

Abbildung 5.9

Korollar 5.2. $(L_{\text{diag}})^c \notin \mathcal{L}_R$.

Beweis. Wir haben gezeigt, dass $L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$ und damit auch $L_{\text{diag}} \notin \mathcal{L}_R$. Nach Lemma 5.4 ist $L_{\text{diag}} \leq_R (L_{\text{diag}})^c$. Weil nach Lemma 5.4 $(L_{\text{diag}})^c \in \mathcal{L}_R$ auch $L_{\text{diag}} \in \mathcal{L}_R$ implizieren würde, können wir schließen, dass $(L_{\text{diag}})^c \notin \mathcal{L}_R$. \square

Aus den bisher präsentierten Tatsachen können wir aber nicht schließen, dass $(L_{\text{diag}})^c \notin \mathcal{L}_{\text{RE}}$. Das Gegenteil ist wahr, und somit beweisen wir $\mathcal{L}_R \subsetneq \mathcal{L}_{\text{RE}}$.

Lemma 5.5. $(L_{\text{diag}})^c \in \mathcal{L}_{\text{RE}}$.

Beweis. Nach der Definition von L_{diag} erhalten wir

$$(L_{\text{diag}})^c = \{x \in (\Sigma_{\text{bool}})^* \mid x = w_i \text{ für ein } i \in \mathbb{N} - \{0\} \text{ und } M_i \text{ akzeptiert das Wort } w_i\}.$$

Eine TM D , die $(L_{\text{diag}})^c$ akzeptiert, kann wie folgt arbeiten.

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

- (i) Berechne i , so dass x das i -te Wort w_i in der kanonischen Ordnung über Σ_{bool} ist.
- (ii) Generiere die Kodierung $\text{Kod}(M_i)$ der i -ten TM M_i .
- (iii) Simuliere die Berechnung von M_i auf dem Wort $w_i = x$.
 Falls M_i das Wort w_i akzeptiert, dann akzeptiert auch D die Eingabe x .
 Falls M_i das Wort w_i verwirft (d. h. in q_{reject} hält), dann hält D und verwirft $x = w_i$ auch.
 Falls M_i auf w_i unendlich lange arbeitet (d. h. $w_i \notin L(M_i)$), simuliert D die unendliche Arbeit von M_i auf w_i . Daher hält D auf x nicht und somit $x \notin L(D)$.

Es ist offensichtlich, dass $L(D) = (L_{\text{diag}})^c$. \square

Korollar 5.3. $(L_{\text{diag}})^c \in \mathcal{L}_{\text{RE}} - \mathcal{L}_R$, und daher $\mathcal{L}_R \subsetneq \mathcal{L}_{\text{RE}}$.

Im Folgenden präsentieren wir weitere Sprachen, die nicht rekursiv sind, aber in \mathcal{L}_{RE} liegen (Abbildung 5.10).

Definition 5.5. Die *universelle Sprache* ist die Sprache

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

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. Es reicht aus, eine 1-Band-TM U mit $L(U) = L_U$ zu konstruieren. U kann wie folgt auf einer Eingabe $z \in \{0, 1, \#\}^*$ arbeiten.

- (i) U überprüft, ob z genau ein $\#$ enthält. Falls nicht, verwirft U das Wort z .
- (ii) Sei $z = y\#x$ mit $y, x \in (\Sigma_{\text{bool}})^*$. U überprüft, ob y eine Kodierung einer TM ist. Falls y keine TM kodiert, verwirft U die Eingabe $y\#x$.
- (iii) Falls $y = \text{Kod}(M)$ für eine TM M , schreibt U die Anfangskonfiguration von M auf x auf das Arbeitsband.
- (iv) U simuliert schrittweise die Berechnung von M auf x wie folgt.
while der Zustand der Konfigurationen von M auf dem Arbeitsband ist unterschiedlich von q_{accept} und q_{reject} **do**
 begin
 Simuliere einen Schritt von M auf dem Arbeitsband;
 {Die Aktion, die zu simulieren ist, kann U aus der Kodierung $\text{Kod}(M)$ auf dem Arbeitsband leicht ablesen.}
 end;
 if der Zustand von M ist q_{accept} **then**
 U akzeptiert $z = \text{Kod}(M)\#x$;
 else
 U verwirft $z = \text{Kod}(M)\#x$;

Man bemerke, dass eine unendliche Berechnung von M auf x zu einer unendlichen Berechnung von U auf $\text{Kod}(M)\#x$ führt und U deshalb in diesem Fall die Eingabe $\text{Kod}(M)\#x$ nicht akzeptiert. Somit ist $L(U) = L_U$. \square

Was bedeutet eigentlich die Tatsache, dass $L_U \in \mathcal{L}_{\text{RE}}$? Dies bedeutet die Existenz einer TM (eines Programms) ohne Haltegarantie, die eine beliebige Turingmaschine auf einer gegebenen Eingabe simulieren kann. Dies ist aber eine ganz natürliche Forderung, die man als ein Axiom an jede formale Definition einer allgemeinen Algorithmenklasse stellt. Die Betriebssysteme mit ihren Interpretern erfüllen gerade diese Aufgabe für die Programmiersprachen als Berechnungsmodelle. Es wäre unzulässig, einen Interpreter (ein Betriebssystem) zu erstellen, der unfähig wäre, jedes syntaktisch korrekte Programm in gegebener Programmiersprache auf einer Eingabe laufen zu lassen.

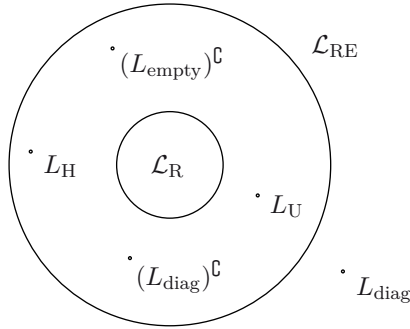


Abbildung 5.10

Im Folgenden beweisen wir, dass $L_U \notin \mathcal{L}_R$. Die Bedeutung dieser Behauptung ist, dass man im Allgemeinen nicht das Resultat der Berechnung einer TM M auf einer Eingabe x anders berechnen kann, als die Berechnung von M auf x zu simulieren. Wenn aber M auf x unendlich lange arbeitet, wissen wir zu keinem Zeitpunkt der Simulation, ob die Berechnung von M auf x unendlich ist oder ob M in den nächsten Schritten halten und entscheiden wird. Deswegen können wir in endlicher Zeit keine Entscheidung über die Zugehörigkeit von x zu $L(M)$ treffen.

Die folgenden Beweise basieren auf der Reduktionsmethode. Die meisten Beweise präsentieren wir zweimal. Einmal anschaulich auf der Ebene von Algorithmen als Programme in irgendeiner Programmiersprache und das zweite Mal im Formalismus der Turingmaschinen und der EE-Reduktion.

Satz 5.7. $L_U \notin \mathcal{L}_R$.

Beweis. Es reicht zu zeigen, dass $(L_{\text{diag}})^{\mathbb{C}} \leq_R L_U$. Nach Korollar 5.2 gilt $(L_{\text{diag}})^{\mathbb{C}} \notin \mathcal{L}_R$ und damit folgt $L_U \notin \mathcal{L}_R$.

Sei A ein Algorithmus (Programm), der L_U entscheidet. Wir bauen einen Algorithmus B , der mit A als Teilprogramm die Sprache $(L_{\text{diag}})^{\mathbb{C}}$ entscheidet. Algorithmus B ist so strukturiert wie in Abbildung 5.11 dargestellt. Für eine Eingabe $x \in (\Sigma_{\text{bool}})^*$ berechnet das Teilprogramm C ein i , so dass $x = w_i$, und die Kodierung $\text{Kod}(M_i)$. Die Wörter $\text{Kod}(M_i)$ und $x = w_i$ bekommt A als Eingabe. Die Entscheidung „akzeptiere“ oder „verwerfe“ von A für $\text{Kod}(M_i)\#x$ wird als das Resultat von B für die Eingabe x übernommen. Offensichtlich gilt $L(B) = (L_{\text{diag}})^{\mathbb{C}}$ und B hält immer, weil A nach der Voraussetzung immer hält und seine Entscheidung liefert.

Jetzt beweisen wir die Aussage im Formalismus der Turingmaschinen. Es reicht zu beweisen, dass $(L_{\text{diag}})^{\mathbb{C}} \leq_{\text{EE}} L_U$. Wir beschreiben eine TM M , die eine Abbildung f_M von $(\Sigma_{\text{bool}})^*$ nach $\{0, 1, \#\}^*$ berechnet, so dass

$$x \in (L_{\text{diag}})^{\mathbb{C}} \iff f_M(x) \in L_U.$$

M arbeitet wie folgt. Für eine Eingabe x berechnet M zuerst ein i , so dass $x = w_i$. Danach berechnet M die Kodierung $\text{Kod}(M_i)$ der i -ten TM. M hält mit dem Inhalt $\text{Kod}(M_i)\#x$

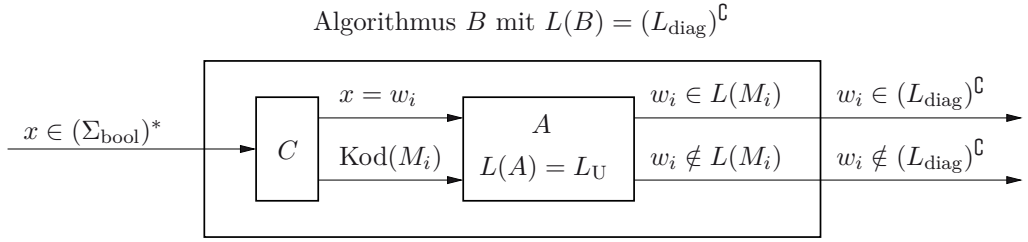


Abbildung 5.11

auf dem Band. Weil $x = w_i$, ist es nach der Definition von $(L_{\text{diag}})^{\mathbb{C}}$ offensichtlich, dass

$$\begin{aligned}
 x = w_i \in (L_{\text{diag}})^{\mathbb{C}} &\iff M_i \text{ akzeptiert } w_i \\
 &\iff w_i \in L(M_i) \\
 &\iff \text{Kod}(M_i)\#x \in L_U.
 \end{aligned}$$

□

Aufgabe 5.16. Zeigen Sie, dass die folgende Sprache

$$\{\text{Kod}(M)\#x\#0^i \mid x \in \{0,1\}^*, i \in \mathbb{N}, M \text{ hat mindestens } i+1 \text{ Zustände und} \\
 \text{während der Berechnung von } M \text{ auf } x \text{ wird der } i\text{-te Zustand} \\
 \text{von } M \text{ mindestens einmal erreicht}\}$$

keine rekursive Sprache ist.

Wir sehen, dass eines der zentralen Probleme in der Theorie der Berechenbarkeit stark mit dem Halten der Turingmaschinen (mit der Endlichkeit der Berechnungen) zusammenhängt. Für die Sprachen $(L_{\text{diag}})^{\mathbb{C}}$ und L_U gibt es Turingmaschinen (Programme), die diese Sprachen akzeptieren, aber es gibt keine Turingmaschinen, die $(L_{\text{diag}})^{\mathbb{C}}$ und L_U entscheiden (d. h. keine unendlichen Berechnungen machen). Deswegen betrachten wir jetzt das folgende Problem.

Definition 5.6. Das **Halteproblem** ist das Entscheidungsproblem $(\{0,1,\#\}, L_H)$, wobei

$$L_H = \{\text{Kod}(M)\#x \mid x \in \{0,1\}^* \text{ und } M \text{ hält auf } x\}.$$

Aufgabe 5.17. Beweisen Sie, dass $L_H \in \mathcal{L}_{\text{RE}}$ gilt.

Das folgende Resultat besagt, dass es keinen Algorithmus gibt, der testen kann, ob ein gegebenes Programm immer terminiert.

Satz 5.8. $L_H \notin \mathcal{L}_R$.

Beweis. Wir führen zuerst einen Beweis auf der Ebene von Programmen. Wir zeigen $L_U \leq_R L_H$. Sei $L_H \in \mathcal{L}_R$, d. h., es existiert ein Algorithmus A , der L_H akzeptiert. Wir beschreiben jetzt einen Algorithmus B , der mit A als Teilprogramm die Sprache L_U

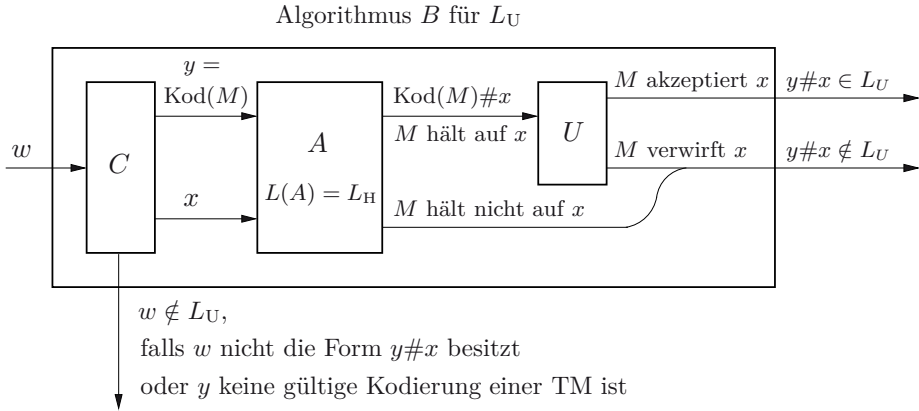


Abbildung 5.12

entscheidet (Abbildung 5.12). B bekommt eine Eingabe w und benutzt ein Teilprogramm C zum Testen, ob die Eingabe die Form $y\#x$ mit $y = \text{Kod}(M)$ für eine TM M hat.

Falls nicht, dann verwirft B die Eingabe.

Falls $w = \text{Kod}(M)\#x$, dann gibt B dem Teilprogramm A diese Eingabe.

Falls A „ M hält nicht auf x “ antwortet, dann weiß B , dass $x \notin L(M)$ und verwirft die Eingabe $\text{Kod}(M)\#x$.

Falls A „ M hält auf x “ antwortet, dann simuliert B im Teilprogramm U die Arbeit von M auf x . Weil M auf x eine endliche Berechnung hat, wird die Simulation von U in endlicher Zeit durchgeführt.

Falls die Ausgabe von U „ M akzeptiert x “ ist, dann akzeptiert B seine Eingabe $y\#x = \text{Kod}(M)\#x$. Falls die Ausgabe von U „ M verwirft x “ ist, dann verwirft B die Eingabe $\text{Kod}(M)\#x$. Offensichtlich gilt $L(B) = L_U$ und B hält immer. Daher erhalten wir $L_U \in \mathcal{L}_R$ und somit gilt $L_U \leq_R L_H$.

Wir beweisen jetzt $L_U \leq_R L_H$ in dem Formalismus der Turingmaschinen. Es reicht zu zeigen, dass $L_U \leq_{EE} L_H$. Wir beschreiben eine TM M , die L_U auf L_H reduziert. Für eine Eingabe w arbeitet M wie folgt.

M überprüft, ob die Eingabe die Form $w = \text{Kod}(\overline{M})\#x$ für eine TM \overline{M} und ein $x \in (\Sigma_{\text{bool}})^*$ hat.

- (i) Falls w diese Form nicht hat, generiert M die Kodierung $\text{Kod}(M_1)$ einer TM M_1 , die für jede Eingabe in einer endlosen Schleife im Zustand q_0 läuft ($\delta(q_0, a) = (q_0, a, N)$ für alle $a \in \{0, 1\}$). Dann hält M mit dem Bandinhalt $M(w) = \text{Kod}(M_1)\#x$.
- (ii) Falls $w = \text{Kod}(\overline{M})\#x$, dann modifiziert M die Kodierung der TM \overline{M} zu folgender TM M_2 mit $L(M_2) = L(\overline{M})$. M_2 arbeitet genau wie \overline{M} , nur dass alle Transitionen zum Zustand q_{reject} von \overline{M} zu einem neuen Zustand p umgeleitet werden, in dem M_2 für jede Eingabe in einer endlosen Schleife läuft. Daher führt M_2 für jede Eingabe $y \notin L(\overline{M}) = L(M_2)$ eine unendliche Berechnung durch. M beendet seine Arbeit mit dem Bandinhalt $M(w) = \text{Kod}(M_2)\#x$.

Wir beweisen jetzt für alle $w \in \{0, 1, \#\}^*$, dass

$$w \in L_U \iff M(w) \in L_H.$$

Sei $w \in L_U$. Daher ist $w = \text{Kod}(\overline{M})\#x$ für eine TM \overline{M} und ein Wort $x \in \{0, 1\}^*$ und $x \in L(\overline{M})$. Dann ist $M(w) = \text{Kod}(M_2)\#x$ mit $L(M_2) = L(\overline{M})$ ein Wort in L_H .

Sei $w \notin L_U$. Wir unterscheiden zwei Möglichkeiten. Wenn w nicht die Form $\text{Kod}(\overline{M})\#x$ für eine TM \overline{M} hat, dann ist $M(w) = \text{Kod}(M_1)\#x$, wobei M_1 auf keiner Eingabe hält. Daher ist $M(w)$ nicht in L_H . Wenn w die Form $\text{Kod}(\overline{M})\#x$ für eine TM \overline{M} hat und $\text{Kod}(\overline{M})\#x \notin L_U$, bedeutet das, dass $x \notin L(\overline{M})$. In diesem Fall ist aber $M(w) = \text{Kod}(M_2)\#x$, wobei M_2 auf keiner Eingabe aus $(\Sigma_{\text{bool}})^* - L(\overline{M})$ hält. Weil $x \notin L(\overline{M})$, gilt $\text{Kod}(\overline{M})\#x \notin L_H$. \square

Aufgabe 5.18. Beweisen Sie die folgenden Aussagen:

- (a) $L_U \leq_R (L_{\text{diag}})^G$,
- (b) $L_H \leq_R (L_{\text{diag}})^G$,
- (c) $L_{\text{diag}} \leq_R L_U$,
- (d) $(L_{\text{diag}})^G \leq_R L_H$,
- (e) $L_H \leq_R L_U$.

Betrachten wir jetzt die Sprache

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

die die Kodierungen aller Turingmaschinen enthält, die die leere Menge (kein Wort) akzeptieren. Offensichtlich ist

$$(L_{\text{empty}})^G = \{x \in (\Sigma_{\text{bool}})^* \mid x \neq \text{Kod}(\overline{M}) \text{ für alle TM } \overline{M} \text{ oder } x = \text{Kod}(M) \text{ und } L(M) \neq \emptyset\}.$$

Lemma 5.6. $(L_{\text{empty}})^G \in \mathcal{L}_{\text{RE}}$.

Beweis. Wir führen zwei unterschiedliche Beweise für $(L_{\text{empty}})^G \in \mathcal{L}_{\text{RE}}$. Im ersten Beweis zeigen wir, dass es nützlich ist, das Modell der nichtdeterministischen Turingmaschinen zur Verfügung zu haben. Der zweite Beweis zeigt, wie man die Ideen aus der Mengenlehre, speziell aus dem Beweis $|\mathbb{N}| = |\mathbb{Q}^+|$, in der Theorie der Berechenbarkeit anwenden kann.

Weil für jede NTM M_1 eine TM M_2 existiert, so dass $L(M_1) = L(M_2)$, reicht es zu zeigen, dass eine NTM M_1 mit $L(M_1) = (L_{\text{empty}})^G$ existiert. M_1 arbeitet auf einer Eingabe x wie folgt.

- (i) M_1 überprüft deterministisch, ob $x = \text{Kod}(M)$ für eine TM M .
Falls x keine TM kodiert, akzeptiert M_1 das Wort x .
- (ii) Falls $x = \text{Kod}(M)$ für eine TM M , wählt M_1 nichtdeterministisch ein Wort $y \in (\Sigma_{\text{bool}})^*$ und simuliert deterministisch die Berechnung von M auf y .

- (iii) Falls M das Wort y akzeptiert (das heißt $L(M) \neq \emptyset$), so akzeptiert M_1 die Eingabe $x = \text{Kod}(M)$.

Falls M das Wort y verwirft, akzeptiert M_1 in diesem Lauf die Eingabe x nicht.

Falls die Berechnung von M auf y unendlich ist, rechnet M_1 auch in diesem Lauf auf x unendlich lange und akzeptiert so dass Wort $x = \text{Kod}(M)$ in diesem Lauf nicht.

Gemäß Schritt (i) akzeptiert M_1 alle Wörter, die keine TM kodieren.

Falls $x = \text{Kod}(M)$ und $L(M) \neq \emptyset$, dann existiert ein Wort y mit $y \in L(M)$. Deswegen existiert eine Berechnung von M_1 auf $x = \text{Kod}(M)$, in der x akzeptiert wird.

Falls $x \in L_{\text{empty}}$, dann existiert keine akzeptierende Berechnung von M_1 auf x , und so schließen wir, dass $L(M_1) = (L_{\text{empty}})^{\mathbb{C}}$.

Im zweiten Beweis von $(L_{\text{empty}})^{\mathbb{C}} \in \mathcal{L}_{\text{RE}}$ konstruieren wir direkt eine (deterministische) TM A , die $(L_{\text{empty}})^{\mathbb{C}}$ akzeptiert. A arbeitet auf einer Eingabe w wie folgt.

- (i) Falls w keine Kodierung einer Turingmaschine ist, so akzeptiert A die Eingabe w .

- (ii) Falls $w = \text{Kod}(M)$ für eine TM M , arbeitet A wie folgt.

A konstruiert systematisch (in der Reihenfolge aus Abbildung 5.3 oder Abbildung 5.4) alle Paare $(i, j) \in (\mathbb{N} - \{0\}) \times (\mathbb{N} - \{0\})$. Für jedes Paar (i, j) generiert A das i -te Wort w_i über dem Eingabealphabet der TM M und simuliert j Berechnungsschritte von M auf w_i .

Falls für ein (k, l) die TM M das Wort w_k in l Schritten akzeptiert, hält A und akzeptiert ihre Eingabe $w = \text{Kod}(M)$. Sonst arbeitet A unendlich lange und akzeptiert damit die Eingabe w nicht.

Der Kernpunkt der Beweisidee ist, dass, wenn ein Wort y mit $y \in L(M)$ existiert, dann ist $y = w_k$ für ein $k \in \mathbb{N} - \{0\}$ und die akzeptierende Berechnung von M auf y hat eine endliche Länge l . Damit garantiert die systematische Überprüfung aller Paare (i, j) in Phase (ii) der Arbeit von A das Akzeptieren von $\text{Kod}(M)$, falls $L(M) \neq \emptyset$. Somit ist $L(A) = (L_{\text{empty}})^{\mathbb{C}}$. \square

Im Folgenden zeigen wir, dass $(L_{\text{empty}})^{\mathbb{C}} \notin \mathcal{L}_{\text{R}}$. Dies entspricht der Nichtexistenz eines Algorithmus zur Überprüfung, ob ein gegebenes Programm die leere Menge akzeptiert. Daher können wir im Allgemeinen nicht die Korrektheit von Programmen testen. Dies geht sogar in den Fällen nicht, in denen es sich um triviale Aufgabenstellungen handelt (zum Beispiel eine konstante Funktion zu berechnen).

Lemma 5.7. $(L_{\text{empty}})^{\mathbb{C}} \notin \mathcal{L}_{\text{R}}$.

Beweis. Wir zeigen $L_{\text{U}} \leq_{\text{EE}} (L_{\text{empty}})^{\mathbb{C}}$. Wir beschreiben eine TM A , die L_{U} auf $(L_{\text{empty}})^{\mathbb{C}}$ reduziert (Abbildung 5.13). Für jede Eingabe $x \in \{0, 1, \#\}^*$ arbeitet A wie folgt.

- (i) Falls x nicht die Form $\text{Kod}(M)\#w$ für eine TM M und ein $w \in (\Sigma_{\text{bool}})^*$ hat, dann schreibt A das Wort $A(x) = \text{Kod}(B_x)$ auf das Band, wobei B_x eine TM ist, die über Σ_{bool} arbeitet und die leere Menge \emptyset akzeptiert ($L(B_x) = \emptyset$).

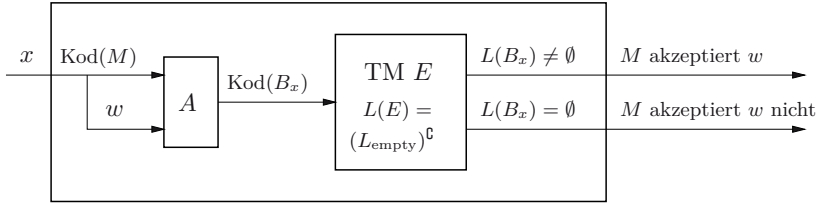


Abbildung 5.13

- (ii) Falls $x = \text{Kod}(M)\#w$ für eine TM M und ein Wort $w \in (\Sigma_{\text{bool}})^*$, dann generiert A die Kodierung $\text{Kod}(B_x)$ einer TM B_x als ihre Ausgabe. Die TM B_x arbeitet für jede Eingabe y (unabhängig vom Inhalt der eigenen Eingabe y) wie folgt.
- (a) B_x löscht y und generiert $x = \text{Kod}(M)\#w$ auf dem Band.
 {Das Wort x kann durch die Zustände und die δ -Funktion von B_x beschrieben werden.}
 - (b) B_x simuliert die Arbeit von M auf w .
 Falls M das Wort w akzeptiert, akzeptiert B_x ihre Eingabe y .
 Falls M das Wort w verwirft, verwirft B_x ihre Eingabe y .
 Falls M auf w nicht hält, dann arbeitet auch B_x unendlich lange. Folglich akzeptiert B_x ihre Eingabe y nicht.

Wir beweisen jetzt, dass

$$x \in L_U \iff A(x) = \text{Kod}(B_x) \in (L_{\text{empty}})^G$$

für alle $x \in \{0, 1, \#\}^*$.

Sei $x \in L_U$. Daher gilt $x = \text{Kod}(M)\#w$ für eine TM M und $w \in L(M)$. In diesem Fall gilt $L(B_x) = (\Sigma_{\text{bool}})^* \neq \emptyset$ und somit $\text{Kod}(B_x) \in (L_{\text{empty}})^G$.

Sei $x \notin L_U$. Dann hat x entweder nicht die Form $\text{Kod}(M')\#z$ für eine TM M' und ein $z \in \{0, 1\}^*$ oder $x = \text{Kod}(M)\#w$ für eine TM M und $w \notin L(M)$. In beiden Fällen gilt $L(B_x) = \emptyset$ und somit $\text{Kod}(B_x) \notin (L_{\text{empty}})^G$. \square

Korollar 5.4. $L_{\text{empty}} \notin \mathcal{L}_R$.

Beweis. Wenn $L_{\text{empty}} \in \mathcal{L}_R$ wäre, müsste nach Lemma 5.4 auch $(L_{\text{empty}})^G \in \mathcal{L}_R$ sein. \square

Aufgabe 5.19.* Beweisen Sie, dass die folgenden Sprachen nicht in \mathcal{L}_{RE} sind:

- (a) L_{empty} ,
- (b) $(L_H)^G$,
- (c) $(L_U)^G$.

Die nächste Folgerung aus Lemma 5.7 ist, dass das Äquivalenzproblem für zwei Turingmaschinen unentscheidbar ist. Man kann also kein Programm entwerfen, das für zwei gegebene Programme testen kann, ob die Programme das gleiche Problem lösen (die gleiche semantische Bedeutung haben).