BFS | / freie Übungsgruppenwahl, 4er-Gruppen, EST Pw: bfs2017

Datum — Aufzeichnung: User BFS — 1x Vorrechnen

BW ~~FAU~~ BFS2011w — Abgabe bis Donnerstag

Name/Projekt — max. 3 Bonuspunkte

Theoretische Informatik - Eine Einführung in die algorith... WS

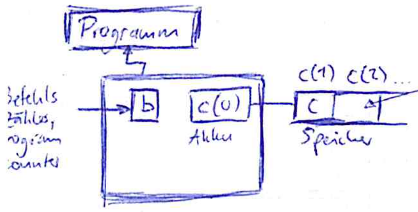
Was ist Rechnen? Mehr unberechenbare Fkt. als berechenbare

→ Was geht und was geht nicht mit welchen Ressourcen?

→ Erkennung und Erzeugung der Objekte aus unendlich großen Mengen mit endlichen Mitteln

Registermaschinen (Random Access Machine, RAM) So RAMs sind äquivalent zu "normalen" Computern

„Maschine mit wahlfreiem Speicherzugriff"



potentiell ∞ viele Speicherzellen

Kann eine bel. nat. Zahl aus $\mathbb{N}$ enthalten

Eingabe steht in $c(1), \ldots, c(?)$, endlich lang, Rest ist mit 0 gefüllt

LOAD i	$c(0)=c(i)$	$b=b+1$
STORE i	$c(i)=c(0)$	"
ADD i	$c(0)=c(0)+c(i)$	"
SUB i	$c(0)=c(0)-c(i)$	"
MULT i	$c(0)=c(0)\cdot c(i)$	"
DIV i	$c(0)=\lfloor c(0)/c(i)\rfloor$	"
GOTO j		$b=j$
if $c(0)$? l GOTO j		

mit Konstanten:

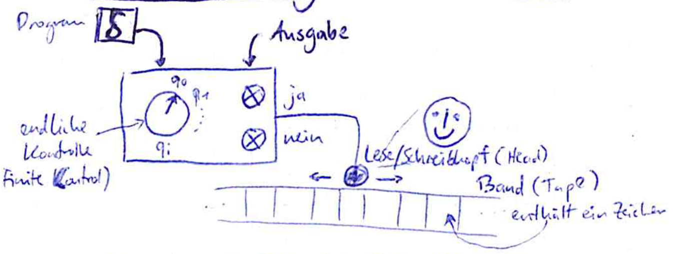
CLOAD l	$c(0)=l$	$b=b+1$
CADD l	$c(0)=c(0)+l$	"
CSUB l	$c(0)=c(0)-l$	"
CMULT l	$c(0)=c(0)\cdot l$	"
CDIV l	$c(0)=\lfloor c(0)/l\rfloor$	"

indirekter Zugriff

INDLOAD i	$c(0)=c(c(i))$	"
INDSTORE i	$c(c(i))=c(0)$	"
INDADD i	$c(0)=c(0)+c(c(i))$	"
INDSUB i	$c(0)=c(0)-c(c(i))$	"
INDMULT i	$c(0)=c(0)\cdot c(c(i))$	"
INDDIV i	$c(0)=\lfloor c(0)/c(c(i))\rfloor$	"

1. Einführung in die Berechenbarkeit

1.1 Die Turing-Maschine (TM)



Def 1.1 Eine deterministische 1-Band-Turingmaschine besteht aus
- Q: endliche Zustandsmenge $Q=\{q_0,\ldots,q_n\}$
- Σ: Endliches Eingabealphabet
- Γ: endliches Bandalphabet mit $\Sigma \subsetneq \Gamma$
- B: Blanksymbol (in jeder Zelle wo kein Zeichen drinsteht, ist B), $B\in\Gamma$, $B\notin\Sigma$
- q_0: $q_0\in Q$, Start- (Anfangs) Zustand
- F: $F\subseteq Q$ akzeptierenden Endzustände (können alle sein!)
- δ: $Q\times\Gamma \to Q\times\Gamma\times\{R,L,N\}$ = Richtungen des Kopfes — Programm der Turingmaschine

Arbeitsweise der TM M (7-Tupel)

Eingabe: $w=\underbrace{w_1\ldots w_n}_{\text{Wort aus } n \text{ Zeichen aus } \Sigma}\in\Sigma^*$ (leeres Wort: ε, Länge von ε ist 0)

Der Lese-Schreib-Kopf steht auf dem ersten Zeichen von w, außer $w=\varepsilon$. Rest vom Band ist mit B gefüllt

$L\subseteq\Sigma^*$, L ist eine Sprache über Σ, L besteht aus den Wörtern z.B. $L=\{0^n1^n \mid n\geq 1\}$ und wird durch M entschieden

Wortproblem gemeint: ist $w\in L$? Konfiguration: aktueller Dump unseres Programms (Kopf-pos, Zustand)

Konfiguration: K ist ein Element aus $\Gamma^*\times Q\times\Gamma^*$, z.B. $\underbrace{00}_{\alpha}q_1\underbrace{0111}_{\beta}$ bedeutet ...BB000111BB... Wir sagen: „TM M ist aktuell in der Konfiguration $\alpha q\beta$"

Startkonfiguration $q_0 w$, akzeptierende Endkonfiguration: $\alpha q\beta$ mit $q\in F$ und α,β bel. Auf dem Band steht $\alpha\beta$, Kopf auf erstem El. von β und Zustand ist q

Sei $\beta=b\bar\beta$, $b\in\Gamma$, $\beta\in\Gamma^*$, $\alpha=\bar\alpha a$, $\alpha\in\Gamma^*$, $a\in\Gamma$. „$\alpha'q'\beta'$ ist direkte Nachfolgekonfiguration von $\alpha q\beta$", falls gilt:

$\delta(q,b)=(q',b',s)$, $s\in\{N,L,R\}$ → $s=N$: $\alpha'=\alpha$, $\beta'=b'\bar\beta$ Schreibweise: $\alpha q\beta \vdash \alpha'q'\beta'$

$s=L$: $\alpha'=\bar\alpha$, $\beta'=ab'\bar\beta$

$s=R$: $\alpha'=\bar\alpha a b'$, $\beta'=\bar\beta$

$\alpha'q'\beta'$ ist die i-te Nachfolgekonfiguration, falls gilt:
- $i=0$: $\alpha q\beta=\alpha'q'\beta'$
- $i>0$: $\exists$ Konfigurationen $K_1,K_2,\ldots,K_{i-1}$ sodass $\alpha q\beta\vdash K_1\vdash K_2\vdash\ldots\vdash K_{i-1}\vdash\alpha'q'\beta'$

Kurzschreibweise: $\alpha q\beta \overset{i}{\vdash} \alpha'q'\beta'$

$\alpha'q'\beta'$ ist eine Nachfolgekonfiguration von $\alpha q\beta$, falls es ein i gibt, sodass $\alpha q\beta\overset{i}{\vdash}\alpha'q'\beta'$. Schreibweise $\alpha q\beta\overset{*}{\vdash}\alpha'q'\beta'$

M ist TM, $x\in\Sigma^*$ Eingabe. M akzeptiert $x\in\Sigma^*$, falls es $\alpha,\beta\in\Gamma^*$ und $q\in F$ gibt mit $q_0x\overset{*}{\vdash}\alpha q\beta$

Die Sprache L von M (L_M, $L(M)$) ist die Menge aller von M akzeptierten Wörter $x\in\Sigma^*$.

Wenn wir $q\in F$ erreicht haben, gibt es keine weitere Nachfolgekonfiguration

Stärkerer Begriff (was zu beweisen ist): Falls M die Sprache L akzeptiert und $\forall x\in\Sigma^*$ nach endlich vielen Schritten nicht mehr weiterrechnen kann (also hält), so entscheidet M die Sprache L

Def. 1.3
- $L\subseteq\Sigma^*$ heißt rekursiv aufzählbar (r.a., engl. r.e.), gdw es eine det. 1-Band-TM M gibt, die L akzeptiert
- $L\subseteq\Sigma^*$ heißt entscheidbar (oder rekursiv) gdw es eine det. 1-Band-TM M gibt, die L entscheidet

Def. 1.4 Eine det. 1-Band-TM berechnet die partielle Fkt. $f:\Sigma^*\to(\Gamma\setminus\{B\})^*$ mit
- $f(x)=y$, falls $q_0x\overset{*}{\vdash}qy$ und M hält in Konfiguration qy für ein $q\in Q$
- $f(x)$ ist nicht definiert, falls M gestartet mit x nicht hält

M berechnet die totale Funktion $f:\mathbb{N}^r\to\mathbb{N}$, falls $\Sigma=\{0,1,\#\}$ und für jedes $a_1,\ldots,a_r\in\mathbb{N}$ gilt:

$q_0\,\mathrm{bin}(a_1)\#\mathrm{bin}(a_2)\#\ldots\#\mathrm{bin}(a_r)\overset{*}{\vdash} q\,\mathrm{bin}(f(a_1,\ldots,a_r))$ für ein $q\in Q$ und sie hält

1.2 Programmiertechniken und einfache Simulationen

1. Endlicher Speicher („im Zustand merken")

$x_1 \dots x_n \in \Sigma^*$. Frage: Ist x_1 in $x_2, \dots x_n$ enthalten?

$\Gamma = \Sigma \cup \{B\}$, $Q = \{q_0, q_1\} \cup (\{q_0\} \times \Sigma)$, Startzustand q_0, $F = \{q_1\}$

(1) $\delta(q_0, a) = ((q_0, a), a, R)$ für jedes $a \in \Sigma$

(2) $\delta((q_0, a), a) = (q_1, a, N)$ für jedes $a \in \Sigma$

(3) $\delta((q_0, a), b) = ((q_0, a), b, R)$ für alle $a, b \in \Sigma$ mit $a \neq b$

2. Spurtechnik

Blank = ⊔

⊔	U	N	I	V	E	R	S	I	T	Ä	T	#	F	Ü	R	T	H	⊔
	E	R	L	A	N	G	E	N	⊔	⊔	⊔		?	?	?	?	?	
	N	Ü	R	N	B	E	R	G	⊔	⊔	⊔							

$\Gamma = \{A, \dots, Z, ⊔\}^3 \cup \{A, \dots, Z, ⊔\}^2 \cup \{A, \dots, Z\} \cup \{⊔\} \cup \{\#\}$

Def 1.5: Eine det. k-Band-TM hat k Bänder und k Köpfe (je Band einen). Die Eingabe steht auf Band 1

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

Die Arbeitsweise ist analog einer det. 1-Band-TM.

Bsp: $L = \{w\#w^R \mid w \in (0,1)^*\}$ w^R ist gespiegeltes Wort, $\Theta(n^2)$ auf 1 Band: $|a_1 \dots a_k | \# | a_k \dots a_1|$

$\Theta(n)$ auf 2 Bändern: $|a_1 \dots a_k | \# | a_k \dots a_1|$, $|a_1 \dots a_k | \#|$

Def 1.6 M sei eine det. k-Band-TM, $x \in \Sigma^*$

Zeitkomplexität $T_M(x) = \#$ Schritte (= Konfigurationsübergänge), die M gestartet mit x ausführt

Platzkomplexität $S_M(x) = \#$ verschiedene Zellen, die M gestartet mit x besucht. Es gilt immer: $T_M(x) \geq S_M(x)$

$T_M(n) = \max\{T_M(x) \mid x \in \Sigma^*, |x| \leq n\}$

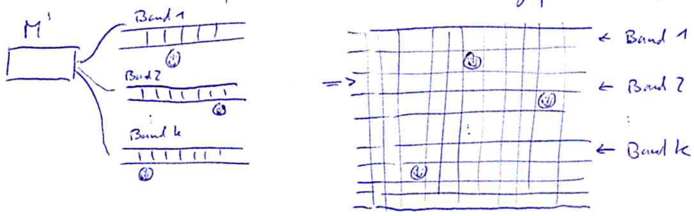
$S_M(n) = \max\{S_M(x) \mid x \in \Sigma^*, |x| \leq n\}$

$t, s: \mathbb{N} \to \mathbb{N}$

M ist $t(n)$-zeitbeschränkt und $s(n)$-platzbeschränkt, falls $\forall n \in \mathbb{N}$ gilt: $T_M(n) \leq t(n)$ und $S_M(n) \leq s(n)$

Satz 1.7 Jede $t(n)$-zeit- und $s(n)$-platzbeschränkte k-Band-TM M' kann durch eine $O(t(n) \cdot s(n))$-zeit- und $O(s(n))$-platzbeschränkte det. 1-Band-TM M simuliert werden

Bew: eine Schritt-für-Schritt-Simulation wird ausgeführt. $M' = (Q', \Sigma', \Gamma', \delta', q_0', F')$ k-Band-TM. Wir benutzen die Spurtechnik



Korollar 1.8. Det. 1-Band-TM und det. k-Band-TM liefern exakt die gleichen Mengen von entscheidbaren Sprachen, rekursiv aufzählbaren Sprachen und berechenbaren Fkt.

3. Unterprogramme

Wenn wir eine Turingmaschine „programmieren", können wir sagen: Wir benutzen an dieser Stelle ein Unterprogramm, das eine bestimmte Aufgabe löst, die bereits von einer anderen TM gelöst wird.

1.3 Simulationen zw. RAMs und TMs

Def. 1.9 Für eine RAM und eine Eingabe x ist $T_M(x) = \#$ Schritte von M gestartet mit x

$S_M(x) =$ größte genutzte Speicheradresse bei der Berechnung von M gestartet mit x

„$t(n)$-zeitbeschränkt" und „$s(n)$-platzbeschränkt" sind analog zu den TM definiert

(das ist ein sog. „uniformes" Kostenmaß, bei Beachtung der Anzahl der involvierten Bits spricht man von einem logarithmischen Kostenmaß

Satz 1.10 Jede RAM kann durch eine det. TM simuliert werden. Ist die RAM $t(n)$-zeitbeschränkt, so ist die TM $O((t(n))^3)$-zeitbeschränkt.

Bew. a) Wir müssen uns überlegen, wie die simulierende TM eine Konfiguration der RAM auf (mehreren) Bändern speichert

b) Die TM muss Konfigurationsübergänge der RAM realisieren

Zu a) $c(1), \dots$ enthalten die Eingabe, Inhalte aus $\mathbb{N}_0$. Konfiguration einer RAM: Speicherinhalt $c(0), c(1), \dots$, Programmzähler b

TM: $|\#| 0 |\#| \mathrm{bin}(c(0)) |\#| \dots |\#| \mathrm{bin}(i) |\#| \mathrm{bin}(c(i)) |\#| \dots$

Adresse 0 — Inhalt von $c(0)$ in Binärdarstellung — Adresse i — Inhalt von $c(i)$

Programmzählerband: $\dots |\#| \mathrm{bin}(b) |\#| \dots$

Arbeitsband: $| \; | \; | \; | \dots$

3FS | Eva Pegler

Datum

Name/Projekt

b) Nun müssen wir für jeden möglichen RAM-Befehl unseres RAM-Befehlssatzes ein TM-Unterprogramm schreiben, das diesen Befehl realisiert

zB. ADD i $(c(0) = c(0) + c(i))$
- Suche bin(i) auf dem Speicherband im „Adressfeld" (inclusive Handling von c(i) noch nicht beschrieben)
- kopiere bin (c(i)) aufs Arbeitsband
- addiere bin (c(0)) auf bin (c(i))
- kopiere den Inhalt des Arbeitsbandes auf das Speicherband → So gehts für alle RAM-Befehle
- addiere 1 auf dem Befehlszähler

Satz 1.11 Jede $t(n)$-zeitbeschränkte det. 1-Band TM kann durch eine $O(t(n))$-zeitbeschränkte RAM simuliert werden

RAM, det. 1-Band-TM, det. k-Band-TM, Halbband-TM, 2D-Band-TM beschreiben exakt dieselbe Klasse von Berechenbarkeit.

1.4 Die Church-Turing-These

Die im intuitiven Sinn berechenbaren Fkt. sind genau die, die durch Turing-Maschinen berechenbar sind
So wie es stetige Fkt gibt, und solche, die es nicht sind, gibt es Fkt, die berechenbar sind, und solche, die es nicht sind

1.5 Universelle TM

Bislang: Unsere TM können genau eine Aufgabe lösen („special purpose computer")

Darstellung von det. 1-Band-TM M „in maschinenlesbarer Form"

ObdA: $\Gamma = \{0, 1, B\}$, $Q = \{q_1, .. q_n\}$, Startzustand q_1, $F = \{q_n\}$

Diese TM soll man durch $\{0,1,\#\}$ dargestellt, kodiert werden

$\delta(q_i, X) = (q_j, Y, D)$ $X, Y \in \Gamma$ wird kodiert wie folgt: $\begin{matrix} 0 & 00 \\ 1 & 01 \\ B & 11 \end{matrix}$, $D \in \{R, L, N\}$: $\begin{matrix} R & 00 \\ N & 01 \\ L & 11 \end{matrix}$

q_i wird durch 1^i kodiert → $\underbrace{111}_{q_3}\#\underbrace{01}_{1}\#\underbrace{11}_{q_2}\#\underbrace{11}_{B}\#\underbrace{11}_{L} \hat{=} \text{code}(\delta(q_3, 1) = (q_2, B, L))$

Die gesamte δ-Tabelle kann folgendermaßen kodiert werden:

$\#\#\#\underbrace{1^n}_{\text{Endzustand } q_n}\#\#$ code (erster Eintrag d. δ-Tabelle) ## code (zweiter Eintrag...) ## ... ## code (letzter Eintrag...) ###

„Gödelnummer" der TM M, Schreibweise <M>

Wenn man nun noch $\begin{matrix}0\\1\\\#\end{matrix}$ durch $\begin{matrix}00\\01\\11\end{matrix}$ kodiert, erhält man eine lange 0-1-Folge, die zB als Binärzahl kodiert werden kann.

Man kann also jedes „TM-Programm" auf eine nat. Zahl abbilden.

Eine TM $\tilde{M}$ heißt universell, wenn sie sich bei Eingabe $\langle M\rangle x$, $x \in \{0,1\}^*$, so verhält wie M gestartet mit x (für jede TM M)

Satz 1.12 Es gibt eine universelle 2-Band-TM, die jede $t(n)$-zeit- und $s(n)$-platzbeschränkte det. 1-Band-Turingmaschine M auf Platz $O(s(n))$ in Zeit $O(t(n))$ simuliert, falls M mit entsprechender Eingabe hält.

Bew: M hat Bandalphabet $\Gamma = \{0, 1, B\}$

$\tilde{M}$: Auf Band 1 steht $\langle M\rangle x$ - Kopiere <M> auf Band 2, lösche <M> von Band 1 $|\langle M\rangle| = O(1)$

jetzt: B | B | x1 | ... | xn | B | B ..   Band 1

|##| |###| .... |###| Band 2

akt. Zustand $\underbrace{\quad\quad\quad}_{\langle M\rangle}$

- Finde zu akt. Zustand q_i und Zeichen X auf Band 1 die Stelle $1^i\#\text{code}(X)\#1^j\#\text{code}(Y)\#\text{code}(D)$ ← Zeit: $O(|\langle M\rangle|^2) = O(1)$, Platz: $O(|\langle M\rangle|) = O(1)$
- Aktualisiere auf Band 2 den aktuellen Zustand zu 1^j, ersetze auf Band 1 das X durch Y und gehe auf Band 1 in Richtung D ← Zeit: $O(|\langle M\rangle|+1) = O(1)$, Platz: $O(|\langle M\rangle|+1) = O(1)$

Gesamtzeit und Gesamtplatz ergeben sich damit direkt.
Handling beim Halten (akzeptierend oder verwerfend) entsprechend □.

Ob eine Zeichenfolge eine berechnete Gödelnummer ist, kann (einfach) entschieden werden → „Syntaxanalyse"

1.6 Unentscheidbare Probleme

Def. 1.13 Die Sprache $H := \{\langle M\rangle w \mid M$ ist det. 1-Band-TM, die, gestartet mit w, hält$\}$ ist das allgemeine Halteproblem

Satz 1.14 (Turing, 1936) H ist unentscheidbar

Bew: Wir nehmen an, dass H entscheidbar ist, d.h. es gibt eine det. 1-Band-TM M_H, die H entscheidet

Also: M_H gestartet mit einer beliebigen Eingabe x hält immer, und am erreichten Zustand ist erkennbar, ob $x \in H$ ist oder nicht

Wenn es M_H gibt, dann auch die folgende Turingmaschine M_{schlau}

TM M_{schlau}:
(1) Eingabe sei y
(2) falls $y = \langle M\rangle$, dann...
(3) entscheide mittels M_H, ob $\langle M\rangle\langle M\rangle \in H$
(4) falls ja, schreibe unendlich viele 1en aufs Band
(5) falls nein, bleibe stehen

M_{schlau} kann programmiert werden, falls es die δ-Tabelle von M_H gibt

Was passiert, wenn M_{schlau} mit $\langle M_{schlau}\rangle$ gestartet wird?

M_{schlau} gestartet mit $\langle M_{schlau}\rangle$ kann halten („Fall a") oder M_{schlau} gestartet mit $\langle M_{schlau}\rangle$ hält nicht („Fall b").

(a) M_{schlau} gestartet mit $\langle M_{schlau}\rangle$ hält $\Rightarrow$ (5).

$\Rightarrow \langle M_{schlau}\rangle\langle M_{schlau}\rangle \notin H \Rightarrow M_{schlau}$ gestartet mit $\langle M_{schlau}\rangle$ hält nicht $\lightning$

(b) M_{schlau} gestartet mit $\langle M_{schlau}\rangle$ hält nicht $\Rightarrow$ (4)

$\Rightarrow \langle M_{schlau}\rangle\langle M_{schlau}\rangle \in H \Rightarrow M_{schlau}$ gestartet mit $\langle M_{schlau}\rangle$ hält $\lightning$

(a) + (b) $\Rightarrow$ M_H gibt es nicht $\Rightarrow$ H ist nicht entscheidbar □.

Satz 1.15 (a) H ist rekursiv aufzählbar
(b) $\overline{H} = \{0,1\}^* \setminus H$ ist nicht rekursiv aufzählbar

Bew: (a) die universelle TM $\tilde{M}$ ist das „rekursive Aufzählen" von H, dh $\langle M\rangle w \in H \Leftrightarrow \tilde{M}$ gestartet mit $\langle M\rangle w$ hält

(b) Annh: $\overline{H}$ ist doch rekursiv aufzählbar.
Eingabe sei $\langle M\rangle w$. führe gleichzeitig (zB auf 2 Bändern) aus und stoppe, falls eine der beiden Berechnungen stoppt
- starte auf 1. Band die universelle TM $\tilde{M}$ mit $\langle M\rangle w$
- starte auf Band 2 die TM M' mit $\langle M\rangle w$

} mind. eine der Rechnungen hält, also hält diese TM immer

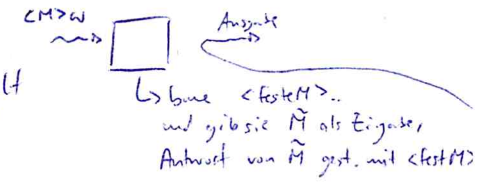
$\Rightarrow$ diese TM entscheidet H $\lightning$ M' gibt es nicht □.

Def 1.16: die Sprache $H_\varepsilon := \{\langle M\rangle \mid M$ ist det. 1-Band-TM, die gestartet mit leerem Band hält$\}$ heißt initiales Halteproblem.

Satz 1.17 H_ε ist nicht entscheidbar.

Bew: Wir programmieren einen Entscheider für H unter der Annahme, dass es einen Entscheider für H_ε gibt.
Geg. ist $\langle M\rangle w$ und die Frage: $\langle M\rangle w \in H$?
Wir nehmen an, dass ganz allgemein die Frage „$\langle M'\rangle \in H_\varepsilon$?" entscheidbar ist $\forall \langle M'\rangle$ (via $\tilde{M}$)

$$\left\langle \begin{array}{l} \text{feste\_Maschine}_{\langle M\rangle w} \\ \quad (1)\ \text{die Eingabe sei } x \\ \quad (2)\ \text{starte } M \text{ mit } w \\ \quad (3)\ \text{falls } x = \varepsilon \text{ dann halte} \end{array} \right\rangle \in H_\varepsilon \Leftrightarrow M \text{ gestartet mit } w \text{ hält} \Leftrightarrow \langle M\rangle w \in H$$



Wir nehmen an, dass H_ε mittels TM $\tilde{M}$ entschieden werden kann
a) wenn $\tilde{M}$ gestartet mit $\langle \text{feste\_Maschine}_{\langle M\rangle w}\rangle$ akzeptierend hält
$\Rightarrow \langle \text{feste\_Maschine}_{\langle M\rangle w}\rangle \in H_\varepsilon \Rightarrow$ (2) wird haltend durchlaufen $\Rightarrow \langle M\rangle w \in H$
b) wenn $\tilde{M}$ gestartet mit $\langle \text{feste\_Maschine}_{\langle M\rangle w}\rangle$ verwerfend hält
$\Rightarrow \langle \text{feste\_Maschine}_{\langle M\rangle w}\rangle \notin H_\varepsilon \Rightarrow$ (3) wird nie erreicht $\Rightarrow \langle M\rangle w \notin H$
D.h. die Antwort auf „$\langle M\rangle w \in H$?" ist die gleiche wie die auf die Frage „$\langle \text{feste\_Maschine}_{\langle M\rangle w}\rangle \in H_\varepsilon$".
$\rightarrow$ da H unentscheidbar ist, kann es $\tilde{M}$ nicht geben!

1.7 Reduktionen und der Satz von Rice

Def. 1.18 Eine Fkt. $f: \{0,1\}^* \to \{0,1\}^*$ heißt berechenbar, wenn es eine det. 1-Band-TM M_f gibt, für die mit $x \in \{0,1\}^*$ gilt:
- Ist $f(x)$ definiert, so hält M_f gestartet mit x, und $f(x)$ steht auf dem Band
- Ist $f(x)$ nicht definiert, so hält M_f gestartet mit x nicht.

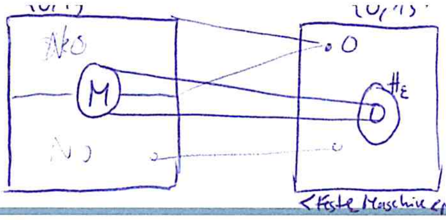
Ist f total, d.h. $\forall x \in \{0,1\}^*$ definiert und berechenbar, so heißt f total berechenbar (/rekursiv)
Eine Sprache L ist genau dann entscheidbar, wenn ihre charakteristische Fkt
$\chi_L: \{0,1\}^* \to \{0,1\}$ mit $\chi_L(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases}$ total berechenbar ist.
Eine Sprache L ist genau dann rek. aufzählbar, wenn ihre (partielle) charakteristische Fkt.
$\chi_L'(x) = \begin{cases} 1 & x \in L \\ \text{undef.} & x \notin L \end{cases}$

Def 1.19 Seien L_1 und L_2 Sprachen über dem Alphabet $\{0,1\}$.
Eine Reduktion ist eine total berechenbare Fkt. $f: \{0,1\}^* \to \{0,1\}^*$ für die gilt:
$x \in L_1 \Leftrightarrow f(x) \in L_2$
Wir schreiben „$L_1 \leq L_2$" und sagen „L_1 wird mittels f auf L_2 reduziert"

$\rightarrow$ $H \leq H_\varepsilon$ mittels f mit $f(x) = \begin{cases} \langle \text{feste\_Maschine}_{\langle M\rangle w}\rangle & \text{falls } x \text{ ist in der Form } \langle M\rangle w \\ 0 & \text{sonst} \end{cases}$

1. f total berechenbar ✓
2. $x \in H \Leftrightarrow f(x) \in H_\varepsilon$ $\rightarrow$ $x \in H \Rightarrow x = \langle M\rangle w$ und M gestartet mit w hält $\Rightarrow f(x) = \langle \text{feste}M\ldots\rangle$ & $f(x) \in H_\varepsilon$

$x \notin H \Rightarrow \begin{cases} x = \langle M\rangle w \text{ und } M \text{ gest. mit } w \text{ hält nicht} \Rightarrow f(x) \notin H_\varepsilon \\ x \neq \langle M\rangle w \Rightarrow f(x) = 0 \notin H \end{cases}$



$f(H) \subseteq H_\varepsilon$
$f(\overline{H}) \subseteq \overline{H}_\varepsilon$

feste Maschine $_{<M>w}$
die Eingabe sei y
starte M mit w
falls $y = \varepsilon$ then stop

falls $<M>w \in \overline{H}$, dann ist $<\text{feste Maschine}_{<M>w}> \in \overline{H}_\varepsilon$

Satz 1.20 Seien L_1 und L_2 Sprachen, und sei $L_1 \leq L_2$ mittels f
(a) Ist L_2 entscheidbar, so ist auch L_1 entscheidbar
(b) Ist L_2 rekursiv aufzählbar, dann auch L_1

Bw: (a) Geg. sei die Frage „$x \in L_1$?" und ein Entscheider M_{L_2} für L_2
entscheidbar für L_1: berechne $f(x)$, gib $f(x)$ in M_{L_2} und gibt deren Antwort aus
(b) Analog, der „Nein"-Fall muss in einer Endlosschleife enden □.

Korollar 1.21 $L_1 \leq L_2$
(a) L_1 unentscheidbar => L_2 unentscheidbar
(b) L_1 nicht rekursiv aufzählbar => L_2 nicht r.a.

$L = \{<M> \mid$ es gibt mind. eine Eingabe w für M, so dass M gestartet mit w hält$\}$ ist unentscheidbar
und $w \neq 01$

-> zeigen durch Reduktion. Wir zeigen: $H \leq L$
Sei f

$$f(x) = \begin{cases} <\text{feste\_Maschine}_{<M>w}> & \text{falls } x \text{ ist v.d. Form } <M>w \\ 0 & \text{sonst} \end{cases}$$

feste Maschine $_{<M>w}$
die Eingabe sei y
starte M mit w
falls $y = 101$ dann stop, sonst endlos

-> f ist total ✓ und berechenbar, da die Form $<M>w$ durch Syntaxanalyse erkennbar ist

$x \in H$ => $x = <M>w$ und M gest. mit w hält. => $f(x) = <\text{feste\_Maschine}_{<M>w}>$ und M gest. m. w hält
=> feste_Maschine$_{<M>w}$ hält für Eingabe 101 => $f(x) = <\text{feste\_Maschine}_{<M>w}> \in L$

$x \in \overline{H}$ => $\begin{cases} x \text{ ist nicht von der Form } <M>w \Rightarrow f(x) = 0 \notin L \\ x = <M>w \text{ und M gest. mit w hält nicht} \Rightarrow \text{feste Maschine}_{<M>w} \text{ hält für keine Eingabe} \\ \Rightarrow f(x) = <\text{feste\_Maschine}_{<M>w}> \notin L \end{cases}$

Also: $x \in H \Leftrightarrow f(x) \in L$

$\mathcal{R} = \{ f \mid f: \{0,1\}^* \to \{0,1\}^* \text{ berechenbar} \}$ sei die Menge der berechenbaren Fkt.
Sei S, $S \subseteq \mathcal{R}$. Bezeichne $Prog(S) = \{<M> \mid M$ berechnet die Fkt $f \in S\}$

$Prog(\mathcal{R})$ ist Menge aller Gödelnummern

$Prog(\mathcal{R})$ ist die Menge aller Gödelnummern, also ist $Prog(\mathcal{R})$ entscheidbar (wg. Syntaxanalyse)
$Prog(\emptyset)$ ist entscheidbar. Daher nennt man die Eigenschaft von x in $Prog(\mathcal{R})$ zu sein, trivial

Satz 1.22 (Satz von Rice, 1933)
Sei S, $S \subseteq \mathcal{R}$, mit $\emptyset \neq S \neq \mathcal{R}$. Dann ist $Prog(S)$ nicht entscheidbar.
-> Alle nicht trivialen Fragen über Programme sind unentscheidbar.

Bew: Sei u die für keine Eingabe definierte Fkt. u ist berechenbar durch diese ganz konkrete det. 1-Band-TM M_u die sofort unendlich oft nach rechts läuft, unabhängig vom Bandinhalt. Also $u \in \mathcal{R}$ und $<M_u> \in Prog(\{u\})$.
$Prog(\{u\})$ ist unendlich groß. Entweder $u \notin S$ oder $u \in S$

a) $u \notin S$ und damit $<M_u> \notin Prog(S)$. Wir zeigen $H_\varepsilon \leq Prog(S)$. Da $S \neq \emptyset$, gibt es eine Fkt $s \in S$.
s ist durch konkrete det. 1-Band-TM M_s mit $<M_s> \in Prog(S)$ berechnet. Nun schreiben wir ein weiteres Programm:

Drumherum$_{<M>}$
Eingabe sei y
starte M mit ε auf Hilfsband
starte M_s mit y

Drumherum$_{<M>}$ berechnet s, falls M gestartet mit ε hält
Drumherum$_{<M>}$ berechnet u, falls M gest. mit ε nicht hält

Reduktionsfkt f: $f(x) = \begin{cases} <\text{Drumherum}_{<M>}> & \text{falls } x = <M> \\ <M_u> & \text{sonst} \end{cases}$

f ist total und berechenbar, da wir nur Syntaxanalyse zu tun ist.

$x \in H_\varepsilon$ => $x = <M>$ und M gest. mit ε hält => $f(x) = <\text{Drumherum}_{<M>}>$ und Drumherum$_{<M>}$ berechnet s
=> $f(x) = <\text{Drumherum}_{<M>}> \in Prog(S)$

$x \notin H_\varepsilon$ => $\begin{cases} x = <M> \text{ und M gest. mit } \varepsilon \text{ hält nicht} \Rightarrow f(x) = <\text{Drumherum}_{<M>}> \text{ und Drumherum}_{<M>} \text{ berechnet u} \\ \quad \Rightarrow f(x) = <\text{Drumherum}_{<M>}> \notin Prog(S) \text{ da } u \notin S \\ x \neq <M> \Rightarrow f(x) = <M_u> \notin Prog(S) \end{cases}$

Also $x \in H_\varepsilon \Leftrightarrow f(x) \in Prog(S)$

b) $u \in S$ und $\langle M_u \rangle \in Prog(S)$. Hier wird gezeigt: $H_\varepsilon \leq Prog(S)$. Rest analog. $\square$

$L = \{ \langle M \rangle \mid$ es gibt ~~unendl.~~ genau eine Eingabe w, so dass M gestartet mit w hält$\}$ ist nicht r.a.

Reduktion: $\overline{H} \leq L$

⟨ Drumherum $\langle M \rangle w$
lies y
~~starte M mit w~~
if $y = 010$ then stop
starte M mit w
stop ⟩

1.8 Rekursive Aufzählbarkeit

Satz 1.23 Sei L eine unendliche Sprache. Dann gilt
L ist r.a. $\Leftrightarrow$ es gibt eine total berechenbare surjektive Fkt $g: \{0,1\}^* \to L$

~~es gibt M~~

Bew: „$\Leftarrow$" Sei $g: \{0,1\}^* \to L$ eine totalberechenbare surjektive Fkt, die von der det. 1-Band-TM M_g berechnet wird. Ziel: Programmiere TM M, die gestartet mit x genau dann hält, wenn $x \in L$ ist, und dabei darf M die TM M_g benutzen.
Wir wissen: $x \in L$, dann gibt es ein $y \in \{0,1\}^*$ mit $g(y) = x$. M bekommt als Eingabe das x.
Wir machen uns auf die Suche nach y

```
TM M
  die Eingabe sei x
  y := ε
  while M_g gest. mit y nicht x ausgibt do  y := lexikographisch.nächstes(y)
```

M hält mit x gdw $x \in L$

Wir haben M mit $x \in L \Leftrightarrow M$ gest. mit x hält. Ziel: Wir müssen M_g programmieren mit genau alle $x \in L$ können von M_g ausgegeben werden, wobei M_g auch für jede Eingabe y halten muss

„$\Rightarrow$" L r.a. $\to$ es gibt det. 1-Band-TM M, die für genau die $x \in L$ gestartet hält (d.h. $x \in L$, dann führt M gestartet mit x endlich viele Schritte (t Schritte) aus und hält; $x \notin L$, dann führt M gest. mit x unendlich viele Schritte aus)

Ziel: Programmiere eine TM M_g mit
(i) M_g hält für jede Eingabe y
(ii) M_g gibt ausschließlich Wörter aus L zurück
(iii) M_g muss jedes Wort aus L ausgeben können

Die Eingabe für M wird sein: $y =$ „(x,t)", und M_g führt t Schritte von M gest. mit x aus.
Wenn das kontrollierte Laufenlassen von M gest. mit x nicht innerhalb von t Schritten hält, gib festes $x_{fix} \in L$ aus, sonst gib x aus.

$$\text{aus\_eins\_mach\_zwei}(y) = \begin{cases} (a_1 \dots a_n, t) & \text{falls } y = a_1 \dots a_n 0 \underbrace{1 \dots 1}_{t \text{ mal}} \\ (0,1) & \text{sonst} \end{cases}$$

```
TM M_g
  Eingabe sei y
  (x,t) := auseinsmachZwei(y)
  simuliere auf ExtraBand M gest. mit x für t Schritte
  falls M einen Endzust. erreicht hat, gib x aus,
      sonst gib x_fix (∈ L) aus.
```

aemz$(011) = (\varepsilon, t)$
aems$(1111) = (0,1)$
aems$(0110) = (011, 0)$
aems$(011\,0100111) = (011010, 3)$

ist total berechenbar

• Eine Menge $\mathcal{L}$ von Sprachen bezeichnet man als Sprachklasse (oder auch Sprachfamilie)
z.B. $\mathcal{E} := \{ L \mid L$ ist entscheidbar$\}$
$\mathcal{L}_0 = \{ L \mid L$ ist r.a.$\}$

Wenn wir eine k-stellige Operation $op(\cdot,\cdot,\cdot)$ auf Sprachen haben, d.h. eine Operation, die aus k Sprachen eine weitere Sprache macht, dann ist die Sprachklasse $\mathcal{L}$ genau dann unter op abgeschlossen, wenn $L_1, \dots L_k \in \mathcal{L} \to op(L_1, \dots L_k) \in \mathcal{L}$ für alle $L_1, \dots, L_k \in \mathcal{L}$

z.B. $L_1, L_2 \in \mathcal{E} \Rightarrow L_1 \cup L_2 \in \mathcal{E}$

BFS | Wir wissen ja, dass $\Pi \in L_0$, aber Π_0 ist nicht in L_0, daher ist L_0 nicht unter Komplement-bildung abgeschlossen

Satz 1.24: Seien L_1 und L_2 r.a. Dann gilt:
(i) $L_1 \cup L_2$ ist r.a.
(ii) $L_1 \cap L_2$ ist r.a.

Bew (i) Für x führe abwechselnd auf 2 Bändern M_1 gest. mit x (für L_1) und M_2 gestartet mit x aus. Wenn $x \in L_1 \cup L_2$, wird dieser Algo auch stoppen
(ii) Analog. Nun muss für beide TM gestoppt werden. □.

Satz. 1.25 L ist entscheidbar => L und $\bar{L}$ sind r.a.

Kapitel 2: Nichtdeterministische TM und das P-NP-Problem

Def. 2.1 Eine nichtdet. 1-Band-TM wird beschrieben durch 6 Komponenten $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, deren Bedeutungen bis auf δ gleich denen der det. 1-Band-TM sind (analog. k-Band-NTM).

Nun $\delta: Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{R, N, L\})$
Potenzmenge

Startkonf: $q_0 x$, $x \in \Sigma^*$
Sei z.B. $(q', b, R) \in \delta(q, a)$ Dann ist $\alpha q a \beta \vdash \alpha a q' \beta$ ein möglicher Übergang
$(q'', c, N) \in$ Dann ist $\alpha q a \beta \vdash \alpha q'' c \beta$ ein weiterer möglicher Übergang
Alle Möglichkeiten sind gleichberechtigt

Eine NTM M akzeptiert x gdw. es eine Rechnung $q_0 x \vdash \alpha q \beta$, $q \in F$ gibt
$L(M) = \{x \mid M \text{ akz. } x\}$

Bsp: NTM, 2 Bänder $\Gamma = \{0, 1, B\}$, $\Sigma = \{0, 1\}$, $F = \{q_2\}$, $Q = \{q_0, q_1, q_2\}$
$a \in \{0, 1\}$ $\delta(q_0, (a, B)) = \{(q_0, (a, a), (R, R)), (q_1, (a, a), (R, N))\}$
$\delta(q_1, (a, a)) = \{(q_1, (a, a), (R, L))\}$
$\delta(q_1, (B, B)) = \{(q_2, (B, B), (N, N))\}$
Diese NTM akzeptiert $L = \{ww^R \mid w \in \{0, 1\}^*\}$

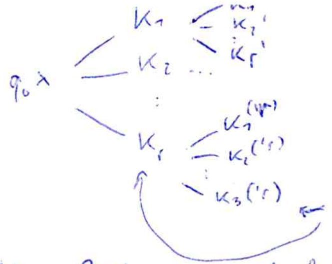
Bsp: $\Gamma = \{0, 1, B\}$, $\Sigma = \{0, 1\}$, $F = \{q_1\}$
$\delta(q_0, B) = \{(q_0, 0, R), (q_0, 1, R), (q_1, B, N)\}$
Diese NTM erlaubt eine bel. 0-1-Folge aufs Band. Wir sagen, sie rät eine 0-1-Folge

Rucksackproblem K (Knapsack Problem)
$K = \{\langle a_1, .. a_n, b \rangle \mid a_i \in \mathbb{N}, b \in \mathbb{N}$, es gibt $\alpha_1, .. \alpha_n \in \{0, 1\}$, $\sum_{i=1}^{n} \alpha_i \cdot a_i = b\}$
- rate eine 0-1-Folge $\alpha_1, \dots, \alpha_n$ Ja-Antwort: ist korrekt
- verifiziere $\sum_{i=1}^{n} \alpha_i \cdot a_i = b$ Nein-Antwort: heißt gar nix

Def 2.2 Sei M eine NTM.
Laufzeit $T_M(x)$ = Länge einer kürzesten akzeptierenden Rechnung von M gest. mit x
Platz $S_M(x)$ = geringster Platzbedarf einer ~~kürzesten~~ akz. Rechnung
$T_M(n)$ - zeitbeschränkt
$S_M(n)$ - platzbeschränkt } analog wie bei den det. TM

Def. 2.3 Jede NTM M kann durch eine det. TM M' simuliert werden
Falls M $t(n)$-zeitbeschränkt ist (und damit auch $t(n)$-platzbeschränkt), so ist M'
$2^{O(t(n))}$-zeitbeschränkt und $O(t(n)^2)$-platzbeschränkt

Bew: Sei M gegeben. Sei $r = \max\{|\delta(q, a)| \mid q \in Q, a \in \Gamma\}$ die max. Anzahl an erlaubten, direkten Nachfolgekonfigurationen einer Rechnung von M.
Startkonf. $q_0 x$
Interpretation von M gest. mit x als r-närer Baum
Eingabe für M' sei x



← Mögl. Nachfolgekonf. von q_0 bzw. deren Nachfolgekonf.

1. Idee: Breitensuche auf dem Baum. Zeitbedarf: $\underbrace{\Theta\left(r^{t(|x|)}\right)}_{\text{Anzahl Knoten bis zur Akzeptanz}} \cdot t(|x|) = 2^{O(t(|x|))}$

Platzbedarf: $2^{O(t(|x|))} \cdot t(|x|)$

2. Idee: Tiefensuche, kontrollierte DFS: Platz: „Tiefe" · „Platz für Konf" $= O(\text{„Tiefe"} \cdot t(|x|)) = O(t(|x|)^2)$

$T := 2$, <u>while</u> noch keine akz. Rechng gefunden (und der Baum noch nicht ganz abgearbeitet ist)
<u>do</u> - Führe Tiefensuche bis zur Tiefe T durch (jedes Mal wieder bei q_0x anfangen)
- $T := T+1$
<u>done</u>

Laufzeit: $O\left(\sum_{T=2}^{t(n)} 2^{O(T)}\right) = O\left(2^{O(t(|x|))}\right) = 2^{O(t(|x|))}$

<u>Corollar 2.4</u> NTM akz. genau die r.a. Sprachen

<u>Def 2.5</u>
- DTIME $(t(n)) = \{L \mid$ es gibt $O(t(n))$-zeitbeschränkte <u>det</u> TM, die L entscheidet$\}$
- NTIME $(t(n)) = \{L \mid$ es gibt $O(t(n))$-zeitbeschränkte <u>N</u>TM, die L akzeptiert$\}$
- $P = \bigcup_{k \in \mathbb{N}} \text{DTIME}(n^k)$
- $NP = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$

⟶ Offensichtlich: $P \subseteq NP$ (jede det. TM ist auch NTM mit $r=1$)

Aus Satz 2.3 folgt: alle Sprachen in NP sind entscheidbar

Vorteile von P:
- robust (hängt nicht vom Maschinenmodell ab)
- Polynome sind unter Hintereinanderausführung abgeschlossen: $p_1(p_2(n))$ ist Polynom, wenn p_1 und p_2 Polynome sind
- enthält die Probleme, die sich in der Praxis als handhabbar erwiesen haben, d.h. einigermaßen schnell gelöst werden können
- ermöglicht eine reichhaltige Theorie, da man wg. der Robustheit in P bleibt

CLIQUE $= \{\langle G,k\rangle \mid G$ ist ein ~~vollständig~~ ungerichteter Graph, der einen vollst. Graph mit k Knoten enthält$\}$

Hamilton-Kreis-Problem: $HC = \{\langle G\rangle \mid G$ ist ungeg. Graph, der einen Hamilton-Kreis enthält$\}$

Knotenüberdeckungsproblem: Geg. Graph $G=(V,E)$. A ist eine Kantenüberdeckung, falls jede Kante aus E mind. einen Knoten aus A besitzt.

$VC = \{\langle G,k\rangle \mid G$ hat Knotenüberdeckung der Größe $k\}$

Traveling Salesperson Problem: $TSP = \{\langle G,c,k\rangle \mid$ Graph $(c: E \to \mathbb{N})$ enthält einen HC mit Gew. $\leq k\}$

<u>Def 2.6</u> Sei L eine Sprache über $\{0,1\}$. Eine det. TM M_L heißt $t(n)$-beschränkter Verifizierer für L, wenn gilt
(i) Die Eingaben von V_L sind von der Form $x\#w$, $x,w \in \{0,1\}^*$
(ii) Die Laufzeit ist $O(t(|x|))$ (kein w!)
(iii) $\forall x \in \{0,1\}^*: x \in L \iff \exists w: |w| \leq t(|x|)$ und V_L akz. $x\#w$

<u>Satz 2.7</u> Sei L eine Sprache
$L \in \text{NTIME}(t(n)) \iff$ es gibt einen $t(n)$-beschränkten Verifizierer V_L für L

<u>Bew</u>: „$\Leftarrow$" TM M: (1) Eingabe sei x
(2) Rate w (in $|w|$ Schritten)
(3) Falls V_L gest. mit $x\#w$ akz. hält, dann akz. x

Offensichtlich akz M nur Eingaben $x \in L$. Aus Def 2.6 (iii) folgt, dass es ein w mit $|w| \leq t(|x|)$ gibt. V akz. in Zeit $O(t(|x|))$. Erraten von w dauert nur $\leq t(|x|)$ Schritte.
M ist also $O(t(n))$-zeitbeschränkt.

FS

"=>" $L \in NTIME(t(n))$. M sei eine $t(n)$-zeitbeschränkte nicht-det. TM für L.
OBdA: jede Konf einer Rechnung von M hat keine oder genau 2 Nachfolgekonf.
0 = links absteigen , 1 = rechts absteigen
In einer nichtdet. akz. Rechnung von M gest. mit x sammle das "links" bzw. "rechts"
mit jew. 0 bzw 1 als w
Der Verifizierer für L führt bei Eingabe $x\#w$ die Rechnung von M gest. mit x (nun det.) aus. (i), (ii), (iii) sind erfüllt

Korollar 2.8 NP = {L | es gibt einen polynomiellen Verifizierer}

2.1 NP-Vollständigkeit

Def 2.9 $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$.
L_1 ist polynomiell reduzierbar auf L_2 $(L_1 \leq_p L_2)$ <=> (i) $L_1 \leq L_2$ (mittels f)
(ii) Die Laufzeit zur Berechnung von $f(x)$ ist $O(|x|^k)$ für ein festes $k \in \mathbb{N}$

Lemma 2.10 "$\leq_p$" ist transitiv

Def. 2.11 L heißt NP-schwer (NP-hard), wenn gilt: $\forall L' \in NP$ $L' \leq_p L$
L ist NP-vollständig (NP-complete) wenn gilt: (i) $L \in NP$ (ii) L ist NP-schwer

Lemma 2.12 (i) L ist NP-schwer. Dann gilt: (a) $L \in P \Rightarrow P = NP$
(b) $P \neq NP \Rightarrow L \notin P$
(ii) L ist NP-vollständig. Dann gilt: (a) $L \in P \Leftrightarrow P = NP$,
(b) $L' \in NP$ und $L \leq_p L' \Rightarrow L'$ ist NP-vollständig

2.2 Der Satz von Cook

Def. 2.13
- $V = \{x_1, x_2, \dots, x_n\}$ Menge Boolescher Variablen
- Ein Literal ist eine Variable oder deren Negation (d.h. zu $x \in V$ ist x ein Literal und $\bar{x}$ ist ein Literal!)
  Eine Klausel K der Länge s ist ein Boolescher Ausdruck aus s Literalen:
  $K = y_1 \vee y_2 \vee \dots \vee y_s$ mit $y_i \in \{x_1, \dots, x_n\} \cup \{\bar{x}_1, \dots \bar{x}_n\}$
- Ein Ausdruck in konjunktiver Normalform (KNF) ist ein Boolescher Ausdruck Φ der Form
  $\Phi = K_1 \wedge K_2 \wedge \dots \wedge K_t$ für Klauseln K_i. Die Anzahl der in Φ vorkommenden $\wedge$ und $\vee$-Zeichen ist die Größe size(Φ) von Φ
- Eine totale Abbildung c: $V \to \{TRUE, FALSE\}$, die jeder Variable einen Wahrheitswert zuweist, heißt Belegung der Variablen. c wird kanonisch auf Literale, Klauseln K und KNFs Φ fortgesetzt, $c(K)$ bzw. $c(\Phi)$ ist das Ergebnis der Auswertung von K bzw Φ unter Anwendung von c.
- Eine KNF Φ heißt erfüllbar, wenn es eine Belegung c gibt, sodass $c(\Phi) = TRUE$ ist
  Φ wird kodiert durch $\langle \Phi \rangle$:
  $\langle x_i \rangle = 1\#bin(i)$ $\langle K \rangle = \langle y_1 \vee y_2 \vee \dots \vee y_s \rangle = \langle y_1 \rangle \#\# \langle y_2 \rangle \#\# \dots \langle y_s \rangle$
  $\langle \bar{x}_i \rangle = 0\#bin(i)$ $\langle \Phi \rangle = \langle K_1 \wedge \dots \wedge K_t \rangle = \langle K_1 \rangle \#\#\# \dots \#\#\# \langle K_t \rangle$
  $|\langle \Phi \rangle| = O(size(\Phi) \cdot \log |V|)$

Das Erfüllbarkeitsproblem (engl. Satisfiability Problem) SAT ist
$SAT = \{\langle \Phi \rangle \mid \Phi$ ist eine erfüllbare KNF$\}$

Satz 2.14 (Satz von Cook, 1971): SAT ist NP-vollständig

Bew: (i) $SAT \in NP$: Rate Belegung und verifiziere, dass sie die KNF erfüllt
Offensichtlich kann dies in Polynomzeit in $|\langle \Phi \rangle|$ ausgeführt werden
(ii) z.z. SAT ist NP-schwer, d.h. $\forall L \in NP$: $L \leq_p SAT$
Sei $L \in NP$ beliebig, aber fest. Sei M_L eine nicht det. 1-Band TM, $M_L = (Q, \Sigma, \Gamma, \delta, q_0, F)$, der Laufzeit $c \cdot n^k$, die L akzeptiert.
Ziel: Für $w \in L$ muss eine KNF Φ_w konstruiert werden (mit vielen, aber nicht zu vielen Variablen) so dass $w \in L \Leftrightarrow \Phi_w \in SAT$ ($\Leftrightarrow \Phi_w$ ist erfüllbar)
soll Rechnung von M_L gest. mit w simulieren

Φ_w soll mögl. Rechnungen von M_L gest. mit w beschreiben; $\langle \Phi\ w \rangle$ kann in Zeit polynomiell in $|w|$ berechnet werden.

OBdA: $F = \{q_F\}$, $\delta(q_F, a) = \{(q_F, a, N)\}$

$w \in L \Leftrightarrow$ es gibt Berechnung von M_L gest. mit w der Länge $T = c \cdot n^k$ ($|w| = n$), die w akzeptiert

$\Leftrightarrow q_0 w = K_0 \vdash K_1 \vdash \ldots \vdash K_T$ und der Zustand in K_T ist q_F (es ex. sowas)

$V_{M_L} = \{zelle_{t,i,a} \mid 0 \le t \le T, -T \le i \le T, a \in \Gamma\}$
$\cup \{kopf_{t,i} \mid 0 \le t \le T, -T \le i \le T\}$
$\cup \{zustand_{t,q} \mid 0 \le t \le T, q \in Q\}$

$zelle_{37,5,\#} = TRUE \Leftrightarrow$ in Konf. 37 steht an Speicherzelle 5 ein „#"

$|V_{M_L}| = (T+1)\cdot(2T+1)\cdot|\Gamma| + (T+1)(2T+1) + (T+1)\cdot|Q|$

$zelle_{t,i,a} = TRUE \Leftrightarrow$ In der Rechnung von M_L gest. mit w steht in der Konfig. K_t in der Zelle i das Zeichen a

$kopf_{t,i} = TRUE \Leftrightarrow$ In der Rechnung von M_L gest. mit w ist in Konfig K_t der Kopf unter Zelle i

$zustand_{t,q} = TRUE \Leftrightarrow$ In der Rechnung von M_L gest. mit w ist in Konfig. K_t der Zustand q

Genau eine Var. ist TRUE $(x_1, \ldots, x_T) = (x_1 \vee \ldots \vee x_T) \wedge \bigwedge_{i \ne j} (\bar{x}_i \vee \bar{x}_j)$ Größe $O(f^2)$

ist gleich $(x,y) = (\bar{x} \vee y) \wedge (x \vee \bar{y})$

Aus V_{M_L} nehmen wir uns nun V_t heraus: $V_t = \{zelle_{t,i,a}, kopf_{t,i}, zustand_{t,q} \mid -T \le i \le T, q \in Q\}$

$Konf(V_t) = TRUE \Leftrightarrow$ die Belegung der Konf. in V_t beschreibt genau eine Konfig.

$Konf(V_t) =$ GenauEineVarIstTRUE $(zustand_{t,q_0}, \ldots, zustand_{t,q_{|Q|-1}})$
$\wedge$ GenauEineVarIstTRUE $(kopf_{t,-T}, \ldots, kopf_{t,T})$
$\wedge \bigwedge_{-T \le i \le T}$ GenauEineVarIstTRUE $(zelle_{t,i,a_1}, zelle_{t,i,a_2}, \ldots, zelle_{t,i,a_{|\Gamma|}})$ Größe $O(f^2)$

$\Phi_w =$ Rechnung $(V_0, V_1, V_2, \ldots, V_T) =$ Startkonf$(V_0) \wedge \bigwedge_{t=0}^{T-1}$ EinSchritt$(V_t, V_{t+1}) \wedge zustand_{T,q_F}$

Startkonf$(V_0) = \bigwedge_{i=0}^{n-1} zelle_{0,i,w_{i+1}} \wedge \bigwedge_{\substack{-T \le i < 0 \\ n \le i \le T}} zelle_{0,i,B} \wedge zustand_{0,q_0} \wedge kopf_{0,0}$

Ein Schritt $(V_t, V_{t+1}) = \bigvee_{\gamma \in \delta\text{-Tabelle}}$ Schritt-durch-γ (V_t, V_{t+1})

$\underbrace{\boxed{(q', a', D) \in \delta(q, a)}}_{\gamma}$ ein mögl. δ-Übergang

Schritt_durch_γ $(V_t, V_{t+1}) =$ Konf$(V_t) \wedge$ Konf$(V_{t+1}) \wedge zustand_{t,q} \wedge zustand_{t+1,q'} \wedge$
{ „unter Kopf in K_t steht a" $\wedge$ „an derselben Pos. in K_{t+1} steht a'" $\wedge$ {„Kopf wird in Richtung D bewegt"
{$\wedge$ „Rest vom Band ist unverändert" }

$\bigwedge_{i=-T}^{T} \overline{kopf_{t,i}} \vee (zelle_{t,i,a} \wedge zelle_{t+1,i,a'})$

$D = L$: $\bigwedge_{i=-T+1}^{T-1}$ SindGleich $(kopf_{t,i}, kopf_{t+1,i-1})$

Analog: R, N

Größe von Φ_w ist $O(T^3) = O(n^{3k})$ und $w \in L \Leftrightarrow$ es gibt akz. Rechnung von M gest. mit w
$\Leftrightarrow \langle \Phi\ w \rangle \in SAT$ $\square$

Die Technik des Beweises von Satz 2.14 nennt man Master-Reduktion. Wir haben bisher nur ein NP-vollst. Problem, nämlich SAT. Unter Anwendung von Lemma 2.12 c) bekommen wir mehr davon

$\rightarrow$ L NP-vollst. und $L' \in NP$ und $L \le_p L' \Rightarrow L'$ ist NP-vollst.

3SAT $= \{\langle \Phi \rangle \mid \Phi$ ist Boolesche Formel in KNF, in der jede Klausel aus genau 3 verschiedenen, nicht-trivialen Literalen besteht$\}$

Satz 2.15: 3SAT ist NP-vollständig

Bew.: (i) $3SAT \in NP$ ✓

(ii) $SAT \le_p 3SAT$

FS „$\langle \Phi \rangle \in SAT$" $\Leftarrow\Rightarrow$, $f(\langle \Phi \rangle) \in 3SAT$

$\Phi = K_1 \wedge K_2 \wedge \dots \wedge K_T$

- $K_i = l$ $\quad f(K_i) = (l \vee y_{i_1} \vee y_{i_2}) \wedge (l \vee y_{i_1} \vee \overline{y_{i_2}}) \wedge (l \vee \overline{y_{i_1}} \vee y_{i_2}) \wedge (l \vee \overline{y_{i_1}} \vee \overline{y_{i_2}})$ $\quad size(K_i) = 0$, $size(f(K_i)) = 11$
- $K_i = l_1 \vee l_2$ $\quad f(K_i) = (l_1 \vee l_2 \vee y_i) \wedge (l_1 \vee l_2 \vee \overline{y_i})$ $\quad size(K_i) = 1$, $size(f(K_i)) = 5$
- $K_i = l_1 \vee l_2 \vee \dots \vee l_k$ für $k \geq 3$ $\quad size(K_i) = k-1$, $size(f(K_i)) = 3k-7$

$f(K_i) = (l_1 \vee l_2 \vee y_{i_1}) \wedge (\overline{y_{i_1}} \vee l_3 \vee y_{i_2}) \wedge (\overline{y_{i_2}} \vee l_4 \vee y_{i_3}) \wedge (\overline{y_{i_3}} \dots) \wedge \dots \wedge (\overline{y_{i_{k-3}}} \vee l_{k-1} \vee l_k)$

$f(\Phi) = f(K_1) \wedge f(K_2) \wedge \dots \wedge f(K_T)$

Nun gilt: Φ erfüllbar $\Leftarrow\Rightarrow$ $f(\Phi)$ ist erfüllbar (etwas Nachdenken bei $k > 3$)
f kann in Polyzeit berechnet werden

Satz 2.10 CLIQUE ist NP-vollständig

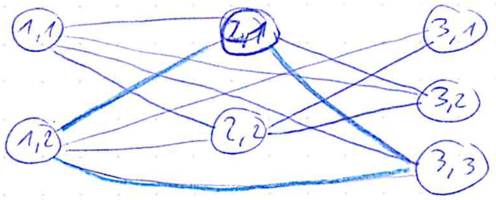
NP ist meistens entscheidbar!

Bew: (i) CLIQUE $\in NP$ ✓
(ii) CLIQUE ist NP-schwer $\qquad$ z: $SAT \leq_p CLIQUE$

$\Phi = \bigwedge_{i=1}^{T} K_i$ über den Variablen $\{x_1, \dots, x_n\}$

⇝ einen Graphen in Abh. von Φ angeben und eine Zahl k, so dass dann dieser eine Clique der Größe k enthält, und nur dann, wenn Φ erfüllt ist.

zB: $\Phi = (x_1 \vee \overline{x_2}) \wedge (x_1 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

$f(\Phi) = ($  $, 3)$

$x_1 =$ TRUE
$x_2 =$ FALSE
$x_3 =$ TRUE
usw.

$K_i = y_1^{(i)} \vee \dots \vee y_{s_i}^{(i)}$, $y_l^{(i)}$ ist Literal

Ziel: konstruiere $\langle G_\Phi, T \rangle$ mit $\langle \Phi \rangle \in SAT \Leftarrow\Rightarrow f(\langle \Phi \rangle) = \langle G_\Phi, T \rangle \in CLIQUE$

$V = \{ (i,j) \mid i \in \{1, \dots, T\} \; j \in \{1, \dots, s_i\} \}$

$E = \{ \{ (i,j), (i',j') \} \mid i \neq i'$ und $y_j^{(i)} \neq \overline{y_{j'}}^{(i')} \}$

Φ erfüllbar $\Leftarrow\Rightarrow$ G_Φ enthält Clique der Größe T

„$\Rightarrow$" $(c_1, \dots, c_n)$ sei eine erfüllende Belegung der Variablen in Φ
Boolesche Werte
$(x_1, \dots, x_n)$ ← Variablen-Namen

Durch die Belegung muss in jeder Klausel mind. ein Literal erfüllt sein. Wähle in G_Φ einen Knoten so, dass das zugehörige Literal in Φ erfüllt ist. Wir haben so in jeder „Spalte" des Graphen einen Knoten gefunden. Da die zugehörigen Literale sich nicht widersprechen, sind diese Variablen untereinander verbunden, also bilden sie eine Clique der Gr. T in G_Φ

„$\Leftarrow$" Sei $C = \{ (i_1, j_1), (i_2, j_2), \dots, (i_T, j_T) \}$ eine bel. Clique der Größe T in G_Φ. Setze in K_{i_1} die Variable, die im Literal mit der Nummer j_1 vorkommt, so dass dieses Literal erfüllt wird. Damit wird K_{i_1} erfüllt. Wiederhole dies für alle (i_k, j_k). Da die Knoten eine Clique bilden, „widersprechen" sich die ausgewählten Literale nie.
Damit ist Φ erfüllt. $\square$

Satz 2.17: VERTEXCOVER ist NP-vollständig

Bew: (i) VERTEX COVER $\in$ NP

(ii) VERTEX COVER ist NP-schwer

wir zeigen: CLIQUE $\leq_p$ VERTEXCOVER

$K(G)$ sei zu G der Komplementärgraph.

„Lösche in G die Kanten und füg die fehlenden hinzu"

$f(\langle G,k\rangle) = \langle K(G), |V|-k\rangle$

„Hat G Clique der Größe k?" „Hat K(G) eine Knotenüberdeckung der Größe $\leq M-k$?"

Beh: $\langle G,k\rangle \in$ CLIQUE $\Leftrightarrow$ $\langle K(G), |V|-k\rangle \in$ VERTEXCOVER

In $K(G)$ gibt es ausschließlich Kanten außerhalb von U und zw. U und $V\setminus U$, aber keine innerhalb von U.

Setze $C = V\setminus U$, $|C| = |V|-k$, C ist wg. Bemerkung eine Knotenüberdeckung

Binary Programming BP

$BP = \{\langle A,b\rangle \mid A$ ist $n\times m$-Matrix mit Einträgen aus $\mathbb{Z}$, b ist n-Vektor mit Einträgen aus $\mathbb{Z}$.

$\exists\, y \in \{0,1\}^n: A\cdot y \leq b$

Satz 2.18: BP ist NP-vollständig

Bew: (i) BP $\in$ NP ✓

(ii) SAT $\leq_p$ BP

$\Phi = K_1 \wedge \dots \wedge K_T \quad K_i = y_1^{(i)} \vee \dots \vee y_{s_i}^{(i)}$ über $V = \{x_1, \dots x_n\}$

Ungleichungssystem über den 0-1-Var: $z_1, z_1', \dots, z_n, z_n'$

- für jedes x_i: $z_i + z_i' = 1 \qquad -z_i + z_i' \leq -1$

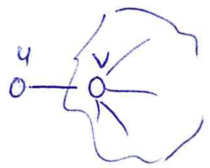
$\qquad\qquad\qquad\qquad\qquad\quad z_i + z_i' \leq 1$

- für jedes K_i: $\sum y_j^{(i)} \geq 1, \quad y_j^{(i)} = \begin{cases} z_l & y_j^{(i)} = x_l \\ z_l' & y_j^{(i)} = \bar{x}_l \end{cases}$

Vermutung: $P \neq NP$

2.3 Ein exakter mild exponentieller Algo. für VERTEX COVER

$\rightarrow$ u nicht in Cover $\Leftrightarrow$ alle Nachbarn von u müssen in C sein



Ist u in Cover C, so ist $((C\setminus\{u\})\cup\{v\})$. Ist v vorher nicht in C, so ist das neue Cover gleich groß. Knoten von Grad 1 können durch „Zwangszuweisung" des Nachbar in die Überdeckung eliminiert werden.

Haben alle Knoten Grad 2, dann ist der Graph eine Sammlung von einem Kreis bzw. mehreren Kreisen. Opt. Lösung: nimm jeden zweiten Knoten in das Cover

Algo VC_{OPT} $(G=(V,E)$, Kandidat$)$ Globale Var.: VC_{OPT} zu Beginn $VC_{OPT} = V$

wenn $|E| = 0$

dann wenn $|VC_{OPT}| > |\text{Kandidat}|$ dann $VC_{OPT} :=$ Kandidat end

sonst solange es Kanten $\{u,v\} \in E$ gibt mit Grad von u ist 1

Kandidat := Kandidat $\cup \{v\}$

lösche u und die Kante $\{u,v\}$

end

wenn alle Knoten Grad 0 oder 2 haben

dann berechne auf diesem Graphen eine optimale Überdeckung (in Polynomzeit), packe diese zu Kandidat

$E := \emptyset$, $VC_{OPT}(G$, Kandidat$)$

Sonst ...

6 ||

ES

Sonst // Knoten mit Grad ≥ 3

wähle v mit Grad von v ist ≥ 3

Nachbarn von v seien $v_1, \dots, v_k$, $k \geq 3$

$VC_{OPT}(G \setminus v,$ Kandidat $U \cup \{v\})$

$VC_{OPT}(G \setminus \{v_1, \dots, v_k\},$ Kandidat $U \cup \{v_1, \dots v_k\}$

VC_{OPT} enthält maximale Knotenüberdeckung

Start: $VC_{OPT}(G, \emptyset)$ correct by construction

Sei $t(n)$ die Laufzeit bei Eingabe mit n Knoten

$\underbrace{t(n) \leq t(n-1)}_{(*)} + \underbrace{t(n-4)}_{(**)} + 1 \quad \leadsto t(n) = A\lambda^n \to A\lambda^n = A\lambda^{n-1} + A\lambda^{n-4}$

$\lambda^4 = \lambda^3 + 1$

Aufbauend auf unsere Inhalte der Kap. 2:

- Komplexitätstheorie
  - Platzhierarchie
  - Zeithierarchie
  - Nichtdeterminismus
    - PSPACE - NPSPACE
    - Co-Klassen
- Approximationsalgorithmen.

- Spiele ($\dots \forall \exists \forall \exists \dots$). PSPACE - Vollständigkeit
- Co-NSPACE $(s(n))$ = NSPACE $(s(n))$

3. Formale Sprachen

Insgesamt in dieser VL: unendl. Objekte mit endl. Mitteln beschreiben

Bisher: Erkennen mit Maschinen, „Syntaxanalyse"

Jetzt: Erzeugen: Regeln, die mächtig genug sind, um aufwendige Konstrukte zu beschreiben

Regeln, die einfach genug sind, um eine effiziente Analyse zu erlauben

3.1 Grammatiken

Def 3.1 Eine Grammatik G vom Typ Chomsky-0 ist beschrieben durch 4 Komponenten (V, Σ, P, S) mit

- V endl. Menge von Variablen (traditionell beschrieben durch große Buchstaben $A, B, \dots$)
- Σ endl. Alphabet von Terminalen (" kleine Buchstaben $a, b, \dots, 0, 1, \dots$)
- S Startsymbol, $S \in V$
- $P \subseteq (V \cup \Sigma)^+ \setminus \Sigma^* \times (V \cup \Sigma)^*$ ist endl. Menge an Produktionen oder Ersetzungen

  Statt $(u, v) \in P$ schreiben wir auch $u \to v$

  Wir sagen:
  - $w' \in (V \cup \Sigma)^*$ ist aus $w \in (V \cup \Sigma)^+$ direkt ableitbar $(w \to w')$, falls es $(u \to v) \in P$ und $\alpha, \beta \in (V \cup \Sigma)^*$ gibt mit $w = \alpha u \beta$ und $w' = \alpha v \beta$
  - w' ist aus w indirekt ableitbar $(w \xrightarrow{*} w')$, falls w' durch endl. viele direkte Ableitungsschritte aus w erzeugbar ist

Die von G erzeugte Sprache $L(G)$ ist: $L(G) = \{w \mid S \xrightarrow{*} w, w \in \Sigma^*$

- G heißt kontextsensitiv oder auch vom Typ Chomsky-1, falls für jede Produktion ~~gilt~~ $(u \to v) \in P$ gilt: $|u| \leq |v|$
- G heißt kontextfrei oder auch vom Typ Chomsky-2, falls für alle Regeln $(u \to v) \in P$ gilt: $u \in V$ (und $v \in (V \cup \Sigma)^*$)
- G heißt regulär oder auch vom Typ Chomsky-3, falls alle Regeln von der Art $u \to v$ sind mit $u \in V$ und ($v = \epsilon$ oder $v = a$, $a \in \Sigma$ oder $v = aW$ mit $a \in \Sigma$, $W \in V$), d.h. $A \to \epsilon$, $A \to a$ o. $A \to bB$

Erweiterung zu kontextsensitiv: Man erlaubt $S \to \epsilon$, aber verbietet S auf rechten Seiten

Satz 3.2 $L(G) = \{a^n b^n c^n \mid n \geq 1\}$ (Bsp!)

Bew: „$\supseteq$" Sei $a^n b^n c^n$ geg.

1. Wende $(n-1)$-mal (1) an und einmal Regel (2): $S \xrightarrow{*} a^n (BC)^n$
2. $1/2\, n(n-1)$-mal (3): $S \xrightarrow{*} a^n B^n C^n$
3. 1x (4), $(n-1)$x (5): $S \xrightarrow{*} a^n b^n C^n$

„⊆" In jeder Regel ist die Anzahl der a's gleich der Anzahl der (b's und B's)
In jeder Regel ist die Anzahl der (b's und B's) gleich der Anzahl der (c's und C's)
⇝ In jeder Satzform ist die Anzahl der a's gleich der Anzahl der (b's und B's) gleich der Anz. d. (c's und C's)

a's können nur aus (1) und (2) kommen, und daher stehen sie immer nur ganz vorne. D.h. in einem bel. $w \in L(G)$ stehen a's immer ganz vorne. b's werden durch (4) und (5) erzeugt. b's folgen immer auf a's. Gleiches gilt für die c's. Insgesamt: a's vor b's vor c's, wenn Satzform keine Variable enthält. Zsm. mit den Zählen folgt: $w = a^n b^n c^n$ für ein $n \geq 1$ □.

Satz 3.3 Eine Sprache L ist genau dann rekursiv aufzählbar, wenn es eine Grammatik G vom Typ Chomsky-0 gibt mit $L = L(G)$. ($\mathcal{L}_0$ ist die Klasse der r.a. Sprachen)

Bew: „⇐" G sei gegeben. Ziel: det 1-Band-TM M mit $L(M) = L(G)$
M muss also genau $w \in L(G)$ akzeptieren

TM: Eingabe sei w
Rate die Ableitung mit den Regeln von G und vergl. das Ergebnis mit w
Falls gleich: stop, sonst Endlosschleife

„⇒" TM M geg. Ziel: Typ-0-Grammatik G mit $L(G) = \{w \mid M \text{ gestartet mit } w \text{ hält}\}$

- $S \to \begin{bmatrix}\varepsilon\\B\end{bmatrix} S \mid q_0 A_1$
- $A_1 \to \begin{bmatrix}a\\a\end{bmatrix} A_1 \mid A_2$ | für alle $a \in \Sigma$
- $A_2 \to \begin{bmatrix}\varepsilon\\B\end{bmatrix} A_2 \mid \varepsilon$
- $q\begin{bmatrix}a\\X\end{bmatrix} \to \begin{bmatrix}a\\Y\end{bmatrix} q'$ für alle $a \in \Sigma \cup \{„\varepsilon"\}$, $q \in Q$ und $X, Y \in \Gamma$ mit $\delta(q,X) = (q', Y, R)$
- $\begin{bmatrix}b\\Z\end{bmatrix} q \begin{bmatrix}a\\X\end{bmatrix} \to q' \begin{bmatrix}b\\Z\end{bmatrix}\begin{bmatrix}a\\Y\end{bmatrix}$ für alle $a, b \in \Sigma \cup \{„\varepsilon"\}$ und alle $q \in Q$, $X, Y, Z \in \Gamma$ mit $\delta(q,X) = (q', Y, L)$
- $q\begin{bmatrix}a\\X\end{bmatrix} \to q'\begin{bmatrix}a\\Y\end{bmatrix}$ für alle $a \in \Sigma \cup \{„\varepsilon"\}$ und alle $q \in Q$, $X, Y \in \Gamma$ mit $\delta(q, X) = (q', Y, N)$
- $\begin{bmatrix}a\\X\end{bmatrix} q_F \to q_F\, a\, q_F$ für alle $a \in \Sigma \cup \{„\varepsilon"\}$, $X \in \Gamma$, $q_F \in F$

$q_F \begin{bmatrix}a\\X\end{bmatrix} \to q_F\, a\, q_F$

$q_F \to \varepsilon$

$$\boxed{G_M = (V = Q \cup (\Sigma \cup \{„\varepsilon"\} \times \Gamma) \cup \{S, A_1, A_2\}, \Sigma, P, S)}$$

⌛

3.7 Kontextfreie Sprachen

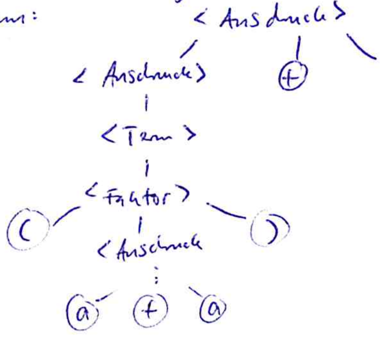
Eine Sprache L heißt kontextfrei (kf), wenn es eine kf. Grammatik G mit $L = L(G)$ gibt.

G_1: $S \to aSb \mid \varepsilon$ $\quad L(G_1) = \{a^n b^n \mid n \geq 0\}$ ⇝ müsste man zeigen

G_2: <Ausdruck> → <Term> | <Ausdruck> + <Term>
<Term> → <Faktor> | <Term> ∗ <Faktor>
<Faktor> → a | (<Ausdruck>) $\quad \Sigma = \{+, *, a, (,)\}$

Linksableitung: es wurde in jeder Satzform die linkeste Variable ersetzt

Syntaxbaum:



Die Blätter von links nach rechts gelesen ergeben das Wort $w \in L(G_1)$

Aus diesem Baum kann einfach die Übersetzung des Ausdrucks in z.B. Maschinensprache durchgeführt werden.

Def 3.24 Eine kf. Grammatik ist in Chomsky-Normalform (CHNF), wenn jede Produktion von der Form $A \to BC$ $\quad B, C \in V \setminus \{S\}$ oder $A \to a$, $A \in V$, $a \in \Sigma$ ist.
Zusätzlich ist $S \to \varepsilon$ erlaubt

ℐ‖

Satz 3.25 Zu jeder kf. Grammatik G kann eine kf Gramm. G' in ChNF konstruiert werden mit $L(G) = L(G')$

Bew:

Lemma 3.26: Zu G gibt es eine kf. Grammatik $G_{\varepsilon\text{-frei}}$ mit $L(G) = L(G_{\varepsilon\text{-frei}})$ und $G_{\varepsilon\text{-frei}}$ enthält keine ε-Regeln (bis auf $S \to \varepsilon$)

↳ $E_0 = \{A \mid (A \to \varepsilon) \in P\}$

$E_i = \{A \mid A \to B_1, \ldots, B_k,\ B_1, \ldots, B_k \in E_{i-1}\} \cup E_{i-1}$

Es gibt ein i_0 mit $E_{i_0} = E_{i_0+1}$

E_{i_0} enthält genau alle Variablen A mit $A \overset{*}{\to} \varepsilon$.

Lösche alle ε-Regeln

$\forall (A \to w) \in P.\ w \in (V \cup \Sigma)^*$. Wenn w k Variablen enthält, die in E_{i_0} sind, dann füge alle bis zu $2^k - 1$ mögl. Regeln, die man durch Weglassen von Var. aus E_{i_0} in w erhalten kann, hinzu, außer $A \to \varepsilon$ würde dadurch herauskommen.

Bsp: $A \to BaAbC \qquad E_{i_0} = \{B, C\}$

wird zu $A \to BaAbC$

$A \to aAbC$

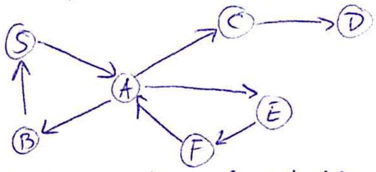
$A \to BaAb$

$A \to aAb$

Lemma 3.27 Zu jeder kf. Grammatik G gibt es eine kf. Gramm. G' ohne Kettenregeln $(A \to B)$ mit $L(G) = L(G')$

↳ Konstruiere den gerichteten Graphen $G_{Kette} = (V, E_{Kette})$, in der die Knoten die Var. sind und die Kanten die Kettenregeln

$S \to A,\ A \to B,\ B \to S,\ A \to E,\ E \to F,\ F \to A,\ A \to C,\ C \to D$



Ersetze in den Großkreisen (d.h. den starken ZSH komponenten, d.h. den Mengen an Knoten, die sich ggseitig erreichen können) die Var. durch eine davon (wenn S dabei, dann S) und ersetze diese in allen Produktionen an Stelle der eliminierten ein. Ersetze nun im übriggebliebenen gerichteten kreisfreien Graphen „von unten nach oben" (umgekehrt topologische Reihenfolge) die Regeln $A \to B$ durch $A \to$ „alle rechten Seiten von B"

Ab jetzt sind die ε-Regeln und Kettenregeln weg

- Für alle Terminalsymb. $a \in \Sigma$ füge die Produktion $A_a \to a$ zur Grammatik hinzu und ersetze alle Vorkommen von a in rechten Seiten von Produktionen durch A_a (außer $A \to a$)

  Jetzt: alle rechten Seiten in Produktionen sind von der Form $A \to a$ oder $A \to B_1 B_2 \ldots B_k$, $k \geq 2$

- Regeln durchnummerieren Regel i: $A \to B_1, \ldots, B_k$, $k \geq 2$ wird ersetzt durch

  $A \to B_1 C_1^{(i)}$

  $C_1^{(i)} \to B_2 C_2^{(i)}$

  $C_{k-2}^{(i)} \to B_{k-1} B_k$

[Eine Variable A heißt nutzlos, wenn es kein Wort $w \in \Sigma^*$ gibt mit $A \overset{*}{\to} w$. Sonst heißt sie nützlich]

Satz 3.28 Nutzlose Variablen bestimmt werden. Alle Produktionen, die diese enthalten, können ersatzlos gestrichen werden ohne die erzeugte Sprache zu ändern

Der CYK-Algorithmus (Cocke, Younger, Kasami)

Geg: kf. Grammatik $G = (V, \Sigma, P, S)$ in ChNF mit nur nützl. Var. und $w \in \Sigma^*$, $w = a_1 \ldots a_n$

Frage: $w \in L(G)$?

$V(i,j) = \{A \mid A \in V,\ A \overset{*}{\to} a_i \ldots a_j\}$

$w \in L(G) \iff S \in V(1,n)$

Dyn. Prog.

$i = j \quad V(i,i) = \{A \mid (A \to a_i) \in P\}$

$i \leq j \quad V(i,j) = \{A \mid (A \to BC) \in P,\ B \in V(i,k),\ C \in V(k+1,j),\ k \in \{i, \ldots, j-1\}\}$

Konstruktion der $V(i,j)$ nach der Länge $\ell = j-i$

$w\in$	a_1	a_2	a_3	a_4	a_5
0	$V(1,1)$	$V(2,2)$	$V(3,3)$	$V(4,4)$	$V(5,5)$
1	$V(1,2)$	$V(2,3)$	$V(3,4)$	$V(4,5)$	
2	$V(1,3)$	$V(2,4)$	$V(3,5)$		
3	$V(1,4)$	$V(2,5)$			
4	$V(1,5)$				

$A \to BC$ in P

$B \in V(1,1)$

$C \in V(2,2)$

dann packe

Rückwärts durch die Tabelle laufend und die Frage, wo kommt dieser Eintrag denn her, kann ein Syntaxbaum für w konstruiert werden.

Satz 3.29 Der CYK-Algo entscheidet in Zeit $O(|V|\cdot|P|\cdot|w|^3)$, ob $w\in L(G)$ ist

Bew: Correct by construction

Laufzeit: Tabelle enthält $O(|w|^2)$ Einträge. Ein Eintrag benötigt $O(|V|\cdot|P|\cdot|w|)$ Schritte $\Rightarrow$ LZ □

Corollar 3.30 $i = \{<G>w \mid G \text{ ist kf. Gramm. in CNF}, w\in L(G)\} \in P$

Das Pumping-Lemma für kf. Sprachen

Def 3.31 Sei L eine kf. Sprache. L hat die kontextfreie Pumpeigenschaft, falls gilt:

$$\exists n_L \in \mathbb{N}\ \forall z\in L, |z|\geq n_L\ \exists u,v,w,x,y\in\Sigma^*, z=uvwxy:$$
(i) $|vx|\geq 1$ (ii) $|vwx|\leq n_L$ (iii) $\forall i\geq 0: uv^iwx^iy\in L$

$i=0$: uwy

$i=0$ $uwy\in L$
$i=1$ $uvwxy\in L$
$i=2$ $uvvwxxy\in L$

Bsp 3.32 (a) $L_1 = \{abc\}$ hat kf PE. Es reicht, $n_L = 4$

allgemein: $n_L \gneq$ |kürzestes Wort in L|

(b) $L_2 = \{a^jb^j \mid j\geq 1\}$ hat kf. PE $\exists \hat{=}$ Setze

Setze $n_{L_2} = 4$, $z = a^k aa\, bb\, b^k$ mit $k\geq 0$ bel, aber fest

Setze $u = a^k a$, $v=a$, $w=\varepsilon$, $x=b$, $y=bb^k$ $(uvwxy = z\ \checkmark)$

(i) $|vx| = |ab| = 2\geq 1$ ✓

(ii) $|vwx| = |ab| = 2\leq 4$ ✓

(iii) Sei $i\in\mathbb{N}$, $i\geq 0$, bel, aber fest. $uv^iwx^iy = a^kaa^ib^ibb^k = a^{1+k+i}b^{1+k+i}$ mit $k\geq 0$, $i\geq 0$

d.h. $a^{1+k+i}b^{1+k+i}\in L_2$

(c) $L_3 = \{a^jb^jc^j \mid j\geq 0\}$ hat die kf. PE nicht

erweiterung Def 3.31 $\forall n_L\in\mathbb{N}\ \exists z\in L, |z|\geq n_L\ \forall u,w,v,x,y\in\Sigma^*, z=uvwxy$
(i) $|vx|\geq 1$ und (ii) $|vwx|\leq n_L \to$ „$\neg$(iii)" $\exists i\geq 0: uv^iwx^iy\notin L$ ⌋

Sei n_{L_3} bel. aber fest. Setze $z = a^{n_{L_3}}b^{n_{L_3}}c^{n_{L_3}}\in L_3$. Sei $uvwxy = z$ eine bel., aber feste Zerlegung von z mit $|vx|\geq 1$ und $|vwx|\leq n_{L_3}$
vwx enthält 1. nur a's, 2. nur a's und b's, 3. nur b's, 4. nur b's und c's, 5. nur c's
vwx enthält mind. eine Zeichensorte, aber höchstens 2 Zeichensorten.
Setze $i=2$

(d) $L_4 = \{a^{j^2} \mid j\geq 0\}$ hat die kf. PE. nicht
Sei n_{L_4} bel., aber fest. Setze $z = a^{n_{L_4}^2}$. Sei u,v,w,x,y bel. aber fest mit $z = uvwxy$ mit $|vx|\geq 1$ und $|vwx|\leq n_{L_4}$. Setze $i=2$
$uv^2wx^2y = a^{n_{L_4}^2+|vx|}$ Aber $n_{L_4}^2 \lneq n_{L_4}^2 + |vx| \leq n_{L_4}^2 + n_{L_4} \lneq (n_{L_4}+1)^2$

D.h. $n_{L_4}^2 + |vx|$ kann keine Quadratzahl

(e) $L_5 = \{c^iw \mid i\geq 1, w\in H_0, w\in\{0,1\}^*\}\cup\{0,1\}^*$ hat die kf PE, ist nicht entscheidbar

Setze $n_{L_5} = 5$

$z\in L_5$ mit $|z|\geq 5$

Fall 1: $z \in \{0,1\}^*$ trivial

Fall 2: $z \in \{c^i w \mid i \geq 1, w \in H_0\}$

Fall 2.1: $z = c c c^i w$. Pumpe ein c (dh. $v = c, x = \varepsilon$)

Fall 2.2: $z = cw$. Pumpe c (dh. $v = c$, $x = \varepsilon$)

Satz 3.33 (Pumping-Lemma für kf Sprachen)

L kontextfreie Sprache $\Rightarrow$ L hat die kf. PE

Bew: L kf, dh. wir haben $G = (V, \Sigma, P, S)$ mit $L(G) = L$ in Ch. NF, kf.

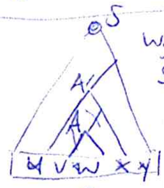
Setze $n_L = 2^{|V|}$

Sei $z \in L$, $|z| \geq 2^{|V|}$ bel, aber fest

Wir wählen auf dem längsten Pfad im Syntaxbaum von unten die erste Var. A, die doppelt vorkommt. Das doppelte Vorkommen nach spätestens $|V|$ vielen „nach oben geh"-Schritten.

Setze u, v, w, x, y wie im Bild.

Syntaxbaum für z:



wg Ch NF ist der Syntaxbaum ein binärer Baum

Level von 0 bis $\geq \log_2(n_L) = |V|$

$\rightarrow$ Anz. der Variablen auf dem längsten Weg von Wurzel zu einem Terminal ist $\geq |V| + 1$

(i) $vx \neq \varepsilon$, da G in Ch NF ist und keine ε-Regeln enthält, es kann höchstens nur ein Wert von v und x leer sein.

(ii) $|vwx| \leq n_L = 2^{|V|}$. Vom oberen A nach unten können nicht mehr als $2^{|V|}$ Zeichen erzeugt werden

(iii) Wir wissen: $A \overset{*}{\Rightarrow} vAx$, also auch $A \overset{*}{\Rightarrow} vvAxx$ ($i = 2$), also auch $A \rightarrow v^i A x^i$

Wir wissen auch: $A \overset{*}{\rightarrow} w$. Also: $A \rightarrow v^i w x^i$

Insgesamt: $S \overset{*}{\Rightarrow} uAy$, also insgesamt ableitbar

$S \overset{*}{\Rightarrow} uwy \in L$ und $S \overset{*}{\Rightarrow} uv^iwx^iy \in L$ für $i \geq 1$ ✓

$\square$

Def 3.19 (reg. Pump-Eigenschaft)

$\exists n_L \in \mathbb{N} \;\; \forall z \in L, |z| \geq n_L \;\; \exists u, v, w \in \Sigma^*, \; z = uvw$

(i) $|v| \geq 1$ (ii) $|uv| \leq n_L$ (iii) $\forall i \geq 0$: $uv^iw \in L$

Satz 3.20 L reg. $\Rightarrow$ L hat die reg. PE

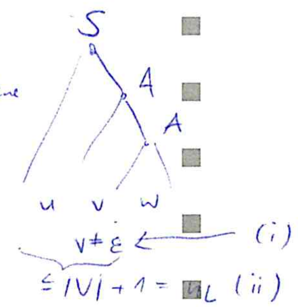
Bew: G sei Gramm. vom Typ Chomsky-3 mit $L = L(G)$, ohne ε-Regeln.

Nur $A \rightarrow a$, $A \rightarrow aB$ als Regeltyp. $G = (V, \Sigma, P, S)$

Setze $n_L = |V| + 1$. Dh. im Syntaxbaum liegt auf dem längsten Pfad mind. eine Var. A doppelt. Sei A die erste doppelt vorhandene Var. von unten

(iii) wie beim Pumpinglemma für kf Sprachen

$\square$



Die Chomsky-Hierarchie

Sei L_i, $i \in \{0, 1, 2, 3\}$ die Klasse der Sprachen vom Typ Chomsky-i.

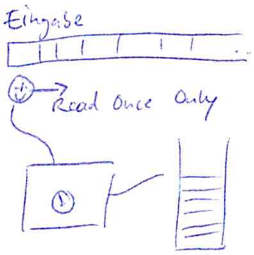
Satz 3.34 $L_3 \subsetneq L_2 \subsetneq L_1 \subsetneq \mathcal{E} \subsetneq L_0 \subsetneq$ „Rest"

L_3: $\{a^i b^j \mid i, j \geq 0\}$; L_2: $\{a^n b^n \mid n \geq 0\}$; L_1: $\{a^n b^n c^n \mid n \geq 0\}$; $\mathcal{E}$: entscheidbare Sprachen; $L_0 \ni H$; „Rest" $\ni \overline{H}$

Kellerautomaten

Def. 3.35: Ein nichtdeterministischer Kellerautomat (NPDA, nondeterministic pushdown acceptor) ist beschrieben durch 7 Komponenten $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit

- Q: Zustände
- Σ: Eingabealphabet
- Γ: Kelleralphabet
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to \underset{\text{Potenzmenge}}{P}(Q \times \Gamma^*)$
- Z_0: $Z_0 \in \Gamma$, Kellergrund-Symbol
- F, $F \subseteq Q$, akz. Endzustände



$L = \{w w^R \mid w \in \{0,1\}^*\}$

$x \in L(M) \iff$ es gibt eine Rechnung, die einen Zustand $q \in F$ erreicht und jedes Zeichen der Eingabe x wurde gelesen

Satz 3.36 L ist kf $\iff$ es gibt einen NPDA M mit $L = L(M)$

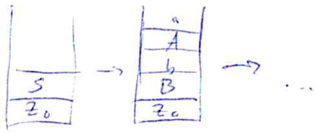
Bew „$\Leftarrow$" extrem schwer, nutzt Nichtdet. auf höchst clevere Art und Weise

„$\Rightarrow$" Sei $G = (V, \Sigma, S, P)$ eine kf. Gramm. für L. Der NPDA M vollzieht eine Linksableitung für das Eingabewort x nach.

Bsp: $S \to \varepsilon \mid aAbB$, $A \to ab \mid aAb$, $B \to aBb \mid aa$

$S \to aAbB \to aaAbbB \to aaabbbB \to aaabbbaa$

a	a	a	b	b	b	a	a



$F = \{Z_0\} \cup V \cup \Sigma$

$Q = \{q_0, \bar{q}_0, q_{accept}\}$

$\delta(q_0, \varepsilon, Z_0) = \{(\bar{q}, \boxed{\begin{smallmatrix} S \\ Z_0 \end{smallmatrix}})\}$

für alle $A \in V$: $\delta(\bar{q}, \varepsilon, A) = \{(\bar{q}, w) \mid (A \to w) \in P\}$

für alle $a \in \Sigma$: $\delta(\bar{q}, a, a) = \{(\bar{q}, \varepsilon)\}$, $\delta(\bar{q}, \varepsilon, Z_0) = \{(q_{accept}, Z_0)\}$

Abschlusseigenschaften ~~für~~ kf. Sprachen

Satz 3.37: Die kf. Sprachen sind abgeschlossen unter $\cup, \circ, (.)^*$

$L_1 \cup L_2$: $S \to S_1 \mid S_2$, $L_1 \circ L_2$: $S \to S_1 S_2$, L_1^*: $S \to S S_1 \mid \varepsilon$ D.

Satz 3.38: Die kf Sprachen sind nicht abgeschlossen unter $\cap$ und Komplement

Endliche Automaten

Def. 3.39 Ein det. endl. Automat (DFA deterministic finite automaton)
A ist beschrieben durch 5 Komponenten $A = (Q, \Sigma, \delta, q_0, F)$ mit

- Q Zustände
- Σ Eingabealphabet
- $\delta: Q \times \Sigma \to Q$
- q_0 Startzust.
- F, $F \subseteq Q$ akz. Endzustände

Def 3.40 Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

- Die kanonische Fortsetzung von δ auf Wörter $w = w_1 w_2 \ldots w_n \in \Sigma^n$ wie folgt:
  $\delta: Q \times \Sigma^* \to Q$
  $\delta(q, w) = q'$ falls $\delta(\delta(q, w_1), w_2 \ldots w_n) = q'$
  oder auch $\delta(\delta(q, w_1 \ldots w_{n-1}), w_n)$
- A akz. $w \in \Sigma^*$, falls $\delta(q_0, w) \in F$ — immer ein Zeichen wird verarbeitet, Akzeptanz erfordert, dass das ganze w verarbeitet wird
- $L(A) = \{w \mid \delta(q_0, w) \in F, w \in \Sigma^*\}$

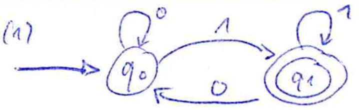
BFS-VL

Datum

Name/Projekt

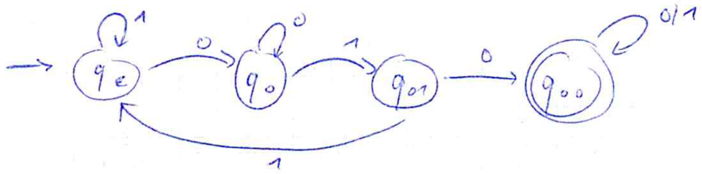
methodpark

~~Bsp~~ Bsp: (1)



$L(A_1) = \{w \mid w \in \{0,1\}^*, w \text{ endet mit } 1\}$

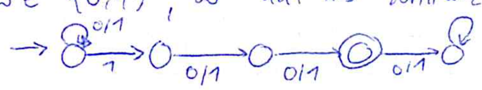
(3)



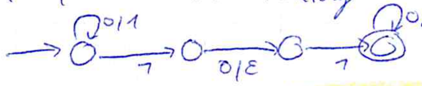
$L(A_3) = \{w \mid w \in \{0,1\}^*, w \text{ enthält } 010\}$

Nicht-deterministische endliche Automaten

$L_1 = \{w \mid w \in \{0,1\}^*, w$ hat als drittletztes Symbol $1\}$



$L_2 = \{w \mid w$ enthält Teilfolge 101 oder 11$\}$



$\delta(q_1, \varepsilon) = \{q_2\}$

$\delta(q_0, 1) = \{q_0, q_1\}$

Def 3.7 Ein nichtdeterministischer endl. Automat (NFA) N ist beschrieben durch die 5 Komponenten $N = (Q, \Sigma, \delta, q_0, F)$ mit

- Q: Endl. Menge d. Zustände
- Σ endl. Alphabet, $Q \cap \Sigma = \emptyset$
- $\delta: Q \times \Sigma_\varepsilon \to P(Q)$, wobei P die Potenzmenge ist und $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
- q_0: Startzustand
- F: $F \subseteq Q$, akzeptierende Endzustände

N akzeptiert $w \in \Sigma^*$, falls $w = w_1 \dots w_n \in \Sigma_\varepsilon^*$ und $\delta(q_0, w_1 \dots w_n) \cap F \neq \emptyset$

Die kanonische Fortsetzung von δ ist def. als $\delta: Q \times \Sigma_\varepsilon^* \to P(Q)$,

$$\delta(q, w) = \delta(q, w_1 \dots w_n) = \{r \in Q \mid \exists q' \in \delta(q, w_1 \dots w_{n-1}): r \in \delta(q', w_n)\} \text{ mit } w_i \in \Sigma_\varepsilon$$
$$= \bigcup_{q' \in \delta(q, w_1 \dots w_{n-1})} \delta(q', w_n)$$

Satz 3.9 Sei N ein NFA. Dann gibt es einen DFA A mit $L(N) = L(A)$

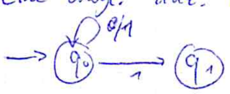
Bew.: ~~Sei N ein NFA. Dann gibt es ein~~

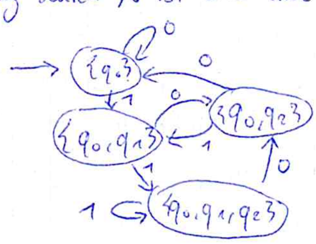
$N = (Q, \Sigma, \delta, q_0, F)$ (Ann: keine ε-Übergänge)

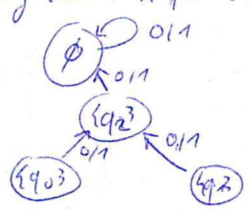
Potenzmengenkonstruktion $A = (Q', \Sigma, \delta', q_0', F')$

- $Q' = P(Q)$ • $q_0' = \{q_0\}$ • $F' = \{R \mid R \cap F \neq \emptyset\}$ • $R \subseteq Q \; (R \in P(Q))$, $R \in Q'$
- $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$

Eine mögl. akz. Rechnung durch N ist eine akz. Rechnung durch A für das gleiche Wort w und umgekehrt







$\delta(q_0, 1) = \{q_0, q_1\}$

$\delta(q_2, 0) = \{q_2\}$

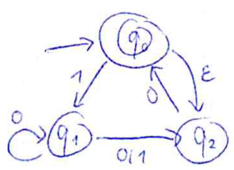
$L(N) = L(A)$, da $w \in L(N) \Leftrightarrow \delta(q_0, w) \cap F \neq \emptyset \Leftrightarrow \delta'(q_0', w) \in F' \Leftrightarrow w \in L(A)$

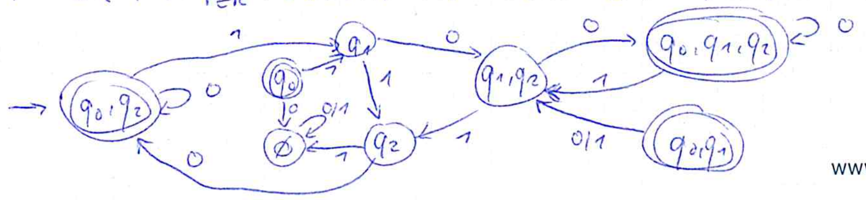
Jetzt: auch ε-Übergänge in δ.

$E(R) = \{q \in Q \mid$ von $r \in R$ kann q durch ε-Übergänge erreicht werden$\}$

wg. null möglicher ε-Übergängen ist $R \subseteq E(R)$

• $q_0' = E(\{q_0\})$ • $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$ • $F' = \{R \in Q' \mid E(R) \cap F \neq \emptyset\}$, • $Q' = P(Q)$



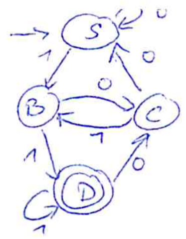


$L_{DFA} = L_{NFA}$

Zu DFA A eine Typ-3-Grammatik angeben.

$S \to 0S \mid 1B$
$B \to 0C \mid 1D$
$C \to 1B \mid 0S \mid \varepsilon$
$D \to 1D \mid 0C \mid \varepsilon$

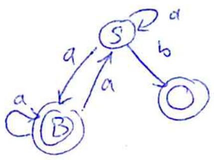
"<=>"



Zu Typ-3-Grammatik einen NFA konstr.

$S \to aB \mid aS \mid b$
$B \to aB \mid aS \mid \varepsilon$

<=>



$\Rightarrow L_{DFA} = L_{NFA} = L_3$

3.6 Regulärer Ausdrücke und Abschlusseigenschaften regulärer Sprachen

Def 3.21 Sei Σ ein endl. Alphabet. R ist ein regulärer Ausdruck über Σ, wenn R wie folgt aufgebaut ist

1. a für $a \in \Sigma$
2. ε
3. $\emptyset$
4. $(R_1 \cup R_2)$ mit R_1, R_2 reg. Ausdrücke
5. $(R_1 \circ R_2)$ " " "
6. (R_1^*) " R_1 " "

Semantik:

1. $R = a$ $\quad L(R) = \{a\}$
2. $R = \varepsilon$ $\quad L(R) = \{\varepsilon\}$ $\qquad \varepsilon \neq \emptyset$
3. $R = \emptyset$ $\quad L(R) = \{\}$
4. $R = R_1 \cup R_2$ $\quad L(R) = L(R_1) \cup L(R_2)$
5. $R = (R_1 \circ R_2)$ $\quad L(R) = L(R_1) \circ L(R_2)$
6. $R = (R_1^*)$ $\quad L(R) = L(R_1)^*$ insb. $L(\emptyset^*) = \{\varepsilon\}$

Bsp $L(0^* 1 0^*)$ $\circ$ wird wie bei Multiplikation übl. vorgelesen, $\circ$ vor $\cup$
$= \{w \mid w \in \{0,1\}^*, w$ enthält genau eine $1\}$

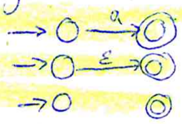
$L((0\cup 1)^* 1 (0 \cup 1)^*) = \{w \mid w \in \{0,1\}^*, w$ enthält mind. eine $1\}$

$L(0(0\cup 1)^* 0 \cup 1 (0 \cup 1)^* 1) \cup 0 \cup 1) = \{w \mid w \in \{0,1\}^*$, erstes u. letzte sind gleich$\}$

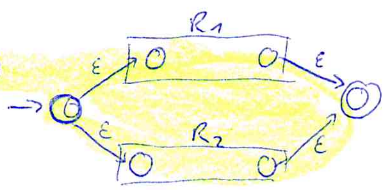
Satz 3.22: Sei $L \subseteq \Sigma^*$ eine Sprache.
L ist genau dann regulär, wenn es einen reg. Ausdruck R über Σ gibt mit $L = L(R)$

Bew: L reg $\Leftarrow$ reg. Ausdruck R

1. $R = a$
2. $R = \varepsilon$
3. $R = \emptyset$

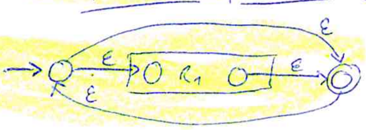


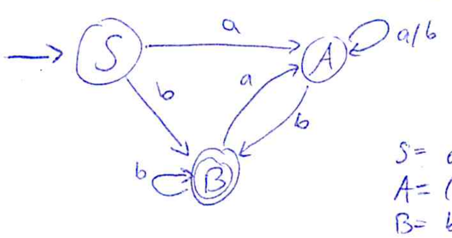
4. $R = (R_1 \cup R_2)$



5. $R = (R_1 \circ R_2)$

6. $R = (R_1^*)$





Wir schreiben $L(S)$ für die Sprache, die akz. wird, falls S der Startzustand ist.
Entsprechend $L(A)$ und $L(B)$. Zur vereinfachung lassen wir das L weg

$S = aA \cup bB = aA \cup bb^*(aA \cup \varepsilon)$
$A = (a \cup b)A \cup bB = (a\cup b)A \cup bb^*(aA \cup \varepsilon)$
$B = bB \cup aA \cup \varepsilon = b^*(aA \cup \varepsilon)$

$A = (a \cup b) A \cup bb^*(aA \cup \varepsilon) = (a \cup b \cup bb^*a)A \cup bb^* = (a \cup b \cup bb^*a)^* bb^*$
$S = aA \cup bb^*(aA \cup \varepsilon) = (a \cup bb^*a)A \cup bb^*$

proof by example $\square$

Satz 3.23: Die reg. Sprachen sind abgeschlossen unter
- Vereinigung
- Konkatenation
- Durchschnitt
- Sternabschluss
- Komplementbildung
- Spiegelung

Quickies :3

Datum

Name/Projekt

methodpark

1 L beliebige Sprache. Ist L nicht rekursiv aufzählbar, so ist das Komplement ebenfalls nicht rekursiv aufzählbar
Antwort falsch: $L = \overline{H}$, $\overline{H}$ nicht r.a, H jedoch schon.

2 L_1 und L_2 bel. Sprachen, $L_1 \neq L_2$, $L_1 \cap L_2$ entscheidbar. -> L_1 oder L_2 auch entscheidbar
Antwort falsch: $L_1 = H$, $L_2 = \overline{H}$ nicht entscheidbar, $H \cap \overline{H} = \emptyset$ jedoch entscheidbar

3 L_1 und L_2 rekursiv aufzählbar. Dann auch $L_1 \cap L_2$ rekursiv aufzählbar
Antwort wahr: TM modellieren, die beide gleichzeitig laufen lässt.

4 Ist L kontextfrei, so auch jede echte Teilmenge von L
Antwort falsch: $L = \{0,1\}^*$ kontextfrei, $\overline{H}$ nicht

5 L_1 unentscheidbar, $L_1 \subseteq L_2$. L_2 unentscheidbar? ‖ L_2 entscheidbar, $L_1 \subseteq L_2$, L_1 entscheidbar?
Antwort: falsch/nein
$L_2 = \{\langle M \rangle \mid M$ ist det. 1-Band-TM$\}$, entscheidbar
$L_1 = H_\varepsilon$, unentscheidbar

6 L_1 und L_2 beliebige Sprachen über $\Sigma = \{0,1\}$. $L_1 \cup L_2$ entscheidbar, => mind. eine der beiden Sprachen entscheidbar
Antwort: falsch: $L_1 = H$, $L_2 = \{\langle M \rangle w \mid M$ ist det. 1-Band-TM und M gestartet mit w hält nicht$\}$

7 Jede Sprache mit kontextfreier Pumpeigenschaft enthält unendlich viele Wörter
Antwort: falsch: $L = \{a\}$ mit $S \to a$ regulär und damit auch kontextfrei

8 L_1 entscheidbar, L_2 rekursiv aufzählbar. ist dann $L_1 \setminus L_2$ rekursiv aufzählbar
Antwort: nein: $L_1 = \{0,1\}^*$, $L_2 = H$ -> $L_1 \setminus L_2 = \overline{H}$ ist nicht r.a.

9 Es gibt kontextfreie Sprache L, sodass $\overline{L}$ unentscheidbar ist
Antwort: falsch: L ist kf -> L ist entscheidbar -> $\overline{L}$ ist entscheidbar (Vertauschung von 0 und 1 in char. Fkt)

10 $L = \{\langle G, w \rangle \mid G$ in ChNF, $w \in L\}$. $L \in NP$?
Antwort ja: ob G in ChNF in Poly. Zeit durch Syntaxanalyse entscheidbar, ob $w \in L(G)$ mit CYK-Algo. in Polynomzeit entscheidbar -> $L \in P \subseteq NP$ ✓

11 CLIQUE ist NP-schwer, da $SAT \leq_p CLIQUE$. Ist auch $CLIQUE \leq_p SAT$?
Antwort ja: SAT NP-vollständig, also auch NP-schwer -> $\forall L \in NP: L \leq_p SAT$ -> also $CLIQUE \leq_p SAT$

12 Alle Sprachen in NP sind entscheidbar
Antwort ja: $L \in NP$ -> ex. poly. Verifizierer für L -> Verifizierer ist Entscheider für L

13 Unbekannt, ob es reg. Sprachen gibt, die außerhalb von NP liegen
Antwort nein: alle Sprachen in NP sind entscheidbar, reguläre Sprachen sind entscheidbar

14 L beliebige unentscheidbare Sprache. Dann besteht L aus unendlich vielen Wörtern
Antwort ja, endliche Sprachen sind regulär und damit entscheidbar

15 $CLIQUE \leq H$?
Antwort ja: FU, ob Element in CLIQUE → ja: festes Element aus H; → nein: festes Element aus $\overline{H}$
H NP-schwer -> $\forall L \in NP: L \leq_p H$, insbesondere $L = CLIQUE$

$\boxed{16}$ $G = (V,E)$ ungerichtetes Graph, V' Teilmenge von V
Es gibt keine Registermaschine, die in poly. Zeit entscheidet, ob V' eine CLIQUE bildet
Antwort: falsch. $\exists$ poly. Verifizierer für CLIQUE, TM kann auch als Registermaschine simuliert werden

$\boxed{17}$ $L = \{ \langle M \rangle \mid M$ 1-Band-DTM, die gest. mit leerem Band nach max. $|\langle M \rangle|^2$ Schritten hält$\}$ ist unentscheidbar
Antwort: falsch: es können $|\langle M \rangle|^2$ Schritte simuliert werden

$\boxed{18}$ L_1 und L_2 kontextfrei $\rightarrow L_1 \cap L_2$ unentscheidbar
Antwort: falsch: L_1 und L_2 kf $\rightarrow L_1$ und L_2 entscheidbar, Schnitt zweier entscheidbarer Sprachen ebenso

$\boxed{19}$ ~~$\overline{H}$~~ $H \leq SAT$
Antwort falsch: dann wäre SAT nicht entscheidbar, ist es aber, da $SAT \in NP$

$\boxed{20}$ Es gibt reguläre L, sodass es L', $L' \subset L$ gibt mit L' unentscheidbar
Antwort: wahr: $L = \{0,1\}^*$, $L' = H$

∽ BFS - Kurzzusammenfassung ∽

Datum

Name/Projekt

methodpark

det. 1-Band-TM: $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit

- Q: Zustände
- Σ: endl. Eingabealphabet
- Γ: endl. Bandalphabet mit $\Sigma \subseteq \Gamma$
- B: Blank, $B \in \Gamma$, $B \notin \Sigma$ (Leerzeichen
- q_0: $q_0 \in Q$, Startzustand
- F: akz. Endzustände, $F \subseteq Q$
- δ: Programm $\delta: Q \times \Gamma \to Q \times \Gamma \times \{R, L, N\}$

bis auf Eingabe ist das Band leer
Lese/Schreibkopf steht auf erstem Zeichen der Eingabe

$L \subseteq \Sigma^* \to L$ Sprache über Alphabet Σ

Konfiguration: $K = \alpha q \beta$ <=> auf Band $\alpha\beta$, Kopf unter erstem Zeichen von β, Zustand q

M akzeptiert $x \in \Sigma^*$, falls $q_0 x \vdash^* \alpha q \beta$ mit $q \in F$. $L(M)$ Menge aller akzeptierten $x \in \Sigma^*$. M akz. $L(M)$

↳ $L \subseteq \Sigma^*$ rekursiv aufzählbar <=> $\exists M$ mit $L(M) = L$

M akzeptiert L und hält für alle Eingaben → M entscheidet L

↳ $L \subseteq \Sigma^*$ entscheidbar/rekursiv <=> $\exists M$, die L entscheidet

det. k-Band-TM: hat k Bänder und k Köpfe, Eingabe steht auf Band 1

Zeitkomplexität: # Schritte, bis gehalten wird $\geq$ Platzkomplexität: # versch. Zellen, die besucht werden

$t(n)$-zeitbeschr., $s(n)$-platzbeschr. k-Band-TM kann durch $O(t(n) \cdot s(n))$-zeitbeschr., $O(s(n))$-platzbeschr. 1-Band-TM simuliert werden.

↳ liefern gleiche Mengen entscheidbarer und rekursiv aufzählbarer Sprachen und berechenbarer Funktionen.

RAM ($t(n)$-zeitbeschr.) kann durch det. TM simuliert werden ($O(t(n)^3)$-zeitbeschr.)

det. 1-Band-TM ($t(n)$-zeitbeschr.) kann durch RAM simuliert werden ($O(t(n))$-zeitbeschr.)

Gödelnummer: Kodierung einer TM M aus 0en und 1en, geschrieben $\langle M \rangle$

Universelle TM: verhält sich bei Eingabe $\langle M \rangle x$ so, wie M gestartet mit Eingabe x

Halteproblem: $H = \{\langle M \rangle w \mid M$ ist det. 1-Band-TM, die gest. mit Eingabe w, hält$\}$ ist unentscheidbar

initiales HP: $H_\varepsilon = \{\langle M \rangle \mid M$ " " " " , " " " ε, hält$\}$ ist unentscheidbar

H ist r.a, $\overline{H} = \{0,1\}^* \setminus H$ ist nicht r.a

Reduktion: L_1, L_2 Sprachen. $f: \{0,1\}^* \to \{0,1\}^*$ mit $x \in L_1 \Leftrightarrow f(x) \in L_2$, kurz $L_1 \leq L_2$

L entscheidbar: $\chi_L: \{0,1\}^* \to \{0,1\}$ mit $\chi_L = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases}$

L r.a: $\chi_L: \{0,1\}^* \to \{0,1\}$ mit $\chi_L(x) = \begin{cases} 1 & x \in L \\ \text{undef.} & x \notin L \end{cases}$

$L_1 \leq L_2$. (a) L_2 entscheidbar => L_1 entscheidbar
(b) L_2 rekursiv aufzählbar => L_1 rekursiv aufzählbar

L rekursiv aufzählbar <=> es gibt total berechenbare ~~Bij~~ surjektive Fkt. $g: \{0,1\}^* \to L$

L entscheidbar <=> L und $\overline{L}$ sind rekursiv aufzählbar

L_1 und L_2 r.a. Sprachen. (a) $L_1 \cup L_2$ ist r.a. (b) $L_1 \cap L_2$ ist r.a.

1-Band-NTM: wie det. 1-Band-NTM, nur $\delta: Q \times \Gamma \to P(Q \times \Gamma \times \{L, R, N\})$

kann durch det TM M' simuliert werden mit $2^{O\,t(n)}$ Zeit und $O(s(n) \cdot t(n))$ Platz

NTMs akzeptieren genau die rekursiv aufzählbaren Sprachen.

BFS - Kurzzusammenfassung

DTIME $(t(n)) := \{L \mid$ gibt det. $O(t(n))$-zeitbeschr. TM, die L entscheidet$\}$ $\rightarrow P = \bigcup_{k\in\mathbb{N}} DTIME(n^k)$

NTIME $(t(n)) := \{L \mid$ gibt nichtdet. $O(t(n))$-zeitbeschr. TM, die L akzeptiert$\}$ $\rightarrow NP = \bigcup_{k\in\mathbb{N}} NTIME(n^k)$

CLIQUE $:= \{\langle G,k\rangle \mid k\in\mathbb{N}$, G ungerichteter Graph, der vollst. Teilgraphen der Größe$\}$

HC $:= \{\langle G\rangle \mid G$ ist ungerichteter Graph und enthält Hamiltonkreis (jeder Knoten genau einmal)$\}$

TSP $:= \{\langle G,c,k\rangle \mid G$ mit $c: E \rightarrow \mathbb{R}$ enthält Rundreise mit Gewicht $\leq k\}$

VC $:= \{\langle G,k\rangle \mid k\in\mathbb{N}$, G ist ungerichteter Graph und hat Knotenüberdeckung der Größe $k\}$

BP $:= \{\langle A,b\rangle \mid A\in M_{n,m}(\mathbb{Z}) = \mathbb{Z}^{n\times m}, b\in\mathbb{Z}^n, \exists y\in\{0,1\}^n : Ay\leq b\}$

Verifizierer, $t(n)$-beschränkt, TM V_L mit (i) Eingabe $x\#w$
(ii) Laufzeit in $O(t(|x|))$
(iii) $x\in L \Leftrightarrow \exists w: |w|\leq t(|x|)$ und V_L akz. $x\#w$

$NP = \{L \mid$ es gibt polynomiellen Verifizierer für $L\}$

$L\in NTIME(t(n)) \Leftrightarrow$ es gibt $t(n)$-beschränkten Verifizierer V_L für L

NP-schwer: $\forall L'\in NP: L' \leq_p L$

NP-vollständig: (i) $L\in NP$ (ii) L ist NP-schwer

L sei NP-schwer. Dann gilt: (a) $L\in P \Rightarrow P = NP$
(b) $P\neq NP \Rightarrow L\notin P$

L sei NP-vollständig. Dann gilt: (a) $L\in P \Leftrightarrow P = NP$
(b) $L\notin P \Leftrightarrow P\neq NP$
(c) $L'\in NP$ und $L\leq_p L' \Rightarrow L'$ ist NP-vollständig

SAT $:= \{\langle\phi\rangle \mid \phi$ ist erfüllbare KNF$\}$ ist NP-vollständig

kSAT $:= \{\langle\phi\rangle \mid \phi$ sei erfüllbare KNF, in der jede Klausel aus k Literalen über k versch. Var. besteht$\}$

2SAT $\in P$

3SAT ist NP-vollständig: (i) 3SAT $\in NP$ (ii) NP-schwer, da SAT $\leq_p$ 3SAT

CLIQUE ist NP-vollständig: (i) CLIQUE $\in NP$ (ii) NP-schwer, da SAT $\leq_p$ CLIQUE

VC ist NP-vollständig: (i) VC $\in NP$ (ii) NP-schwer, da CLIQUE $\leq_p$ VC

BP ist NP-vollständig: (i) BP $\in NP$ (ii) NP-schwer, da SAT $\leq_p$ BP

BFS-Kurzzusammenfassung

Grammatik $G = (V, \Sigma, P, S)$ mit
- V: Endl. Menge von Variablen
- Σ: Endl. Menge der Terminale / Terminalsymbole
- S: $S \in V$: Startsymbol
- $P \subseteq ((V \cup \Sigma)^+ \setminus \Sigma^*) \times (V \cup \Sigma)^*$: Endl. Menge von Produktionen

rekursiv aufzählbar, $\mathcal{L}_0$, Chomsky-0, TM

kontextsensitiv, wenn für alle Produktionen $u \to v$ $|u| \leq |v|$ gilt — $\mathcal{L}_1$, Chomsky-1, lin. beschr. TM
kontextfrei, wenn für alle $u \to v \in P$ gilt: $u \in V$ (und damit $|u| = 1$) — $\mathcal{L}_2$, Chomsky-2, Kellerautomaten
regulär, wenn für alle $u \to v \in P$ gilt:
$u \in V$ und $v \in \{\varepsilon\} \cup \Sigma$ oder $v \in \Sigma \circ V$ — $\mathcal{L}_3$, Chomsky-3, Automaten

L rekursiv aufzählbar $\Leftrightarrow$ gibt Chomsky-0-Grammatik G mit $L = L(G)$

kontextfreie Grammatik in Chomsky-Normalform, wenn alle $p \in P$ $A \to BC$ oder $A \to a$ $(A \in V, a \in \Sigma, B, C \in V \setminus \{S\})$
G kontextfrei $\to$ gibt G ε-frei ohne ε-Regeln bis auf $S \to \varepsilon$
$\to$ gibt G' ohne Kettenregeln
CYK-Algo entscheidet in $O(|V| \cdot |P| \cdot |w|^3)$, ob $w \in L(G)$
$L = \{\langle G \rangle w \mid G$ ist kf. Grammatik in ChNF, $w \in L(G)\} \in P$

kontextfreie Pumpeigenschaft:
$\exists n_L \in \mathbb{N}\ \forall z \in L, |z| \geq n_L\ \exists u,v,w,x,y \in \Sigma^*, z = uvwxy$:
(i) $|vx| \geq 1$ (ii) $|vwx| \leq n_L$ (iii) $\forall i \geq 0: uv^iwx^iy \in L$

nicht kontextfreie Pumpeigenschaft:
$\forall n_L \in \mathbb{N}\ \exists z \in L, |z| \geq n_L\ \forall u,v,w,x,y \in \Sigma^*, z = uvwxy$:
(i) $|vx| \geq 1$ (ii) $|vwx| \leq n_L$ $\Rightarrow \neg$(iii) $\exists i \geq 0: uv^iwx^iy \notin L$

L kontextfrei $\Downarrow$ L hat kf. Pumpeigenschaft

reguläre Pumpeigenschaft:
$\exists n_L \in \mathbb{N}\ \forall z \in L, |z| \geq n_L\ \exists u,v,w \in \Sigma^*, z = uvw$
(i) $|v| \geq 1$ (ii) $|uv| \leq n_L$ (iii) $\forall i \geq 0: uv^iw \in L$

nicht reguläre Pumpeigenschaft:
$\forall n_L \in \mathbb{N}\ \exists z \in L, |z| \geq n_L\ \forall u,v,w \in \Sigma^*, z = uvw$
(i) $|v| \geq 1$ (ii) $|uv| \leq n_L$ $\Rightarrow \neg$(iii) $\exists i \geq 0: uv^iw \notin L$

L regulär $\Downarrow$ L hat reg. Pumpeigenschaft

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{E} \subsetneq \mathcal{L}_0 \subsetneq \text{„Rest"}$$

$\{a^ib^j \mid i,j \geq 0\}$ — $\{a^nb^n \mid n \geq 0\}$ — $\{a^nb^nc^n \mid n \geq 0\}$ — entscheidbare Sprachen — H^ε — $\overline{H}^\varepsilon$

Kontextfreie Sprachen: abgeschlossen unter $\cup, \circ, ^*$
nicht abgeschlossen unter $\cap$ und Komplement

reguläre Sprachen: abgeschlossen unter $\cup, \circ, ^*, \cap$, Komplement, Spiegelung

CYK: $V(i,i) = \{A \mid (A \to a_i) \in P\}\ \forall i \in \{1,..,n\}$
$V(i,j) = \{A \mid (A \to BC) \in P, B \in V(i,k), C \in V(k+1,j), k \in \{i,..,j-1\}\}$
$\to$ geg: kf Grammatik und Wort w. CYK beantwortet Frage $w \in L(G)$?

BFS - Kurzzusammenfassung

Datum

Name/Projekt

nichtdeterministischer Kellerautomat (NPDA, nondeterministic pushdown acceptor) $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit

- Q: Zustände
- q_0: Startzustand
- Σ: Eingabealphabet
- Γ: Kelleralphabet
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow P(Q \times \Gamma^*)$
- Z_0: $Z_0 \in \Gamma$, Kellergrund-Symbol
- F, $F \subseteq Q$: akz. Endzustände

$x \in L(M) \Leftrightarrow$ gibt Rechnung, die einen Zustand $q \in F$ erreicht <u>und</u> jedes Zeichen der Eingabe x wurde gelesen

L ist kontextfrei $\Leftrightarrow$ gibt NPDA M mit $L = L(M)$

endlicher Automat (DFA, deterministic finite automaton) $A = (Q, \Sigma, \delta, q_0, F)$ mit

- Q Zustände
- Σ Eingabealphabet
- $\delta: Q \times \Sigma \rightarrow Q$
- q_0 Startzustand
- F, $F \subseteq Q$ akz. Endzustände

A akz. $w \in \Sigma^*$, falls $\delta(q_0, w) \in F$. Akzeptanz erfordert, dass das ganze w verarbeitet wird

nichtdet. endl. Automat (NFA, nondeterministic finite automaton) $N = (Q, \Sigma, \delta, q_0, F)$ wie DFA außer $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$

Sei N ein NFA. Dann gibt es einen DFA A mit $L(N) = L(A)$

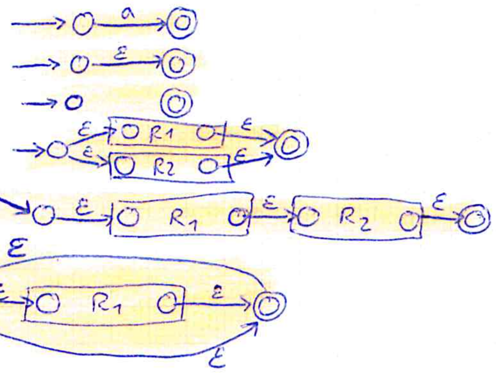
L regulär $\Leftrightarrow$ gibt DFA A mit $L = L(A)$

L regulär $\Rightarrow$ gibt NFA, der L akzeptiert

Regulärer Ausdruck über Σ:

1. a mit $a \in \Sigma$ — $L(R) = \{a\}$
2. ε — $L(R) = \{\varepsilon\}$
3. $\emptyset$ — $L(R) = \emptyset$
4. $(R_1 \cup R_2)$ — $L(R) = L(R_1) \cup L(R_2)$
5. $(R_1 \circ R_2)$ — $L(R) = L(R_1) \circ L(R_2)$
6. (R_1^*) — $L(R) = L(R)^*$

mit R_1, R_2 reg. Ausdrücke



L regulär $\Leftrightarrow$ gibt regulären Ausdruck über Σ mit $L = L(R)$

Datum

Name/Projekt

Aufpassen BFS

1. $L = L(G)$: $L \subseteq L(G)$ 1) Starte mit S 2) Produktionen/Regeln anwenden bis Ende
$L \supseteq L(G)$ Inv. finden, Regeln durchgehen ändert nichts an Inv.
"aus Inv. folgt „$\supseteq$"" zeigen

2. Reduktion $L_1 \leq L_2$

$\boxed{\text{unentscheidbar}}$ $L_1 = H$

$$f(x) = \begin{cases} \langle FM_{\langle M \rangle w} \rangle \ \text{[illegible]} & \text{falls } x = \langle M \rangle w \\ 0 & \text{sonst} \end{cases}$$

← total und berechenbar

$FM_{\langle M \rangle w}$:
1. Eingabe sei z
2. Starte M mit w
3. Falls < Bed, sodass L_2 erfüllt >: halte
4. Endlosschleife

zz: $x \in L_1 \Leftrightarrow f(x) \in L_2$

$x \in H \Rightarrow x = \langle M \rangle w$ und M gest. mit w hält -> Z3 wird erreicht, hält nur für $z \in L_2 \Rightarrow f(x) \in L_2$

$x \notin H \begin{cases} \Rightarrow x = \langle M \rangle w \text{ und } M \text{ " " } w \text{ hält nicht} \Rightarrow f(x) \notin L_2 \\ \Rightarrow x \neq \langle M \rangle w \Rightarrow f(x) = 0 \notin L_2 \end{cases}$

$\boxed{\text{nicht r.a}}$ $L_1 = \overline{H}$

$$f(x) = \begin{cases} \langle FM_{\langle M \rangle w} \rangle & \text{falls } x = \langle M \rangle w \\ \langle \text{sonst } M \rangle & \text{sonst} \end{cases}$$

$FM_{\langle M \rangle w}$:
1. Eingabe sei z
2. Falls < Bed, dass L_2 erfüllt >: Halte
3. Starte M mit w
4. Halte

SonstM:
1. Eingabe sei ?
2. Falls < Bed, dass L_2 erfüllt >: Halte
3. Endlosschleife

$x \in \overline{H} \Rightarrow \begin{cases} 1. \ x = \langle M \rangle w \text{ und } M \text{ gest. mit } w \text{ hält nicht} \\ \quad \Rightarrow 4 \text{ wird nicht erreicht, hält nur für Eingaben aus } L_2 \\ \quad \Rightarrow f(x) = \langle FM_{\langle M \rangle w} \rangle \in L_2 \\ 2. \ x \neq \langle M \rangle w \Rightarrow f(x) = \langle \text{sonst} M \rangle \in L_2 \end{cases}$

$x \notin \overline{H} \Rightarrow x = \langle M \rangle w$ und M gest. mit w hält => Program hält für alle Eingaben
$\Rightarrow f(x) = \langle FM_{\langle M \rangle w} \rangle \notin L_2$

BFS - Übung stefan-schwanecke@gmx.de

Datum

Name/Projekt

Blatt 0 - Aufgabe 1

(a) $30\cdot n^2+104\, n\cdot\log_2 n \le 30n^2+104n^2 \le 134n^3 = O(n^3)$

(b) $40n\,\log_2 n \ge 40n \ge n = \Omega(n)$

(c) $30n^2+104n\,\mathrm{ld}\,n \le 134n^2 = O(n^2)$ 1

$n^2 \le 30n^2 \le 30n^2+104n\,\mathrm{ld}\,n = \Omega(30n^2+104n\,\mathrm{ld}\,n)$

(d) $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \le n^2$

(e) $n\,(\mathrm{ld}\,n)^2-7n = o(n^2\,\mathrm{ld}\,n)$, $\lim\limits_{n\to\infty}\frac{n\,(\mathrm{ld}\,n)^2-7n}{n^2\,\mathrm{ld}\,n} = \lim\limits_{n\to\infty}\frac{(\mathrm{ld}\,n)^2}{n\,\mathrm{ld}\,n} - \frac{7}{n\,\mathrm{ld}\,n} = \lim\limits_{n\to\infty}\frac{\mathrm{ld}\,n}{n} = 0$

(f) $\lim\limits_{n\to\infty}\frac{\sqrt{n\,\mathrm{ld}\,\mathrm{ld}\,n}}{\sqrt{n\,\mathrm{ld}\,n}} = \lim\limits_{n\to\infty}\sqrt{\frac{\mathrm{ld}(\mathrm{ld}(n))}{\mathrm{ld}(n)}} = \sqrt{0}=0$

(g) $3^{2801}n^{27} = o(1{,}0007^{n/34000})$ $\quad\lim\limits_{n\to\infty}\frac{3^{2801}n^{27}}{1{,}0007^{n/34000}} = \lim\limits_{n\to\infty}\frac{c\,n^{27}}{(1+\varepsilon)^n} = \lim\limits_{n\to\infty} c\,n^{27}\cdot(1+\varepsilon)^{-n} = 0.$

(h) $\log_2(n!) \le \log_2(n^n) = n\,\log_2 n$

(i) $3\cdot\left(\frac{n}{3}\right)^n = o\left(\left(\frac{n}{2}\right)^n\right)$ $\quad\lim\limits_{n\to\infty}\frac{3\cdot\left(\frac{n}{3}\right)^n}{\left(\frac{n}{2}\right)^n} = 3\lim\limits_{n\to\infty}\frac{2^n}{3^n}3 = 0$

(j) $\lim\limits_{n\to\infty}\left(\frac{2}{e}\right)^n = 0$

(k) $\max\{f_1(n), f_2(n)\} \le \max\{f_1(n),f_2(n)\}+\min\{f_1(n),f_2(n)\} = f_1(n)+f_2(n) = O(f_1(n)+f_2(n))$ 1

$f_1(n)+f_2(n) = \max\{f_1(n),f_2(n)\}+\min\{f_1(n),f_2(n)\} \le 2\max\{f_1(n),f_2(n)\} = c\cdot\max\{f_1(n),f_2(n)\} = O(\max\ldots)$

$f(n) = \Omega(n)$ n lässt sich abschätzen durch $f(n)$

$O(n^3)$ → obere S

Θ eine durch andere abschätzen, beide Richtungen mit $O(n)$

Blatt 0 - Aufgabe 2

(a) $n^{1/\mathrm{ld}(n)} = \left(2^{\mathrm{ld}\,n}\right)^{1/\mathrm{ld}(n)} = 2 = O(1)$

(b) $n\cdot|\sin(n\cdot\pi)| = O(0)$ da $\sin(\pi\cdot n)=0\ \forall n\in\mathbb{N}$

(c) $2^{2n+\log_2(n)} = 2^{2n}\cdot 2^{\mathrm{ld}\,n} = n\cdot 4^n$

(d) $f(n) = n^{\frac{1}{2}\sin 2n} \le \sqrt{n} = O(\sqrt{n})$

(e) $\frac{n(n+1)}{2} = O(n^2)$

(f) $\sum_{k=1}^{n} c^k = \sum_{k=0}^{n} c^k - 1 = \begin{cases} n & \text{wenn } c=1 \\ \frac{1-c^{n+1}}{1-c}-1 & \text{ansonsten}\end{cases}$ $\quad O(n)$, $O(c^n)$

(g) $n! \le \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\cdot e^{\frac{1}{12n}}$

$\le e\sqrt{2\pi n}\left(\frac{n}{e}\right)^n = O\left(\sqrt{n}\left(\frac{n}{e}\right)^n\right)$

Blatt 0 - Aufgabe 3

$\Theta(2^{\log n})$ $\quad 2^{\Theta(\log n)}$ $\quad f_1 = 2^{\mathrm{ld}\,n} = O(n)$, $f_2 = 2^{c\cdot\mathrm{ld}\,n} = n^c = O(n^c)$

Blatt 0 - Aufgabe 4

M ist $n\times m$, N ist $m\times n$ → Matrix-Multiplikation R ist $n\times n$

äußere Schleife: n-mal

darin n-mal

darin m-mal → $n^2\cdot m$ = addition 1

unten: $n\times n$ Matrix: n Schritte

Blatt 0,5 - Aufgabe 5

$q_0 000111 \vdash 0q_0 00111 \vdash 00q_0 0111 \vdash 000q_0 111 \vdash 0001q_1 11 \vdash 00011q_1 1 \vdash 000111q_1 B \vdash 00011q_2 1$
$\vdash 0001q_3 1B \vdash 000q_3 11 \vdash 00q_3 011 \vdash 0q_3 0011 \vdash q_3 00011 \vdash q_3 B00011 \vdash q_4 00011 \vdash Bq_5 0011 \vdash^* q_5 01 \vdash 0q_6 1$
$\vdash 01q_6 B \vdash 0q_2 1 \vdash q_3 0 \vdash q_3 B0 \vdash q_4 0 \vdash q_5 \vdash q_7$

$q_0 00101 \vdash 0q_0 0101 \vdash 00q_0 101 \vdash 001q_1 01$

Blatt 0,5 - Aufgabe 7

$M = (\Sigma, Q, \Gamma, \delta, q_5, F)$, $Q = \{q_0 - q_4, q_5\}$ $\Sigma = \{0,1\}$ $F = \{q_4\}$, $\Gamma = \Sigma\cup\{B\}$

	0	1	B
q_5	–	$(q_1,1,R)$	–
q_0	(q_0,B,R)	$(q_1,1,R)$	(q_4,B,L)
q_1	$(q_1,0,R)$	$(q_1,1,R)$	(q_2,B,L)
q_2	$(q_2,1,L)$	$(q_3,0,L)$	–
q_3	$(q_1,0,R)$	$(q_1,1,R)$	(q_0,B,R)
q_4	–	–	–

$q_5 11 \vdash 1q_1 1 \vdash 11q_1 B \vdash 1q_2 1 \vdash q_3 10 \vdash 1q_1 0 \vdash 10q_1 \vdash 1q_2 0$
$\vdash q_2 11 \vdash q_3 B01 \vdash q_0 01 \vdash q_0 1 \vdash 1q_1 B \vdash q_2 1 \vdash q_3 B0 \vdash q_0 0 \vdash q_0$
$\vdash q_4$

i) $n+O(1) = O(n)$, $n = \log_2 p + 1$, $\Theta(2^n)$

$(a_1, \dots, a_{n-1}, a_n)_{(2)} \leq p + \frac{p}{2} + \frac{p}{4} + \dots = \sum_{i=1}^{n} \lceil \frac{p}{2^{i-1}} \rceil \leq \sum_{i=1}^{n} 2^{(n-i+1)} = 2 \sum_{j=0}^{n-1} 2^j = 2 \cdot (2^n - 1) = \Theta(2^n)$

Blatt 0,5 - Aufgabe 6

$\Gamma = \{0-9, /, +, -, \cdot, \sqrt{\ }\}$
$\Sigma = \Gamma \setminus \{/\}$

Idee: $ax^2 + bx + c$
$= a(x^2 + 2(\frac{1}{2}\frac{b}{a})x) + c$
$= a(x^2 + 2xe + e^2 - e^2) + c$
$= a((x+e)^2 - e^2) + c$

a-i, Res

a, b, c → Arbeitsbereiche

$b/a \to d$
$\frac{1}{2} d \to e$
$e^2 \to f$
$e^2 \cdot a \to g$
$g - c \to h$
$\frac{h}{a} \to i$

$ax^2 + bx + c = 0$
$a(x^2 + dx) + c = 0$
$a(x^2 + 2ex) + c = 0$
$a((x+e)^2 - f) + c = 0$
$a(x+e)^2 - g + c = 0$
$a(x+e)^2 - h = 0$
$(x+e)^2 = i$
$\pm\sqrt{i} - e \to Res$

Blatt 2 - Aufgabe 10

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

Band[ ]
köpfe[k] ← Array aus Bandindizes
band[0] ← Eingabe, Zustand = q_S
gelesene Z[k] = func(band, köpfe)
switch (gelesene Z, Zustand)
  update Zustand()
  update Band()
  update Köpfe()
 } if (Zustand == END) { END)
loop

Blatt 3 - Aufgabe 15

1 Eingabe M (2 Band)
 Syntaxcheck(M) $\xrightarrow{\text{nein}}$ stop
 Schreibe Iterationslimit auf Band 2
 Loop i limit. 0
 4.1 Schreibe i-tes Wort auf Band 1
 4.2 Führe limit Iterationen aus von M
 4.3 Versuche falls M hält
 4.4. 2 Merker → halte akzep.
 4.5 limit *= 2

⇒ L r.a. □.
(+ Begründung + Erklärungen!)

a)

b)
z.z. L nicht entscheidbar
Annahme: L entscheidbar

$FM_{<M>w}$
  Eingabex = args
  Starte M mit w
  halte

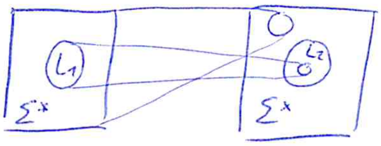
→ <M>w
Baue $FM_{<M>w}$
L Entscheider

2 Fälle:
1. $FM_{<M>w} \in L \Rightarrow <M>w \in H$
2. $\notin L \Rightarrow <M>w \notin H$

würden das Halteproblem entscheiden ⚡

B+S Übung

Reduktion: $x \in L_1 \rightarrow f(x) \in L_2$



neg →
pos ⇐

$x \in H \Leftrightarrow f(x) \in L$

$H \leq L$

① → $f(x) = \begin{cases} \langle FM_{\langle M \rangle w} \rangle & x = \langle M \rangle w \\ 0 & \text{sonst} \end{cases}$

F(x) ist total, weil sonst-Fall
" " berechenbar, weil Syntaxanalyse

② $FM_{\langle M \rangle w}$
- Eingabe y
- Starte M mit w
- halte

③ $x \in H \Rightarrow x = \langle M \rangle w$ und M mit w hält
$\Rightarrow f(x) = \langle FM_{\langle M \rangle w} \rangle$ und M mit w hält
$\Rightarrow FM_{\langle M \rangle w}$ hält immer $\Rightarrow f(x) \in L$

$x \notin H$ 1. $x \neq \langle M \rangle w \Rightarrow f(x) = 0 \notin L$
2. $x = \langle M \rangle w$ und M mit w hält nicht
$\Rightarrow f(x) = \langle FM_{\langle M \rangle w} \rangle$
$\Rightarrow FM_{\langle M \rangle w}$ hält nie $\rightarrow f(x) \notin L$

~~$\Rightarrow FM_{\langle M \rangle w}$ hält nie~~

$\Rightarrow H \leq L$

Blatt 4 - Aufgabe 19

$M_{\langle M \rangle w}$
Eingabe x
Falls x = 101010 halte
Starte M mit w
halte

$f(x) = \begin{cases} \langle FM_{\langle M \rangle w} \rangle\ 101010 & x = \langle M \rangle w \\ \langle M_L \rangle\ 10 & \text{sonst} \end{cases}$

M_L halte nur für 101010

$x \neq \langle M \rangle w \rightarrow x \in \overline{H} \rightarrow f(x) = \langle M_L \rangle \in L$

$x = \langle M \rangle w$

$x \in H$ M mit w gest. hält
$\Rightarrow f(x) = \langle FM_{\langle M \rangle w} \rangle\ 101010$
$\Rightarrow FM_{\langle M \rangle w}$ hält immer
$\Rightarrow f(x) \notin L$

$x \notin H$ M hält nicht
$\Rightarrow FM_{\langle M \rangle w}$ hält nur für 101010 $\Rightarrow f(x) \in L$

Blatt 5 - Aufgabe 24

1 Schreibe links # vor Eing.
2 Kopiere B1 auf B2
Starte $Entsch_{w\#w}$ auf B2
" $Entsch_{w\#w}$ akz
halte akz
Lösche B2
Falls # am Ende → verwerfe
Verschiebe # nach rechts, goto 2

$M = (Q, \Sigma, \Gamma, \delta, q_{copy}, F)$
$Q = \{ q_{copy}, q_{left1}, q_{left2}, q_{comp}, q_{af} \}$
$\Sigma = \{0, 1\}$
$\Gamma = \Sigma \cup \{B\}$
$F = \{q_{af}\}$
$\delta: Q \times \Gamma^2 \rightarrow P(\{R, N, L\}^2 \times Q \times \Gamma^2)$
$\delta_{alg}: Q \times \Gamma^2$

$\delta((q_0, (B,B))) = \{(q_F, (B,B), (N,N))\}$
$\forall z \in \Sigma: \delta((q_0, (z,B))) = \{(q_1, (z,B), (N,N))\}$
$\forall z \in \Sigma: \delta((q_1, (z,B))) = \{(q_1, (B,z), (R,R)), (q_2, (z,B), (N,L))\}$
$\forall z_1, z_2 \in \Sigma: \delta((q_2, (z_1, z_2))) = \{(q_2, (z_1, z_2), (N,L))\}$
$\forall z \in \Sigma: \delta((q_2, (z,B))) = \{(q_3, (z,B), (N,R))\}$
$\forall z \in \Sigma: \delta((q_3, (z,z))) = \{(q_3, (B,B), (R,R))\}$
$\delta((q_3, (B,B))) = \{(q_F, (B,B), (R,R))\}$

f(n) - Verifizierer M
- Eingabe x#w
- O(f(|x|))
- $x \in L \Leftrightarrow \exists w: w \leq f(|x|)$, M akz. x#w

Blatt 6 - A30

- sollte halten
- richtig :)
- falsch :(

TM M heißt $f(n)$- beschr. Verifizierer wenn
- Eingabe haben Form
- Laufzeit $O(f(|x|))$
- $\forall x \in \Sigma^*: x \in L \iff \exists w: |w| \leq f(|x|)$ und M akz. $x\#w$

$\boxed{30a}$ M_{IS}:

Eingabe $\underbrace{z}_{x}$

Falls $z \neq \langle G,k \rangle \# w$ halte verwerfend

Falls $|w| \neq k$ "

for u in w:

for v in w

falls (u,v) in E:

verwerfe

Halte akz.

$|w|$ Anz. Knoten

- Laufzeit: $O(|w|^2)$
- Eingaben $x\#w$ ✓ $|x| = |\langle G,k \rangle| \geq |w|$
- $O(|x|^2)$ ✓
- ~~$x \in IS$~~

- $\cancel{X} \in IS$

$\Rightarrow \cancel{X} = \langle G,k \rangle$, G hat unabh. Teilfolge U, $|U| = k$

$\Rightarrow \exists w$, nämlich genau U und M_{IS} akz. $x\#w$ per Konstruktion. $w \leq f(|x|)$, da $U \subseteq V \Rightarrow |U| \leq |\langle G,k \rangle|$

- $x \notin IS \Rightarrow x \neq \langle G,k \rangle \lor x = \langle G,k \rangle, |w| \neq k$ trivial.

$x = \langle G,k \rangle$ und es ex. keine unabh. Teilmenge U

$\Rightarrow \forall U \subseteq V: \exists u,v: (u,v) \in E \Rightarrow M_{IS}$ schlägt bei "if" fehl, M_{IS} akz. nicht ✓

□.

Blatt 7 - A35

a) $COL = \{\langle G,k \rangle | \ldots\}$

Eingabe λ

Falls $\lambda \neq \langle G,k \rangle \# w$ $w = bin(f(1)) \# bin(f(2)) \# \ldots$

Falls ein $f(i) < 1$ oder $> k$ verwerfe

for v in V:

for u in V:

falls $f(u) = f(v)$ && $(u,v) \in E$:

verwerfe

akz.

$x\#w$ ✓

$O(|x|^2)$ siehe

Blatt 8 - A 41

1. wahr

$L \in NP \rightarrow$ es gibt poly. Verif. V_L für L

$\Rightarrow$ Es gibt $f(n)$-beschr. Verif. V_L und $f(n)$ ist Polynom in n

$\Rightarrow$ TM M_L (M_L: Teste parallel mittels V_L alle Wörter bis Länge $f(n)$, ob sie Zertifikat für Eingabe von M_L sind) ist Entscheider für L

2. Satz VL (Simulation NTM auf DTM)

3. $L_1 \cap L_2$ ra, falls L_1, L_2 r.a.

TM M_1, M_2, welche L_1, L_2 akz. Damit kann TM M gebaut werden, welche erst M_1, dann M_2 mit der Eing. x startet. M hält, wenn $x \in L_1 \wedge x \in L_2$.

4. Falsch: f muss berechenbar sein

5. Wahr

$\exists$ U; ob Element in CLIQUE: ja -> Festes El. aus H
nein => " " " $\overline{H}$

H NP-schwer => $\forall L \in NP$ $L \leq_p H$ -> insb. CLIQUE

6. Falsch Ex. poly. Verif. für CLIQUE. Diese TM kann durch Registermaschine simuliert werden

7. Falsch: Es können $|\langle M \rangle|^2$ Schritte von M simuliert werden

Blatt 9 – A 45

$L = \{ w \mid w \in \{a,b,c\}^*, \#_a(w) = 2 \cdot \#_b(w) = \#_c(w) \}$

$L = L(G)$
$L \subseteq L(G)$
$L \supseteq L(G)$

$G = (V, \Sigma, P, S)$
$\Sigma = \{a,b,c\}$
$V = \{A, B, C, S\}$

1) $S \to \varepsilon$
2) $S \to AABCCS$
3) $A \to a$
4) $B \to b$
5) $C \to c$
6) $AB \to BA$
7) $BA \to AB$
8) $AC \to CA$
9) $CA \to AC$
10) $BC \to CB$
11) $CB \to BC$

„$\subseteq$" Sei $w \in L$
1) Starte mit S
2) Wende $\#_b(w)$ mal Regel 2) an und anschließend 1)
3) Vertausche mit 6) – 11) die Reihenfolge
4) Wende 3) – 5) an

„$\supseteq$" $\#_a(w) + \#_A(w) = 2(\#_b(w) + \#_B(w)) = \#_c(w) + \#_C(w)$

- Inv. gilt auch für nur das Startsymbol
- R1 entfernt nur S
- R2 generiert As, Bs, Cs, und zwar genau so, dass Inv. erfüllt bleibt
- R3 – R5 $\#_X \to \#_X - 1 + 1 \, \hat{\#}_x$ -> gl. Summe
- R6 – R11. Ändern nichts an Anzahl

z.z.: Aus Inv. folgt „$\supseteq$"
Wenn Inv. gilt, können 3–5 angewandt werden, wodurch sich die Anzahl $\#_X + \#_x$ nicht ändert -> Jedes erzeugte Wort ist gültig $\square$.

Blatt 10 – A 50

$G = (V, \Sigma, P, S)$, kontextfrei

a) 1) ~~Setze~~ $E_1 := \emptyset$
2) ~~Setze~~ $E_0 := \{ v \in V : \exists x \in \Sigma^* : (v \to x) \in P \}$
3) $i := 0$
4) Wiederhole solange $E_i \neq E_{i-1}$
4.1) $i := i+1$
4.2) Berechne $E_i := \{ v \in V : (\exists x \in (E_{i-1} \cup \Sigma)^* : (v \to x) \in P) \}$
5) Gebe $V \setminus E_i$ aus

Laufzeit $O(|V| \cdot |P| \cdot \ell)$ ℓ: Länge d. längsten Produktion

zz: A nützlich => A wird nicht ausgegeben

Bew: Sei A nützlich => Es gibt Folge von Produktionen $(u_1 \to v_1), \dots (u_k \to v_k)$, sodass
$A \xrightarrow{k} w \in \Sigma^*$ und w wird durch Produktionsfolge erzeugt

Im Algorithmus wird somit u_{k-i} spätestens in E_i aufgenommen. Also wird A nicht ausgegeben

| A nicht ausgegeben => A nützlich
| A " => A wird in E_i aufgenommen

Beim Einfügen kann man sich jew. das Wort merken, das erzeugt werden kann. Sei $W(v)$ dieses Wort, initial für alle Variablen undef. und $W(v)$ wird beim erste Einfügen in E_i bestimmt. Beim Einfügen einer Var. v in E_i sind alle $W(\tilde{v})$ $\tilde{v} \in E_{i-1}$ bekannt. Bei der Produktion $u \to v$, die zum Einfügen von v führt, kann deshalb durch weitere Produktionen je nach Variable $\tilde{v}$ in U das Wort $W(\tilde{v})$ erzeugt werden.

Das so entstandene Wort kann aus v erzeugt werden, ist in Σ^* enthalten und kann als $W(v)$ benutzt werden. Insgesamt: $w(A) \in \Sigma^*$ und $A \xrightarrow{*} w(A)$

i) $L(G) = \phi \iff S$ nutzlos

Blatt 11 - 55

) $L = \{a^i b^j \mid 4 < i < j\}$
n_L: 13 (sonst Problem bei Fall 2
Sei $z \in L$ mit $|z| \geq n_L$
~~Sei o.B.d.A.~~ Fall 1: $i+1 < j$ $z = a^i b^j$. Wähle $u = a^i$, $v = \varepsilon$, $w = \varepsilon$, $x = b$, $y = b^{j-1}$
$|vwx| \leq n_L$
$vx \neq \varepsilon$
$\forall k \in \mathbb{N}_0$: $uv^k w x^k y \in L$, denn $a^i \varepsilon^k b^k b^{j-1}$, $i > 4$ und $i < j-1+k$, da $i < j-1$
Fall 2: $i+1 = j$ $z = a^i b^j$. Wähle $u = a^{i-1}$ $v = a$, $w = \varepsilon$, $x = b$, $y = b^{j-1}$
$|vwx| \leq n_L$, $vx \neq \varepsilon$ ✓
$\forall k \in \mathbb{N}_0$ $uv^k wx^k y \in L$, denn $a^{i-1} a^k b^k b^{j-1}$ ist für $k=0$ $a^{i-1} b^{j-1} \in L$ und $k > 1$ $a^{i-1} a^k b^k b^{j-1} \in L$, da sich die Anzahl der a's und b's gleich erhöht

) $L = \{a^i b^j c^k \mid 0 \leq i < j < k\}$
Sei $n_L \in \mathbb{N}$, wähle $z = a^{n_L+1} b^{n_L+2} c^{n_L+3}$ $|z| \geq n_L$
Sei $uvwxy = z$ mit $|vwx| \leq n_L$ und $vx \neq \varepsilon$
Fall 1: vx enthält mind 1a, weil $|vwx| \leq n_L$ gilt, dass kein c in vx ist
Sei $i = 42$ => $u v^{42} w x^{42} y$ erhöht die Anzahl der a's um mind 41 => $\#a > \#c$
=> $z \notin L$
Fall 2: vx nur b's und c's
wegen 1 enthält vx keine a's.
Wähle $i=0$ $u v^0 w x^0 y$ reduziert die Anzahl d. b's um mind. 1, die der a's nicht -> $z \notin L$
Fall 3: vx enthält nur c's
Wähle $i=0$ $u v^0 w x^0 y$ reduziert die Anzahl d. c's um min. 1
=> $\#(b) \geq \#(c)$ => $z \notin L$

NPDA = $(Q_L \ldots)$

DFA = $(Q_R \ldots)$

Neu: $Q_L \times Q_R$ $\quad q_0 = (q_{0L}, q_{R0})$ $\quad F = F_L \times F_R$

$\delta_L : Q_L \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to Q_L \times \Gamma^*$

$\delta_R : Q_R \times \Sigma \to Q_R$

$\forall q_L \in Q_L \; \forall q_R \in Q_R \; \forall \sigma \in \Sigma, \forall \gamma \in (\Gamma \cup \{\varepsilon\})$:

falls $\delta_L(q_L, \sigma, \gamma) = (\tilde{q}_L, \tilde{\gamma})$ und $\delta_R(q_R, \sigma) = \tilde{q}_R$, dann sei

$\delta((q_L, q_R), \sigma, \gamma) = ((\tilde{q}_L, \tilde{q}_R), \tilde{\gamma})$,

$\forall q_L \in Q_L \; \forall q_R \in Q_R \; \forall \gamma \in (\Gamma \cup \{\varepsilon\})$:

falls $\delta_L(q_L, \varepsilon, \gamma) = (\tilde{q}_L, \tilde{\gamma})$ dann sei $\delta((q_L, q_R), \varepsilon, \gamma) = ((\tilde{q}_L, q_R), \tilde{\gamma})$

$L_2 = \{a^k b^{k+l} c^l \mid l, k \in \mathbb{N}, l, k \geq 0\}$

Sei $n_L \in \mathbb{N}$ Setze $a^{n_L} b^{2n_L+1} c^{n_L+1}$

Sei $uvwxy = z$ mit $vx \neq \varepsilon$ und $|vwx| \leq 1$

Fall 1: mind 1 c in vwx Setze $i = 0$, nun ist $\#(a) \geq \#c$ $\quad uv^0wx^0y \notin L$

Fall 2: mind 1 a in vwx Setze $i = 2$ $\quad uv^2wx^2y \notin L$, da die Anzahl der a's um mind 1 erhöht wird und somit größer gleich der Anzahl der c's ist

Fall 3: nur b's in vwx

Setze $i = 0$: $uv^0wx^0y = a^{n_L} b^{2n_L+1-|vx|} c^{n_L+1}$, da $|vx| \geq 1$

ist $n_L + n_L + 1 \neq 2n_L + 1 - |vx|$ $\Rightarrow$ $uwy \notin L$

$L = \{\langle M \rangle w \mid M$ ist det 1-Band-TM für die mind eine Eingabe y, $|y| > 7$, $|y|$ prim ex., sodass M hält

$$f(x) = \begin{cases} \langle FM_{\langle M \rangle w} \rangle 42 & \text{falls } x = \langle M \rangle w \\ 0 & \text{sonst} \end{cases}$$

$FM_{\langle M \rangle w}$:

Eingabe y

Starte M mit w

Falls $|y| = 11$ halte,

sonst Endlosschleife

$x \in H$ $\wedge$ $x = \langle M \rangle w$ M gest. mit w hält

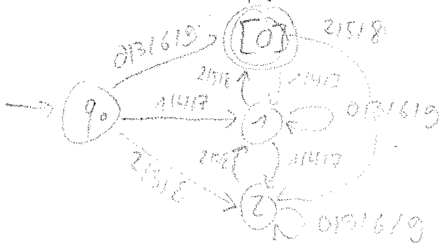
$\Rightarrow f(x) = \langle FM_{\langle M \rangle w} \rangle 42$ und $FM_{\langle M \rangle w}$ hält nur für y mit $|y| = 11$ $\Rightarrow f(x) \in L$

$x \notin H$ $\wedge$ $x \neq \langle M \rangle w$ $\Rightarrow$ $f(x) = 0 \notin L$

$x \notin H$ $\wedge$ $x = \langle M \rangle w$ M gest. mit w hält nie

$\Rightarrow f(x) = \langle FM_{\langle M \rangle w} \rangle 42$, und $FM_{\langle M \rangle w}$ hält für keine Eingabe

$\Rightarrow f(x) \notin L$ $\quad \Rightarrow$ $H \leq L$



3,5/4

/61

a)

V = ~~{S}~~ {S, A, B, C}

P:

S → aBC | bBBCC | aACCA

A → a

B → b | aAB

C → c

NPDA M = (Q, Σ, Γ, δ, Z_0, F, q_0) mit:

Q = {q_0, q_1, q_a}

Σ = {a, b, c}

Γ = {Z_0} ∪ {S} ∪ Σ ✓V

Z_0 = Z_0

F = {q_a}

q_0 = q_0

δ:

(q0 = q_0, q1 = q_1, qa = q_a, Z0 = Z_0)

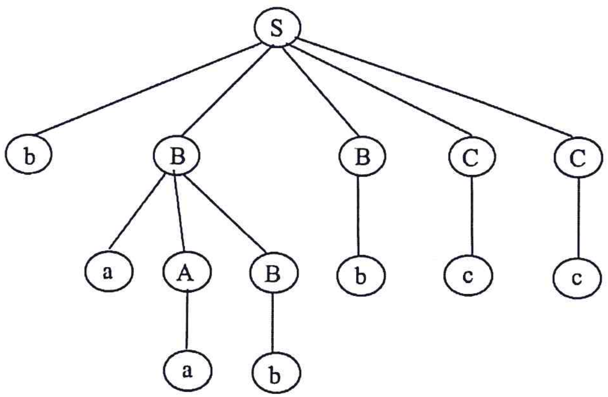
$\delta(q_0, \varepsilon, Z_0) = \left\{\left(q1, \binom{S}{Z0}\right)\right\}$

$\forall A \in V: \delta(q_1, \varepsilon, A) = \left\{\left(q1, \begin{pmatrix} w1 \\ \dots \\ wk \end{pmatrix}\right) \middle| (A \to w1, \dots, wk) \in P\right\}$

$\forall a \in \Sigma: \delta(q_1, a, a) = \{(q1, \varepsilon)\}$

$\delta(q_1, \varepsilon, Z_0) = \{(qa, Z0)\}$

b)



✓

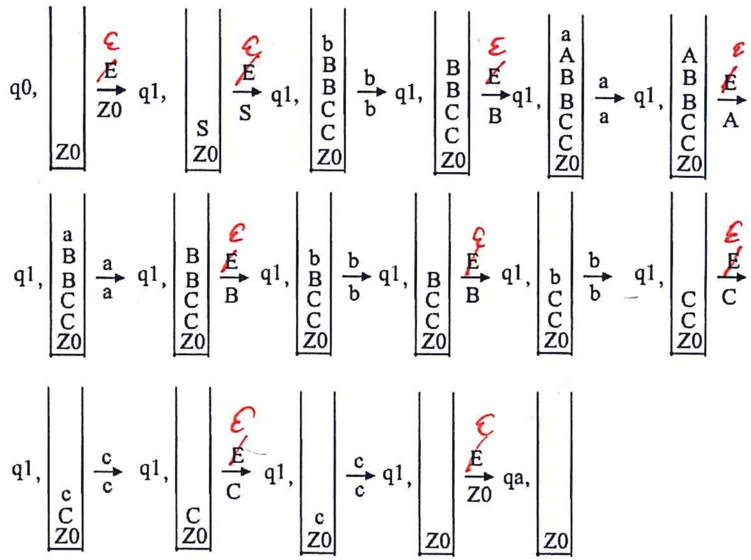
c)

Beschreibung wie M eine akzeptierende Rechnung für w = baabbcc durchführt:

Zu lesen als:

Zustand, Kellerinhalt $\xrightarrow[\text{von Keller gelesene(s)Zeichen}]{\text{von Eingabe gelesenes Zeichen}}$ neuer Zustand, neuer Kellerinhalt

E = ε



Abgabe BFS

Eva Dengler ✓

Johannes Zink ✓

Jonas Dürr ✓

Lorenz Kästle ✓

Fr. 14^{15} - 15^{45}

Jonas Schreiner

61	62	63	64	65	66
3,5	0	0	3	1	3,5

A 62 $\Sigma = \{a, b, c\}$

zZ: $\Sigma^* \setminus a^n b^n c^n = \{a^p b^q c^r \mid p \neq q$ oder $q \neq r\}$

$\cup \{ \{a,b,c\}^* \{ba, ca, cb\} \{a,b,c\}^* \} \cup \{\varepsilon\}$

$L_1 = \{a,b,c\}^* \{ba, ca, cb\} \{a,b,c\}^*$ ist regulär, da wir eine reguläre Grammatik für L_1 angeben können:

$G = ?$

~~$S \to \varepsilon \mid A \mid aA \mid bA \mid cA \mid B$~~

~~$A \to aA \mid bA \mid cA \mid aB \mid$~~

Die Grammatik erzeugt $\{\varepsilon\} \cup L_1$

$S \to \varepsilon \mid bX \mid cY \mid aK \mid bK \mid cK \mid aA \mid bA \mid cA$

$K \to aK \mid bK \mid cK \mid aA \mid bA \mid cA$

$A \to bX \mid cY$

$X \to aH$

$Y \to aH \mid bH$

$H \to \varepsilon \mid a \mid b \mid c \mid aH \mid bH \mid cH$

Fehlt Begründung der Korrektheit

Jede reg. Sprache ist auch kf.

$\to L_1$ kf.

$\to$ irrelevant - hab mir leichtere Sprachen gesucht

$\Sigma^* \setminus a^n b^n c^n = \underbrace{\{a^x b^y c^z \mid x \neq y\}}_{L_1} \cup \underbrace{\{a^x b^y c^z \mid y \neq z\}}_{L_2}$ Nein! da: $\neg(\#_a = \#_b = \#_c)$

$\Leftrightarrow \neg((\#_a = \#_b) \wedge (\#_b = \#_c))$

$\Leftrightarrow (\#_a \neq \#_b) \vee (\#_b \neq \#_c)$

Grammatiken: das würde weiter die Form $a^0 b^0 c^n$ erzeugen $\notin L_1$

L_1: $S \to \cancel{\varepsilon} \mid X \mid Y \mid Z \mid XZ \mid YZ$

$X \to a \mid aX \mid aW$ (Fall $\#_a > \#_b$)

$Y \to b \mid Yb \mid Wb$ (Fall $\#_a < \#_b$)

$W \to ab \mid aWb$ (gleich viele as und bs erzeugen)

$Z \to c \mid cZ$ (erzeuge bel. viele cs)

$G_1 = ?$

Begründung der Korrektheit

L_2: $S \to \cancel{\varepsilon} \mid X \mid Y \mid Z \mid ZX \mid ZY$ s.o.

$X \to b \mid bX \mid bW$ (Fall $\#_b > \#_c$)

$Y \to c \mid Yc \mid Wc$ (Fall $\#_b < \#_c$)

$W \to bc \mid bWc$ (gleich viele bs und cs erzeugen)

$Z \to a \mid aZ$ (erzeuge bel. viele as)

$G_2 = ?$

s.o.

Da wir für L_1 und L_2 eine kf. Grammatik angeben können, sind L_1 und L_2 kontextfreie Sprachen.

Nach Satz 3.37 sind die kf. Sprachen unter $\cup$ abgeschlossen

$\Rightarrow L_3 = L_1 \cup L_2$ ist kontextfrei

Da $L_3 = \{a,b,c\}^* \setminus L$ ist die Aussage bewiesen.

0/4

A 63 L kontextfrei und R regulär $\Rightarrow L \cap R$ kontextfrei.

L ist kontextfreie Sprache $\rightarrow$ es gibt Kellerautomaten, sodass L akzeptiert wird (siehe Hinweis) $\rightarrow$ gibt NPDA

R ist reguläre Sprache $\rightarrow$ es gibt DFA, der R akzeptiert.

~~Zu jedem~~ Zu jedem DFA kann ich einen PDA bauen – jede Übergang schreibt nichts auf den Stack und nimmt nichts herunter. Da jede DFA auch ein NFA ist, gibt es einen NPDA, der R akzeptiert.

$\Rightarrow$ es gibt also zu L und zu R einen NPDA.

Nun kann ich einen NPDA konstruieren, der alle Wörter aus $L \cap R$ akzeptiert, indem mein neuer NPDA zunächst überprüft, ob $w \in L \cap R$ von $NPDA_L$ ~~überprüf~~ akz. wird, und dann überprüft, ob das $w \in L \cap R$ auch von $NPDA_R$ akzeptiert wird.

$\Rightarrow$ es gibt NPDA, der $L \cap R$ akzeptiert
$\Rightarrow$ (Hinweis) / (Satz 3.36) $L \cap R$ kontextfrei. □.

dazu muss dann aber w zweimal gelesen werden und da ihr würdet damit auch $\cap$-Abgeschlossenheit von kf. Sprachen begründen

A 64 1/4

$L = \{a,b,c\}^* \cup \{☺^i a^n b^n c^n \mid i \geq 1, n \geq 1\}$

(a) L besitzt die kf PE.

$\exists n_L \in \mathbb{N} \ \forall z \in L, |z| \geq n_L \ \exists u,v,w,x,y \in \Sigma^*, z = uvwxy$
(i) $|vx| \geq 1$ (ii) $|vwx| \leq n_L$ (iii) $\forall i \geq 0 : uv^i w x^i y \in L$

$\{a,b,c\}^*$ ist reguläre Sprache, also auch kf Sprache $\Rightarrow$ besitzt kf PE
nämlich: $S \rightarrow A \mid \varepsilon$, $A \rightarrow a \mid b \mid c \mid aA \mid bA \mid cA$

hier muss die Fallunterscheidung gemacht werden und mindestens noch begründet werden dass $n \geq 4$ groß genug ist

für $\{☺^i a^n b^n c^n \mid i \geq 1, n \geq 1\}$:
wähle $n_L = 4$, und sei $z \in L$ mit $|z| \geq n_L$ beliebig, aber fest.
Wähle $u = ☺^{i-1}$, $v = ☺$, $w = \varepsilon$, $x = \varepsilon$, $y = a^n b^n c^n$
(i) erfüllt, (ii) ebenfalls.
Smiley!

das ist nicht in diese Sprache

(iii) auch erfüllt: für $k \geq 1$ gilt $uv^k w x^k y = ☺^{i-1} ☺^k a^n b^n c^n \in \{☺^i a^n b^n c^n \mid i \geq 1, n \geq 1\}$
für $k = 0$ gilt $uv^0 w x^0 y = ☺^{i-1} a^n b^n c^n$.
mit $i = 1$ gilt $a^n b^n c^n \in \{a,b,c\}^*$
mit $i > 1$ gilt $☺^x a^n b^n c^n \in \{☺^i a^n b^n c^n \mid i \geq 1, n \geq 1\}$

$\Rightarrow uv^i w x^i y$ ist $\forall i \geq 0 \in L$

$\Rightarrow L$ hat kf Pumpeigenschaft.

b) Widerspruchsbeweis: Angenommen, L sei kontextfrei. Dann gilt nach

A 63: L kf. und R reg $\Rightarrow L \cap R$ kf.

Sei L nun, ~~dass~~ $L = \{a,b,c\}^* \cup \{☺^i a^n b^n c^n \mid i \geq 1, n \geq 1\}$

und $R = ☺ \{☺, a, b, c\}^*$

Dann ist $L \cap R = \{☺^i a^n b^n c^n \mid i \geq 1, n \geq 1\}$

zz $L \cap R = \{ ☺^i a^n b^n c^n \mid i \geq 1, n \geq 1 \}$ ist nicht kontextfrei
=> also: $L \cap R$ hat die kf. PE nicht -> $L \cap R$ nicht ~~fda~~ kf.

zz: $\forall n_L \in \mathbb{N} \; \exists z \in L \cap R, |z| \geq n_L \; \forall u, v, w, x, y \in \Sigma, z = uvwxy$
(i) $|v\cancel{x}| \geq 1$ und (ii) $|vwx| \leq n_L$ -> „$\neg$(iii)" $\exists i \geq 0: uv^iwx^iy \notin L \cap R$

Sei $n_L \in \mathbb{N}$ beliebig, aber fest.
Wähle $z \in L \cap R$ mit $z = ☺^1 a^{n_L} b^{n_L} c^{n_L}$

Für eine beliebige, aber feste Aufteilung von $z = uvwxy$
gilt nun:
da $|vwx| \leq n_L$ gilt nun:

1. v, x enthält nur ☺
2. v, x enthält nur ☺ und a
3. v, x " nur a
4. v, x " " a und b
5. v, x " " b
6. v, x " " b und c
7. v, x " " c

1. und 2. setze $i = 0$ => keine ☺ im Wort => ~~[illegible]~~
$w \notin \{ ☺^i a^n b^n c^n \mid i \geq 1, n \geq 1 \}$ da $i = 0$

3., 5., 7. setze $i = 0$ => weniger as / bs / cs als bs und cs / as und cs / as und bs
=> $w \notin \{ ☺^i a^n b^n c^n \mid i \geq 1, n \geq 1 \}$

4., 6. setze $i = 0$ => weniger as und bs / bs und cs als cs / as
=> $w \notin \{ ☺^i a^n b^n c^n \mid i \geq 1, n \geq 1 \}$

=> $L \cap R$ besitzt die kf PE nicht
=> $L \cap R$ nicht kf

=> aus Annahme L kf folgt mit
L kf und R reg => ~~[illegible]~~ $L \cap R$ kf ein Widerspruch

=> Annahme falsch

=> L nicht kontextfrei, obwohl sie die kf PE besitzt

1/4/

/65

Pump-Eigenschaft:

$\exists n_L \in N\ \forall z \in L\ |z| \geq n_L\ \exists u,v,w \in \Sigma^*$: $uvw = z$ und:

(i) $|uv| \leq n_L$
(ii) $v \neq \varepsilon$
(iii) $\forall i \geq 0$: $uv^i w \in L$

=> besitzt Pump-Eigenschaft nicht, wenn:

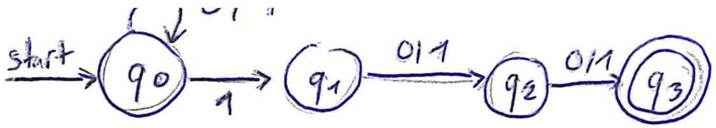
$\forall n_L \in N\ \exists z \in L\ |z| \geq n_L\ \forall u,v,w \in \Sigma^*$: $uvw = z$ und:

(i) $|uv| \leq n_L$
(ii) $v \neq \varepsilon$
Und aus (i) und (ii) folgt (iii):
(iii) $\exists i \geq 0$: $uv^i w \notin L$

Sei $n_L \in N$ beliebig aber fest
Setze $y = 0^{n_L} 1^{n_L}$ ✓
Sei $uvw = y$ mit $|uv| \leq n_L$ und $v \neq \varepsilon$. beliebig aber fest
Wegen (i) $|uv| \leq n_L$ besteht v somit nur aus Nullen.
Sei $i = 0$:
$uw = 0^{n_L - |v|} 1^{n_L}$
Es gilt:
$|v| \geq 1$
=> $uv^0 w = uw \notin L$ das muss man schon beweisen!
$z \in L$ beliebig

=> besitzt Pump-Eigenschaft **nicht**.

66| geg:

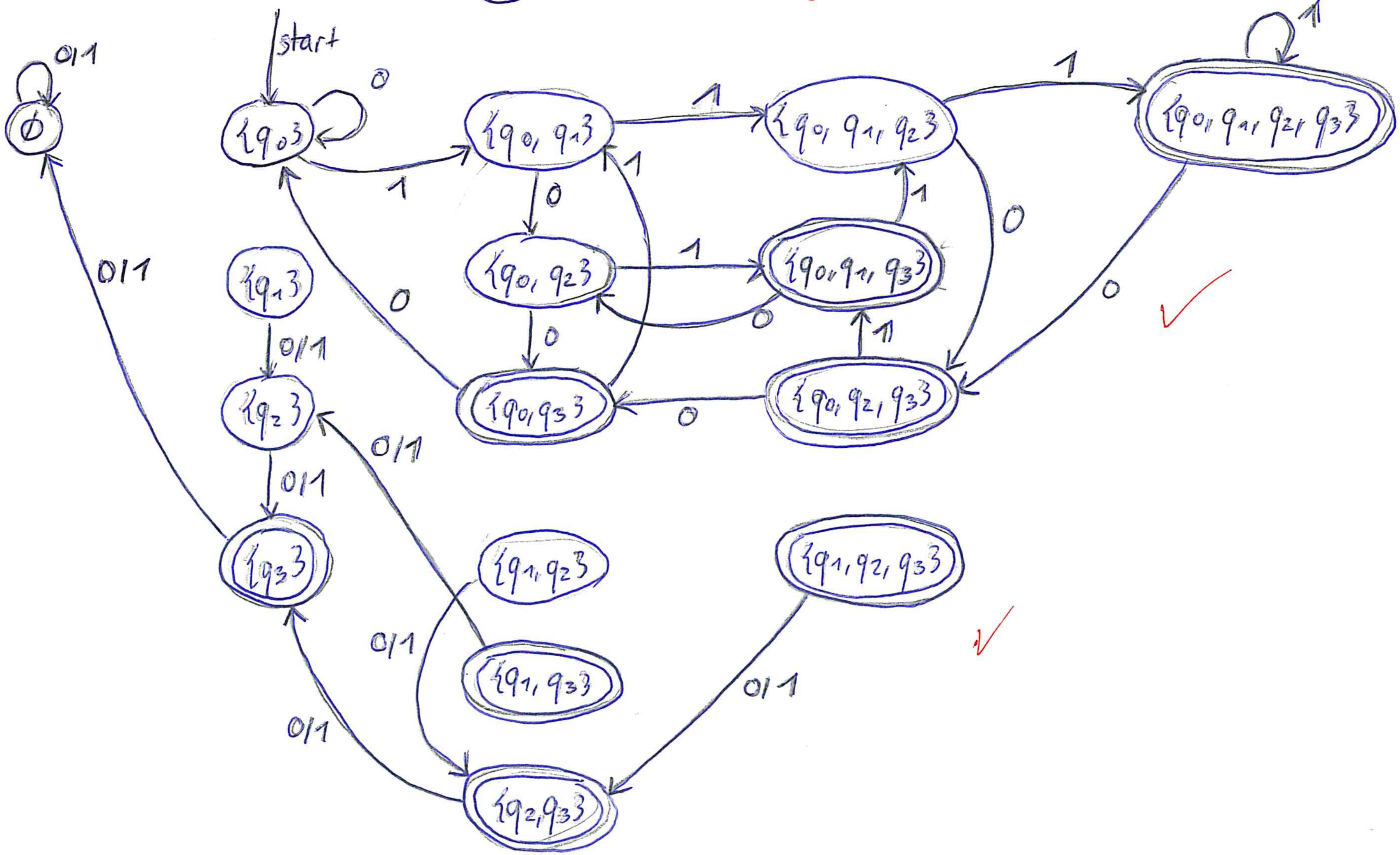


a) $L = \{w \mid w \in \{0,1\}^*$, die drittletzte Stelle ist eine $1\}$ ✓

+ Begründung

b)

3,5/4



✓

✓

BFS - Abgabe

Eva Dengler VEST
~~Jonas Schreiner~~
Jonas Dürr VEST
Johannes Zink VEST
Lorenz Kästle VEST

Abgabe Fr. $14^{15} - 15^{45}$

Jonas Schreiner

56	57	58	59
3	2,5	0,5	0

$\boxed{A58}$

a) $L_1 = \{w \mid w \in \{a,b,c\}^*, \#_a(w) = \#_b(w) = \#_c(w)\}$

Sei n_{L_1} beliebig, aber fest. Sei u,v,w,x,y bel. aber fest mit $z = uvwxy$, $|vx| \geq 1$ und $|vwx| \leq n_{L_1}$

(Nun $\rightarrow \exists i \geq 0: uv^iwx^iy \notin L$) also muss v,x nicht nur ein Zeichen enthalten!

(A) Angenommen, es gäbe ein i, sodass $z' = uv^iwx^iy \in L$, also $\#_a(z') = \#_b(z') = \#_c(z')$

Betrachten wir nun $uv^{i+1}wx^{i+1}y$. Nach Voraussetzung ist $|vx| \geq 1$, also ~~entweder~~ nur $v = \varepsilon$ oder nur $x = \varepsilon$ oder beide $\neq \varepsilon$

x=abc funktioniert ↯

1. $v = \varepsilon$, $x \in \{a,b,c\}^+$ oder 2. $x = \varepsilon$, $v \in \{a,b,c\}^+$

für $uv^{i+1}wx^{i+1}y = z''$ gilt nun
$\#_a(z'') - 1 = \#_b(z'') = \#_c(z'')$ oder
$\#_a(z'') = \#_b(z'') - 1 = \#_c(z'')$ oder
$\#_a(z'') = \#_b(z'') = \#_c(z'') - 1$

da wir nun einmal x / v mehr haben $\rightarrow z'' \notin L \rightarrow \exists i \geq 0: uv^iwx^iy \notin L$

2. $x, v \in \{a,b,c\}$

für $uv^{i+1}wx^{i+1}y = z'''$ gilt nun: ~~$\#_a(z'') + 1 = \#_b(z'')$~~

1. es gibt 2 a mehr / 2 b mehr / 2 c mehr
2. es gibt 1 a und 1 b / 1 b und 1 c / 1 c und 1 a mehr

$\Rightarrow \#_a(z'') = \#_b(z'') = \#_c(z'')$ ist nicht erfüllt!

$\rightarrow \exists i \geq 0: uv^iwx^iy \notin L$

(B) Angenommen, es gäbe kein i, sodass $uv^iwx^iy \in L$, dann gilt auch $\exists i \geq 0: uv^iwx^iy \notin L$

$\Rightarrow L_1$ besitzt die kontextfreie Pumpeigenschaft nicht.

/58

b)

Sei n_L beliebig, aber fest.

Setze $z = a^{n_L-1}c^{n_L+1}b^{n_L} \in L$ BdA $n_L \geq 2$?

Seien $u,v,w,x,y \in \{a, b, c\}^*$ beliebig, aber fest mit $uvwxy = z = a^{n_L-1}c^{n_L+1}b^{n_L}$

Zudem soll gelten: $|vwx| \leq n_L$ und $vx \neq \varepsilon$

Wähle: $u = a^{n_L-1}c^{n_L+1}$, $v = \varepsilon$, $w = \varepsilon$, $x = b^{n_L}$, $y = \varepsilon$

=> $|vwx| = |x| = n_L \leq n_L$ und $vx \neq \varepsilon$

Wähle $i = 2$ mit $n_L = 1$:

=> $uv^iwx^iy = cbb$ nicht $\in L$

=> nicht kontextfrei

u, v, w, x, y sind beliebig dürfen also nicht gewählt werden

c)

Sei n_L beliebig, aber fest.

Setze $z = I^p$ mit p ist eine Primzahl $\cancel{\in L}$ $p \geq n_L$

Seien $u, v, w, x, y \in \{ I \}^*$ beliebig, aber fest mit $uvwxy\cancel{z} = I^p$ mit p ist eine Primzahl

Zudem soll gelten: $|vwx| \leq n_L$ und $vx \neq \varepsilon$

Wähle $u = \varepsilon$, $v = \varepsilon$, $w = \varepsilon$, $x = I$, $y = I^{p-1}$

=> $|vwx| = |x| = 1 \leq n_L$ und $vx \neq \varepsilon$

Fall 1: $p > 2$

Wähle $i = 2$:

=> $uv^iwx^iy = I^{p+1}$ nicht $\in L$ (da Anzahl der I somit gerade und somit keine Primzahl)

Fall 2: $p = 2$

Wähle $i = 3$

=> $uv^iwx^iy = I^{p+2} = IIII$ nicht $\in L$ (da Anzahl der I gerade und somit ebenfalls keine Primzahl)

=> nicht kontextfrei

0,5/6

56. Aufgabe 56

Idee: Wenn ein Produktion ε aus einer Variablen erstellt, wird diese Produktion entfernt und es wird eine zusätzliche Produktion eingeführt, für Produktionen, die diese Variable erstellen, in der die Variable nicht erstellt wird (ausgenommen für S).

Variablen, die ε erstellen: $A \to aBB \mid \varepsilon$ und $B \to AS \mid b \mid \varepsilon$

$B' \to A'S' \mid b$
$A' \to aB'B' \mid aB' \mid a$
$S' \to A'B'A'c \mid A'B'c \mid A'A'c \mid B'A'c \mid A'c \mid B'c \mid c$

Damit ist die Gramatik: $G' = (V', \Sigma, P', S')$ mit $V = \{S', A', B'\}$ $\Sigma = \{a, b, c\}$ und den Produktionen:
$B' \to A'S' \mid b \mid S'$
$A' \to aB'B' \mid aB' \mid a$
$S' \to A'B'A'c \mid A'B'c \mid A'A'c \mid B'A'c \mid A'c \mid B'c \mid c$

3/4

– ich mach BFS Hausis!

A 57

a)

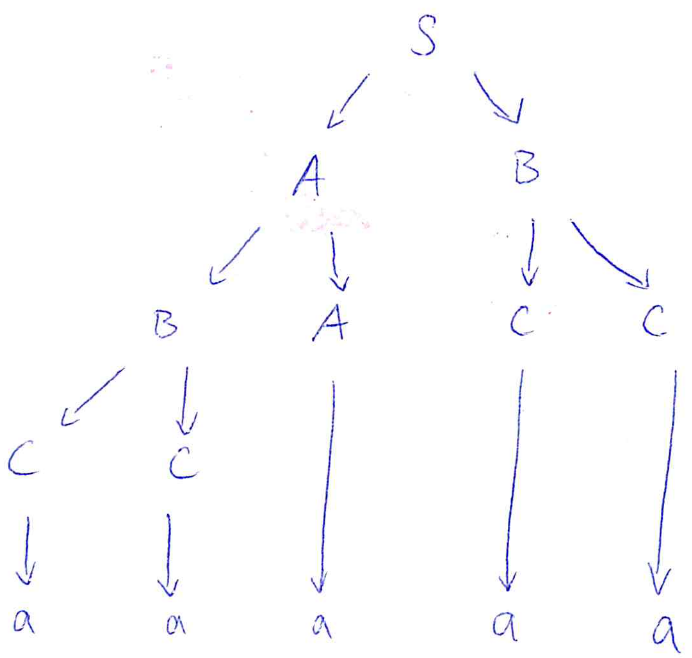
	a	a	a	a	a
0	{A,C}	{A,C}	{A,C}	{A,C}	{A,C}
1	{B}	{B}	{B}	{B}	
2	{S,A,~~B~~,C}	{S,A,~~B~~,C}	{S,A,~~B~~,C}		
3	{~~S~~,~~A~~,B,~~C~~}	{~~S~~,~~A~~,B,~~C~~}			
4	{S,A,~~B~~,C}				

✓ aaaaa ∈ L(G), weil ~~aaaaa~~ S ∈ V(1,4)

	b	a	a	b	a
0	{B}	{A,C}	{A,C}	{B}	{A,C}
1	{S,A}	{B}	{S,C}	{S,A}	
2	{~~S~~,~~A~~,~~C~~}	{~~S~~,~~A~~,B}	~~{S, C}~~ B		
3	{~~S~~,~~A~~,B,~~C~~}	{~~S~~,~~A~~,B} S,A,C			
4	{S,A,~~B~~,C}				

baaba ∈ L(G), weil S ∈ V(1,4)

b)



✓

2,5/4

50022045/6.99