Berechenbarkeit und formale Sprachen, Algorithmen und Komplexität – Prof. Dr. Johannes Blömer, Zusammenfassung von Florian Schoppmann

Zusammenfassung BuFS/AuK

1. Semester

Wichtige Sprachen

Halteproblem H := $\{\langle M\rangle x \mid M$ ist DTM, die gestartet mit Eingabe x hält.$\}$

Gödel := $\{w \in \{0,1\}^* \mid w$ ist die Gödelnummer einer DTM M.$\}$

Useful := $\{(\langle M\rangle, q) \mid M$ ist DTM mit Zustand q, und es gibt eine Eingabe w, so dass M gestartet mit w den Zustand q erreicht.$\}$

Diag := $\{w \in \{0,1\}^* \mid$ Die DTM M_i akzeptiert w_i nicht, wobei $w = w_i\}$

Akzeptanzproblem A := $\{\langle M\rangle x \mid M$ ist DTM, die die Eingabe x akzeptiert.$\}$

2. Berechenbarkeit

1) **Definition:** Turingmaschine (DTM)
   akzeptierte Sprache einer DTM M: Menge aller von M akzeptierten Wörter
   Eine DTM entscheidet eine Sprache, wenn sie sie akzeptiert und bei jeder Eingabe anhält.
   Eine Sprache heißt rekursiv aufzählbar, wenn es eine DTM gibt, die sie akzeptiert.
   Eine Sprache heißt rekursiv oder entscheidbar, wenn es eine DTM gibt, die sie entscheidet.
3) **Definition:** Charakteristische Funktion χ_L einer Sprache L ordnet einem $x \in \Sigma^*$ 1 zu, wenn $x \in L$, 0 sonst.
   Church'sche These: Die im intuitiven Sinne berechenbaren Funktionen sind genau die, die durch Turingmaschinen berechenbar sind.
4) **Definition:** Gödelnummer ist Codierung einer Turinmaschine in Binärcode:
   $X_1 \mathrel{\widehat{=}} 0,\ X_2 \mathrel{\widehat{=}} 1,\ X_3 \mathrel{\widehat{=}} \sqcup,\ X_4 \mathrel{\widehat{=}} \triangleright,\ D_1 \mathrel{\widehat{=}} L,\ D_2 \mathrel{\widehat{=}} R$, dann ist die Codierung eines Eintrags der Funktionstabelle $\delta(q_i, X_j) = (q_k, X_l, D_m)$: $0^{i+1}10^j10^{k+1}10^l10^m$.
   Universelle Turingmaschine: Simuliert jede Turingmaschine M bei jeder Eingabe $x \in \{0,1\}^*$
5) Diag ist nicht rekursiv aufzählbar.
6) **Satz:** L_1, L_2 seien rekursive (= entscheidbare) Sprachen. Dann gilt: $\overline{L_1},\ L_1 \cap L_2,\ L_1 \cup L_2$ sind rekursiv.
   Satz: L_1, L_2 seien rekursiv aufzählbare Sprachen. Dann gilt: $L_1 \cap L_2,\ L_1 \cup L_2$ sind rekursiv aufzählbar.
   Satz: L rekursiv $\Leftrightarrow$ L und $\overline{L}$ rekursiv aufzählbar.
7) **Definition:** $L' \subseteq \{0,1\}^*$ heißt reduzierbar auf $L \subseteq \{0,1\}^*$ ($L' \leq L$), falls es eine Funktion $f : \{0,1\}^* \to \{0,1\}^*$ gibt mit:
   1. $\forall w \in \{0,1\}^* : w \in L' \Leftrightarrow f(w) \in L$
   2. f ist berechenbar.
   Lemma: $L', L \subseteq \{0,1\}^*$ seien Sprachen mit $L' \leq_f L$. Es gilt: L entscheidbar $\Rightarrow L'$ entscheidbar, L rekursiv aufzählbar $\Rightarrow L'$ rekursiv aufzählbar
8) **Satz:** Die Sprache $\overline{H}$ ist nicht rekursiv aufzählbar.
   Satz: Die Sprachen A, Useful sind nicht entscheidbar.
9) **Satz:** (von Rice): Sei $\mathcal{R}$ die Menge aller (partiell) berechenbaren Funktionen, $\emptyset \neq S \subsetneq \mathcal{R}$. Dann ist $L(S) := \{\langle M\rangle \mid M$ berechnet eine Funktion aus $S\}$ nicht entscheidbar.

3. Formale Sprachen und Automaten

1) **Definition:** Grammatik (vom Typ Chomsky-0) beschrieben durch 4-Tupel (V, Σ, P, S) mit V endl. Alphabet von Variablen, Σ endl. Alphabet von Terminalen, S Startsymbol, $P \subseteq ((V \cup \Sigma)^+ \setminus \Sigma^* \times (V \cup \Sigma)^*)$ engl. Menge von Produktionen.
   Satz: Sei L eine Sprache. Es gilt: L rekursiv aufzählbar $\Leftrightarrow$ $\exists$ Chomsky-0-Grammatik G mit $L(G) = L$.
   Definition: Eine Grammatik heißt:
   kontextsensitiv (Chomsky-1), wenn für jede Produktion $u \to v$ gilt: $|u| \leq |v|$. Ausnahme: Es ist $S \to \epsilon$ erlaubt, wenn gür alle anderen Produktionen $u \to v$ gilt, dass S nicht in v auftaucht.
   kontextfrei (Chomsky-2), wenn alle Produktionen der Art $u \to v$ mit $u \in V$ sind.
   regulär (Chomsky-3), wenn alle Produktionen der Art $u \to v$ mit $u \in V$ und $v \in \{\epsilon\} \cup \Sigma \cup \{aw \mid a \in \Sigma, w \in V\}$ sind.
   Satz: Kontextsensitive Sprachen sind

entscheidbar.

2) **Definition:** Deterministischer endlicher Automat (DFA) beschrieben durch 5-Tupel $(Q, \Sigma, \delta, q_0, F\}$.
3) **Definition:** Nichtdeterministischer endlicher Automat (NFA), Unterschied zum DFA: Übergangsfunktion δ hat Signatur $Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$.
   Satz: Sei N ein NFA. Dann ex. ein DFA A mit $L(A) = L(N)$.
   Satz: Sei L eine Sprache. Es gilt: L regulär $\Leftrightarrow \exists$ DFA A mit $L = L(A)$.
4) **Definition:** Minimalautomat einer Sprache L.
   Definition: Äquivalente Zustände
   Definition: Äquivalenzautomat zu einem DFA.
   Der NEQ-Algorithmus
   Satz: Ein DFA A und sein Äquivalenzautomat A' akzeptieren dieselbe Sprache L. A' ist ein Minimalautomat für L.
   Satz: Sei L eine reguläre Sprache und $A = (Q, \Sigma, \delta, q_0, F)$ ein DFA, der A akzeptiert. Dann kann ein Minimalautomat für L in Zeit $\mathcal{O}(|Q|^2|\Sigma|)$ auf einer RAM konstruiert werden.
5) **Lemma:** („Pumping-Lemma“) Sei $L \subseteq \Sigma^*$ regulär. Dann gilt:
   $\exists p \in \mathbb{N} : \forall x \in L, |x| \geq p : \exists u, v, w \in \Sigma^*, x = uvw, |uv| \leq p, |v| \geq 1 : \forall i \in \mathbb{N}_0 : uv^i w \in L$.
   Kontraposition:
   $(\forall p \in \mathbb{N} : \exists x \in L, |x| \geq p : \forall u, v, w \in \Sigma^*, x = uvw, |uv| \leq p, |v| \geq 1 : \exists i \in \mathbb{N}_0 : uv^i w \notin L) \Rightarrow L$ nicht regulär.
6) **Definition:** Reguläre Ausdrücke
   Satz: Sei Σ ein Alphabet, $L \subseteq \Sigma^*$ eine Sprache. Es gilt: L regulär $\Leftrightarrow \exists$ regulärer Ausdruck R über Σ^* mit $L(R) = L$.
   Satz: Reguläre Sprachen sind abgeschlossen bzgl. Sternbildung, Komplementbildung, Konkatenation, Durchschnitt, Vereinigung.
7) **Definition:** Kellerautomat beschrieben durch 6-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, F)$ mit Q endl. Menge von Zuständen, Σ endl. Eingabealphabet, Γ endl. Stapelalphabet, $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ Übergangsfunktion, q_0 Startzustand, $F \subseteq Q$ endl. Menge von akzeptierenden Zuständen.
   Satz: Sei L eine Sprache. Es gilt: L kontextfrei $\Leftrightarrow \exists$ PDA P mit $L = L(P)$.
8) Das Wortproblem: Entscheide, ob ein Wort in einer durch eine kontextfreie Grammatik erzeugten Sprache liegt.
   Definition: Chomsky-Normalform für eine kontextfreie Grammatik (Σ, V, P, S): Jede Produktion in P ist von der Form $v \to uz,\ u, z \in V \backslash \{S\}$ oder $v \to a,\ a \in \Sigma$. Zusätzlich ist $S \to \epsilon$ erlaubt.
   Satz: Zu jeder kontextfreien Grammatik kann eine Grammatik in CHomsky-Normalform konstruiert werden, die dieselbe Sprache erzeugt.
   Der CYK-Algorithmus
9) **Lemma:** („Pumping-Lemma“) Sei $L \subseteq \Sigma^*$ kontextfrei. Dann gilt:
   $\exists p \in \mathbb{N} : \forall w \in L, |w| \geq p : \exists u, v, x, y, z \in \Sigma^*, w = uvxyz, |vy| \geq 1, |vxy| \leq p : \forall i \in \mathbb{N}_0 : uv^i xy^i z \in L$.
10) **Satz:** Seien L_1, L_2 kontextfreie Sprachen. Dann sind auch $L_1 \cup L_2$, $L_1 \cdot L_2$ (Konkatenation) und L_1^* kontextfreie Sprachen. Kontextfreie Sprachen sind nicht unter Durchschnitt abgeschlossen.

2. Semester

Klassen von Sprachen

DTIME$(t(n)) := \{L \mid L$ ist eine Sprache, die von einer 1-Band DTM mit Laufzeit $\mathcal{O}(t(n))$ entschieden werden kann.$\}$

$$\text{P} := \bigcup_{k \in \mathbb{N}} \text{DTIME}(n^k)$$

NTIME$(t(n)) := \{L \mid L$ ist eine Sprache, die von einer NTM mit Laufzeit $\mathcal{O}(t(n))$ entschieden werden kann.$\}$

$$\text{NP} := \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$$

d.h. die Klasse aller Sprachen, die polynomiell verifizierbar sind.

Wichtige Sprachen

SAT $:= \{\langle \phi \rangle \mid \phi$ ist eine erfüllbare Boole'sche Formel.$\}$

3SAT $:= \{\langle \phi \rangle \mid \phi$ ist eine erfüllbare 3-KNF-Formel.$\}$

Clique $:= \{\langle (G, k) \rangle \mid G$ ist ein ungerichteter Graph mit einer k-Clique.$\}$

Knotenüberdeckung := $\{\langle G, k\rangle \mid G$ besitzt eine k-Knotenüberdeckung.$\}$

$\mathrm{RS}_{\mathrm{ent}} := \{\langle G, W, g, w\rangle \mid$ Es ex. ein $S \subseteq \{1, \ldots, n\}$ mit $\sum_{i\in S} g_i \leq g$ und $\sum_{i\in S} w_i \geq w\}$ mit $G = (g_1, \ldots, g_n)$ und $W = (w_1, \ldots, w_n)$

$\mathrm{TSP}_{\mathrm{ent}} := \{\langle \Delta, L\rangle \mid L \in \mathbb{N}$ und es gibt eine Rundreise durch alle n Städte der Länge $\leq L.\}$ mit $\Delta = (d_{ij}) \in \mathbb{N}^{n\times n},\ d_{ij} = d_{ji}$

Cut := $\{\langle G, k\rangle \mid G$ besitzt einen Schnitt der Größe mindestens $k.\}$

1. Einleitung

3) **Definition:** $f = \mathcal{O}(g) \Leftrightarrow \exists c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
$f = \Omega(g) \Leftrightarrow g = \mathcal{O}(f)$
$f = \Theta(g) \Leftrightarrow g = \mathcal{O}(f)$ und $f = \mathcal{O}(g)$
$f = o(g) \Leftrightarrow \forall c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) < c \cdot g(n)$
$f = \omega(g) \Leftrightarrow g = o(f)$

2. Zeitkomplexität und die Klasse P

1) **Definition:** Rechenschritt einer DTM ist das einmalige Anwender der Übergangsfunktion einer Turingmaschine.
Laufzeit (= Zeitkomplexität) einer Turingmaschine, die bei jeder Eingabe hält, ist die Abbildung
$T_M : \mathbb{N} \to \mathbb{N},\ n \to \max\{T_M(w) \mid w \in \Sigma^{\leq n}\}$.
Dabei bezeichnet $T_M(w),\ w \in \Sigma^*$ die Anzahl der Rechenschritte der DTM M bei Eingabe w.
Definition: Eine DTM M hat Laufzeit $\mathcal{O}(f(n))$, wenn gilt: $T_M(n) = \mathcal{O}(f(n))$.
Satz: Jede Sprache, die durch eine 1-Band-DTM mit Laufzeit $o(n \log n)$ entschieden werden kann, ist regulär.
Satz: Jede Mehrband-DTM (bzw. RAM) mit Laufzeit $\mathcal{O}(t(n)),\ t$ monoton wachsend, $t(n) \geq n$, kann durch eine 1-Band DTM mit Laufzeit $\mathcal{O}(t(n)^2)$ (bzw. $\mathcal{O}(t(n)^3)$) simuliert werden.

3) **Definition:** Ein Verifizierer V für eine Sprache $L \subseteq \Sigma^*$ ist eine DTM, wenn gilt:
$\exists k \in \mathbb{N} : \forall w \in \Sigma^* : (w \in L \Leftrightarrow \exists$ ein Wort $c, |c| \leq |w|^k : V$ akzeptiert $\langle w, c\rangle)$.
Definition: Ein Verifizierer V für L heißt polynomiell, wenn die Laufzeit von V für jede Eingabe $\langle w, c\rangle$ polynomiell zu $|w|$ ist.
Definition: Nichtdeterministische Turingmaschine (NTM) beschrieben durch 4-Tupel $(Q, \Sigma, \Gamma, \delta)$. Unterschied zur DTM: Signatur der Übergangsfunktion ist $\delta : Q \setminus \{q_{\mathrm{accept}}, q_{\mathrm{reject}}\} \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{R, L\})$.
Definition der Laufzeit analog zu DTMs.
Satz: Für jede NTM mit Laufzeit $\mathcal{O}(t(n)),\ t$ monoton wachsend, $t(n) \geq n$, gibt es eine (1-Band) DTM mit Laufzeit $2^{\mathcal{O}(t(n))}$.

4) Bei allen bekannten Optimierungsproblemen, zu denen sich ein Entscheidungsproblem/eine Sprache konstruieren lässt, ist aus der effizienten Lösung für das Entscheidungsproblem auch eine effiziente Lösung für das Optimierungsproblem ableitbar. (Kein Satz, da nicht allgemein bewiesen!)

5) Sei Σ ein Alphabet. Eine Funktion $f : \Sigma^* \to \Sigma^*$ heißt polynomiell berechenbar, wenn es eine DTM M mit polynomieller Laufzeit gibt, die f berechnet. M berechnet die Funktion f, wenn für alle $w \in \Sigma^*$ die DTM M bei Eingabe w im Zustand q_{accept} hält, der Kopf auf der ersten Zelle steht und der Inhalt des Bandes $f(w)$ ist ($\sqcup$s am Ende werden ignoriert).
Definition: Seien $A, B \in \Sigma^*$ Sprachen. A heißt auf B polynomiell reduzierbar, wenn es eine polynomiell Berechenbare Funktion f gibt mit:
$\forall w \in \Sigma^* : (w \in A \Leftrightarrow f(w) \in B)$.
Satz: $A \leq_p B \wedge B \in \mathrm{P} \Rightarrow A \in \mathrm{P}$
Satz: $3\mathrm{SAT} \leq_p$ Clique
Definition: Eine Sprache L heißt NP-Vollständig, wenn gilt:
$L \in \mathrm{NP}$ und $\forall A \in \mathrm{NP} : A \leq_p L$.
Satz: Sei L eine NP-vollständige Sprache. Dann gilt: $A \in \mathrm{NP} \wedge L \leq_p A \Rightarrow A$ ist NP-vollständig
Satz: (von Cook-Levin) SAT ist NP-vollständig.
Satz: 3SAT ist NP-vollständig.

6) Clique, Knotenüberdeckung, $\mathrm{RS}_{\mathrm{ent}}$, SubsetSum, $\mathrm{TSP}_{\mathrm{ent}}$ sind NP-vollständig.

4. Approximationsalgorithmen

1) **Definition:** Sei I eine Instanz eines Optimiertungsproblems. $w(s) > 0$ bezeichne den Wert einer Lösung s für I, $\mathrm{opt}(I)$ den Wert einer optimalen Lösung für die Instanz I. Der Approximationsfaktor ist dann definiert als:

$\delta_A(I) = \frac{w(A(I))}{\text{opt}(I)}$
Ein Algorithmus hat Approximationsfaktor k, wenn für jede Instanz I gilt:
$\delta_A(I) \geq k$ bei einem Maximierungsproblem und
$\delta_A(I) \leq k$ bei einem Minimierungsproblem.

3) Gilt P $\neq$ NP, dann gibt es für TSP_{opt} keinen Approximationsalgorithmus mit Güte $c > 1$.

5) Ein Approximationsschema für ein Optimierungsproblem ist eine Menge $\{A(\varepsilon) \mid \varepsilon > 0\}$ von Approximationsalgorithmen, wobei $A(\varepsilon)$ Approximationsgüte $1-\varepsilon$ bei Maximierungsproblemen und $1+\varepsilon$ bei Minimierungsproblemen hat.