

Theoretische Informatik HS24

Nicolas Wehrli

Übungsstunde 08

12. November 2024

ETH Zürich

nwehrli@ethz.ch

- ① Feedback
- ② Reduktion
- ③ How To Reduktion
- ④ Reduktion continued
- ⑤ Satz von Rice

Feedback

Reduktion

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 $f_M : \Sigma_1^* \rightarrow \Sigma_2^*$ 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.

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

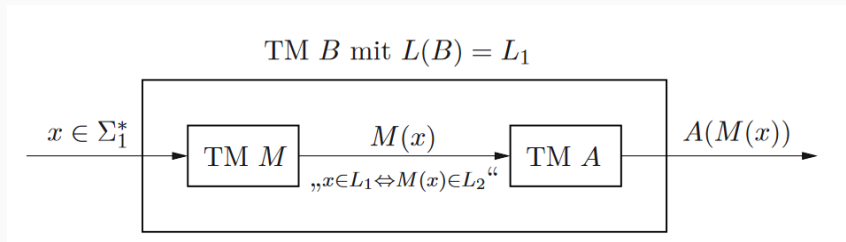


Abbildung 1: Abbildung 5.7 vom Buch

Lemma 5.3

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

$$L_1 \leq_{EE} L_2 \implies L_1 \leq_R L_2$$

Beweis:

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

Wir zeigen nun $L_1 \leq_R L_2$, i.e. $L_2 \in \mathcal{L}_R \implies L_1 \in \mathcal{L}_R$.

Sei $L_2 \in \mathcal{L}_R$. 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:

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

Beweis:

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

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

Korollar 5.2

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

Beweis:

Aus Lemma 5.4 haben wir $L_{\text{diag}} \leq_R (L_{\text{diag}})^{\mathbb{C}}$. Daraus folgt
 $L_{\text{diag}} \notin \mathcal{L}_R \implies (L_{\text{diag}})^{\mathbb{C}} \notin \mathcal{L}_R$. Da $L_{\text{diag}} \notin \mathcal{L}_{RE}$ gilt auch $L_{\text{diag}} \notin \mathcal{L}_R$.

Folglich gilt $(L_{\text{diag}})^{\mathbb{C}} \notin \mathcal{L}_R$.



Beweise

$$L_H \leq_{EE} L_U$$

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

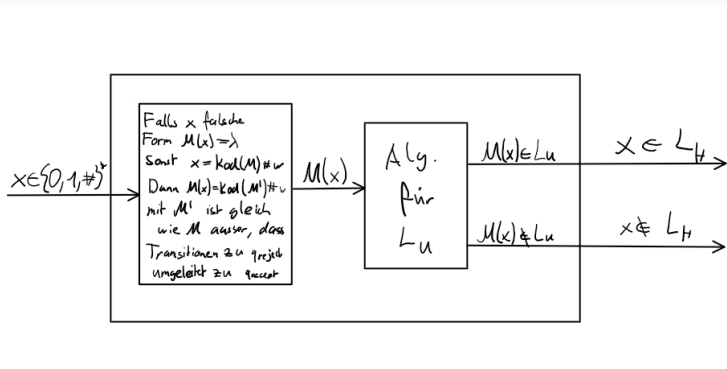


Abbildung 2: EE-Reduktion von L_H auf L_U

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

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

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

Aus (i) und (ii) folgt

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

Somit ist die Reduktion korrekt.



Sei

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

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

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:

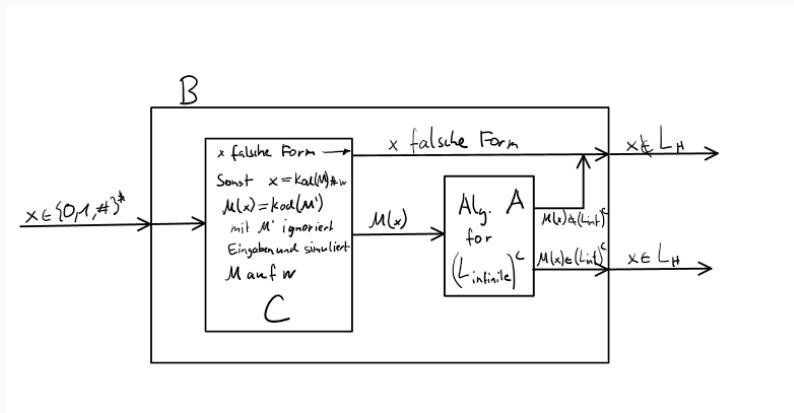


Abbildung 3: R-Reduktion von L_H auf $(L_{\text{infinite}})^C$

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.

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

$$L \in \mathcal{L}_R$$

Wir kennen zwei Methoden um dies zu beweisen:

I. Reduktion

- (a) Wir finden eine Sprache $L' \in \mathcal{L}_R$ (entweder schon in Vorlesung bewiesen oder selbst beweisen).
- (b) Zeige die Reduktion $L \leq_R L'$ (folgt trivial aus Lemma 5.4 für $L' = L^c$).

II. Direkter Beweis: TM Konstruktion

- (a) Beschreibung einer TM (bzw. ein Algorithmus) M mit $L(M) = L$. Dabei kann man eine schon bekannte TM A verwenden, die immer hält (i.e. $L(A) \in \mathcal{L}_R$).
- (b) Beweise $L(M) = L$ und dass die TM M **immer** hält.

Wir kennen hier auch 3 Arten:

- **Trivial**

Folgt sofort aus $L \notin \mathcal{L}_{RE}$, da $\mathcal{L}_R \subset \mathcal{L}_{RE}$.

- **Reduktion**

(a) Finde eine Sprache L' , so dass $L' \notin \mathcal{L}_R$ (muss bewiesen werden, falls nicht im Buch).

(b) Beweise $L' \leq_{R/EE} L$.

(c) Geeignete Sprachen als L' sind: $L_{empty}^C, L_{diag}^C, L_H, L_U, L_{H,\lambda}$. (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. $\exists \text{ TM } M: \text{Kod}(M) \in L$
 - iii. $\exists \text{ TM } M: \text{Kod}(M) \notin L$
 - iv. $\forall \text{ TM } M_1, M_2: L(M_1) = L(M_2) \implies (\text{Kod}(M_1) \in L \iff \text{Kod}(M_2) \in L)$

Für den letzten Punkt (4) muss man überprüfen, ob in der Definition von $L = \{\text{Kod}(M) \mid M \text{ ist TM und } \dots\}$ 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.

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

- I. Wir beschreiben eine TM M mit $L(M) = L$, die nicht immer halten muss.
- II. Meistens muss die TM eine Eigenschaft, für alle möglichen Wörter prüfen.
- III. Bsp: $L = \{\text{Kod}(M_1) \mid \text{Kod}(M_1) \in L_H^c\}$: Wir gehen alle Wörter durch, um dasjenige zu finden, für das M_1 hält.
- IV. 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:

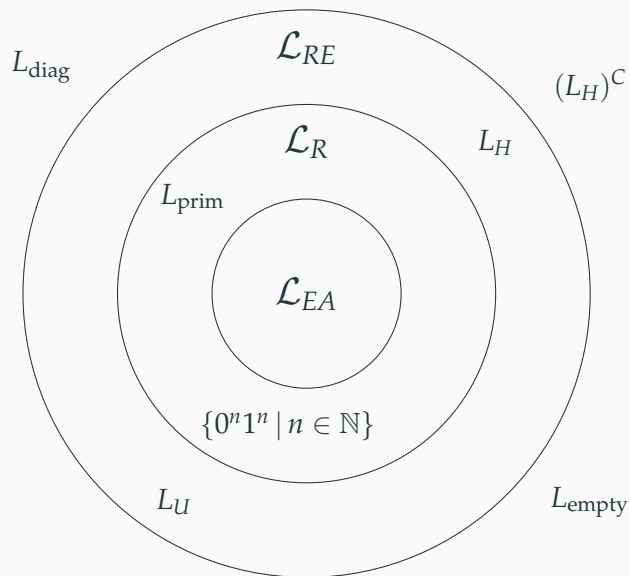
- Diagonalisierungsargument mit Widerspruch, wie beim Beweis von $L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$.
- Widerspruchsbeweis mit der Aussage $L \in \mathcal{L}_{\text{RE}} \wedge L^{\complement} \in \mathcal{L}_{\text{RE}} \implies L \in \mathcal{L}_{\text{R}}$ (Aufgabe 5.22, muss begründet werden!).

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 Korrektheitsbeweis, 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 simulieren).
- Die Kodierung einer TM generieren, dessen Sprache gewisse Eigenschaften hat(z.B. sie akzeptiert alle Eingaben, läuft immer unendlich etc.)

Klassifizierung verschiedener Sprachen



Für eine Sprache L gilt folgendes

$$L \text{ regulär} \iff L \in \mathcal{L}_{\text{EA}} \iff \exists \text{ EA } A \text{ mit } L(A) = L$$

$$L \text{ rekursiv} \iff L \in \mathcal{L}_{\text{R}} \iff \exists \text{ Alg. } A \text{ mit } L(A) = L$$

$$L \text{ rekursiv aufzählbar} \iff L \in \mathcal{L}_{\text{RE}} \iff \exists \text{ TM } M. L(M) = L$$

„Algorithmus“ = TM, die immer hält.

L rekursiv = L entscheidbar

L rekursiv aufzählbar = L erkennbar

Reduktion continued

Lemma 5.5

$$L_{\text{diag}}^{\mathbb{C}} \in \mathcal{L}_{\text{RE}}$$

Beweis

Direkter Beweis: Wir beschreiben eine TM A mit $L(A) = L_{\text{diag}}^{\mathbb{C}}$.

Eingabe: $x \in (\Sigma_{\text{bool}})^*$

- (i) Berechne i , so dass $w_i = x$ in kanonischer Ordnung
- (ii) Generiere $\text{Kod}(M_i)$.
- (iii) Simuliere die Berechnung von M_i auf $w_i = x$.

Lemma 5.5 Beweis continued

- Falls $w_i \in L(M_i)$ akzeptiert, akzeptiert A die Eingabe x .
- Falls M_i verwirft (hält in q_{reject}), dann hält A und verwirft $x = w_i$ auch.
- Falls M_i unendlich lange arbeitet, wird A auch nicht halten und dann folgt auch $x \notin L(A)$.

Aus dem folgt $L(A) = L_{\text{diag}}^c$.



Korollar 5.3

$L_{\text{diag}}^c \in \mathcal{L}_{\text{RE}} \setminus \mathcal{L}_{\text{R}}$ und daher $\mathcal{L}_{\text{R}} \subsetneq \mathcal{L}_{\text{RE}}$.

Sei

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

Satz 5.6

Es gibt eine TM U , **universelle TM** genannt, so dass

$$L(U) = L_U$$

Daher gilt $L_U \in \mathcal{L}_{\text{RE}}$.

Beweis

Direkter Beweis: Konstruktion einer TM.

Siehe Buch/Vorlesung.

Satz 5.7

$$L_U \notin \mathcal{L}_R$$

Beweis

Wir zeigen $L_{\text{diag}}^{\mathbb{C}} \leq_R L_U$.

Siehe Buch/Vorlesung.

Sei

$$L_H = \{\text{Kod}(M)\#w \mid M \text{ hält auf } w\}$$

Satz 5.8

$$L_H \notin \mathcal{L}_R$$

Beweis

Wir zeigen $L_U \leq_R L_H$.

Siehe Buch/Vorlesung.

Sei

$$L_{\text{empty}} = \{\text{Kod}(M) \mid L(M) = \emptyset\}$$

und

$$L_{\text{empty}}^{\mathbb{C}} = \{\text{Kod}(M) \mid L(M) \neq \emptyset\} \cup \{x \in \{0, 1, \#\} \mid x \notin \mathbf{KodTM}\}$$

Lemma 5.6

$$L_{\text{empty}}^{\mathbb{C}} \in \mathcal{L}_{\text{RE}}$$

Beweis

Da für jede NTM M_1 eine TM M_2 existiert mit $L(M_1) = L(M_2)$, reicht es eine NTM M_1 mit $L(M_1) = L_{\text{empty}}^c$ zu finden. Eingabe: $x \in \{0, 1, \#\}$

- (i) M_1 prüft deterministisch, ob $x = \text{Kod}(M)$ für eine TM M .
Falls x keine TM kodiert, wird x akzeptiert.
- (ii) Sonst gilt $x = \text{Kod}(M)$ für eine TM M und M_1 wählt nichtdeterministisch ein Wort $y \in (\Sigma_{\text{bool}})^*$.
- (iii) Dann simuliert M_1 die TM M auf y deterministisch und übernimmt die Ausgabe.

Wir unterscheiden zwischen 3 Fällen

I $x = \text{Kod}(M)$ und $L(M) = \emptyset$

Dann gilt $x \notin L_{\text{empty}}^{\mathbb{C}}$ und da es keine akzeptierende Berechnung gibt, auch $x \notin L(M_1)$.

II $x = \text{Kod}(M)$ und $L(M) \neq \emptyset$

Dann gilt $x \in L_{\text{empty}}^{\mathbb{C}}$ und da es eine akzeptierende Berechnung gibt, auch $x \in L(M_1)$.

III x kodiert keine TM

Wir haben $x \in L_{\text{empty}}^{\mathbb{C}}$ und wegen Schritt (i) auch $x \in L(M_1)$.

Somit gilt $L(M_1) = L_{\text{empty}}^{\mathbb{C}}$.



Alternativer Beweis

Wir konstruieren eine TM A mit $L(A) = L$ direkt. Eingabe: $x \in \{0, 1, \#\}$

- I. Falls x keine Kodierung einer TM ist, akzeptiert A die Eingabe.
- II. Falls $x = \text{Kod}(M)$ für eine TM M , arbeitet A wie folgt
 - Generiert systematisch alle Paare $(i, j) \in (\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$.
(Abzählbarkeit)
 - Für jedes Paar (i, j) , generiert A das kanonisch i -te Wort w_i und simuliert j Berechnungsschritte der TM M auf w_i .
 - Falls M an ein Wort akzeptiert, akzeptiert A das Wort x .

Falls $L(M) \neq \emptyset$ existiert ein $y \in L(M)$. Dann ist $y = w_k$ für ein $k \in \mathbb{N} \setminus \{0\}$ und die akzeptierende Berechnung von M auf y hat eine endliche Länge l .

Das Paar (k, l) wird in endlich vielen Schritten erreicht und somit akzeptiert A die Eingabe x , falls $L(M) \neq \emptyset$.

Somit folgt $L(A) = L_{\text{diag}}^c$.



Aufgabe 5.22

Wir zeigen

$$L \in \mathcal{L}_{\text{RE}} \wedge L^{\complement} \in \mathcal{L}_{\text{RE}} \iff L \in \mathcal{L}_{\text{R}}$$

(\implies) :

Nehmen wir $L \in \mathcal{L}_{\text{RE}} \wedge L^{\complement} \in \mathcal{L}_{\text{RE}}$ an.

Dann existiert eine TM M und M_C mit $L(M) = L$ und $L(M_C) = L^{\complement}$.

Wir konstruieren eine TM A , die für eine Eingabe w die beiden TM's M und M_C parallel auf w simuliert.

A akzeptiert w , falls M das Wort akzeptiert und verwirft, falls M_C das Wort akzeptiert.

Aufgabe 5.22

Bemerke, dass $L(M) \cap L(M_C) = \emptyset$ und $L(M) \cup L(M_C) = \Sigma^*$.

Da $w \in L(M)$ oder $w \in L(M_C)$, hält A immer.

Da A genau dann akzeptiert, falls $w \in L(M)$, folgt $L(A) = L(M) = L$.

Demnach gilt $L \in \mathcal{L}_R$.

(\Leftarrow) :

Nehmen wir $L \in \mathcal{L}_R$ an. Per Lemma 5.4 gilt $L^c \leq_R L$ und daraus folgt auch $L^c \in \mathcal{L}_R$.

Da $\mathcal{L}_R \subset \mathcal{L}_{RE}$, folgt $L \in \mathcal{L}_{RE} \wedge L^c \in \mathcal{L}_{RE}$.



Lemma 5.7

$$L_{\text{empty}}^{\mathbb{C}} \notin \mathcal{L}_R$$

Beweis

Wir zeigen $L_U \leq_{EE} L_{\text{empty}}^{\mathbb{C}}$.

Siehe Buch/Vorlesung.

Korollar 5.4

$$L_{\text{empty}} \notin \mathcal{L}_R$$

Korollar 5.5

$$L_{\text{EQ}} \notin \mathcal{L}_R$$

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

Satz von Rice

Wir definieren $L_{H,\lambda} = \{\text{Kod}(M) \mid M \text{ hält auf } \lambda\}$.

Lemma 5.8

$$L_{H,\lambda} \notin \mathcal{L}_R$$

Beweis:

Wir zeigen $L_H \leq_{EE} L_{H,\lambda}$. Wir beschreiben einen Algorithmus B , so dass $x \in L_H \iff B(x) \in L_{H,\lambda}$.

Lemma 5.8 - Beweis continued

Für jede Eingabe arbeitet B wie folgt:

Falls x von der falschen Form, dann $B(x) = M_{inf}$, wobei M_{inf} unabhängig von der Eingabe immer unendlich läuft.

Sonst $x = \text{Kod}(M)\#w$: Dann $B(x) = M'$, wobei M' die Eingabe ignoriert und immer M auf w simuliert.

Wir sehen, dass M' genau dann auf λ hält, wenn $x \in L_H$.

Daraus folgt $x \in L_H \iff B(x) \in L_{H,\lambda}$.



Satz 5.9

Jedes semantisch nichttriviale Entscheidungsproblem über Turingmaschinen ist unentscheidbar.

'über Turingmaschinen' = $L \subseteq \mathbf{KodTM}$.

'nichttrivial' = $\exists M_1 : \mathbf{Kod}(M_1) \in L$ und $\exists M_2 : \mathbf{Kod}(M_2) \notin L$

'semantisch' = Für A, B mit $L(A) = L(B)$ gilt $\mathbf{Kod}(A) \in L \iff \mathbf{Kod}(B) \in L$.

Wir definieren

$$L_{\text{all}} = \{\text{Kod}(M) \mid L(M) = (\Sigma_{\text{bool}})^*\}.$$

Zeige

$$L_{\text{all}} \notin \mathcal{L}_{\text{R}}.$$