Vorlesung Informatik III – Theoretische Informatik

Prof. Dr. Peter Thiemann
LaTeX von Ralph Lesch*

WS 2014/15
2015-10-21 – 2016-08-08

*ralph.lesch@neptun.uni-freiburg.de

Inhaltsverzeichnis

1 Vorspann: Sprachen **4**

2 Turing und Church **8**
2.1 Turingmaschine (informell) 8
2.2 Formalisierung der Turing-Maschine (TM) 10
2.3 Techniken zur TM Programmierung 14
2.4 Registermaschinen 17
2.5 Simulation 19
2.6 Das Gesetz von Church-Turing (Churchsche These) 21

3 Reguläre Sprachen und endliche Automaten **22**
3.1 Endliche Automaten 22
3.2 Pumping Lemma (PL) für reguläre Sprachen 29
3.3 nichtdeterministischer endlicher Automat (NEA) 31
3.4 Abschlusseigenschaften 34
3.5 Reguläre Ausdrücke 36
3.6 Entscheidungsprobleme 40

4 Grammatiken und kontextfreie Sprachen **43**
4.1 Kontextfreie Sprachen 45
4.2 Die Chomsky Normalform für Menge der kontextfreien Grammatiken (CFG) . 48
4.3 Pumping Lemma für CFL 50
4.4 Entscheidungsprobleme für CFL 52
4.5 Abschlusseingenschaften für CFL 54

5 Kellerautomaten (PDA) **57**

6 Berechenbarkeit **64**
6.1 Typ-0 und Typ-1 Sprachen 64
6.2 Universelle TM und das Halteproblem 68
6.3 Eigenschaften von entscheidbaren und semi-entscheidbaren Sprachen 72
6.4 Weitere unentscheidbare Probleme 72

7 Komplexitätstheorie **77**
7.1 Komplexitätsklassen und P/NP 77
7.2 Weitere *NP*-vollständige Probleme 80

8 Rekursive Funktionen **82**

Liste der Definitionen **86**

Liste der Sätze **87**

Abbildungsverzeichnis **89**

Abkürzungsverzeichnis **90**

Anmerkungsverzeichnis **91**

1 Vorspann: Sprachen

Vorlesung: 23.10.15

Zeichen, Symbole: Buchstaben, Ziffern
hier: abstrakte Zeichen

Def. 1.1: Ein Alphabet Σ ist eine endl. Menge von Zeichen.

Aus Zeichen $\rightarrow$ Wörter durch Hintereinanderschreiben. $\oplus$

Bsp.: $\Sigma = \{a, \ldots, z\}$
Worte: rambo (5 Zeichen), ist, hungrig, ϵ (0 Zeichen) = leeres Wort
Wörter verketten = konkatenieren

Bsp.: rambo·ist·hungrig
„·“ ist Konkat.-Operator

Wörter potenzieren: $(\text{la})^3 = \text{la} \cdot \text{la} \cdot \text{la} = \text{lalala}$

$(\text{rambo})^0 = \epsilon$

Def. 1.2: Ein Wort w über Σ ist eine endliche Folge von Zeichen $w = a_1 a_2 \ldots a_n$ mit $n \in \mathbb{N}$ und $a_i \in \Sigma$ $(1 \leq i \leq n)$.
Schreibe ϵ falls $n = 0$
$|w| = n$ ist die Länge des Wortes w.
Σ^* ist die Menge aller Wörter über Σ. $\oplus$

Def. 1.3 (Konkatenation von Wörtern)**:**

$$\text{Sei } u = a_1 \ldots a_n \in \Sigma^*$$
$$v = b_1 \ldots b_m \in \Sigma^*$$

dann ist $u \cdot v = c_1 \cdots c_{n+m}$, $c_i \in \Sigma$

$$c_i = \begin{cases} a_i & 1 \leq i \leq n \\ b_{i-n} & n+1 \leq i \leq n+m \end{cases}$$

Eigenschaften von „·“:

- assoziativ
- ϵ ist neutrales Element

Die Potenz v^k , $v \in \Sigma^*, k \in \mathbb{N}^*$ ist def. durch

$$v^0 = \epsilon, \; v^{k+1} = v \cdot v^k$$

$\oplus$

Eine Sprache ist Menge von Wörtern über $\Sigma = \{a, \ldots, z, \ä, \ö, \ü\}$

$$\begin{aligned} L_{\text{obst}} &= \{\text{banane,aprikose,orange,}\ldots\} \\ L_{\text{farbe}} &= \{\text{rot,gelb,grün}\} \\ L_{\text{krach}} &= \{\text{ra} \cdot (\text{ta})^n \cdot \text{mm} \mid n \in \mathbb{N}\} \\ L &= \{\} \quad \text{leere Sprache} \\ L &= \Sigma^* \end{aligned}$$

Def. 1.4: Eine Sprache über Σ ist Menge $L \subseteq \Sigma^*$.

$$L_{\text{lala}} = \{(\text{la})^n \mid n \in \mathbb{N}\} \ni \Sigma$$ ⊕

sämtliche Mengenoperationen sind auch Sprachoperationen, insbesondere:

$L_1 \cap L_2$	,	$L_1 \cup L_2$	,	$\Sigma^* \setminus L$
Schnitt		Verein.		Komplement

$\rightsquigarrow$ Weitere Operationen auf Sprachen: Konkatenation

Bsp.:

$$\begin{aligned} L_{\text{farbe}} \cdot L_{\text{obst}} \subseteq \{&\text{rotbanane, rotaprikose, gelbbanane, gelbaprikose,} \\ &\text{gelborange, grünbanane, grünaprikose, grünorange}\} \\ L_{\text{farbe}} \cdot \{\epsilon\} \cdot L_{\text{obst}} \subseteq \{&\text{rot}\mathbf{e}\text{banane, rot}\mathbf{e}\text{aprikose, rot}\mathbf{e}\text{orange, gelb}\mathbf{e}\text{banane, } \ldots\} \end{aligned}$$

Def. 1.5 (Konkatenation von Sprachen)**:** Sei $U, V \subseteq \Sigma^*$ dann ist

$$U \cdot V = \{uv \mid u \in U, v \in V\}$$ ⊕

Potenzieren

$$\begin{aligned} L^2_{\text{farbe}} &= L_{\text{farbe}} \cdot L_{\text{farbe}} \\ &= \{\text{rotorot, rotgelb, rotgrün, gelbrot, gelbgelb, } \ldots\} \\ L^0 = \{\epsilon\} \qquad &\{\epsilon\} \cdot L = \{\epsilon \cdot w \mid w \in L\} = L \end{aligned}$$

Def. 1.6 (Potenzierung von Sprachen)**:** Sei $U \subseteq \Sigma^*$

$$U^0 = \{\epsilon\} \qquad\qquad U^{n+1} = U \cdot U^n$$

⊕

Gegeben L, sind sämtliche Kombinationen von Worten aus L gesucht.

Bsp.: $\{(la)^n \mid n \in \mathbb{N}\} \cdot \{la\}^*$

Def. 1.7 (Stern-Operator, Abschluss, Kleene-Stern)**:** Sei $U \in \Sigma^*$ dann ist

$$U^* = \bigcup_{n\in\mathbb{N}} U^n \quad [\ni \epsilon] \qquad \text{Leeres Wort ist darin } n = 0$$
$$U^+ = \bigcup_{n\geq 1} U^n \qquad \text{wenn } \epsilon \in U \Rightarrow \text{auch in } U^* \qquad \oplus$$

Bemerkung: $\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n$ ebenso $\Sigma^+ = \bigcup_{n\geq 1} \Sigma^n$

Def. 1.8 (Alternative Definition von Σ^*)**:**
Die Menge Σ^* ist kleinste Menge, so dass

(1) $\epsilon \in \Sigma^*$

(2) $a \in \Sigma\ , w \in \Sigma^* \ \Rightarrow aw \in \Sigma^*$ $\oplus$

Die Definitionen 1.8 und 1.2 sind gleichwertig.

Beweis:

- „Def. 1.2“ $\Rightarrow$ „Def. 1.8“:
  $\forall n \in \mathbb{N}\ w = a_1 \dots a_n \quad , a_i \in \Sigma$

  Zeige $w \in \Sigma^*$ (gemäß Def. 1.8)

  I.A.: $\underline{n = 0} \Rightarrow w = \epsilon \in \Sigma^*$ (Def. 1.8)

  $\underline{n \to n+1}\ w = a_1 \dots a_{n+1}$ nach Def. 1.2 $w' = \overbrace{a_2 \dots a_{n+1}}^{n \text{ Buchstaben}} \in \Sigma^*$
  $\to$ Nach Def. 1.8 (2) $a_1 a_2 \dots a_{n+1} \in \Sigma^*$ (Def. 1.8)
- „Def. 1.8“ $\Rightarrow$ „Def. 1.2“:
  Induktion über $w \in \Sigma^*$ (1.8) Zeige: für jedes $w \in \Sigma$ gibt es ein $n \in \mathbb{N}$ und $a_1, \dots, a_n \in \Sigma$, so dass $w = a_1 \dots a_n$.
  - $\epsilon \in \Sigma^*$ Wähle $n = 0$ dann $w = \epsilon$.
  - $aw' :\ a \in \Sigma,\ w' \in \Sigma^*$ (1.8)
    Nach Induktionsbehauptung $w' \in \Sigma^*$ (Def. 1.2)
    Also $\exists n : w' = a_1 \dots a_n \in \Sigma^*$ (Def. 1.2)
    Wähle (als neues n) $m = n + 1$ und $b_1, \dots b_{n+1} \in \Sigma$ mit $b_1 = a$ und $b_{i+1} = a_i$ für $1 \leq i \leq n$.
    Dann ist $w = b_1 b_2 \dots b_{n+1} \in \Sigma^*$ (1.2) $\square$

Alternative Definition von Konkatenation:

$$\epsilon \cdot v = v$$
$$(aw) \cdot v = a(w \cdot v)$$

Bsp.: Für Eigenschaft von Sprachen:

$$U^* = \{\epsilon\} \cup U \cdot U^*$$

Beweis durch Kalkulation: $U^* = \{\epsilon\} \cup U \cdot U^* \subseteq U^* \cup U \cdot \bigcup_n U^n = U^* \cup \bigcup_n U^{n+1} \subseteq U^* \cup U^* = U^*$

Elementarer Beweis:

Zeige (1) $U^* \subseteq \{\epsilon\} \cup U \cdot U^*$
Sei $w \in U^* = \bigcup_{n \in \mathbb{N}} U^n$
$\curvearrowright \exists n : w \subset U^n$

- Falls $n = 0 : w = \epsilon \subset \{\epsilon\} \cup U \cdot U^*$
- Falls $n = n' + 1 : w \in U \cdot U^n \subseteq U \cdot U^* \cup \{\epsilon\}$

Zeige (2) $\{\epsilon\} \cup U \cdot U^* \subseteq U^*$
Sei $w \in \{\epsilon\} \cup U \cdot U^*$

- Falls $w = \epsilon : w = \epsilon \in U^*$
- Falls $w \in U \cdot U^n : \exists n \; w \in U \cdot U^n = U^{n+1} \subseteq U^*$

2 Turing und Church

1930er Jahre
Suche nach formalem Modell für maschinelle Berechenbarkeit

Alaen Turing: (1912-1954) Turingmaschine 1936

Church: Lambdakalkül 1936

Kleene Sturgis: partielle rekursive Funktionen

Chomsky: Typ-0-Grammatiken 1956

Alan Turing:
- Informatik, Logik
- Kryptographie (Enigma Entschlüsselung, Sprachverschlüsselung)
- KI (Turing-Test)

außerdem: Turing-Award

2.1 **Turingmaschine** (informell)

Ein primitives Rechenmodell:

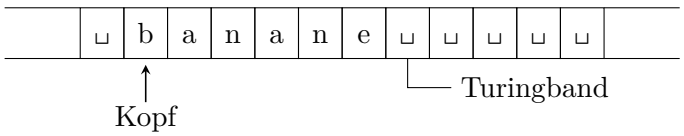


Abb. 1: Turingband
q = Zustand

Turingband
- unendliches Band
- Jedes Feld enthält ein Symbol aus einem Bandalphabet Γ
- uninitialisiert: Blank ␣ist ein spezielles Symbol $␣ \in \Gamma$

Kopf
- zeigt immer auf ein Feld
- nur am Kopf kann die TM ein Zeichen lesen und schreiben
- kann nach rechts /links bewegt werden
- kann verändert werden

Zustand
- kann verändert werden
- kann gelesen werden
- es gibt nur endlich viele Zustände

Turingtabelle

q	a	q'	a'	d

$\sim$ Programm $\sim$ Transitionsfunktion
$\rightarrow$ Wenn TM in Zustand q und Kopf liest gerade Symbol $a \in \Gamma$ dann wechsle in Zustand q'. Schreibe a' (über altes a) und bewege den Kopf gemäß $d \in \{L, R, N\}$

Bsp.:

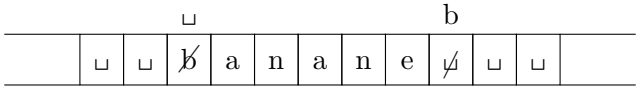


Abb. 2: Bsp.: Turingmaschine

q_1	b	q_2	$\sqcup$	R
q_1	$x \pm b$	q_1	x	N
q_2	$\sqcup$	q_3	b	L
q_2	$x + \sqcup$	q_2	x	R
q_3	$x + \sqcup$	q_3	x	L
q_3	$\sqcup$	q_4	$\sqcup$	L
q_4	x	q_4	x	N

$\rightarrow$ Endzustand

Vorlesung: 28.10.15

Was kann die TM ausrechnen?

1. Die TM kann eine Sprache $L \subseteq \Sigma^*$ erkennen.
   - Wörter müssen auf Band repräsentierbar sein $\Sigma \subseteq \Gamma \setminus \{\sqcup\}$

   Ein Wort w wird von einer TM erkannt, wenn
   - zu Beginn steht nur w auf dem Band, alle anderen Zellen $= \sqcup$
   - Kopf auf erstem Zeichen von w
   - Zustand ist Startzustand q_0
   - Abarbeitung der Turingtabelle (TT)
   - Falls TM nicht terminert: $w \notin L$
   - Falls TM terminert betrachte den errechneten Zustand q.
     Falls $q \in F$ (akzeptierter Zustand), dann $w \in L$, anderenfalls $w \notin L$

   Bsp.:

   $\Sigma = \{0, 1\}$
   $L = \{w \in \Sigma^* \mid w \text{ ist Palindrom}\}$
   $Q = \{q_0, q_1, q_r^0, q_r^1, q_r^{0\prime}, q_r^{1\prime}, q_l^0, q_l^1\} \quad F = \{q_1\}$

q_0	$\sqcup$	q_1	$\sqcup$	N	$q_1 \times q_1 \times N$
q_0	0	q_r^0	$\sqcup$	R	
q_0	1	q_r^1	$\sqcup$	R	
q_r^0	$\sqcup$	q_1	$\sqcup$	N	
q_r^0	0	$q_r^{0'}$	0	R	
q_r^0	1	$q_r^{0'}$	1	R	
$q_r^{0'}$	$\sqcup$	q_l^0	$\sqcup$	L	$q_l \rightarrow$ prüfe 0, fahre zum linken Rand und weiter mit q_0
$q_r^{0'}$	0	$q_r^{0'}$	0	R	} Rechtslauf
$q_r^{0'}$	1	$q_r^{0'}$	1	R	} Rechtslauf

Alternative 1:
TM hält bei jeder Eingabe an.

q_l^0	$\sqcup$	q_l^0	$\sqcup$	N	$\leftarrow$ Halt
	0	q_l	$\sqcup$	L	
	1	q_l^0	1	N	$\leftarrow$ Halt

Alternative 2:
TM hält nur bei Palindrom an.

q_l^0	$\sqcup$	q_l^0	1	N
q_l^0	0	q_l	$\sqcup$	L
q_l^0	1	q_l^0	$\sqcup$	N

2. Die TM errechnet Funktion $f : \Sigma^* \longrightarrow \Sigma^*$
   Die Berechnung von $f(w),\ w \in \Sigma^*$
   - w auf leeres Band
   - Kopf auf erstes Zeichen, Standardzustand q_0
   - Abarbeitung der TT
   - Falls terminiert, dann Kopf zuerst auf erste Symbol von $v \in \Sigma^*$
     Dann $f(w) = v$

Schreibe	$A \longrightarrow B$	totale Funktion von A nach B
	$A \dashrightarrow B$	partielle Funktion von A nach B

Bsp. 2.1: $\Sigma = \{0, 1\}$
Gesucht die TM, die die Nachfolgefunktion auf natürliche Zahlen in Binärdarstellung berechnet.
Ausnahme: niederwertigste Stelle von der Zahl.

$\xrightarrow{\text{Start}}$ q_0	$\sqcup$	q_2	1	L	
q_0	0	q_1	1	L	
q_1	1	q_0	0	R	
q_1	$\sqcup$	q_1	$\sqcup$	N	$\leftarrow$Halt
q_1	0	q_2	0	L	} Linksmaschine
q_1	1	q_2	1	L	} Linksmaschine

2.2 Formalisierung der TM

Def. 2.1: Eine TM ist ein 7-Tupel

$$\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

- Q ist endliche Menge der Zustände
- Σ ist endliches Alphabet
- $\Gamma \supsetneq \Sigma$ ist endliches Bandalphabet
- $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{R, L, N\}$
- $q_0 \in Q$ Standardzustand
- $\sqcup \in \Gamma \setminus \Sigma$ das Blank
- $F \subseteq Q$ Menge der akzeptierenden Zustände

$\oplus$

Im Folgenden sei $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ eine TM.

Def. 2.2: Eine Konfiguration einer TM ist ein Tupel

$$(v, q, w) \in \text{Konf}(\mathcal{A}) = \Gamma^* \times Q \times \Gamma^+$$

$\oplus$

- v linke Bandhälfte,
- q Zustand,
- w rechte Bandhälfte,
- Kopfpos auf erstem Symbol von w

Abkürzend $vqw \in \text{Konf}(\mathcal{A})$ steht für Band:

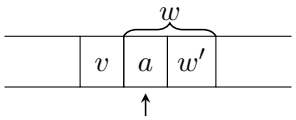


Abb. 3: vqw-Band

Forts.: Die Startkonfiguration bei Eingabe w ist: $\begin{cases} q_0\, w & , w \neq \epsilon \\ q_0\, \sqcup & , w = \epsilon \end{cases}$
Eine Haltekonfiguration hat folgende Form: $vqaw$, so dass $\delta(q, a) = (q, a, N)$

Def. 2.3: Die Rechenschrittrelation

$$\vdash \subseteq \text{Konf}(\mathcal{A}) \times \text{Konf}(A)$$

ist definiert durch

$$\begin{array}{rll} 1. & vqaw \vdash vq'a'w & \text{falls } \delta(q,a) = (q', a', N) \\ 2. & vqaw \vdash va'q'w & \text{falls } \delta(q,a) = (q', a', R), w \neq \epsilon \\ & va'q'\sqcup & \text{———— " ————}, w = \epsilon \\ 3. & qaw \vdash q'\sqcup a'w & \delta(q,a) = (q', a', L) \\ & vbqaw \vdash vq'ba'w & \text{———— " ———— } b \in \Gamma \end{array}$$

$\vdash$ Einzelschritt, gesuchte Relation für endlich viele Schritte
$\vdash^* \subseteq \text{Konf}(\mathcal{A}) \times \text{Konf}(\mathcal{A})$ die reflexive transitive Hülle von $\vdash$ ⊕

Vorlesung: 30.10.15

$\vdash$ Relation auf $\text{Konf}(\mathcal{A}) \hat{=}$ Berechnungsschritt
$\vdash^*$ reflexiv, transitiver Abschluss $\hat{=}$ endl. viele Berechnungsschritte

Exkurs: binäre Relationen
A Menge, jedes $R \subseteq A \times A$ ist binäre Relation auf A

Bsp.:

$$\begin{array}{rl} \varnothing & \text{leere Relation} \\ A \times A & \text{volle Relation} \\ \mathbb{1}_A = \{(x,x) \mid x \in A\} & \text{Gleichheit auf } A \\ \leq \subseteq \mathbb{N} \times \mathbb{N} \quad < \subseteq \mathbb{N} \times \mathbb{N} & \mid \subseteq \mathbb{N} \times \mathbb{N} \end{array}$$

Mengenoperationen auf Rel. ok

$$\begin{aligned} R_1 &= < \cup \mathbb{1}_{\mathbb{N}} = \leq \\ (x,y) \in R_1 &\Leftrightarrow x < y \ \vee \ x = y \\ R_2 &= < \cap \mathbb{1}_{\mathbb{N}} = \varnothing \\ F_3 &= \{(x,y) \mid y = 3x\} \subseteq \mathbb{N} \times \mathbb{N} \end{aligned}$$

Def. 2.4.1: $R, S \subseteq A \times A$ Relation
Die Verkettung (Komposition) von R und S

$$R \circ S = \{(x,y) \in A \times A \mid \exists z \in A : (x,z) \in R, (z,y) \in S)\}$$

⊕

Bsp.:

$$\underbrace{< \circ \mathbb{1}_{\mathbb{N}}}_{=S_1} = <$$

$$\begin{aligned}
& (x,y) \in S_1 \\
\Leftrightarrow\ & \exists z : x < z \ , z = y \\
\Leftrightarrow\ & x < y \\
& F_3 \circ F_3 \\
& (x,y) \in F_3 \circ F_3 \\
\Leftrightarrow\ & \exists z : (x,z) \in F_3 \ , \ (z,y) \in F_3 \\
\Leftrightarrow\ & \exists z : z = 3x \ \wedge \ y = 3z \\
\Leftrightarrow\ & y = 9x
\end{aligned}$$

Def. 2.4.2: $R \subseteq A \times A$

a) R ist <u>reflexiv</u>, falls $\mathbb{1}_A \subseteq R$

b) R ist <u>transitiv</u>, falls $R \circ R \subseteq R$

$\oplus$

Bsp.:			
	$\mathbb{1}_A$	refl.	trans.
	$\leq_{\mathbb{N}}$	refl.	trans.
	$\varnothing$	nicht refl.	trans.
	$\mid$	refl.	trans.

Def. 2.4.3: $R \subseteq A \times A$ Relation
Der reflexiv transitive Abschluss (Hülle) von R ist

$$R^* = \bigcup_{n \in \mathbb{N}} R^n$$

mit

$$R^0 = \mathbb{1}_A \quad , \ R^{n+1} = R \circ R^n$$

$\oplus$

Bem: $R^* = 1 \cup R \circ R^*$ gilt auch.

Es gilt: Für bel. R:

- R^* refl. : $\mathbb{1} = R^0 \subseteq \bigcup_{n \in \mathbb{N}} R^n = R^*$

– R^* trans.

$$\begin{aligned} & R^* \circ R^* \subseteq R^* \\ & R^* \circ \bigcup_n R^n \\ \forall n : & R^* \circ R^n \subseteq R^* \\ \Rightarrow & R^* \circ \bigcup_n R^n \subseteq R^* \end{aligned}$$

$\vdash^2 = \vdash \circ \vdash$ zwei Schritte
$\vdash^*$ endl. viele Schritte

Def. 2.5 (Die von TM $\mathcal{A}$ erkannte Sprache)**:**

$$\begin{aligned} L(\mathcal{A}) = \{w \in \Sigma^* \mid & \ q_0 w \vdash^* uqv \\ & uqv \text{ Haltekonfiguration} \\ & q \in F\} \end{aligned}$$

⊕

Beachte: $w \notin L(\mathcal{A}) \swarrow\!\!\searrow \begin{matrix} \mathcal{A} \text{ kann anhalten} \\ \mathcal{A} \text{ kann nicht terminieren} \end{matrix}$

Def. 2.6 (Die von TM $\mathcal{A}$ berechnete Funktion)**:**

$$\begin{array}{lll} & f_{\mathcal{A}} : \Sigma^* - \to \Sigma^* & \\ & f_{\mathcal{A}}(w) = v & \\ \text{falls} & q_0 w \vdash^* uqv' & \text{Haltekonf.} \\ \text{und} & v = \text{out}(v') & \end{array}$$

$$\begin{array}{lll} \text{out} : \Gamma^* \to \Sigma^* & & \\ \text{out}(\epsilon) & = \epsilon & \\ \text{out}(au) & = a \cdot \text{out}(u) & a \in \Sigma \\ \text{out}(bu) & = \epsilon & b \in \Gamma \setminus \Sigma \end{array}$$

⊕

Beachte: Falls $q_0 w$ nicht terminiert, dann ist $f_{\mathcal{A}}(w)$ nicht definiert.

Eine TM $\mathcal{A}$ terminiert nicht bei Eingabe w, falls für alle $uq'v$, so dass $q_0 w \vdash^* uq'v$ $uq'v$ ist keine Haltekonfiguration.

2.3 Techniken zur TM Programmierung

– Endlicher Speicher
Zum Abspeichern eines Elements aus endl. Menge A verwende

$$Q' = Q \times A$$

– Mehrspurmachinen

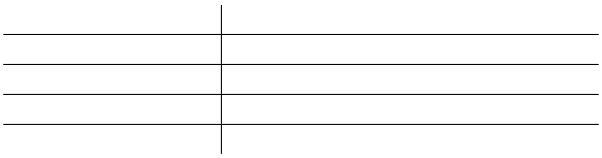


Abb. 4: Mehrspurmachine

Eine k-Spur TM kann gleichzeitig $k \geq 1$ Symbole $\leftarrow \Gamma$ unter dem Kopf lesen.
Kann durch Standard TM simuliert werden:

$$\Gamma' = \Sigma \dot{\cup} \Gamma^k \text{ mit } \sqcup' = \sqcup^k$$

... vereinfacht die Programmierung

Bsp.: Schulalg. für binäre Addition, Multiplikation

– Mehrbandmachinenen
Eine k-Band TM besitzt $k \geq 1$ Bänder und k Köpfe, die bei jedem Schritt lesen, schreiben und sich unabhängig voneinander bewegen.

$$\delta_K : Q \times \Gamma^k \to Q \times \Gamma^k \times \{R, L, N\}^k$$

– keine herkömmliche TM (für $k > 1$)

– kann durch $2k + 1$ Spur TM simuliert werden:

Spur		
1	——— Band 1 ———	
2	# Kopf für Band 1	
3	——— Band 2 ———	
4	# Kopf	
	⋮	
$2k$	... # Kopf k	
$2k+1$	#	##
	linker Rand	rechter Rand

Vorlesung: 04.11.15

Satz 2.1: Eine k-Band TM kann durch eine 1-Band TM simuliert werden. $M = (Q \dots)$

Beweis: Zeige: ein Schritt der k-Band TM wird durch endlich viele Schritte auf einer 1-Band TM simuliert.

1. Schritt: Kodierung der Konfiguration der k-Band TM
Definiere M' als TM mit $2k + 1$ Spuren und $\Gamma' = \Gamma \cup \{\#\}$

  – Die Spuren $1, 3, \dots, 2k-1$ enthalten das entspr. Band von M: Band $i \leftrightarrow$ Spur $2i-1$

- Die Spuren $2, 4, \ldots, 2k$ sind leer bis auf eine Marke #, die auf Spur $2i$ die Position des Kopfes auf Band i markiert
- Spur $2k+1$ enthält
  # Marke für linken Rand
  ## Marke für rechten Rand
  Zwischen den beiden Marken befindet sich der bearbeitete Bereich des Bands. D.h. die TM arbeitet zwischen der linken und rechten Marke und schiebt die Marken bei Bedarf weiter.

2. Schritt: Herstellen der Start-Konfiguration.
Annahme: Eingabe für M auf Band 1
Jetzt Eingabe (für M') $w = a_1 \ldots a_n$

a) Kopiere w auf Spur 1

b) Kopf setzen auf Spur $2, \ldots, 2k$ an die Position des ersten Symbols von w

c) auf Spur $2k+1$: #␣##

$2k+1$	#␣##
$2k$	#
$2k-1$	␣
⋮	
4	#
3	␣
2	#
Spur 1	$a_1 a_2 \ldots a_2$

Springe nach $\mathrm{Sim}(q_0)$, der Zustand in M', an dem die Simulation des Zustands q aus M beginnt.

3. Simulation eines Rechnerschritts im Zustand $\mathrm{Sim}(q)$:
Kopf auf linker Begrenzung, d.h. linker # auf Spur $2k+1$

- Durchlauf bis rechter Rand, sammle dabei Symbole unter den Köpfen, speichern in endl. Zustand $\overrightarrow{\gamma} \in \Gamma^k$
- Berechne $\delta(q, \overrightarrow{\gamma}) = (q', \overrightarrow{\gamma'}, \overrightarrow{d})$
  neuer Zustand, für jeden Kopf ein neues Symbol $\overrightarrow{\gamma'}$ und Richtung $\overrightarrow{d}$.
- Rücklauf nach links, dabei Schreiben um $\overrightarrow{\gamma}'$ und Versetzen der Köpfe gemäß $\overrightarrow{d}$.

Falls eine Kopfbewegung den Rand auf Spur $2k+1$ überschreitet, dann verschiebe Randmarke entsprechend.

Beim Rücklauf: Test auf Haltekonfiguration der k-Band TM.
Falls ja, dann Sprung in Haltekonf. von M'

Weiter im Zustand $\mathrm{Sim}(q')$.

□

Korollar: Beim Erkunden eines Worts der Länge n benötige die k-Band Maschine M $T(n)$ Schritte und $S(n)$ Zellen auf den Bändern.

- M' benötigt $O(S(n))$ Zellen
- M' benötigt $O(S(n \cdot T(n)))$ Schritte $= O(T(n)^2)$

Weitere TM-Booster

- Unbeschränkt großer Speicher

  $\rightarrow$ für jede „Variable“ ein neues Band
- Datenstrukturen

  ↳ ensprechend kodieren.

$\oplus$

2.4 Registermaschinen

Abb. 5: Registermaschine

Register:

- unendlich viele Register
- jedes enthält natürliche Zahl

pc Befehlszähler $\in \mathbb{N}$
Programm = endliche Tabelle von Instruktionen.

Folgende Instruktionen gibt es:

- inc(i) Inkrementiere r_i
- dec(i) Dekrementiere r_i
- if0(i) goto j Falls $r_i = 0$ springe nach Instruktion j

Bsp. 2.2: Addiere r_1 und r_2 Ergebnis in Register r_1.

0: if0(2) goto 4
1: dec(2)
2: inc(1)
3: if0(3) goto 0

Terminert, falls *pc* nicht mehr auf Instruktion zeigt.

Def. 2.7 (Registermaschine (RM))**:** Die RM-Instruktion sind gegeben durch

$$\text{Instr} = \{\text{inc}, \text{dec}\} \times \mathbb{N} \ \cup \ \{\text{if0}\} \times \mathbb{N} \times \mathbb{N}$$

Ein RM-Programm ist eine Abbildung
$P : [0 \dots m] \to \text{Instr}$ ⊕

Def. 2.8 (Semantik der RM bezüglich eines Programms P)**:**

Der Zustandsraum der RM ist

$$Z = \mathbb{N} \times \mathbb{N}^{\mathbb{N}}$$
$$pc \quad \text{Registerinhalte entspr. } \mathbb{N} \to \mathbb{N}$$

Die Semantik eines RM-Befehls:

$$[\![\cdot]\!] : \text{Instr} \to Z \to Z$$
$$[\![\text{inc}(i)]\!](pc, r) = (pc + 1, r[i \mapsto r(i) + 1])$$
$$[\![\text{dec}(i)]\!](pc, r) = (pc + 1, r[i \mapsto r(i) \dot{-} 1])$$
$$[\![\text{if0}(i) \text{ goto } j]\!](pc, r) = \begin{cases} (pc + 1, r) & , r(i) \neq 0 \\ (j, r) & , r(i) = 0 \end{cases}$$

Semantik eines Programms

$$[\![\cdot]\!] : \text{Prog} \to Z - \to Z$$
$$[\![P]\!](pc, r) = \begin{cases} [\![P]\!]\,([\![\text{ins}]\!](pc, r)) & P(pc) = \text{ins} \\ (pc, r) & P(pc) \text{ nicht definiert} \end{cases}$$

⊕

Dabei wird verwendet:

1. Funktionsupdate $f[a \mapsto b]$

$$f : A \to B$$
$$f[a \mapsto b] =: f'$$
$$\text{mit } f'(x) = \begin{cases} b & x = a \\ f(x) & x \neq a \end{cases}$$

2. abgeschnittene Subtraktion

$$m \dot{-} n = \begin{cases} m - n & , m \geq n \\ 0 & , m < n \end{cases}$$

Registermaschine

Vorlesung: 06.11.15

$$P : ([0 \dots m] \to \text{Instr}) = \text{Prog}$$

Die Ein- und Ausgabefunktionen seien wie folgt definiert:

$$\text{in}^{(n)} : \mathbb{N}^n \to Z \qquad \text{in}^{(n)}(\overrightarrow{x}) = (0, r_0[i \to x_i]) \quad \forall n : r_0(n) = 0$$
$$\text{out}^{(m)} : Z \to \mathbb{N}^m \qquad \text{out}^{(m)}(pc, r) = (r(1), \dots, r(m))$$

Die von P berechnete Funktion

$$f_P : \mathbb{N}^n \to \mathbb{N}^m \qquad f_P = \text{out}^{(m)} \circ [[P]] \circ \text{in}^{(n)}$$

2.5 Simulation

Satz 2.2: Jedes RM-Programm kann durch eine TM simuliert werden.

Beweis: Das RM-Programm benutzt nur Register $[0 \dots K]$
Verwende eine $K + 2$-Band TM:

- Band $1 \dots k + 1$: Registerinhalt binär kodiert
- Band 0: Ein-/Ausgabe
- Kodiere pc $[0 \dots m]$ im Zustand der TM

Initialzustand

- Dekodiere Eing. $v_1 \# v_2 \dots \# v_n$
  Kopiere $v_i \to \text{Band}_i$
- Lösche Band 0
- Initialisiere weitere Bänder 0
- Zustand $\curvearrowright \text{Sim}(pc = 0)$

Schritt:

- Falls $P(pc) = \text{Inc}(i)$ wende TM aus Beispiel 2.2 auf Band $i + 1$ an
- Zustand $\curvearrowright \text{Sim}(pc + 1)$
- Falls $P(pc) = \text{Dec}(i)$
  - wende TM für Dec auf Band $i + 1$ an
  - Zustand $\curvearrowright \text{Sim}(pc + 1)$
- Falls $P(pc) = \text{If } O(i) \text{ goto } j$
  - Teste ob Band $i + 1 = 0$:
    Zustand $\curvearrowright \begin{cases} \text{Sim}(pc = j) \\ \text{Sim}(pc + 1) \end{cases}$, Band inf 0

– Falls $P(pc)$ nicht definiert ist
Kopiere Ausgaben von Band $1 - n$ nach Band 0
$\rightarrow$ Haltezustand

□

Satz 2.3: Jede TM kann durch ein RM-Programm simuliert werden.

Beweis: Wir nummerieren:

– die Zustände $Q = \{q_0, \ldots, q_z\}$

– die Elemente des Bandalphabets $\Gamma = \{a_0, \ldots, a_s\}$ wobei $a_0 = \sqcup$ (Blank hat Nummer 0)

Nun können wir Zustände/Elemente des Bandalphabets als Zahlen $[0 \ldots z]/[0 \ldots s]$ interpretieren.

– Register 0 rechter Teil des Bandes

– Register 1 linker Teil des Bandes

– Register Z enthält Zustand

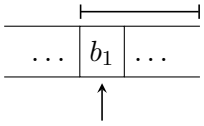


Abb. 6: TM Simulation durch RM

Allgemein wird Konfiguration $c_m \ldots \; c_1 \; q \; b_1 \ldots b_n$ repräsentiert als

$$r_0 = b_1 \cdot (s+1)^0 + b_2 \cdot (s+1)^1 + \cdots + b_n \cdot (s+1)^{n-1}$$
$$r_1 = c_1 \cdot (s+1)^0 + c_2 \cdot (s+1)^1 + \cdots + c_m \cdot (s+1)^{m-1}$$

Initialisierung:

- Eingabe von $b_1, \ldots, b_n$ kodiert als Zahl in $r_0 = b_1 + b_2(s+1) + \cdots + b_n(s+1)^{n-1}$
- $r_1 = 0$
- $r_2 = 0$ Startzustand q_0

Schritt

$r_3 := r_0 \mod (s+1)$	erstes Symbol
$r_0 := r_1 \div (s+1)$	"$(r_0 << 1)$"
$\delta(q, a) = (q', a', d)$	Hält, falls $q = q', a = a', d = N$
r_2, r_3	
$r_2 := q'$	
$r_3 := a'$	

– If $d = N$

$$r_0 := r_3 + (s+1)r_0$$

– else if $d = R$

$$r_1 := r_3 + (s+1)r_1 \quad \text{"}(r_1 >> 1)\text{"}$$

– else if $d = L$

$$\begin{aligned} r_0 &:= r_3 + (s+1)r_0 \\ r_3 &:= r_1 \mod (s+1) \\ r_1 &:= r_1 \div (s+1) \\ r_0 &: \quad r_3 + (s+1)r_0 \end{aligned}$$

□

2.6 Das Gesetz von Church-Turing (Churchsche These)

Satz 2.4: Jede intuitiv berechenbare Funktion ist mit TM (in formalem Sinn) berechenbar.

„Intuitiv berechenbar“ ≡ man kann Algorithmus hinschreiben

– endliche Beschreibung

– jeder Schritt effektiv durchführbar

– klare Vorschrift

Status wie Naturgesetz – nicht beweisbar

→ allgemein anerkannt

→ weitere Versuche Berechenbarkeit zu formulieren, äquivalent zu TMen erwiesen.

3 Reguläre Sprachen und endliche Automaten

Vorlesung: 11.11.15

- TM mächtig
- Wir werden sehen, dass Fragen, wie
  - $w \in L(\text{TM})$? (Wortproblem),
  - $L(\text{TM}) \neq \varnothing$? (Leerheitsproblem)

  für Turingmaschinen im Allgemeinen nicht beantwortbar sind. Daher beschneiden wir die Fähigkeiten der TM und betrachten ein sehr einfaches Maschinenmodell, den endlichen Automaten, wo all diese Fragen entscheidbar sind. D.h. für jede dieser Fragen existiert ein Algorithmus.

3.1 Endliche Automaten

Endliches Band (read-only, jede Zelle enthält ein $a_i \in \Sigma$)

a_0	...	a_n

Abb. 7: Endliches Band

Lesekopf

- ein Zeichen
- nur Bewegung nach rechts
- kann auch hinter Eingabe zeigen: Eingabeende = Maschine stoppt.

Eingabeband

- read-only
- start mit $w \in \Sigma^*$

$\boxed{\text{q}}$ Zustand aus endl. Zustandsmenge

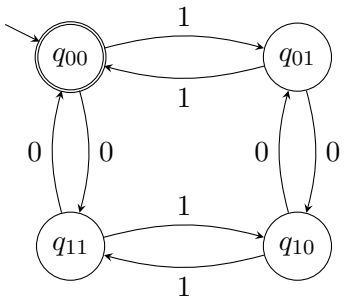
Startzustand: q_0
Ein Teil F davon sind akzeptierende Zustände.

Transitionsfunktion: Im Zustand q beim Lesen von a gehe nach Zustand q'

q	a	q'

Der endliche Automat akzeptiert eine Eingabe, falls er in einem akzeptierenden Zustand stoppt.

Bsp. 3.1: $L = \{w \in \{0,1\}^* \mid w \text{ enthält gerade Anzahl von 0 und gerade Anzahl von 1}\}$



Graphische Darstellung $\hat{=}$ gerichteter Graph mit Knoten Q und markierten Kanten gemäß δ.
$Q = \{q_{00}, q_{01}, q_{10}, q_{11}\}$
q_{00} einziger akzeptierender Zustand ($F = \{q_{00}\}$)

Abb. 8: Automat zu L

	0	1	
q_{00}	q_{10}	q_{01}	gerade Anzahl von 0 und 1 gesehen
q_{01}	q_{11}	q_{00}	gerade —— " ——, ungerade Anzahl von 1 gesehen
q_{10}	q_{00}	q_{11}	ungerade —— " ——, gerade ———— " ————
q_{11}	q_{01}	q_{10}	ungerade —— " ——, ungerade ———— " ————

Def. 3.1 (DEA)**:** Ein deterministischer endlicher Automat (DEA), (DFA $\hat{=}$ deterministic finite automaton) ist ein 5-Tupel

$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q endliche Zustandsmenge
- Σ endl. Alphabet
- $\delta : Q \times \Sigma \to Q$ Transitionsfunktion
- $q_0 \in Q$ Startzustand
- $F \subseteq Q$ akzeptierende Zustande

$\oplus$

Bsp. 3.2: Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA.

- Wenn $F = Q$, dann ist $L(M) = \Sigma^*$.
- Wenn $F = \emptyset$, dann ist $L(M) = \emptyset$.

Def. 3.2: Die Erweiterung von $\delta : Q \times \Sigma \to Q$ auf Worte

$\hat{\delta} : Q \times \Sigma^* \to Q$ ist induktiv def. durch:

$$\hat{\delta}(q, \epsilon) = q \qquad \hat{\delta}(q, aw) = \hat{\delta}(\delta(q, a), w)$$

(Wortende erreicht) (Rest im Folgezustand verarbeiten)

$\oplus$

Def. 3.3: Sei $M = (Q, \Sigma, \delta, q_0, F)$
Die von M erkannte Sprache ist

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

Eine durch einen DEA erkannte Sprache heißt regulär. ⊕

Bsp. 3.3: für reg. Sprachen

$$L = \{w \in \{0,1\}^* \mid w \text{ enthält höchstens eine } 1\} \tag{3.1}$$

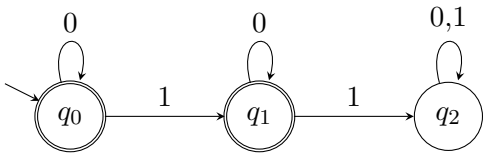


Abb. 9: Endlicher Automat, der die reguläre Sprache L erkennt. Sobald mehr als eine 1 gelesen wurde, wird in den Zustand q_2, eine sogenannte Senke, gewechselt, von der alle ausgehenden Kanten in sich selbst enden.

Sei $A >= 0$ nat. Zahl, $\Sigma = \{0, 1, \ldots, A\}$

$$L = \{a_1 \ldots a_n \mid \exists J \subseteq \{1, \ldots, n\}, \sum_{i \in J} a_i = A\} \subseteq \Sigma^* \tag{3.2}$$

D.h. gegeben eine Liste von Zahlen $\in \Sigma$. Akzeptiere diejenigen Listen, für die eine Teilliste existiert, deren Summe genau A ist.

$$\begin{aligned} Q &= \mathbb{P}\{0, 1, \ldots, A\} \text{ Menge der möglichen Summen} \\ \delta(q, a) &= q \cup \{x \in \{0, \ldots, A\} \mid x - a \in q\} \\ q_0 &= \{0\} \\ F &= \{q \in Q \mid A \in q\} \end{aligned}$$

Beispiel für eine nicht-reguläre Sprache.

$$L = \{0^n 1^n \mid n \in \mathbb{N}\} \tag{3.3}$$

erkennbar durch TM die immer anhält, aber nicht von einem DEA [nicht regulär] akzeptiert werden kann.

Beweis: Angenommen $L = L(M)$ für DEA $M = (Q, \Sigma, q_0, \delta, F)$

Beobachtung: $\exists m \neq n$, sodass $\hat{\delta}(q_0, 0^m) = \hat{\delta}(q_0, 0^n) = q'$ weil Q endlich.

- Falls nun $\hat{\delta}(q', 1^m) \in F$, dann ist auch $\hat{\delta}(q_0, 0^n 1^m) \in F$ und somit $0^n 1^m \in L(M)$ mit $n \neq m$ ↯

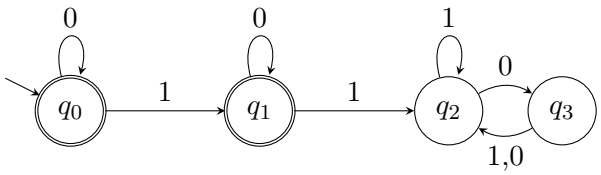
– Falls $\hat{\delta}(q', 1^m) \notin F$, dann gilt auch $\hat{\delta}(q_0, 0^m 1^m) \notin F$ und somit $0^m 1^m \notin L$ ↯

Also kann M nicht existieren! □

Vorlesung: 13.11.15

Beobachtung: Auch ein Automat mit lauter erreichbaren Zuständen muss nicht minimal sein

Bsp. 3.4:



höchstens eine „1“

Abb. 10: Automat zu Bsp. 3.4

Erkennt die gleiche Sprache wie in (3.1), hat nur erreichbare Zustände, aber mehr Zustände als in (3.1).

Beobachtung: q_2 und q_3 verhalten sich gleich in dem Sinn, dass

$$\forall w : \hat{\delta}(q_2, w) \notin F \text{ und } \hat{\delta}(q_3, w) \notin F$$

Def. 3.4: Zwei Zustände $q, p \in Q$ eines DFA sind äquivalent, geschrieben $p \equiv q$, falls $\forall w \in \Sigma^*$, $\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$ ⊕

Lemma 3.1: $\equiv$ ist Äquivalenzrelation

Eine Relation ist genau dann eine Äquivalenzrelation, wenn sie reflexiv, transitiv und symmetrisch ist.

Beweis: $\equiv$ ist offensichtlich reflexiv.

transitiv
symmetrisch } wegen $\Leftrightarrow$

Also q_2, q_3 aus Bsp. 3.4 sind äquivalent. □

Erinnerung: Hauptlemma über Äquivalenzrelationen

$$[q] = \{p \in Q \mid p \equiv q\} \qquad [q] \text{ ist Äquivalenzklasse von } q$$

Äquivalenzklassen sind paarweise disjunkt:

Für alle $p, q \in Q$ gilt entweder $[p] = [q]$ oder $[p] \cap [q] = \emptyset$ (folgt aus Transitivität).

D.h. Q wird in disjunkte Äquivalenzklassen aufgeteilt. Anzahl der Äquivalenzklassen ist der **Index**. ⊕

(3.1):

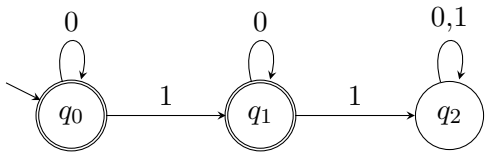


Abb. 11: Automat zu (3.1)

Allgemein gilt für alle $p, q \in Q$:

$$p \equiv q \Rightarrow \forall a \in \Sigma : \ \delta(p,a) \equiv \delta(q,a) \tag{1}$$

Denn

$$\begin{aligned} p \equiv q &\Leftrightarrow \forall w \in \Sigma^* : \hat{\delta}(p,w) \in F \Leftrightarrow \hat{\delta}(q,w) \in F \\ &\Leftrightarrow (p \in F \Leftrightarrow q \in F) \wedge \forall a \in \Sigma : \forall w \in \Sigma^* : \hat{\delta}(p,aw) \in F \Leftrightarrow \hat{\delta}(q,aw) \in F \\ &\Rightarrow \forall a \in \Sigma : \forall w \in \Sigma^* : \hat{\delta}(\delta(p,a),w) \in F \Leftrightarrow \hat{\delta}(\delta(q,a),w) \in F \\ &\Leftrightarrow \forall a \in \Sigma : \delta(p,a) \equiv \delta(q,a) \end{aligned}$$

Also können wir äquivalente Zustände zusammenfassen und Transitionen verschmelzen, wie in der folgenden Definition formalisiert.

Def. 3.5: Der Äquivalenzklassenautomat $M' = (Q', \Sigma, \delta', q_0', F')$ zu M ist bestimmt durch:

$$\begin{aligned} Q' &= \{[q] \mid q \in Q\} & \delta'([q],a) &= [\delta(q,a)] \\ q_0' &= [q_0] & F' &= \{[q] \mid q \in F\} \end{aligned}$$

$\oplus$

Dabei ist $[q] = \{p \in Q \mid p \equiv q\}$

Satz 3.2: Der Äquivalenzklassenautomat ist wohldefiniert und $L(M) = L(M')$.

Beweis:

1. Wohldefiniert: zu zeigen $\delta'([q],a) = [\delta(q,a)]$ ist nicht abhängig von der Wahl des Repräsentanten $q \in [q]$. Das folgt direkt aus (1) gezeigt.
2. $L(M) = L(M')$ zeige für alle $w \in \Sigma^*$ und alle q: $\hat{\delta}(q,w) \in F \Leftrightarrow \hat{\delta}'([q],w) \in F'$
   Induktion über w:
   I.A. $w = \epsilon$: $\hat{\delta}(q,\epsilon) = q \in F \Leftrightarrow \hat{\delta}'([q],\epsilon) = [q] \in F'$ nach Definition.
   I.V.: $\forall w' \in \Sigma$, $\forall q \in Q$, $\hat{\delta}(q,w') \in F \Leftrightarrow \hat{\delta}'([q],w') \in F'$
   I.S.:

$$\begin{aligned} \hat{\delta}(q,aw') \in F &\Longleftrightarrow \hat{\delta}(\delta(q,a),w') \in F \\ &\overset{I.V.}{\Longleftrightarrow} \hat{\delta}'([\delta(q,a)],w') \in F' \\ &\Longleftrightarrow \hat{\delta}'(\delta'([q],a),w') \in F' \\ &\Longleftrightarrow \hat{\delta}'([q],aw') \in F' \end{aligned}$$

   Also für $q = q_0$: $\forall w \in \Sigma^*$, $w \in L(M) \Longleftrightarrow \hat{\delta}(q_0,w) \in F \Leftrightarrow \hat{\delta}'([q_0],w) \in F' \Longleftrightarrow w \in L(M')$

□

Bem: Die Konstruktion von M' kann in $O(|Q||\Sigma| \log |Q|)$ passieren.

Warum ist nun der Äquivalenzklassenautomat minimal?
→ Satz von Myhill-Nerode

Def. 3.6: Eine Äquivalenzrelation $R \subseteq \Sigma^* \times \Sigma^*$ heißt rechtsinvariant, falls

$$(u, v) \in R \Rightarrow \forall w \in \Sigma^*, (u \cdot w, v \cdot w) \in R$$

⊕

Bsp. 3.5: Für einen DEA M definiere

$$R_M = \{(u, v) \mid \hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)\}$$

- ist Äquivalenzrelation
- ist rechtsinvariant
- Anzahl der Äquivalenzklassen(Index von R_M)
  = Anzahl der „nützlichen“ Zustände, die von q_0 erreichbar sind.

Bsp. 3.6: Für eine Sprache $L \subseteq \Sigma^*$ definiere die Nerode Relation

$$R_L = \{(u, v) \mid \forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L\}$$

- ist Äquivalenzrel.
- ist rechtsinvariant. Sei $(u, v) \in R_L$
  Zeige $\forall w \in \Sigma^*$ $(uw, vw) \in R_L$
  Induktion:
  I.A. $w = \epsilon$: $(u\epsilon, v\epsilon) = (u, v) \in R_L$
  I.S. $w = w'a$:
  I.V.: $(uw', vw') \in R_L$

$$\begin{aligned}(uw', vw') \in R_L &\Leftrightarrow \forall z \in \Sigma^*, \quad uw'z \in L \Leftrightarrow vw'z \in L \\ &\Rightarrow \forall a \in \Sigma, z' \in \Sigma^* : uw'az' \in L \Leftrightarrow vw'az' \in L \\ &\Leftrightarrow \ (uw'a, vw'a) \in R_L\end{aligned}$$

$$\left.\begin{array}{ll} L = \{\epsilon\} & [\epsilon] \equiv [\epsilon] \\ w, v \in \Sigma^*,\ w, v \neq \epsilon & [w] = [v] \end{array}\right\} \text{Index} = 2$$

$$L = \varnothing, L = \Sigma^* \qquad \text{Index} = 1$$

Satz 3.3 (Nerode)**:** Die folgende Aussagen sind äquivalent:

1. $L \subseteq \Sigma^*$ wird von DEA akzeptiert.
2. L ist Vereinigung von Äquivalenzklassen einer rechtsinvarianten Äquivalenzrel. mit endl. Index.
3. Die Nerode Relation R_L hat endl. Index

Beweis: (1) $\Rightarrow$ (2): Sei M ein DEA mit $L(M) = \{w \mid \hat{\delta}(q_0, w) \in F\} = \bigcup_{q \in F} [q]_M$ wenn $[q]_M = \{w \mid \hat{\delta}(q_0, w) = q\}$
Äquivalenzklassen und #Klassen$\leq |Q|$ endl.

(2) $\Rightarrow$ (3): Sei R rechtsinv. Äquivrel. mit endl. Index sodass $L = \bigcup R$-Äquivalenzklassen

Sei $(u, v) \in R$. Zeige $(u, v) \in R_L$.
Wegen R rechtsinv. $\forall w \in \Sigma^*$ $(uw, vw) \in R$

$$\begin{aligned}
& \forall w \in \Sigma^* : (uw, vw) \in R \\
\Leftrightarrow \quad & \forall w \in \Sigma^* \quad uw \text{ und } vw \text{ in gleicher } R\text{-Klasse} \\
\Rightarrow \quad & \forall w \in \Sigma^* \quad uw \in L \Leftrightarrow vw \in L \\
\Leftrightarrow \quad & (u, v) \in R_L \\
\Rightarrow \quad & \#\text{Klassen}(R_L) \leq \#\text{Klassen}(R) < \infty
\end{aligned}$$

Vorlesung: 18.11.15

(3) $\Rightarrow$ (1) Gegeben R_L
Konstruiere $\mathcal{A}' = (Q, \Sigma, \delta', q_0', F')$

$$\begin{aligned}
Q' &= \{[w]_{R_L} \mid w \in \Sigma^*\} && \text{endlich, weil index}(R_L) \text{ endl.} \\
\delta'([w], a) &= [wa] && \text{wohldefiniert, da } R_L \text{ rechtsinvariant} \\
q_0' &= [\epsilon] \\
F' &= \{[w] \mid w \in L\}
\end{aligned}$$

Zeige $L(\mathcal{A}') = L$, d.h.

$$\begin{aligned}
\forall w \in \Sigma^* \colon w \in L(\mathcal{A}') &\Leftrightarrow \hat{\delta}([\epsilon], w) \in F' \\
&\overset{???}{\Leftrightarrow} [w] \in F' \\
&\Leftrightarrow w \in L
\end{aligned}$$

Zeige nun noch, dass ??? gilt: $\hat{\delta}([\epsilon], w) = [w]$. Dafür müssen wir wie folgt verallgemeinern um eine funktionierende Induktionsvoraussetzung zu erhalten.

$$\forall w \in \Sigma^* : \forall v \in \Sigma^* : \hat{\delta}'([v], w) = [v \cdot w]$$

Induktion über w

$$\begin{aligned}
\text{I.A.:} \qquad & \epsilon : \hat{\delta}'([v], \epsilon) = [v] = [v \cdot \epsilon] \\
\text{I.S.:} \quad & \hat{\delta}'([v], aw') = \hat{\delta}'(\delta'([v], a), w') \\
& = \hat{\delta}'([v \cdot a], w') \\
& \overset{\text{I.V.}}{=} [va \cdot w'] \\
& = [v \cdot \underbrace{aw'}_{=w}]
\end{aligned}$$

Das gewünschte Ergebnis $\boxed{???}$ ergibt sich für $v = \epsilon$. □

Korollar 3.5: Der im Beweisschritt (3) ⇒ (1) konstruierte Automat $\mathcal{A}'$ ist minimaler Automat für L. ⊕

BEWEIS: L regulär. Sei $\mathcal{A}$ beliebiger DFA mit $L = L(\mathcal{A})$

$\left.\begin{array}{l} \mathcal{A} \text{ induziert } R_{\mathcal{A}} \text{ mit} \\ |Q| \geq \text{index}(R_{\mathcal{A}}) \end{array}\right\} 1 \overset{vgl.}{\Leftrightarrow} 2$

In $2 \Rightarrow 3$: $R_{\mathcal{A}} \subseteq R_L$, $\text{index}(R_{\mathcal{A}}) \geq \text{index}(R_L)$
In $3 \Rightarrow 1$ A' mit $|Q'| = \text{index}(R_L) \leq \text{index}(R_{\mathcal{A}}) \leq |Q|$

Minimalität von A' folgt aus freier Wahl von $\mathcal{A}$ □

Angenommen $\exists\, \mathcal{A}$ mit $L = L(\mathcal{A})$ und $|\mathcal{A}| < |\mathcal{A}'|$
Nach Folgerung gilt aber

$$|\mathcal{A}'| \leq |\mathcal{A}| \; \lightning$$

3.2 Pumping Lemma (PL) für reguläre Sprachen

Suche: Notwendiges Kriterium für Regularität

Bsp.: $L = \{w \in \{0,1\}^* \mid \text{bin}(w) = 0 \mod 3\}$ ist regulär, dabei ist „bin“ die Dekodierung von einem Bitstring in eine natürliche Zahl.

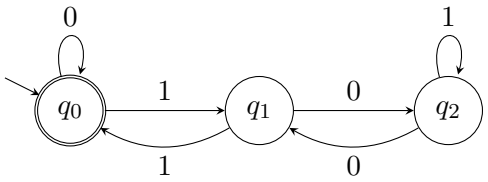


Abb. 12: Automat: Pumping Lemma

$$\begin{aligned}
1\underline{00}1 &\in L \\
\hat{\delta}(q_1, 00) &\overset{\text{Schleife}}{=} q_1 \\
11 &\in L \\
100001 &\in L \\
\forall i \in \mathbb{N} : 1(00)^i 1 &\in L
\end{aligned}$$

Lemma 3.6 (Pumping Lemma)**:** Sei L regulär:

$$\begin{aligned}
&\exists n \in \mathbb{N},\ n > 0 \quad \forall z \in L,\ |z| \geq n : \\
&\exists u, v, w \in \Sigma^* : \\
&z = uvw,\ |uv| \leq n,\ |v| \geq 1 \\
\text{sodass } &\forall i \in \mathbb{N} :\ uv^i w \in L
\end{aligned}$$

Beweis: Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA für L.
Wähle $n = |Q|$ und $z \in L$ mit $|z| \geq n$.

Beim Erkunden von z durchläuft $\mathcal{A}$ $\underbrace{|z| + 1}_{\geq n+1}$ Zustände.

$\rightarrow \exists\, q$, das mehrmals besucht wird.

Wähle das q, dessen zweiter Besuch zuerst passiert.

$$\begin{aligned}
\text{D.h. :} \quad \hat{\delta}(q_0, u) &= q && u \text{ Präfix von } z \\
\exists v : \quad \hat{\delta}(q, v) &= q && uv \text{ Präfix von } z \\
\exists w : \quad \hat{\delta}(q, w) &\in F && uvw = z \\
|v| &\geq 1 \\
|uv| &\leq n && \text{ergibt sich aus Wahl von } q
\end{aligned}$$

$$\begin{aligned}
\text{jetzt:} \quad \hat{\delta}(q_0, uv^i w) &\quad i \in \mathbb{N} \\
&= \hat{\delta}(q, v^i w) \\
&= \hat{\delta}(q, w) \qquad \text{denn } \forall i : \hat{\delta}(q, v^i) = q \\
&\in F
\end{aligned}$$

□

Bsp.: $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ ist nicht regulär.
Sei n die Konstante aus dem PL.

Wähle $z = 0^n 1^n \quad |z| = 2n \geq n$
Laut PL existieren u, v, w, sodass $z = uvw$ mit $|v| \geq 1, |uv| \leq n$ und $\forall i \in \mathbb{N}\ uv^i w \in L$. Nach Wahl von z gilt nun

$$\begin{aligned}
uv &= 0^m && m \leq n \\
v &= 0^k && k \geq 1 \\
\text{Betr. } uv^2 w &= 0^{n+k} 1^n \notin L \\
\Rightarrow L &\text{ nicht regulär}
\end{aligned}$$

$$\underbrace{0\ \ldots\ldots\ 0}_{n}\ \underbrace{1\ \ldots\ldots\ 1}_{n}$$
$$\vdash\text{u}\dashv\text{v}\dashv\!\!-\!\!-\!\!-\text{w}\!\!-\!\!-\!\!-\dashv$$

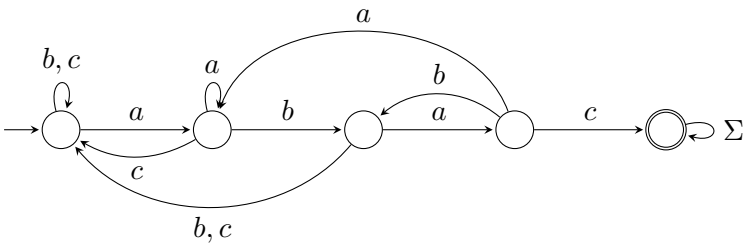


Abb. 13: DFA für L

Bsp.: $L_2 = \{0^p \mid p \text{ ist Primzahl}\}$ ist nicht regulär.

Sei n Konst. aus dem PL, p Primzahl mit $p \geq n$.
Wähle $z = 0^p \in L_2$

$$
\begin{aligned}
&\text{PL} : z = uvw \text{ mit } |uv| \leq n && , |v| \geq 1 \\
&\quad \curvearrowright |z| = p = a + b && , a = |uw| \quad , b = |v| \\
&\quad \curvearrowright |uv^i w| = a + ib && , \text{wähle } i = p + 1 \\
&\quad \curvearrowright |uv^{p+1} w| = a + (p+1)b && = a + pb + b = p + pb \text{ keine Primzahl} \\
&\text{Also } uv^{p+1} w \notin L_2 \\
&\quad \curvearrowright L_2 \text{ nicht regulär.}
\end{aligned}
$$

3.3 nichtdeterministischer endlicher Automat (NEA)

Vorlesung: 25.11.15

Bsp.: Mustererkennung
kommt $\underbrace{\text{ein String}}_{\text{konsistent}}$ in einem anderen vor?

Gegeben: festes Wort w.
Gesucht: Sprache aller Worte, in denen w als Teilwort vorkommt.

$$
\begin{aligned}
&L = \{v \in \Sigma^* \mid \exists u, x \in \Sigma^*, v = uwx\} \\
&\Sigma = \{a, b, c\} \\
&\quad \text{konkretes Beispiel:} \\
&w = abac
\end{aligned}
$$

Abbildung 13 enthält einen DFA für die Sprache L. Beobachtung: nicht-trivial zu konstruieren.

Abbildung 14 enthält einen nicht-deterministischen endlichen Automat für die Sprache L. Idee: Ein Wort w wird akzeptiert, falls es einen mit w markierten Pfad von q_0 zu einen akzeptierenden Zustand gibt.

Abbildung 15 zeigt (einen Ausschnitt) aus dem deterministischen Automaten, der schematisch aus dem NFA in Abb. 14 konstruiert werden kann. Idee: bei Schritt mit Symbol a ist der NFA

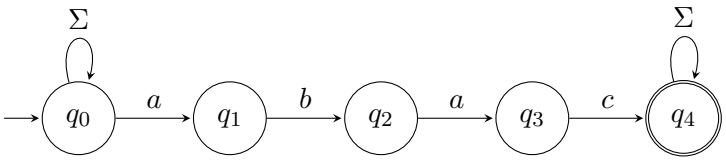


Abb. 14: Bsp.: Mustererkennung

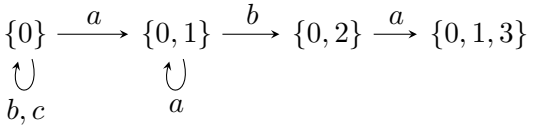


Abb. 15: Potenzmengenkonstruktion auf dem NFA

gleichzeitig in allen Zuständen, die durch a von (der Menge der) aktuellen Zustände erreichbar sind.

Variante: erkenne **Subwort** $w = a_1, \ldots, a_n$

$$L' = \{v \in \Sigma^* \mid \exists x_0, \ldots, x_n \in \Sigma^*, v = x_0 a_1 x_1 a_2 \ldots a_n x_n\}$$

Nicht det. Automat für L' mit $(w = abac)$ ist sehr einfach. Der entsprechende deterministische Automat ist deutlich komplizierter. (selbst)

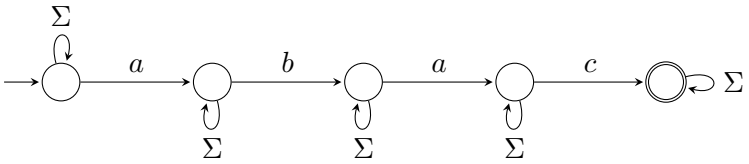


Abb. 16: Nichtdet. Automat für L'

Weiteres Beispiel, bei dem der deterministische Automat beweisbar exponentiell größer ist.

$$L_n = \{w \in \{0, 1^* \mid \text{das } n\text{-letzte Symbol von } w \text{ ist } 1\}$$

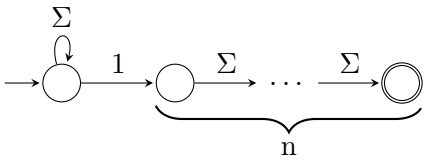


Abb. 17: Nichtdet. Automat für L_n

deterministischer Automat für L_n hat $\sim 2^n$ Zustände.

Def. 3.7: Ein NEA (NFA = nondeterministic finite automaton) $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ mit

- Q endliche Zustandsmenge
- Σ endl. Alphabet
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ Transitionsfunktion
- $q_0 \in Q$ Startzustand
- $F \subseteq Q$ akzeptierende Zustände

$\oplus$

Def. 3.8: Ein Lauf des Automaten $\mathcal{A}$ auf $w = a_1 \dots a_n$ ist eine Folge $q_0 q_1 \dots q_n$ mit $q_i \in Q$, q_0 Startzustand,
$\forall 1 \leq i \leq n,\ q_i \in \delta(q_{i-1}, a_i)$
Ein Lauf heißt akzeptierend, falls $q_n \in F$. $\oplus$

Def. 3.9: $L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists \text{ akzeptierender Lauf von } \mathcal{A} \text{ auf } w\}$ $\oplus$

Satz 3.7 (Rabin)**:** Zu jedem NFA $\mathcal{A}$ mit n Zuständen gibt es einen DFA $\mathcal{A}'$ mit 2^n Zuständen, so dass $L(\mathcal{A}) = L(\mathcal{A}')$.

Beweis (Potenzmengenkonstruktion): Definiere $\mathcal{A}'$ durch

$$\begin{aligned} Q' &= \mathcal{P}(Q) \\ \delta'(q', a) &= \bigcup_{q \in q'} \delta(q, a) \\ q_0' &= \{q_0\} \\ F' &= \{q' \in Q' \mid q' \cap F \neq \varnothing\} \end{aligned}$$

Zeige $L(\mathcal{A}) = L(\mathcal{A}')$

$$\begin{aligned} \text{Es gilt } w \in L(\mathcal{A}') &\Leftrightarrow \hat{\delta}'(q_0', w) \in F' \\ &\Leftrightarrow \hat{\delta}'(q_0', w) \cap F \neq \varnothing \\ w \in L(\mathcal{A}) &\Leftrightarrow \exists \text{ akzeptierender Lauf von } \mathcal{A} \text{ auf } w. \end{aligned}$$

Zeige $\forall w \in \Sigma^*$

$$\begin{aligned} \forall q' \in Q' \quad \hat{\delta}'(q', w) \cap F &\neq \varnothing \\ &\Leftrightarrow \exists \text{ akzeptierender Lauf von } \mathcal{A} \text{ ab } q' \text{ auf } w \\ &\Leftrightarrow \exists \underbrace{p_0 p_1 \dots p_n}_{\text{Lauf}} \in Q \quad p_0 = q',\ p_n \in F,\ n = |w| \end{aligned}$$

Induktion nach w

I.A.

$$\begin{aligned} &\epsilon: \\ &\quad \forall q' \in Q'\ \hat{\delta}(q', \epsilon) \cap F \neq \varnothing \\ \Leftrightarrow &\qquad\quad q' \cap F \neq \varnothing \end{aligned}$$

Wähle einen beliebigen Zustand $p_0 = p_n \in q' \cap F$

I.S.

$$\begin{aligned} &aw' \\ &\quad \hat{\delta}'(q', aw') \cap F \neq \varnothing \\ \Leftrightarrow &\quad \hat{\delta}'(\underbrace{\delta'(q', a)}_{\in Q'}, w') \cap F \neq \varnothing \\ \overset{\text{I.V.}}{\Longleftrightarrow} &\quad \exists \text{Lauf } p_1 \dots p_{n+1} \in Q : p_0 \in \hat{\delta}'(q', a),\ p_{n+1} \in F \end{aligned}$$

Suche $p_0 \in q'$ mit $p_1 \in \delta(p_0, a)$
existiert, denn

$$\begin{aligned} &p_1 \in \delta'(q', a) = \bigcup_{q \in q'} \delta(q, a) \\ \Leftrightarrow &\quad \exists p_0 \in q' : \delta(p_0, a) \ni p_1 \end{aligned}$$

Gesuchter Lauf ist

$$p_0 p_1 \dots p_{n+1}$$

□

Also: Eine Sprache L ist regulär, falls

- $L = L(\mathcal{A})$ für einen DFA
  oder
- $L = L(\mathcal{A})$ für NFA

3.4 Abschlusseigenschaften

Def. 3.10: Eine Menge $\mathcal{L} \subseteq \mathcal{P}(\Sigma^*)$ von Sprachen heißt abgeschlossen unter Operation $f : \mathcal{P}(\Sigma^*)^n \to \mathcal{P}(\Sigma^*)$ falls $\forall L_1, \dots, L_n \in \mathcal{L} : f(L_1, \dots, L_n) \in \mathcal{L}$. ⊕

Satz 3.8: Die Menge REG der regulären Sprachen ist abgeschlossen unter $\cup$ (Vereinigung), $\cap$ (Durchschnitt), $\overline{\ }$ (Komplement), Produkt (Konkatenation), Stern.

Beweis: Sei $\mathcal{A}_i := (Q_i, \Sigma, \delta_i, q_{0i}, F_i) \quad i = 1, 2$ NFAs

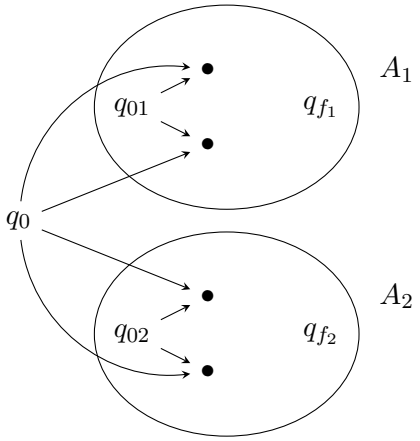


Abb. 18: NFA für Vereinigung

– $\cup$: Def $\mathcal{A}$ durch (vgl. Abb. 18)

$$Q = Q_1 \dot\cup Q_2 \dot\cup \{q_0\}$$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \delta_1(q,a) \cup \delta_2(q,a) & q = q_0 \end{cases}$$

$$F = F_1 \dot\cup F_2 \dot\cup (q_{01} \in F_1 \vee q_{02} \in F_2) \rhd \{q_0\}$$

Zeige $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ (selbst: betrachte die Läufe).

– $\cap$: Annahme: A_1 und $\mathcal{A}_1$ deterministisch.
Def. $\mathcal{A}$ durch den Produktautomaten
DFA :

$$Q = Q_1 \times Q_2$$
$$\delta((q_1,q_2),a) = (\delta_1(q_1,a), \delta_2(q_2,a))$$
$$q_0 = (q_{01}, q_{02})$$
$$F = F_1 \times F_2$$
$$\text{Zeige } L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$$

– Komplement: Ang. $\mathcal{A}_1$ ist DFA.
Ersetze F_1 durch $Q_1 \setminus F_1$.

– Produkt: Seien L_1, L_2 regulär.

Vorlesung: 27.11.15

Zeige $L_1 \cdot L_2$ regulär.

$$
\begin{aligned}
Q &= Q_1 \dot\cup Q_2 \\
\delta(q,a) &= \begin{cases} \delta_1(q,a) & q \in Q_1 \setminus F_1 \\ \delta_1(q,a) \cup \delta_2(q_{02},a) & q \in F_1 \\ \delta_2(q,a) & q \in Q_2 \end{cases} \\
q_0 &= q_{01} \\
F &= F_2 \cup (q_{02} \in F_2) \rhd F_1
\end{aligned}
$$

Zeige $L(\mathcal{A}) = L(\mathcal{A}_1) \cdot L(A_2)$

– Stern

$$
\begin{aligned}
Q &= Q_1 \dot\cup \{q_0\} \\
\delta(q,a) &= \begin{cases} \delta_1(q,a) & q \in Q_1 \setminus F_1 \\ \delta_1(q,a) \cup \delta_1(q_{01},a) & q \in F_1 \\ \delta_1(q_{01},a) & q = q_0 \end{cases} \\
F &= \{q_0\} \cup F_1 \\
\ldots\ L(\mathcal{A}) &= L(\mathcal{A}_1)^*
\end{aligned}
$$

□

3.5 Reguläre Ausdrücke

Def. 3.11: Die Menge $RE(\Sigma)$ der reguläre Ausdrücke über Σ ist induktiv definiert durch:

- $\mathbb{0} \in RE(\Sigma)$
- $\mathbb{1} \in RE(\Sigma)$
- $\forall a \in \Sigma,\ a \in RE(\Sigma)$
- falls $r, s \in RE(\Sigma)$
  - $r + s \in RE(\Sigma)$
  - $r \cdot s \in RE(\Sigma)$
  - $r^* \in RE(\Sigma)$

⊕

Def. 3.12: Die Semantik eines regulären Ausdrucks

$$
\begin{aligned}
[\![\cdot]\!] : RE(\Sigma) &\to \mathcal{P}(\Sigma^*) \text{ ist induktiv def. durch} \\
[\![\mathbb{0}]\!] &= \varnothing \\
[\![\mathbb{1}]\!] &= \{\epsilon\} \\
[\![a]\!] &= \{a\} \quad a \in \Sigma \\
[\![r + s]\!] &= [\![r]\!] \cup [\![s]\!] \\
[\![r \cdot s]\!] &= [\![r]\!] \cdot [\![s]\!] \\
[\![r^*]\!] &= [\![r]\!]^*
\end{aligned}
$$

⊕

$+, \cdot, *$ reguläre Operatoren.

Bsp.: Mustererkennung

- $\Sigma^*\ abac\ \Sigma^*$
  $\hookleftarrow = (a_1 + a_2 + \dots)$ alle $a_i \in \Sigma$ aufgezählt
- n-letztes Symbol $= 1$ $(\Sigma = \{0, 1\})$

$$(0+1)^* 1 \underbrace{(0+1)\dots(0+1)}_{n-1}$$

$\cancel{0^n 1^n} \notin RE(\Sigma)$

- Binärdarstellung modulo $3 = 0$

$$(0 + 1(01^*0)^*1)^*$$

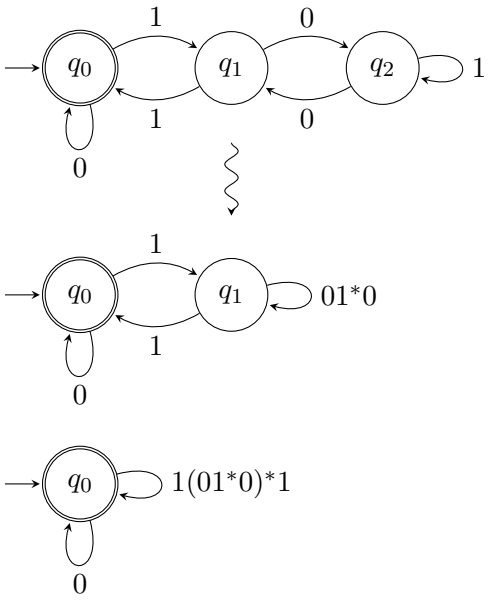


Abb. 19: Informell vom Automaten zum regulären Ausdruck für mod 3

Satz 3.9 (Kleene)**:** L ist regulär
$\Leftrightarrow$ L ist Sprache eines regulären Ausdrucks.

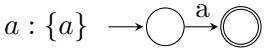
Beweis:

„⇐“ Sei $L = [\![r]\!]$ für $r \in RE(\Sigma)$
Zeige per Induktion über $r : \forall r \in RE(\Sigma) : [\![r]\!]$ regulär.

I.A.:

$\mathbb{0} : \varnothing$ ist reg.

$\mathbb{1} : \{\epsilon\}$ ist reg.

$a : \{a\}$ 

NFA

I.S.:

$$\begin{aligned} r+s: \llbracket r+s \rrbracket &= \llbracket r \rrbracket \cup \llbracket s \rrbracket \quad \text{reg. nach Satz 3.8} \\ r \cdot s: \llbracket r \cdot s \rrbracket &= \llbracket r \rrbracket \cdot \llbracket s \rrbracket \quad \text{——— " ———} \\ r^*: \llbracket r* \rrbracket &= \llbracket r \rrbracket^* \quad \text{——— " ———} \end{aligned}$$

„⇒“ Sei $L = L(\mathcal{A})$ für einen DFA $\mathcal{A} = (\underbrace{Q}_{=\{q_0, q_1, \ldots, q_n\}}, \Sigma, \delta, q_0, F)$.

Def. $L_i = \{w \mid \hat{\delta}(q_i, w) \in F\}$ (Also $L(\mathcal{A}) = L_0$)

$$\begin{aligned} &\text{Betrachte} && \delta(q_i, a) = q_j \\ &\curvearrowright && L_i \supseteq a_i \cdot L_j \\ &\text{Betrachte} && \text{alle Transitionen ab } q_i \\ &\curvearrowright && L_i = N(q_i) + \sum_{0 \le j \le n} A_{ij} \cdot L_j \\ & && A_{ij} = \sum_{a \in \Sigma, \delta(q_i, a) = q_j} a \\ & && N(q_i) = \begin{cases} \mathbb{1} & , q_i \in F \\ \mathbb{0} & , q_i \notin F \end{cases} \end{aligned}$$

Also: $\mathcal{A}$ berechnet die Lösung eines Gleichungssystems

$$L_i = N(q_i) + \sum_j A_{ij} L_j \quad \text{mit } \epsilon \notin A_{ij}$$

Zur Lösung dieses Gleichungssystems verwenden wir das Lemma von Arden:

□

Lemma 3.10 (Arden's Lemma)**:**
Sei $X = A \cdot X + B$ für $\underset{\text{↰ Unbekannte}}{X}, A, B \subseteq \Sigma^*$
dann ist $X = A^* B$ falls $\epsilon \notin A$.

Beweis (Fortsetzung): Verfahren zur Lösung des Gleichungssystems:

Eliminiere sukzessive die Gleichung für L_n (bis nur noch eine Gleichung für L_0 übrig ist)

(1) Falls Gleichung für L_n rekursiv

$$\text{dann hat sie die Form}: \quad L_n = \underbrace{\left(N(q_n) + \sum_{j \neq n} A_{nj} \cdot L_j\right)}_{=:B} + \underbrace{A_{nn}}_{=:A \not\ni \epsilon} \cdot L_n$$

$$\begin{aligned} \text{Nach Ardens Lemma}: \quad L_n &= A_{nn}^* \cdot \left(N(q_n) + \sum_{j \neq n} \underbrace{A_{nj}}_{\not\ni \epsilon} \cdot L_j\right) \\ &= A_{nn}^* \cdot N(q_n) + \sum_{j \neq n} \underbrace{A_{nn}^* \cdot A_{nj}}_{\not\ni \epsilon} \cdot L_j \end{aligned}$$

Gleichungssystem der gleichen Form: linear in den L_j mit Koeffizienten, die nicht ϵ enthalten.

(2) Gleichung für L_n nicht rekursiv:
Setze rechte Seite für L_n in die Gleichungen $L_0, L_1, \ldots, L_{n-1}$ ein. $\longrightarrow$ Gleichungssystem der gleichen Form.

Nach $n+1$ Iterationen erhalten wir ein Gleichungssytem der Form $L_0 = r$. $\oplus$ □

BEWEIS: (Arden's Lemma)

$$\begin{aligned}
\text{Sei} \quad & X = AX + B \text{ mit } \epsilon \notin A. \\
\text{Zeige} \quad & A^*B \subseteq X. \\
& A^*B = (\mathbb{1} + AA^*)B = B + A(A^*B) \quad \checkmark
\end{aligned}$$

Angenommen $A^*B \subsetneq X$, d.h. $\exists w \in X$ mit $w \notin A^*B$, davon sei w das kürzeste.

$$\begin{aligned}
\exists n \geq 1: \quad & X = \underbrace{A^n X}_{\ni w} + \underbrace{A^{n-1}B + \cdots + AB + B}_{\not\ni w} \\
\curvearrowright \quad & w = u_1 \ldots u_n w' \text{ mit } u_1, \ldots, u_n \in A \text{ und } w' \in X \\
\curvearrowright \quad & |w'| < |w| \\
\text{Falls} \quad & w' \in A^*B \curvearrowright w \in A^n A^*B \subseteq A^*B \quad \lightning \\
\text{Also} \quad & w' \notin A^*B \quad \lightning \text{ gegen Minimalität von } w \\
\curvearrowright \quad & X \subseteq A^*B \\
\rightarrow \quad & X = A^*B
\end{aligned}$$

□

Vorlesung: 02.12.15

Bsp.: für Konv. DFA$\rightarrow RE$

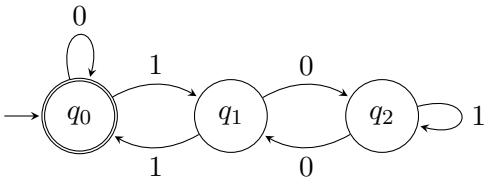


Abb. 20: DFA „modulo 3"

lineares Gleichungssystem mit 3 Unbekannten.

$$\begin{aligned}
L_0 &= \mathbb{1} + 0 \cdot L_0 + 1 \cdot L_1 \\
L_1 &= 1 \cdot L_0 + 0 \cdot L_2 \\
L_2 &= \underbrace{0 \cdot L_1}_{B} + \underbrace{1}_{A} \cdot L_2
\end{aligned}$$

Arden's Lemma auf q_2:

$$L_2 = 1^* \cdot 0 \cdot L_1$$

Einsetzen in q_1

$$L_1 = \underbrace{1 \cdot L_0}_{B} + \underbrace{0 \cdot 1^* \cdot 0}_{A} \cdot L_1$$

Arden's Lemma auf q_1:

$$L_1 = (01^*0)^* \cdot 1 \cdot L_0$$

Einsetzen:

$$\begin{aligned} L_0 &= \mathbb{1} + 0 \cdot L_0 + 1 \cdot (01^*0)^* \cdot 1 \cdot L_0 \\ &= \mathbb{1} + (0 + 1 \cdot (01^*0)^* \cdot 1) \cdot L_0 \end{aligned}$$

Arden's Lemma auf q_0:

$$L_0 = (0 + 1 \cdot (01^*0)^* \cdot 1)^*$$

3.6 Entscheidungsprobleme

Satz 3.11: Das Wortproblem ist für reguläre Sprachen entscheidbar.

D.h. Falls L reg. Sprache und $w \in \Sigma^*$, dann ex. Algorithmus, der entscheidet, ob $w \in L$.

Beweis: L sei durch DFA gegeben.
Berechnung von $\hat{\delta}(q_0, w)$ entspricht Durchlauf durch Graph des DFA + Test ob erreichter Zustand $\in F$ in Zeit $O(n)$, $n = |w|$. □

Satz 3.12: Das Leerheitsproblem ist für reg. Sprachen entscheidbar. Falls L reg. Sprache, dann existiert ein Algorithmus, der entscheidet, ob $L = \varnothing$.

Beweis: Sei $\mathcal{A}$ DFA für L.
Setze Tiefensuche auf den Graphen von $\mathcal{A}$ an. Start bei q_0.
Falls die Suche einem akzeptierenden Zustand findet: Nein.
Ansonsten: Ja: $L = \varnothing$
Zeit: $O(|\Sigma||Q|)$ □

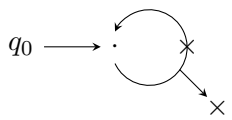
RL: Beweis vollständig?

Satz 3.13: Das Endlichkeitsproblem für reg. Sprachen ist entscheidbar.

Beweis: Falls L durch $r \in RE(\Sigma)$ gegeben.
r enthält keinen $^* \Rightarrow [\![r]\!]$ endlich.
Zeit: $O(|r|)$

[Reicht nicht, liefert nur eine Richtung]

Falls L durch DFA $\mathcal{A}$ gegeben.



Oder: mit Pumping Lemma.
Sei L regulär und n die Konstante aus dem PL.
L unendlich $\Leftrightarrow \exists w \in L : n \leq |w| < 2n$

"$\Leftarrow$": w erfüllt Voraussetzung des PL, also $w = uvx$ mit $|uv| \leq n$ und $|v| \geq 1$.
Nach PL: $\forall i \in \mathbb{N}$, $uv^i x \in L$, also L unendlich.

"$\Rightarrow$" Angenommen L unendlich, aber $\forall w \in L : |w| < n$ oder $|w| \geq 2n$
Sei $w \in L$ minimal gewählt, so dass $|w| \geq 2n$.

w erfüllt Voraussetzung vom PL, also $w = xyz$ mit $|xy| \leq n$ und $|y| \geq 1$
also $\forall i \in \mathbb{N} : xy^i z \in L$ insbes. $i = 0 : xz \in L$ mit $|xz| < |w|$.

Zwei Möglichkeiten:

(a) $|xz| \geq 2n$ ↯ Minimalität von w

(b) $|xz| < 2n$

$$|xz| + |y| = |w| \geq 2n \text{ mit } 1 \leq |y| \leq n$$
$$\curvearrowright |xz| = |w| - |y| \geq 2n - n = n \quad \text{↯ zur Annahme}$$

Also $\exists w \in L$ mit $n \leq |w| < 2n$ □

Satz 3.14: Das Schnittproblem ist für REG entscheidbar.
D.h. L_1, L_2 reguläre Sprachen. Ist $L_1 \cap L_2 = \varnothing$?

Beweis: Nach Satz 3.14 ist $L_1 \cap L_2$ regulär. $L_1 \cap L_2 = \varnothing$ entscheidbar nach Satz 3.12. □

Satz 3.15: Das Äquivalenzproblem ist für REG entscheidbar.
D.h. gegeben DFAs für L_1 und L_2, $\mathcal{A}_1$ und $\mathcal{A}_2$

$$L_1 = L(\mathcal{A}_1) = L(\mathcal{A}_2) = L_2 \ ? \qquad \boxed{\text{Inklusionsproblem}}$$

Beweis:

$$L_1 \cap \overline{L}_2 = \varnothing \quad \Leftrightarrow \quad L_1 \subseteq L_2$$
$$(L_1 \cap \overline{L}_2) \cup (L_2 \cap \overline{L}_1) = \varnothing \quad \Leftrightarrow \quad L_1 = L_2$$ □

Satz 3.16: Äquivalenzproblem $\Leftrightarrow$ Inklusionsproblem (für REG)

Beweis: $-$ = entspricht $\subseteq \wedge \supseteq$

- $L_1 \subseteq L_2$ genau dann, wenn $L_1 \cup L_2 = L_2$; REG ist abgeschlossen unter Vereinigung

□

Vorlesung: 04.12.15

Anwendungsbeispiel für reguläre Sprachen.

N – liest vom Netz
R – liest lokalen Speicher (ggf. vertrauliche Info)
W – postet auf FB

Programm:

$$p = N|R|W| \text{ if } * \text{ then } p_1 \text{ else } p_2 \mid \text{while } * \text{ do } p$$

$$\left.\begin{array}{l} \text{while } * \text{ do } N; \\ R; \\ \text{if } * \text{ then } N \text{ else } W \end{array}\right\} N^*R \cdot (N + W)$$

Sicherheitspolitik: nach Lesen von lokalem Speicher kein Posten auf FB $\overline{\Sigma^* R \Sigma^* W \Sigma^*}$

Programm erfüllt Sicherheitspolitik nicht, denn

$$\begin{array}{c} N^* \cdot R(N + W) \not\subseteq \overline{\Sigma^* R \Sigma^* W \Sigma^*} \\ \cup| \\ N^*RW \end{array}$$

4 Grammatiken und kontextfreie Sprachen

Wechsel des Standpunkts Spracherkennung → Spracherzeugung
Werkzeug: Phasenstrukturgrammatiken
Ansatz: Neben Alphabet gibt es weitere Symbole (Variable) und ein Regelsystem mit dem Worte, die Variable enthalten, geändert werden können.

Bsp. 4.1:

$$\Sigma = \{a, (,), *, +\}$$
$$N = \{E\}$$

Regeln:

$$\{\ E \to a,$$
$$E \to (E * E),$$
$$E \to (E + E)$$
$$\}$$

Startsymbol E

$$E \Rightarrow a$$
$$E \Rightarrow (E * E) \Rightarrow (a * E)$$
$$\Rightarrow (a * (E + E)) \Rightarrow \ldots \Rightarrow (a * (a + a))$$

Def. 4.1 (Chomsky)**:** Eine Grammatik ist ein 4-Tupel (N, Σ, P, S)

- N ist endliche Menge von Nichtterminalsymbolen
- Σ ist Alphabet von Terminalsymbolen (Variablen)
- $P \subset (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$ endliche Menge von Regeln bzw. Produktionen
- $S \in N$ das Startsymbol

$\oplus$

Def. 4.2: Die Ableitungsrelation zur Grammatik

$$\mathcal{G} = (N, \Sigma, P, S) \text{ ist Relation}$$
$$\Longrightarrow_{\mathcal{G}} \subseteq (N \cup \Sigma)^* \times (N \cup \Sigma)^* \text{ mit}$$
$$v \Rightarrow_{\mathcal{G}} w \ , \quad \text{falls ex. Produktion}$$
$$l \to r \in P \text{ und}$$
$$v = v_1 l v_2$$
$$w = v_1 r v_2$$
$$\Rightarrow^* \text{ refl. trans. Hülle von } \Rightarrow$$

Sprache von $\mathcal{G}$ erzeugt.

$$L(\mathcal{G}) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow}_{\mathcal{G}} w\}$$

Jede Folge $\alpha_0, \alpha_1, \ldots, \alpha_n$ mit $\alpha_i \in (N \cup \Sigma)^*$ mit $\alpha_0 = S$ und $\alpha_n \in \Sigma^*$ und $\forall 1 \leq i \leq n : \alpha_{i-1} \Rightarrow_{\mathcal{G}} \alpha_i$ heißt Ableitung von α_n

[Jedes solche α_i heißt Satzform von $\mathcal{G}$] ⊕

n ist Länge der Ableitung.

Bsp. 4.2: $\mathcal{G} = (N, \Sigma, P, S)$ mit

$$\begin{aligned}
N &= \{S, B, C\} \\
\Sigma &= \{a, b, c\} \\
8P = &\ \{S \to aSBC,\ S \to aBC,\ CB \to BC,\ aB \to ab, \\
&\quad bB \to bb, bC \to bc, cC \to cc\} \qquad\qquad \text{Startsymbol } S \\
L(\mathcal{G}) &= \{a^n b^n c^n \mid n \geq 1\} \\
&S \Rightarrow aBC \Rightarrow abC \Rightarrow abc \qquad S \Rightarrow a\underline{S}BC \Rightarrow aa\underline{S}BCBC \\
&S \Rightarrow aSBC \overset{n-2}{\Longrightarrow} n \Rightarrow a^{n-1}S(BC)^{n-1} \Rightarrow a^n(BC)^n = a^nBCBC\ldots \Rightarrow a^nBBCCBC \\
&\Rightarrow a^nB^nC^n \overset{*}{\Rightarrow} a^nb^nc^n
\end{aligned}$$

Die Chomsky-Hierarchie teilt die Grammatiken in vier Typen unterschiedlicher Mächtigkeit ein.

Def. 4.3 (Chomsky Hierarchie)**:**

- Jede Grammatik ist eine Typ-0 Grammatik.
- Eine Grammatik ist Typ-1 oder kontextsensitiv, falls alle Regeln expansiv sind, d.h. $\alpha \to \beta \in P$, dann ist $|\alpha| \leq |\beta|$ mit einer Ausnahme: falls S nicht in einer rechten Regelseite auftritt, dann ist $S \to \epsilon$ erlaubt.
- Eine Grammatik heißt Typ-2 oder kontextfrei, falls alle Regeln die Form $A \to \alpha$ mit $A \in N$ und $\alpha \in (N \cup \Sigma)^*$ haben.
- Eine Grammatik heißt Typ-3 oder regulär, falls alle Regeln die Form

$$\begin{aligned}
\text{Form} \quad & A \to w \qquad w \in \Sigma^* \\
\text{oder} \quad & A \to aB \qquad a \in \Sigma,\ B \in N
\end{aligned}$$

Eine Sprache heißt Typ-i Sprache, falls $\exists$ Typ-i Grammatik für sie. ⊕

Beobachtung: Jede Typ $i+1$ Sprache ist nicht Typ-i Sprache.

Jede Typ-3 Grammatik ist Typ-2 Grammatik.
Jede Typ-2 Grammatik kann in äquivalente ϵ-freie Typ-2 Grammatik transformiert werden.
$\rightarrow$ Typ-1 Grammatik. (Vgl. Elmination von ϵ-Produktionen)

Ziel: Hierarchie-Satz (Chomsky)
Sei $\mathcal{L}_i$ die Menge der Typ-i Sprachen.
Es gilt $\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$

Satz 4.1: L ist regulär $\Leftrightarrow$ L ist Type-3 Sprache

BEWEIS: „$\Rightarrow$“ Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ DFA für L.
Konstruiere Grammatik $\mathcal{G} = (N, \Sigma, P, S)$

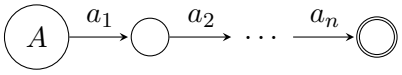
$$N = Q \qquad S = q_0$$
$$q \in Q\ \forall a : \delta(q, a) = q' \ \curvearrowright\ q \to aq' \in P$$
$$q \in F \qquad q \to \epsilon \in P$$

Zeige noch $L(\mathcal{G}) = L(\mathcal{A})$

„$\Leftarrow$“ Sei $\mathcal{G} = (N, \Sigma, P, S)$ Typ-3 Grammatik Def. $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ NFA

$$Q = N \cup \ldots$$
$$q_0 = S$$
$$A \to aB \in P \qquad \delta(A, a) \ni B$$
$$A \to a, \ldots a_n \qquad n = 0 \Rightarrow A \in F$$

$n > 0$: n weitere Zustände erforderlich, letzter Endzustand



Zeige noch $L(\mathcal{G}) = L(\mathcal{A})$ □

Vorlesung: 09.12.15

Lemma 4.2: $\mathcal{L}_3 \subsetneq \mathcal{L}_2$

BEWEIS: Betrachte $L = \{a^n b^n \mid n \in \mathbb{N}\}$
bekannt: L nicht regulär.
Aber $\exists$ Typ-2 Grammatik für L:

$$\mathcal{G} = (\{S\}, \{a, b\}, \{S \to \epsilon, S \to aSb\}, S)$$

Zeige (selbst) $L = L(\mathcal{G})$ □

4.1 Kontextfreie Sprachen

- Arithmetische Ausdrücke
  $E \to a, E \to (E + E), E \to (E * E)$
- Syntax von Programmiersprachen

$$\begin{aligned}\text{<Stmt>} \to\ & \text{<Var>} = \text{<Exp>} \\ \mid\ & \text{<Stmt>}\,;\text{<Stmt>} \\ \mid\ & \text{if}\,(\text{<Exp>})\ \text{<Stmt>} \\ & \text{else <Stmt>} \\ \mid\ & \text{while}\,(\text{<Exp>})\ \text{<Stmt>}\end{aligned}$$

Hier: Nichtterminal$\hat{=}$ Wort in spitzen Klammern <Stmt>; Terminalsymbol — alles andere.

– Palindrome über $\{a, b\}$

$$S \to aSa \mid bSb \mid a \mid b \mid \epsilon$$

– Sprache der Werte, die gleich viele Nullen wie Einsen haben $= L$

$$\begin{aligned}\mathcal{G} &: S \to 1S0S \mid 0S1S \mid \epsilon \\ L(\mathcal{G}) &\subseteq L \quad \text{klar} \\ \underline{\text{Def.:}}\quad d &: \Sigma^* \to \mathbb{Z} \\ d(\epsilon) &= 0 \\ d(1w) &= d(w) + 1 \\ d(0w) &= d(w) - 1 \\ &\text{Es gilt(Beweis per Induktion über } v) \\ d(v \cdot w) &= d(v) + d(w)\end{aligned}$$

$$\begin{aligned}L &= \{w \mid d(w) = 0\} \\ \text{Zeige:}\quad L &\subseteq L(\mathcal{G}) \\ \forall w \in L &: w \in L(\mathcal{G})\end{aligned}$$

Induktion über Länge von w

Länge 0 : $\epsilon \in L \quad , \quad \epsilon \in L(\mathcal{G}) \quad , \quad S \to \epsilon$

Länge > 0 :

$$\begin{aligned}&\text{Ang. } 0w \in L \\ &\curvearrowright d(0w) = 0 \\ &\curvearrowright d(w) = 1 \\ &\curvearrowright \exists \text{ kürzestes Suffix } 1w_2 \text{ von } w \text{ mit } d(1w_2) = 1 \\ &\curvearrowright \exists w_1 : w = w_1 1 w_2 \\ &\curvearrowright d(w_1) = 0 \wedge d(w_2) = 0\end{aligned}$$

Per Induktion sind sowohl $S \Rightarrow^* w_1$ als auch $S \Rightarrow^* w_2$; die erste Produktion ist $S \to 0S1S$.

Der Fall $1w$ geht analog.

Def. 4.4: Sei $\mathcal{G}$ Menge der kontextfreien Grammatiken (CFG).
Ein Ableitungsbaum zu $w \in L(\mathcal{G})$ ist ein geordneter, markierter Baum B.

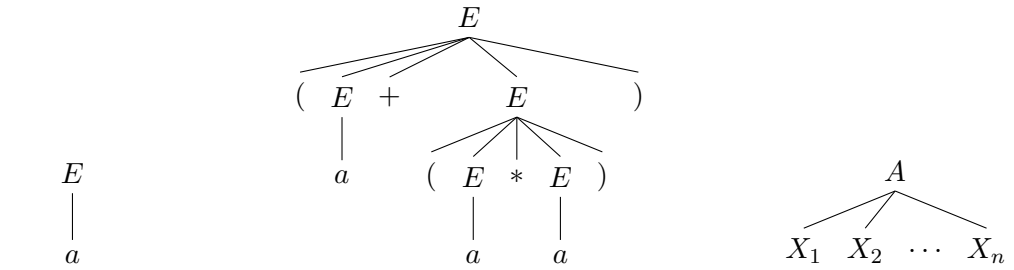
– Die Wurzel von B ist mit S markiert.

- Die inneren Knoten von B sind mit Nichtterminalen markiert.
- Jedes Blatt ist mit $x \in \Sigma \cup \{\epsilon\}$ markiert.
- Falls Knoten k_0 mit $A \in N$ markiert ist und die Nachfolgerknoten von k_0 sind $k_1, \ldots, k_n$, dann gibt Produktion $A \to X_1 \ldots X_n$ mit $X_i \in \Sigma \cup N$ und k_i ist mit X_i markiert.

  Oder: $n = 1$, die Produktion ist $A \to \epsilon$ und k_1 ist mit ϵ markiert.

Die Markierungen der Blätter von links nach rechts gelesen ergeben w. ⊕

Bsp. 4.3: Siehe Abb. 21.



(a) $E \Rightarrow a$ **(b)** $E \overset{*}{\Rightarrow} (a + (a * a))$ $\longrightarrow E \Rightarrow (E+E) \Rightarrow (E+(E*E)) \overset{*}{\Rightarrow}$ **(c)** $A \to X_1 \ldots X_n$

Abb. 21: Ableitungsbäume zu Bsp. 4.3

$\exists$ Ableitungsbaum für w genau dann, wenn $S \overset{*}{\Rightarrow} w$

Def. 4.5:

- $\mathcal{G} \in$ CFG heißt eindeutig, falls es für jedes Wort $\in L(\mathcal{G})$ genau einen Ableitungsbaum gibt.
- Eine kontextfreie Sprache heißt eindeutig, falls $\exists$ eindeutige CFG für sie.

⊕

Bsp. 4.4:

$L = \{a^i b^j c^k \mid i = j \text{ oder } j = k\}$

$S \to AC \mid DB$

$A \to aAb \mid \epsilon$ $\qquad$ $D \to aD \mid \epsilon$

$C \to cC \mid \epsilon$ $\qquad$ $B \to bBc \mid \epsilon$ [für L gibt es keine eindeutige CFG]

Werte der Form $a^n b^n c^n$ haben zwei Ableitungen $\curvearrowright$ Grammatik ist nicht eindeutig.

4.2 Die Chomsky Normalform für CFG

Notw. für PL + Wortproblem.

Def. 4.6: Eine CFG ist in Chomsky Normalform (CNF), falls jede Produktion die Form $A \to a$ oder $A \to BC$ hat, wobei $A, B, C \in N$, $a \in \Sigma$. $\bigoplus$

Beobachtung: $\mathcal{G}$ in CNF $\curvearrowright \epsilon \notin L(\mathcal{G})$

Transformation einer Grammatik $\mathcal{G}$ mit $\epsilon \notin L(\mathcal{G})$ nach CNF.

1. Schritt: Separierte Grammatik
d.h. jede Regel hat die Form

$$\begin{array}{lll} & A \to A_1 \dots A_n & A_i \in N, n \geq 0 \\ \text{oder} & A \to a & a \in \Sigma \end{array}$$

Verfahren:

- Erweitere N um neue NT $\{Y_a \mid a \in \Sigma\}$
- Zusätzliche neue Regeln $Y_a \to a$
- Ersetze in den „alten“ Regeln alle Terminale a durch Y_a

$\curvearrowright$ Grammatik separiert, Sprache ist unverändert.

2. Schritt: Eliminiere alle Regeln der Form $A \to \epsilon$ (ϵ-Produktion)
Ziel: ϵ-freie Grammatik.
Gesucht: $M = \{A \mid A \overset{*}{\Rightarrow} \epsilon\} \subseteq N$

$$\begin{array}{rl} \text{Def.:} & M_0 = \{A \mid A \to \epsilon\} \\ \forall i \in \mathbb{N} & M_{i+1} = M_i \cup \{A \mid A \to \alpha, \alpha \in M_i^*\} \\ \text{Es gilt } \forall i \in \mathbb{N} & M_i \subseteq N \text{ endlich} \\ \curvearrowright & \exists n \in \mathbb{N} : M_n = M_{n+1} = \bigcup_{i \in \mathbb{N}} M_i \end{array}$$

$$\begin{array}{rl} \text{Beh.:} & M = \bigcup_{i \in \mathbb{N}} M_i \\ \forall i \in \mathbb{N} : & M_i \subseteq M \\ i = 0 : & M_0 = \{A \mid A \to \epsilon \in P\} \subseteq M \\ i > 0 : & M_{i+1} = M_i \cup \{A \mid A \to \alpha, \alpha \in M_i^*\} \end{array}$$

$$
\begin{array}{lll}
\text{Falls} & A \to A_1 \dots A_n \in P,\ A_j \in M_i & \\
\text{dann} & A \Rightarrow A_1 \dots A_n & \\
 & \xrightarrow{\text{I.V.}} A_2 \dots A_n & \\
 & \vdots & \\
 & \xrightarrow{\text{I.V.}} A_n & \\
 & \xrightarrow{\text{I.V.}} \epsilon & \\
\curvearrowright & A \in M & \\
\to & \bigcup_{i \in \mathbb{N}} M_i \subseteq M &
\end{array}
$$

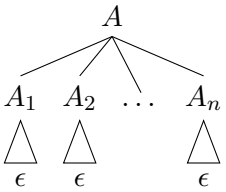
Zeige noch $M \subseteq \bigcup\limits_{i \in \mathbb{N}} M_i$

Sei $A \in M : A \overset{*}{\Rightarrow} \epsilon$ mit Ableitungsbaum der Höhe $h+1$, dann: $A \in M_h$

h=0: Ableitungsbaum $\quad A \quad A \to \epsilon \curvearrowright A \in M_0$

ϵ

h>0:



so dass $A_j \Rightarrow^* \epsilon$ mit Ableitungsbaum der Höhe $< h+1$

Nach I.V. $A_j \in M_{h-1}$ und $A \to A_1 \dots A_n \in P$, so dass $A \in M_h$.

Vorlesung: 11.12.15

Transformation im CNF (Fortsetzung)

1. Schritt: Separierte Grammatik: $\mathcal{A} \to A_1 \dots A_n$ oder $A \to a\ ,\ A_i \in N$
2. Schritt: ϵ-freie Grammatik, bereits bestimmt:

   $$M = \{A \mid A \Rightarrow \epsilon\}$$

   inkl. Algorithmus dazu: berechne M_i für $i = 0, 1, \dots$ bis keine neuen NT hinzukommen.

   Anpassen der Produktionen

   a) Falls Regel $A \to A_1 \dots A_n \in P$ und $\exists j : A_j \in M$, dann füge neue Regel $A \to A_1 \dots A_{j-1} A_{j+1} \dots A_n$ zu P hinzu.

   b) Wiederhole 2. Schritt: bis keine neuen Regeln erzeugt werden können.

   c) Entferne alle ϵ-Regeln.

   Ergebnis: Grammatik, die ϵ-frei ist und die gleiche Sprache erzeugt (o.B.).
3. Schritt: Beseitige Kettenregeln der Form $A \to B$

- Betrachte den gerichteten Graphen mit Knotenmenge N und den Kettenregeln als Kanten.
- Suche Zyklus mit Tiefensuche $A_1 \to A_2 \to \ldots \to A_r \to A_i$. Dann ersetzt Vorkommen von A_j $(j > 1)$ durch A_1 (auf linker und rechter Seite aller Produktionen).
- Wiederhole bis alle Zyklen eliminiert.
- Betrachte topologische Sortierung des verbleibenden DAG: $B_1 \ldots B_m$, so dass B_m keine Kettenregel besitzt.
- Starte mit B_m mit Produktion $B_m \to \alpha_i \mid \cdots \mid \alpha_k$
  Ersetze alle Vorkommen von B_m auf der rechten Seite von Kettenproduktionen durch $\alpha_1 \mid \cdots \mid \alpha_k$
- Wiederhole für $B_{m+1} \ldots B_1$
  $\curvearrowright$ alle Kettenproduktionen eliminiert.

4. Schritt: jetzt haben alle Produktionen die Form $A \to A_1 \ldots A_n$, $n \geq 2$ oder $A \to a$.
   - Falls $A \to A_1 A_2 \ldots A_n \in P$ mit $n > 2$ dann ersetze durch zwei neue Produktionen

     $$A \to A_1 B \qquad \text{mit } B \text{ neues NT}$$
     $$B \to A_2 \ldots A_n$$

   - Wdhl. bis alle rechten Seiten eine Länge ≤ 2 haben.

   $\curvearrowright$ CNF erreicht.

Bemerkung: Sei $|\mathcal{G}| = \sum_{A \to \alpha \in P} (|\alpha| + 1)$ die Größe einer CFG.
Die Transformation nach CNF benötigt Zeit $O(|\mathcal{G}|^2)$. Die Größe der CNF-Grammatik ist $O(|\mathcal{G}|^2)$.

4.3 Pumping Lemma für CFL

Satz 4.3 (Pumping lemma für CFL, uvwxy Lemma)**:** Sei $L \in CFL$. Dann $\exists n > 0$, so dass $\forall z \in L$ mit $|z| \geq n$ $\exists u, v, w, x, y$: mit

- $z = uvwxy$
- $|vwx| \leq n$
- $|vx| \geq 1$
- $\forall i \in \mathbb{N} : uv^i wx^i y \in L$

Beweis: Sei $\mathcal{G} = (N, \Sigma, P, S)$ in CNF mit $L(\mathcal{G}) = L \setminus \{\epsilon\}$.

Sei $k = |N|$ und wähle $n = 2^k$

Betrachte den Ableitungsbaum B von z mit $|z| \geq n$.

$\curvearrowright$ B ist (im wesentlichen) ein Binärbaum mit $|z| \geq n = 2^k$ Blättern.

$\curvearrowright$ in einem solchen Binärbaum B $\exists$ Pfad der Länge $\geq k$.

Auf diesem Pfad liegen $\geq k+1$ Nichtterminale (NT)
Also: mindestens ein NT A muss mehrmals auf dem Pfad vorkommen. Suche so ein A beginnend vom Blatt dieses Pfads, dann ist das zweite Vorkommen von $A \leq k$ Schritte vom Blatt entfernt.

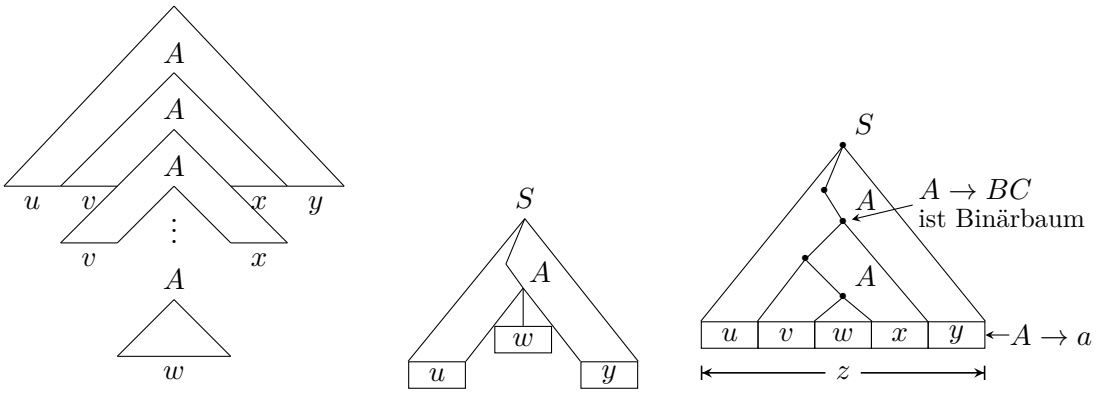
Aufgrund der Wahl von A hat der Ableitungsbaum für $A \Rightarrow^* vAx \overset{*}{\Rightarrow} vwx$ Höhe $\leq k$.

$\curvearrowright$ hat $\leq 2^k = n$ Blätter

$\curvearrowright$ $|vwx| \leq 2^k = n$

Es gilt $|vx| \geq 1$, weil

- Ableitungsbaum ist Binärbaum, mit inneren Knoten $A \to BC$
- Pfad belegt nur eine Abzweigung
- Wegen CNF $\not\exists C, C \overset{*}{\Rightarrow} \epsilon$



(a) $uv^iwx^iy \in L$ **(b)** $\curvearrowright uwy \in L$ **(c)** $\exists$ Pfad mit Länge $\geq k$

Abb. 22: Schema zu Satz 4.3

□

Lemma 4.4: $\mathcal{L}_2 \subsetneq \mathcal{L}_1$

Beweis: Sei $L = \{L = \{a^n b^n c^n \mid n \geq 1\}$.
L ist nicht kontextfrei. Verwende PL. Angenommen $L \in CFL$.
Sei n die Konstante aus dem PL.

Wähle $z = a^n b^n c^n$

$$\text{Also: } \exists u, v, w, x, y : z = uvwxy$$
$$|vwx| \leq n$$

$\curvearrowright$ vwx kann nicht gleichzeitig a, b und c enthalten.
Angenommen vwx enthält kein c. Dann muss $uwy \in L$

– y endet mit c^n

– $|vx| \geq 1$, also $|uwy| + 1 \leq |z|$ also „fehlt“ mindestens ein a oder b

$\curvearrowright uvw \notin L$ □

4.4 Entscheidungsprobleme für CFL

Satz 4.5: Das Wortproblem „$w \in L$?“ ist für Menge der kontextfreien Sprachen (CFL) entscheidbar. Falls $|w| = n$, benötigt der Algorithmus $O(n^3)$ Schritte und $O(n^2)$ Platz.

Beweis: Algorithmus Cocke, Younger, Kasami (CYK). □

Bsp. 4.5: $L = \{a^n b^n c^m \mid n, m \geq 1\}$ mit Grammatik (CFG)

$$S \to XY$$
$$X \to ab \mid aXb$$
$$Y \to c \mid cY$$

Umformen in CNF

$$S \to XY \qquad A \to a$$
$$X \to AB \mid AZ \qquad B \to b$$
$$Y \to c \mid CY \qquad C \to c$$
$$Z \to XB$$

Vorlesung: 18.12.15

$S \to XY$
$X \to AB$
$X \to AZ$
$Z \to XB$
$Y \to c$
$Y \to CY$
$A \to a$
$B \to b$
$C \to c$

Beweis: CYK-Algorithmus

Eingabe: – CFG $\mathcal{G}$ in CNF

– $w = a_1 \dots a_n \in \Sigma^*, |w| = n$

Ausgabe: „$w \in L(\mathcal{G})$“

Idee: Berechne eine $(n \times n)$ Matrix M mit Einträgen in $\mathcal{P}(N)$ mit folgender Spezifikation:

$M_{ij} = \{A \mid A \overset{*}{\Rightarrow} a_i \dots a_j\}$ $\qquad 1 \le i \le j \le n$ nur diese $M_{ij} \neq \varnothing$

falls $i = j$:

$$\begin{aligned} M_{ii} &= \{A \mid A \overset{*}{\Rightarrow} a_i\} \\ &= \{A \mid A \to a_i\} \end{aligned}$$

falls $1 \le i < j \le n$:

$$\begin{aligned} M_{ij} &= \{A \mid A \overset{*}{\Rightarrow} a_i \dots a_j\} \\ &= \{A \mid A \Rightarrow BC \overset{*}{\Rightarrow} a_i \dots a_j\} \\ &= \{A \mid A \to BC \wedge BC \overset{*}{\Rightarrow} a_i \dots a_j\} \\ &= \{A \mid A \to BC \wedge \exists k : i \le k < j \quad B \overset{*}{\Rightarrow} a_i \dots a_k \wedge \\ &\qquad\qquad C \overset{*}{\Rightarrow} a_{k+1} \dots a_j\} \\ &= \bigcup_{k=i}^{j-1} \{A \mid A \to BC \wedge B \overset{*}{\Rightarrow} a_i \dots a_k \wedge \\ &\qquad\qquad C \overset{*}{\Rightarrow} a_{k+1} \dots a_j\} \\ &= \bigcup_{k=i}^{j-1} \{A \mid A \to BC \wedge B \in M_{i,k} \wedge C \in M_{k+1,j}\} \end{aligned}$$

Damit: $w \in L$ genau dann, wenn $S \Rightarrow^* w$ genau dann, wenn $S \in M_{1,n}$.

Tabelle 1: $w = aaa\ bbb\ cc$ [1]

$M =$

							M_{1n}
A	•	•	•	•	X	$\underline{S}$	S
	A	•	•	$\underline{X}$	$\underline{Z}$	•	•
		A	X	Z	•	•	•
			B	•	•	•	•
				B	•	•	•
					B	•	•
						C, Y	Y
							C, Y

□

CYK$(\mathcal{G}, a_1, \dots, a_n)$
M $n \times n$ Matrix mit $M_{ij} = \varnothing$

```
for i=1 .. n do   // O(|G|) · O(n)
  M_ii = {A | A → a_i}
for i=n−1 .. 1 do
  for j=i+1 .. n do
    for k=i .. j−1 do   // O(|G|) · O(n^3)
```

[1] Zur Veranschaulichung: das Unterstrichene wird aus den Elementen der entsprechenden Farbe gebildet.

$$M_{ij} = M_{ij} \cup \{A \mid A \to BC, B \in M_{ik}, C \in M_{k+1,j}\}$$
return $S \in M_{1n}$

Satz 4.6: Das Leerheitsproblem ist für CFL entscheidbar.

BEWEIS: Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine CFG für L. Gefragt ist, ob $L(\mathcal{G}) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\} \overset{?}{=} \varnothing$. Betrachte dazu die Menge M der nützlichen Nichtterminalsymbole, aus denen ein Terminalwort herleitbar ist.

$$\begin{aligned} M &= \{A \mid A \overset{*}{\Rightarrow} w, w \in \Sigma^*\} \\ M_0 &= \{A \mid A \to w \in P, w \in \Sigma^*\} \\ M_{i+1} &= M_i \cup \{A \mid A \to \alpha \in P, \alpha \in (\Sigma \cup M_i)^*\} \\ \exists n : M_n &= M_{n+1} \overset{!}{=} M \\ L = \varnothing &\Leftrightarrow S \notin M \end{aligned}$$
□

Offenbar ist $M_0 \subseteq M$ eine Approximation von M, $M \supseteq M_{i+1} \supseteq M_i$ verbessert M_i und das jeweils nächste M_i ist in Linearzeit berechenbar. Da $M \subseteq N$ endlich ist, muss es ein n geben, sodass $M_n = M_{n+1}$. Für dieses n kann man zeigen, dass $M = M_n$.

Bem.: M ist die Menge der nützlichen Nichterminale. Nicht nützliche Nichtterminale können entfernt werden, ohne dass $L(\mathcal{G})$ sich ändert.

Satz 4.7: Das Endlichkeitsproblem für CFL ist entscheidbar.

BEWEIS: Mit PL analog zum Endlichkeitsproblem für reguläre Sprachen. □

4.5 Abschlusseigenschaften für CFL

Satz 4.8: Die Menge CFL ist abgeschlossen unter $\cup, \cdot, ^*$, jedoch nicht unter $\cap, \overline{\ }$.

BEWEIS:

$$\begin{aligned} \mathcal{G}_i &:= (N_i, \Sigma, P_i, S_i) \quad i = 1, 2 \\ \text{„}\cup\text{“} &: N = N_1 \dot\cup N_2 \dot\cup \{S\} \\ &\quad P = \{S \to S_1, S \to S_2\} \cup P_1 \cup P_2 \\ \text{„}\cdot\text{“} &:: N = N_1 \dot\cup N_2 \dot\cup \{S\} \\ &\quad P = \{S \to S_1 S_2\} \cup P_1 \cup P_2 \\ \text{„}^*\text{“} &: N = N_1 \dot\cup \{S\} \\ &\quad P = \{S \to \epsilon, S \to S_1 S\} \dot\cup P_1 \end{aligned}$$

– für $n \geq 1$

$$\underbrace{\{a^n b^n c^n\}}_{\notin \text{CFL}} = \underbrace{\{a^n b^n c^m \mid n, m \geq 1\}}_{\in \text{CFL}} \cap \underbrace{\{a^m b^n c^n \mid m, n \geq 1\}}_{\in \text{CFL}}$$

Also CFL nicht abgeschlossen unter $\cap$.

– Angenommen CFG abgeschlossen unter $\overline{}$
Falls $L_1, L_2 \in$ CFL, dann ist $\overline{L}_1, \overline{L}_2 \in$ CFL nach Annahme.
$\curvearrowright \overline{L}_1 \cup \overline{L}_2 \in$ CFL wegen Teil "$\cup$".
$\curvearrowright \overline{\overline{L}_1 \cup \overline{L}_2} = L_1 \cap L_2 \in$ CFL $\lightning$ zu Teil "$\cap$". □

Satz 4.9: Die Menge CFG ist abgeschlossen unter $\cap$ mit reg. Sprachen. „CFL $\cap$ REG = CFL".

Beweis:

$$\begin{aligned} \text{Sei } L &= L(\mathcal{G}) \quad \text{CFL} \\ R &= L(M) \quad REG \\ \mathcal{G} &= (N, \Sigma, P, S) \\ M &= (Q, \Sigma, \delta, q_0, F) \quad \text{DFA} \end{aligned}$$

Zeige $L \cap R$ CFL
Konstruiere CFG für $L \cap R$
Ang. $\mathcal{G}$ ist in CNF

$$\begin{array}{rl} A \to a & A \to BC \\ qAq' = \delta(q,a) & \quad q_1Aq_3 \\ \quad | & \quad\ \diagup\ \diagdown \\ \quad a & q_1Bq_2 \quad q_2Cq_3 \end{array}$$

$$\begin{array}{rl} N' = Q \times N \times Q \cup \{S'\} & \\ S' \to (q_0, S, q') & \forall q' \in F \\ (q, A, q') \to a & \text{falls } \delta(q,a) = q' \text{ und } A \to a \in P \\ (q_1, A, q_3) \to (q_1, B, q_2)(q_2, C, q_3) & \text{falls } A \to BC \in P \forall q_1, q_2, q_3 \in Q \end{array}$$

Zeige $(p, A, q) \overset{*}{\Rightarrow} w \Leftrightarrow \exists$ Lauf $p \ldots q$ von M auf w und $A \Rightarrow^* w$
per Induktion über Höhe h des Ableitungsbaums:

$h = 1$: $(p, A, q) \to a$ $\exists$ Lauf denn $\delta(p, a) = q$; offenbar $A \Rightarrow a$

$h > 1$: $(p, A, q) \to (p, \underset{\Downarrow *}{B}, r)(r, \underset{\Downarrow *}{C}, q)$

$$\text{mit } w \quad = \quad w_1 \qquad w_2$$

und Höhen $< h$:
Nach IV $\exists$ Lauf $p \ldots r$ auf w_1 und $B \Rightarrow^* w_1$
$\exists$ Lauf $r \ldots q$ auf w_2 und $C \Rightarrow^* w_2$
$\curvearrowright$ $\exists$ Lauf $p \ldots r \ldots q$ auf $w_1 w_2 = w$ und $A \Rightarrow BC \Rightarrow^* w$

Nun gilt

$$\begin{aligned} w \in L(\mathcal{G}') &\Leftrightarrow S \Rightarrow^* w \\ &\Leftrightarrow \exists q' \in F : (q_0, S, q') \Rightarrow^* w \\ &\Leftrightarrow \exists q' \in F : \exists \text{ Lauf von } q_0 \text{ nach } q' \text{ auf } w \wedge S \Rightarrow^* w \\ &\Leftrightarrow w \in R = L(M) \wedge w \in L = L(\mathcal{G}) \\ &\Leftrightarrow w \in R \cap L \end{aligned}$$

□

Vorlesung: 21.12.15

Satz 4.10: Falls $L \in \text{CFL}$ und $R \in \text{REG}$, dann ist „$L \subseteq R$?“ entscheidbar.

Beweis: $L \subseteq R \curvearrowleft \underbrace{L \cap \overline{R}}_{\text{CFL}} = \varnothing$ entscheidbar. □

5 Kellerautomaten pushdown automaton (PDA)

Kellerautomat ≈ Endlicher Automat + Kellerspeicher von unbeschränkter Größe (Stack, push down)

Neu:

- bei jedem Schritt darf der PDA das oberste Kellersymbol inspizieren und durch beliebiges Kellerwort ersetzen.
- der PDA darf auf dem Keller rechnen, ohne in der Eingabe weiter zu lesen. (ϵ-Transition oder Spontantransition).

Bsp. 5.1:

$$\begin{aligned}
\Sigma &= \{a, b, \#\} && \text{Eingabealphabet} \\
\Gamma &= \{a, b, \bot\} && \text{Kelleralphabet} \\
Q &= \{q_0, q_1\} && \bot \mathrel{\hat{=}} \text{Kellerbodensymbol} \\
\delta : (q_0, x, Z) &\mapsto (q_0, xZ) && x \in \{a, b\}, Z \in \Gamma && (2) \\
(q_0, \#, Z) &\mapsto (q_1, Z) && && (3) \\
(q_1, x, x) &\mapsto (q_1, \epsilon) && && (4) \\
(q_1, \epsilon, \bot) &\mapsto (q_1, \epsilon) && && (5) \\
\text{Konfiguration: } (q_0, w, \bot) & && \text{Start} && (6) \\
\downarrow & && && (7) \\
(q', \epsilon, \epsilon) & && \text{akzeptiert!} && (8)
\end{aligned}$$

Erkannte Sprache $L = \{w\#w^R \mid w \in \{a, b\}^*\}$

$$\begin{aligned}
& (q_0, abb\#bba, \bot) \\
\vdash\ & (q_0, bb\#bba, a\bot) \\
\vdash\ & (q_0, b\#bba, ba\bot) \vdash (q_0, \#bba, bba\bot) \\
\vdash\ & (q_1, bba, bba\bot) \\
\vdash\ & (q_1, ba, ba\bot) \vdash (q_1, a, a\bot) \\
\vdash\ & (q_1, \epsilon, \bot) \vdash (q_1, \epsilon, \epsilon) \qquad \text{akzeptiert!}
\end{aligned}$$

deterministisch.

Nicht deterministisch: $L' = \{ww^R \mid w \in \{a, b\}^*\}$
Ersetze (3) durch $(q_0, x, Z) \mapsto (q_1, Z) \quad Z \in \Gamma$

Alle Palindrome zusätzlich $(q_0, x, Z) \mapsto (q_1, Z)\ Z \in \Gamma,\ x \in \{a, b\}$ auch nicht deterministisch.

Def. 5.1: Ein nichtdeterministischer Kellerautomat (NPDA) ist Tupel $(Q, \Sigma, \Gamma, q_0, Z_0, \delta)$

- Q endl. Zustandsmenge

- Σ endl. Eingabealphabet
- Γ endl. Kelleralphabet
- $q_0 \in Q$ Startzustand
- $Z_0 \in \Gamma$ Kellerbodensymbol
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ $\oplus$

Im weiteren sei $M = (\dots)$ ein NPDA.

Def. 5.2: Die Menge der Konfigurationen von M ist $\text{Konf}(M) = Q \times \Sigma^* \times \Gamma^*$
Die Schrittrelation von M ist:

$$\begin{aligned} \vdash &\subseteq \text{Konf}(M)^2 && \text{definiert durch} \\ (q, aw, Z\gamma) &\vdash (q', w, \beta\gamma) && \text{falls } \delta(q, a, z) \ni (q', \beta) \\ (q, w, Z\gamma) &\vdash (q', w, \beta\gamma) && \text{falls } \delta(q, \epsilon, z) \ni (q', \beta) \end{aligned}$$

Die von M erkannte Sprache

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q', \epsilon, \epsilon)\}$$ $\oplus$

Satz 5.1:

$$L \in \text{CFL} \overset{\text{gdw.}}{\iff} L = L(M) \quad \text{für NPDA } M.$$

Beweis: "⇒" Sei $L \in$ CFL
Sei $\mathcal{G} = (N, \Sigma, P, S)$ Grammatik für $L \setminus \epsilon$ in CNF.
Def. NPDA M durch

$$\begin{aligned} Q &= \{q\} && (= \text{Startzustand}) \\ \Gamma &= \Sigma \dot\cup N \\ Z_0 &= S \\ \delta(q, a, a) &= \{(q, \epsilon)\} && a \in \Sigma \\ \delta(q, \epsilon, A) &= \{(q, \alpha)\} && A \to \alpha \in P \end{aligned}$$

für alle $X \in \Sigma \cup N$, $v \in \Sigma^*$

Zeige $X \overset{*}{\Longrightarrow} v$ $\quad \curvearrowright \quad (q, v, X) \vdash^* (q, \epsilon, \epsilon)$

per Induktion über Ableitungsbaum von $X \overset{*}{\Longrightarrow} v$
(Höhe von)

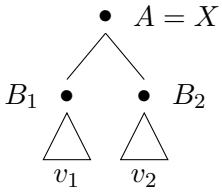
Höhe 0**:**

- $a = X = v$

$(q, a, a) \vdash (q, \epsilon, \epsilon)$ ✓

Höhe > 0**:**

1. Fall:  $(q,a,A) \overset{A\to a}{\vdash} (q,a,a) \vdash (q,\epsilon,\epsilon)$

2. Fall:  Regel $A \to B_1 B_2$

d.h. $v = v_1 v_2$ $B_1 \overset{*}{\Longrightarrow} v_1$ mit kleinerem Abeitungsbaum

$B_2 \overset{*}{\Longrightarrow} v_2$ ——— " ———

$(q, v_1 v_2, A) \vdash (q, v_1 v_2, B_1 B_2)$

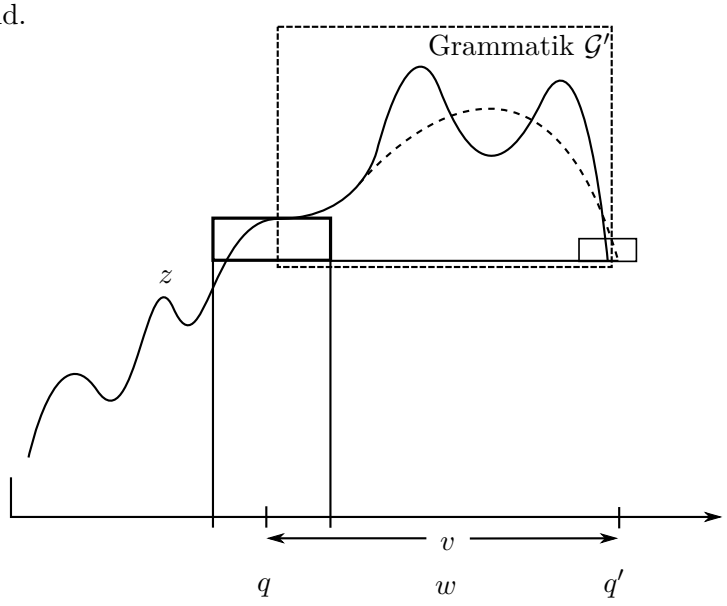
$\vdash^* (q, v_2, B_2)$ nach I.V. für B_1

$\vdash^* (q, \epsilon, \epsilon)$ nach I.V. für B_2

D.h. $S \overset{*}{\Longrightarrow} w \iff (q, w, S) \vdash^* (q, \epsilon, \epsilon)$

$\iff w \in L(M)$

"$\Leftarrow$" Gegeben NPDA $M = (Q, \dots)$ bei dem alle Transitionen pop, top oder push sind.



$(q, Z, q') \overset{*}{\Rightarrow} v$

$(q, v, Z) \vdash^* (q', \epsilon, \epsilon)$

Abb. 23: Beweis zu Satz 5.1

□

Grammatik $\mathcal{G}'$ leitet gerade v ab, falls Z im Stack.

Lemma 5.2: Zu jedem NPDA gibt es einen äquivalenten NPDA, so dass falls $\delta(q, x, Z) \ni (q', \alpha) \quad x \in \Sigma \cup \{\epsilon\}$ dann ist entweder

- $\alpha = \epsilon$
- $\alpha = Z'$
- $\alpha = Z'Z''$

BEWEIS: Sei $(q', \alpha) \in \delta(q, x, Z)$ mit $\alpha = Z_n \dots Z_1$ für $n > 2$:

- neue Zustände $q_2 \dots q_{n-1}$
- Ersetze (q', α) durch $(q_2, Z_2 Z_1)$
- Definiere $\delta(q_i, \epsilon, Z_i) = \{(q_{i+1}, Z_{i+1} Z_i)\}$, für $2 \leq i < n-1$
- Definiere $\delta(q_{n-1}, \epsilon, Z_{n-1}) = \{(q', Z_n Z_{n-1})\}$

Wiederhole bis alle Transitionen die gewünschte Form haben. □

Vorlesung: 22.12.15

$L = L(M)$ für NPDA M
$\curvearrowright$ L ist CFL

- Def. CFG mit $N = Q \times \Gamma \times Q \cup \{S\}$, so dass $(q, Z, q') \overset{*}{\Rightarrow} v$ gdw. $(q, v, Z) \vdash^* (q', \epsilon, \epsilon)$
  Transitionen von M haben eine von drei Formen:

$$(q, \epsilon) \qquad (q, Z) \qquad (q, Z_1 Z_2)$$

- Def. P durch $P \supseteq \{S \to (q_0, Z_0, q') \mid q' \in Q\}$ sowie die weiteren Produktion wie folgt

Produktion für (q, Z, q')

Fall 1: $\delta(q, x, Z) \ni (q', \epsilon)$, $x \in \Sigma \cup \{\epsilon\}$
$P \supseteq \{(q, Z, q') \to x\}$

Fall 2: $\delta(q, x, Z) \ni (q'', Z')$, $x \in \Sigma \cup \{\epsilon\}$
$P \supseteq \{(q, Z, q') \to x(q'', Z', q') \mid q' \in Q\}$

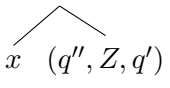
Fall 3: $\delta(q, x, Z) \ni (q_1, Z_1 Z_2)$, $x \in \Sigma \cup \{\epsilon\}$
$P \supseteq \{(q, Z, q') \to x(q_1, Z_1, q_2)(q_2, Z_2, q') \mid q_1, q_2 \in Q\}$

Korrektheit: Zeige $(q, Z, q') \overset{*}{\Rightarrow} w \curvearrowright (q, w, Z) \vdash^* (q', \epsilon, \epsilon)$
Induktion über Ableitungsbaum von $(q, Z, q') \overset{*}{\Rightarrow} w$ (siehe oben)

Höhe 1: (q, Z, q') Fall 1 $\curvearrowright (q, x, Z) \vdash (q', \epsilon, \epsilon)$
$\Big|$
x

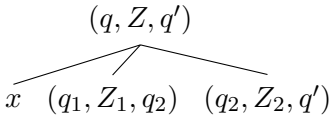
Höhe > 1: Zwei Möglichkeiten:

Fall 2:



$$\curvearrowright w = xw' \wedge (q'', Z', q') \overset{*}{\Rightarrow} w'$$
$$\curvearrowright (q, xw', Z) \vdash (q'', w', Z')$$
$$\text{I.V.} \quad \vdash^* (q', \epsilon, \epsilon)$$

Fall 3:



$$w = xw_1w_2$$
$$(q_1, Z_1, q_2) \overset{*}{\Rightarrow} w_1 \quad \text{mit kleinerem Ableitungsbaum}$$
$$(q_2, Z_2, q') \overset{*}{\Rightarrow} w_2 \quad \text{mit kleinerem Ableitungsbaum}$$
$$\curvearrowright (q, xw_1w_2, Z) \vdash (q_1, w_1w_2, Z_1Z_2)$$
$$\text{I.V. für } w_1 + \text{Lemma} \vdash^* (q_2, w_2, Z_2)$$
$$\text{I.V. für } w_2 \vdash^* (q', \epsilon, \epsilon)$$

Def. 5.3: Ein deterministischer Kellerautomat (DPDA) ist ein Tupel $(\underbrace{Q, \Sigma, \Gamma, q_0, Z_0}_{\text{wie gehabt}}, \delta, F)$

- $F \subseteq Q$ akzeptierende Zustände
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$
  $|\delta(q, a, Z)| + |\delta(q, \epsilon, Z)| \leq 1 \quad \forall q \in Q, a \in \Sigma, Z \in \Gamma$
- $\vdash$ wie gehabt
- $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q', \epsilon, \gamma) \wedge q' \in F\}$ $\oplus$

Lemma 5.3: Zu jedem DPDA gibt es einen äquivalenten DPDA, der die gesamte Eingabe verarbeitet.

Beweis: Erste Möglichkeit: Die Transitionsrelation ist nicht total.

Führe einen neuen Zustand $q_s \notin F$ ein.

Falls $\exists a \in \Sigma, q, Z$

$$\delta(q, a, Z) \cup \delta(q, \epsilon, Z) = \varnothing$$

dann erweitere δ um

$$\delta(q, a, Z) = \{(q_s, Z)\}$$

und $\forall a \in \Sigma, Z \in \Gamma : \delta(q_s, a, Z) = \{(q_s, Z)\}$

Weitere Möglichkeit: der Automat bleibt wegen leerem Keller stecken.
Abhilfe: Neues Kellerbodensymbol Z_0' und neuer Startzustand q_0'.

$$\delta(q_0', \epsilon, Z_0') = \{(q_0, Z_0 Z_0')\}$$

Erweitere δ für alle originalen zustände $q \in Q$ um

$$\delta(q, \epsilon, Z_0') = \{(q_s, Z_0')\}$$

„Falls Keller abgeräumt, Wechsel nach q_s“ □

Satz 5.4: Die deterministischen CFL sind unter Komplement abgeschlossen.

Beweis: Sei $L = L(M)$ für DPDA M. Nach Lemma 5.3 liest M die komplette Eingabe.
Def. M' mit $Q' = q \times \{0, 1, 2\}$
Zustand $(q, 0)$: seit Lesen des letzten Symbols wurde kein akz. Zustand durchlaufen.
Zustand $(q, 1)$ seit Lesen $\ldots \geq 1$ akz. Zustand durchlaufen
$(q, 2)$ …akz. Zustände d.h. $F' = F \times \{2\}$

$$q_0' = \begin{cases} (q_0, 0) & q_0 \notin F \\ (q_0, 1) & q_0 \in F \end{cases}$$

Falls $\delta(q, \epsilon, Z) = \{(q', \gamma)\}$ dann

$$\delta'((q, 0), \epsilon, Z) = \begin{cases} ((q', 0), \gamma) & q' \notin F \\ ((q', 1), \gamma) & q' \in F \end{cases}$$

$$\delta'((q, 1), \epsilon, Z) = ((q', 1), \gamma)$$

Falls $\delta(q, a, Z) = \{(q', \gamma)\}$

$$\delta'((q, 0), \epsilon, Z) = \{((q, 2), Z)\}$$

$$\delta'((q, 2), a, Z) = \begin{cases} ((q', 0), \gamma) & q' \notin F \\ ((q', 1), \gamma) & q' \in F \end{cases}$$

$$\delta'((q, 1), a, Z) = \begin{cases} ((q', 0), \gamma & q' \notin F \\ ((q', 1), \gamma & q' \in F \end{cases}$$

□

Satz 5.5: Die deterministischen CFL sind **nicht** unter Vereinigung und Durchschnitt abgeschlossen.

Beweis: Betrachte

$$L_1 = \{a^n b^n c^m \mid n, m \geq 1\}$$
$$L_2 = \{a^m b^n c^n \mid n, m \geq 1\}$$

Sowohl L_1 als auch L_2 sind DCFL, aber $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$ ist nicht kontextfrei.

DCFL ist nicht abgeschlossen unter Vereinigung. Angenommen doch: Seien U, V DCFL. Dann sind auch $\overline{U}$ und $\overline{V}$ DCFL. Bei Abschluss unter Vereinigung wäre $\overline{U} \cup \overline{V}$ eine DCFL und somit auch $\overline{\overline{U} \cup \overline{V}} = U \cap V$, ein Widerspruch gegen den ersten Teil. □

Satz 5.6: DCFL ist abgeschlossen unter Schnitt mit REG.

Beweis: Sei L DCFL und R regulär. Bilde das Produkt aus einem DPDA für L und einem DFA für R. Offenbar ist das Ergebnis ein DPDA, der $L \cap R$ erkennt. □

Satz 5.7: Sei L DCFL und R regulär. Es ist entscheidbar, ob $R = L$, $R \subseteq L$ und $L = \Sigma^*$.

BEWEIS: Es gilt $R \subseteq L$ gdw. $R \cap \overline{L} = \emptyset$.

Weiter ist $R = L$ gdw. $R \subseteq L$ und $L \subseteq R$. Für den zweiten Teil betrachte $L \cap \overline{R}$.

Für kontextfreie Sprachen ist $L \neq \emptyset$ entscheidbar, also betrachte $L = \Sigma^*$ gdw. $\overline{L} = \emptyset$. □

Satz 5.8: DPDA Äquivalenzproblem Seien L_1, L_2 DCFL. Dann ist $L_1 = L_2$ entscheidbar.

BEWEIS: Siehe Senizergues (2000) und Stirling (2001). □

6 Berechenbarkeit

Vorlesung: 08.01.16

6.1 Typ-0 und Typ-1 Sprachen

PT: checkpoint

Def. 6.1: Eine nichtdeterministische TM (NTM) ist ein Tupel $(Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$, wobei alles wie bei einer deterministische TM (DTM) außer $\delta : Q \times \Gamma \to \wp(Q \times \Gamma \times \{N, L, R\})$.
Konfiguration und Berechnungsrelation wie gehabt. ⊕

Def. 6.2: Sei M TM.

- M akzeptiert $w \in \Sigma^*$, falls $q_0 w \vdash^* uq'v$, $q' \in F$
- M akzeptiert $L \subseteq \Sigma^*$, falls M akzeptiert $w \leftrightarrow w \in L$
- M entscheidet $L \subseteq \Sigma^*$, falls M akzeptiert L und M hält für jede Eingabe an.
- $L \subseteq \Sigma^*$ ist semi-entscheidbar (rekursiv aufzählbar), falls $\exists M$, so dass diese L akzeptiert.
- $L \subseteq \Sigma^*$ ist entscheidbar (rekursiv), falls $\exists M$, so dass diese L entscheidet.

⊕

Def. 6.3: Laufzeit und Platzbedarf einer TM M :

$$\text{Laufzeit} : T_M(w) = \begin{cases} \text{Anzahl der Schritte einer kürzesten Berechnung, die zur Akz. von } w \text{ führt (falls } \exists) \\ 1, \text{ sonst} \end{cases}$$

$$\text{Platzbedarf} : S_M(w) = \begin{cases} \text{geringster Platzbedarf (Länge einer Konf.) einer akz. Berechnung von } w \text{ (falls } \exists) \\ 1, \text{ sonst} \end{cases}$$

Zeitbeschränkt mit $t(n)$: $\forall w \in \Sigma^* : |w| \leq n \Rightarrow T_M(w) \leq t(n)$,
platzbeschränkt analog. ⊕

Satz 6.1: Zu jeder NTM gibt es eine DTM M', so dass

- M akzeptiert $L(M)$
- M' terminiert gdw. M terminiert
- Falls M Zeit- und platzbeschränkt ist, mit $t(n)$ bzw. $s(n)$ (n = Länge der Eingabe), dann ist M' zeitbeschränkt mit $2^{O(t(n))}$ und platzbeschränkt mit $O(s(n) \cdot t(n))$.

Beweis: Die Konfiguration von M bilden einen Baum, dessen Kanten durch $\vdash$ gegeben sind. Er ist endlich verzweigt, hat aber ggf. unendlich lange Äste.

Definiere eine (Mehrband-)DTM, die den Konfigurationsbaum systematisch durchläuft und akzeptiert, sobald eine Konf. erreicht ist, in der M akzeptiert.

Die DTM terminiert ebenfalls, wenn alle Blätter des Baumes besucht worden sind, ohne dass eine akzeptierende Konfiguration gefunden wurde.

Baumsuche mit Kontrollinformation und bereits besuchten Konf. auf ein Extraband.

- Tiefensuche? Nicht geeignet, sie könnte in unendlichen Ast laufen.
- Breitensuche? OK, aber Platzbedarf $O(2^{t(n)} \cdot s(n))$

– iterative deepending : Tiefensuche mit vorgegebener Schranke, bei erfolgloser Suche Neustart mit erhöhter Schranke. □

Nächstes Ziel: Charakterisierung von Typ-1 Sprachen.

Def. 6.4:

– DTAPE($s(n)$) : Menge der Sprachen, die von einer DTM in Platz $s(n)$ akzeptiert werden können.

– NTAPE($s(n)$) : Wie für DTAPE, aber mit NTM. ⊕

Bemerkung:

1. Für „$s(n) \leq n$“ betrachte 2-Band TM, bei denen die Eingabe read-only ist und nur das zweite Arbeitsband der Platzschranke unterliegt (so ist $s(n)$ sublinear möglich).
2. Jede Platzbeschränkung impliziert Laufzeitschranke.
   Angenommen Platzschranke $s(n)$
   $\curvearrowright$ TM hat nur endlich viele Konfigurationen

$$N := \underbrace{n}_{\substack{\uparrow \\ \text{Kopfpos. im} \\ \text{Eingabeband}}} \underbrace{|Q|}_{\substack{\uparrow \\ \text{mögliche Inhalte} \\ \text{des} \\ \text{Arbeitsbands}}} \cdot \underbrace{|\Gamma|^{s(n)}}_{\substack{\uparrow \\ \text{Kopfpos. auf} \\ \text{Arbeitsband}}} \cdot \underbrace{vs(n)}_{\uparrow} \in 2^{O(\log n + s(n))}$$

3. DTM mit Platzschranke : M entscheidet,
   falls sie akzeptiert, dann in weniger als N Schritten,
   falls nach N Schritten keine Termination erfolgt
   $\curvearrowright$ Endlosschleife – Abbruch
4. NTM : nutze den Nicht-Determinismus (ND) optimistisch aus :
   falls eine akzeptierende Berechnung existiert, dann muss es eine Berechnung ohne wiederholte Konfiguration geben.

Satz 6.2:

– $L \in \text{DTAPE}(n) \curvearrowright \exists$ DTM, die L in Zeit $2^{O(n)}$ entscheidet.

– $L \in \text{NTAPE}(n)$ analog.

Beweis: siehe oben. □

Bemerkung: Die Klasse NTAPE heißt auch Linear Bounded Automaton (LBA).

Satz 6.3: $L = \text{NTAPE}(n)$

Beweis:

„⇒“ : Sei $G = (N, \Sigma, P, S)$ Typ-1 Grammatik für L.
Konstruiere NTM M mit $L = L(M),\ \Gamma = \Sigma \cup N \cup \{\sqcup\}$

1. M rät (ND) eine Position auf dem Band und Produktion $\alpha \to \beta$. Falls β gefunden wird, ersetze durch α, weiter bei 1.
2. Falls Bandinhalt $= S$ stop, akzeptiert.

Vorlesung: 13.01.16

„$\Leftarrow$“: Gegeben: NTM M linear beschränkt.
Gesucht: Typ-1 Grammatik $\mathcal{G}$ mit $L(\mathcal{G}) = L(M)$
Idee:

$$a_1 \cdot a_n \longrightarrow \begin{pmatrix} a_1 \\ a_1 \end{pmatrix} \begin{pmatrix} a_2 \\ a_2 \end{pmatrix} \begin{pmatrix} (q,a) \\ a_3 \end{pmatrix} \begin{pmatrix} a_4 \\ a_4 \end{pmatrix} \begin{pmatrix} a_n \\ a_n \end{pmatrix} \qquad \begin{matrix} \text{Spur 1} \\ \text{Spur 2} \end{matrix}$$

ad[2] Spur 1: Alphabet $\Gamma \cup (Q \times \Gamma) = \triangle$

$$P' \begin{cases} (q,a) & \to (q',a') \quad q \in Q, a \in \Gamma & \delta(q,a) \ni (q',a',N) \\ (q,a)b & \to a'(q',b) \quad b \in \Gamma & \delta(q,a) \ni (q',a',R) \\ b(q,a) & \to (q',b)a' & \delta(q,a) \ni (q',a',L) \end{cases}$$

Def. $\widetilde{uqav} = u(q,a)v \ , \ u,v \in \Gamma^*, a \in \Gamma$
Es gilt: $uqav \vdash^* k' \curvearrowright \widetilde{uqav} \overset{*}{\Rightarrow} \widetilde{k'}$ mit Produktion P'.

Def. $\mathcal{G}$ durch $N = \{S\} \dot\cup \triangle \times \Sigma$

mit $P =$

$$S \to \begin{pmatrix} (q_0,a) \\ a \end{pmatrix} \qquad \forall a \in \Sigma$$

$$S \to S \begin{pmatrix} a \\ a \end{pmatrix} \qquad \forall a \in \Sigma$$

$$\begin{pmatrix} \alpha \\ a \end{pmatrix} \to \begin{pmatrix} \beta \\ a \end{pmatrix} \qquad \forall \alpha \to \beta \in P', \ \alpha, \beta \in \triangle$$

$$\begin{pmatrix} \alpha_1 \\ a_1 \end{pmatrix} \begin{pmatrix} \alpha_2 \\ a_2 \end{pmatrix} \to \begin{pmatrix} \beta_1 \\ a_1 \end{pmatrix} \begin{pmatrix} \beta_2 \\ a_2 \end{pmatrix} \qquad \forall \alpha_1\alpha_2 \to \beta_1\beta_2 \in P', \ \alpha_i, \beta_i \in \triangle$$

$$\begin{pmatrix} x \\ a \end{pmatrix} \to a \qquad x \in \Gamma, \ a \in \Sigma$$

$$\begin{pmatrix} (q',x) \\ a \end{pmatrix} \to a \qquad x \in \Gamma, \ a \in \Sigma$$

$$\begin{aligned} S &\overset{*}{\Rightarrow} \begin{pmatrix} (q_0,a_1) \\ a_1 \end{pmatrix} \begin{pmatrix} a_2 \\ a_2 \end{pmatrix} \dots \begin{pmatrix} a_n \\ a_n \end{pmatrix} \\ &\quad \text{TM} \ \dots \\ &\overset{*}{\Rightarrow} \begin{pmatrix} x_1 \\ a_1 \end{pmatrix} \dots \begin{pmatrix} (q',x_i) \\ a_i \end{pmatrix} \dots \begin{pmatrix} x_n \\ a_n \end{pmatrix} \\ &\overset{*}{\Rightarrow} a_1 \dots a_i \dots a_n \end{aligned}$$

[2] ad $\approx$ zur

Damit gesehen $L(\mathcal{G}) \subseteq L(M)$
Rückrichtung: selbst □

Satz 6.4: Die Typ-1 Sprachen sind abgeschlossen unter $\cup$, $\cap$, $\cdot$, * und Komplement.

Beweis: Zu $\cup$ und $\cap$ betrachte NTM.
Für $\cdot$ und * konstruiere Grammatik.
ad Komplement „2. LBA-Problem[3]“ bis 1987, dann gelöst durch Immerman und Szelepcsényi.

1. LBA-Problem (1964): Ist NTAPE(n) = DTAPE(n)? Bisher ungelöst. □

Satz 6.5: Das Wortproblem für Typ-1 Sprachen ist entscheidbar.

Beweis:

$$L \in \mathcal{L}_1 \curvearrowleft L \in \text{NTAPE}(n)$$
$$\curvearrowright \text{Satz 6.2 } L \text{ entscheidbar}$$

Nach Satz 6.1 sogar mit DTM. □

Die Umkehrrichtung „L entscheidbar. $\not\curvearrowleft L$ ist Typ-1 Sprache“ gilt nicht!

Satz 6.6: $\mathcal{L}_0 = \text{NTM}$

Beweis: “$\Rightarrow$“ Kontruktion einer NTM M wie in Satz 6.3, aber ohne Platzbeschränkung.

“$\Leftarrow$“ Konstruktion analog zu Satz 6.3 + Startsymbol S'

$$S' \to \begin{pmatrix} \sqcup \\ \epsilon \end{pmatrix} S' \begin{pmatrix} \sqcup \\ \epsilon \end{pmatrix} \qquad \text{Schaffe Platz für Berechnung von } M$$
$$S' \to S$$

Erweitere N

$$= \{S', S\} \cup \triangle \times (\Sigma \cup \{\epsilon\})$$

Neue Löschregeln:

$$\begin{pmatrix} x \\ \epsilon \end{pmatrix} \to \epsilon \qquad \forall x \in \Gamma$$

$\leftharpoondown$ einzigen Regeln, die Typ-0 einsetzen. □

Satz 6.7: Die Typ-0 Sprachen sind unter $\cup$, $\cap$, $\cdot$, * abgeschlossen.

Beweis: Konstruiere NTM für $\cup$, $\cap$; Typ-0-Grammatiken für $\cdot$ und *. □

Bem.: Typ-0 Sprachen sind nicht unter Komplement abgeschlossen!

[3]LBA = Linear Bounded Automaton – 1964 Kuroda

6.2 Universelle TM und das Halteproblem

Ziel: Universelle TM (eine TM, die TMs interpretiert) ist eine TM U, die zwei Eingaben minimiert:

1. Kodierung einer (anderen) TM M_0
2. Eingabe w für M_0

so dass

$$w \in L(M_0) \curvearrowright (M_0, w) \in L(U)$$
$$M_0 \text{ terminiert mit } w \curvearrowright U \text{ terminiert mit } (M_0, w)$$

Zur Vereinfachung:

$$\Sigma = \{0, 1\}$$
$$\Gamma = \{\sqcup, 0, 1\}$$

Die Kodierung von M_0

$$= (Q, \ldots, q_1, \delta, \ldots, \{q_2\})$$

mit $Q = \{q_1, q_2, \ldots, q_t\}$ ist im wesentlichen die Kodierung von δ. Dazu zwei Hilfsfunktionen:

$x \in \Gamma$	$f(x)$
0	1
1	2
$\sqcup$	3

D	$g(D)$
L	1
N	2
R	3

Kodiere δ Zeilenweise:

$$\delta(q_i, X) = (q_k, Y, D) \ (= \text{Zeile}z)$$

durch

$$\text{code}(z) = 0^i 10^{f(x)} 10^k 10^{f(y)} 10^{g(D)}$$

Wenn δ durch s Zeilen $z_1 \ldots z_s$ gegeben, dann setze

$$\ulcorner M_0 \urcorner = 111\text{code}(z_1)11\text{code}(z_2)11 \ldots 11\text{code}(z_s)111$$

ist die Gödelnummer von M_0

Definiere U als 3-Band Maschine mit

B_1 : Eingabe + Arbeitsband (für M_0)

B_2 : $\ulcorner M_0 \urcorner$

B_3 : 0^k für Zustand q_k

1. Schritt: Transformierte Eingabe $\ulcorner M_0 \urcorner$

- Beginnt die Eingabe mit gültiger Gödelnummer?
- Gleichzeitig: Verschiebe $\ulcorner M_0 \urcorner$ auf B_2

– Wenn Ende von $\ulcorner M_0 \urcorner$ erreicht, schiebe 0 auf B_3.

Jetzt:

B_1 : w

B_2 : $\ulcorner M_0 \urcorner$

B_3 : $0' \sim Z, q$

Vorlesung: 14.01.16

Satz 6.8: Es gibt eine universelle TM U mit $L(U) = \{\ulcorner M \urcorner w \mid w \in L(M)\}$

BEWEIS: Initialisierung:

B_1 : w Eingabewort/Arbeitsband

B_2 : $\ulcorner M \urcorner$ Gödelnummer

B_3 : 0 Zustand

Hauptschleife von U:

– Test auf Haltekonfiguration.

– Falls ja: Falls in qz : akzeptiert.
sonst : nicht

– Ausführung des nächsten Schritts:
Suche Zeile in $\ulcorner M \urcorner$ gemäß Zustand und aktuellem Symbol auf Arbeitsband. Ändere B_1 und B_3 gemäß δ.
B_2, B_3: Kopf zurück zum Anfang.

□

Def. 6.5: Die length-lexicographic order auf $\{0,1\}^*$ (mit $0 < 1$) ist definiert durch

$$\begin{aligned} v \leq w \Leftrightarrow\ & |v| < |w| \\ & \vee\ v = w \\ & \vee\ |v| = |w| \text{ und } \exists u \in \{0,1\}^* \\ & \quad \text{mit } v = u0v' \text{ und } w = u1w' \end{aligned}$$

⊕

Satz 6.9:

– $\leq$ ist totale Ordnung auf $\{0,1\}^*$

– $\exists$ bijektive Abbildung $w : \mathbb{N} \to \{0,1\}^*$ mit $i \leq j \to w(i) \leq w(j)$

Def. 6.6: Sei M_i die Turingmaschine mit Gödelnummer $\ulcorner M_i \urcorner = w(i)$. (falls $w(i)$ kein gültiger Code, dann sei M_i eine beliebige TM mit $L(M_i) = \varnothing$). Die Diagonalsprache

$$D = \{w(i) \mid w(i) \notin L(M_i)\}$$

das heißt M_i akzeptiert $w(i)$ nicht. ⊕

Satz 6.10: D ist nicht entscheidbar.

Beweis: Angenommen $\exists M$, die D entscheidet.
M muss in Aufzählung vorkommen, das heißt $\exists j \in \mathbb{N}$, sodass $M = M_j$. Wende M_j auf $w(j)$ an:

- M_j stoppt/ja: $w(j) \in L(M_j) = D$ $\quad$ $\lightning$ Def. von D
- M_j stoppt/nein: $w(j) \notin L(M_j) = D$ $\quad$ $\lightning$ Def. von D □

Korollar 6.11: $\overline{D} = \{w(i) \mid M_i \text{ akzepztiert } w(i)\}$ ist nicht entscheidbar. ⊕

Beweis: Angenommen $\overline{D}$ sei entscheidbar durch M. Dann entscheidet M' D. M' führt zuerst M aus und negiert das Ergebnis. $\lightning$ Satz 6.10 □

Lemma 6.12: $\overline{D}$ ist semi-entscheidbar.

Beweis: Bei Eingabe w

- Falls w kein gültiger Code: stop mit Ergebnis nein.
- Falls w gültiger Code, dann $\exists i$, sodass $ww = \ulcorner M_j \urcorner w(i)$ $\quad$ $w = w(i)$

Wende U auf ww an.
Insgesamt: TM, die $\overline{D}$ akzeptiert. □

Satz 6.13: Falls L semi-entscheidbar und $\overline{L}$ semi-entscheidbar, dann ist L entscheidbar.

Beweis: Sei M die TM für L, $\overline{M}$ die TM für $\overline{L}$.
Führe M und $\overline{M}$ „parallel“ mit der gleichen Eingabe aus.
Falls M akzeptiert $\Rightarrow$ Ja
Falls $\overline{M}$ akzeptiert $\Rightarrow$ Nein.
Eine muss anhalten, wegen Voraussetzung. □

D nicht entscheidbar
$\overline{D}$ nicht entscheidbar
$\overline{D}$ semi-entscheidbar (Typ-0)
D nicht-semientscheidbar (keine Typ-0)

Def. 6.7: Das Halteproblem ist definiert durch

$$H = \{\ulcorner M \urcorner w \mid M \text{ hält bei Eingabe } w \text{ an}\}$$ ⊕

Satz 6.14: H ist unentscheidbar.

Beweis: Angenommen M_0 entscheidet H.
Konstruiere M' wie folgt:
Bei Eingabe w bestimme i, sodass $w = w(i)$
Verwende M_0 um festzustellen, ob $\ulcorner M_i \urcorner w$ anhält.
Antwort von M_0 :
nein: $\curvearrowright w(i) \notin L(M_i)$
$\curvearrowright M'$ akzeptiert w nicht.

ja: Führe $\ulcorner M_i \urcorner w$ mit U aus (muss ja terminieren) und akzeptiere entsprechend das Ergebnis von U.
Insgesamt: M' entscheidet $\overline{D}$ $\quad \lightning$ Korollar 6.11
$\curvearrowright M_0$ existiert nicht. □

Satz 6.15: H ist semi-entscheidbar.

Beweis: Modifiziere U, sodass sie jede Eingabe akzeptiert, bei der sie anhält. □

Korollar 6.16: $\mathcal{L}_0 \supsetneq \mathcal{L}_1$ ⊕

Beweis: $\overline{D}$ ist unentscheidbar (also $\notin \mathcal{L}_1$), aber semi-entscheidbar (also $\in \mathcal{L}_0$) □

Fragen:

1. Ist $\mathcal{L}_1$ = Menge der entscheidbaren Sprachen?
   - Nein:
     Konstruiere eine Aufzählung aller Typ-1 Grammatiken.
     $D_1 = \{w(i) \mid w(i) \notin L(G_i)\}$ ist entscheidbar, weil das Wortproblem für Typ-1 Sprachen entscheidbar, aber es $\nexists j$, sodass $L(G_j) = D_1$s

	w_1 w_2 w_3 $\cdots$
G_1	
G_2	
G_3	
$\vdots$	

2. Ist $\mathcal{L}_0$ = Menge aller Sprachen?
   - Nein: $D \notin \mathcal{L}_0$

Def. 6.8: Das spezielle Halteproblem $H_\varepsilon = \{\ulcorner M \urcorner \mid M \text{ terminiert auf leeren Band}\}$ ⊕

Satz 6.17: H_ε ist unentscheidbar.

Beweis: Angenommen M_ε entscheidet H_ε.
Konstruiere M' (mit Hilfe von M_ε), sodass M' entscheidet H.
M' : Bei Eingabe $\ulcorner M \urcorner w$.
Konstruiere M^*, M^* schreibt zuerst w aufs (leere) Band und startet dann M auf w.
Wende M_ε auf $\ulcorner M^* \urcorner$ an.
$\curvearrowright M'$ entscheidet H $\quad \lightning$ Satz 6.14 □

Vorlesung: 20.01.15

Nun betrachten wir TMs vom Blickwinkel der von ihnen berechneten (partiellen) Funktionen.
Sei R die Menge der von TMs berechneten Funktionen.

Satz 6.18 (Satz von Rice)**:**
Sei R die Menge aller partiellen TM-berechenbaren Funktionen und $\varnothing \neq S \subsetneq R$ eine nichttriviale (nicht-leere, echte) Teilmenge davon.
Dann ist $L(S) = \{\ulcorner M \urcorner \mid M \text{ berechnet Funkt. aus } S\}$ unentscheidbar.

Beweis: Angenommen M_S entscheidet $L(S)$.
Sei $\Omega \in R$ die überall undefinierte Funktion. Wir nehmen an, dass $\Omega \in S$ (anderenfalls betrachten wir $\overline{L(S)}$).

Da $R \setminus S \neq \varnothing$ folgt $\exists f \in R \setminus S$ und f werde von TM M_f berechnet.

Definiere $M' = M'_{(M,f)}$ wie folgt: M' führt zunächst M (beliebige TM) auf leerer Eingabe aus. Falls M anhält, wendet M' dann M_f auf die tatsächliche Eingabe an.

Die von M' berechnete Funktion ist also $f_{M'} = \begin{cases} f & \text{falls } M \text{ auf leerem Band hält,} \\ \Omega & \text{sonst.} \end{cases}$

Definiere nun M'' wie folgt:

- Bei Eingabe $\ulcorner M \urcorner$ berechne die Gödelnummer von M'.
- Wende nun M_S auf $\ulcorner M' \urcorner$ an.

$$\begin{aligned} M_s \text{ akzeptiert } \ulcorner M' \urcorner &\iff M' \text{ berechnet Funktion in } S \\ &\iff M' \text{ berechnet } \Omega \ (f_{M'} \in \{\Omega, f\}, \Omega \in S, f \notin S\ ) \\ &\iff M \text{ hält } \underline{\text{nicht}} \text{ auf leerem Band an} \end{aligned}$$

Also entscheidet M'' H_ε. $\lightning$ □

6.3 Eigenschaften von entscheidbaren und semi-entscheidbaren Sprachen

Satz 6.19: Seien L_1 und L_2 entscheidbar. Dann sind $\overline{L_1}$, $\overline{L_2}$, $L_1 \cup L_2$ und $L_1 \cap L_2$ entscheidbar.

Beweis: Übung oder selbst. □

Satz 6.20: Seien L_1 und L_2 semi-entscheidbar. Dann sind $L_1 \cup L_2$ und $L_1 \cap L_2$ semi-entscheidbar.

Beweis: vgl. Satz 6.7. □

Satz 6.13 (Wiederholung)**:** Falls L semi-entscheidbar und $\overline{L}$ semi-entscheidbar, dann ist L entscheidbar.

Satz 6.21: Die Menge der semi-entscheidbaren Sprachen ist <u>nicht</u> unter Komplement abgeschlossen.

Beweis: Laut Satz 6.10 und Korollar 6.11 sind Diagonalsprache D und $\overline{D}$ nicht entscheidbar.

$\overline{D}$ ist semi-entscheidbar: Verwende universelle TM U um $\ulcorner M_i \urcorner w(i)$ auszuführen.

Angenommen, $D = \overline{\overline{D}}$ ist ebenfalls semi-entscheidbar
$\curvearrowright$ (mit Satz 6.13) D ist entscheidbar. $\lightning$ □

6.4 Weitere unentscheidbare Probleme

Das Postsche Korrespondenzproblem (PCP)

<u>Gegeben:</u>
Endliche Folge von Wortpaaren $K = ((x_1, y_1), \ldots, (x_k, y_k))$ mit $x_i, y_i \in \Sigma^+$

Gesucht:
Indexfolge $i_1, \ldots, i_n \in \{1, \ldots, k\}$ $(n \geq 1)$, so dass $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$

Die Folge $i_1, \ldots, i_n$ (falls diese existiert) heißt Lösung des Korrespondenzproblems K.

Bsp.:

$$K = ((\underbrace{1}_{x_1}, \underbrace{101}_{y_1}), (\underbrace{10}_{x_2}, \underbrace{00}_{y_2}), (\underbrace{011}_{x_3}, \underbrace{11}_{y_3}))$$

besitzt die Lösung $(1, 3, 2, 3)$, denn

$$x_1 x_3 x_2 x_3 = \underbrace{1 \cdot 01}_{y_1} \underbrace{1 \cdot 1}_{y_3} \underbrace{0 \cdot 0}_{y_2} \underbrace{11}_{y_3} = y_1 y_3 y_2 y_3$$

Frage:

$x_1 = 001$	$x_2 = 01$	$x_3 = 01$	$x_2 = 10$
$y_1 = 0$	$y_2 = 011$	$y_3 = 101$	$y_2 = 001$

Besitzt dieses PCP eine Lösung? [Schöning, S.124]

Bemerkung:
Offensichtlich ist das PCP semi-entscheidbar: Systematisches Ausprobieren von Indexfolgen findet Lösung nach endlicher Zeit, sofern es eine gibt.

Ziel: PCP ist unentscheidbar. Vorbereitung: Es interessiert uns ab hier nur, ob das Problem eine Lösung hat oder nicht.

Def. 6.9 (Reduktion)**:**
Seien $U, V \subseteq \Sigma^*$ Sprachen.
U ist auf V reduzierbar $(U \preceq V)$, falls eine totale berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ existiert, so dass $\forall x \in \Sigma^* : x \in U \iff f(x) \in V$. $\oplus$

Lemma 6.22: Falls $U \preceq V$ und V (semi-)entscheidbar, dann ist auch U (semi-)entscheidbar.

Beweis: Wenn M ein (Semi-)Entscheidungsverfahren für V ist, dann konstruiere M' wie folgt

- wende erst f auf die Eingabe x an (f ist Funktion gemäß Reduktion)
- führe M auf dem Ergebnis $f(x)$ aus

$\curvearrowright M'$ ist (Semi-)Entscheidungsverfahren für U $\square$

Anwendung: $U \preceq V$ und U unentscheidbar $\curvearrowright V$ unentscheidbar.

Das modifizierte PCP (MPCP)
Gegeben: wie bei PCP

Gesucht: Lösung des CP mit $i_1 = 1$

Lemma 6.23: MPCP $\preceq$ PCP

Beweis: Betrachte MPCP $K = ((x_1, y_1), \dots (x_k, y_k))$ über Σ.
Sei $\Sigma' = \Sigma \cup \{\#, \$\}$ mit neuen Symbolen.
Für ein Wort $w = a_1 \dots a_n \in \Sigma^+$ sei

$$\begin{aligned} \bar{w} &= \#a_1\#a_2\# \dots \#a_n\# \\ \grave{w} &= \#a_1\#a_2\# \dots \#a_n && \text{(am Ende kein \#)} \\ \acute{w} &= \quad a_1\#a_2\# \dots \#a_n\# && \text{(am Anfang kein \#)} \end{aligned}$$

Definiere nun

$$f(K) = (\underbrace{(\bar{x}_1, \grave{y}_1)}_{1}, \underbrace{(\acute{x}_1, \grave{y}_1)}_{2}, \underbrace{(\acute{x}_2, \grave{y}_2)}_{2+1}, \dots, \underbrace{(\acute{x}_k, \grave{y}_k)}_{k+1}, \underbrace{(\$, \#\$)}_{k+2})$$

eine totale berechenbare Funktion.

Zeige $K \in \text{MPCP} \iff f(K) \in \text{PCP}$:

„$\Longrightarrow$“: $1, i_2, \dots, i_n$ Lösung für K
$\curvearrowright$ $1, i_2 + 1, \dots, i_n + 1, k + 2$ Lösung für $f(K)$

„$\Longleftarrow$“: $i_1, \dots, i_n$ Lösung für $f(K)$
$\curvearrowright i_1 = 1,\ i_n = k + 2 \quad ,\ 1 < j < n : i_j \in \{2, \dots, k + 1\}$
$\curvearrowright 1, i_2 - 1, \dots, i_{n-1} - 1$ Lösung für K □

Es reicht nun zu zeigen, dass MPCP unentscheidbar ist!

Vorlesung: 28.01.16

Lemma 6.24: H $\preceq$ MPCP

Beweis: TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ und Eingabewort $w \in \Sigma^*$.

Gesucht: totale berechenbare Funktion, die $(\ulcorner M \urcorner, w) \mapsto \underbrace{(x_1, y_1), \dots, (x_k, y_k)}_{k}$, sodass

$\ulcorner M \urcorner w \in H \Leftrightarrow K$ eine Lösung als MPCP besitzt.

Idee: Definiere K so, dass die Berechnung von M simuliert wird.
Alphabet für K : $\triangle = \Gamma \cup Q \cup \{\#\}$
$(x_1, y_1) = (\#, \#q_0w\#)$

1. Kopieren

$$(a, a) \quad , a \in \Gamma \cup \{\#\}$$

2. Transition

$$\begin{array}{rl}
(qa, q'a') & \forall q, a : \delta(q,a) \ni (q', a', N) \\
(qa, a'q') & \quad \ldots \quad \ni (q', a', R) \\
(bqa, q'ba') & \ni (q', a', L), b \in \Gamma \\
(q\#, q'a'\#) & \forall q : \delta(q, \sqcup) \ni (q', a', N) \\
(q\#, a'q'\#) & \ni (q', a', R) \\
(bq\#, q'ba'\#) & \ni (q', a', L), b \in \Gamma \\
(\#qa, \#q'\sqcup a') & \forall q, a : \delta(q,a) \ni (q', a', L) \\
(\#q\#, \#q'\sqcup a'\#) & \forall q : \delta(q, \sqcup) \ni (q', a', L)
\end{array}$$

3. Löschen

$$\begin{array}{l}
\forall q \in \Gamma \\
\forall q \in F : (aq, a) \\
\qquad\quad (qa, a)
\end{array}$$

4. Abschluss

$$\forall q \in F : (q\#\#, \#)$$

$\ulcorner M \urcorner w \in H \iff$ Folge von Konf von M, $k_0 \ldots k_t$ mit $k_0 = q_0 w$ und $k_t = uq'v$ mit $q' \in F$ mit $h_{i-1} \vdash k_i \quad \forall 1 \leq i \leq t \iff$ Die Instanz K von MPCP besitzt Lösung und ein Lösungswort der Form

$$\#k_0\#k_1\# \ldots \#h_t\#k_t^1\#k_t^2\# \ldots \#q'\#\#$$

wobei die k_t^j durch Streichen eines Bandsymbols rechts oder links von q' aus ihrem Vorgänger entsteht. □

Satz 6.25: PCP ist unentscheidbar.

Beweis: H $\leq$ MPCP und MPCP $\leq$ PCP □

Satz 6.26: Das Schnittproblem „$L(\mathcal{G}_1) \cap L(\mathcal{G}_2) \neq \varnothing$?“ für CFL ist unentscheidbar.

Beweis: Durch Reduktion PCP $\leq$ Schnittproblem.
Sei $K = \{(x_i, y_i) \mid 1 \leq i \leq k\}$ Instanz von PCP über Σ.
Berechne aus K zwei CFG $\mathcal{G}_1$ und $\mathcal{G}_2$, so dass K eine Lösung hat $\iff L(\mathcal{G}_1) \cap L(\mathcal{G}_2) \neq \varnothing$

$$\begin{array}{lll}
\mathcal{G}_1 : S_1 \to & 1x_1 | \ldots | kx_k & \text{Alphabet} : \Sigma \cup \{1 \ldots k\} \\
& |1S_1x_1| \ldots |kS_1x_k & \\
\mathcal{G}_2 : S_2 \to & 1y_1 | \ldots | ky_k & \\
& |1S_2x_1| \ldots |kS_2y_k &
\end{array}$$

$$\begin{array}{rl}
& w \in L(\mathcal{G}_1) \cap L(\mathcal{G}_2) \\
\iff & w = k_n \ldots k_1, xk_1 \ldots xk_n \\
& \quad = k_n \ldots k_1, yk_1 \ldots yk_n \\
\iff & (k_1 \ldots k_n) \text{ ist Indexfolge zur Lösung von PCP } k
\end{array}$$

□

Folgerung : Schnittproblem für Typ 1 und Typ 0 Sprachen ist ebenfalls unentscheidbar.

Korollar 6.27: Das Schnittproblem ist auch für deterministische CFL (DCFL) unentscheidbar. $\bigoplus$

BEWEIS: $L(G_1)$ ist auch DPDA erkennbar. $\square$

Satz 6.28: Das Äquivalenzproblem für CFL ist unentscheidbar.

BEWEIS: Sei $A = \{\mathcal{G}_1, \mathcal{G}_2 \mid L(\mathcal{G}_1) = L(\mathcal{G}_2)\}$
Angenommen $\mathcal{G}_1, \mathcal{G}_2$ sind Typ 2 Grammatiken für DCFL.
Dann ist $(\mathcal{G}_1, \mathcal{G}_2) \in$ Schnittproblem.

$$\begin{aligned} &\iff L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \varnothing \\ &\iff L(\mathcal{G}_1) \subseteq \overline{L(\mathcal{G}_2)} \end{aligned}$$

Da $\mathcal{G}_2$ eine deterministische CFG (DCFG) $\exists\, \mathcal{G}_2$ mit $L(\mathcal{G}_2') = \overline{L(\mathcal{G}_2)}$ (Abschluss unter Komplement).

$$\begin{aligned} &\iff L(\mathcal{G}_1) \subseteq L(\mathcal{G}_2') \quad \rightsquigarrow \text{Inklusionsproblem} &(*) \\ &\iff L(\mathcal{G}_1) \cup L(\mathcal{G}_2') = L(\mathcal{G}_2') \end{aligned}$$

Wegen Abschluss unter $\cup : \exists\, \mathcal{G}_3 \in$ CFG mit $L(\mathcal{G}_3) = L(\mathcal{G}_1) \cup L(\mathcal{G}_2')$

$$\begin{aligned} &\iff L(\mathcal{G}_3) = L(\mathcal{G}_2') \\ &\iff (\mathcal{G}_3, \mathcal{G}_2') \in A \end{aligned}$$

$\curvearrowright$ Äquivalenzproblem ist unentscheidbar.
$(*) \rightarrow$ (Inklusionsproblem ist ebenfalls unentscheidbar.) $\square$

Satz 6.29: Das Leerheitsproblem für Typ 1 Sprachen ist unentscheidbar.

BEWEIS: Reduktion auf Schnittproblem für CFL.
Sei $(\mathcal{G}_1, \mathcal{G}_2) \in$ Schnittproblem (Typ 1).
Insbesondere $\mathcal{G}_1, \mathcal{G}_2$ Typ 1 Grammatiken.
Typ 1 Sprachen sind unter $\cap$ abgeschlossen, also $\exists\, \mathcal{G}$ Typ 1 Gramatik mit $L(\mathcal{G}) = L(\mathcal{G}_1) \cap L(\mathcal{G}_2)$
Also „$L(\mathcal{G}) = \varnothing$“ unentscheidbar. $\square$

7 Komplexitätstheorie

Vorlesung: 29.01.15

7.1 Komplexitätsklassen und P/NP

Def. 7.1: Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion.
Die Klasse NTIME($f(n)$) besteht aus allen Sprachen, die von einer (Mehrkanal-)TM M in $T_M(w) \leq f(|w|)$ akzeptiert werden.
Dabei $T_M(w) = \begin{cases} \text{Anzahl der Schritte einer kürzesten akzeptierenden Berechnung von } M \text{ auf } w \\ 1 \quad \text{falls } \nexists \end{cases}$

$\oplus$

Def. 7.2: Ein Polynom ist eine Funktion $p : \mathbb{N} \to \mathbb{N}$ mit $\exists k \in \mathbb{N}\ a_0, \ldots, a_k \in \mathbb{N}$ und $p(n) = \sum_i^k a_i n^k$ $\oplus$

Def. 7.3: Die Klasse NP besteht aus allen Sprachen, die von NTM in polynomieller Zeit akzeptiert werden können.

$$NP = \cup_{p \text{ Polynom}} \text{NTIME}(p(n))$$

$\oplus$

Analog für deterministische TM:

Def. 7.4: Sei $f : \mathbb{N} \to \mathbb{N}$ Funktion
DTIME($f(n)$) = Klasse der Sprachen, die von DTM in $T_M(w) \leq f(|w|)$ Schritten akzeptiert wird.

$$P = \cup_{p \text{ Polynom}} DTIME(P(n)) \qquad \oplus$$

Offenbar $P \leq NP$. Seit 1970 weiß man nicht, ob $P = NP$ oder $P \neq NP$

Praktische Relevanz: Es existieren wichtige Probleme, die offensichtlich in NP liegen, aber die besten bekannten Algorithmen sind exponentiell.
Beispiel: Traveling Salesman ($O(2^n)$), Erfüllbarkeit der Aussagenlogik.

Struktur: Viele der NP-Probleme haben sich als gleichwertig erwiesen, in dem Sinn, dass eine P-Lösung für alle anderen liefert.
$\rightsquigarrow$ NP-Vollständigkeit.

Def. 7.5: Seien $A, B \subseteq \Sigma^*$ Sprachen. A ist polynominell reduzierbar auf B, $A \preceq_p B$, falls $\exists$ totale berechenbare Funktion $f : \Sigma^* \to \Sigma^*$, deren Konfigurationszeit durch ein Polynom beschränkt ist und $w \in A \iff f(w) \in B \quad \forall w \in \Sigma^*$ $\oplus$

Lemma 7.1: Falls $A \preceq_p B$ und $B \in P$ (NP) dann auch $A \in P$ (NP).

Beweis: $B \in P : \exists M$, die B in $p(n)$ Schritten akzeptiert.
$\exists M_f$, die die Reduktion $A \preceq_p B$ implementiert. Die Laufzeit von M_f sei durch q Polynom beschränkt.
Betrachte $M' =$"erst M_f, dann M auf dem Ergebnis" M' akzeptiert A.

$w \in A$ $M_f(w)$ liefert $f(w)$ in $\subseteq q(|w|)$ Schritten ohne $|f(m)| \subseteq q(|w|)$
$M(f(w))$ benötigt $\leq p(|f(w)|) \leq p(q(|w|))$ Schritte zum akzeptieren.
$\curvearrowright A \in \text{DTIME}(q(w) + p(q(w)) \subseteq P$ □

Lemma 7.2: $\preceq_p$ ist reflexiv und transitiv.

Beweis: Identität; ähnlich wie Beweis von Lemma 7.1. □

Def. 7.6:

- Eine Sprache A heißt NP-hart (NP-schwer), falls $\forall L \in NP : L \preceq_p A$.
- Eine Sprache A heißt NP-vollständig, wenn A NP-hart und $A \in NP$. ⊕

Bem.: Sobald eine NP-hartes Problem A bekannt ist, reicht es $A \preceq_p B$ zu finden, um zu teigen, dass B ebenfalls NP-hart ist.

Satz 7.3: Sei A NP-vollständig.

$$A \in P \iff P = NP$$

Beweis:
„$\Longleftarrow$“ trivial.
„$\Longrightarrow$“ $A \in P \subseteq NP \quad \forall L \in NP : L \preceq_p A$. Nach Lemma 7.1 : $L \in P$ □

Ein erstes NP-vollständiges Problem.

Def. 7.7: SAT, das Erfüllbarkeitsproblem der Aussagenlogik (AL) ist definiert durch

Eingaben: Formal F der Aussagenlogik.

Frage: Ist F erfüllbar, d.h. existiert eine Belegung β der Variablen mit $\{0, 1\}$, so dass $F[\beta] = 1$ ist.

$SAT = \{\text{code}(F) \mid F\text{ist erfüllbare Formel der AL}\}$ ⊕

Satz 7.4 (Cook)**:** SAT ist NP-vollständig.

Beweis:

1. $SAT \in NP$
   Rate nicht deterministisch eine Belegung β
   Werte $F[\beta]$ aus
   $\curvearrowright$ in NTIME(n) polynomiell
2. SAT ist NP-hart.
   Zeige: $\forall L \in NP : L \preceq_p SAT$
   $L \in NP : \exists p$ Polynom, NTM M mit $L = L(M)$ mit Zeitschranke $T_M(w) \leq p(|w|)$.

   Sei $w = x_1 \dots x_n \in \Sigma^*$ Eingabe für M.
   Definiere F, so dass F erfüllbar $\iff$ M akzeptiert w

Sei $Q = Q(M)$ mit $\{q_1, \ldots, q_k\} = Q$
Sei $\Gamma = \Gamma(M)$ mit $\{a_i, \ldots, a_l\} = \Gamma$
Definiere folgende Variablen zur Ver. in F

- $\text{state}(t, q) = 1$, genau dann wenn M nach T Schritten im Zustand q
- $\text{pos}(t, i) = 1$, gdw. der Kopf von M steht nach t Schritten auf Position i.
  $t \in \{0, \ldots P(n)\}$
  $i \in \{-p(n), \ldots, 0, 1, \ldots, p(n)\}$
- $\text{tape}(t, i, a) = 1$, gdw. nach t Schritten befindet sich a an Position i auf dem Band.
  $t \in \{0, \ldots, p(n)\}$
  $i \in \{-p(n), \ldots, p(n)\}$
  $a \in \Gamma$ □

Lemma 7.5: Für jedes $k \in \mathbb{N}$ $\exists$ Formel G mit $G(x_i, \ldots, x_k) = 1$ gdw. einer der $x_i, \ldots, x_k = 1$ ist. Die Größe von G ist polynomiell in k.

Beweis:

$$G(x_i, \ldots, x_k) = \bigvee_{i=1}^{k} x_i \wedge \bigwedge_{i \neg j} \neg(x_i \wedge x_j)$$

$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ erkennt L in NTIME(p), p Polynom.

Vorlesung: 03.02.15

Ziel: Konstruiere aus M, w eine Formel F, so dass

$$\begin{aligned}
F \text{ erfüllbar} &\leftrightarrow M \text{ akzeptiert } w \\
\text{state}(t, q) \quad & t \in 0, \ldots, p(n), q \in Q \\
&\Leftrightarrow \text{ nach } t \text{ Schritten ist } M \text{ in Zeile } q \\
\text{pos}(t, i) &\Leftrightarrow \text{ nach } t \text{ Schritten ist Kopf amn Pos } i \quad , \quad -p(n) \leq i \leq p(n) \\
\text{tape}(t, i, a) &\Leftrightarrow \text{ nach } t \text{ Schritten enthält Band}[i] = q \in \Gamma \\
F &= R \wedge A \wedge T_1
\end{aligned}$$

1. Randbedingungen

$$\begin{aligned}
R = \quad & \bigwedge_t G(\text{state}(t, q1), \ldots, \text{state}(t, q_k)) \\
& \wedge \bigwedge_t G(\text{pos}(t, -p(n)), \ldots, \text{pos}(t, D), \ldots, \text{pos}(t, p(n))) \\
& \wedge \bigwedge_{t,i} G(\text{tape}(t, i, a_1), \ldots, \text{tape}(t, i, a_l)
\end{aligned}$$

2. Anfangskonfiguration

$$\begin{aligned}
A = \quad & \text{state}(0, q_1) \wedge \text{pos}(0, 1) \\
& \wedge \text{tape}(0, 1, x_1) \wedge \cdots \wedge \text{tape}(0, n, x_n) \\
& \wedge \bigwedge_{-p(n) \leq i \leq p(n)} \text{tape}(0, i, \sqcup)
\end{aligned}$$

3. Transitionsschritte

$$\begin{aligned} T_1 = & \bigwedge_{\substack{t\in 0,\ldots,p(n)-1,\\ i,n}} \text{state}(t,q) \wedge \text{pos}(t,i) \wedge \text{tape}(t,i,a) \\ & \rightarrow \text{state}(z+1,q') \wedge \text{pos}(t+1,i+d) \wedge \text{tape}(t+1,i,a') \\ & \delta(q,a) \ni (q',a',d) \\ & d \in \{-1,a,1\} \\ T_2 = & \bigwedge_{\substack{t,i,q\\ t<p(n)}} \neg\,\text{pos}(t,i) \wedge \text{tape}(t,i,a) \rightarrow \text{tape}(t+1,i,a) \end{aligned}$$

4. Endkonfiguration

$$E = \bigvee_{q\in F} \text{state}(p(n),q)$$

$|F|$ ist polznomiell beschränkt in $|M,w|$, also $L \preceq_p$ SAT
$\curvearrowright$ SAT ist NP-vollständig.

□

7.2 Weitere *NP*-vollständige Probleme

Def. 7.8: $3SAT$ ist das Erfüllbarkeitsproblem der AL für Formeln in CNF mit höchstens drei Literalen pro Klausel ⊕

Satz 7.6: $3SAT$ ist NP-vollständig.

Beweis: $3SAT \in NP$ offensichtlich.
Reduktion $SAT \preceq_p 3SAT$. Sei F eine Formel der AL.
Definiere ϕ : Formel $\rightarrow \{0,1\}^* \rightarrow$ Formel, sodass $F' = \phi(F,\epsilon)$ erfüllbar ist, gdw. F erfüllbar.

$$\begin{aligned} \phi(\text{true},\pi) &= [y_\pi] \qquad ,y_\pi \text{ neue Variable, nicht aus } F \\ \phi(\text{false},\pi) &= [\overline{y_\pi}] \\ \phi(x_i,\pi) &= [y_\pi \leftrightarrow x_i] \\ \phi(F_0 \wedge F_1,\pi) &= \phi(F_0,\pi_0) \wedge \phi(F_1,\pi_1) \wedge [y_\pi \leftrightarrow y_{\pi_0} \wedge y_{\pi_1}] \\ \phi(F_0 \vee F_1,\pi) &= \phi(F_0,\pi_0) \wedge \phi(F_1,\pi_1) \wedge [y_\pi \leftrightarrow y_{\pi_0} \vee y_{\pi_1}] \\ \phi(F,\pi) &\text{ ist min. um einen linearen Faktor größer als } F \\ &\quad \text{ist Konj. von eingeklammerten Termen } [\ldots] \end{aligned}$$

- $y_\pi \leftrightarrow x_i = (\overline{y_\pi} \vee x_i) \wedge (y_\pi \vee \overline{x_i})$
- $y_\pi \leftrightarrow y_{\pi_0} \wedge y_{\pi_1} = (\overline{y_\pi} \vee y_{\pi_0} y_{\pi_1}) \wedge (y_\pi \vee \overline{y_{\pi_0} y_{\pi_1}}) = (\overline{y_\pi} \vee y_{\pi_0}) \wedge (\overline{y_\pi} \vee y_{\pi_1})$
- $y_\pi \leftrightarrow y_{\pi_0} \vee y_{\pi_1} = (y_\pi \vee \overline{y_{\pi_0}}) \wedge (y_\pi \vee \overline{y_{\pi_1}}) \wedge (\overline{y_\pi} \vee y_{\pi_0} \vee y_{\pi_1}) \wedge (y_\pi \vee \overline{y_{\pi_0}} \vee \overline{y_{\pi_1}})$

□

Def. 7.9 (CLIQUE)**:** Sei $\mathcal{G} = (V, E)$ ein ungerichteter Graph und $k \in \mathbb{N}$.
$(\mathcal{G}, k) \in$ CLIQUE, falls $\exists$ Clique der Größe k in $\mathcal{G}$.
Eine Clique $C \subseteq V$, so dass $\forall u \neq v \in C : \{u, v \in E\}$ $\oplus$

Satz 7.7: CLIQUE ist NP-vollständig.

Beweis: Durch Reduktion: $3SAT \preceq_p$ CLIQUE
Sei F eine Formel in 3CNF, erweitert, so dass jede Klausel 3 Literale

$$(x \vee y) \rightsquigarrow (x \vee y \vee x)$$
$$x \rightsquigarrow (x \vee x \vee x)$$

Jetzt $F = \bigwedge_{i=1}^{m} (z_{i,1} \vee z_{i,2} \vee z_{i,3})$ $\qquad z_{i,j} \in \{x_i, \ldots, x_n\} \cup \{\overline{x}_1, \ldots, \overline{x}_n\}$

Definiere $\mathcal{G} = (V, E)$ und k wie folgt:

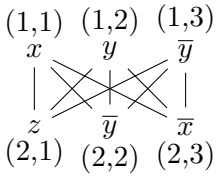
$$V = \{(i, j) \mid 1 \leq i \leq m, j \in \{1, 2, 3\}\}$$
$$E = \{\{(i, j), (p, q)\} \mid i \neq p, z_{i,j} \neq \neg z_{p,q}$$
$$k = m$$

F ist erfüllbar.
$\iff$ in jeder Klausel i muss mindestens ein Literal $= 1$ sein, unter Bedingung β.
$\iff \exists$ Folge $z_{1,j_2}, \ldots, z_{m,j_m}$ mit $z_{i,j_2}[\beta] = 1$
$\iff \exists$ Folge $z_{1,j_1}, \ldots, z_{m,j_m}$, sodass $\forall i \neq p : z_{i,j_i} \neq \neg z_{p,j_p}$
$\iff \exists$ Menge von Knoten $\{(1, j_1), \ldots, (m, j_m)\}$ die paarweise durch Kanten verbunden sind.
$\iff \exists$ Clique der Größe $k = m$ in $\mathcal{G}$ $\square$

Bsp.:

$$F = \underbrace{(x \vee y \vee \overline{y})}_{1} \wedge \underbrace{(z \vee \overline{y} \vee \overline{x})}_{2}$$

$\mathcal{G}$:



RL: Grafik überprüfen

8 Rekursive Funktionen

Vorlesung: 10.02.16

Def. 8.1 (Schema der primitiven Rekursion)**:**

$$\begin{aligned} \text{Sei } \mathcal{G} : \mathbb{N}^k \to \mathbb{N} \quad &, \ k \geq 0 \\ h : \mathbb{N}^{k+R} \to \mathbb{N} & \\ \text{Sei } f : \mathbb{N}^{k+1} \to \mathbb{N} &\text{ eine Funktion, die folgende Gleichungen erfüllt:} \\ f(0, \overline{x}) = g(\overline{x}) \quad &, \ \overline{x} \in \mathbb{N}^k \\ f(n+1, \overline{x}) = h(f(n, \overline{x}), n, \overline{x}) & \end{aligned} \tag{9}$$

Dann sei

$$f =: PR(g, h)$$

aus g und h durch primitive Rekursion definiert. $\oplus$

Lemma 8.1: $PR(g, h)$ ist wohldefiniert.

Beweis:

$$\begin{aligned} \text{Angenommen } f &= PR(g, h) \\ \text{und } f' &= PR(g, h) \end{aligned}$$

d.h. f und f' erfüllen (9).

$$\begin{aligned} \text{Zeige } \forall n \in \mathbb{N}, \forall \overline{x} \in \mathbb{N}^k : & \\ f(n, \overline{x}) &= f'(n, \overline{x}) \\ \text{I.A. } n = 0 : f(0, \overline{x}) &\overset{(9)}{=} g(\overline{x}) \overset{(9)}{=} f'(0, \overline{x}) \\ \text{I.S. } n \to n+1 : f(n+1, \overline{x}) &\overset{(9)}{=} h(f(n, \overline{x}), n, \overline{x}) \\ &\overset{\text{I.V.}}{=} h(f'(n, \overline{x}), n, \overline{x}) \\ &\overset{(9)}{=} f'(n+1, \overline{x}) \end{aligned}$$

□

Def. 8.2: Die PREC der primitiv rekursiven Funktionen über $\mathbb{N}$, ist induktiv definiert:

1. $\forall k \in \mathbb{N} : \mathcal{O}^k : \mathbb{N}^k \to \mathbb{N}$
   $\mathcal{O}^k(\overline{x}) = 0$
   $\mathcal{O}^k \in$ PREC (die Nullfunktion)
2. $\forall k \geq 1 : \forall 1 \leq j \leq k : \quad \pi_j^k : \mathbb{N}^k \to \mathbb{N}$
   $\pi_j^k(x_1, \ldots, x_k) = x_j$
   $\pi_j^k \in$ PREC (Projektionen)
3. $S : \mathbb{N} \to \mathbb{N} \ \in$ PREC (Nachfolgerfunktion)

4. Feld $g : \mathbb{N}^k \to \mathbb{N}$ PREC

$$\begin{aligned} \forall 1 \leq i \leq k : h_i : \mathbb{N}^m \to \mathbb{N} \quad &\in \text{PREC} \\ \text{dann ist } g \circ [h_1, \ldots, h_k] : \mathbb{N}^m \to \mathbb{N} \quad &\in \text{PREC} \\ (g \circ [h_1, \ldots, h_k])(\overline{x}) = g(h_1(\overline{x}), \ldots, h_k(\overline{x})) \quad &\times \in \mathbb{N}^m \end{aligned}$$

(Funktionskomposition)

5. Falls $g : \mathbb{N}^k \to \mathbb{N}$ und $h : \mathbb{N}^{k+2} \to \mathbb{N} \quad \in \text{PREC}$
dann ist auch $PR(g, h) \in \text{PREC}$ (Rekursion) ⊕

Bsp.:

1. Addition ∈ PREC

$$\begin{aligned} \text{add}(0, x) &= x \\ \text{add}(n+1, x) &= \text{add}(n, x) + 1 \\ &= h(\text{add}(n, x), n, x) \\ \text{add} &= PR(g, h) \\ \text{mit } g(x) &= x \quad ; g = \pi_1' \\ h(y, n, x) &= y + 1 \quad ; h = S \circ \pi_1^S \\ \text{add} &= PR(\pi_1', S \circ \pi_1^3) \end{aligned}$$

2. Multiplikation:

$$\begin{aligned} \text{mult} &: \mathbb{N}^2 \to \mathbb{N} \\ \text{mult}(0, x) &= 0 \\ \text{mult}(n+1, x) &= \text{add}(x, \text{mult}(n, x)) \\ \text{mult} &= PR(g, h) \\ \text{mit} \qquad g &= \mathcal{O}^1 \\ h(y, n, x) &= \text{add}(x, y) \\ \text{d.h.} \qquad h &= \text{add} \circ [\pi_3^3, \pi_1^3] \end{aligned}$$

Beobachtung: Alle primitiv rekursiven Funktionen sind total.
Aber: ∃ totale Funktion, die nicht primitiv rekursiv ist.

Def. 8.3 (Minimierung)**:** Sei $f : \mathbb{N}^{k+1} - \to \mathbb{N}$
dann ist $\mu f := g \in \mathbb{N}^k - \to \mathbb{N}$
definiert durch:

$$g(\overline{x}) = \min\{n \mid f(n, \overline{x}) = 0 \text{ und } \forall j < n : f(j, \overline{x}) \text{ def. und } \neq 0\} \qquad \oplus$$

Falls $f(n, \overline{x}) \neq 0 \ \forall n : g(\overline{x})$ undef.
Falls $f(n, \overline{x}) = 0$ aber $\exists j < n \ f(j, \overline{x})$ undef.
$\curvearrowright g(\overline{x})$ undef.

Def. 8.4: Die Klasse der μ-rekursiven Funktionen über $\mathbb{N}$ ist die kleinste Klasse von Funktionen, die die Basisfunktionen (1,2,3 aus der Def. 8.2) enthalten und abgeschlossen ist unter Komposition 4 Rekursion 5 und Minimierung ⊕

Satz 8.2: Die Klasse der μ-Funktionen stimmt mit der Klasse der TM-berechenbaren Funktionen über $\mathbb{N}$ überein.

Beweis:

- simuliere URM mit μ-Rekursion.
- simuliere μ-Rekursion mit TM (analog zur universellen TM) □

RL: URM = unbeschränkte Registermaschine?

Satz 8.3 (Kleene)**:** Für jede μ-rekursive Funktion $F : \mathbb{N}^k - \to \mathbb{N}$ gibt es zwei primitiv rekursive Funktionen $p, q : \mathbb{N}^{k+1} \to \mathbb{N}$, sodass $f(\overline{x}) = p(\overline{x}, \mu q(\overline{x}))$

↷ Jede berechenbare Funktion über $\mathbb{N}$ kann mit einer WHILE-Schleife programmiert werden.

Def. 8.5: Die Ackermannfunktion

$$\begin{aligned} A &: \mathbb{N}^2 \to \mathbb{N} \\ A(0, y) &= y + 1 \\ x > 0 : A(x, y) &= A(x - 1, 1) \\ y > 0 : A(x, y) &= A(x - 1, A(x.y - 1)) \end{aligned}$$ ⊕

Satz 8.4: Sei $\mathcal{A}$ die Menge der Funktionen, die durch A majorisiert werden.

$$\text{Es gilt: } \mathrm{PREC} \subseteq \mathcal{A}$$

Beweis: $A = \{f : \mathbb{N}^k \to \mathbb{N} \mid \exists n \forall \overline{x} : f(\overline{x}) < A(m, \max \overline{x})\}$.
Induktion über PREC:

1. $\mathcal{O}(\overline{x}) = 0 < A(0, \max \overline{x}) = \max \overline{x} + 1, \quad \mathcal{O} \in \mathcal{A}$
2. $S(x) = \mathcal{A} + 1 < x + 2 = A(\underline{1}, x)$, also $S \in \mathcal{A}$
3. $\pi_j^k(x_1, \ldots, x_k) = x_j \leq \max \overline{x} < \max \overline{x} + 1 = A(\underline{0}, \max \overline{x})$, also $\pi_j^k \in \mathcal{A}$
4. $$\left.\begin{array}{rl} \text{Sei} & g_1, \ldots, g_m : \mathbb{N}^k \to \mathbb{N} \\ & h : \mathbb{N}^m \to \mathbb{N} \end{array}\right\} \in \mathcal{A}$$
$$\begin{array}{rll} \text{also} & g_i(\overline{x}) < A(r_i, \max \overline{x}) & r_i \exists \text{ nach I.V.} \\ & h(\overline{x}) < A(s, \max \overline{y}) & s \exists \text{ nach I.V.} \end{array}$$

Betrachte $f = h \circ [g_i, \ldots, g_m]$
Wähle j sodass $r_j = \max r_i$
$\curvearrowright\ g_i(x) < A(r_j, \max \overline{x})$

$$\begin{aligned}\text{Dann } f(\overline{x}) &= h(g_1(\overline{x}), \ldots, g_m(\overline{x}) \\ &< A(s, \max(g_1(\overline{x}), \ldots, g_m(\overline{x})) \\ &< A(s, \max(A(r_1, \max \overline{x}), \ldots, A(r_m, \max \overline{x})) \\ &= A(s, A(r_j, \max \overline{x})) \\ &\overset{!}{<} A(\underline{\underline{s + r_j + 2}}, \max \overline{x})\end{aligned}$$

Also $f \in \mathcal{A}$

5. Sei $g : \mathbb{N}^k \to \mathbb{N},\ h : \mathbb{N}[k+2] \to \mathbb{N} \qquad \in \mathcal{A}$

$$\begin{aligned}\curvearrowright \exists r, s : g(\overline{x} &< A(r, \max \overline{x})) \\ h(\overline{y} &< A(s, \max \overline{y}))\end{aligned}$$

Sei $f = PR(g, h)$. Zeige $f \in \mathcal{A}$
Zunächst $\forall n \in \mathbb{N} : f(n, \overline{x}) < A(q, n + \max \overline{x})$, wobei q nicht von n oder $\overline{x}$ abhängt.
Wähle $q = \max(r, s) + 1$.

Induktion über n:

$n = 0 : \qquad f(0, \overline{x}) = g(\overline{x}) < A(r, \max \overline{x}) < A(g, \max x)$

$n \to n + 1 : \qquad f(n+1, \overline{x}) = h(f(n, \overline{x}), n, \overline{x}) < A(s, z) \quad , \ z = \max(f(n, \overline{x}), n, \overline{x})$

$$\begin{aligned}\text{nach I.V.}: \quad f(n, \overline{x}) &< A(g, n + \max \overline{x}) \\ \max(n, \overline{x}) \leq n + \max \overline{x} &< A(q, n + \max \overline{x}) \\ \curvearrowright z &< A(q, n + \max \overline{x}) \\ f(n+1, \overline{x}) &< A(s, z) \\ &< A(s, A(q, n + \max \overline{x})) \\ &\leq A(q-1, A(q, n + \max \overline{x})) \\ &= A(q, A(q, n + 1 + \max \overline{x}))\end{aligned}$$

□ Ind. n

$$\begin{aligned}\text{Nun setze } w = \max(n, \overline{x}) \\ f(n, \overline{x}) < A(q, n + \max \overline{x}) \\ &\leq A(q, 2w) \\ &\overset{!}{<} A(\underline{\underline{q+2}}, w)\end{aligned}$$

Also $f \in \mathcal{A}$ □

Korollar 8.5: Die Ackermannfunktion A ist nicht primitiv rekursiv. ⊕

Beweis: $A \notin \mathcal{A}$ □

Liste der Definitionen

1.1 Def. (Alphabet Σ) . . . 4
1.2 Def. (Wort w über Σ) . . . 4
1.3 Def. (Konkatenation von Wörtern) . . . 4
1.4 Def. (Sprache über Σ) . . . 5
1.5 Def. (Konkatenation von Sprachen) . . . 5
1.6 Def. (Potenzierung von Sprachen) . . . 5
1.7 Def. (Stern-Operator, Abschluss, Kleene-Stern) . . . 6
1.8 Def. (Alternative Definition von Σ^*) . . . 6
2.1 Def. (TM) . . . 10
2.2 Def. (Konfiguration einer TM) . . . 11
2.3 Def. (Rechenschrittrelation) . . . 11
2.4.1 Def. ($R, S \subseteq A \times A$ Relation) . . . 12
2.4.2 Def. ($R \subseteq A \times A$) . . . 13
2.4.3 Def. ($R \subseteq A \times A$ Relation) . . . 13
2.5 Def. (Die von TM $\mathcal{A}$ erkannte Sprache) . . . 14
2.6 Def. (Die von TM $\mathcal{A}$ berechnete Funktion) . . . 14
2.7 Def. (Registermaschine (RM)) . . . 18
2.8 Def. (Semantik der RM bezüglich eines Programms P) . . . 18
3.1 Def. (DEA) . . . 23
3.2 Def. (Erweiterung von δ auf Worte) . . . 23
3.3 Def. (Die durch einen DEA erkannte Sprache) . . . 24
3.4 Def. (Äquivalenz von DFA-Zuständen) . . . 25
3.5 Def. (Äquivalenzklassenautomat) . . . 26
3.6 Def. (Rechtsinvariante Äquivalenzrelation) . . . 27
3.7 Def. (NEA) . . . 33
3.8 Def. (Lauf eines Automaten) . . . 33
3.9 Def. (NFA zu DFA) . . . 33
3.10 Def. (Abgeschlossenheit von $\mathcal{L}$) . . . 34
3.11 Def. (RE(Σ)) . . . 36
3.12 Def. (Semantik eines regulären Ausdrucks) . . . 36
4.1 Def. (Chomsky) . . . 43
4.2 Def. (Ableitungsrelation) . . . 43
4.3 Def. (Chomsky Hierarchie) . . . 44
4.4 Def. (Ableitungsbaum) . . . 46
4.5 Def. (Eindeutigkeit von CFG und CFL) . . . 47
4.6 Def. (CFG in CNF) . . . 48
5.1 Def. (NPDA) . . . 57
5.2 Def. (Menge der Konfigurationen eines NPDA) . . . 58
5.3 Def. (DPDA) . . . 61
6.1 Def. (NTM) . . . 64
6.2 Def. (Eigenschaften einer TM bzgl. Sprachen) . . . 64
6.3 Def. (Laufzeit und Platzbedarf einer TM) . . . 64
6.4 Def. (DTAPE und NTAPE) . . . 65

6.5 Def. (length-lexicographic order) . . . 69
6.6 Def. (Diagonalsprache) . . . 69
6.7 Def. (Halteproblem) . . . 70
6.8 Def. (Spezielles Halteproblem H_ε) . . . 71
6.9 Def. (Reduktion) . . . 73
7.1 Def. (NTIME Klasse) . . . 77
7.2 Def. (Polynom) . . . 77
7.3 Def. (NP Klasse) . . . 77
7.4 Def. (DTIME Klasse) . . . 77
7.5 Def. (Polynominell reduzeduzierbare Sprache $A \preceq_p B$) . . . 77
7.6 Def. (NP-hart und NP-vollständig) . . . 78
7.7 Def. (SAT: Erfüllbarkeitsproblem der AL) . . . 78
7.8 Def. ($3SAT$) . . . 80
7.9 Def. (CLIQUE) . . . 81
8.1 Def. (Schema der primitiven Rekursion) . . . 82
8.2 Def. (PREC primitiv rekursiven Funktionen über $\mathbb{N}$) . . . 82
8.3 Def. (Minimierung) . . . 83
8.4 Def. (Klasse der μ-rekursiven Funktionen) . . . 84
8.5 Def. (Ackermannfunktion) . . . 84

Liste der Sätze

2.1 Satz (Simulation von k-Band TM durch 1-Band TM) . . . 15
Korollar . . . 17
2.2 Satz (Simulation von RM durch TM) . . . 19
2.3 Satz (Simulation von TM durch RM) . . . 20
2.4 Satz (Intuitiv berechenbare Funktionen sind mit TM berechenbar) . . . 21
3.1 Lemma ($\equiv$ ist Äquivalenzrelation) . . . 25
3.2 Satz (Äquivalenzklassenautomat ist wohldefiniert) . . . 26
3.3 Satz (Nerode) . . . 28
3.5 Korollar . . . 29
3.6 Lemma (Pumping Lemma) . . . 30
3.7 Satz (Rabin) . . . 33
3.8 Satz (Abgeschlossenheit von REG) . . . 34
3.9 Satz (Kleene) . . . 37
3.10 Lemma (Arden's Lemma) . . . 38
3.11 Satz (Wortproblem) . . . 40
3.12 Satz (Leerheitsproblem) . . . 40
3.13 Satz (Endlichkeitsproblem) . . . 40
3.14 Satz (Schnittproblem) . . . 41
3.15 Satz (Äquivalenzproblem) . . . 41
3.16 Satz (Inklusionsproblem) . . . 41
4.1 Satz (Typ-3 Sprache ist regulär) . . . 45
4.2 Lemma . . . 45

4.3 Satz (Pumping lemma für CFL, uvwxy Lemma) . . . 50
4.4 Lemma . . . 51
4.5 Satz (Wortproblem für CFL entscheidbar) . . . 52
4.6 Satz (Entscheidbarkeit des Leerheitsproblems für CFL) . . . 54
4.7 Satz (Entscheidbarkeit des Endlichkeitsproblem für CFL) . . . 54
4.8 Satz . . . 54
4.9 Satz . . . 55
4.10 Satz ($L \subseteq R$ entscheidbar) . . . 56
5.1 Satz . . . 58
5.2 Lemma . . . 60
5.3 Lemma (DPDA, der gesamte Eingabe verarbeitet) . . . 61
5.4 Satz (Abgeschlossenheit der deterministischen CFL) . . . 62
5.5 Satz . . . 62
5.6 Satz . . . 62
5.7 Satz . . . 63
5.8 Satz . . . 63
6.1 Satz (Zu jeder NTM gibt es DTM) . . . 64
6.2 Satz ($L \in \mathrm{DTAPE}(n),\ L \in \mathrm{NTAPE}(n)$) . . . 65
6.3 Satz . . . 65
6.4 Satz . . . 67
6.5 Satz . . . 67
6.6 Satz . . . 67
6.7 Satz (Abgeschlossenheit von Typ-0 Sprachen) . . . 67
6.8 Satz . . . 69
6.9 Satz ($\leq$: totale Ordnung) . . . 69
6.10 Satz (D ist nicht entscheidbar) . . . 70
6.11 Korollar . . . 70
6.12 Lemma ($\overline{D}$ ist semi-entscheidbar) . . . 70
6.13 Satz ($L, \overline{L}$ semi-entscheidbar $\Rightarrow L$ entscheidbar) . . . 70
6.14 Satz (H ist unentscheidbar) . . . 70
6.15 Satz (H ist semi-entscheidbar) . . . 71
6.16 Korollar . . . 71
6.17 Satz (H_ε ist unentscheidbar) . . . 71
6.18 Satz (Satz von Rice) . . . 71
6.19 Satz (Eigenschaften von Entscheidbarkeit) . . . 72
6.20 Satz (Eigenschaften von Semi-Entscheidbarkeit) . . . 72
6.13 Satz (Wiederholung) . . . 72
6.21 Satz . . . 72
6.22 Lemma . . . 73
6.23 Lemma (MPCP $\preceq$ PCP) . . . 74
6.24 Lemma (H $\preceq$ MPCP) . . . 74
6.25 Satz (PCP ist unentscheidbar.) . . . 75
6.26 Satz . . . 75
6.27 Korollar . . . 76
6.28 Satz . . . 76

6.29 Satz . . . 76
7.1 Lemma . . . 77
7.2 Lemma ($\preceq_p$ ist reflexiv und transitiv) . . . 78
7.3 Satz . . . 78
7.4 Satz (Cook) . . . 78
7.5 Lemma . . . 79
7.6 Satz ($3SAT$ ist NP-vollständig) . . . 80
7.7 Satz (CLIQUE ist NP-vollständig) . . . 81
8.1 Lemma ($PR(g,h)$ ist wohldefiniert.) . . . 82
8.2 Satz (μ-Funktionen-Klasse $\hat{=}$ Klasse der TM-berechenbaren Fkt.) . . . 84
8.3 Satz (Kleene) . . . 84
8.4 Satz . . . 84
8.5 Korollar . . . 85

Abbildungsverzeichnis

1 Turingband . . . 8
2 Bsp.: Turingmaschine . . . 9
3 vqw-Band . . . 11
4 Mehrspurmachine . . . 15
5 Registermaschine . . . 17
6 TM Simulation durch RM . . . 20
7 Endliches Band . . . 22
8 Automat zu L . . . 23
9 Endlicher Automat, der die reguläre Sprache L erkennt. Sobald mehr als eine 1 gelesen wurde, wird in den Zustand q_2, eine sogenannte Senke, gewechselt, von der alle ausgehenden Kanten in sich selbst enden. . . . 24
10 Automat zu Bsp. 3.4 . . . 25
11 Automat zu (3.1) . . . 26
12 Automat: Pumping Lemma . . . 29
13 DFA für L . . . 31
14 Bsp.: Mustererkennung . . . 32
15 Potenzmengenkonstruktion auf dem NFA . . . 32
16 Nichtdet. Automat für L' . . . 32
17 Nichtdet. Automat für L_n . . . 32
18 NFA für Vereinigung . . . 35
19 Informell vom Automaten zum regulären Ausdruck für mod 3 . . . 37
20 DFA „modulo 3“ . . . 39
21 Ableitungsbäume zu Bsp. 4.3 . . . 47
22 Schema zu Satz 4.3 . . . 51
23 Beweis zu Satz 5.1 . . . 59

Abkürzungsverzeichnis

AL Aussagenlogik

CFL Menge der kontextfreien Sprachen

CFG Menge der kontextfreien Grammatiken

CNF Chomsky Normalform

CP Korrespondenzproblem

CYK Cocke, Younger, Kasami

DAG gerichteter azyklischer Graph

DCFG deterministische CFG

DCFL deterministische CFL

DEA deterministischer endlicher Automat

DFA engl.: deterministic finite automaton

DPDA deterministischer Kellerautomat

DTM deterministische TM

EA endlicher Automat

LBA Linear Bounded Automaton

MPCP Das modifizierte PCP

ND Nicht-Determinismus

NEA nichtdeterministischer endlicher Automat

NFA engl.: nondeterministic finite automaton

NPDA nichtdeterministischer Kellerautomat

NT Nichtterminal

NTM Eine nichtdeterministische TM

PCP Das Postsche Korrespondenzproblem

PDA pushdown automaton (Kellerautomat)

PL Pumping Lemma

RE Menge der regulären Ausdrücke

REG Menge der regulären Sprachen

RM Registermaschine

TM Turing-Maschine

TT Turingtabelle

Anmerkungsverzeichnis

Vorlesung: 23.10.15 4
Vorlesung: 28.10.15 9
Vorlesung: 30.10.15 12
Vorlesung: 04.11.15 15
Vorlesung: 06.11.15 19
Vorlesung: 11.11.15 22
Vorlesung: 13.11.15 25
Vorlesung: 18.11.15 28
Vorlesung: 25.11.15 31
Vorlesung: 27.11.15 35
Vorlesung: 02.12.15 39
RL: Beweis vollständig? 40
Vorlesung: 04.12.15 41
Vorlesung: 09.12.15 45
Vorlesung: 11.12.15 49
Vorlesung: 18.12.15 52
Vorlesung: 21.12.15 56
Vorlesung: 22.12.15 60
Vorlesung: 08.01.16 64
PT: checkpoint 64
Vorlesung: 13.01.16 66
Vorlesung: 14.01.16 69
Vorlesung: 20.01.15 71
Vorlesung: 28.01.16 74
Vorlesung: 29.01.15 77
Vorlesung: 03.02.15 79
RL: Grafik überprüfen 81
Vorlesung: 10.02.16 82
RL: URM = unbeschränkte Registermaschine? 84