Theoretische Informatik HS24

Nicolas Wehrli

Übungsstunde 06
29. Oktober 2024

ETH Zürich
nwehrl@ethz.ch

Heute

1. Feedback zur Serie
2. Recap: Turing Maschinen
3. Nichtdeterministische Turingmaschinen
4. Einstieg Berechnenbarkeit
   - Diagonalisierung
5. Reduktion

Feedback zur Serie

Feedback

- Justification beim Widerspruch im Pumping Lemma
- Davon abgesehen, relativ gut gelöst

Recap: Turing Maschinen

Turing Maschinen - Formalisierung von Algorithmen

Informell

Eine Turingmaschine besteht aus

(i) einer endlichen Kontrolle, die das Programm enthält,

(ii) einem unendlichen Band, das als Eingabeband, aber auch als Speicher (Arbeitsband) zur Verfügung steht, und

(iii) einem Lese-/Schreibkopf, der sich in beiden Richtungen auf dem Band bewegen kann.

Für formale Beschreibung siehe Buch.

Turing Maschinen - Formalisierung von Algorithmen

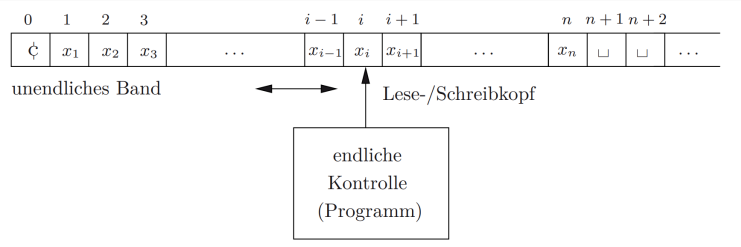


Abbildung 1: Abb. 4.1 vom Buch

Elementare Operation einer TM - Informell

Input

- Zustand der Maschine (der Kontrolle)
- Symbol auf dem Feld unter dem Lese-/Schreibkopf

Aktion

(i) ändert Zustand

(ii) schreibt auf das Feld unter dem Lese-/Schreibkopf

(iii) bewegt den Lese-/Schreibkopf nach links, rechts oder gar nicht. Ausser wenn ¢, dann ist links nicht möglich.

Turing Maschinen - Formalisierung von Algorithmen

Eine **Konfiguration** C von M ist ein Element aus

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

- Eine Konfiguration $¢w_1qaw_2$ mit $w_1, w_2 \in \Gamma^*$, $a \in \Gamma$ und $q \in Q$ sagt uns: M im Zustand q, Inhalt des Bandes $¢w_1aw_2␣␣␣...$, Kopf an Position $|w_1| + 1$ und liest gerade a.
- Eine Konfiguration $p¢w$ mit $p \in Q$, $w \in \Gamma^*$: Inhalt des Bandes $¢w␣␣␣...$, Zustand p und Kopf an Position 0.

Bmk: Im Buch haben sie in der Definition von Konf Γ^+ anstatt Γ^* an ”letzter Stelle”.

Turing Maschinen - Formalisierung von Algorithmen

Es gibt wieder eine Schrittrelation $\vdash_M \subseteq \text{Konf}(M) \times \text{Konf}(M)$.

Abbildung 2: Diagramm von Adeline

Turing Maschinen - Formalisierung von Algorithmen

Berechnung von M, Berechnung von M auf einer Eingabe x etc. durch $\vdash_M$ definiert.

Die Berechnung von M auf x heisst

- **akzeptierend**, falls sie in einer akzeptierenden Konfiguration $w_1 q_{\text{accept}} w_2$ endet (wobei ¢ in w_1 enthalten ist).
- **verwerfend**, wenn sie in in einer verwerfenden Konfiguration $w_1 q_{\text{reject}} w_2$ endet.
- **nicht-akzeptierend**, wenn sie entweder eine **verwerfende** oder unendliche Berechnung ist.

Turing Maschinen - Formalisierung von Algorithmen

Die von der **Turingmaschine M akzeptierte Sprache** ist

$$\mathbf{L}(\mathbf{M}) = \{w \in \Sigma^* \mid q_0 ¢ w \vdash_M^* y q_{\text{accept}} z, \text{ für irgendwelche } y, z \in \Gamma^*\}$$

Wichtige Klassen

Reguläre Sprachen

$$\mathcal{L}_{\mathbf{EA}} = \{L(A) \mid A \text{ ist ein EA}\} = \mathcal{L}_{\mathbf{NEA}}$$

Rekursiv aufzählbare Sprachen

Eien Sprache $L \subseteq \Sigma^*$ heisst **rekursiv aufzählbar**, falls eine TM M existiert, so dass $L = L(M)$.

$$\mathcal{L}_{\mathbf{RE}} = \{L(M) \mid M \text{ ist eine TM}\}$$

ist die **Klasse aller rekursiv aufzählbaren Sprachen**.

Wichtige Klassen

Halten

Wir sagen das M **immer hält**, wenn für alle Eingaben $x \in \Sigma^*$

(i) $q_0 ¢ x \vdash_M^* y q_{\text{accept}} z, y, z \in \Gamma^*,$ falls $x \in L$ und

(ii) $q_0 ¢ x \vdash_M^* u q_{\text{reject}} v, u, v \in \Gamma^*,$ falls $x \notin L$.

Rekusive Sprachen

Eine Sprache $L \subseteq \Sigma^*$ heisst **rekursiv (entscheidbar)**, falls $L = L(M)$ für eine TM M, die **immer hält**.

$$\mathcal{L}_{\mathbf{R}} = \{L(M) \mid M \text{ ist eine TM, die immer hält}\}$$

ist die **Klasse der rekursiven (algorithmisch erkennbaren) Sprachen**.

Mehrband-Turingmaschine

Mehrband-TM - Informelle Beschreibung

Für $k \in \mathbb{N} \setminus \{0\}$ hat eine k-Band Turingmaschine

- eine endliche Kontrolle
- ein endliches Band mit einem Lesekopf (Eingabeband)
- k Arbeitsbänder, jedes mit eigenem Lese-/Schreibkopf (nach rechts unendlich)

Insbesondere gilt 1-Band TM $\neq$ "normale" TM

Am Anfang der Berechnung einer MTM M auf w

- Arbeitsbänder "leer" und die k Lese-/Schreibköpfe auf Position 0.
- Inhalt des Eingabebands ¢w\$ und Lesekopf auf Position 0.
- Endliche Kontrolle im Zustand q_0.

Äquivalenz von Maschinen (TM, MTM)

Seien A und B zwei Maschinen mit **gleichem** Σ.
Wir sagen, dass **A äquivalent zu B ist**, wenn für jede Eingabe $x \in \Sigma^*$

(i) A akzeptiert $x \iff B$ akzeptiert x
(ii) A verwirft $x \iff B$ verwirft x
(iii) A arbeitet unendlich lange auf $x \iff B$ arbeitet unendlich lange auf x

Wir haben

$$A \text{ und } B \text{ äquivalent} \implies L(A) = L(B)$$

aber

$$L(A) = L(B) \not\implies A \text{ und } B \text{ äquivalent}$$

da A auf x unendlich lange arbeiten könnte, während B x verwirft.

Äquivalenz von 1-Band TM zu TM

Lemma 4.1

Zu jeder TM A existiert eine zu A äquivalente 1-Band-TM B

Beweisidee

B kopiert die Eingabe zuerst aufs Arbeitsband und simuliert dann A.

Äquivalenz von TM zu k-Band-TM

Lemma 4.2

Zu jeder Mehrband-TM A existiert eine zu A äquivalente TM B

Beweisidee

Vergrösserung des Alphabets, jedes Zeichen enthält jetzt $2(k+1)$ Zeichen.

B simuliert A einen Schritt von A indem es den ganzen Inhalt liest und dann durch die endliche Kontrolle von A jede Schreib und Bewegungsoperation einzeln ausführt.

Dies verwendet immer nur **endlich** viele Schritte um einen Schritt von A zu simulieren.

Äquivalenz Folgerung

Aus Lemma 4.1 und 4.2 folgt direkt

Satz 4.1

Die Maschinenmodelle von Turingmaschinen und Mehrband-Turingmaschinen sind äquivalent.

Note:

> ”Äquivalenz” für Maschinenmodelle wird in Definition 4.2 definiert. Maschinenmodelle sind Klassen von Maschinen (i.e. Mengen von Maschinen mit gewissen Eigenschaften).

Nichtdeterministische Turingmaschinen

Definition von NTM

Eine **nichtdeterministische Turingmaschine (NTM)** ist ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, wobei

(i) $Q, \Sigma, \Gamma, q_{\text{accept}}, q_{\text{reject}}$ die gleiche Bedeutung wie bei einer TM haben, und

(ii) $\delta : (Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\text{L}, \text{R}, \text{N}\})$ die **Übergangsfunktion** von M ist und die folgende Eigenschaft hat:

$$\delta(p, ¢) \subseteq \{(q, ¢, X) \mid q \in Q, X \in \{R, N\}\}$$

für alle $p \in Q$

Konfiguration ähnlich wie bei TMs.

Konfiguration akzeptierend $\iff$ enthält q_{accept}
Konfiguration verwerfend $\iff$ enthält q_{reject}

Die üblichen Sachen

- Schrittrelation $\vdash_M$ "verbindet zwei Konfigurationen, wenn man von der einen in die andere kommen kann"
- Reflexive und transitive Hülle ist $\vdash_M^*$.
- Berechnung von M ist eine Folge von Konfigurationen $C_1, C_2, ...$, so dass $C_i \vdash_M C_{i+1}$.
- Eine Berechnung von M auf x ist beginnt in $q_0 ¢ x$ und endet entweder unendlich oder endet in $\{q_{\text{accept}}, q_{\text{reject}}\}$.

Akzeptierte Sprache

$$L(M) = \{w \in \Sigma^* \mid q_0 ¢ w \vdash_M^* y q_{\text{accept}} z \text{ für irgendwelche } y, z \in \Gamma^*\}$$

Berechnungsbaum

Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ eine NTM und sei x ein Wort über dem Eingabealphabet Σ von M. Ein **Berechnungsbaum $\mathbf{T_{M,x}}$ von M auf x** ist ein (potentiell unendlicher) gerichteter Baum mit einer Wurzel, der wie folgt definiert wird.

(i) Jeder Knoten von $T_{M,x}$ ist mit einer Konfiguration beschriftet.

(ii) Die Wurzel ist der einzige Knoten von $T_{M,x}$ mit dem Eingangsgrad 0 und ist mit der Startkonfiguration $q_0 ¢ x$ beschriftet.

(iii) Jeder Knoten des Baumes, der mit einer Konfiguration C beschriftet ist, hat genauso viele Kinder wie C Nachfolgekonfigurationen hat, und diese Kinder sind mit diesen Nachfolgekonfigurationen C markiert.

Äquivalenz NTM und TM

Satz 4.2

Sei M eine NTM. Dann existiert eine TM A, so dass

(i) $L(M) = L(A)$ und

(ii) falls M keine unendlichen Berechnungen auf Wörtern aus $L(M)^{\complement}$ hat, dann hält A immer.

Beweisidee:

”BFS im Berechnungsbaum”, i.e. wir simulieren einzelne Schritte der verschiedenen Berechnungsstränge.

Einstieg Berechnenbarkeit

Bijektion, Injektion, Schreibweise

Seien A und B zwei Mengen.
Wir sagen, dass

i. $|\mathbf{A}| \leq |\mathbf{B}|$, falls eine injektive Funktion $f : A \to B$ existiert.
ii. $|\mathbf{A}| = |\mathbf{B}|$, falls $|A| \leq |B|$ und $|B| \leq |A|$.
iii. $|\mathbf{A}| < |\mathbf{B}|$, falls $|A| \leq |B|$ und keine injektive Abbildung von B nach A existiert.

Zur Erinnerung:

$$f : A \to B \text{ injektiv } \iff \forall x, y \in A, x \neq y . f(x) \neq f(y)$$

Abzählbarkeit

Eine Menge A heisst **abzählbar**, falls A endlich ist oder $|A| = |\mathbb{N}|$.

Lemma 5.1

Sei Σ ein beliebiges Alphabet. Dann ist Σ^* abzählbar.

Beweisidee

kanonische Ordnung gibt uns eine Bijektion zwischen $\mathbb{N}$ und Σ^*.

Abzählbarkeit

Satz 5.1

Die Menge **KodTM** der Turingmaschinenkodierungen ist abzählbar.

Beweisidee

KodTM $\subseteq (\Sigma_{\text{bool}})^*$ und Lemma 5.1

Lemma 5.2

$(\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ ist abzählbar.

Beweisidee

Unendliche 2-dimensionale Tabelle, so dass an der i-ten Zeile und j-ten Spalte, sich das Element $(i, j) \in (\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ befindet.

Formal definiert man dabei die lineare Ordnung

$$(a, b) < (c, d) \iff a + b < c + d \text{ oder } (a + b = c + d \text{ und } b < d)$$

Abzählbarkeit

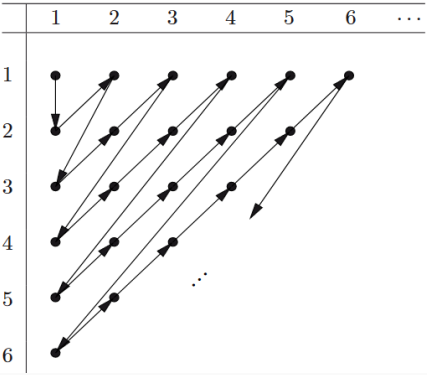


Abbildung 3: Abbildung 5.3 im Buch

Abzählbarkeit

Die i-te Diagonale hat i Elemente. Ein beliebiges Element $(a, b) \in (\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ ist das b-te Element auf der $(a + b - 1)$-ten Diagonale.

Auf den ersten $a + b - 2$ Diagonalen gibt es

$$\sum_{i=1}^{a+b-2} i = \frac{(a + b - 2) \cdot ((a + b - 2) + 1)}{2} = \binom{a + b - 1}{2}$$

Elemente.

Folglich ist

$$f((a, b)) = \binom{a + b - 1}{2} + b$$

eine Bijektion von $(\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ nach $\mathbb{N} \setminus \{0\}$.

Satz 5.3

$[0, 1]$ ist nicht abzählbar.

Beweisidee

Klassisches Diagonalisierungsargument. Aufpassen auf 0 und 9. I.e. $1 = 0.\overline{99}$.

Überabzählbarkeit

$f(x)$	$x \in [0,1]$							
1	0.	$\boxed{a_{11}}$	a_{12}	a_{13}	a_{14}	$\dots$		
2	0.	a_{21}	$\boxed{a_{22}}$	a_{23}	a_{24}	$\dots$		
3	0.	a_{31}	a_{32}	$\boxed{a_{33}}$	a_{34}	$\dots$		
4	0.	a_{41}	a_{42}	a_{43}	$\boxed{a_{44}}$	$\dots$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\dots$		
i	0.	a_{i1}	a_{i2}	a_{i3}	a_{i4}	$\dots$	$\boxed{a_{ii}}$	$\dots$
$\vdots$	$\vdots$							

Abbildung 5.5

Satz 5.4

$\mathcal{P}((\Sigma_{\text{bool}})^*)$ ist nicht abzählbar.

Beweis:

Wir definieren eine injektive Funktion von $f : [0,1] \to \mathcal{P}((\Sigma_{bool})^*)$ und beweisen so $|\mathcal{P}((\Sigma_{bool})^*)| \geq |[0,1]|$.

Sei $a \in [0,1]$ beliebig. Wir können a wie folgt binär darstellen:

$$\text{Nummer}(a) = 0.a_1a_2a_3a_4... \text{ mit } a = \sum_{i=1}^{\infty} a_i \cdot 2^{-i}.$$

Hier ist zu beachten, dass wir für eine Zahl a immer die lexikographisch letzte Darstellung wählen.

Überabzählbarkeit

Dies tun wir, weil eine reelle Zahl 2 verschiedene Binärdarstellungen haben kann. Beispiel: $\frac{1}{2} = 0.1\overline{0} = 0.0\overline{1}$.

Für jedes a definieren wir:

$$f(a) = \{a_1, a_2a_3, a_4a_5a_6, ..., a_{\binom{n}{2}+1}a_{\binom{n}{2}+2}...a_{\binom{n+1}{2}}, ...\}$$

Da $f(a) \subseteq (\Sigma_{bool})^*$ gilt $f(a) \in \mathcal{P}((\Sigma_{bool})^*)$.

Wir haben für alle $n \in \mathbb{N} \setminus \{0\}$, dass $f(a)$ **genau** ein Wort dieser Länge enthält. Nun können wir daraus folgendes schliessen:

Weil die Binärdarstellung zweier unterschiedlichen reellen Zahlen an mindestens einer Stelle unterschiedlich ist, gilt $b \neq c \implies f(b) \neq f(c), \forall b, c \in [0, 1]$.

Überabzählbarkeit

Folglich ist f injektiv und wir haben $|\mathcal{P}((\Sigma_{bool})^*)| \geq |[0,1]|$.

Da $[0,1]$ nicht abzählbar ist, folgt daraus:

$\mathcal{P}((\Sigma_{bool})^*)$ ist nicht abzählbar.

■

Diagonalsprache L_{diag}

Zur Erinnerung:

Rekursiv aufzählbare Sprachen

Eien Sprache $L \subseteq \Sigma^*$ heisst **rekursiv aufzählbar**, falls eine TM M existiert, so dass $L = L(M)$.

$$\mathcal{L}_{\mathbf{RE}} = \{L(M) \mid M \text{ ist eine TM}\}$$

ist die **Klasse aller rekursiv aufzählbaren Sprachen**.

Wir zeigen jetzt per Diagonalisierung, die Existenz einer Sprache die nicht rekursiv aufzählbar ist.

Diagonalsprache L_{diag}

Sei $w_1, w_2, \ldots$ die kanonische Ordnung aller Wörter über Σ_{bool} und sei $M_1, M_2, M_3, \ldots$ die Folge aller Turingmaschinen.

Wir definieren eine unendliche (bool'sche) Matrix $A = [d_{ij}]_{i,j=1,2,\ldots}$ mit

$$d_{ij} = 1 \iff M_i \text{ akzeptiert } w_j.$$

Wir definieren

$$L_{\text{diag}} = \{w \mid w = w_i \text{ und } M_i \text{ akzeptiert } w_i \text{ nicht für ein } i \in \mathbb{N} \setminus \{0\}\}$$

Diagonalsprache L_{diag}

Satz 5.5

$$L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$$

Beweis:

Wir haben

$$L_{\text{diag}} = \{w \mid w = w_i \text{ und } M_i \text{ akzeptiert } w_i \text{ nicht für ein } i \in \mathbb{N} \setminus \{0\}\}$$

Widerspruchsbeweis:

Sei $L_{\text{diag}} \in \mathcal{L}_{\text{RE}}$. Dann existiert eine TM M, so dass $L(M) = L_{\text{diag}}$. Da diese TM eine TM in der Nummerierung aller TM ist, existiert ein $i \in \mathbb{N}$, so dass $M_i = M$.

Diagonalsprache L_{diag}

Wir betrachten nun das Wort w_i für diese $i \in \mathbb{N}$. Per Definition von L_{diag}, gilt:

$$w_i \in L_{\text{diag}} \iff w_i \notin L(M_i)$$

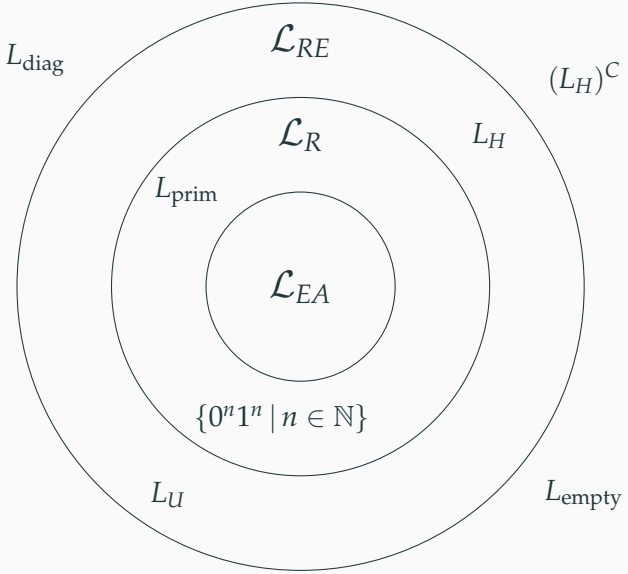
Da aber $L(M_i) = L_{\text{diag}}$, haben wir folgenden Widerspruch:

$$w_i \in L_{\text{diag}} \iff w_i \notin L_{\text{diag}}$$

Folglich gilt $L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$.

■

Klassifizierung verschiedener Sprachen



Begrifflichkeiten

Für eine Sprache L gilt folgendes

$$L \text{ regulär} \iff L \in \mathcal{L}_{\mathbf{EA}} \iff \exists \text{ EA } A \text{ mit } L(A) = L$$
$$L \text{ rekursiv} \iff L \in \mathcal{L}_{\mathbf{R}} \iff \exists \text{ Alg. A mit } L(A) = L$$
$$L \text{ rekursiv aufzählbar} \iff L \in \mathcal{L}_{\mathbf{RE}} \iff \exists \text{ TM } M.\ L(M) = L$$

"Algorithmus" = TM, die immer hält.

L rekursiv = L entscheidbar

L rekursiv aufzählbar = L erkennbar

Reduktion

Things

Reduktionen sind klassische Aufgaben an dem Endterm. Ein bisschen wie Nichtregularitätsbeweise.
Ist aber auch nicht so schlimm.

Definition 5.3

Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ zwei Sprachen. Wir sagen, dass **L_1 auf L_2 rekursiv reduzierbar ist, $L_1 \leq_R L_2$**, falls

$$L_2 \in \mathcal{L}_R \implies L_1 \in \mathcal{L}_R$$

Bemerkung:

Intuitiv bedeutet das *"L_2 mindestens so schwer wie L_1"* (bzgl. algorithmischen Lösbarkeit).

Definition 5.4

Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ zwei Sprachen. Wir sagen, dass **L_1 auf L_2 EE-reduzierbar ist, $L_1 \leq_{EE} L_2$,** wenn eine TM M existiert, die eine Abbildung $f_M : \Sigma_1^* \to \Sigma_2^*$ mit der Eigenschaft

$$x \in L_1 \iff f_M(x) \in L_2$$

für alle $x \in \Sigma_1^*$ berechnet. Wir sagen auch, dass die TM M die Sprache L_1 auf die Sprache L_2 reduziert.

EE-Reduktion

Wir sagen, dass M eine Funktion $F : \Sigma^* \to \Gamma^*$ **berechnet**, falls für alle $x \in \Sigma^*$: $q_0 ¢ x \vdash_M^* q_{\text{accept}} ¢ F(x)$.

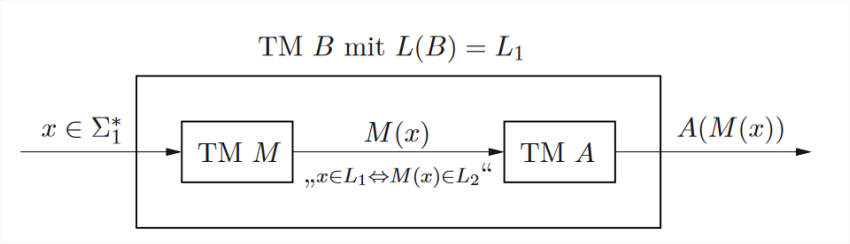


Abbildung 4: Abbildung 5.7 vom Buch

Verhältnis von EE-Reduktion und R-Reduktion

Lemma 5.3

Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ zwei Sprachen.

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

Beweis:

$$L_1 \leq_{\text{EE}} L_2 \implies \exists \text{TM } M.\ x \in L_1 \iff M(x) \in L_2$$

Wir zeigen nun $L_1 \leq_{\text{R}} L_2$, i.e. $L_2 \in \mathcal{L}_{\text{R}} \implies L_1 \in \mathcal{L}_{\text{R}}$.

Sei $L_2 \in \mathcal{L}_{\text{R}}$. Dann existiert ein Algorithmus A (TM, die immer hält), der L_2 entscheidet.

Verhältnis von EE-Reduktion und R-Reduktion

Wir konstruieren eine TM B (die immer hält) mit $L(B) = L_1$

Für eine Eingabe $x \in \Sigma_1^*$ arbeitet B wie folgt:

(i) B simuliert die Arbeit von M auf x, bis auf dem Band das Wort $M(x)$ steht.

(ii) B simuliert die Arbeit von A auf $M(x)$.
Wenn A das Wort $M(x)$ akzeptiert, dann akzeptiert B das Wort x.
Wenn A das Wort $M(x)$ verwirft, dann verwirft B das Wort x.

A hält immer $\implies$ B hält immer und somit gilt $L_1 \in \mathcal{L}_R$

■

Lemma 5.4

Sei Σ ein Alphabet. Für jede Sprache $L \subseteq \Sigma^*$ gilt:

$$L \leq_R L^{\complement} \text{ und } L^{\complement} \leq_R L$$

Beweis:

Es reicht $L^{\complement} \leq_R L$ zu zeigen, da $(L^{\complement})^{\complement} = L$ und somit dann $(L^{\complement})^{\complement} = L \leq_R L^{\complement}$.

Sei M' ein Algorithmus für L, der immer hält ($L \in \mathcal{L}_R$). Dann beschreiben wir einen Algorithmus B, der $L^{\complement}$ entscheidet.

B übernimmt die Eingaben und gibt sie an M' weiter und invertiert dann die Entscheidung von M'. Weil M' immer hält, hält auch B immer und wir haben offensichtlich $L(B) = L$.

Anwendung vom Lemma 5.4

Korollar 5.2

$$(L_{\text{diag}})^{\complement} \notin \mathcal{L}_{\text{R}}$$

Beweis:

Aus Lemma 5.4 haben wir $L_{\text{diag}} \leq_{\text{R}} (L_{\text{diag}})^{\complement}$. Daraus folgt $L_{\text{diag}} \notin \mathcal{L}_{\text{R}} \implies (L_{\text{diag}})^{\complement} \notin \mathcal{L}_{\text{R}}$. Da $L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$ gilt auch $L_{\text{diag}} \notin \mathcal{L}_{\text{R}}$.

Folglich gilt $(L_{\text{diag}})^{\complement} \notin \mathcal{L}_{\text{R}}$.

■

Beispielaufgabe 17a HS22

Beweise

$$L_H \leq_{\text{EE}} L_U$$

wobei

$$L_H = \{\text{Kod}(M)\#w \mid M \text{ hält auf } w \wedge w \in (\Sigma_{\text{bool}})^*\}$$

und

$$L_U = \{\text{Kod}(M)\#w \mid M \text{ akzeptiert } w \wedge w \in (\Sigma_{\text{bool}})^*\}$$

Beispielaufgabe 17a HS22

Wir wollen $L_H \leq_{\text{EE}} L_U$ zeigen. Wir geben die Reduktion zuerst als Zeichnung an.

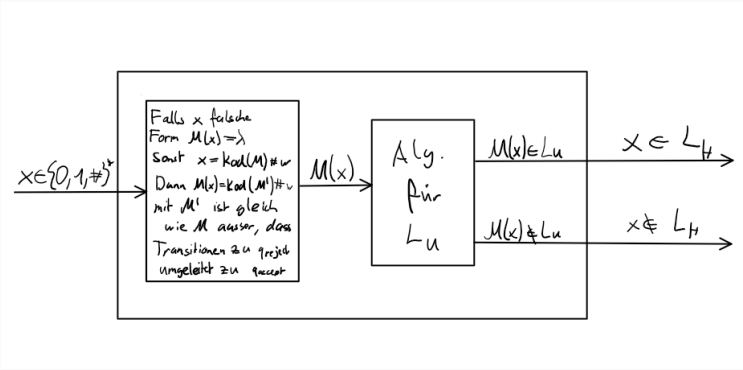


Abbildung 5: EE-Reduktion von L_H auf L_U

Beispielaufgabe 17a HS22

Wir definieren eine Funktion $M(x)$ für ein $x \in \{0, 1, \#\}^*$, so dass

$$x \in L_H \iff M(x) \in L_U \qquad (1)$$

Falls x nicht die richtige Form hat, ist $M(x) = \lambda$, sonst ist $M(x) = \text{Kod}(M')\#w$ wobei M' gleich aufgebaut ist wie M, ausser dass alle Transitionen zu q_{reject} zu q_{accept} umgeleitet werden. Wir sehen, dass M' genau dann w akzeptiert, wenn M auf w hält.

Dieses $M(x)$ übergeben wir dem Algorithmus für L_U.

Beispielaufgabe 17a HS22

Wir beweisen nun $x \in L_H \iff M(x) \in L_U$:

(i) $x \in L_H$

Dann ist $x = \text{Kod}(M)\#w$ von der richtigen Form, und M hält auf w. Das heisst die Simulation von M auf w endet entweder in q_{reject} oder in q_{accept}. Folglich wird M' w immer akzeptieren, da alle Transitionen zu q_{reject} zu q_{accept} umgeleitet wurden.

$x \in L_H \implies M(x) \in L_U$

(ii) $x \notin L_H$

Dann unterscheiden wir zwischen zwei Fällen:

(a) x hat nicht die richtige Form, i.e. $x \neq \text{Kod}(M)\#w$. Dann ist $M(x) = \lambda$ und da es keine Kodierung einer Turingmaschine M gibt, so dass $\text{Kod}(M) = \lambda$, gilt $\lambda \notin L_U$.

Beispielaufgabe 17a HS22

(i) $x \in L_H$

done above.

(ii) $x \notin L_H$

(a) **falsche Form**

done above.

(b) $x = \text{Kod}(M)\#w$ hat die richtige Form. Dann haben wir $M(x) = \text{Kod}(M')\#w$.

Da aber $x \notin L_H$, hält M nicht auf w. Da M nicht auf w hält, erreicht es nie q_{reject} oder q_{accept} in M und so wird w von M' nicht akzeptiert.

$\implies M(x) \notin L_U$

So haben wir mit diesen Fällen (a) und (b) $x \notin L_H \implies M(x) \notin L_U$ bewiesen.

Aus indirekter Implikation folgt $M(x) \in L_U \implies x \in L_H$

Beispielaufgabe 17a HS22

Aus (i) und (ii) folgt

$$x \in L_H \iff M(x) \in L_U \qquad (1)$$

Somit ist die Reduktion korrekt.

■

Beispielaufgabe 18b HS22

Sei

$$L_{\text{infinite}} = \{\text{Kod}(M) \mid M \text{ hält auf keiner Eingabe}\}$$

Zeige $(L_{\text{infinite}})^C \notin \mathcal{L}_R$

Beispielaufgabe 18b HS22

Wir zeigen, dass $(L_{\text{infinite}})^C \notin \mathcal{L}_R$ mit einer geeigneten Reduktion.

Wir beweisen $L_H \leq_R (L_{\text{infinite}})^C$

Um dies zu zeigen nehmen wir an, dass wir einen Algorithmus A haben, der $(L_{\text{infinite}})^C$ entscheidet. Wir konstruieren einen Algorithmus B, der mit Hilfe von A, die Sprache L_H entscheidet.

Beispielaufgabe 18b HS22

Wir betrachten folgende Abbildung:

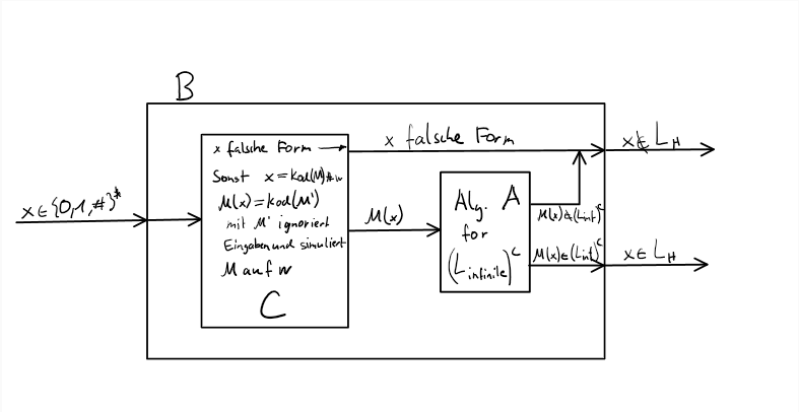


Abbildung 6: R-Reduktion von L_H auf $(L_{\text{infinite}})^C$

Beispielaufgabe 18b HS22

I. Für eine Eingabe $x \in \{0, 1, \#\}^*$ berechnet das Teilprogramm C, ob x die richtige Form hat(i.e. ob $x = \text{Kod}(M)\#w$ für eine TM M).
II. Falls nicht, verwirft B die Eingabe x.
III. Ansonsten, konstruiert C eine Turingmaschine M', die Eingaben ignoriert und immer M auf w simuliert. Wir sehen, dass M' genau dann hält, wenn M auf w hält.
IV. Folglich hält M' entweder für jede Eingabe (M hält auf w) oder für keine (M hält nicht auf w).
V. Da A genau dann akzeptiert, wenn die Eingabe keine gültige Kodierung ist(ausgeschlossen, da C das herausfiltert) oder wenn die Eingabe $M(x) = \text{Kod}(M')$ und M' für mindestens eine Eingabe hält, akzeptiert A $M(x)$ genau dann, wenn $x = \text{Kod}(M)\#w$ die richtige Form hat und M auf w hält.

Beispielaufgabe 18b HS22

Folglich gilt

$$x \in L_H \iff M(x) \in (L_{\text{infinite}})^C$$

$$\implies L_H \leq_R (L_{\text{infinite}})^C$$

Also folgt die Aussage

$$(L_{\text{infinite}})^C \in \mathcal{L}_R \implies L_H \in \mathcal{L}_R$$

Da wir $L_H \notin \mathcal{L}_R$ (**Satz 5.8**), folgt per indirekter Implikation:

$$(L_{\text{infinite}})^C \notin \mathcal{L}_R$$

■