Theoretische Informatik HS23

Nicolas Wehrli

Übungsstunde 07
10. November 2023

ETH Zürich
nwehrl@ethz.ch

Heute

1. Feedback zur Serie
2. Reduktion
3. How To Reduktion

Feedback zur Serie

Feedback

- $f : \mathcal{P}(\mathbb{Q}^+) \to [0,1]$ mit

$$f(A) = \sum_{q_i \in A} \frac{1}{2^i},$$

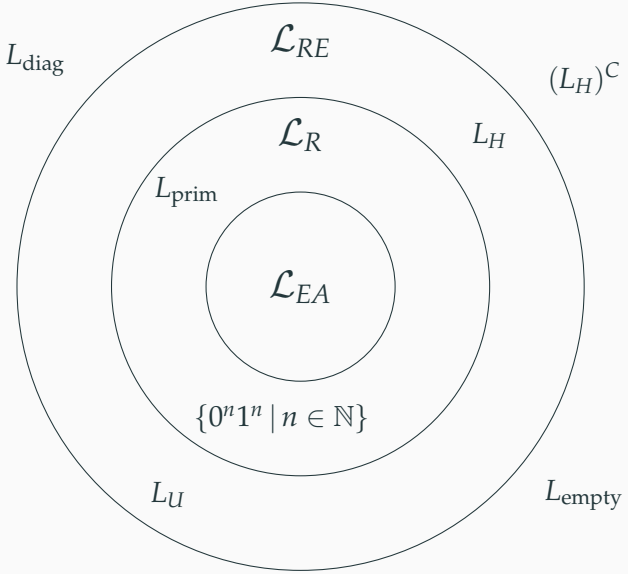
wobei q_i das i-te Element von $\mathbb{Q}^+$ ist.
Da

$$f(\{q_1\}) = \frac{1}{2} = \sum_{i=2}^{\infty} \frac{1}{2^i} = f(\mathbb{Q}^+ \setminus \{q_1\})$$

ist f nicht injektiv.

- A und B überabzählbar $\not\Longrightarrow$ Existiert Bijektion zw. A und B

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 $x \in \Sigma^*$: $q_0 ¢ x \vdash_M^* q_{\text{accept}} ¢ F(x)$.

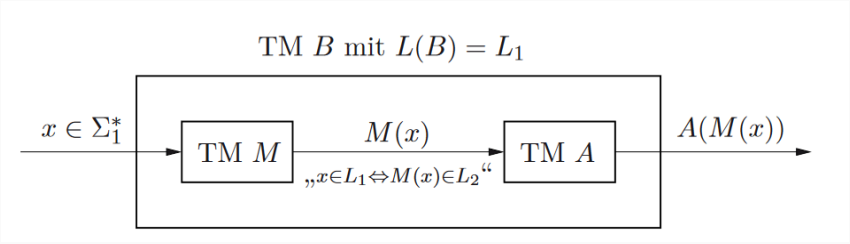


Abbildung 1: Abbildung 5.7 vom Buch

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:

$$L \leq_{\mathrm{R}} L^{\complement} \text{ und } L^{\complement} \leq_{\mathrm{R}} L$$

Beweis:

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

Sei M' ein Algorithmus für L, der immer hält ($L \in \mathcal{L}_{\mathrm{R}}$). 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.

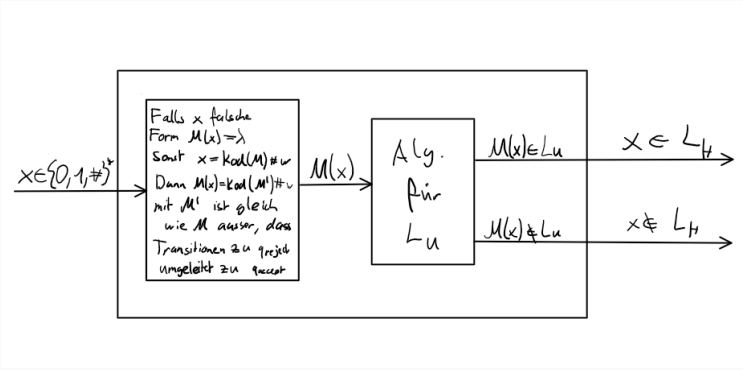


Abbildung 2: EE-Reduktion von L_H auf L_U

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:

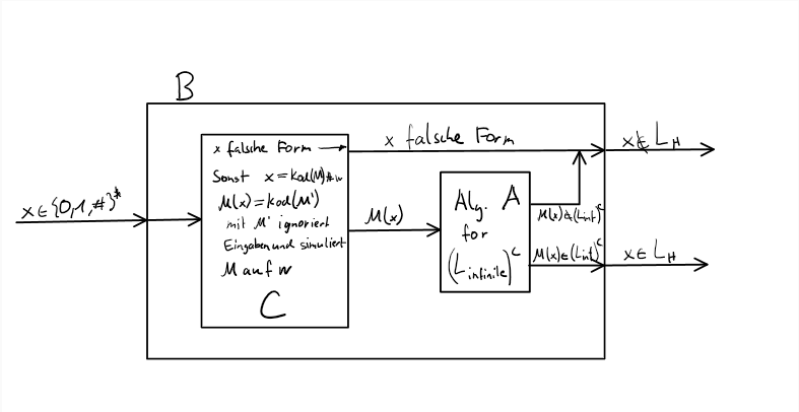


Abbildung 3: R-Reduktion von L_H auf $(L_{\text{infinite}})^C$

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:

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

■

How To Reduktion

Wir kennen zwei Methoden um dies zu beweisen:

- Wir finden eine Sprache $L' \in \mathcal{L}_{\mathrm{R}}$ und zeigen $L \leq_{\mathrm{R}} L'$. (Meistens ein wenig umständlich)
- Direkter Beweis: Eine TM (bzw. ein Algorithmus) A beschreiben, so dass $L(A) = L$ und A immer terminiert.

Wir kennen hier auch 3 Arten:

- Folgt sofort aus $L \notin \mathcal{L}_{\mathrm{RE}}$, da $\mathcal{L}_{\mathrm{R}} \subset \mathcal{L}_{\mathrm{RE}}$.
- Wir wählen eine Sprache L', so dass $L' \notin \mathcal{L}_{\mathrm{R}}$ und beweisen $L' \leq_{\mathrm{R/EE}} L$. Geeignete Sprachen als L' sind: $L^{\mathsf{C}}_{empty}, L^{\mathsf{C}}_{diag}, L_{\mathrm{H}}, L_{\mathrm{U}}, L_{\mathrm{H},\lambda}$. (Alle im Buch bewiesen)
- Satz von Rice

Anwendung von Satz von Rice

Für den **Satz von Rice**:

- Wir können mit diesem Satz nur $L \notin \mathcal{L}_R$ beweisen!
- Wir haben folgende Bedingungen:
  i. $L \subseteq \text{KodTM}$
  ii. $\exists$ TM M: $\text{Kod}(M) \in L$
  iii. $\exists$ TM M: $\text{Kod}(M) \notin L$
  iv. $\forall$ TM M_1, M_2: $L(M_1) = L(M_2) \implies (\text{Kod}(M_1) \in L \iff \text{Kod}(M_2) \in L)$

  Für den letzten Punkt (4) muss man überprüfen, ob in der Definition von $L = \{\text{Kod}(M) \mid M \text{ ist TM und } ...\}$ überall nur $L(M)$ vorkommt und nirgends M direkt. Beziehungsweise reicht es, wenn man die Bedingung so umschreiben kann, dass sie nur noch durch $L(M)$ beschrieben ist.

$L \in \mathcal{L}_{\mathbf{RE}}$

Wir beschreiben eine TM M mit $L(M) = L$, die nicht immer halten muss.

Meistens muss die TM eine Eigenschaft, für alle möglichen Wörter prüfen. (Bsp: $\text{Kod}(M_1) \in L_{\text{H}}^{\complement}$: Wir gehen alle Wörter durch, um dasjenige zu finden, für das M_1 hält.)

Wir verwenden oft einen von den folgenden 2 Tricks, um dies zu tun:

> Da es für jede NTM M', eine TM M gibt, so dass $L(M') = L(M)$, können wir eine solche definieren, für die $L(M') = L$ gilt.
> Die andere Variante, ist die parallele Simulation von Wörtern, bei dem man das Diagonalisierungsverfahren aus dem Buch verwendet. (Bsp: Beweis $L_{\text{empty}} \in \mathcal{L}_{\text{RE}}$, S. 156 Buch)

Hier haben wir 2 mögliche (offizielle) Methoden:

- Diagonalisierungsargument mit Widerspruch, wie beim Beweis von $L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$.
- Widerspruchsbeweis mit der Aussage $L \in \mathcal{L}_{\text{RE}} \wedge L^{\complement} \in \mathcal{L}_{\text{RE}} \implies L \in \mathcal{L}_{\text{R}}$.

Inoffiziell könnten wir auch die EE-Reduktion verwenden, wird aber weder in der Vorlesung noch im Buch erwähnt.

EE- und R-Reduktionen: Tipps und Tricks

- Die vorgeschaltete TM A muss immer terminieren! I.e. sie muss ein Algorithmus sein.
- Die Eingabe sollte immer zuerst auf die Richtige Form überprüft werden! Auch im Korrektsheitsbeweis, sollte dieser Fall als erstes abgehandelt werden.
- Für Korrektheit müssen wir immer $x \in L_1 \iff A(x) \in L_2$ beweisen.
- Wir verwenden meistens folgende 2 Tricks:
  i. Transitionen nach q_{accept} oder q_{reject} umleiten nach q_{reject}/q_{accept} oder einer **Endlosschleife**.
  ii. TM M' konstruieren, die ihre Eingabe ignoriert und immer dasselbe tut (z.B. eine TM dessen Kodierung gegeben ist, auf ein fixes Wort simuliern).
- Die Kodierung einer TM generieren, dessen Sprache gewisse Eigenschaften hat(z.B. sie akzeptiert alle Eingaben, läuft immer unendlich etc.)