingmaschinen



Das Band ist einseitig unendlich, Schreib-/Lesekopf kann auf dem Band lesen und schreiben, es gibt endlich viele Symbole und q ist eine endliche Kontrolle mit $q \in Q, |Q| < \infty$.

Schreib-/Lesekopf:

- Liest aktuelles Symbol,
- kann es überschreiben,
- kann nach links/rechts gehen oder stehenbleiben.

Definition 11.1 Eine nichtdeterministische Turingmaschine (NTM) über dem Eingabealphabet Σ ist von der Form

$$A = (Q, \Sigma, \Gamma, q_0, \Delta, q_{acc}, q_{rej})$$

wobei

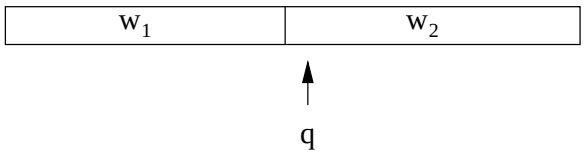
- Q endliche Zustandsmenge
- Γ Arbeitsalphabet mit $\Sigma \subseteq \Gamma, \sqcup, ¢ \in \Gamma \setminus \Sigma$

- $\Delta \subseteq$
  $\underbrace{Q \setminus \{q_{acc}, q_{rej}\}}_{\text{aktueller Zust.}} \times \underbrace{\Gamma}_{\text{gelesenes Symb.}} \times \underbrace{Q}_{\text{neuer Zust.}} \times \underbrace{\Gamma}_{\text{neues Symb.}} \times \{L, R, 0\}$
  wobei $(q, ¢, q', a', B) \in \Delta \Rightarrow a' = ¢, B \in \{R, 0\}$ damit die Randmarke nicht überschrieben wird, und die Maschine nicht über den linken Rand hinauslaufen kann.

- $q_0, q_{acc}, q_{rej} \in Q, q_{acc} \neq q_{rej}$ Hierbei ist q_0 der Anfangszustand, q_{acc} der akzeptierende Haltezustand und q_{rej} der verwerfende Haltezustand.

Berechnungszustand (Konfiguration) einer NTM:

$w_1 q w_2 \in \Gamma^* \times Q \times \Gamma^+$



Schreib-/Lesekopf steht auf dem ersten Symbol von w_2.

Linker Rand: Ist $w_1 \neq \varepsilon$, so beginnt w_1 mit ¢. .
Ist $w_1 = \varepsilon$, so beginnt w_2 mit ¢

Konfigurationsübergänge:

1. $w_1 q a w_2 \underset{A}{\vdash} w_1 q' a' w_2$ falls $(q, a, q', a', 0) \in \Delta$

2. $w_1 b q a w_2 \underset{A}{\vdash} w_1 q' b a' w_2$ falls $(q, a, q', a', L) \in \Delta$

3. $w_1 q a \underbrace{w_2}_{\neq \varepsilon} \underset{A}{\vdash} w_1 a' q' w_2$ falls $(q, a, q', a', R) \in \Delta$

   Wichtig: $w_1 q a \underset{A}{\vdash} w_1 a' q' \sqcup$ falls $(q, a, q', a', R) \in \Delta$

Definition 11.2 Die NTM A akzeptiert die Sprache

$$L(A) := \{w \in \Sigma^* \mid q_0 ¢ w \overset{*}{\underset{A}{\vdash}} w_1 q_{acc} w_2 \text{ für } w_1, w_2 \in \Gamma^*\}$$

Eine Sprache L heißt rekursiv aufzählbar, falls es eine NTM A gibt mit $L = L(A)$.

Beispiel 11.3 eine sehr einfache NTM

$$A = (\{q_0, q_{acc}, q_{rej}\}, \{a, b\}, \{a, b, ¢, \sqcup\}, q_0, \Delta, q_{acc}, q_{rej})$$

mit

Δ : $q_0, ¢ \longrightarrow q_0, ¢, R$
$q_0, a \longrightarrow q_0, a, R$
$q_0, \sqcup \longrightarrow q_{acc}, \sqcup, 0$

z.B. aa wird akzeptiert: $q_0¢aa \underset{A}{\vdash} ¢q_0aa \underset{A}{\vdash} ¢aq_0a \underset{A}{\vdash} ¢aaq_0\sqcup \underset{A}{\vdash} ¢aaq_{acc}\sqcup$

$L(A) = a^*$

Aus der Vorlesung Berechenbarkeit ist bekannt:

- Deterministische TM sind genauso stark wie nichtdeterministische TM, d.h. sie akzeptieren auch die Klasse der rekursiv aufzählbaren Sprachen.
- Mehrband-TM sind nicht stärker als die eingeführten NTM.

12 Turingmaschinen und Typ0 Sprachen

Satz 12.1 $\mathcal{L}_0 = \{L \mid L \text{ ist rekursiv aufzählbar}\}$

Lemma 12.2 Zu jeder Typ0 Grammatik $G = (N, \Sigma, P, S)$ kann man effektiv eine NTM A konstruieren mit $L(G) = L(A)$.

Beweisskizze:

Wir geben eine 2-Band-TM an:

1. Band: speichert Eingabe w
2. Band: Hierauf werden ausgehend von S gemäß den Regeln von G alle aus S ableitbaren Wörter erzeugt.

Das erzeugte Wort wird mit dem Eingabewort verglichen. Falls die Wörter identisch sind, so geht sie in q_{acc}. Sonst erzeugt sie das nächste Wort.

Genauer:

1. Kopiere w vom Eingabeband auf Band 1.

2. Schreibe S auf Band 2 und gehe nach links an den Bandanfang.

3. Gehe auf Band 2 nach rechts und wähle (nichtdeterministisch) eine Stelle, an der die linke Seite der anzuwendenden Produktion steht.

4. Wähle (nichtdeterministisch) einen der Zustände Regel $1, \ldots,$ Regel k (wobei $P = \{\alpha_i \to \beta_i \mid i = 1, \ldots, k\}$).

5. Es sei Regel i gewählt. Überprüfe, ob α_i tatsächlich die Beschriftung des Bandstücks der Länge $|\alpha_i|$ ab der gewählten Bandstelle ist.

6. Falls der Test erfolgreich ist, so ersetze α_i durch β_i. Vorher müssen bei $|\alpha_i| < |\beta_i|$ bzw $|\alpha_i| > |\beta_i|$ die Symbole $\neq \sqcup$ rechts von α_i entsprechend nach rechts bzw links verschoben werden.

7. Vergleiche das auf Band 2 erzeugte Wort mit dem auf Band 1 gespeicherten Eingabewort.

8. Wenn die Wörter gleich sind, so gehe nach q_{acc}. Sonst fahre fort mit 3.

□

Lemma 12.3 Zu jeder NTM $A = (Q, \Sigma, \Gamma, q_0, \Delta, q_{acc}, q_{rej})$ kann man effektiv eine Typ0 Grammatik G konstruieren mit $L(G) = L(A)$.

Beweisskizze: Die Grammatik G erzeugt die Wörter $w \in L(A)$ wie folgt:

1.Phase: Erzeuge genügend großes Arbeitsband.

Dazu wird ein Wort w und "genügend" viele Blanks rechts von w erzeugt.

Idee: Zur Akzeptanz von $w \in L(A)$ benötigt A nur einen endlichen Bandbereich. Die Anzahl wird nichtdeterministisch geraten.

2.Phase: Auf diesem Arbeitsband simuliert G die Berechnung von A.

3.Phase: War die Berechnung erfolgreich, so erzeuge nochmals das Wort w.

Damit in der zweiten Phase das w nicht verloren geht, verwenden wir spezielle Nichtterminalsymbole aus

$$\underbrace{(\Sigma \cup \{\sqcup\}}_{\text{Speichern von } w} \times \underbrace{\Gamma}_{\text{Berechnung}}$$

Formale Definition:

$$N := \{ \underbrace{S, A, B}_{\text{Aufbauen des Bandes}}, \underbrace{E}_{\text{Aufräumen am Schluß}} \} \cup \underbrace{Q}_{\text{Zustand der TM}} \cup (\Sigma \cup \{\sqcup\}) \times \Gamma$$

Regeln:

1. Phase: $S \longrightarrow q_0[\not\vdash, \not\vdash]A$

$A \longrightarrow [a, a]A$ für alle $a \in \Sigma$

$A \longrightarrow B$

$B \longrightarrow [\sqcup, \sqcup]B \mid \varepsilon$

Man erhält so für alle $a_1, \ldots, a_n \in \Sigma^*$ und $l \geq 0$

$$S \overset{*}{\underset{G}{\vdash}} q_0[\not\vdash, \not\vdash][a_1, a_1] \ldots [a_n, a_n][\sqcup, \sqcup]^l$$

2. Phase: Simuliert die TM-Berechnung in der "zweiten Spur".

1. $q[c, a] \longrightarrow q'[c, a']$ falls $(q, a, q', a', 0) \in \Delta$
2. $[c, b]q[c', a] \longrightarrow q'[c, b][c', a']$ falls $(q, a, q', a', L) \in \Delta$
3. $q[c, a] \longrightarrow [c, a']q'$ falls $(q, a, q', a', R) \in \Delta$

Beachte: Da wir am Anfang genügend viele Blanks erzeugt haben, müssen wir hier "Nachschieben eines Blanks" nicht behandeln.

3. Phase: Aufräumen und Erzeugen von $a_1, \ldots, a_n$, wenn die TM akzeptiert hat.

- $q_{acc}[a, b] \longrightarrow EaE$ für alle $a \in \Sigma, b \in \Gamma$
  $q_{acc}[\not\vdash, \not\vdash] \longrightarrow E$
  $q_{acc}[\sqcup, b] \longrightarrow E$ für alle $b \in \Gamma$
- $E[a, b] \longrightarrow aE \quad a \in \Sigma, b \in \Gamma$ Aufräumen nach rechts
- $[a, b]E \longrightarrow Ea \quad a \in \Sigma, b \in \Gamma$ Aufräumen nach links
- $E[\sqcup, b] \longrightarrow E \quad b \in \Gamma$
- $[\sqcup, b]E \longrightarrow E \quad b \in \Gamma$
- $[\not\vdash, \not\vdash]E \longrightarrow E$
- $E \longrightarrow \varepsilon$

Man zeigt leicht: $L(G) = L(A)$

□

Abschlußeigenschaften von Typ0 Sprachen

Satz 12.4

1. $\mathcal{L}_0$ ist abgeschlossen unter $\cup$, $\cap$, $\cdot$ und $*$.
2. $\mathcal{L}_0$ ist nicht abgeschlossen unter Komplement!

Beweis:

1. Abschluß unter $\cup$, $\cdot$, und $*$: wie bei kontextfreien Sprachen (Konstruktion von Grammatiken)

   Zu zeigen: Abschluß unter $\cap$; verwendet eine NTM.

   Die NTM für $L_1 \cap L_2$ verwendet zwei Bänder und simuliert auf einem die Berechnung der NTM für L_1 und auf dem anderen die Berechnung der NTM für L_2.

2. Die rekursiv aufzählbaren Sprachen sind nicht unter Komplement abgeschlossen (siehe BuK: L_H ist zwar rekursiv aufzählbar, aber nicht rekursiv $\Rightarrow \bar{L_H}$ ist nicht rekursiv aufzählbar!).

□

Satz 12.5 Für $\mathcal{L}_0$ sind das Leerheitsproblem, das Wortproblem und das Äquivalenzproblem unentscheidbar.

Beweis: siehe BuK: L_{empty}, L_u, L_{eq}

□

13 Linear beschränkte Automaten und Typ1 Sprachen

Typ1 Sprachen können durch eine NTM mit linearer Platzbeschränkung akzeptiert werden.

Definition 13.1 Ein linear beschränkter Automat (LBA) ist eine NTM $A = (Q, \Sigma, \Gamma, q_0, \Delta, q_{acc}, q_{rej})$, die zusätzlich ein rechtes Randsymbol $\$ \in \Gamma \setminus (\Sigma \cup \{¢\})$ enthält. Dieser Randmarker darf nicht überschrieben werden, und die NTM läuft nicht nach rechts über diesen Randmarker hinaus, d.h.

$(q, \$, q', a', B) \in \Delta \Rightarrow a' = \$$ und $B \in \{L, 0\}$.

Die von dem LBA A akzeptierte Sprache ist:

$L(A) := \{w \in \Sigma^* \mid q_0 ¢ w \$ \overset{*}{\underset{A}{\vdash}} w_1 q_{acc} w_2 \text{ für } w_1, w_2 \in \Gamma^*\}$.

Die Einschränkung sorgt dafür, daß ein LBA als Arbeitsband nur den ursprünglich vom Eingabewort eingenommenen Bandbereich verfügbar hat.

Korollar 13.2 $\mathcal{L}_1 = \{L \mid L \text{ wird von einem LBA akzeptiert }\}$

Beweis: Dieser ergibt sich direkt aus dem Beweis von Satz 12.1.

"$\subseteq$":

Da alle Produktionen von kontextsensitiven Grammatiken nichtkürzend sind (mit der Ausname $S \to \varepsilon \mid \ldots$) muß man auf dem zweiten Band nur ableitbare Wörter bis zur Länge $|w|$ erzeugen. Daher kommt man mit $|w|$ Feldern aus.

Beachte: Zwei Bänder liefern nicht ein doppelt so langes Band, sondern werden durch ein größeres Alphabet codiert.

"$\supseteq$":

Durch Modifikation der Beweisidee aus Lemma 12.3 gelingt es, zu einem LBA eine Grammatik zu definieren, die keine kürzenden Regeln enthält (d.h. $\alpha \to \beta$ mit $|\alpha| > |\beta|$). Wie im Beweis von Satz 9.4 kann man zeigen, daß man diese in eine äquivalente kontextsensitive Grammatik überführen kann.

Idee der Modifikation: Da man mit $|w|$ Arbeitsfeldern auskommt, muß man keine zusätzlichen $\sqcup$ erzeugen, und sie daher am Ende nicht löschen. Dadurch fallen kürzende Regeln weg, z.B.: $q_{acc}[\sqcup, b] \to E$ oder $E[\sqcup, b] \to E, \ldots$.

Es gibt noch einige technische Probleme:

- Löschen der Randmarker
- Löschen von E und des Zustandes q_{acc}

Lösung:

Führe diese nicht explizit ein, sondern kodiere sie in den anderen Symbolen: z.B.: statt $[a, b]q[a', b'][a'', b'']$ verwende $[a, b][q, a', b'][a'', b'']$.

□

Satz 13.3 $\mathcal{L}_1$ ist abgeschlossen unter $\cup$, $\cap$, $\cdot$, $*$ und $^-$.

Beweis:

$\cap$, $\cdot$, $*$, $\cup$: wie bei $\mathcal{L}_0$

Komplement: Schwierig; War lange ein offenes Problem; In den 80ern wurde es unabhängig voneinander von zwei Forschern im selben Jahr gelöst.

□

Es ist noch nicht bekannt, ob deterministische LBAs genauso stark sind, wie nichtdeterministische LBAs.

Satz 13.4 Für $\mathcal{L}_1$ ist das Wortproblem entscheidbar.

Beweis: Da die Produktionen nicht kürzend sind, muß man zur Entscheidung "$w \in L(G)$?" nur alle aus S ableitbaren Wörter aus $(N \cup \Sigma)^*$ der Länge $\leq |w|$ erzeugen. Dies sind nur endlich viele.

□

Korollar 13.5 $\mathcal{L}_1 \subset \mathcal{L}_0$

Beweis: Der Beweis dafür gestaltet sich trivial: In $\mathcal{L}_1$ ist das Wortproblem entscheidbar, in $\mathcal{L}_0$ allerdings schon unentscheidbar. Also stimmt obige Behauptung.

$$\mathcal{L}_0 \supset \mathcal{L}_1 \supset \mathcal{L}_2 \supset \mathcal{L}_3$$

□

Teil IV

Schwierige Entscheidungsprobleme

14 Unentscheidbare Probleme

Wir verwenden zum Nachweis der Unentscheidbarkeit eine Reduktion des Postschen Korrespondenzproblems:

Definition 14.1 Eine Instanz des Postschen Korespondenzproblems (PKP) ist gegeben durch zwei Folgen von Wörtern:

$A = w_1, \ldots, w_k$ und $B = x_1, \ldots, x_k$

Eine Lösung des Problems ist eine Indexfolge $i_1, \ldots, i_m$ mit $m \geq 1$ und $i, j \in \{1, \ldots, k\}$, so daß gilt:

$w_{i_1} \ldots w_{i_m} = x_{i_1} \ldots x_{i_m}$,

also zum Beispiel $A = ab, a$ und $B = a, ba$. Eine Lösung ist $1, 2$, da $aba = aba$.

Satz 14.2 Das PKP ist nicht entscheidbar.

Beweis: siehe BuK

□

Lemma 14.3 Es ist nicht entscheidbar, ob für gegebene kontextfreie Grammatiken G_1, G_2 gilt:

$L(G_1) \cap L(G_2) \neq \emptyset$.

Beweis: Wir reduzieren das PKP auf dieses Problem, d.h. zu jeder Instanz PKP(A, B) des PKP kann man effektiv kontextfreie Grammatiken G_A und G_B konstruieren mit:

PKP(A, B) hat eine Lösung gdw $L(G_A) \cap L(G_B) \neq \emptyset$.

Wäre "$L(G_A) \cap L(G_B) \neq \emptyset$" entscheidbar, dann auch das PKP. Da das PKP unentscheidbar ist, folgt die Aussage des Lemmas.

Sei also $A = w_1, \ldots, w_k$ und $B = x_1, \ldots, x_k$.

Sei $G_A := (N_A, \Sigma_A, P_A, S_A)$ mit

- $N_A := \{S_A\}$,
- $\Sigma_A := \Sigma \cup \underbrace{\{1, \ldots, k\}}_{\text{Indizes}}$
- $P_A := \{S \to w_i S_A i, S_A \to w_i i \mid 1 \leq i \leq k\}$

G_B wird entsprechend definiert.

Es gilt:

$L(G_A) = \{w_{i_1} \ldots w_{i_m} i_m \ldots i_1 \mid m \geq 1, i, j \in \{1, \ldots, k\}\}$

$L(G_B) = \{x_{i_1} \ldots x_{i_m} i_m \ldots i_1 \mid m \geq 1, i, j \in \{1, \ldots, k\}\}$

Damit gilt:

$L(G_A) \cap L(G_B) \neq \emptyset$

gdw $\exists m \geq 1, \exists i_1, \ldots, i_m \in \{1, \ldots, k\}$ mit
$w_{i_1} \ldots w_{i_m} i_m \ldots i_1 = x_{i_1} \ldots x_{i_m} i_m \ldots i_1$

gdw PKP(A, B) hat Lösung $(i_1 \ldots i_m)$

□

Da jede kontextfreie Sprache auch kontextsensitiv ist, und $\mathcal{L}_1$ unter Durchschnitt abgeschlossen ist, folgt:

Satz 14.4 Für $\mathcal{L}_1$ ist das Leerheitsproblem und das Äquivalenzproblem unentscheidbar.

Beachte: Das Leerheitsproblem ist ein Spezialfall des Äquivalenzproblems:

$L(G) = \emptyset$ gdw $L(G) = L(\underbrace{G_\emptyset}_{L(G_\emptyset)=\emptyset})$

Bei kontextfreien Sprachen ist das Leerheitsproblem entscheidbar. Aber kontextfreie Sprachen sind nicht unter $\cap$ abgeschlossen.

Satz 14.5 Für $\mathcal{L}_2$ ist das Äquivalenzproblem unentscheidbar.

Beweis: Die Sprachen $L(G_A)$ und $L(G_B)$ sind sogar deterministisch kontextfrei.

Was macht ein deterministischer Kellerautomat für $L(G_A)$?

$w_{i1} \ldots w_{im} i_m \ldots . i_1$

Liest $w_{i1} \ldots w_{im}$ in den Keller. Wenn der erste Index (natürliche Zahl) auf dem Eingabeband gelesen wird, so testet (mit ε-Übergängen) der Automat, ob $\overleftarrow{w_{im}}$ im

Keller steht etc . . . Man kann sich leicht überlegen, daß dies wirklich deterministisch geht.

Da die deterministisch kontextfreien Sprachen unter Komplement abgeschlossen sind, kann man effektiv eine kontextfreie Grammatik $\overline{G_A}$ konstruieren mit $L(\overline{G_A}) = \overline{L(G_A)}$.

Nun gilt: $L(G_A) \cap L(G_B) \neq \emptyset$ gdw
$L(G_B) \subseteq L(\overline{G_A})$ gdw
$\underbrace{L(G_B) \cup L(\overline{G_A})}_{\text{kf wegen Abschluß unter } \cup} = \underbrace{L(\overline{G_A})}_{\text{kf}}$

ist ein Äquivalenzproblem.

Wäre das Äquivalenzproblem entscheidbar, so auch die Frage "$L(G_A) \cap L(G_B) \neq \emptyset$?". Dies ist jedoch nicht entscheidbar.

15 Das Äquivalenzproblem für reguläre Sprachen

Wissen: Das Leerheitsproblem und das Wortproblem sind polynomiell entscheidbar. Das Äquivalenzproblem ist entscheidbar durch Reduktion auf das Leerheitsproblem.

$$L_1 = L_2 \text{ gdw } (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

Problem: Bei nichtdeterministischen Automaten kann der Potenzmengenautomat exponentiell groß sein. Wir werden zeigen:

1. Das Äquivalenzproblem kann mit polynomiellem Platzaufwand gelöst werden, d.h. das Problem ist in der Komplexitätsklasse PSPACE.

2. Es geht nicht besser. Das Problem ist PSPACE-hart, d.h. jedes Problem in PSPACE kann mit polynomiellem Zeitaufwand auf das Äquivalenzproblem reduziert werden.

   Da NP $\subseteq$ PSPACE, heißt das, daß das Äquivalenzproblem für reguläre Sprachen wahrscheinlich nicht in polynomieller Zeit lösbar ist (nur wenn P=NP ist, was unwahrscheinlich ist).

Satz 15.1 Das Äquivalenzproblem für reguläre Sprachen ist in PSPACE.

Beweis: Da die Transformation von regulären Ausdrücken in NEAs polynomiell (Zeit und Platz) ist, können wir ohne Einschränkung annehmen, daß die Sprachen durch NEAs

$$A_i = (Q_i, \Sigma, q_{0i}, \Delta, F_i) \quad (i = 1, 2)$$

gegeben sind.

Offenbar ist "$L(A_1) = L(A_2)$?" mit polynomiellem Platzaufwand entscheidbar gdw die Frage "$L(A_1) \neq L(A_2)$?" mit polynomiellem Platzaufwand entscheidbar ist.

Satz von Savitch Komplexitätstheorie

PSPACE = NPSPACE

PSPACE polynomiell-platzbeschränkte DTM

NPSPACE polynomiell-platzbeschränkte NTM

Es genügt daher, ein nichtdeterministisches Verfahren zur Entscheidung von "$L(A_1) \neq L(A_2)$?" anzugeben.

Idee: Es sei A'_i der durch Potenzmengenkonstruktion erhaltene DEA zu A_i. Man kann A'_i zwar nicht vollständig mit polynomiellem Platz konstruieren, aber jeden einzelnen Zustand $P \subseteq Q_i$ kann man mit linearem Aufwand speichern.

Jeder Zustandsübergang von A'_i:

$\delta_{A'_i}(P, a) := \{q \in Q_i \mid \exists p \in P : (p, a, q) \in \Delta_i\}$

benötigt polynomiellen Platz. Das nichtdeterministische Verfahren rät jetzt sukzessive Buchstaben $a_1, a_2, \ldots \in \Sigma$ und berechnet in jedem Schritt $\delta_{A_i}(\{q_{0i}\}, a_1 \ldots a_n) \subseteq Q_i$. Nur diese beiden aktuellen Zustände von A'_1 und A'_2 müssen im Speicher gehalten werden.

Das Verfahren akzeptiert (d.h. antwortet "$L(A_1) \neq L(A_2)$"), falls einer der beiden Zustände ein Endzustand ist und der andere nicht.

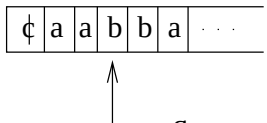
Noch zu zeigen: Das Problem ist PSPACE-hart, d.h. die Berechnung jeder polynomiell platzbeschränkten TM läßt sich durch das Äquivalenzproblem mit polynomiellem Aufwand simulieren.

Lemma 15.2 Es sei $A = (Q, \Sigma, \Gamma, q_0, \delta, q_{acc}, q_{rej})$ eine deterministische Turingmaschine, deren Platzbedarf durch ein Polynom $p(n)$ beschränkt ist. Dann gibt es ein Polynom q so, daß man für jedes $x \in \Sigma^*$ einen regulären Ausdruck $r(x)$ in Zeit $q(|x|)$ konstruieren kann, mit $x \notin L(A)$ gdw $r(x)$ ist äquivalent zu Σ^*.

Beweis: Konfigurationsfolgen von A kann man darstellen in der Form: $\#w_1\#w_2\# \ldots \#w_n\#$, wobei w_i Länge p(|x|) hat.

denn: A benötigt ja zum Akzeptieren von x nur $p(|x|)$ viel Platz. Um den jeweiligen Zustand zu kodieren, verwenden wir das Alphabet $\Delta_\# := \Delta \cup \{\#\},$ wobei $\Delta := \Gamma \cup Q \times \Gamma$.

z.B.



wird beschrieben durch $\#aa[q, b]ba\#$

Idee: $r(x)$ beschreibt diejenigen Wörter, die nicht Konfigurationsfolgen einer akzeptierenden Berechnung für x darstellen.

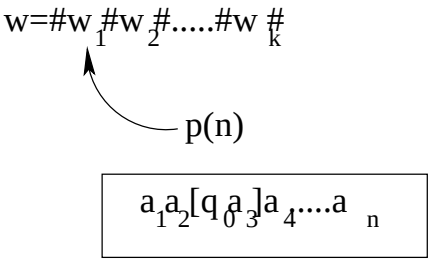
Insbesondere beschreibt r(x) genau dann alle Wörter, wenn es keine solche Folge gibt, d.h. $x \notin L(A)$ ist.

Es sei nun $|x| = n$ und $x = a_1 \ldots a_n$. Ein Wort $w \in \Delta_\#^*$ heißt $p(n)$-Konfigurationswort, wenn es ein $k \geq 1$ gibt mit $w = \#w_1\# \ldots .w_k\#$, wobei $|w_i| = p(n)$ und $w_i \in \Gamma^*(Q \times \Gamma)Q^*$.

Man überlegt sich leicht, daß $w \in \Delta_\#^*$ genau dann keine akzeptierende Berechnung von A für x beschreibt, wenn einer der folgenden Fälle eintrifft:

1. w ist kein Konfigurationswort, d.h. es beschreibt gar keine Berechnung, insbesondere keine akzeptierende.
2. Die Anfangskonfiguration ist falsch, d.h. w beginnt nicht mit $\#w_1\#$ für $w_1 = [q_0, a_1]a_2 \ldots .a_n(\sqcup)^{p(n)-n}$.
3. Es wird nicht q_{acc} erreicht, d.h. w enthält kein Symbol aus $q_{acc} \times \Gamma$
4. w enthält $\#w_i\#w_{i+1}\#$, wobei w_{i+1} nicht die Folgekonfiguration zu w_i ist.

Wir schreiben $r(x)$ als $A + B + C + D$, wobei A,B,C,D genau die Fälle 1.-4. abdecken.



$r(x) = A + B + C + D$

Abkürzung: Für $S = \{d_1, \ldots d_m\} \subseteq \Delta_\#$ schreiben wir S anstelle von $d_1 + \ldots + d_m$. S^i steht für $\underbrace{S \cdot \ldots \cdot S}_{i\text{-mal}}$.

Der reguläre Ausdruck A: $A =$

$\Delta^* +$	kein #
$\Delta^* \# \Delta^*$	nur ein #
$\Delta\Delta^*_\# + \Delta^*_\#\Delta +$	beginnt oder endet nicht mit #
$\Delta^*_\# \# \Gamma^* \# \Delta^*_\# +$	kein Zustand zwischen 2 #
$\Delta^*_\# \# \Delta^*(Q \times \Gamma)\Delta^*(Q \times \Gamma)\Delta^* \# \Delta^*_\# +$	mindestens 2 Zustände zwischen 2 #
$\Delta^*_\# \# \Delta^{p(n)+1}\Delta^* \# \Delta^*_\# +$	mehr als p(n) Symbole zwischen 2 #
$A_0 + A_1 + \ldots + A_{p(n)-1}$ wobei $A_i = \Delta^*_\# \# \Delta^i \# \Delta^*_\#$.	weniger als p(n) Symbole zwischen zwei #

Man sieht leicht:

- A beschreibt genau die Wörter über $\Delta^*_\#$, die nicht p(n)-Konfigurationswörter sind.
- Die Länge von A ist in $O(p(n)^2)$.

Wenn wir die Ausdrücke für 2.-4. schreiben, können wir ohne Einschränkung davon ausgehen, daß die zu beschreibenden Wörter p(n)-Konfigurationswörter sind, da sie sonst schon in A sind.

Der reguläre Ausdruck B: wird geschrieben als $B = B_1 + \ldots + B_{p(n)}$, wobei

$$\begin{aligned} B_1 &= \#(\Delta \setminus \{[q_0, a_1]\})\Delta^{p(n)-1}\#\Delta^*_\# \\ B_i &= \#\Delta^{i-1}(\Delta \setminus \{a_i\})\Delta^{p(n)-i}\#\Delta^*_\# \quad (1 < i \leq n) \\ B_i &= \#\Delta^{i-1}(\Delta \setminus \{\sqcup\})\Delta^{p(n)-i}\#\Delta^*_\# \quad (n < i \leq p(n)) \end{aligned}$$

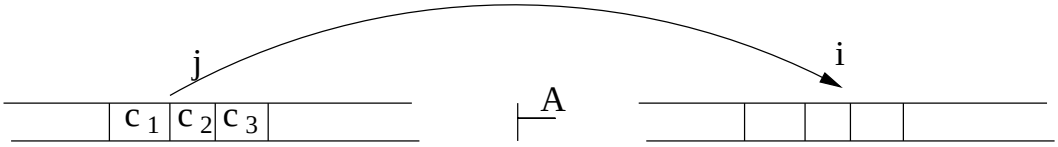
Größe: $O(p(n)^2)$.

Der reguläre Ausdruck C:

$$(\Delta_\# \setminus (\{q_{acc}\} \times \Gamma))^*$$

Größe: konstant

Der reguläre Ausdruck D: Wir betrachten zunächst den Fall, bei dem die Konfigurationen zusammenpassen, d.h. $\ldots \#w_i\#w_{i+1}\# \ldots$ Für drei aufeinanderfolgende Symbole c_1, c_2, c_3 in $\#w_i$ ist dann das Symbol,das $p(n)+1$ Symbole rechts von c_2 steht,



eindeutig durch c_1, c_2, c_3 bestimmt:

- Sind $c_1, c_2, c_3 \in \Gamma$ (kein Zustand), so ist dieses Symbol gleich c_2.
- Sind $c_1, c_2 \in \Gamma$ und ist $c_3 = [q, a]$ und ist $\delta(q, a) = (p, b, L)$, so ist dieses Symbol gleich $[p, c_2]$.
- etc ...

Es bezeichne $f(c_1, c_2, c_3)$ jeweils dieses Symbol (das man aus der Übergangsfunktion der Turingmaschine ablesen kann). Damit ist D die Summe der Ausdrücke

$D_{c_1,c_2,c_3} := \Delta_\#^* c_1 c_2 c_3 \Delta_\#^{p(n)-1} (\Delta_\# \setminus \{f(c_1, c_2, c_3)\}) \Delta_\#^*$
für $c_1, c_2, c_3 \in \Delta_\#$

Größe: $O(p(n))$

Insgesamt: $r(x)$ kann mit Zeitaufwand $O(p(n)^2)$ berechnet werden. d.h. $p' := p^2$

□

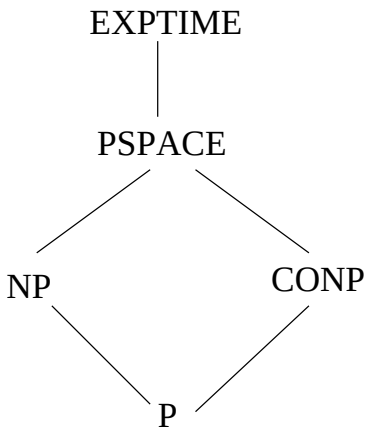
Satz 15.3 Das Äquivalenzproblem für reguläre Ausdrücke ist PSPACE-vollständig.

Beweis: Mit Satz 15.1 ist das Problem in PSPACE. Es genügt zu zeigen, daß das Problem PSPACE-hart ist, d.h. jedes Problem in PSPACE kann in polynomieller Zeit auf das Äquivalenzproblem reduziert werden.

Es sei ein Problem in PSPACE gegeben, d.h. eine Sprache L, die durch eine deterministische Turingmaschine A mit Platzbeschränkung $p(n)$ für ein Polynom p akzeptiert wird.

Mit Lemma 15.2 gibt es ein Polynom p', so daß wir zu jedem Wort x mit Aufwand $O(p'(|x|))$ einen regulären Ausdruck $r(x)$ konstruieren können mit:

$$\underbrace{x \notin L = L(A)}_{\text{bel. PSPACE Problem}} \quad \Leftrightarrow \underbrace{r(x) \text{ beschreibt die Menge aller Wörter}}_{\text{Äquivalenzproblem für reguläre Ausdrücke}}$$



□

Index