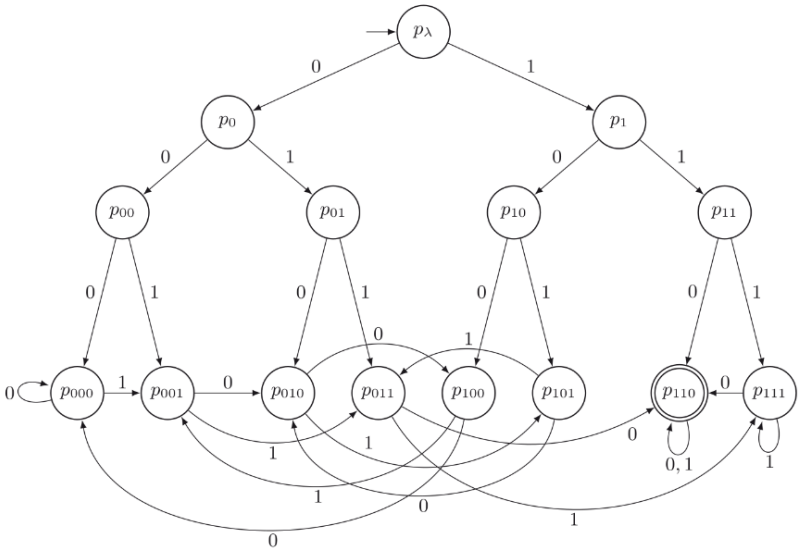
Theoretische Informatik

Formelsammlung



Contents

Anmerkung: *Unterkapitel-Nummern entsprechen nicht dem Buch*

Einleitung
Der Sinn dieses Dokuments ist, alle Resultate und Definitionen schnell auffindbar an einem Ort zu haben, z.B. für Hausaufgaben. Dieses Dokument ist keine Zusammenfassung, enthält aber einige Kommentare und Intuitive Erläuterungen (Text in grau).
Gute ausführliche Zusammenfassungen existieren bereits: z.B. die von Nicolas Wehrli, auf Community Solutions.

Wie immer: Keine Garantie auf Komplettheit oder Korrektheit.

Robin Bacher

ETH Zürich, HS25

Basierend auf:
Theoretische Informatik, J. Hromkovic

2 Formale Sprachen

Def: **Alphabet** $\Sigma \overset{\text{def}}{\iff} \Sigma$ endlich und $\Sigma \neq \emptyset$
$x \in \Sigma$ heisst Buchstabe, Zeichen, Symbol.

Def: **Wort** w über $\Sigma \overset{\text{def}}{\iff} w = (x_1, \ldots, x_n)$ endlich, $x_i \in \Sigma$.
$\Sigma^* := \{w \mid w \text{ ist Wort über } \Sigma\}$
$\lambda := w$ s.d. $|w| = 0$ (Leeres Wort)
$\Sigma^+ := \Sigma^* \setminus \{\lambda\}$

Teilwort	v von w	$\overset{\text{def}}{\iff}$	$\exists x, y \in \Sigma^* : w = xvy$
Präfix	v von w	$\overset{\text{def}}{\iff}$	$\exists y \in \Sigma^* : w = vy$
Suffix	v von w	$\overset{\text{def}}{\iff}$	$\exists x \in \Sigma^* : w = xv$

Note: Notation: $x_1 x_2 \ldots x_n = (x_1, x_2, \ldots, x_n)$

Nummer$(x) := \sum_{i=1}^{n} x_i \cdot 2^{n-i}$.
Bin(m) := kürzeste Binärkodierung von m in Σ_{Bool}.
Bin$(0) := 0$

Def: **Konkatenation** Kon$(x, y) = x \cdot y = xy$
$\forall w : \lambda \cdot w = w$
Kon ist assoziativ

Def: **Reversal** Für $a \in \Sigma^*$: $a^{\mathsf{R}} := a_n a_{n-1} \ldots a_1$

Def: **Iteration** Für $x \in \Sigma^*$: $x^0 := \lambda, x^1 := x, x^i := xx^{i-1}$

Def: **Vorkommen** von a in $w \in \Sigma^*$: $|w|_a := |\{i \mid w_i = a\}|$

Def: **Kanonische Ordnung** von Σ^*: Sei $<$ eine Ordnung über Σ. $u, v \in \Sigma^*$. $x, u', v' \in \Sigma^*$ und $i < j$.

$$u < v \iff |u| < |v| \quad \vee \quad |u| = |v| \wedge u = x \cdot s_i \cdot u' \wedge v = x \cdot s_j \cdot v'$$

(Das ist die lexikographische Ordnung bekannt aus DM)

Def: **Sprache** L über $\Sigma \overset{\text{def}}{\iff} L \subset \Sigma^*$

Def: **Komplement** $L^{\mathsf{C}} := \Sigma^* \setminus L$

Def: **Konkatenation** von L_1, L_2: $L^0 := L_\lambda, L^i := L^{i-1} \cdot L$

Def: **Kleene'scher Stern** $L^* := \bigcup_{i \geq 0} L^i, \quad L^+ := L^* \setminus \{\lambda\}$

Lemma: $\cup$ ist distributiv über Sprachen:

$$L_1 L_2 \cup L_1 L_3 = L_1 (L_2 \cup L_3)$$

2.1 Algorithmische Probleme

2.2 Kolmogorov Komplexität

Ziel: Komprimierung von Wörtern, Schliessen auf Informationsdichte basierend auf Komprimierbarkeits-schwierigkeit.

Def: **Kolmogorov Komplexität** $\forall x \in \Sigma^*_{\text{bool}} : K(x) :=$ kürzestes `Pascal`-Programm für x.

Vermeidet die Festlegung auf einen spezifischen Komprimieralgorithmus. Buch beweist ebenfalls, dass `Pascal` hier *keine* Einschränkung ist.

Def: $K(x)$ von Natürlichen Zahlen: $K(n) := K(\text{Bin}(n))$

Bin$(|x|)$ hat Länge $\lceil \log_2(|x| + 1) \rceil$

Lemma: $\exists d \; \forall x \in \Sigma^*_{\text{bool}} : \quad K(x) \leq |x| + d$

Lemma: $\forall n \geq 1 \; \exists w_n \in \Sigma^n_{\text{bool}} : \quad K(w_n) \geq |w_n| = n$

Intuitiv: Es existieren unkomprimierbare w jeder Länge.

Formalisierung der Zufälligkeit

Def: **Zufällig** $\overset{\text{def}}{\iff} x \in \Sigma^*_{\text{bool}}$ erfüllt $K(x) \geq |x|$

$n \geq 0$ ist zufällig $\overset{\text{def}}{\iff} K(n) = K(\text{Bin}(n)) \geq \lceil \log_2(n + 1) \rceil - 1$

Diese Definition hat nichts mit dem Zufallsbegriff aus der Wahrscheinlichkeit zu tun, hier geht es um den Informationsgehalt.

Verbindung: Entscheidungsprobleme und Komplexität

Theorem: **(2.2)** $\exists$ Programmm A_L welches $(\Sigma_{\text{bool}}, L)$ löst $\implies \forall n \geq 1 : K(z_n) \leq \lceil \log_2(n + 1) \rceil + c$

$L \subset \Sigma^*_{\text{bool}}$. $z_n :=$ n-tes Wort bzgl. kan. Ordnung. $(\Sigma_{\text{bool}}, L)$ ist ein Entscheidungsproblem.

Primzahlverteilung

Theorem: $\lim_{n \to \infty} \frac{\text{Prim}(n)}{n / \ln(n)} = 1$

Prim(x) := Anzahl Primzahlen kleiner x. Intuitiv: Anzahl Primzahlen wächst gleich schnell wie Anzahl Zahlen.

3 Endliche Automaten (EA)

Def: **Endlicher Automat** $M := (Q, \Sigma, \delta, q_0, F)$

Zustände	Q *(endlich)*
Eingabealphabet	Σ *(Alphabet)*
Anfangszustand	$q_0 \in Q$
Akzeptierende Zustände	$F \subseteq Q$
Übergangsfunktion	$\delta : Q \times \Sigma \to Q$

$\delta(q, a) = p \implies$ im Zustand q bei Eingabe a, gehe zu p.

Darstellungsformen: goto-Programm, gerichteter Graph

Def: **Konfiguration** von M: $(q, w) \in Q \times \Sigma^*$

Intuitiv: M hat Zustand q und liest noch den Suffix w

(q, w) Endkonfiguration $\overset{\text{def}}{\iff} (q, w) \in Q \times \{\lambda\}$

Def: **Schritt** $:= \vdash_M \subseteq (Q \times \Sigma^*) \times (Q \times \Sigma^*)$

wobei $(q, w) \vdash_M (p, x) \overset{\text{def}}{\iff} w = ax \wedge a \in \Sigma \wedge \delta(q, a) = p$

Intuitiv: Übergangsfunktion auf M im Zustand q anwenden, bei a.

Def: **Berechnung** $C := C_0, \ldots, C_n$ s.d. C_i Konfiguration wobei $\forall i \leq n-1 : C_i \vdash_M C_{i+1}$ gilt

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

Intuitiv: Menge aller Wörter, die M akzeptiert

Klasse regulärer Sprachen: $\mathcal{L}_{EA} := \{L(M) \mid \text{M ist EA}\}$

Def: **Klasse** $\text{Kl}[p] := \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) = p\}$

Klassen bilden eine Partition von Σ^*. Ähnlich zu Äquivalenzklassen aus DM.

Lemma: $L(M) = \bigcup_{p \in F} \text{Kl}[p]$

Def: **Relationen** bzgl. endlichen Automaten

$(q, w) \vdash_M^* (p, u) \overset{\text{def}}{\iff} \exists$ Berechnung in M von (q, w) zu (p, u).

Die formelle Definition ist sehr lang. (S.54)

$\hat{\delta}(q, w) = p \overset{\text{def}}{\iff} (q, w) \vdash_M^* (p, \lambda)$

Intuitiv: Wenn M in Zustand q Wort w liest, endet M in p

Lemma: $\forall \odot \in \{\cup, \cap, -\}\ \exists M : \quad L(M) = L(M_1) \odot L(M_2)$

Für alle EA M_1, M_2 über Σ

Wegen diesem Lemma ist es möglich, EAs aus Teilautomaten zu bauen

3.1 Irregularität beweisen

Einige Ansätze um Aussagen der Art $L \notin \mathcal{L}_{\text{EA}}$ zu beweisen:

Lemma: **(Direkt via Zustände)** Sei $A = (Q, \Sigma, \delta_A, q_0, F)$, $x \neq y \in \Sigma^*$:

$$\exists p \in Q : \underbrace{(q_0, x) \vdash_A^* (p, \lambda) \ \wedge \ (q_0, y) \vdash_A^* (p, \lambda)}_{\hat{\delta}_A(q_0, x) \,=\, \hat{\delta}_A(q_0, y) \,=\, p \text{ und } x, y \,\in\, \text{Kl}[p]} \implies \forall z \in \Sigma^*,\ \exists r \in Q : \quad xz \in L(A) \iff yz \in L(A)$$

Intuitiv: Wenn man für zwei (auch unterschiedliche!) Eingaben die selbe Konfiguration erreicht, ist der weitere Verlauf identisch. Dieses Lemma formalisiert die intuitiv klare "Gedächtnislosigkeit" von EAs, d.h. dass ein EA keinen Speicher (ausser dem aktuellen Zustand) besitzt.

Lemma: **(Pumping)** für $L \in \mathcal{L}_{\text{EA}}$: $\exists n_0 \in \mathbb{N}$ s.d. $\forall w \in \Sigma^*$ mit $|w| \geq n_0 : \ \exists x, y, z : w = yxz$ und:

$$(i)\ |yx| \leq n_0 \qquad (ii)\ |x| \geq 1$$
$$(iii)\ \{yx^k z \mid k \in \mathbb{N}\} \in L \text{ oder } \{yx^k z \mid k \in \mathbb{N}\} \cap L = \emptyset$$

Intuitiv: Alle Wörter länger als n_0 lassen sich als $w = yxz$ zerlegen: wenn w (nicht) akzeptiert wird müssen alle anderen $w_k = yx^k z$ auch (nicht) akzeptiert werden. Ein n_0 welches diese Zerlegung erlaubt existiert immer, wenn L regulär ist.

Theorem: **(Kolmogorov)** Sei $L \subseteq (\Sigma_{\text{bool}})^*$ regulär. Sei $L_x = \{y \in (\Sigma_{\text{bool}})^* \mid xy \in L\}$, y_n das n-te Wort in L_x

$$\exists c \quad \forall x, y \in (\Sigma_{\text{bool}})^* : \quad K(y_n) \leq \lceil \log_2(n+1) \rceil + c$$

Intuitiv: Suffixe von Wörtern einer regulären Sprache besitzen eine kleine Kolmogorov-Komplexität. Man versucht meistens eine unendliche Menge unterschiedlicher y_1 zu finden, was dann einen Widerspruch bildet zu diesem Satz.

3.2 Nicht-deterministische endliche Automaten (NEA)

Def: **NEA** $M = (Q, \Sigma, \delta, q_0, F)$

Übergangsfunktion $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

Intuitiv: δ gibt alle möglichen Zustände, statt nur Einen.

Def: **Relationen** bzgl. NEAs

$\hat{\delta}(q, \lambda) := \{q\} \quad \forall q \in Q$

$\hat{\delta}(q, wa) := \{p \in Q \mid \exists r \in \hat{\delta}(q, w) : p \in \delta(r, a)\}$

Intuitiv: $\hat{\delta}$ gibt alle möglichen Endzustände, statt nur Einen.

Def: **Akzeptierte Sprache** in NEAs

$$L(M_{\text{NEA}}) := \{w \in \Sigma^* \mid \hat{\delta}(q, w) \cap F \neq \emptyset\}$$

Intuitiv: Alle Wörter mit möglichen Berechnungsweg zu $q \in F$.

D.h. Akzeptierte Wörter müssen nicht immer akzeptiert werden.

Theorem: **(3.2) Potenzmengen Konstruktion**

$$\mathcal{L}_{\text{EA}} = \mathcal{L}_{\text{NEA}}$$

Intuitiv: Für jeden NEA gibt es einen äquivalenten EA.

4 Turing-Maschinen

Eine Formalisierung des Begriffs "Algorithmus".

Def: **Turing Maschine** $M := (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$

Zustände	Q *(endlich)*
Eingabealphabet	Σ *(Alphabet)*
Arbeitsalphabet	Γ s.d. $\Sigma \subset \Gamma, \Gamma \cap Q = \emptyset$
Anfangszustand	$q_0 \in Q$
Akzeptierende Zustände	$F \subseteq Q$
Übergangsfunktion	$\delta : Q \times \Sigma \to Q$

Def: **Konf**$(M) := \{¢\} \cdot \Gamma^* \cdot Q \cdot \Gamma^+ \cup Q \cdot \{¢\} \cdot \Gamma^*$

Beispiel: $¢w_1qaw_2 \in$ Konf(M) heisst:
M in Zustand q, hat Kopf bei $|w_1| + 1$ auf a. Bandinhalt: $¢w_1aw_2$

Def: **Äquivalenz** TMs A, B s.d. $\Sigma_A = \Sigma_B$:

1. $x \in L(A) \iff x \in L(B)$
2. A hält nicht auf $x \iff B$ hält nicht auf x

D.h. $\neg(L(A) = L(B) \implies A, B$ äquivalent).

4.1 Mehrband Turing-Maschinen

Def: **Mehrband TMs**

1. Endliche Kontroll-logik
2. Endliches Eingabeband
3. k nach rechts unendliche Arbeitsbänder

Die Formelle Definition im Skript ist sehr lang.
Intuitiv bleiben alle Definitionen gleich, akzeptieren aber nun k Bänder.

Lemma: $\forall$ TM $A : \exists$ MTM B s.d. A, B äquivalent.
Lemma: $\forall$ MTM $B : \exists$ TM A s.d. A, B äquivalent.

Theorem: TMs und MTMs sind äquivalente Modelle.
D.h. es existiert immer eine äquivalente Maschine im jeweils anderen Modell.

4.2 Nicht-deterministische TMs

Theorem: $\forall$ NTM $M : \exists$ TM A s.d.

1. $L(M) = L(A)$
2. A hält immer falls M keine unendlichen Berechnungen hat

D.h. auch NTMs sind konzeptuell äquivalent zu regulären TMs.

4.3 Sprach-Klassen

Def: **Rekursiv aufzählbar** $\mathcal{L}_{\text{RE}} := \{L(M) \mid M$ ist TM $\}$

Def: **Rekursiv entscheidbar** $\mathcal{L}_{\text{R}} := \{L(M) \mid M$ ist TM, hält immer $\}$

5 Berechenbarkeit

Methoden zur Klassifizierung Algorithmischer Lösbarkeit.

5.1 Diagonalisierung

Def: **Mächtigkeit**

$$|A| \leq |B| \quad \overset{\text{def}}{\iff} \quad \exists f : A \to B \text{ injektiv}$$
$$|A| = |B| \quad \overset{\text{def}}{\iff} \quad |A| \leq |B| \wedge |B| \leq |A|$$
$$|A| < |B| \quad \overset{\text{def}}{\iff} \quad |A| \leq |B| \wedge \neg |B| \leq |A|$$

Lemma: $A \subset B \implies |A| \leq |B|$

Lemma: $\leq$ ist Transitiv.

Def: **Abzählbarkeit** $\overset{\text{def}}{\iff}$ $|A| = |\mathbb{N}| \vee A$ endlich

Lemma: $\forall \Sigma : \Sigma^*$ ist abzählbar

Intuitiv: Da Σ endlich ist.

Weitere abzählbare Mengen: $\mathbb{Z}, \mathbb{N}^k, \mathbb{Q}$, KodTM

Überabzählbare Mengen: $\mathbb{R}, [0,1], \mathcal{P}((\Sigma_{\text{bool}})^*)$

Theorem: $|\text{KodTM}| \leq \mathcal{P}((\Sigma_{\text{bool}})^*)$

D.h. existieren unendliche viele nicht rekursiv aufzählbare Sprachen.

5.2 Reduktion

Ansatz für Beweise von Aussagen der Form $L \in \mathcal{L}_{\text{R}}$ oder $L \notin \mathcal{L}_{\text{R}}$.

Def: **Rekursive Reduzierbarkeit**

$L_1 \leq_{\text{R}} L_2 \overset{\text{def}}{\iff} L_2 \in \mathcal{L}_{\text{R}} \implies L_1 \in \mathcal{L}_{\text{R}}$

D.h. L_2 zu lösen, bedeuted auch L_1 zu lösen.

Def: **Eingabe-zu-Eingabe Reduzierbarkeit**

$L_1 \leq_{\text{EE}} L_2 \overset{\text{def}}{\iff} \exists M \text{ (TM)} : \exists f_M : \Sigma_1^* \to \Sigma_2^* \text{ s.d. } x \in L_1 \iff f_M(x) \in L_2$

D.h. Es existiert eine TM M, die eine Abbildung f_M darstellt, mit welcher man L_1 via L_2 direkt bestimmen kann.

Lemma: $L_1 \leq_{\text{EE}} L_2 \implies L_1 \leq_{\text{R}} L_2$

Lemma: $\leq_{\text{EE}}$ ist Transitiv.

Lemma: $\forall L \subseteq \Sigma^* : L \leq_{\text{R}} L^{\text{C}} \ \wedge \ L^{\text{C}} \leq_{\text{R}} L$

Lemma: $\mathcal{L}_{\text{R}} \subsetneq \mathcal{L}_{\text{RE}}$

Def: **Universelle Sprache**

$L_{\text{U}} := \{\text{Kod}(M)\#w \mid w \in \Sigma_{\text{bool}}^* \wedge w \in L(M)\}$

Theorem: $L_{\text{U}} \in \mathcal{L}_{\text{RE}}$ aber $L_{\text{U}} \notin \mathcal{L}_{\text{R}}$

Def: **Halteproblem**

$L_{\text{H}} := \{\text{Kod}(M)\#x \mid x \in \Sigma_{\text{bool}}^* \wedge M \text{ hält auf } x\}$

Theorem: $L_{\text{H}} \notin \mathcal{L}_{\text{R}}$

D.h. man kann nie wissen, ob eine Turingmaschine anhalten wird.

Def: $L_{\text{Empty}} := \{\text{Kod}(M) \mid L(M) = \emptyset\}$

Theorem: $(L_{\text{Empty}})^{\text{C}} \in L_{\text{RE}}$ aber $(L_{\text{Empty}})^{\text{C}} \notin \mathcal{L}_{\text{R}}$

Def: **Äquivalenzproblem**

$L_{\text{EQ}} = \{\text{Kod}(M)\#\text{Kod}(\overline{M}) \mid L(M) = L(\overline{M})\}$

Theorem: $L_{\text{EQ}} \notin \mathcal{L}_{\text{R}}$

D.h. man kann nicht 2 TMs auf Äquivalenz prüfen, in endlicher Zeit.

TODO: reformat Def, Theorems as a table for languages

5.3 Rice

Ansatz für Beweise von Aussagen der Form $L \notin \mathcal{L}_{\text{R}}$, für $L \subseteq$ KodTM.

Def: **Semantisch nicht-triviales Entscheidungsproblem über TMs**

$$L \subseteq \text{KodTM} : \underbrace{(\exists \text{ TM } M_1 : \text{Kod}(M_1) \in L)}_{L \neq \emptyset} \wedge \underbrace{(\exists \text{ TM } M_2 : \text{Kod}(M_2) \notin L)}_{L \neq \text{KodTM}} \wedge \underbrace{(\forall \text{ TM } A, B : L(A) = L(B) \implies (A \in L \iff B \in L))}_{L \text{ behandelt semantisch gleiche TMs gleich}}$$

Def: $L_{\text{H},\lambda} := \{\text{Kod}(M) \mid M \text{ hält auf } \lambda\}$

Theorem: $L_{\text{H},\lambda} \notin \mathcal{L}_{\text{R}}$

Theorem: **Satz von Rice**: Alle sem. nicht-triv. Probleme L sind unentscheidbar. $(L \notin \mathcal{L}_{\text{R}})$

D.h. es reicht aus zu zeigen, dass $L \subseteq$ KodTM die Bedingungen oben erfüllt, um $L \notin \mathcal{L}_{\text{R}}$ zu zeigen.

5.4 Kolmogorov

Theorem: **Unlösbarkeit von Kolmogorov**: Das Problem, $\forall x \in (\Sigma_{\text{bool}})^*$ die Komplexität $K(x)$ zu berechnen, ist unlösbar.

Ein Alternativer Ansatz um Unlösbarkeit zu zeigen, unabhängig von Diagonalisierung.

6 Komplexität

Eine Formalisierung der "Schwierigkeit" von Algorithmisch lösbaren Problemen.

6.1 Zeit & Speicher

Def: $\textbf{Time}_{\textbf{M}}(x) := k - 1$

Def: $\textbf{Time}_{\textbf{M}}(n) := \max\{\text{Time}_{\text{M}}(x) \mid x \in \Sigma^n\}$

Intuitiv: Die Komplexität im Schlechtesten Fall einer Eingabe der Länge n

Wobei: M immer hält, $x \in \Sigma^*$ und $D = C_1 C_2 \dots C_k$ Die Berechnung von M auf x

Def: $\textbf{Space}_{\textbf{M}}(C) := \max\{|\alpha_i| \mid i = 1, \dots, k\}$

Intuitiv: Die Länge des längsten Arbeitsbandes in M, bei der Konfiguration C.

Def: $\textbf{Space}_{\textbf{M}}(x) := \max\{\text{Space}_M(C_i) \mid i = 1, \dots, l\}$

Def: $\textbf{Space}_{\textbf{M}}(n) := \max\{\text{Space}_M(x) \mid x \in \Sigma^n\}$

Wobei M eine k-Band MTM, $C = (q, x, i, \alpha_1, i_1 \dots, \alpha_k, i_k)$ eine Konfiguration.

Lemma: $\forall$ k-MTM A : $\exists$ äquivalente 1-MTM B : $\text{Space}_B(n) \leq \text{Space}_A(n)$

Lemma: $\forall$ k-MTM A : $\exists$ äquivalente k-MTM B : $\text{Space}_B(n) \leq \frac{\text{Space}_A(n)}{2} + 2$

Intuitiv: Die Speicherkomplexität von M lässt sich für jedes $d \in \mathbb{N}$ um den Faktor d verkleinern. Das selbe gilt für $\text{Time}_M(n)$.

Theorem: $\exists (L, \Sigma_{\text{bool}})$ $\forall$ MTM A s.d. $L(A) = L$: $\exists$ MTM B s.d. $L(B) = L$ und $\text{Time}_B(n) \leq \log_2(\text{Time}_A(n))$

Intuitiv: Es gibt Probleme, wobei wir einen Lösungsalgorithmus unendlich oft signifikant verbessern können. D.h. macht es keinen Sinn allgemein von einem "Besten Algorithmus" für ein Problem zu reden.

6.2 O-Notation

Def: $\mathcal{O}(f(n)) := \{r : \mathbb{N} \to \mathbb{R}^+ \mid \exists n_0 \in \mathbb{N}, \exists c \in \mathbb{N} \text{ s.d. } \forall n \geq n_0 : \quad r(n) \leq c \cdot f(n)\}$

Def: $\Omega(f(n)) := \{r : \mathbb{N} \to \mathbb{R}^+ \mid \exists n_0 \in \mathbb{N}, \exists c \in \mathbb{N} \text{ s.d. } \forall n \geq n_0 : \quad r(n) \geq \frac{1}{c} \cdot f(n)\}$

Def: $\Theta(f(n)) := \mathcal{O}(f(n)) \cap \Omega(f(n))$

Def: $\text{o}(f(n)) := \{r : \mathbb{N} \to \mathbb{R}^+ \mid \lim_{n \to \infty} \frac{r(n)}{f(n)} = 0\}$

Intuitiv: f wächst asymptotisch schneller als r.

6.3 Komplexitätsklassen

$f, g : \mathbb{N} \to \mathbb{R}^+$

Def: $\textbf{TIME}(f) := \{L(M) \mid M \text{ s.d. } \text{Time}_M(n) \in \mathcal{O}(f(n))\}$

Def: $\textbf{SPACE}(f) := \{L(M) \mid M \text{ s.d } \text{Space}_M(n) \in \mathcal{O}(f(n)))\}$

Def: $\textbf{P} := \bigcup_{c \in \mathbb{N}} \text{TIME}(n^c)$

Def: $\textbf{PSPACE} := \bigcup_{c \in \mathbb{N}} \text{SPACE}(n^c)$

Def: $\textbf{EXPTIME} := \bigcup_{d \in \mathbb{N}} \text{TIME}(2^{n^d})$

Lemma: $\forall t : \mathbb{N} \to \mathbb{R}^+ : \quad \text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$

Lemma: DLOG $\subseteq$ P $\subseteq$ PSPACE $\subseteq$ EXPTIME

TODO: Konstruierbarkeit

6.4 Nicht-deterministische Komplexität

$M :=$ Nicht deterministische (M)TM. $C = C_1 \dots C_m$ ist eine akzeptierende Berechnung auf x.

Def: $\textbf{Time}_M(x) :=$ Länge kürzester akzept. Berechnung für x.

Def: $\textbf{Time}_M(n) := \max(\{\text{Time}_M(x) \mid x \in L(M) \wedge |x| = n\} \cup \{0\})$

Def: $\textbf{Space}_M(C) := \max\{\text{Space}_M(C_i) \mid i \leq m\}$

Def: $\textbf{Space}_M(x) := \min\{C \mid C \text{ akzeptiert } x\}$

Def: $\textbf{Space}_M(n) := \max(\{x \in L(M) \wedge |x| = n\} \cup \{0\})$

Def: **Kompelxitätsklassen**: **NTIME**, **NSPACE**, **NLOG**, **NP**, **NSPACE** analog zur detereministischen Definition.

6.5 NP-Vollständigkeit

Unter der Annahme: $P \subsetneq NP$, kann man Beweise der Form $L \notin P$ machen.

Def: **Polynomielle Reduzierbarkeit**

$L_1 \leq_\text{p} L_2 \overset{\text{def}}{\iff} \exists$ polynomielle M s.d. $\forall x \in (\Sigma_1)^* : \quad x \in L_1 \iff M(x) \in L_2$

Intuitiv: EE-Reduktion, muss aber polynomielle Zeitkomplexität haben.

Def: **NP-Schwer** L s.d. $\forall L' \in \text{NP} : L' \leq_\text{p} L$

L NP-Schwer bedeutet *nicht*, dass L in NP ist.

Def: **NP-Vollständigkeit** L s.d. $L \in$ NP und NP-Schwer

Lemma: $\exists L : \; L \in \text{P} \wedge \text{NP-Schwer} \implies \text{P} = \text{NP}$

Ein NP-Schweres Problem polynomiell zu lösen beutet alle NP-Probleme polynomiell zu lösen.

Lemma: $L_1 \leq_\text{p} L_2 \implies (L_1 \text{ NP-schwer} \implies L_2 \text{ NP-schwer})$

D.h. Mit P-Reduktionen kann man beweisen, dass L_2 NP-Schwer ist.

Def: **SAT** $:= \{x \in (\Sigma_\text{logic})^* \mid x \text{ kodiert erfüllbare Formel in KNF}\}$

Theorem: **(Cook)** SAT ist NP-vollständig.

Der Beweis ist sehr lang. Im Endeffekt bedeutet dies, Boole'sche Formeln sind enorm ausdrucksstark.

6.6 Klausel-Formeln

Nützliche Gleichungen für Beweise mit KNF-Formeln