Automatentheorie

22.10.09
Robert B.

user: theorie
pwd: spass

DOL → Deterministisch und Kontextfrei

Grundlegende Definitionen

Def: Ein Alphabet ist eine endliche Menge von Symbolen, Buchstaben oder Zeichen.
Bezeichnung meist: Σ, Δ, Γ
Ein Wort über dem Alphabet Σ ist eine endliche Folge von Buchstaben aus Σ. Ein Wort wird auch String oder Zeichenkette genannt.
Bezeichnungen: u, v, w, x, y, z ← kleine lat. Buchstabe

Schreibweise: $w = a_1 a_2 \dots a_n$, wobei $a_1 \dots a_n$ die Reihenfolge der Buchstaben im Wort angibt

Die Länge. $|w|$ Die Länge eines Wortes w ist die Anzahl der Buchstaben in diesem Wort.
|| ← Länge von

Sei $w = \underbrace{aaa\dots a}_{n\text{-mal}}$, dann kürzen wir das durch a^n ab

Sei Σ ein Alphabet, dann bezeichnet Σ^* die Menge aller Wörter über Σ inkl. dem leeren Wort ε (epsilon). $\Sigma^+ =_{def} \Sigma^* \setminus \{\varepsilon\}$

* kleine Sigma

Sei $w \in \Sigma^*$ und $a \in \Sigma$, dann bezeichnet $|w|_a$ die Anzahl des Buchstabens a in w

Sind $u, v, w \in \Sigma^*$ so ist uov oder uv die Konkatenation von u und v

Also gilt: $u \in \Sigma^*$ $u\varepsilon = u = \varepsilon u$ (leeres Wort/neutrale Element (0))

Chomsky Hierarchie

Def: Eine (Chomsky) Grammatik G ist ein 4-Tupel

$G = (\Sigma, N, P, S)$ wobei

- Σ das Alphabet ist (Terminale)
- N ist die endliche Menge von Nichtterminalen, wobei $N \cap \Sigma = \emptyset$ (disjunkt) (wenn der Ableitungsprozess zu Ende ist.)
- P ist die Menge der Produktionen. P ist eine endliche Teilmenge von $(N \cup \Sigma)^* \times (N \cup \Sigma)^*$ (damit kein ε ou steht; kreuz Produkt)
  Meist schreiben wir statt $(u, v) \in P$ $u \to v$
- $S \in N$ ist das Startsymbol

Def: Eine Sprache L über Σ ist eine Teilmenge von Σ^*, $\subseteq$ = Teilmenge
d.h. $L \subseteq \Sigma^*$

Def: Sei $G = (\Sigma, N, P, S)$ eine Grammatik, dann gilt:

1. Sind $u_1, u_2, v, w \in (\Sigma \cup N)^*$ und $u \to v \in P$, dann heißt $u_1 v u_2$ von G aus $u_1 u u_2$ in einem Schritt erzeugt.
   (Schreibweise: $u_1 u u_2 \vdash_G u_1 v u_2$)
2. Das Wort w heißt aus v von G in t Schritten erzeugt, wenn es eine Folge von Worten $w_0, \dots, w_t$ gibt mit $v = w_0 \vdash w_1 \vdash w_2 \vdash \dots \vdash w_t = w$
   (Schreibweise: $v \vdash^t w$)
3. Das Wort w aus v von G erzeugt falls es ein $t \in N$ gibt mit $v \vdash^t w$
   (Schreibweise $v \vdash^* w$)
4. Die Sprache $L(G) = \text{def} \{ w \in \Sigma^* \mid S \vdash^* w \}$ heißt die von G erzeugte Sprache.

Def: Sei $G = (\Sigma, N, P, S)$, dann gilt

1) Jede solche Grammatik ist vom Typ 0

2) G heißt Grammatik vom Typ 1 oder kontext-sensitive Grammatik, falls jede Produktion von G der Form $u_1 A u_2 \to u_1 w u_2$ entspricht, wobei $A \in N$, $u_1, w, u_2 \in (\Sigma \cup N)^*$ und $|w| \geq 1$

   nichtterminal, kontext: $u_1 A u_2 \to u_1 w u_2$; keine Regel zum „schrumpfen" verkürzen; kontextfrei: $A \to$

3) G heißt Grammatik vom Typ 2 oder kontextfreie Grammatik falls jede Produktion der Form $A \to w$ mit $A \in N$, $w \in (\Sigma \cup N)^*$ und $|w| \geq 1$

4) G heißt Grammatik vom Typ 3, reguläre Grammatik oder rechtsreguläre Grammatik, falls jede Produktion von G der Form $A \to aB$ oder $A \to a$ mit $A, B \in N$ und $a \in \Sigma$ entspricht.

Def: Sei $i \in \{0,1,2,3\}$ (Grammatiktyp), dann heißt eine Sprache L vom Typ i, falls es eine Grammatik G gibt mit $L = L(G)$

Offensichtlich ist für $i \in \{1,2,3\}$ jede Grammatik vom Typ $i+1$ auch eine Grammatik vom Typ i.

$L_i =_{def} \{L \mid L$ ist eine Sprache vom Typ $i\}$

Eigenschaft: $L_3 \subseteq L_2 \subseteq L_1 \subseteq L_0$, diese Folge heißt Chomsky Hierachi

Def.: Zwei Grammatiken G und G' heißen äquivalent wenn $L(G) = L(G')$ gilt.

Bsp: $G_1 = (\{a,b\}, \{S\}, \{S \to aSa, S \to bSb, S \to a, S \to b, S \to \varepsilon\}, S)$

$L(G_1)$ = „Menge der Palindrome über $\{a,b\}^*$"

Peano-Axiome
22. Okt. 2008
Übung

① $P \to 0,\ P \to 1$
$P \to 1P1,\ P \to 0P0$

Beweis $P \to 0, P \to 1$ ist ungerade, $P \to 1P1 \mid 0P0$ auch weil kann immer über 2 Zeichen erweitert werden.

$P \to 0, P \to 1$ abspricht dem (IA)

Wir können beliebig lange Strings ungerader Länge erzeugen.

Die Regeln $P \to 0P0$ (I) und $P \to 1P1$ (II) fügen immer zwei Oen oder 1en ein und zwar so, dass an jedem Zeichen das gleiche Zeichen an der gespiegelten Position entsteht.

$$P \overset{I}{\Rightarrow} 0P0 \overset{II}{\Rightarrow} 01P10 \overset{I}{\Rightarrow} 010P010$$
$$\overset{II}{\Rightarrow} 011P110$$

D.h. wir können so Palindrome erzeugen.

⇒ Wir können Palindrome über dem Alphabet $\{0,1\}$ erzeugen

② Mit den Regeln $I \to x,\ I \to y,\ I \to I0,\ I \to I1$ können Wörter der Form $x, y, x0, y0, x1, y1, x10, 1, \ldots$ erzeugt werden.

Allgemein x, y oder x Bitstring $010\ldots1$ oder y Bitstring $1010\ldots1$

(Binär Zahl oder Index)

"Array" Variable mit Index

Mit den Regeln $E \to I$
$E \to E+E \mid E*E \mid (E)$

Können Worte der Form $E \Rightarrow E+E \Rightarrow E+E+E$
$\hookrightarrow E*E \hookrightarrow E+E*E$
$\hookrightarrow (E) \hookrightarrow E+E+E$
$\hookrightarrow I$

Beobachtung: Für gleiche Worte kann es verschiedene Ableitungswege geben.

① $E \Rightarrow E*E \Rightarrow E*E+E$
② $E \Rightarrow E+E \Rightarrow E*E+E$

1/

Beobachtung: Es gibt Mehrdeutigkeiten
(evtl. Ärger mit Compiler)

D.h. wir können arithmetische Ausdrücke mit wiederverwendbaren Variablen (x und y) erzeugen.

Aufgabe 3

Beispiel $G_1 = (\{a,b\}, \{S\}, \{S \to aSa, S \to bSb, S \to a, S \to b, S \to \varepsilon\}, S)$
↳ dazu sagen Typ 0

erzeugt die Menge der Palindrome über $\{a,b\}$

$S \vdash aSa \vdash abSba \vdash abba$

Also $L(G_1) = \{ w \in \{a,b\}^* \mid a_1 a_2 \cdots a_n = w = w^R = a_n a_{n-1} \cdots a_1 \}$

Beispiel $G_2 = (\{a,b\}, \{S\}, \{S \to aSb, S \to \varepsilon\}, S)$

$L(G_2) = \{ w \in \{a,b\}^* \mid w = a^n b^n, n \geq 0 \}$

$(S \vdash aSb \vdash a a^n S b^n b \vdash a a b b)$

Beispiel $G_3 = (\{(,), \times, +, *\}, \{E, T, F\}, \{E \to T, E \to E+T, T \to F, T \to T \times F, F \to x, F \to (E)\}, E)$

$L(G_3)$ = Menge der arithmetischen Ausdrücke ↗ syntaktisch korrekt (punkt vor Strich)

Beispiel $G_4 = (\{a,b,c\}, \{S,B\}, \{S \to aSBc, S \to abc, cB \to Bc, bB \to bb\}, S)$

$L(G_4) = \{ w \in \{a,b,c\}^* \mid$ (Übung$\}$

Wortproblem

Beobachtung: Sei G eine beliebige Grammatik vom Typ 1, 2, 3. Anhand der Definition sieht man leicht ein, dass für jede Produktion $u \to v$ gilt $|u| \leq |v|$ ↗ u wird nicht kürzer

Das heißt Zwischenergebnisse beim Ableitungsprozess schrumpfen nicht

Wir definieren:

PROBLEM : MEMBER
EINGABE : Grammatik G und ein Wort w
FRAGE : Gilt $w \in L(G)$?

Bsp: Entspricht dem Text ob ein XML-Dokument valide ist.
DTD wäre die Grammatik
XML-Doc das Wort

Bsp: Überprüfung ob ein Programm syntaktisch korrekt ist

Satz: Es gibt einen Algorithmus, der das Wortproblem für Typ 1 Grammatiken löst.

Beweis: Sei $n, m \in \mathbb{N}$, dann definieren wir die Mengen T^n_m wie folgt: $T^n_m = \{ w \in (N \cup \Sigma)^* \mid |w| \leq n$ und w läßt sich aus der gegebenen Grammatik in max m Schritten erzeugen$\}$

wir definieren die Menge der Erzeugnisse der Länge max n, die durch einen Ableitungsschritt aus einer Menge von Zwischenergebnissen erzeugt werden können.

$$Abl_n(x) = def \; x \cup \{ w \in (\Sigma \cup N)^* \mid |w| \leq n \text{ und } w' \vdash w \text{ für ein } w' \in x \}$$

Die Mengen T^n_i können wie folgt berechnet werden:

$$T^n_0 = \{S\} \quad \text{„Startsymbol} \triangleq \text{Start"}$$
$$T^n_{m+1} = Abl_n(T^n_m)$$

↓ Induktion

Dies ist für Typ 0-Sprachen nicht notwendigerweise Richtig da hier Ableitungsregeln „verkürzend" wirken können.

Klar: Es gibt nur endlich viele Wörter der Länge $\leq n$ aber $(N \cup \Sigma)^*$ und daher muß es ein n geben mit $T^n_m = T^{n+1}_{m+1} = T^{n+2}_{m+2} = \ldots$ „Fixpunkt"

Dh. genau dann wenn $w \in L(G)$, $|w| = n$ so muß w in $\bigcup_{m \geq 0} T^n_m$ liegen. Damit ergibt sich der folgende Algorithmus:

$\bigcup_{m \geq 0} T^n_m$ ← Vereinigung aller Mengen T^n

```
bool member (G, w){
    T = {S}
    do {
        T_old = T
        T = Abl_n (T)
    } while (T ≠ T_old) & & !(w ∈ T))
    if w ∈ T
        return true
    else
        return false
}
```

Bem: Dieser Algorithmus hat ein schlechtes Laufzeitverhalten. (exponentielle Laufzeit) Es gibt vermutlich keinen besseren für Typ 1 Grammatiken. Das Wortproblem ist PSPACE-vollständig

Bsp: Welche Worte der Länge ≤ 4 ex. in $L(G_4)$

$T_0^4 = \{S\}$

$T_1^4 = \{S, aSBc, abc\}$

$T_2^4 = \{S, aSBc, abc\} = T_3^4 = T_4^4 \ldots$

Also $L(G_4) \cap \{w \in \{a,b,c\}^* \mid |w| \leq 4\} = \{abc\}$

$\rightarrow abc \in L(G_4)$

2. Reguläre Sprachen

Ziel: Typ 3 Sprachen genauer untersuchen

Dazu werden wir endliche Automaten einführen und reguläre Ausdrücke.

Def: Ein (deterministischer) endlicher Automat M (kurz DEA, DFA) ist ein 5 Tupel $M = (Z, \Sigma, \delta, z_0, E)$

wobei

- Z - endliche Menge der Zustände
- Σ ist das Alphabet $Z \cap \Sigma = \emptyset$ (Zustand kann nie ein Endlose sein)
- δ ist die Überführungsfkt mit $\delta: Z \times \Sigma \to Z$
- z_0 ist der Startzustand wobei $z_0 \in Z$
- E Menge der Endzustände, wobei $E \subseteq Z$

Die erweiterte Überführungsfunktion $\hat{\delta}: Z \times \Sigma \to Z$ wird induktiv wie folgt definiert:

$$\hat{\delta}(z, \varepsilon) = \text{def } z \quad \text{(wenn nix gelesen wird soll z)}$$

$$\hat{\delta}(z, a\omega) = \hat{\delta}(\delta(z,a), \omega) \text{ wobei } a \in \Sigma, \omega \in \Sigma^*, z \in Z$$

} Rekursion

Die von M akzeptierte Sprache ist

$$L(M) = \text{def } \{\omega \in \Sigma^* \mid \hat{\delta}(z_0, \omega) \in E\}$$

Test

Test

Automatentheorie Übung

(Bei L-Systemen wird alles "immer" ersetzt Bei Chomsky Hierachien hört es irgendwann auf.

$L_1 \cup L_2$ L_1 vereinigt L_2

$L_1 \cap L_2$ L_1 schneidet L_2

Def: $\mathcal{C}$ ist abgeschlossen unter Vereinigung wenn für alle $L_1, L_2 \in \mathcal{C}$ auch $L_1 \cup L_2 \in \mathcal{C}$ gilt.

Zeige für den Spezialfall „kontextfrei" die Abgeschlossenheit.

Wir haben eine kontextfreie Sprache L gegeben.

Was wissen wir dann?

- Es gibt zu L eine Grammatik $G = \{\Sigma, N, P, S\}$ mit $L = L(G)$
- L ist eine Menge von Worten
- Die Produktionen sind der Form $A \mapsto \omega$, $|\omega| \geq 1$ $\omega \in (N \cup \Sigma)^*$
- Jede kontextfreie Sprache ist auch kontextsensitiv
- Wenn $w \in L$, dann gibt es eine Kette $S \vdash \omega_1 \vdash \omega_2 \vdash \dots \vdash w$ → w kann über die Folge von Ersetzungen erzeugt werden (Bei S stabil)

Plan: Baue aus den Grammatiken für L_1 und L_2 eine neue für $L_1 \cup L_2$

- Neues Startsymbol S' und Regeln $S' \vdash S_1$, $S' \vdash S_2$

Aufgabe 2: Seien L_1 und L_2 kontextfreie Sprachen und $G_1 = (\Sigma, N_1, P_1, S_1)$ bzw. $G_2 = (\Sigma, N_2, P_2, S_2)$ kontextfreie Grammatiken mit $L_1 = L(G_1)$ bzw. $L_2 = L(G_2)$

o.B.d.A kann wir davon ausgehen, dass $N_1 \cap N_2 = \emptyset$ da Nichtterminalsymbole umbenannt werden können.

Sei $G = (\Sigma, N, P, S)$ wobei $S \notin N_1$ und $S \notin N_2$

$N = \{S\} \cup N_1 \cup N_2$ $P = P_1 \cup P_2 \cup \{S \vdash S_1, S \vdash S_2\}$

G ist Kontextfrei und $L(G) = L_1 \cup L_2$, d.h. $L_1 \cup L_2$ ist Kontextfrei

⚠ Beweistechnik ist nicht auf reguläre Sprachen anwendbar ⚠

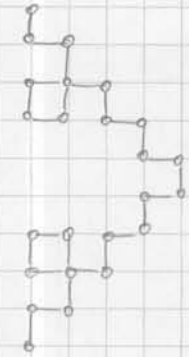
1. Aufgabe

→ Startrichtung

n = 0 $\hat{=}$ -F

n = 1 $\hat{=}$ -F+F-F-F+F

n = 2



Aufgabe 5

Überblick

$\omega_1 \in L$
$\omega_2 \notin L$

gesucht ist eine Funktion der Form bool member L_1 (word w)

Pseudocode

```
bool member_L1 (word w){

    drehe w um und speichere das Ergebnis in w';
    if (w == w')
        return true;
    else
        return false;
}
```

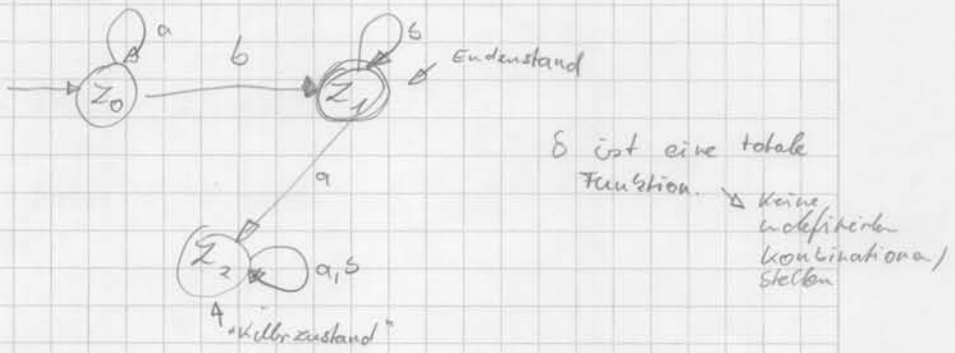
$2^{1024} \approx 10^{300}$

10^{80} → Atome im Universum

```
bool memberPrim (p)

    for(i=2 to p-1){

        if (p%i == 0)
            return false;

    }
    return true;
```

Def: $M_1 = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_1\})$

$\delta(z_0, a) = z_0, \quad (z_1, b) = z_1, \quad (z_2, a) = z_2$
$\delta(z_0, b) = z_1, \quad (z_1, a) = z_2, \quad (z_2, b) = z_2$



δ ist eine totale Funktion. ↯ Keine undefinierten Konfigurationen/Stellen

$L(M_1) = \{w \in \{a,b\}^* \mid w = a^n b^m, n \geq 0, m \geq 1\}$

$\hat{\delta}(z_0, abb) = \hat{\delta}(\delta(z_0, a), bb) = \hat{\delta}(z_0, bb) =$

$\hat{\delta}(\delta(z_0, b), b) = \hat{\delta}(z_1, b) = \hat{\delta}(\hat{\delta}(z_1, b), \varepsilon) = \hat{\delta}(z_1, \varepsilon) = z_1$

abb wird akzeptiert, damit gilt $abb \in L(M_1)$

Beispiel: $M_2 = (\{z_0\}, \{a, b\}, \delta, z_0, \{z_0\})$

$\delta(z_0, a) = z_0$
$\delta(z_0, b) = z_0$

$L(M_2) = \{a, b\}^*$

Nichtdeterministische Automaten

Wir verallgemeinern DEA wie folgt:

Def: Ein nichtdeterministischer Automat M (NEA) ist ein 5-Tupel $M = (Z, \Sigma, \delta, z_0, E)$, wobei

Z - die endliche Zustandsmenge

Σ - das Alphabet

δ - die Überführungsfunktion mit $\delta: Z \times \Sigma \to \mathcal{P}(Z)$

z_0 - Startzustand

$E \subseteq Z$ die Menge der Endzustände

Anschaulich:

- Lese ein Zeichen der Eingabe
- Für jede Möglichkeit der Übergangsfunktion wird eine Kopie des NEA erzeugt und jede dieser Alternativen wird parallel verfolgt
- Gibt es einen Pfad der zu einem Zustand $z \in E$ führt, dann akzeptiert der NEA

Wir definieren wie folgt:

$$\hat{\delta} : \mathcal{P}(Z) * \Sigma^* \to \mathcal{P}(Z) \text{ mit}$$

$$\hat{\delta}(Z', \epsilon) =_{def} Z' \text{ für alle } Z' \in \mathcal{P}(Z)$$

$$\hat{\delta}(Z', ax) =_{def} \bigcup_{z \in Z'} \hat{\delta}(\delta(z,a), x)$$

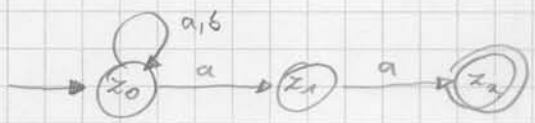
$\bigcup_{z \in Z'}$ → Vereinigung

Die von M akzeptierte Sprache ist dann:

$$L(M) =_{def} \{ w \in \Sigma^* \mid \hat{\delta}(\{z_0\}, w) \cap E \neq \emptyset \}$$

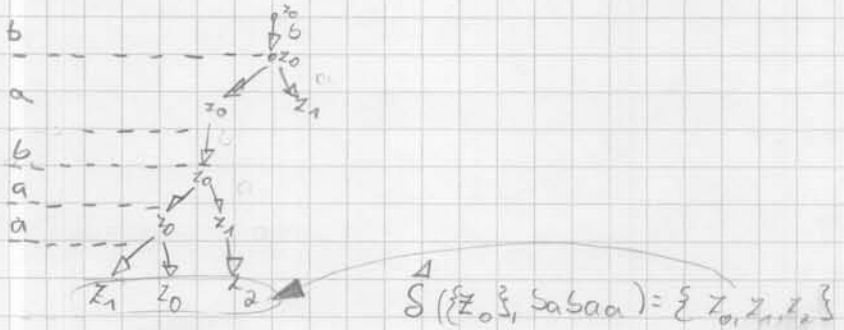
BSP: $M_3 = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, z_0, \{z_2\})$

δ:



$$L(M_3) = \{ w \in \{a,b\}^* \mid \text{wendet mit } aa \}$$

Beispiel $w = babaa$



Bem: Die DEAs sind ein Spezialfall der NEAS, d.h. jede Sprache die durch ein DEA akzeptiert wird, wird auch durch einen NEA akzeptiert

Satz: „Potenzmenge Konstruktion"

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

Sei $L \subset \Sigma^*$ mit $L = L(M)$ und $M = (Z, \Sigma, \delta, z_0, E)$

Gesucht ist ein DEA M' mit $L(M) = M' = L$

Idee: Ein Zustand von M' entspricht einer Menge von Zuständen von M.

Sei $M' = (Z', \Sigma, \delta', z_0', E')$

- $Z' = \mathcal{P}(Z)$
- $z_0' = \{z_0\}$
- $E = \{F \subseteq Z \mid F \cap E \neq \emptyset\}$
- $\delta'(F, a) = \bigcup_{z \in F_R} \delta(z, a) = \hat{\delta}(F, a)$ wo sei $F \in Z'$

($\delta'(F,a)$: Zustand von M'; F_R: Menge aller Zustände von M)

05.11.09
Automatentheorie
Übung

Aufgabe 1

$A = \{1,2,3\}$

$\mathcal{P}(A) = 2^A = \{B \mid B \subseteq A\} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$ (Die Menge aller Teilmengen von A)

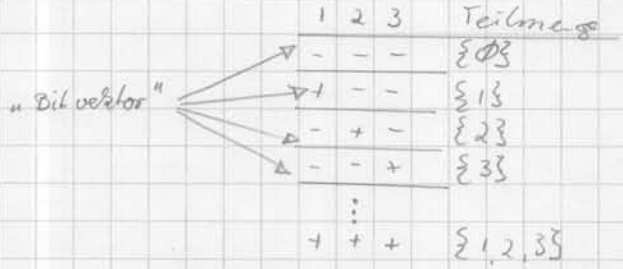
$\#\mathcal{P}(A) = 8$

$A' = \{a, b\}$

$\mathcal{P}(A') = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$

$\#\mathcal{P}(A') = 4$

Andere Darstellung



Formuliere die Frage um: Wieviele Bitstrings der Länge n gibt es?

Wir wissen es gibt 2^n Bitstrings der Länge n, d.h. wenn

$\#A = n$, dann $\#\mathcal{P}(A) = 2^n$ #

Induktion

(IA) $n = 0$: $A = \emptyset$ $\mathcal{P}(A) = \{\emptyset\}$
Stimmt, da $2^0 = 1 = \#\mathcal{P}(A)$

(IV) Sei A eine Menge mit n Elementen, dann gilt $\#\mathcal{P}(A) = 2^n$

(IS) $n \to n+1$

Sei $A = \{a_1, a_2, \ldots a_{n+1}\}$ und $A' = A \setminus \{a_{n+1}\}$

$\Rightarrow \# A' = n$

Dann gilt $\mathcal{P}(A) = \underbrace{\mathcal{P}(A')}_{2^n \text{ (nach IV)}} \cup \underbrace{\{C \cup \{a_{n+1}\} \mid C \in \mathcal{P}(A')\}}_{2^n}$

$\Rightarrow 2 \cdot 2^n = 2^{n+1}$, da $\mathcal{P}(A')$ und $\{C \cup \{a_{n+1}\} \mid C \in \mathcal{P}(A')\}$ disjunkt sind

②

Vermutung: $L(G_u) = \{a^u b^u c^u \mid u \geq 1\}$

Beobachtung: Jede Anwendung von $S \vdash aSBc$ bzw. $S \vdash aSBc$ erzeugt die gleiche Anzahl von a, b und c.

Die Regel $cB \vdash Bc$ ändert diese Anzahl nicht.

Da $bB \vdash bb$ muss für $w \in L(G_u)$ gelten:

$$|w|_a = |w|_b = |w|_c$$

Die Regel $cB \vdash Bc$ verschiebt die c nach rechts an das Ende des Wortes.

Alle B nur umgewandelt werden wenn sie in einem Block stehen

Die a's sind am Anfang des Wortes ganz links.

Aufgabe 3 $\Sigma = \{a, b, \ldots, z, A, B, \ldots, Z, 0, \ldots 9\}$ $\left(\begin{smallmatrix} A_{lex} \\ l_{ex} \end{smallmatrix}\right)$

$P = \{ S \vdash \Sigma \setminus \{0, \ldots 9\} B \mid \Sigma \setminus \{0, \ldots, 9\}$

$B \vdash \Sigma B$

$B \vdash \Sigma$

$\}$

12.11
Automatentheorie
Übung

①

Def: Es reicht eine Grammatik G für $L_1 \setminus \{\varepsilon\}$ zu finden

Idee: Erst mal das „Grobe"

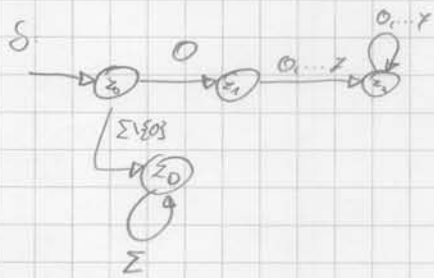
$\Sigma = \{a, b, c\}$, $G_1 = (\Sigma, N, P, S)$

$P = \{S \to aSc,\ S \to B,\ B \to bBc,\ B \to bc,\ S \to ac\}$

$N = \{S, B\}$

$L_2 =\quad S \to aSa$
$\quad\quad\to bSb$
$\quad\quad\to \#$

② $M_1 = (Z, \Sigma, \delta, z_0, E)$

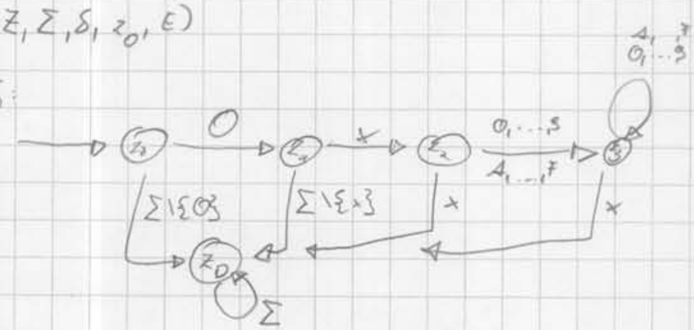


$Z = \{z_0, z_1, z_2, z_D\}$

$\Sigma = \{0, \dots, 7\}$

$E = \{z_2\}$

$M_2 = (Z, \Sigma, \delta, z_0, E)$



$Z = \{z_0, z_1, z_2, z_3, z_D\}$

$\Sigma = \{0, \dots, 9, A, \dots, F\}$

$E = \{z_3\}$

Aufgabe 3

Sei $M = (Z, \Sigma, \delta, z_0, E)$

Wir wissen: Ein Wort $w \in L(M)$ gdw. $\hat{\delta}(z_0, w) \in E$

Also $w = a_1 a_2 \dots a_n \in L(M)$ gdw. es gibt Zustände

$z_{i_1}, z_{i_2}, \dots z_{i_n} \in Z$ wobei $\delta(z_0, a_1) = z_{i_1}$, $\delta(z_{i_1}, a_2) = z_{i_2}$, ..., → zusätzliche Indices legt keine Reihenfolge fest.

$\delta(z_{i_{n-1}}, a_n) = z_{i_n} \in E$

gdw. es gibt einen Pfad von z_0 zu einem Zustand $z_{i_n} \in E$

Algorithmus

Automatentheorie Übung

$x = \{x \mid x \notin x\}$

Potenzmengenkonstruktionen 8

```
bool empty L (DEA M){

    markiere z0;
    do {
        for (z ∈ Z){

            if ((z ist markiert) && (Nachfolge von z ist unmarkiert)){
                markiere Nachfolger
            }
        }

    }while (es wurden Zustände neu markiert);

    if (ein Zustand aus E ist markiert){

        return false;
    }else {

        return true;
    }
}
```

Letztes Übungsblatt:

④

Idee:

$$A \vdash aBc \Rightarrow A \vdash R_a B R_b \,,\; R_a \vdash a \,,\; R_b \vdash c$$

Für jedes Symbol $a \in \Sigma$ wird ein neues Symbol R_a eingeführt.

Sei $\alpha \vdash \omega$ eine Regel, dann ersetze jedes $a \in \Sigma$ in ω durch R_a und füge die Regeln $\bigcup_{a \in \Sigma} \{R_a \vdash a\}$ zu den Produktionen hinzu.

⑤ $A \vdash D|EF \rightarrow A \mapsto D\,x$, $x \rightarrow EF$

$A \vdash D|E|FG \rightarrow A \rightarrow Dx$, $x \rightarrow E\,y$, $y \rightarrow FG$

Wir ersetzen jede Regel $A \vdash B_1 B_2 \dots B_k$ durch $A \vdash B_1 C_1$, $C_1 \vdash B_2 C_2$, $k \geq 3$ $\dots C_{k-1} \vdash B_{k-1} B_k$, wobei $C_1, C_2, \dots, C_{k-1}$ neue nicht-determinale sind #

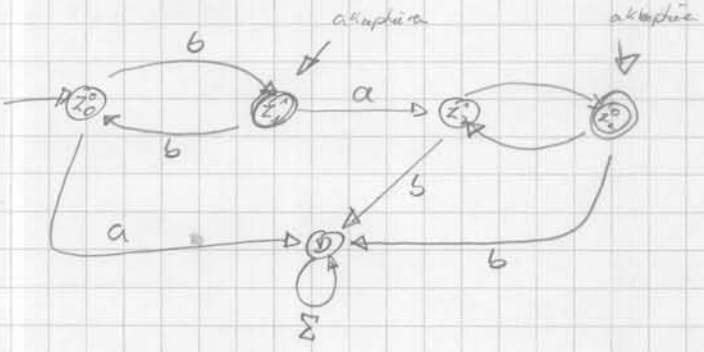
Idee: Bsp für L_3 finden

$\in L_3$	$\notin L_3$
baa	ab, aab
bbbaaaa	aaabbb
bbbbb aa	aa
b	

Wenn $w \in L_3$ ist mindestens ein b in w

a's dürfen nie vor b's kommen, denn sonst ist ab ein Teilstring

$L_3 = \{ w \in \{a,b\}^* \mid w = a^u b^w, u \text{ ungerade}, w \text{ ungerade} \}$



Z = Menge von Zustände
z = Zustände

Sei $M' = (Z', \Sigma, \delta', z_0', E')$

$Z' = \mathcal{P}(Z)$

$z_0' = \{z_0\}$

$E' = \{F \in \mathcal{P}(Z) \mid F \cap E \neq \emptyset\}$ ← enthält mind. 1 Endzustand

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

- $\omega \in L(M)$ genau dann wenn $\hat{\delta}(\{z_0\}, \omega) \cap E \neq \emptyset$

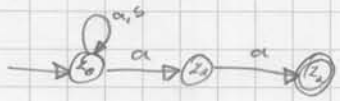
  gdw. es gibt eine Folge Teilmengen $z_1, z_2, \ldots, z_n$ von Z mit der Eigenschaft $\delta'(\{z_0\}, a_1) = z_1$, $\delta'(z_1, a_2) = z_2, \ldots, \delta'(z_{n-1}, a_n) = z_n$ wobei $\omega = a_1 \ldots a_n$ und $z_n \cap E \neq \emptyset$

  gdw. $\hat{\delta}'(\{z_0\}, \omega) \in E'$

  gdw. $\omega \in L(M')$

Bsp: Bestimme einen DEA M_3' mit $L(M_3') = L(M_3)$

M_3



$M_3' = (\mathcal{P}(\{z_0, z_1, z_2\}), \{a, b\}, \delta', \{\{z_0\}, \{\{z_1, z_0, z_2\}\})$

$\hat{\delta}(\{z_0\}, a) = \{z_0, z_1\}$

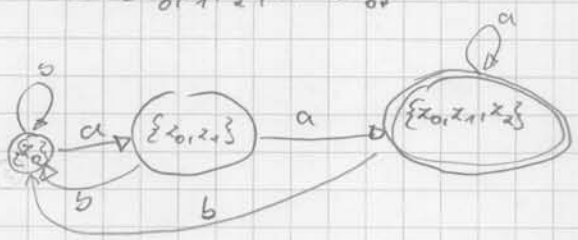
$\hat{\delta}(\{z_0\}, b) = \{z_0\}$

$\hat{\delta}(\{z_0, z_1\}, a) = \{z_0, z_1, z_2\}$

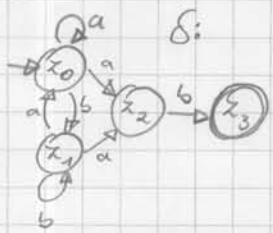
$\hat{\delta}(\{z_0, z_1\}, b) = \{z_0\}$

$\hat{\delta}(\{z_0, z_1, z_2\}, a) = \{z_0, z_1, z_2\}$

$\hat{\delta}(\{z_0, z_1, z_2\}, b) = \{z_0\}$



Beispiel: M_4



δ:

$M_4 = (\{z_0, z_1, z_2, z_3\}, \{a,b\}, \delta, z_0, \{z_3\})$

$L(M_4) = \{w \in \{a,b\}^* \mid w \text{ endet mit } ab\}$

$M_4' = (\mathcal{P}(\{z_0, z_1, z_2, z_3\}), \{a,b\}, \delta', \{z_0\}, \{\{z_1, z_3\}\})$

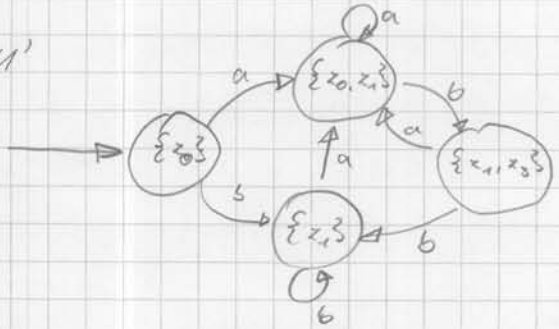
$\delta'(\{z_0\}, a) = \{z_0, z_2\}$ $\quad$ $\delta'(\{z_1\}, a) = \{z_0, z_2\}$

$\delta'(\{z_0\}, b) = \{z_1\}$ $\quad$ $\delta'(\{z_1\}, b) = \{z_1\}$

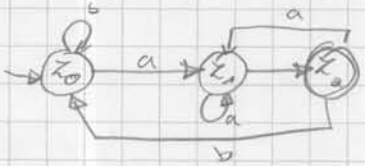
$\delta'(\{z_0, z_2\}, a) = \{z_0, z_2\}$ $\quad$ $\delta'(\{z_1, z_3\}, a) = \{z_0, z_2\}$

$\delta'(\{z_0, z_2\}, b) = \{z_1, z_3\}$ $\quad$ $\delta'(\{z_1, z_3\}, b) = \{z_1\}$

M'



M''



Klar $L(M_4'') = L(M_4')$

[kleinste Automat]

Def: $\Sigma^k =_{def} \{$Menge aller Worte über Σ der Länge $k\}$

$\Sigma^{\leq k} =_{def} \{$Menge aller Worte über Σ mit höchstens k Buchstaben$\}$

Frage: Wie groß können DEA's bei der Potenzmengenkonstruktion werden?

Satz: Die Sprache $L_k = \{w \in \{a,b\}^k \mid |w| \geq k$ und das k-letzte Zeichen ist $a\}$, $k \geq 1$

kann durch einen NEA mit $k+1$ Zuständen akzeptiert werden. Es gibt keinen DEA mit weniger als 2^k Zuständen der L_k akzeptiert.

Beweis: Der folgende NEA hat das gewünschte Verhalten

Behauptung: Es gibt keinen DEA der L_k akzeptiert und weniger als 2^k Zustände hat.

Annahme: Es gibt einen DEA mit weniger als 2^k Zuständen der L_k akzeptiert.
Dann müssen zwei verschiedene Worte $w, w' \in \{a,b\}^k$ ex, so dass $\hat{\delta}(z_0, w) = \hat{\delta}(z_0, w')$

(Bemerkung: $\#(\{a,b\}^k) = 2^k$, aber $\#\{z \mid \hat{\delta}(z_0, w) = z$ und $w \in \{a,b\}^k\} < 2^k \Rightarrow$ Schubfachschluß: verteile 2^k Strings auf $< 2^k$ Zustände)

Sei $y \in \{a,b\}^{k-1}$ beliebig und sei i die erste Position an der sich w und w' unterscheiden.

Klar $1 \leq i \leq k$

Dann gilt o.B.d.A. $wy = \underbrace{uav}_{=w} y$ und $w'y = \underbrace{ubv'}_{=w'} y$

wobei $|v| = |v'| = k - i$ und $|u| = i - 1$

Das heißt a (bzw b) in wy (bzw $w'y$) steht an der k-letzten Stelle.

D.h. $wy \in L_k$ und $w'y \notin L_k$

Es gilt: $\hat{\delta}(z_0, wy) = \hat{\delta}(\hat{\delta}(z_0, w), y)$

$= \hat{\delta}(\hat{\delta}(z_0, w'), y)$

$= \hat{\delta}(z_0, w'y)$

$\Rightarrow wy \in L_k$ gdw $w'y \in L_k$ ↯

$\Rightarrow$ Ein solcher Automat existiert nicht.

Beweis: Der folgende NEA hat das gewünschte Verhalten

Behauptung: Es gibt keinen DEA der L_k akzeptiert und weniger als 2^k Zustände hat.

Annahme: Es gibt einen DEA mit weniger als 2^k Zuständen der L_k akzeptiert.
Dann müssen zwei verschiedene Worte $w, w' \in \{a,b\}^k$ ex, so dass $\hat{\delta}(z_0, w) = \hat{\delta}(z_0, w')$

(Bemerkung: $\#(\{a,b\}^k) = 2^k$, aber $\#\{z \mid \hat{\delta}(z_0, w) = z$ und $w \in \{a,b\}^k\} < 2^k \Rightarrow$ Schubfachschluß: verteile 2^k Strings auf $< 2^k$ Zustände)

Sei $y \in \{a,b\}^{k-1}$ beliebig und sei i die erste Position an der sich w und w' unterscheiden.

Klar $1 \leq i \leq k$

Dann gilt o.B.d.A. $wy = \underbrace{uav}_{=w} y$ und $w'y = \underbrace{ubv'}_{=w'} y$

wobei $|v| = |v'| = k - i$ und $|u| = i - 1$

Das heißt a (bzw b) in wy (bzw $w'y$) steht an der k-letzten Stelle.

D.h. $wy \in L_k$ und $w'y \notin L_k$

Es gilt: $\hat{\delta}(z_0, wy) = \hat{\delta}(\hat{\delta}(z_0, w), y)$

$= \hat{\delta}(\hat{\delta}(z_0, w'), y)$

$= \hat{\delta}(z_0, w'y)$

$\Rightarrow wy \in L_k$ gdw $w'y \in L_k$ ↯

$\Rightarrow$ Ein solcher Automat existiert nicht.

Automatentheorie
Ü
13.11.09

zeige == Beweise

Aufgabe ①

$\mathcal{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$

$\mathcal{P}(\{x,y,z\}) = \{\emptyset, \{x\}, \{y\}, \{z\}, \{x,y\}, \{x,z\}, \{y,z\}, \{x,y,z\}\}$

$x = y$ ↳ beweise durch: $x \subseteq y$, $y \subseteq x$

ⓑ Zeige $\mathcal{P}(A) \cap \mathcal{P}(B) = \mathcal{P}(A \cap B)$

$\mathcal{P}(A) \cap \mathcal{P}(B) \subseteq \mathcal{P}(A \cap B)$, denn wenn $x \in \mathcal{P}(A) \cap \mathcal{P}(B)$, dann $x \in \mathcal{P}(A)$ und $x \in \mathcal{P}(B)$, D.h. $x \subseteq A$ und $x \subseteq B$ d.h. $x \subseteq A \cap B$ also $x \in \mathcal{P}(A \cap B)$

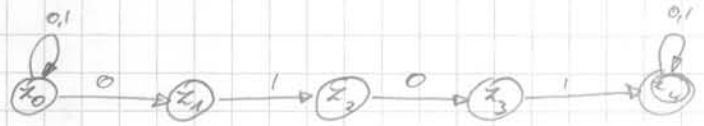
$\mathcal{P}(A \cap B) \subseteq \mathcal{P}(A) \cap \mathcal{P}(B)$, denn wenn $x \in \mathcal{P}(A \cap B)$, dann $x \subseteq (A \cap B)$, d. h. $x \subseteq A$ und $x \subseteq B$. Also $x \in \mathcal{P}(A)$ und $x \in \mathcal{P}(B)$, d. h. $\mathcal{P}(A \cap B) \subseteq \mathcal{P}(A) \cap \mathcal{P}(B)$

Zeigen sei das $\mathcal{P}(A) \cup \mathcal{P}(B) \neq \mathcal{P}(A \cup B)$ gilt

$A = \{1,2\}$ $\quad \mathcal{P}(\{1,2\}) \cup \mathcal{P}(\{3,4\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{3,4\}\}$ $\quad \mathcal{P}(A) = 2^{|A|}$ ← 7 Elemente
$B = \{3,4\}$ $\quad \mathcal{P}(\{1,2,3,4\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{1,3\}, \ldots\}$

da $\{1,3\} \in \mathcal{P}(\{1,2,3,4\})$ aber $\{1,3\} \notin \mathcal{P}(\{3,4\}) \cup \mathcal{P}(\{1,2\})$

Aufgabe ②

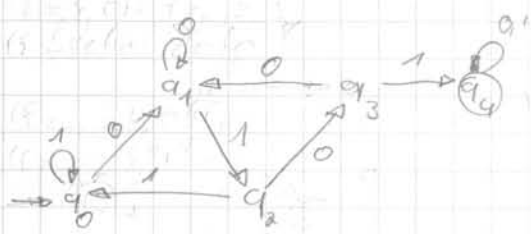


M_1

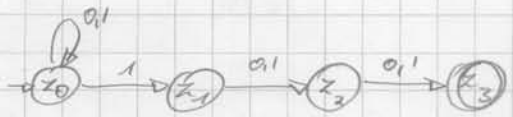
$M_1 (\{z_0, z_1, z_2, z_3, z_4\}, \{1, \emptyset\}, \delta, z_0, z_4)$

ⓑ $M_1' = (\mathcal{P}(\{z_0, z_1, z_2, z_3, z_4\}), \{0,1\}, \delta', \{z_0\}, E)$

$E = (\{z_0, z_2, z_4\}, \{z_0, z_4\}, \{z_0, z_1, z_4\}, \{z_0, z_1, z_3, z_4\})$



Aufgabe 3



$M_2' = (P(\{z_0, z_1, z_2, z_3\}), \delta', \{z_0\}, E)$

$E = \{\{z_0, z_3\}, \{z_0, z_1, z_3\}, \{z_0, z_2, z_3\}, \{z_0, z_1, z_2, z_3\}\}$

$\delta'(\{z_0\}, 0) = \{z_0\}$ ✓
$\delta'(\{z_0\}, 1) = \{z_0, z_1\}$ ✓

$\delta'(\{z_0, z_1\}, 0) = \{z_0, z_2\}$ ✓
$\delta'(\{z_0, z_1\}, 1) = \{z_0, z_1, z_2\}$ ✓

$\delta'(\{z_0, z_2\}, 0) = \{z_0, z_3\}$ ✓
$\delta'(\{z_0, z_2\}, 1) = \{z_0, z_1, z_3\}$ ✓

$\delta'(\{z_0, z_1, z_2\}, 0) = \{z_0, z_2, z_3\}$ ✓
$\delta'(\{z_0, z_1, z_2\}, 1) = \{z_0, z_1, z_2, z_3\}$ ✓

$\delta'(\{z_0, z_3\}, 0) = \{z_0\}$ ✓
$\delta'(\{z_0, z_3\}, 1) = \{z_0, z_1\}$ ✓

$\delta'(\{z_0, z_1, z_3\}, 0) = \{z_0, z_2\}$ ✓
$\delta'(\{z_0, z_1, z_3\}, 1) = \{z_0, z_1, z_2\}$ ✓

$\delta'(\{z_0, z_2, z_3\}, 0) = \{z_0, z_3\}$ ✓
$\delta'(\{z_0, z_2, z_3\}, 1) = \{z_0, z_1, z_3\}$ ✓

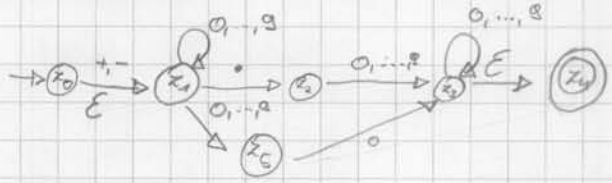
$\delta'(\{z_0, z_1, z_2, z_3\}, 0) = \{z_0, z_2, z_3\}$ ✓
$\delta'(\{z_0, z_1, z_2, z_3\}, 1) = \{z_0, z_1, z_2, z_3\}$ ✓

Für praktische Zwecke sind sogenannte ε-Übergänge ($\triangleq$ spontaner Übergang) oft sehr nützlich. Anschaulich:

ε-Übergänge sind mit ε beschriftet und nicht mit $a \in \Sigma$.

D.h. diese Übergänge konsumieren keinen Buchstaben der Eingabe.

Bsp: „Dezimalzahlen"



Def: „NEA mit ε-Übergängen"

Erweitern die Übergangsfunktion:

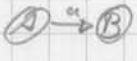
$$\delta : Z \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Z)$$

Satz: Für jeden „NEA" mit ε-Übergängen M existiert ein DEA M' mit $L(M) = L(M')$.

Beweis: Übung *

2.3 Reguläre Grammatiken und endliche Automaten

Satz: Jede durch einen endlichen Automaten akzeptierbare Sprache ist vom Typ 3.

Idee: Die Produktion $A \mapsto aB \triangleq$ dem Übergang 

$A \mapsto a \triangleq$ 

Beweis: Wurde L durch $M = (Z, \Sigma, \delta, z_0, E)$ akzeptiert, wobei M ein DEA ist. Nach obiger Idee wird aus einem Zustandsübergang von z nach $\delta(z,A)$ die Regel $z \mapsto a\,\underbrace{\delta(z,a)}_{\text{Nichtterminal}}$

Genauer: $G = (\Sigma, Z, P, z_0)$

und $P =_{def} \{ z \to a\,\delta(z,a) \mid z \in Z, a \in \Sigma \}$

$\cup \{ z \to a \mid \delta(z,a) \in E, z \in Z, a \in \Sigma \}$

Es gilt $a_1 a_2 \dots a_n \in L(M)$ gdw. es gibt Zustände $z_1 \dots z_n \in Z$ und $z_n \in E$, sodass $\delta(z_0, a_1) = z_1$, $\delta(z_1, a_2) = z_2$ ....

$$\delta(z_{n-1}, a_n) = z_n$$

gdw. es gibt eine Folge von Nichtterminalen $z_0, \dots, z_n$ der Grammatik G mit $z_0 \to a_1 z_1$, $z_1 \to a_2 z_2$, ... $z_{n-1} \to a_n$ D.h. $z_0 \vdash a_1 z_1 \vdash a_1 a_2 z_2 \overset{*}{\vdash} a_1 a_2 \dots a_n \in L(G)$

Also $L(G) = L(M) \setminus \{E\}$ und damit ist G auch vom Typ 3 und L auch vom Typ 3.

Satz: Für jede reguläre Sprache L, die von einer Typ 3 Grammatik G erzeugt wird, existiert ein NEA M mit $L = L(M)$

Idee: Regeln der Form $A \to aB$ werden zu $S(A,a) \ni B$

$A \to a$ werden zu $x \in \delta(A,a)$ und x ist ein Endzustand.

Beweis: Sei die reguläre Grammatik $G = (\Sigma, N, P, S)$ gegeben. Der gewünschte Automat $M = (Z, \Sigma, \delta, S, E)$ ist wie folgt definiert:

- $Z = N \cup \{x\}$, $x \notin N$
- $E = \{x\}$
- $B \in \delta(A,a)$ falls $A \to aB \in P$
  $x \in \delta(A,a)$ falls $A \to a \in P$

Dann gilt: $a_1 \dots a_n \in L(G)$ gdw. es gibt eine Folge von Nichtterminalen $A_1, \dots, A_{n-1}$ und $S \vdash a_1 A_1 \vdash a_1 a_2 A_2 \vdash \dots \vdash a_1 a_2 \dots a_{n-1} A_{n-1} \vdash a_1 a_2 \dots a_n$

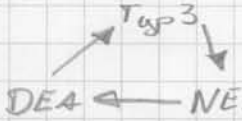
gdw. es gibt eine Folge von Zuständen $S, A_2, \dots A_{n-2}, x$ mit $A_2 \in \delta(S, A_1)$, $A_3 \in \delta(A_2, a_2)$, $\dots$, $x \in \delta(A_{n-2}, a_n)$

gdw. $a_1 \dots a_n \in L(M)$

Bsp: $G = (\{a, b\}, \{S, A\}, P, S)$

$P = \{S \to aA,\ A \to bA,\ A \to b\} \to ab^n \mid n \geq 1$

Folgerung: Die regulären Sprachen sind genau die durch DEA bzw. NEA akzeptierbare Sprachen, d.h.



2.4 Reguläre Ausdrücke

Reguläre Ausdrücke sind ein spezieller Formalismus mit dem Sprachen definiert werden können. Wir werden sehen, dass die genau die Typ 3 Sprache sind.

Reguläre Ausdrücke werden induktiv wie folgt definiert:

(IA) $\emptyset, \varepsilon$ und $a \in \Sigma$ sind reguläre Ausdrücke
(IS) wenn α und β reguläre Ausdrücke sind, dann auch $(\alpha)^*, \alpha\beta$ und $(\alpha \mid \beta)$

} Syntax

Alles andere sind keine regulären Ausdrücke.

Was bedeuten diese Ausdrücke?

Sei α ein regulärer Ausdruck, dann ist $L(\alpha)$ die Sprache die durch α festgelegt ist.

(IA) $L(\emptyset) = \text{def } \emptyset$

$L(\varepsilon) = \text{def } \{\varepsilon\}$ und $L(a) = \{a\}$

(IS) Sei $\gamma = \alpha\beta$, dann $L(\gamma) = \text{def } L(\alpha) \cdot L(\beta)$

Sei $\gamma = (\alpha | \beta)$, dann $L(\gamma) = \text{def } L(\alpha) \cup L(\beta)$

Sei $\gamma = (\alpha)^*$, dann $L(\gamma) = \{\epsilon\} \cup L(\gamma) \cup L(\gamma) \cdot L(\gamma) \cup L(\gamma) \cdot L(\gamma) \cdot L(\gamma) \cdots$

Aufgabe 1

$\bar{L} =_{def} \{ w \in \Sigma^* \mid w \notin L \}$

$L_1, L_2 \in \mathcal{C} \Rightarrow L_1 \cap L_2 \in \mathcal{C}$ — Abgeschlossen unter Schnitt

$L \in \mathcal{C} \Rightarrow \bar{L} \in \mathcal{C}$ — Abgeschlossen unter Komplementbildung

Beh: Die regulären Sprachen sind unter Komplementbildung abgeschlossen.

Idee:
- Für L vom Typ 3 gibt es eine reguläre Grammatik $G = (\Sigma, N, P, S)$

  Suche eine Grammatik
- Da L vom Typ 3 gibt es einen endlichen Automaten M mit $L = L(M)$

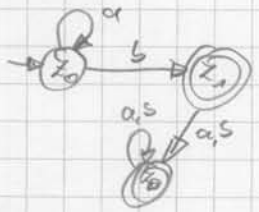
Bsp: $P = \{ S \to aS, S \to b \}$ $L(G) = \{ w \in \{a,b\}^* \mid w = a^n b, n \geq 0 \}$

etwas ist $\overline{L(G)}$ $bu \in L(G)$ ✓ wenn $u \in \Sigma^* \setminus \{\varepsilon\}$

$abba \in L(G)$ ✓

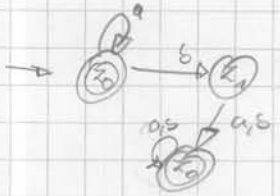
Scheint schwer zu sein? Etwas geht noch?

M:



akzeptiert $L(G)$

Idee: vertausche Endzustände mit Nichtendzustände



AFS
Ü
26.11.09

Sei L_1 regulär, dann ex. ein Automat M_1 mit $L_1 = L(M_1)$

und $M_1 = (Z, \Sigma, \delta, z_0, E)$

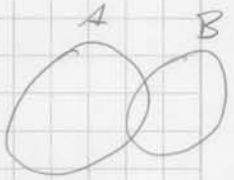
$\bar{M}_1 = (Z, \Sigma, \delta, z_0, Z \setminus E)$

D.h. $z \in Z$ ist ein Endzustand von M_1 gdw. es ist kein Endzustand von $\bar{M}_1$

Klar: $\omega \in L(M_1)$ gdw $\hat{\delta}(z_0, \omega) \in E$

gdw $\hat{\delta}(z_0, \omega) \in Z \setminus E$

gdw $\omega \notin L(\bar{M}_1)$



Mengenoperationen $A \cap B = B \cap A$, $A \cap (B \cap C) = (A \cap B) \cap C$

$A \cap (B \cup C) = (A \cap B) \cup (A \cup C)$

Beh: Die regulären Sprachen sind unter Vereinigung abgeschlossen.

Seien L_1 und L_2 vom Typ 3, dann gibt es reguläre Ausdrücke α und β mit $L_1 = L(\alpha)$ bzw $L_2 = L(\beta)$

Dann gilt $L_1 \cup L_2 = L(\alpha | \beta)$

Mit endlichen Automaten und ε-Übergängen.

$L(M_1) = L_1$ und $L(M_2) = L_2$ M

, dann $L(M) = L_1 \cup L_2$

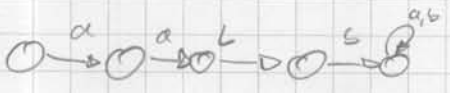
Seien L_1 und L_2 regulär, dann auch $\bar{L}_1$ und $\bar{L}_2$ regulär.

Also sind auch $L_1 \cup \bar{L}_2$ und $\overline{\bar{L}_1 \cup \bar{L}_2} = L_1 \cap L_2$ regulär

Aufgabe 2

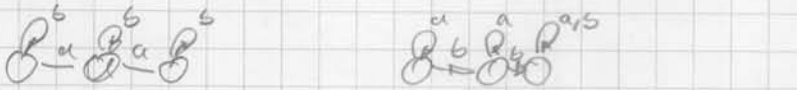
$L_3 = \{ w \in \{a,b\}^* \mid |w|_a = 2 \text{ und } |w|_b \geq 2 \}$

Idee: Baue einen Automaten



baab ↯

$L_3 = \{ w \in \{a,b\}^* \mid |w_a| = 2 \} \cap \{ w \in \{a,b\}^* \mid |w_a| \geq 2 \}$



Da die beiden Sprachen nach Schnitt abgeschlossen ist auch L_3 regulär.

$L_4 = \{ w \in \{a,b\}^* \mid |w|_a \text{ gerade und jedem a folgt ein b} \}$

$L_4 = \{ w \in \{a,b\}^* \mid |w|_a \text{ gerade} \} \cap \{ w \in \{a,b\}^* \mid \text{jedem a folgt ein b} \}$

→ L_4 ist regulär

Bsp: Durch $\gamma = (a \mid (a \mid b)^* aa)$ wird die Sprache

$\{a\} \cup \{w \in \{a,b\}^* \mid w \text{ endet mit } aa\}$ beschrieben

$L(a^* b c^*) = \{w \in \{a,b,c\} \mid w = a^n b c^m, n,m \geq 0\}$

Bem: Oft wird auch α^+ verwendet. Dies ist eine Abkürzung für $\alpha\alpha^*$ und $\alpha^?$ ist eine Abkürzung für $(\alpha \mid \varepsilon)$

Satz (Kleene): Die Menge der regulären Sprachen stimmt genau mit der Menge von Sprachen überein, die man mit regulären Ausdrücken beschreiben kann.

Beweis (Skizze):

Zwei Richtungen „$\Rightarrow$" Für jede Typ 3 Grammatik G gibt es einen regulären Ausdruck R mit $L(G) = L(R)$

„$\Leftarrow$" Für jeden regulären Ausdruck gibt es einen NEA mit ε-Übergängen M mit $L(R) = L(M)$

„$\Leftarrow$" Idee: Mache einen Induktionsbeweis über die Struktur der regulären Ausdrücke

(IA) $\gamma = \alpha$ wird zu $\rightarrow \bigcirc \xrightarrow{\alpha} \textcircled{\bigcirc}$

$\gamma = \varepsilon$ $\rightarrow \textcircled{\bigcirc}$

$\gamma = \emptyset$

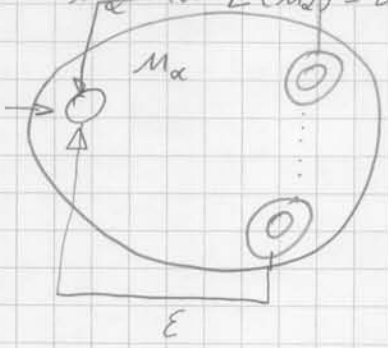
(IV) Sei γ ein regulärer Ausdruck, dann sei M_γ ein NEA mit $L(M_\gamma) = L(\gamma)$

(IS) Fall $\gamma = \alpha \cdot \beta$, nach (IV) haben wir NEAs M_α bzw M_β mit $L(\alpha) = L(M_\alpha)$ bzw $L(\beta) = L(M_\beta)$
„Verkleben" Endzustände von M_α mit dem Startzustand von Kopien von M_β zu einem neuen Zustand

Fall $\gamma = \alpha | \beta$ Nach (IV) gibt es NEA M_α bzw M_β mit $L(\alpha) = L(M_\alpha)$ $L(\beta) = L(M_\beta)$

„Parallelschaltung"

Fall $\gamma = (\alpha^*)$ Nach (IV) haben wir einen NEA M_α mit $\varepsilon \notin L(M_\alpha) = L(\alpha)$



1. Mache Startzustand zu einem Endzustand, da $\varepsilon \in L(\gamma)$
2. Rückwärtskanten von den Endzustände zum Startzustand einführen.

„⇒" Sei $G = (\Sigma, N, P, S)$ eine Grammatik vom Typ 3

Wir konstruieren einen regulären Ausdruck R aus G mit $L(G) = L(R)$.

Sei $A \in N$ und $A \to a_1 B_1, \dots, A \to a_m B_m$

$A \to b_1, \dots, A \to b_k$ sind alle Regeln in P mit A auf der linken Seite.

$$L_A \overset{\text{def}}{=} \{ w \in \Sigma \mid A \vdash w \} = a_1 L_{B_1} \cup \dots \cup a_m L_{B_m} \cup b_1 \cup \dots \cup b_k$$

Sowas kann man in einen regulären Ausdruck schreiben, wenn man die Ausdrücke für $L_{B_1}, \dots, L_{B_m}$ hat.

⇒ Schreibe ein Gleichungssystem für alle Nichtterminale von G.
Dieses Gleichungssystem kann man in den regulären Ausdruck R umwandeln.

#

AFS VL
26.11.09

Das Pumpinglemma

Bis jetzt können wir von einer Sprache nur zeigen dass sie regulär ist. (Automat, Typ 3, reg. Ausdruck)

Was ist mit der Sprache $L = \{ w \in \{a,b\}^* \mid w = a^n b^n \}$

Offensichtlich ist $G = (\{a,b\}, \{S\}, \{S \to aS,\ S \to aSb\}, S)$ eine Typ 2 Grammatik mit $L(G) = L$.

Frage: Geht das auch mit einer Typ 3 Grammatik?

Satz (Pumping Lemma / uvw-Theorem)

Sei L eine reguläre Sprache, dann gibt es eine Zahl $n \in \mathbb{N}$, so dass sich alle Wörter $x \in L$ mit $|x| \geq n$ zerlegen lasse in $x = uvw$ und folgende Eigenschaften gelten:

i) $v \geq 1$

ii) $uv \leq n$

iii) für alle $i \in \mathbb{N}$ gilt $uv^i w \in L$

Bsp: Die Sprache $L = a^m b^m$, $m \geq 1$ ist nicht regulär.

Beweis: Annahme L wäre regulär. Dann ex. eine Zahl $n \geq 1$ sodass jedes Wort $x \in L$ mit $|x| \geq n$ läßt sich wie im Pumpinglemma beschrieben zerlegen.

Wählen $x = a^n b^n$ also ein Wort der Länge $2n$.

Damit ex. eine Zerlegung von x, wobei

v nicht leer und $|uv| \leq n$

Der String v besteht also nur aus dem ersten a.

Nach Pumping Lemma ist dann auch $uv^0 w = uvw = a^{n-|v|} b^n \in L$

d.h. $a^{n-|v|} b^n$ hat echt weniger a's als b's ↯

Das heißt die Annahme war falsch! $\Rightarrow$ L ist nicht regulär ✱ 26.11.09

Folgerung $L_3 \subset L_2$

Bemerkung Das Pumping-lemma nach dem gleichen Kochrezept angewandt

1. Annahme L sei regulär
2. Wähle geschickt ein Wort das ausreichend lang ist
3. zeige, dass einer der Wörter uvw nicht in L sein kann $\lightning$ Annahme falsch.

Automaten Theorie
Übung
03.12.09

③

$L(\gamma_1) = \{ w \in \{0,1\}^* \mid w \text{ enthält genau eine } 1 \}$

$L(\gamma_2) = \{ w \in \{0,1\}^* \mid w \text{ enthält mindestens eine } 1 \}$

$L(\gamma_3) = \{ w \in \{0,1\}^* \mid w \text{ hat die Form } 1^{m}0^{n_1}0^{n_2}\dots 0^{n_k}, \text{ wobei } m \geq 0,\ n_0 \dots\ n_k > 0,\ k \geq 0 \}$

$\Rightarrow$ in w folgt auf jede 0 mindestens eine 1

$L(\gamma_4) = \{ w \in \{0,1\}^* \mid |w| \text{ ist durch 3 teilbar} \}$ $\quad |w| \bmod 3 = = 0$

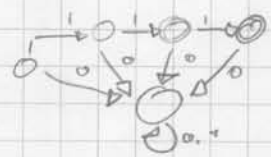
②

$\gamma_1 = 1(011)^*0$

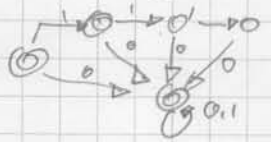
$\gamma_2 = (110)^* 1 (110)^* 1 (10)^* 1 (110)^*$

$\gamma_3 = (110)^* 0101 (110)^*$

$\gamma_4 = \varepsilon \mid 11 \mid (0 \mid 110 \mid 1101 \mid 111 (110)) (011)^*$

M  $\quad L(M) = \{11, 111\}$

↓ Komplementbildung

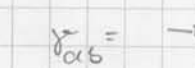
$\bar{M}$  $\quad L(\bar{M}) = \overline{L(M)}$

Untersuche alle Pfade von einem Start-Zustand, bis zu einem Endzustand in $\bar{M}$ um eine für $L(\bar{M})$ zu bekommen

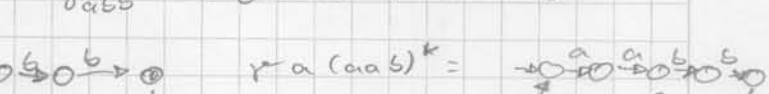
$L(\bar{M}) = \{\varepsilon\} \cup \{0\} \cup \{1\} \cup \{10\} \cup \{110\} \cup \{1110, 1111\} \cup \dots$

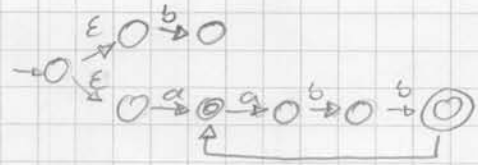
① $\Sigma = \{a, b\}$

$\gamma_{a^*b} = (a(abb)^*) \mid b$

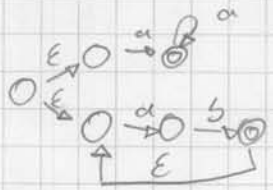
$\gamma_a =$  $\qquad \gamma_{ab} =$ 

$\gamma_b =$ $\qquad \gamma_{abb} =$ 

$\gamma_{(abb)^*} =$  $\qquad \gamma_{a(aab)^*} =$ 

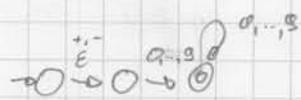
$\gamma_1 =$ 

Siehe Buch Aho, Sethi, Ullmann - The Principles of Compiler Design „Dragonbook"

$\gamma_2 =$ 

Klausur:
Epsilon Übergänge + Potenzmengenfunktion

Potenzmengenkonstruktion für NEAS mit ε Übergängen

Bsp: 

ε-closure $(z_0) = \{z_0, z_n\}$

Def: Sei ε-closure $(z) : Z \Rightarrow P(Z)$

Potenzfkt mit ε-closure aufschreiben

(IA) $z \in \varepsilon$-closure (z)

(IS) Sei $p \in \varepsilon$-closure (z) und es gilt $\delta(p, \varepsilon) \exists q \; (q \in \delta(p, \varepsilon))$, dann ist auch $q \in \varepsilon$-closure (z)

Erweitern den ε-closure auf eine Menge von Zuständen

$$\varepsilon\text{-closure}(Z) = \bigcup_{z \in Z} \varepsilon\text{-closure}(z)$$

Satz Pumping Lemma

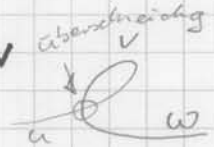
Sei L regulär, dann gibt es ein $n \in \mathbb{N}$ mit der Eigenschaft:

jedes $x \in L$ mit $|x| \geq n$ läßt sich zerlegen in $x = uvw$

wobei

$|v| \geq 1$ (kleines N)

$|uv| \leq n$ (kleines N)



$uv^i w \in L$ für $i \in \mathbb{N}$

Beweis: Da L regulär ist, gibt es einen DEA M mit $L = L(M)$

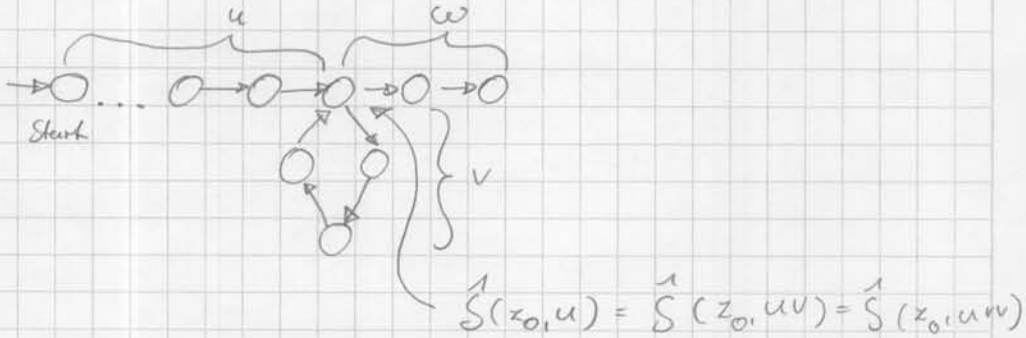
wir legen fest: $n = \#Z$, d.h. ist die Anzahl der Zustände von M

Sei nun x ein beliebiges Wort, das von M akzeptiert wird und $|x| \geq n$

Beim Abarbeiten von x besucht M genau $|x|+1$ Zustände von M

Da $|x| \geq n$ können diese Zustände nicht verschieden sein. D.h. mindestens ein Zustand wird mindestens zwei mal besucht.

(Schubfachschluß)



D.h. wir wählen die Zerlegung von $x = uvw$ so, daß der Zustand von u gleich dem Zustand ist nach Lesen von uv.

AFS VL
03.12.09

Klar: $|v| \geq 1$ (Schleife kann nicht leer sein)

und $|uv| \leq n$ (klar weil nur n Zustände existieren)

Wiederholung: Wenn $uvw \in L$, dann auch $uv^0w \in L$

$uvvw = uv^2w \in L$, $uvvvw = uv^3w \in L$ ... $uv^nw \in L$ *

Bsp: Die Sprache $L = \{w \in \{0\}^* \mid w = 0^p, p \text{ eine Primzahl}\}$

$= \{0^0, 0^3, 0^5, 0^7 ...\}$ ist nicht regulär

Beweis: Annahme L ist regulär, dann existiert ein $n \in \mathbb{N}$ und jedes $x \in L$ mit $|x| \geq n$ läßt sich wie im Pumping Lemma beschrieben zerlegen.

Es gibt endlich viele Primzahlen, also auch eine Primzahl p mit der Eigenschaft $p \geq n+2$.

Also $0^p \in L$

Also müssen alle Wörter der Form $0^{|uw|+i|v|}$ in L liegen

Wähle $i \in |uw|$ und erhalten $|uw| + |uw|\cdot|v| = |uw| \cdot \underbrace{(1+|v|)}_{\geq 2}$

Das heißt $0^{|uw|+|uw|\cdot|v|} \notin L$ ↯ ⇒ Aussage ist falsch

⇒ L ist nicht regulär

Bsp: Sei $= \{w \in \{0\}^* \mid w = 0^n$, n ist eine Quadratzahl$\}$ ist nicht regulär

Beweis: Übung* finde $|uw| \to$ kein Quadratzahl *

Test

3. Kontextfreie Sprachen

Ziel: Typ 2 Sprachen näher untersuchen

Pumping Lemma a für a"b"

Wissa: Eine Grammatik $G = (\Sigma, N, P, S)$ ist vom Typ 2 gdw. jede Produktion der Form $A \to w$ mit $A \in N$, $w \in (N \cup \Sigma)^*$, $|w| \geq 1$ entspricht

3.1 Chomsky Normalform

Def: Eine kfG (Kontextfreie Grammatik) heißt in Chomsky-Normalform, falls alle Regeln entweder die Form $A \to BC$ mit $A, B, C \in N$ oder $A \to a$ mit $A \in N$, $a \in \Sigma$ haben.

Bem: Die Chomsky Normalform ist interessant, weil die Ableitungsbäume (bis zum letzten Schritt) Binärbäume sind. (evtl. unbalanciert)

Test

Lemma: Sei $G = (\Sigma, N, P, S)$ in Chomsky Normalform, wenn $w \in L(G)$, dann $S \overset{2|w|-1}{\vdash} w$.

Beweis: $S \overset{2|w|-1}{\vdash} w$ selber machen → siehe Binärbaum Höhe $\overset{+}{\circ}$ |

Satz: Zu jeder kfG G gibt es G' in Chomsky Normalform mit $L(G) = L(G')$

Beweis: Wir führen 3 Schritte durch:

i) „Eliminieren von Terminalen an der falschen Position"

Wir formen G in G_1 um, so dass Terminale nur noch in Regeln der Form $A \to a$ vorkommen. Ersetze in jeder Regel $A \to w$ jedes Terminal $a \in \Sigma$ durch das neue Nichtterminal D_a wobei $D_a \notin N$ und füge die Regel $D_a \to a$ zu der Menge der Produktionen hinzu.

Klar $L(G) = L(G')$

(ii) „Entferne Kettenregeln"

Regeln der Form $A \vdash B$ heißen Kettenregeln.

Wir erzeugen eine Grammatik G_2 mit $L(G_2) = L(G_1)$
bei der nur Regeln der Form $A \vdash B_1 B_2 \dots B_k$
mit $k \geq 2$ und $A \vdash a$ vorkommen, wobei
$A, B_1 \dots B_k \in N$ und $a \in \Sigma$

Jede Folge $A_1 \to A_2$, $A_2 \to A_3$, $A_{n-1} \to A_n$ und
$A_n \vdash B_1 \dots B_k$ mit $k \geq 2$ und $A_1, A_2 \dots A_n$ und $B_1 \dots B_k$
wird überführt in $A_1 \vdash B_1 \dots B_k$ und die Kettenregeln
werden entfernt. Analog: aus $A_1 \to A_2$, $A_2 \vdash A_3$, $A_n \to a$
wird $A_1 \vdash a$

Klar $L(G_2) = L(G_3)$

Blatt 8

AFS
Ü
10.12.09

Aufgabe 3:

Für eine Grammatik in Chomsky-NF ergeben sich Syntaxbäume der Form

} Regeln der Form $A \mapsto B\,C$
} Regeln der Form $A \mapsto a$

Dh. gilt $x \in L(G)$, dann sieht der Ableitungsprozeß wie folgt aus:

In jedem Ableitungs-Schritt kommt genau ein Nichtterminal hinzu

$$\left.\begin{aligned} S &\vdash A_1 A_2 \vdash A_1 A_2' A_3' \vdash \dots \vdash A_1' A_2'' \dots A_{|x|}'' \\ &\vdash a_1 A_2'' \dots A_{|x|}'' \vdash a_1 a_2 \dots A_{|x|}'' \vdash \dots \vdash a_1 \dots a_{|x|} \end{aligned}\right\} \begin{aligned} &\text{Phase I} \\ &\text{Phase II} \end{aligned}$$

Schritte insgesamt: $\underbrace{|x|-1}_{\text{Phase I}} + \underbrace{|x|}_{\text{Phase II}} = 2|x|-1$

Aufgabe 1:

a) $\{w \in \{0,1\}^* \mid |w|_1 = |w|_0\}$ $(\supseteq a^m b^m)$
Teilmenge

Satz: L_1 ist nicht regulär

Beweis: Annahme L_1 ist regulär, dann gilt das Pumping-Lemma und es gibt eine Konstante $n \in \mathbb{N}$, sodass die 3 Eigenschaften für die Zerlegung von Wörter $x \in L$ mit $|x| \geq n$ gelten.

Wählen $x = 0^n 1^n \in L_1$ mit Länge $2n$.
Nach Pumping Lemma kann x in 3 Teile aufgeteilt werden.
$x = uvw$
$|uv| \geq 1$
$|uv| \leq n$
$uv^i w \in L_1 \; \forall i \in \mathbb{N}$

Da $|uv| \leq n$ besteht uv aus 0en, d.h. v enthält mindestens eine 0.
Das Wort $uv^0 w = uw$ enthält deshalb mindestens eine 0 weniger als es vor enthält. $\lightning$ $uw \notin L_1$
Daraus folgt: L_1 ist nicht regulär $\lightning$

b) $\{www \mid w \in \{0,1\}^*\}$

Satz: L_2 ist nicht regulär

Beweis: Annahme L_2 ist regulär, dann gilt das Pumping Lemma und es gilt eine Konstante $n \in \mathbb{N}$, so dass jedes Wort zerlegt werden kann $x = uvw$ und

$|uv| \geq 1$
$|uv| \leq n$
$uv^i w \in L_1 \; \forall i \in \mathbb{N}$ gilt.

Wähle $x = \underbrace{0^n 1 0^n 1}_{w\,w}$

$x = \underbrace{0^i}_{u} \underbrace{0^i}_{w} \in L_2$

Da $x = uvw$, besteht v nur aus Os und enthält mindestens eine 0.
Dies bewirkt, dass $uv^i w \notin L_2$ für $i \geq 2$ ist. ↯
Annahme war falsch. L_2 ist also nicht regulär. ⁂

(3) $L_3 = \{ 0^{2^m} \mid m \geq 0 \} = \{ 0, 00, 0000, \dots \}$

Satz: L_3 ist nicht regulär

Annahme: L_3 ist regulär...

Sei $x = 0^{2^n} \in L_3$, dann gilt $|x| = 2^n \geq n$
Damit gilt $|uvvw| = \underbrace{|uvw|}_{\text{keine 2er Potenz}} + |v| < 2^n + 2^n + 1$

$\Rightarrow uvvw \notin L_3$

(2) Pumping Lemma mit Regulären Grammatiken beweisen

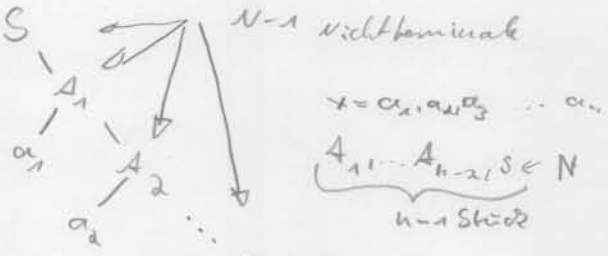
Sei L regulär, dann gibt es ein $n \in \mathbb{N}$ mit der Eigenschaft:
Für alle $x \in L$ mit $|x| \geq n$ und $x = uvw$ gilt

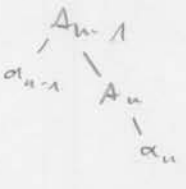
i) $|v| \geq 1$
ii) $|uv| \leq n$
iii) $uv^i w \in L$, $\forall$ a. $i \in \mathbb{N}$

Beweis: Da L regulär ist, existiert eine Typ 3 Grammatik

$G = (\Sigma, N, P, S)$ und $L(G) = L$,

Ein Syntaxbaum für $x \in L$ sieht dann wie folgt aus:





Wir definieren $n =_{def} \#(N) + 2$

Für jedes Wort $x \in L$ mit $|x| \geq n$
gilt: Im Syntaxbaum kommen mindestens
$\#(N) + 1$ Nichtterminale vor $\Rightarrow$ mind. ein
Nichtterminal kommt doppelt vor. (Schubfachschluß) Etwa

D.h. x kann zerlegt werden in $uvw = x$ und $|v| \geq 1$,
denn bei jedem Ersetzungsschritt wird ein Terminal erzeugt.
Und spätestens $\#(N) - 1$ Nichtterminale kommt die "Schleife", d.h. $|uv| \leq \#(N) + 2 = n$
Vervielfältigung von (I) ergibt $uv^i w \in L$ für alle $i \in \mathbb{N}$ ✓

iii) „Regeln kürzen"

Nun wird die Grammatik G' erzeugt. Alle zu langen Regeln der Form $A \to B_1 B_2 \dots B_k$, $k \geq 3$ werden wie folgt umgeformt:

Füge für jede Regel neue Nichtterminale $D_1, \dots, D_{k-1}$ ein und ersetze $A \to B_1 B_2 \dots B_k$ durch $A \to B_1 D_2$, $D_2 \to B_2 D_3$, ... $D_i \to B_i D_{i+1}$ für $2 \leq i \leq k-2$ ... $D_{k-1} \to B_{k-1} B_k$

Wieder gilt $L(G') = L(G)$

Teilaufgabe einer Klausur

Bsp: $G = (\{a,b,c\}, \{S, A, B\}, P, S)$ — keine Chomsky NF-Regeln

$P = \{S \to AB,\ A \to aAb,\ A \to ab,\ B \to cB,\ B \to c\}$

$L(G) = \{w \in \{a,b,c\}^* \mid a^n b^n c^m,\ n, m > 0\}$

zu lang siehe (iii)

Schritt (i) $A \to aAb$ wird zu $A \to A_a A A_b$, $A_a \to a$, $A_b \to b$

$A \to ab$ — " — $A \to A_a A_b$

$B \to cb$ — " — $B \to A_c B$, $A_c \to c$

Schritt (ii) Kettenregeln entfernen, entfällt weil wir keine Kettenregeln haben

Schritt (iii) Lange Regeln kürzen

Die Regel $A \to A_a A A_b$ wird zu $A \to A_a D_2$, $D_2 \to A A_b$

$G' = (\{a,b,c\}, \{S, A, B, A_a, A_b, A_c, D_2\}, P', S)$

$P = \{S \to AB,\ A \to A_a D_2,\ D_2 \to A A_b,\ A_a \to a,\ A_b \to b,\ A \to A_a A_b$
$B \to A_c B,\ A_c \to c,\ B \to c\}$ → ist in Chomsky Normalform.

Lemma: Sei B ein Binärbaum. Wenn $B \geq 2^k$ Blätter hat, so hat B mindestens einen Pfad der Länge $\geq k$ von der Wurzel bis zu einem Blatt.

Beweis: Induktion über die Länge des längsten Pfades. ⇒ Übung #
Oder: Kürzester Beweis! Preis: eine Packung Gummibärchen

3.2. Pumping Lemma für kontextfreie Sprachen

Satz: Pumping Lemma für kontextfreie Sprachen, beruht auf uvwxy-Theorem

Sei L eine kontextfreie Sprache, dann gibt es eine Zahl $n \in \mathbb{N}$, so dass sich alle Wörter $z \in L$ mit $|z| \geq n$ zerlegen lassen in $z = uvwxy$, wobei:

$|vx| \geq 1$

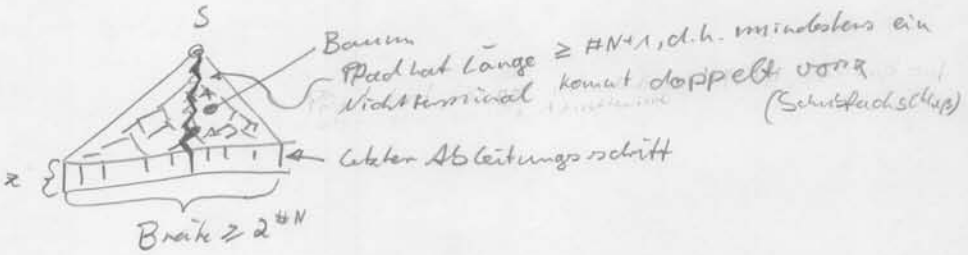
$|vwx| \leq n$

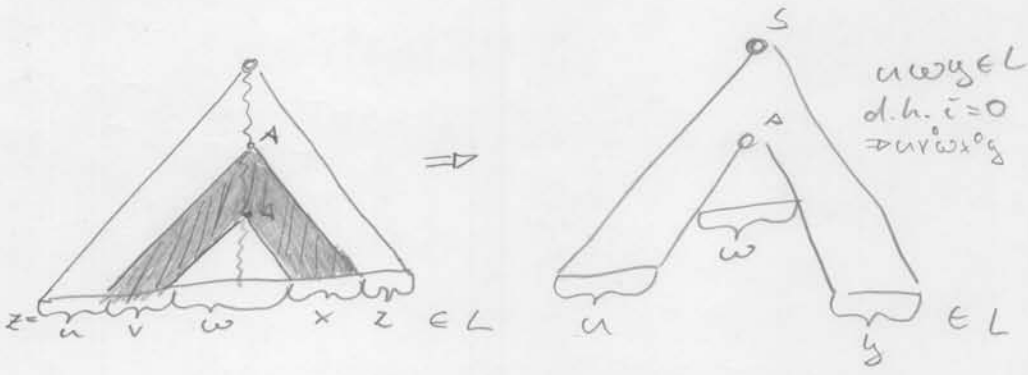
Für alle $i \in \mathbb{N}$ gilt $uv^i wx^i y \in L$ (i-fach wird an 2 Stellen "gepumpt")

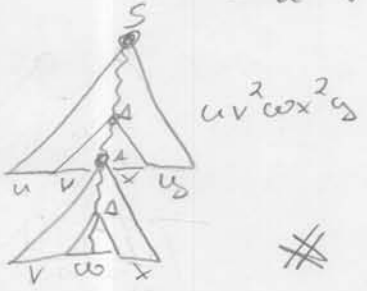
Beweis (Idee): o.B.d.A sei G eine Grammatik in Chomsky-NF mit nichtterminale und $L \setminus \{\varepsilon\} = L(G)$

Wir wählen $n = \text{def } 2^{\#N}$ als Pumpingkonstante, wenn $G = (\Sigma, N, P, S)$

Der Syntaxbaum bei der Ableitung von z sieht wie folgt aus







uv^2wx^2y

$\lightning$

Beispiel: Die Sprache $L = \{z \in \{a,b,c\}^* \mid z = a^m b^m c^m, m \geq 1\}$
ist nicht kontextfrei, aber kontextsensitiv (Übung).

Beweis: Annahme L ist kontextfrei, dann gibt es nach dem Pumping Lemma eine Zahl n, so dass sich alle Wörter $z = a^m b^m c^m$, $|z| \geq n$ zerlegen lassen in $z = uvwxy$ mit der Eigenschaft i), ii), iii).

Wähle speziell $z = a^n b^n c^n$ mit der Länge $3n$

Wegen Eigenschaft (ii) kann vx nicht alle 3 Arten von Buchstaben enthalten, denn die Länge von $|vwx| \leq n$

Wegen Eigenschaft (iii) muß aber $uv^0wx^0y = uwy \in L$ gelten, d.h. mindestens eine Art von Buchstaben in uwy kommt öfter vor als die anderen Arten. ↯ So ein Wort ist nicht in L
$\Rightarrow$ Annahme falsch
$\Rightarrow L$ ist nicht kontextfrei #

Folgerung: $L_2 \subset L_1$

es ex. kontextsensitive Sprachen die nicht kontextfrei sind!

Bsp: Die Sprache $L = \{z = \{0\}^* \mid z = 0^p, p \text{ Primzahl}\}$
ist nicht kontextfrei.

Beweis: Übung #

3.3 Der CYK-Algorithmus

Wissen: Das Wortproblem für Typ1, Typ2, Typ3 ist lösbar
(Typ1: langsam, Typ3: schnell)

Für Typ3 kennen wir einen schnellen Algorithmus für das Wortproblem, für Typ1 gibt es wohl keinen und für Typ2 haben wir noch keine Plan.

Automatentheorie und Formale Sprachen

Übungsblatt 5

1 Eine Potenzmenge ist wie folgt definiert: $\mathcal{P}(x) = \text{def } \{Y \mid Y \subseteq X\}$

(a) geben sie $\mathcal{P}(\{1,2\})$ und $\mathcal{P}(\{x,y,z\})$ explizit an.

$\mathcal{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$

$\mathcal{P}(\{x,y,z\}) = \{\emptyset, \{x\}, \{y\}, \{z\}, \{x,y\}, \{x,z\}, \{y,z\}, \{x,y,z\}\}$

(b) Seien A und B beliebige Mengen

zeigen sie dass $\mathcal{P}(A) \cap \mathcal{P}(B) = \mathcal{P}(A \cap B)$ gilt (geschnitten)

$\mathcal{P}(A) \cup \mathcal{P}(B) \neq \mathcal{P}(A \cup B)$ (vereinigt)

$\mathcal{P}(A) \cap \mathcal{P}(B) \rightarrow$ Alle Elemente die in der Potenzmenge von A und B liegen.

Bsp. Keine Schnittmenge $\{1,2\} \cap \{3,4\} \Rightarrow \{\emptyset\}$

$\mathcal{P}(\{1,2\}) = \emptyset, \{1\}, \{2\}, \{1,2\}$

$\mathcal{P}(\{3,4\}) = \emptyset, \{3\}, \{4\}, \{3,4\}$

$\mathcal{P}(\{1,2\}) \cap \mathcal{P}(\{3,4\}) = \{\emptyset\}$

$\mathcal{P}(\{1,2\} \cap \{3,4\}) = \{\emptyset\}$ ✓

ein Element in der Schnittmenge $\{1,2\} \cap \{2,3\} \Rightarrow \{\emptyset, 2\}$

$\mathcal{P}(\{1,2\}) = \emptyset, \{1\}, \{2\}, \{1,2\}$

$\mathcal{P}(\{2,3\}) = \emptyset, \{2\}, \{3\}, \{2,3\}$

$\mathcal{P}(\{1,2\}) \cap \mathcal{P}(\{2,3\}) = \emptyset, \{2\}$

$\mathcal{P}(\{1,2\} \cap \{2,3\}) = \mathcal{P}(\{2\}) = \emptyset, \{2\}$ ✓

$\mathcal{P}(A) \cup \mathcal{P}(B)$

$\mathcal{P}((A) \cup (B))$

$\mathcal{P}(\{1,2\}) = \emptyset, \{1\}, \{2\}, \{1,2\}$

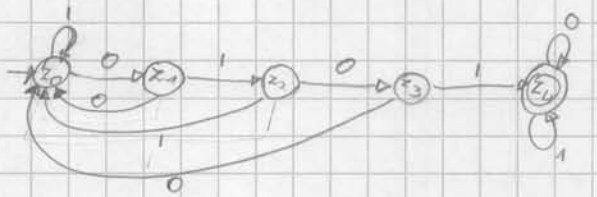
$\mathcal{P}(\{3,4\}) = \emptyset, \{3\}, \{4\}, \{3,4\}$

$\mathcal{P}(\{1,2\}) \cup \mathcal{P}(\{3,4\}) = \emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{3,4\}$

$\mathcal{P}(\{1,2\} \cup \{3,4\}) = \mathcal{P}(\{1,2,3,4\}) = \emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \ldots$

Sei $L_1 = \{\omega \in \{0,1\}^* \mid \omega \text{ enthält } 0101 \text{ als Teilstring}\}$

M_1 δ



$M_1 (\{z_0, z_1, z_2, z_3, z_4\}, \{1,0\}, \delta, z_0, z_4)$

$\hat{\delta}(\{z_0\}, 1) = \{z_0\}$ $\quad$ $\hat{\delta}(\{z_4\}, 0) = \{z_0\}$

$\hat{\delta}(\{z_0\}, 0) = \{z_1\}$ $\quad$ $\hat{\delta}(\{z_1\}, 1) = \{z_2\}$

$\hat{\delta}(\{z_2\}, 0) = \{z_3\}$ $\quad$ $\hat{\delta}(\{z_3\}, 1) = \{z_4\}$

$\hat{\delta}(\{z_2\}, 1) = \{z_0\}$ $\quad$ $\hat{\delta}(\{z_3\}, 0) = \{z_0\}$

$\hat{\delta}(\{z_4\}, 1) = \{z_4\}$

$\hat{\delta}(\{z_4\}, 0) = \{z_4\}$

$L_2 = \{\omega \in \{0,1\}^* \mid \text{vorletzter Buchstabe ist eine } 1\}$? → Literatur

yacc / bison	AFS 1
compilerbautools	Übung

① $G = (\{(,), V, \wedge, \neg, x\}, \{F\}, F, P)$

$P = \{\underbrace{F \to (F \wedge F), F \to (F \vee F), F \to \neg F,}_{\text{Entfernen}} F \to x\}$

$G' \to$ Chomsky Normalform

1. $F \to (F \wedge F)$ wird zu $F \to AFUFZ$

   $F \to (F \vee F)$ $F \to AFOFZ$

   $F \to \neg F$ $F \to NF$

Neue Regeln $A \to (,\ Z \to),\ U \to \wedge,\ O \to \vee,\ N \to \neg$

Regeln kürzen:

$F \to AFUFZ$ wird zu $F \to AR_1, R_1 \to FR_2, R_2 \to UR_3, R_3 \to FZ$

$F \to AFOFZ$ —"— —"— —"— $R_2 \to OR_3$ —"—

$G' = (\{\}, (, \wedge, \vee, \neg, x\}, \{F, A, Z, U, O, N, R_1, R_2, R_3\}, P, F)$

$P = \{A \to (, Z \to), U \to \wedge, O \to \vee, N \to \neg, F \to NF, F \to AR_1, R_1 \to FR_2, R_2 \to OR_3,$

$R_2 \to UR_3, R_3 \to FZ, F \to x\}$

② $L_1 = \{0^p \mid p$ ist Primzahl$\}$ ist nicht kontextfrei

Annahme ist kontextfrei, dann $\exists\ n \in \mathbb{N}$ und es gelten die Eigenschaften uvwxy-Theorem

Wählen $z = 0^p$, wobei p Primzahl und $n \leq p$. Also existiert eine Zerlegung $z = uvwxy$ und alle Wörter $uv^i wx^i y$, $i \geq 0$ sind in L_1 enthalten.

Wähle $i = p+1$ also $z' = uv^{i+1}wx^{i+1}y \in L_1$

Also $|z'| = |u| + |v|(p+1) + |w| + |x|(p+1) + |y| = \overbrace{|u| + |v| + |w| + |x| + |y|}^{=p} + p|v| + p|x|$

$= p + |v|p + |x| \cdot p = p \cdot (1 + |v| + |x|)$

0^{n^2} → Typ 1 Grammatik

Da nach PL $|uv| \geq 1$, d.h. $(1 + |v| + |w|) \geq 2$ Also ist $|z'|$ keine Primzahl $\overset{\text{↯}}{0}$

⇒ Annahme falsch, ⇒ L ist nicht kontextfrei

④ $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$

$L_3 = \{a^n b^n c^m \mid m, n \neq 0\}$

L_3 ist kontextfrei vermöge $G = (\{a, b, c\}, \{S, A, B\}, S, P_3)$

$P_3 = \{S \to A|B|AB,\ A \to aAb|ab,\ B \to cB,\ B \to c\}$

Analog: L_3 ist kontextfrei (A und B vertauschen)

$L_2 \cap L_3 = \{a^m b^m c^m \mid m \geq 0\}$ ist nicht kontextfrei

a b c
a b c

d.h. Typ 2 ist nicht Schnitt abgeschlossen

Beweistechnik: Gegenbeispiel

⑤ Annahme: Die kontextfreien Sprachen sind unter Komplement abgeschlossen:

$\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$ ↯

⇒ nicht unter Komplement abgeschlossen

① Dynamisches Programmieren → Zwischenergebnisse in Tabelle aufheben

CYK - Algorithmus

$O(n^c)$ → schnell

Wissen: Das Wortproblem für Typ 1,2,3 ist lösbar.

Ziel: Ein schneller Algorithmus für das Wortproblem für kontextfreie Sprachen

CYK - Algorithmus (Cocke, Younger, Kasami)

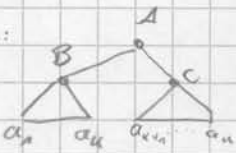
Idee: Für jede Typ-2 Sprache gibt es eine erzeugende Grammatik in Chomsky-Normalform.

Damit:
- Worte der Länge 1: Es kommt nur eine Regel der Form $A \to a$ in Frage
- Worte die länger sind als 1: Wenn gilt $A \vdash^* x$, dann wurde im ersten Schritt eine Regel der Form $A \to BC$ angewendet

$\Rightarrow$ Wenn $x = A_1 \overset{(k)}{\cdots} A_n$ wäre, dann wird ein Anfangsstück aus x von B erzeugt und das Endstück aus C $(A \to BC)$

Es muß also ein k $(1 \le k < n)$ geben, so daß gilt

Syntaxbaum:



Damit können wir das Wortproblem für x auf 2 Wortprobleme für $x_1 \dots x_k$ bzw $x_{k+1} \dots x_n$ zurückführen.

Aber: k ist unbekannt $\Rightarrow$ alle Möglichkeiten muss probieren.

Dazu verwenden wir die Methode des Dynamischen Programmierens

Notation: Sei $x = a_1 \dots a_n$ dann ist $x_{i,j}$ das Teilwort von x, dass an Position i startet und j Buchstaben lang.

Algorithmus verwendet Tabelle $T[1 \dots n][1 \dots n]$, wobei nur die obere Dreiecksmatrix verwendet wird.

In $T[i][j]$ sind alle Nichtterminale gespeichert aus denen $x_{i,j}$ abgeleitet werden kann. (Jeder Eintrag in Array ist eine Menge).

Es gilt $x = a_1 \dots a_n \in L(G)$ gdw $T[1][n] \ni S$

Algorithmus:

```
bool CYK( G=(Σ,N,P,S), x=a_1 ... a_n, n ){
    for(i=1 to n){                                  // alle Produktionen wie A↦a
        T[i][1] = { A∈N | A↦a_i ∈P}                 // A Wort der Länge 1
    }
    for(j=2 to n){ // Worte mit Länge > 1
        for(i=1 to n+1-j){ // alle mögliche Startpositionen
            T[i][j] = ∅
            for(k=1 to n j-1){ // Alle möglichen Schnittpunkte
                T[i][j] = T[i][j] ∪ { A∈N | A↦BC und
                                      B∈T[i][k] und
                                      C∈T[i+k][j-k]}
            }
        }
    }
    if(S∈T[1][n]){                                  Laufzeit O(n^3)
        true;        x∈L(G)
    }else{
        false;       x∉L(G)
    }
```

Beispiel: Die Sprache $L = \{ a^n b^n c^m \mid n, m \geq 1 \}$ ist kontextfrei,

da $S \to AB$, $A \to ab$, $A \to aAb$, $B \to cB$, $B \to c$

diese Sprache erzeugt:

Umform in Chomsky Normalform:

$S \to AB$, $A \to CD$, $A \to CF$ (neues Nichtterminal), $B \to c$, $B \to EB$, $C \to a$,
$D \to b$, $E \to c$, $F \to AD$

Sei nun $x = aaabbbcc \in L(G)$?

	a	a	a	b	b	b	c	c
1	C	C	C	D	D	D	E,B	E,B
2	∅	∅	A	∅	∅	∅	B	
3	∅	∅	F	∅	∅	∅		
4	∅	A	∅	∅	∅			
5	∅	F	∅	∅				
6	A	∅	∅					
7	S	∅						
8	S							

Positionen

Längen

Üben!

Da ist ein S

Baum

$x \in L(G)$

3.4 Kellerautomaten

Wissen: Es gibt keine endliche Automaten für die Sprache $a^n b^n$

Frage: Wie muß man das Berechnungsmodell des endlichen Automaten „aufbohren", damit es „klappt"

Idee: Wir brauchen einen Keller.

Skizze: