VL-06: Rekursive Aufzählbarkeit

(Berechenbarkeit und Komplexität, WS 2018)

Gerhard Woeginger

WS 2018, RWTH

Organisatorisches

- Nächste Vorlesung:
  Donnerstag, November 22, 12:30–14:00 Uhr, Aula

- Webseite:
  `http://algo.rwth-aachen.de/Lehre/WS1819/BuK.php`

Wiederholung

Wdh.: Bisher betrachtete unentscheidbare Probleme

Die folgenden Probleme sind unentscheidbar:

Die Diagonalsprache:

$$D = \{w \in \{0,1\}^* \mid w = w_i \text{ und } M_i \text{ akzeptiert } w \text{ nicht}\}$$

Das Diagonalsprachenkomplement:

$$\overline{D} = \{w \in \{0,1\}^* \mid w = w_i \text{ und } M_i \text{ akzeptiert } w\}$$

Das Halteproblem:

$$H = \{\langle M \rangle w \mid M \text{ hält auf } w\}$$

Das Epsilon-Halteproblem:

$$H_\epsilon = \{\langle M \rangle \mid M \text{ hält auf der Eingabe } \epsilon\ \}$$

Wdh.: Der Satz von Rice

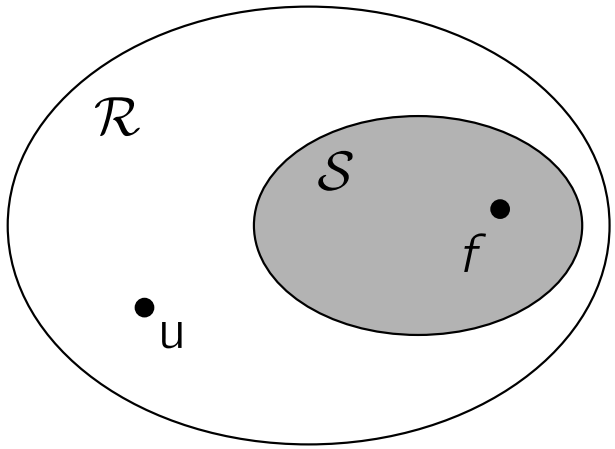
Satz

Es sei $\mathcal{R}$ die Menge der von TMen berechenbaren partiellen Funktionen.
Es sei $\mathcal{S}$ eine Teilmenge von $\mathcal{R}$ mit $\emptyset \subsetneq \mathcal{S} \subsetneq \mathcal{R}$.

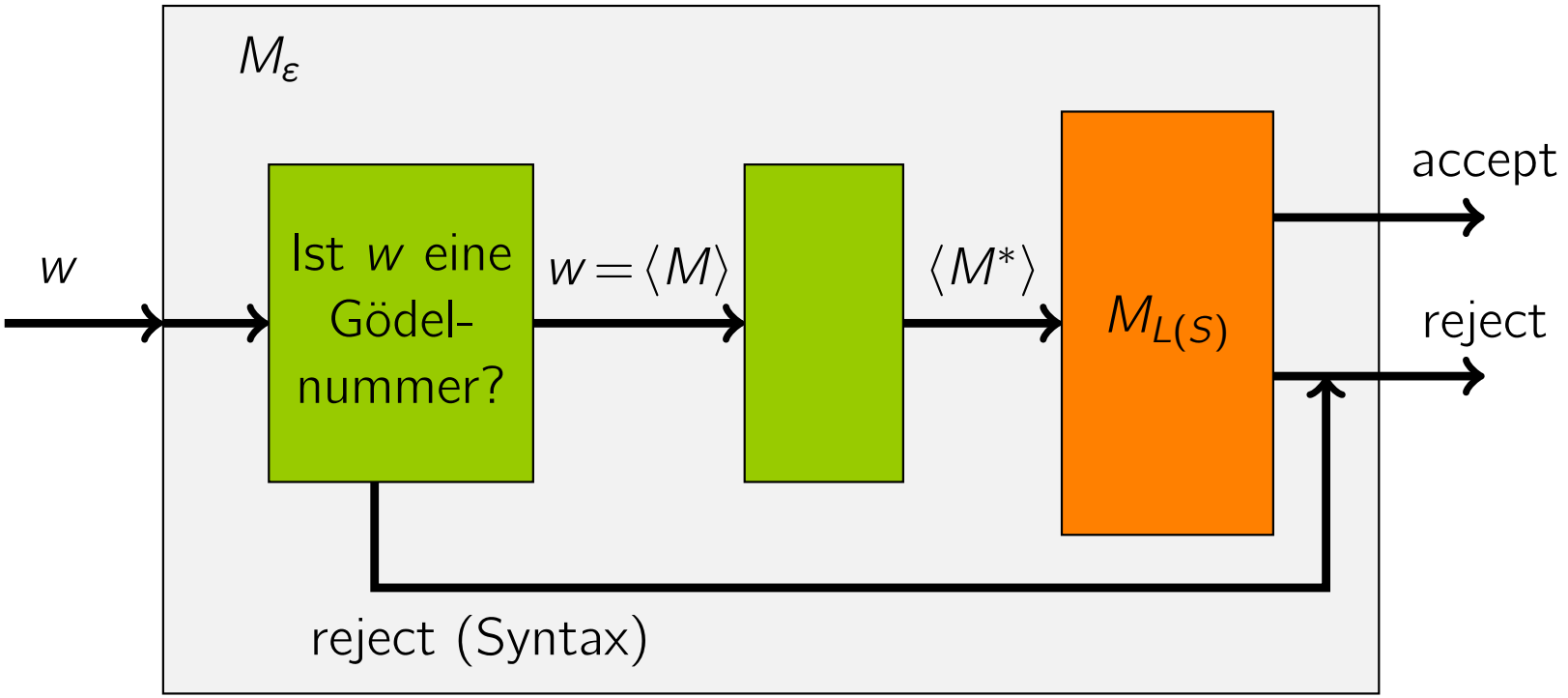
Dann ist die Sprache

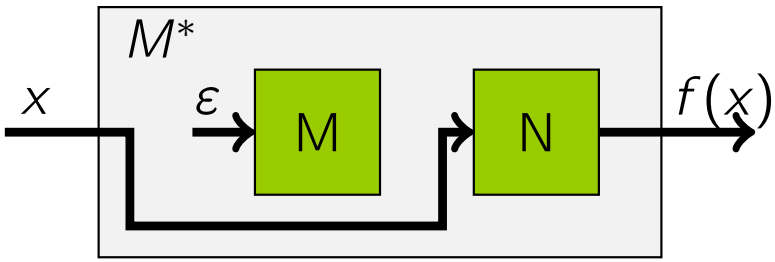
$$L(\mathcal{S}) = \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } \mathcal{S}\}$$

unentscheidbar.



Wdh.: Der Satz von Rice / Beweis





Wdh.: Satz von Rice / Anwendungsbeispiele

Beispiel 2

- Es sei $\mathcal{S} = \{f_M \mid \forall w \in \{0,1\}^* \colon f_M(w) \neq \perp\}$.
- Dann ist

$$\begin{aligned} L(\mathcal{S}) &= \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } \mathcal{S}\} \\ &= \{\langle M \rangle \mid M \text{ hält auf jeder Eingabe}\} \end{aligned}$$

- Diese Sprache ist auch als das *totale Halteproblem* H_{tot} bekannt.
- Gemäss dem Satz von Rice ist die Sprache H_{tot} nicht entscheidbar.

Beispiel 5

- Es sei $H_{32} = \{\langle M \rangle \mid$ auf jeder Eingabe hält M nach höchstens 32 Schritten $\}$.
- Über diese Sprache sagt der Satz von Rice nichts aus!

Wdh.: Satz von Rice für Java Programme

Konsequenzen für Java:

Es gibt keine algorithmische Methode (von Hand oder automatisiert; heute oder morgen oder in ferner Zukunft) um festzustellen, ob ein gegebenes Java Programm einer nicht-trivialen Spezifikation entspricht.

Analoge Konsequenzen gelten für alle anderen höheren Programmiersprachen wie C, C++, Pascal, Algol, COBOL, Python, FORTRAN, LISP, Prolog, Haskell, Scala, Idris, etc.

Vorlesung VL-06
Rekursive Aufzählbarkeit

- Semi-Entscheidbarkeit
- Rekursive Aufzählbarkeit
- Abschlusseigenschaften
- Berechenbarkeitslandschaft

- Reduktionen
- Das totale Halteproblem

Semi-Entscheidbarkeit & Rekursive Aufzählbarkeit

Semi-Entscheidbarkeit (1)

Eine Sprache L wird von einer TM M entschieden, wenn

- M auf jeder Eingabe hält, und
- M genau die Wörter aus L akzeptiert.

Wenn eine TM existiert, die die Sprache L entscheidet, so wird L als rekursiv oder entscheidbar bezeichnet.

Eine Sprache L wird von einer TM M erkannt, wenn

- M jedes Wort aus L akzeptiert, und
- M kein Wort akzeptiert, das nicht in L enthalten ist.

Also: Die von M erkannte Sprache ist genau $L(M)$.

Semi-Entscheidbarkeit (2)

Definition

Wenn eine TM existiert, die die Sprache L erkennt,
so wird L als semi-entscheidbar bezeichnet.

Anmerkung: Den Begriff semi-entscheidbar findet man in der Literatur oft auch als Turing-akzeptierbar oder Turing-erkennbar.

Anmerkung: L entscheidbar $\Longrightarrow$ L semi-entscheidbar

Semi-Entscheidbarkeit (3): Beispiel

Beispiel

Das Halteproblem $H = \{\langle M \rangle w \mid M \text{ hält auf } w\}$ ist nicht entscheidbar, aber semi-entscheidbar.

Beweis: Die folgende TM M_H erkennt die Sprache H.

Erhält M_H eine syntaktisch inkorrekte Eingabe,

- so verwirft M_H die Eingabe.

Erhält M_H eine Eingabe der Form $\langle M \rangle w$,

- so simuliert M_H die TM M mit Eingabe w
- und akzeptiert, sobald/falls M auf w hält.

Aufzähler (1)

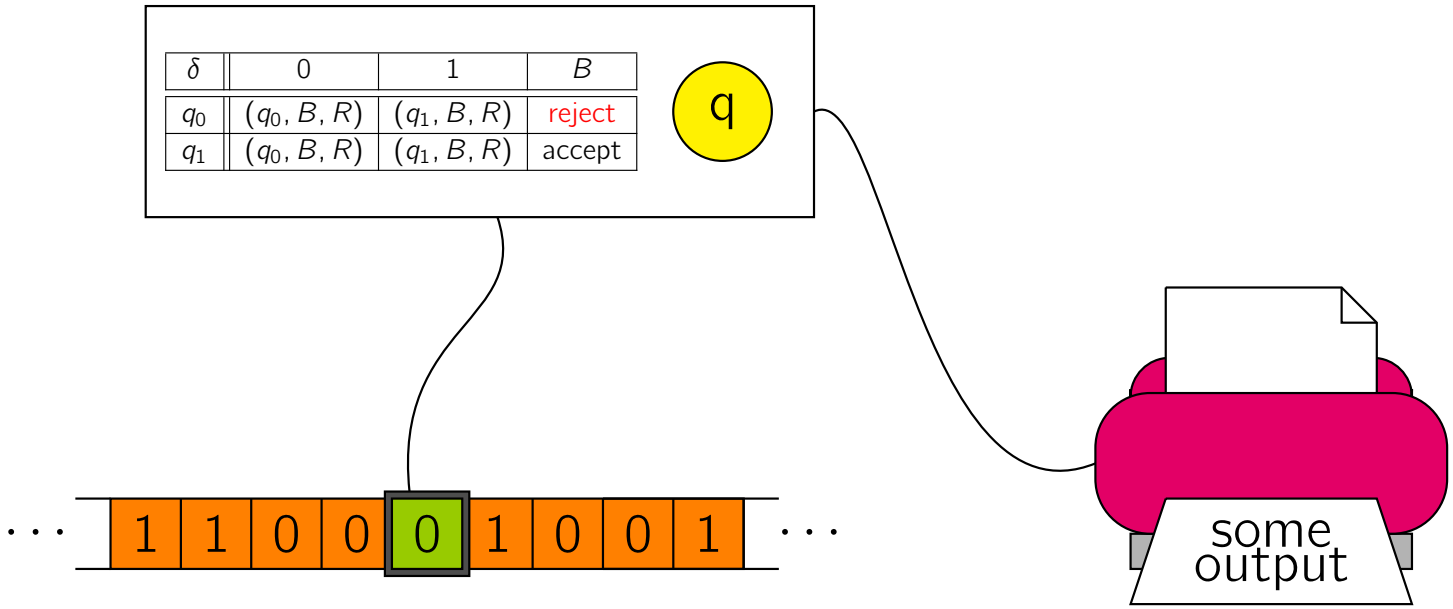
Definition

Ein Aufzähler für eine Sprache $L \subseteq \Sigma^*$ ist eine Variante der TM mit einem angeschlossenen Drucker.

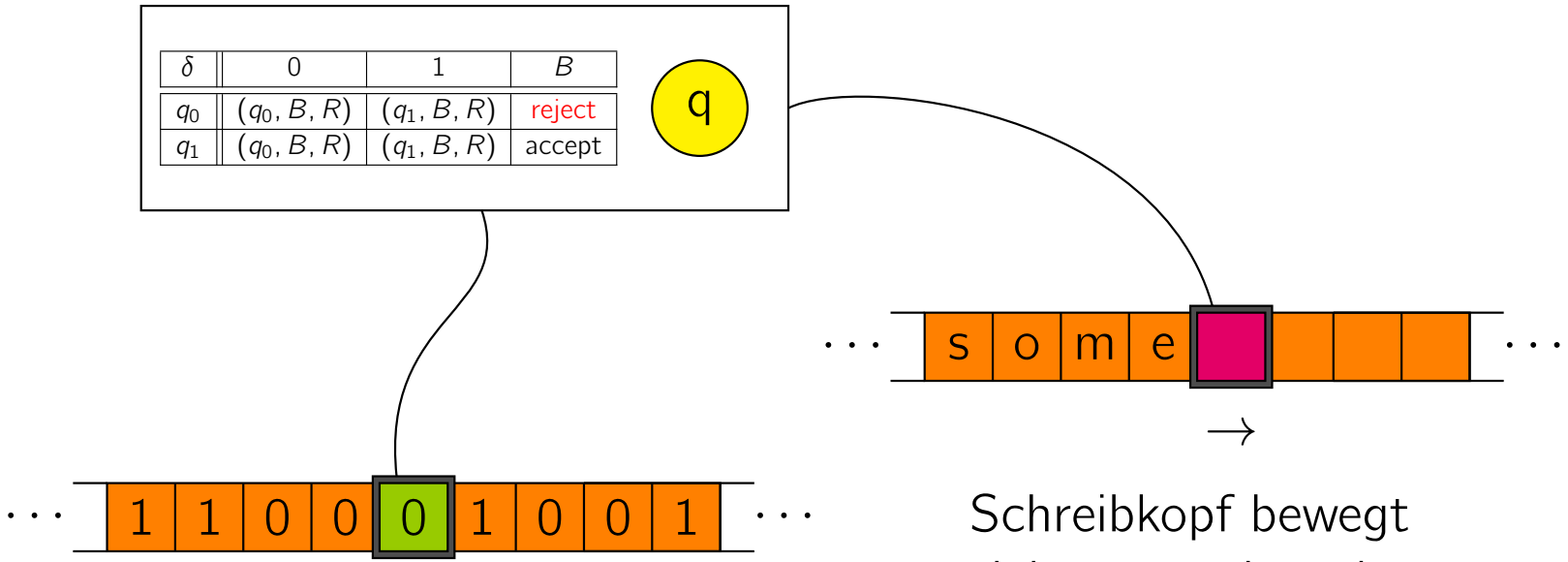
Der Drucker ist ein zusätzliches Ausgabeband, auf dem sich der Kopf nur nach rechts bewegen kann und auf dem nur geschrieben wird.

- Der Aufzähler wird mit leerem Arbeitsband gestartet, und gibt mit der Zeit alle Wörter in L (möglicherweise mit Wiederholungen) auf dem Drucker aus.
- Die ausgegebenen Wörter werden dabei immer durch ein Trennzeichen separiert, das nicht in Σ enthalten ist.
- Der Aufzähler druckt ausschliesslich Wörter in L.

Aufzähler (2)


δ
0
1
B
q0
(q0, B, R)
(q1, B, R)
reject
q1
(q0, B, R)
(q1, B, R)
accept
q
1 1 0 0 0 1 0 0 1
some output

Aufzähler (2)



Schreibkopf bewegt
sich nur nach rechts

Rekursive Aufzählbarkeit

Definition

Wenn es für die Sprache L einen Aufzähler gibt,
so wird L als rekursiv aufzählbar bezeichnet.

Zentraler Satz:

Satz (Rekursive Aufzählbarkeit $\Leftrightarrow$ Semi-Entscheidbarkeit)

Eine Sprache L ist genau dann rekursiv aufzählbar,
wenn L semi-entscheidbar ist.

Beweis (1): Rekursiv aufzählbar $\rightarrow$ semi-entscheidbar

Angenommen, L ist rekursiv aufzählbar und hat einen Aufzähler A.
Wir konstruieren eine TM M, die L erkennt.

Bei Eingabe des Wortes w arbeitet M wie folgt:

- M simuliert A mit Hilfe eines Bandes, das die Rolle des Druckers übernimmt.
- Immer wenn ein neues Wort auf das Band gedruckt worden ist, vergleicht M dieses Wort mit w und akzeptiert bei Übereinstimmung.

Korrektheit:

- Falls $w \in L$, so wird w irgendwann gedruckt und zu diesem Zeitpunkt von M akzeptiert.
- Falls $w \notin L$, so wird w niemals gedruckt und somit auch niemals von M akzeptiert.

Beweis (2): Semi-entscheidbar $\rightarrow$ rekursiv aufzählbar

Angenommen, L ist semi-entscheidbar und wird von der TM M erkannt.
Wir konstruieren einen Aufzähler A für L.

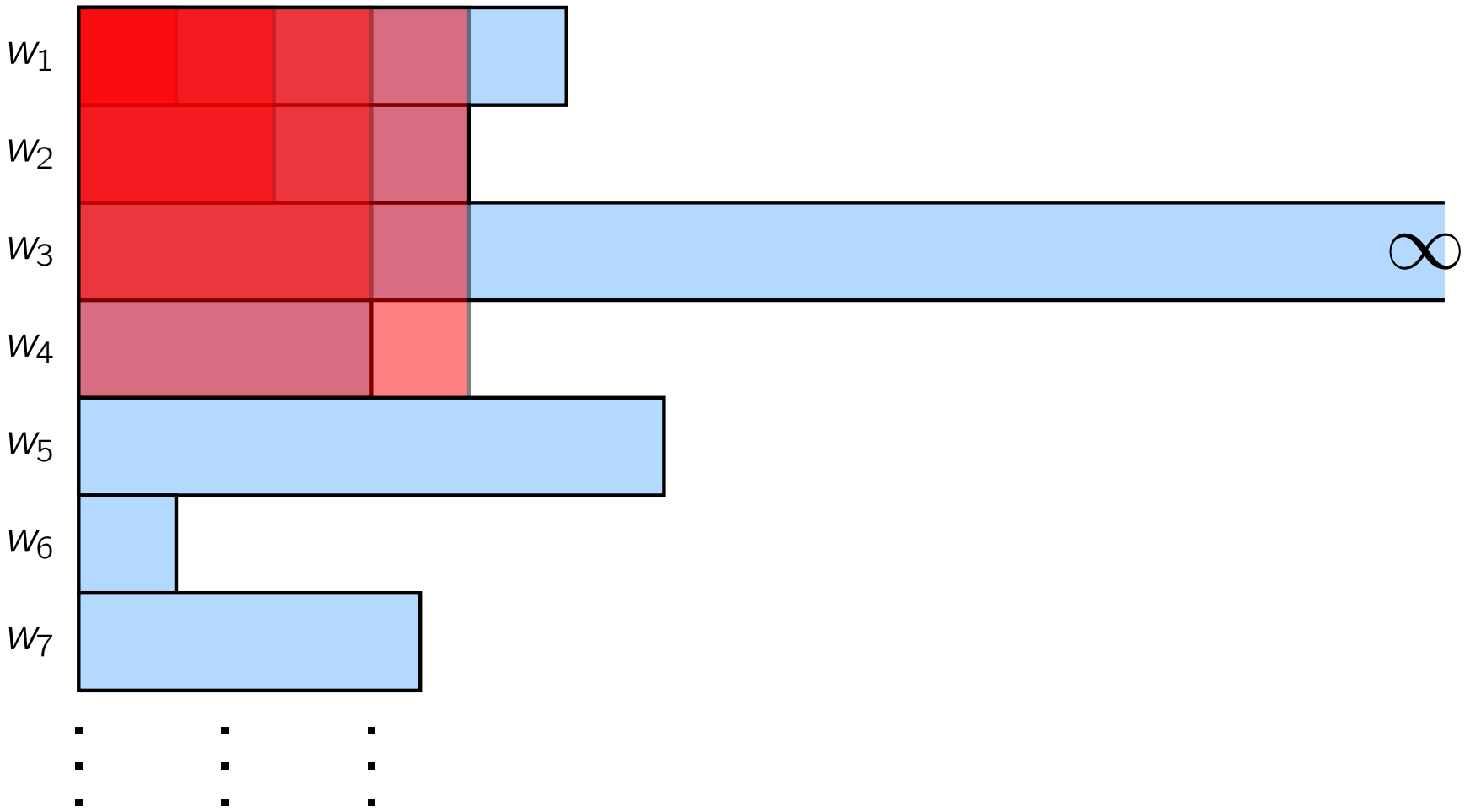
In der k-ten Runde (mit $k = 1, 2, 3, \ldots$)

- simuliert der Aufzähler je k Schritte von M auf jedem der Wörter $w_1, \ldots, w_k$.
- Immer wenn die Simulation eines der Worte akzeptiert, druckt der Aufzähler dieses Wort aus.

Korrektheit:
Der Aufzähler A druckt offensichtlich nur Wörter aus L aus.
Aber druckt er auch wirklich alle Wörter aus L aus?

- Es sei w_i ein Wort in der Sprache L. Dann wird w_i von der TM M nach einer endlichen Anzahl t_i von Schritten akzeptiert.
- Deshalb wird w_i in jeder Runde k mit $k \geq \max\{i, t_i\}$ vom Aufzähler A ausgedruckt.

Anzahl der Schritte, die TM M auf w_i benötig: $\longrightarrow$



Abschlusseigenschaften

Durchschnitt (1)

Satz

(a) Wenn die beiden Sprachen L_1 und L_2 entscheidbar sind, so ist auch die Sprache $L_1 \cap L_2$ entscheidbar.

(b) Wenn die beiden Sprachen L_1 und L_2 rekursiv aufzählbar sind, so ist auch die Sprache $L_1 \cap L_2$ rekursiv aufzählbar.

Durchschnitt (2): Beweis von Teil (a)

Es seien M_1 und M_2 zwei TMen, die L_1 respektive L_2 entscheiden.

Eine TM M, die $L_1 \cap L_2$ entscheidet:

- Bei Eingabe w simuliert M zunächst das Verhalten von M_1 auf w und dann das Verhalten von M_2 auf w.
- Falls M_1 und M_2 beide das Wort w akzeptieren, so akzeptiert auch M; andernfalls verwirft M.

Korrektheit:

- Falls $w \in L_1 \cap L_2$, so wird w akzeptiert.
- Andernfalls wird w verworfen.

Durchschnitt (3): Beweis von Teil (b)

Es seien nun M_1 und M_2 zwei TMen, die L_1 respektive L_2 erkennen.
Wir verwenden die gleiche Konstruktion für M wie in (a).

Eine TM M, die $L_1 \cap L_2$ erkennt:

- Bei Eingabe w simuliert M zunächst das Verhalten von M_1 auf w und dann das Verhalten von M_2 auf w.
- Falls M_1 und M_2 beide akzeptieren, so akzeptiert auch M.

Korrektheit:

- Falls $w \in L_1 \cap L_2$, so wird w von M akzeptiert.
- Andernfalls wird w nicht akzeptiert.

Vereinigung (1)

Satz

(a) Wenn die beiden Sprachen L_1 und L_2 entscheidbar sind, so ist auch die Sprache $L_1 \cup L_2$ entscheidbar.

(b) Wenn die beiden Sprachen L_1 und L_2 rekursiv aufzählbar sind, so ist auch die Sprache $L_1 \cup L_2$ rekursiv aufzählbar.

Vereinigung (2): Beweis von Teil (a)

Es seien M_1 und M_2 zwei TMen, die L_1 respektive L_2 entscheiden.

Eine TM M, die $L_1 \cup L_2$ entscheidet:

- Bei Eingabe w simuliert M zunächst das Verhalten von M_1 auf w und dann das Verhalten von M_2 auf w.
- Falls M_1 oder M_2 das Wort w akzeptiert, so akzeptiert auch M; andernfalls verwirft M.

Korrektheit:

- Falls $w \in L_1 \cup L_2$, so wird w von M_1 oder von M_2 und somit auch von M akzeptiert.
- Andernfalls verwerfen sowohl M_1 als auch M_2, und damit auch M.

Vereinigung (3): Beweis von Teil (b)

Es seien nun M_1 und M_2 zwei TMen, die L_1 respektive L_2 erkennen.

Eine TM M, die $L_1 \cup L_2$ erkennt

- Wir nehmen o.B.d.A. an, dass M über zwei Bänder verfügt.
- Auf Band 1 wird M_1 auf w simuliert.
- Auf Band 2 wird M_2 auf w simuliert.
- Sobald ein Schritt gemacht wird, in dem M_1 oder M_2 akzeptiert, akzeptiert auch die TM M.

Korrektheit:

- Falls $w \in L_1 \cup L_2$, so wird w von M_1 oder von M_2 und somit auch von M akzeptiert.
- Andernfalls wird w nicht akzeptiert.

Komplement (1)

Lemma

Wenn sowohl die Sprache $L \subseteq \Sigma^*$ als auch ihr Komplement $\overline{L} := \Sigma^* \setminus L$ rekursiv aufzählbar sind, so ist L entscheidbar.

Beweis:

- Es seien M und $\overline{M}$ zwei TMen, die L respektive $\overline{L}$ erkennen.
- Für ein Eingabewort w simuliert die neue TM M' das Verhalten von M auf w und das Verhalten von $\overline{M}$ auf w parallel auf zwei Bändern.
- Wenn M akzeptiert, so akzeptiert M'.
  Wenn $\overline{M}$ akzeptiert, so verwirft M'.
- Da entweder $w \in L$ oder $w \notin L$ gilt, tritt eines der beiden obigen Ereignisse (M akzeptiert; $\overline{M}$ akzeptiert) nach endlicher Zeit ein. Damit ist die Terminierung von M' sichergestellt.

Komplement (2)

Satz 1

Wenn die Sprache L entscheidbar ist, so ist auch ihr Komplement $\overline{L}$ entscheidbar.

Beweis: Wir können das Akzeptanzverhalten einer TM M, die L entscheidet, invertieren.

Satz 2

Wenn die Sprache L rekursiv aufzählbar ist, so ist ihr Komplement $\overline{L}$ nicht notwendigerweise rekursiv aufzählbar.

Beispiel:

- Das Halteproblem H ist rekursiv aufzählbar.
- Falls $\overline{H}$ ebenfalls rekursiv aufzählbar, so wäre H entscheidbar.
- Daher ist $\overline{H}$ nicht rekursiv aufzählbar.

Die Berechenbarkeitslandschaft

Berechenbarkeitslandschaft (1)

Beobachtung

Jede Sprache L fällt in genau eine der folgenden vier Familien.

(1) L ist entscheidbar, und sowohl L als auch $\overline{L}$ sind rekursiv aufzählbar.

(2) L ist rekursiv aufzählbar, aber $\overline{L}$ ist nicht rekursiv aufzählbar

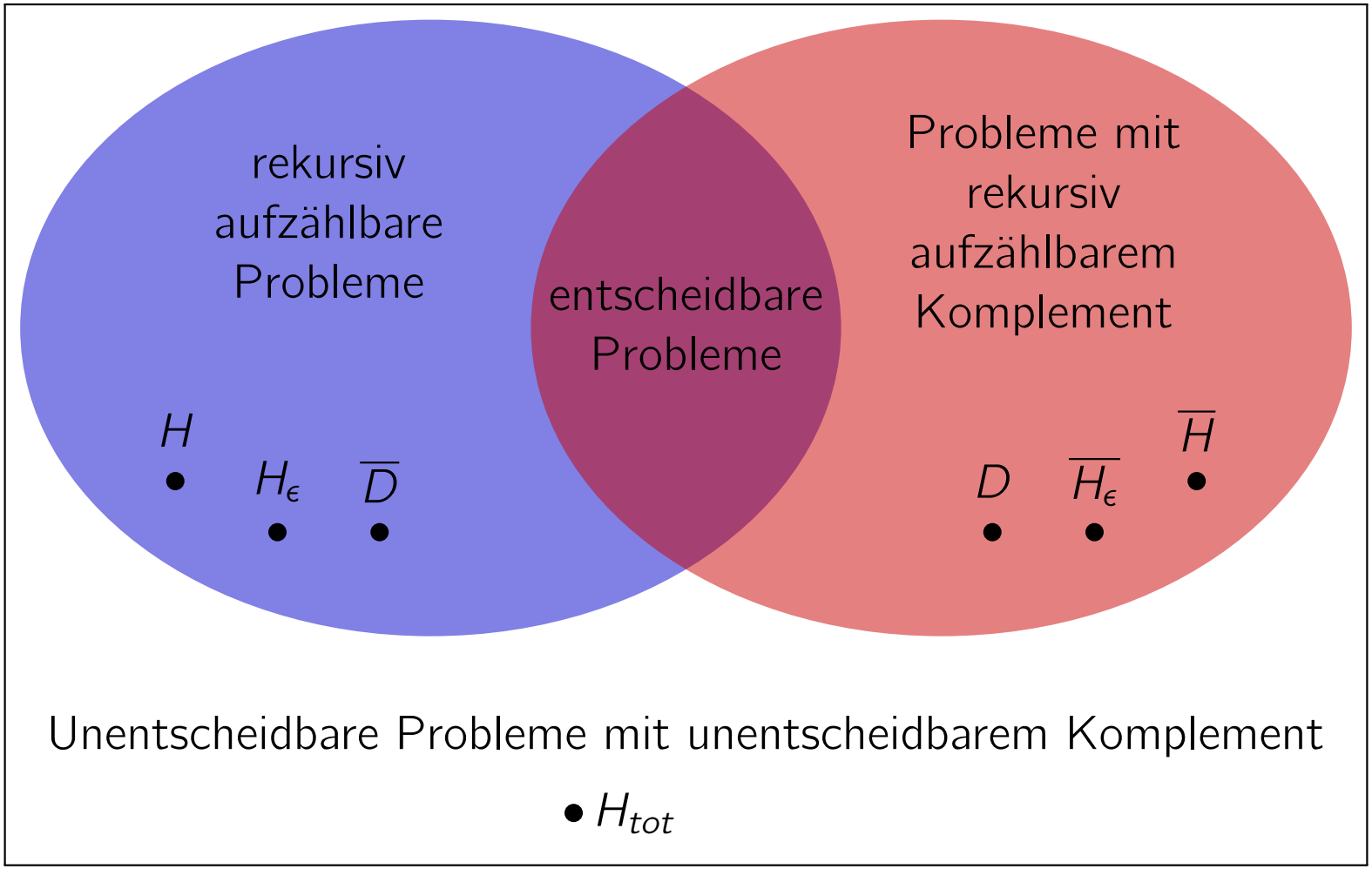
(3) $\overline{L}$ ist rekursiv aufzählbar, aber L ist nicht rekursiv aufzählbar

(4) Weder L noch $\overline{L}$ sind rekursiv aufzählbar

Beispiele

- Familie 1: Graphzusammenhang; Hamiltonkreis
- Familie 2: H, H_ϵ, $\overline{D}$
- Familie 3: $\overline{H}$, $\overline{H}_\epsilon$, D,
- Familie 4: $H_{tot} = \{\langle M \rangle \mid M \text{ hält auf jeder Eingabe}\}$

Berechenbarkeitslandschaft (2)

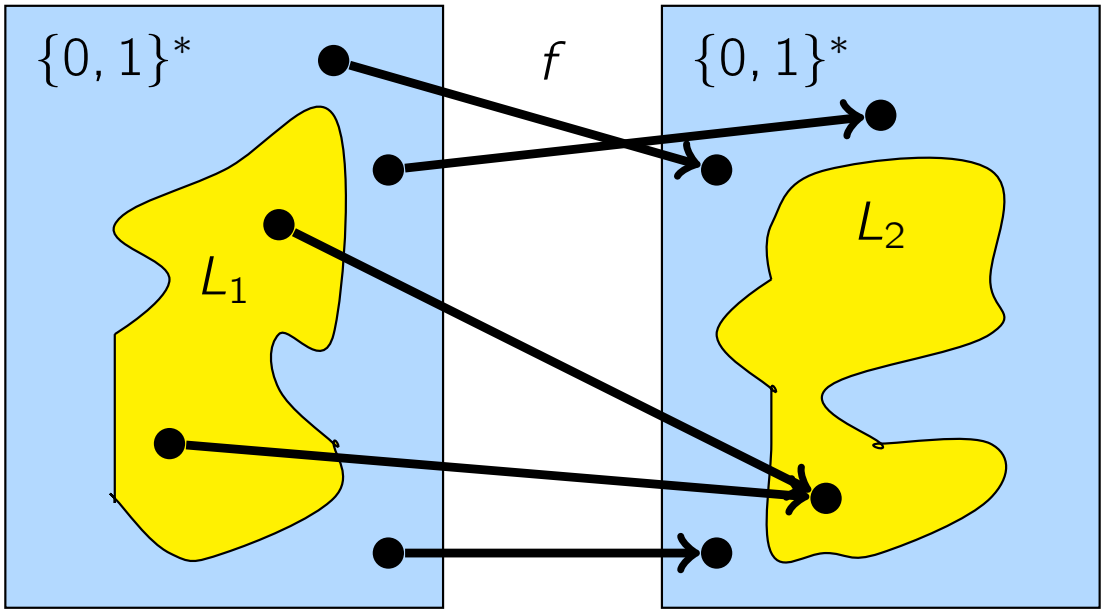


Reduktionen

Reduktionen (1)

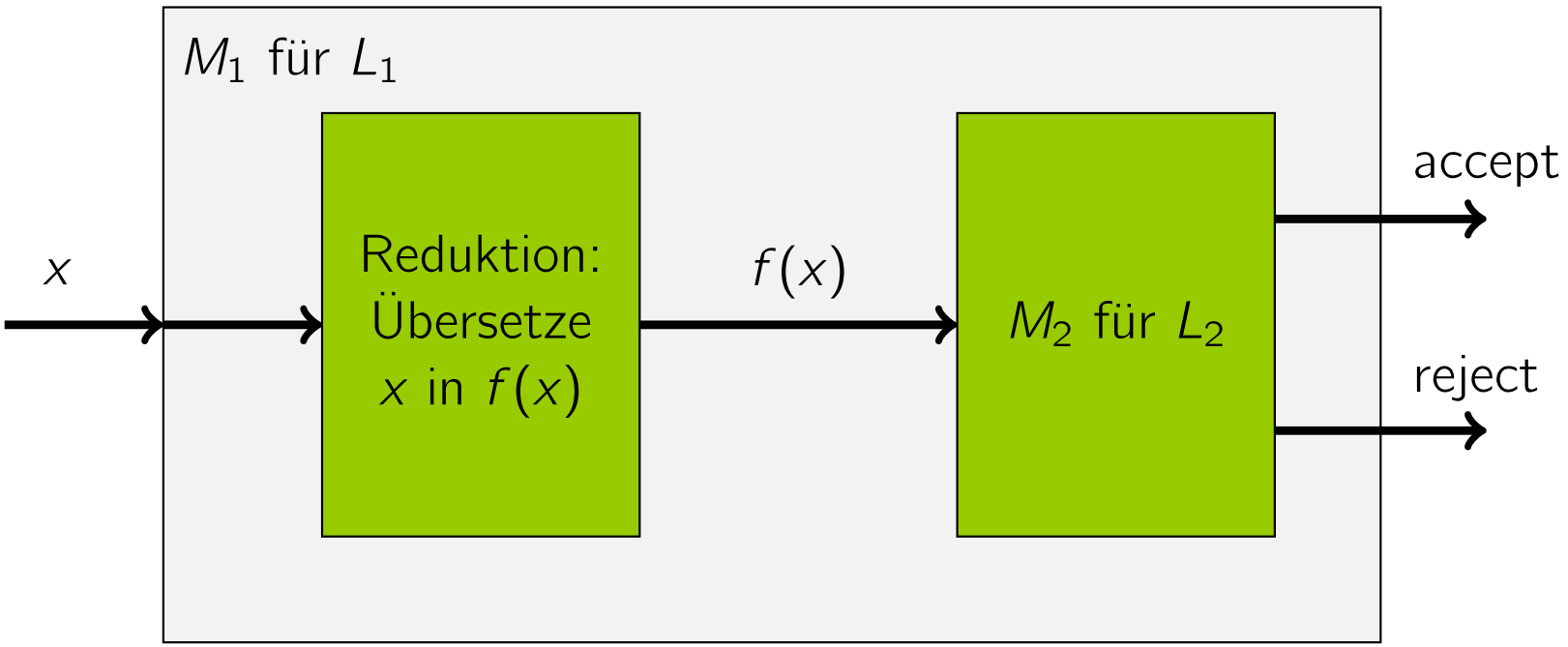
Definition

Es seien L_1 und L_2 zwei Sprachen über einem Alphabet Σ.
Dann heisst L_1 auf L_2 reduzierbar (mit der Notation $L_1 \leq L_2$),
wenn eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ existiert,
so dass für alle $x \in \Sigma^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.



Reduktionen (2)

Eine Reduktion ist ein Algorithmus,
der die Instanzen eines Startproblems
als Spezialfälle eines Zielproblems formuliert.



Reduktionen (3)

Satz

Falls $L_1 \leq L_2$ und falls L_2 rekursiv aufzählbar ist,
so ist auch L_1 rekursiv aufzählbar.

Beweis: Wir konstruieren eine TM M_1, die L_1 erkennt, indem sie als Unterprogramm eine TM M_2 verwendet, die L_2 erkennt:

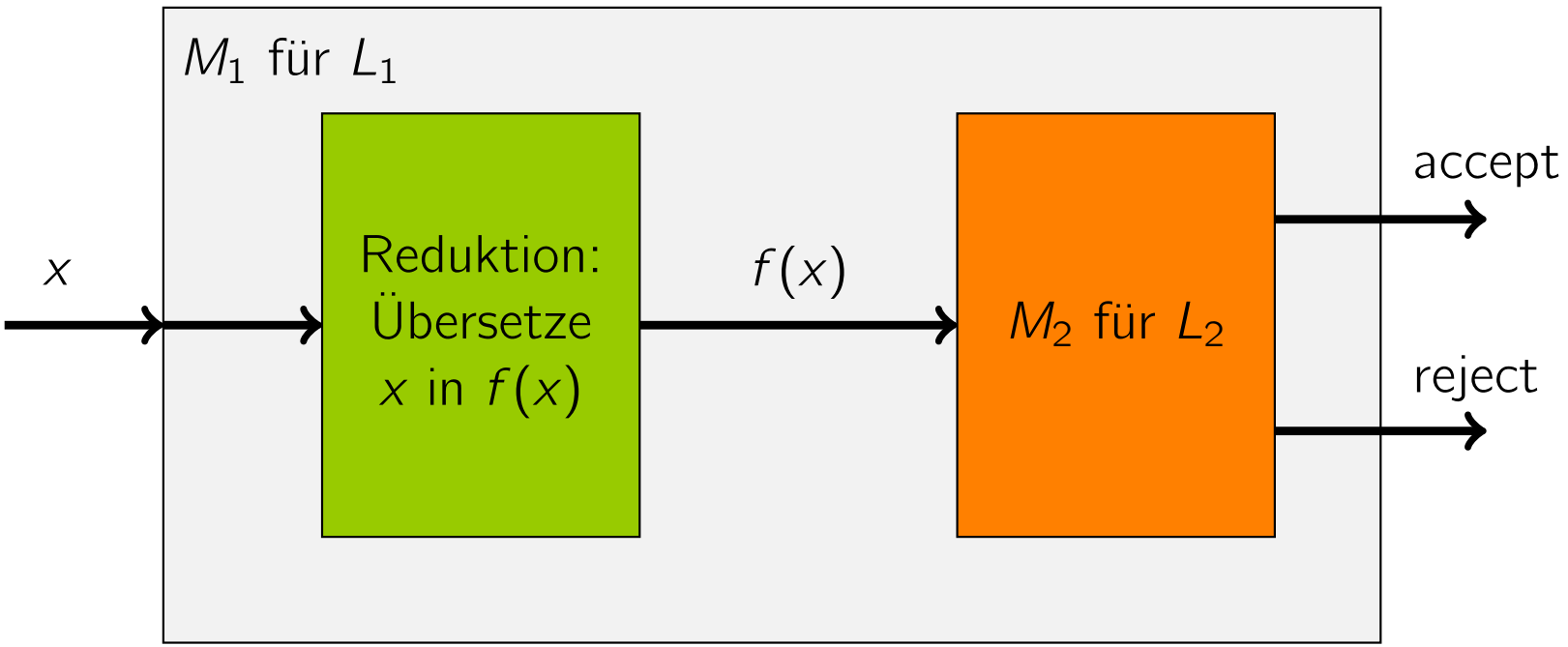
- Für eine Eingabe x berechnet die TM M_1 zunächst $f(x)$.
- Danach simuliert M_1 die TM M_2 mit der Eingabe $f(x)$.
- M_1 akzeptiert die Eingabe x, falls M_2 die Eingabe $f(x)$ akzeptiert.

$$\begin{aligned} M_1 \text{ akzeptiert } x &\Leftrightarrow M_2 \text{ akzeptiert } f(x) \\ &\Leftrightarrow f(x) \in L_2 \\ &\Leftrightarrow x \in L_1 \end{aligned}$$

Reduktionen (4)

Der bewiesene Satz und der logisch äquivalente Umkehrschluss

- Falls $L_1 \leq L_2$ und falls L_2 rekursiv aufzählbar ist, so ist auch L_1 rekursiv aufzählbar.
- Falls $L_1 \leq L_2$ und falls L_1 nicht rekursiv aufzählbar ist, so ist auch L_2 nicht rekursiv aufzählbar.



Das totale Halteproblem

Das totale Halteproblem

Definition (Totales Halteproblem)

$$H_{\text{tot}} = \{\langle M \rangle \mid M \text{ hält auf jeder Eingabe}\}$$

Wir wissen bereits: H_ϵ ist unentscheidbar, aber rekursiv aufzählbar.
Daraus folgt: $\overline{H}_\epsilon$ ist nicht rekursiv aufzählbar.

Wir werden zeigen:

Behauptung A: $\overline{H}_\epsilon \leq \overline{H}_{\text{tot}}$

Behauptung B: $\overline{H}_\epsilon \leq H_{\text{tot}}$

Aus diesen beiden Reduktionen folgt dann:

Satz

Weder $\overline{H}_{\text{tot}}$ noch H_{tot} ist rekursiv aufzählbar.

Behauptung A: $\overline{H}_\epsilon \leq \overline{H}_{\text{tot}}$ Beweis (1)

Wir beschreiben eine berechenbare Funktion f,
die Ja-Instanzen von $\overline{H}_\epsilon$ auf Ja-Instanzen von $\overline{H}_{\text{tot}}$ und
Nein-Instanzen von $\overline{H}_\epsilon$ auf Nein-Instanzen von $\overline{H}_{\text{tot}}$ abbildet.

Es sei w die Eingabe für $\overline{H}_\epsilon$.

- Wenn w keine gültige Gödelnummer ist, so setzen wir $f(w) = w$.
- Falls $w = \langle M \rangle$ für eine TM M, so sei $f(w) := \langle M^*_\epsilon \rangle$ die Gödelnummer der TM M^*_ϵ mit folgendem Verhalten:

  M^*_ϵ ignoriert die Eingabe und simuliert M mit der Eingabe ϵ.

Die beschriebene Funktion f ist berechenbar. (Warum?)

Behauptung A: $\overline{H}_\epsilon \leq \overline{H}_{\text{tot}}$ Beweis (2)

Für die Korrektheit zeigen wir:

(a) $w \in \overline{H}_\epsilon \Rightarrow f(w) \in \overline{H}_{\text{tot}}$

(b) $w \notin \overline{H}_\epsilon \Rightarrow f(w) \notin \overline{H}_{\text{tot}}$

Falls w keine Gödelnummer ist, gilt $w \in \overline{H}_\epsilon$ und $f(w) \in \overline{H}_{\text{tot}}$.
Dieser Unterfall von (a) ist also korrekt erledigt.

Falls $w = \langle M \rangle$ für eine TM M, so betrachten wir $f(w) = \langle M^*_\epsilon \rangle$.
Dann gilt:

$$
\begin{aligned}
w \in \overline{H}_\epsilon &\Rightarrow M \text{ hält nicht auf der Eingabe } \epsilon. \\
&\Rightarrow M^*_\epsilon \text{ hält auf gar keiner Eingabe.} \\
&\Rightarrow \langle M^*_\epsilon \rangle \notin H_{\text{tot}} \\
&\Rightarrow f(w) = \langle M^*_\epsilon \rangle \in \overline{H}_{\text{tot}} \qquad \text{und (a) ist korrekt.}
\end{aligned}
$$

Behauptung A: $\overline{H}_\epsilon \leq \overline{H}_{\text{tot}}$ Beweis (3)

Für die Korrektheit zeigen wir:

(a) $w \in \overline{H}_\epsilon \Rightarrow f(w) \in \overline{H}_{\text{tot}}$

(b) $w \notin \overline{H}_\epsilon \Rightarrow f(w) \notin \overline{H}_{\text{tot}}$

Falls $w = \langle M \rangle$ für eine TM M, so betrachten wir $f(w) = \langle M^*_\epsilon \rangle$.
Dann gilt:

$$
\begin{aligned}
w \notin \overline{H}_\epsilon \quad &\Rightarrow \quad w \in H_\epsilon \\
&\Rightarrow \quad M \text{ hält auf der Eingabe } \epsilon. \\
&\Rightarrow \quad M^*_\epsilon \text{ hält auf jeder Eingabe} \\
&\Rightarrow \quad \langle M^*_\epsilon \rangle \in H_{\text{tot}} \\
&\Rightarrow \quad f(w) = \langle M^*_\epsilon \rangle \notin \overline{H}_{\text{tot}} \qquad \text{und (b) ist korrekt.}
\end{aligned}
$$

Damit ist Behauptung A bewiesen. □

Behauptung B: $\overline{H}_\epsilon \leq H_{\text{tot}}$ Beweis (1)

Wir beschreiben eine berechenbare Funktion f,
die Ja-Instanzen von $\overline{H}_\epsilon$ auf Ja-Instanzen von H_{tot} und
Nein-Instanzen von $\overline{H}_\epsilon$ auf Nein-Instanzen von H_{tot} abbildet.

Es sei w die Eingabe für $\overline{H}_\epsilon$. Es sei w' irgendein Wort aus H_{tot}.

- Wenn w keine gültige Gödelnummer ist, so setzen wir $f(w) = w'$.
- Falls $w = \langle M \rangle$ für eine TM M, so sei $f(w) := \langle M' \rangle$ die Gödelnummer der TM M', die sich auf Eingaben der Länge ℓ wie folgt verhält:

  M' simuliert die ersten ℓ Schritte von M auf der Eingabe ϵ.
  Wenn M innerhalb dieser ℓ Schritte hält, dann geht M' in eine Endlosschleife; andernfalls hält M'.

Die beschriebene Funktion f ist berechenbar. (Warum?)

Behauptung B: $\overline{H}_\epsilon \leq H_{\text{tot}}$ Beweis (2)

Für die Korrektheit zeigen wir:

(a) $w \in \overline{H}_\epsilon \Rightarrow f(w) \in H_{\text{tot}}$

(b) $w \notin \overline{H}_\epsilon \Rightarrow f(w) \notin H_{\text{tot}}$

Falls w keine Gödelnummer ist, gilt $w \in \overline{H}_\epsilon$ und $f(w) = w' \in H_{\text{tot}}$.
Dieser Unterfall von (a) ist also korrekt erledigt.

Falls $w = \langle M \rangle$ für eine TM M, so betrachten wir $f(w) = \langle M' \rangle$.
Dann gilt:

$$
\begin{aligned}
w \in \overline{H}_\epsilon &\Rightarrow M \text{ hält nicht auf der Eingabe } \epsilon \\
&\Rightarrow \neg\exists i\text{: } M \text{ hält innerhalb von } i \text{ Schritten auf } \epsilon \\
&\Rightarrow \forall i\text{: } M \text{ hält nicht innerhalb von } i \text{ Schritten auf } \epsilon \\
&\Rightarrow \forall i\text{: } M' \text{ hält auf allen Eingaben der Länge } i \\
&\Rightarrow M' \text{ hält auf jeder Eingabe} \\
&\Rightarrow f(w) = \langle M' \rangle \in H_{\text{tot}} \qquad \text{und (a) ist korrekt.}
\end{aligned}
$$

Behauptung B: $\overline{H}_\epsilon \leq H_{\text{tot}}$ Beweis (3)

Für die Korrektheit zeigen wir:

(a) $w \in \overline{H}_\epsilon \Rightarrow f(w) \in H_{\text{tot}}$

(b) $w \notin \overline{H}_\epsilon \Rightarrow f(w) \notin H_{\text{tot}}$

Falls $w = \langle M \rangle$ für eine TM M, so betrachten wir $f(w) = \langle M' \rangle$.
Dann gilt:

$$
\begin{aligned}
w \notin \overline{H}_\epsilon \quad &\Rightarrow \quad M \text{ hält auf der Eingabe } \epsilon. \\
&\Rightarrow \quad \exists i\text{: } M \text{ hält innerhalb von } i \text{ Schritten auf } \epsilon. \\
&\Rightarrow \quad \exists i\text{: } M' \text{ hält auf keiner Eingabe mit Länge} \geq i. \\
&\Rightarrow \quad M' \text{ hält nicht auf jeder Eingabe.} \\
&\Rightarrow \quad f(w) = \langle M' \rangle \notin H_{\text{tot}} \qquad \text{und (b) ist korrekt.}
\end{aligned}
$$

Damit ist Behauptung B bewiesen. □