

Theorie der Informatik

III.4. Halteproblem, Unentscheidbarkeit, Reduzierbarkeit

Malte Helmert Christian Tschudin

Universität Basel

17. April 2013

(Semi-) Entscheidbarkeit und rekursive Aufzählbarkeit

Berechenbarkeit vs. Entscheidbarkeit

- Letzte Vorlesung: Berechenbarkeit von Funktionen
- Jetzt: Analoges Konzept für Sprachen

Warum Sprachen?

- nur Ja/Nein-Fragen („Ist $w \in L$?“)
statt allgemeiner Funktionsberechnungen („Was ist $f(w)$?“)
macht es einfacher, Fragestellungen zu untersuchen
- Ergebnisse sind auf allgemeineres Problem
der Berechnung von Funktionen direkt übertragbar

Entscheidbarkeit

Definition (Entscheidbarkeit)

Eine Menge (Sprache) $A \subseteq \Sigma^*$ heisst **entscheidbar**, falls die **charakteristische Funktion von A** , nämlich $\chi_A : \Sigma^* \rightarrow \{0, 1\}$, **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi_A(w) := \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

Entscheidbarkeit

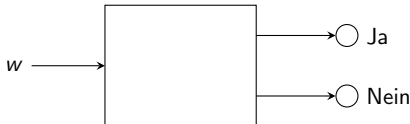
Definition (Entscheidbarkeit)

Eine Menge (Sprache) $A \subseteq \Sigma^*$ heisst **entscheidbar**, falls die **charakteristische Funktion von A** , nämlich $\chi_A : \Sigma^* \rightarrow \{0, 1\}$, **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi_A(w) := \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

Anschaulich:



Entscheidbarkeit

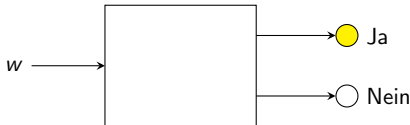
Definition (Entscheidbarkeit)

Eine Menge (Sprache) $A \subseteq \Sigma^*$ heisst **entscheidbar**, falls die **charakteristische Funktion von A** , nämlich $\chi_A : \Sigma^* \rightarrow \{0, 1\}$, **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi_A(w) := \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

Anschaulich:



Entscheidbarkeit

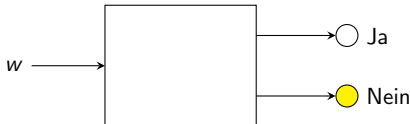
Definition (Entscheidbarkeit)

Eine Menge (Sprache) $A \subseteq \Sigma^*$ heisst **entscheidbar**, falls die **charakteristische Funktion von A** , nämlich $\chi_A : \Sigma^* \rightarrow \{0, 1\}$, **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi_A(w) := \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

Anschaulich:



Semi-Entscheidbarkeit

Definition (Semi-Entscheidbarkeit)

Eine Menge $A \subseteq \Sigma^*$ heisst **semi-entscheidbar**, falls die „**halbe**“ **charakteristische Funktion von A** , nämlich $\chi'_A : \Sigma^* \rightarrow \{0, 1\}$ **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi'_A(w) = \begin{cases} 1, & w \in A \\ \text{undefiniert}, & w \notin A \end{cases}$$

Semi-Entscheidbarkeit

Definition (Semi-Entscheidbarkeit)

Eine Menge $A \subseteq \Sigma^*$ heisst **semi-entscheidbar**, falls die „**halbe**“ **charakteristische Funktion von A** , nämlich $\chi'_A : \Sigma^* \rightarrow \{0, 1\}$ **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi'_A(w) = \begin{cases} 1, & w \in A \\ \text{undefiniert}, & w \notin A \end{cases}$$

Ist das nicht praktisch dasselbe wie Entscheidbarkeit?

Semi-Entscheidbarkeit

Definition (Semi-Entscheidbarkeit)

Eine Menge $A \subseteq \Sigma^*$ heisst **semi-entscheidbar**, falls die „**halbe**“ **charakteristische Funktion von A** , nämlich $\chi'_A : \Sigma^* \rightarrow \{0, 1\}$ **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi'_A(w) = \begin{cases} 1, & w \in A \\ \text{undefiniert}, & w \notin A \end{cases}$$

Ist das nicht praktisch dasselbe wie Entscheidbarkeit? Nein!

Anschaulich:



Semi-Entscheidbarkeit

Definition (Semi-Entscheidbarkeit)

Eine Menge $A \subseteq \Sigma^*$ heisst **semi-entscheidbar**, falls die „**halbe**“ **charakteristische Funktion von A** , nämlich $\chi'_A : \Sigma^* \rightarrow \{0, 1\}$ **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi'_A(w) = \begin{cases} 1, & w \in A \\ \text{undefiniert}, & w \notin A \end{cases}$$

Ist das nicht praktisch dasselbe wie Entscheidbarkeit? Nein!

Anschaulich:



Semi-Entscheidbarkeit

Definition (Semi-Entscheidbarkeit)

Eine Menge $A \subseteq \Sigma^*$ heisst **semi-entscheidbar**, falls die „**halbe**“ **charakteristische Funktion von A** , nämlich $\chi'_A : \Sigma^* \rightarrow \{0, 1\}$ **berechenbar** ist.

Hierbei ist für alle $w \in \Sigma^*$:

$$\chi'_A(w) = \begin{cases} 1, & w \in A \\ \text{undefiniert}, & w \notin A \end{cases}$$

Ist das nicht praktisch dasselbe wie Entscheidbarkeit? Nein!

Anschaulich:



Zusammenhang Entscheidbarkeit/Semi-Entscheidbarkeit

Satz (entscheidbar vs. semi-entscheidbar)

Eine Sprache A ist entscheidbar genau dann, wenn sowohl A als auch $\bar{A}$ semi-entscheidbar sind.

$\Rightarrow$: offensichtlich ([Warum?](#))

$\Leftarrow$: Sei M_A ein Semi-Entscheidungsalgorithmus für A ,
 $M_{\bar{A}}$ ein Semi-Entscheidungsverfahren für $\bar{A}$.

Folgender Algorithmus ist dann Entscheidungsverfahren für A , d.h. berechnet χ_A :

INPUT (x):

FOR $s := 1, 2, 3, \dots$ DO

 IF M_A stoppt in s Schritten THEN OUTPUT(1)

 IF $M_{\bar{A}}$ stoppt in s Schritten THEN OUTPUT(0)

DONE

Rekursive Aufzählbarkeit: Definition

Definition (rekursive Aufzählbarkeit)

Eine Sprache $A \subseteq \Sigma^*$ heisst **rekursiv aufzählbar**, falls $A = \emptyset$ oder falls es eine **totale und berechenbare Funktion** $f : \mathbb{N}_0 \rightarrow \Sigma^*$ gibt, so dass

$$A = \{f(0), f(1), f(2) \dots\}.$$

Sprechweise: f zählt A (rekursiv) auf.

f muss nicht injektiv sein!

$\rightsquigarrow$ **nicht verwechseln** mit **abzählbar**!

Rekursive Aufzählbarkeit: Beispiele

- $f(x) = a^x$
- $f(x) = \begin{cases} \text{Hund,} & \text{falls } x \bmod 3 = 0 \\ \text{Katze,} & \text{falls } x \bmod 3 = 1 \\ \text{Superpapagei,} & \text{falls } x \bmod 3 = 2 \end{cases}$
- $f(x) = \begin{cases} 2^x - 1 & \text{falls } 2^x - 1 \text{ Primzahl} \\ 3 & \text{sonst} \end{cases}$

Rekursive Aufzählbarkeit: Beispiele

- $f(x) = a^x$

- $f(x) = \begin{cases} \text{Hund,} & \text{falls } x \bmod 3 = 0 \\ \text{Katze,} & \text{falls } x \bmod 3 = 1 \\ \text{Superpapagei,} & \text{falls } x \bmod 3 = 2 \end{cases}$

- $f(x) = \begin{cases} 2^x - 1 & \text{falls } 2^x - 1 \text{ Primzahl} \\ 3 & \text{sonst} \end{cases}$

Rekursive Aufzählbarkeit: Beispiele

- $f(x) = a^x$ zählt $\{\epsilon, a, aa, \dots\}$ auf.
- $f(x) = \begin{cases} \text{Hund,} & \text{falls } x \bmod 3 = 0 \\ \text{Katze,} & \text{falls } x \bmod 3 = 1 \\ \text{Superpapagei,} & \text{falls } x \bmod 3 = 2 \end{cases}$
- $f(x) = \begin{cases} 2^x - 1 & \text{falls } 2^x - 1 \text{ Primzahl} \\ 3 & \text{sonst} \end{cases}$

Rekursive Aufzählbarkeit: Beispiele

- $f(x) = a^x$ zählt $\{\epsilon, a, aa, \dots\}$ auf.
- $f(x) = \begin{cases} \text{Hund,} & \text{falls } x \bmod 3 = 0 \\ \text{Katze,} & \text{falls } x \bmod 3 = 1 \\ \text{Superpapagei,} & \text{falls } x \bmod 3 = 2 \end{cases}$
- $f(x) = \begin{cases} 2^x - 1 & \text{falls } 2^x - 1 \text{ Primzahl} \\ 3 & \text{sonst} \end{cases}$

Rekursive Aufzählbarkeit: Beispiele

- $f(x) = a^x$ zählt $\{\epsilon, a, aa, \dots\}$ auf.
- $f(x) = \begin{cases} \text{Hund,} & \text{falls } x \bmod 3 = 0 \\ \text{Katze,} & \text{falls } x \bmod 3 = 1 \\ \text{Superpapagei,} & \text{falls } x \bmod 3 = 2 \end{cases}$
zählt $\{\text{Hund, Katze, Superpapagei}\}$ auf.
- $f(x) = \begin{cases} 2^x - 1 & \text{falls } 2^x - 1 \text{ Primzahl} \\ 3 & \text{sonst} \end{cases}$

Rekursive Aufzählbarkeit: Beispiele

- $f(x) = a^x$ zählt $\{\epsilon, a, aa, \dots\}$ auf.
- $f(x) = \begin{cases} \text{Hund,} & \text{falls } x \bmod 3 = 0 \\ \text{Katze,} & \text{falls } x \bmod 3 = 1 \\ \text{Superpapagei,} & \text{falls } x \bmod 3 = 2 \end{cases}$
zählt $\{\text{Hund, Katze, Superpapagei}\}$ auf.
- $f(x) = \begin{cases} 2^x - 1 & \text{falls } 2^x - 1 \text{ Primzahl} \\ 3 & \text{sonst} \end{cases}$

Rekursive Aufzählbarkeit: Beispiele

- $f(x) = a^x$ zählt $\{\epsilon, a, aa, \dots\}$ auf.
- $f(x) = \begin{cases} \text{Hund,} & \text{falls } x \bmod 3 = 0 \\ \text{Katze,} & \text{falls } x \bmod 3 = 1 \\ \text{Superpapagei,} & \text{falls } x \bmod 3 = 2 \end{cases}$
zählt $\{\text{Hund, Katze, Superpapagei}\}$ auf.
- $f(x) = \begin{cases} 2^x - 1 & \text{falls } 2^x - 1 \text{ Primzahl} \\ 3 & \text{sonst} \end{cases}$
zählt **Mersenne-Primzahlen** auf.

Rekursive Aufzählbarkeit und Semi-Entscheidbarkeit

Satz (rekursiv aufzählbar = semi-entscheidbar)

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

Spezialfall $A = \emptyset$ ist kein Problem. Sei daher $A \neq \emptyset$.

Rekursive Aufzählbarkeit und Semi-Entscheidbarkeit

Satz (rekursiv aufzählbar = semi-entscheidbar)

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

Spezialfall $A = \emptyset$ ist kein Problem. Sei daher $A \neq \emptyset$.

$\Rightarrow$:

Angenommen A ist rekursiv aufzählbar mittels Funktion f .
Dann ist dies ein Semi-Entscheidungsverfahren für A :

```
INPUT (x):  
FOR  $n := 0, 1, 2, 3, \dots$  DO  
    IF  $f(n) = x$  THEN OUTPUT(1)  
DONE
```

Rekursive Aufzählbarkeit und Semi-Entscheidbarkeit

⇔: **Erinnerung:** Wir kennen Abbildung von $\mathbb{N}_0^2$ auf $\mathbb{N}_0$ als
 $n = c(x, y)$ mit Umkehrfunktionen $x = e(n)$ und $y = f(n)$.

Rekursive Aufzählbarkeit und Semi-Entscheidbarkeit

⇔: **Erinnerung:** Wir kennen Abbildung von $\mathbb{N}_0^2$ auf $\mathbb{N}_0$ als
 $n = c(x, y)$ mit Umkehrfunktionen $x = e(n)$ und $y = f(n)$.

Sei A semi-entscheidbar mittels Algorithmus M .

Sei a ein festes Element aus A . Definiere:

$$f(n) := \begin{cases} x & \text{falls } n \text{ Kodierung eines Paares } (x, y) \text{ ist} \\ & \text{und } M \text{ angesetzt auf } x \text{ stoppt in } y \text{ Schritten} \\ a & \text{sonst} \end{cases}$$

Rekursive Aufzählbarkeit und Semi-Entscheidbarkeit

⇐: **Erinnerung:** Wir kennen Abbildung von $\mathbb{N}_0^2$ auf $\mathbb{N}_0$ als
 $n = c(x, y)$ mit Umkehrfunktionen $x = e(n)$ und $y = f(n)$.

Sei A semi-entscheidbar mittels Algorithmus M .

Sei a ein festes Element aus A . Definiere:

$$f(n) := \begin{cases} x & \text{falls } n \text{ Kodierung eines Paares } (x, y) \text{ ist} \\ & \text{und } M \text{ angesetzt auf } x \text{ stoppt in } y \text{ Schritten} \\ a & \text{sonst} \end{cases}$$

f ist **total** und **berechenbar** und hat als **Wertebereich** A .

Also zählt f die Sprache A rekursiv auf.

Berechnungs- und Entscheidungsmodelle

Zusammen mit den Ergebnissen aus dem vorherigen Kapitel folgt, dass folgende Aussagen über Sprache A alle äquivalent sind:

- ① A ist rekursiv aufzählbar.
- ② A ist semi-entscheidbar.
- ③ A ist vom Typ 0.
- ④ $A = T(M)$ für eine Turingmaschine M .
- ⑤ χ'_A ist (Turing-, WHILE-, GOTO-) berechenbar. (*)
- ⑥ χ'_A ist μ -rekursiv. (*)
- ⑦ A ist Definitionsbereich einer berechenbaren Funktion.
- ⑧ A ist Wertebereich einer berechenbaren Funktion.

(*): WHILE-/GOTO-Berechenbarkeit und μ -Rekursion
erfordern Kodierung des Eingabeworts als Zahl.

Spezielles Halteproblem

Probleme

Begriff: Entscheidungsproblem = Problem = Sprache
(Teilmenge von Σ^* für endliches Alphabet Σ .)

Unentscheidbare Probleme

- Wir kennen jetzt viele Charakterisierungen von Semi-Entscheidbarkeit und Entscheidbarkeit.
- Was noch fehlt, ist ein **konkretes Beispiel** für ein **unentscheidbares** (= nicht entscheidbares) Problem.
- Gibt es unentscheidbare Probleme überhaupt?

Unentscheidbare Probleme

- Wir kennen jetzt viele Charakterisierungen von Semi-Entscheidbarkeit und Entscheidbarkeit.
- Was noch fehlt, ist ein **konkretes Beispiel** für ein **unentscheidbares** (= nicht entscheidbares) Problem.
- Gibt es unentscheidbare Probleme überhaupt?
- Ja! **Zählargument**: es gibt (für festes Σ) so viele **Entscheidungsverfahren** (z.B.: Turingmaschinen) wie Zahlen in $\mathbb{N}_0$, aber so viele **Sprachen** wie Zahlen in $\mathbb{R}$.
Da $\mathbb{R}$ mächtiger („grösser“) ist als $\mathbb{N}_0$, muss es daher Sprachen ohne Entscheidungsverfahren geben.

Unentscheidbare Probleme

- Wir kennen jetzt viele Charakterisierungen von Semi-Entscheidbarkeit und Entscheidbarkeit.
- Was noch fehlt, ist ein **konkretes Beispiel** für ein **unentscheidbares** (= nicht entscheidbares) Problem.
- Gibt es unentscheidbare Probleme überhaupt?
- Ja! **Zählargument**: es gibt (für festes Σ) so viele **Entscheidungsverfahren** (z.B.: Turingmaschinen) wie Zahlen in $\mathbb{N}_0$, aber so viele **Sprachen** wie Zahlen in $\mathbb{R}$.
Da $\mathbb{R}$ mächtiger („grösser“) ist als $\mathbb{N}_0$, muss es daher Sprachen ohne Entscheidungsverfahren geben.
- Dieses Argument liefert uns aber kein **konkretes** unentscheidbares Problem. $\rightsquigarrow$ Ziel dieses Abschnitts

Turingmaschinen als Eingaben

- Das erste unentscheidbare Problem, das wir kennenlernen werden, verwendet Turingmaschinen als **Eingabe**.
 $\rightsquigarrow$ „Programme, die Programme als Eingabe haben“:
 vgl. Compiler, Interpreter, virtuelle Maschinen, etc.
- Wir müssen uns also zunächst Gedanken darüber machen, wie wir **beliebige Turingmaschinen** als **Wörter über einem festen Alphabet** kodieren können.
- Wir verwenden das binäre Alphabet $\Sigma = \{0, 1\}$.
- Als Zwischenschritt kodieren wir zunächst über dem Alphabet $\Sigma' = \{0, 1, \#\}$.

Kodierung einer Turingmaschine als Wort (1)

Schritt 1: Kodierung einer Turingmaschine als Wort über $\{0, 1, \#\}$

Erinnerung: Turingmaschine $M = \langle Z, \Sigma, \Gamma, \delta, z_0, \square, E \rangle$

Idee:

- Zustände in Z und Symbole in Γ nummerieren und als Zahlen $0, 1, 2, \dots$ betrachten
- Anfangszustand erhält immer Nummer 0
- Eingabealphabet Σ soll immer $\{0, 1\}$ sein
- Blank-Zeichen wird immer mit 2 nummeriert

Kodierung einer Turingmaschine als Wort (1)

Schritt 1: Kodierung einer Turingmaschine als Wort über $\{0, 1, \#\}$

Erinnerung: Turingmaschine $M = \langle Z, \Sigma, \Gamma, \delta, z_0, \square, E \rangle$

Idee:

- Zustände in Z und Symbole in Γ nummerieren und als Zahlen $0, 1, 2, \dots$ betrachten
- Anfangszustand erhält immer Nummer 0
- Eingabealphabet Σ soll immer $\{0, 1\}$ sein
- Blank-Zeichen wird immer mit 2 nummeriert

Dann reicht es aus, **nur** δ explizit zu kodieren:

- Z : alle in der Kodierung von δ erwähnten Zustände
- E : alle Zustände, die nicht links in einer δ -Regel auftauchen
- $\Gamma = \{0, 1, \square, a_3, a_4, \dots, a_k\}$, wobei k die grösste Symbolnummer ist, die in den δ -Regeln erwähnt wird

Kodierung einer Turingmaschine als Wort (2)

Kodierung der Regeln:

- Sei $\delta(z_i, a_j) = \langle z_{i'}, a_{j'}, y \rangle$ eine Regel in δ , wobei die Indizes i, i', j, j' der Nummerierung der Zustände/Symbole entsprechen und $y \in \{L, R, N\}$.
- Kodiere diese Regel als
 $w_{i,j,i',j',y} = ##\text{bin}(i)\#\text{bin}(j)\#\text{bin}(i')\#\text{bin}(j')\#\text{bin}(m)$, wobei
$$m = \begin{cases} 0, & y = L \\ 1, & y = R \\ 2, & y = N \end{cases}$$
- Für jede Regel in δ erhalten wir ein solches Wort.
- Alle diese Wörter hintereinander (in beliebiger Reihenfolge) kodieren die Turingmaschine.

Kodierung einer Turingmaschine als Wort (3)

Schritt 2: Transformation in Wort über $\{0, 1\}$ mit Abbildung

$$0 \mapsto 00$$

$$1 \mapsto 01$$

$$\# \mapsto 11$$

Turingmaschine kann aus ihrer Kodierung rekonstruiert werden.

Wie?

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Kodierung einer Turingmaschine als Wort (III)

Beispiel (Schritt 1)

$\delta(z_2, a_3) = (z_0, a_2, N)$ wird zu ##10#11#0#10#10

$\delta(z_4, a_1) = (z_3, a_2, L)$ wird zu ##100#1#11#10#0

Beispiel (Schritt 2)

##10#11#0#10#10##1#1#11#10#0

11110100110101110011010011010011110111011101011101001100

Anmerkung: Wir können das Kodierungswort auch (eindeutig; warum?) als **Zahl** auffassen, die diese TM nummeriert.

Dies ist nicht für das Halteproblem wichtig, aber in anderen Kontexten, wo wir mit Zahlen statt Wörtern operieren.

Turingmaschine eines Wortes

Ziel: Funktion, die ein beliebiges Wort in $\{0, 1\}^*$ auf eine Turingmaschine abbildet.

Problem: Nicht jedes Wort in $\{0, 1\}^*$ ist Kodierung einer Turingmaschine.

Lösung: Sei $\hat{M}$ eine beliebige, feste Turingmaschine (zum Beispiel eine TM, die immer sofort hält). Dann:

Definition (Turingmaschine eines Wortes)

Für $w \in \{0, 1\}^*$ ist

$$M_w = \begin{cases} M, & \text{falls } w \text{ Codewort von } M \text{ ist} \\ \hat{M} & \text{sonst} \end{cases}$$

Spezielles Halteproblem

Unsere Vorarbeiten sind nun fertig und wir können definieren:

Definition (spezielles Halteproblem)

Das **spezielle Halteproblem** oder **Selbstanwendbarkeitsproblem** ist die Sprache

$$K = \{w \in \{0, 1\}^* \mid M_w \text{ angesetzt auf } w \text{ hält}\}.$$

$\rightsquigarrow$ w spielt Doppelrolle zur Kodierung der Turingmaschine
und als Eingabe für die kodierte Turingmaschine

Unentscheidbarkeit des speziellen Halteproblems (1)

Satz (Unentscheidbarkeit des speziellen Halteproblems)

Das spezielle Halteproblem ist nicht entscheidbar.

Sei K also entscheidbar. Dann ist χ_K berechenbar (Warum?).

Sei M eine Turingmaschine, die χ_K berechnet, also für gegebenes Wort w auf das Band 0 oder 1 schreibt (je nach dem, ob $w \in K$) und dann hält. ...

Unentscheidbarkeit des speziellen Halteproblems (1)

Satz (Unentscheidbarkeit des speziellen Halteproblems)

Das spezielle Halteproblem ist nicht entscheidbar.

Beweis.

Widerspruchsbeweis: wir nehmen an, das spezielle Halteproblem K wäre entscheidbar und leiten einen Widerspruch her.

Unentscheidbarkeit des speziellen Halteproblems (1)

Satz (Unentscheidbarkeit des speziellen Halteproblems)

Das spezielle Halteproblem ist nicht entscheidbar.

Beweis.

Widerspruchsbeweis: wir nehmen an, das spezielle Halteproblem K wäre entscheidbar und leiten einen Widerspruch her.

Sei K also entscheidbar. Dann ist χ_K berechenbar (**Warum?**).

Unentscheidbarkeit des speziellen Halteproblems (1)

Satz (Unentscheidbarkeit des speziellen Halteproblems)

Das spezielle Halteproblem ist nicht entscheidbar.

Beweis.

Widerspruchsbeweis: wir nehmen an, das spezielle Halteproblem K wäre entscheidbar und leiten einen Widerspruch her.

Sei K also entscheidbar. Dann ist χ_K berechenbar (**Warum?**).

Sei M eine Turingmaschine, die χ_K berechnet, also für gegebenes Wort w auf das Band 0 oder 1 schreibt (je nach dem, ob $w \in K$) und dann hält.

...

Unentscheidbarkeit des speziellen Halteproblems (2)

Beweis (Fortsetzung).

Konstruiere neue Maschine M' wie folgt:

- 1 Führe M auf der Eingabe w aus.
- 2 Wenn 0 auf dem Band steht: halte an.
- 3 Ansonsten: Endlosschleife.

Unentscheidbarkeit des speziellen Halteproblems (2)

Beweis (Fortsetzung).

Konstruiere neue Maschine M' wie folgt:

- 1 Führe M auf der Eingabe w aus.
- 2 Wenn 0 auf dem Band steht: halte an.
- 3 Ansonsten: Endlosschleife.

Sei w' Codewort für M' . Wie verhält sich M' bei Eingabe w' ?

Unentscheidbarkeit des speziellen Halteproblems (2)

Beweis (Fortsetzung).

Konstruiere neue Maschine M' wie folgt:

- 1 Führe M auf der Eingabe w aus.
- 2 Wenn 0 auf dem Band steht: halte an.
- 3 Ansonsten: Endlosschleife.

Sei w' Codewort für M' . Wie verhält sich M' bei Eingabe w' ?

M' angesetzt auf w' hält

gdw. M angesetzt auf w gibt 0 aus

Unentscheidbarkeit des speziellen Halteproblems (2)

Beweis (Fortsetzung).

Konstruiere neue Maschine M' wie folgt:

- 1 Führe M auf der Eingabe w aus.
- 2 Wenn 0 auf dem Band steht: halte an.
- 3 Ansonsten: Endlosschleife.

Sei w' Codewort für M' . Wie verhält sich M' bei Eingabe w' ?

M' angesetzt auf w' hält

gdw. M angesetzt auf w gibt 0 aus

gdw. $\chi_K(w') = 0$

Unentscheidbarkeit des speziellen Halteproblems (2)

Beweis (Fortsetzung).

Konstruiere neue Maschine M' wie folgt:

- 1 Führe M auf der Eingabe w aus.
- 2 Wenn 0 auf dem Band steht: halte an.
- 3 Ansonsten: Endlosschleife.

Sei w' Codewort für M' . Wie verhält sich M' bei Eingabe w' ?

M' angesetzt auf w' hält

gdw. M angesetzt auf w' gibt 0 aus

gdw. $\chi_K(w') = 0$

gdw. $w' \notin K$

Unentscheidbarkeit des speziellen Halteproblems (2)

Beweis (Fortsetzung).

Konstruiere neue Maschine M' wie folgt:

- 1 Führe M auf der Eingabe w aus.
- 2 Wenn 0 auf dem Band steht: halte an.
- 3 Ansonsten: Endlosschleife.

Sei w' Codewort für M' . **Wie verhält sich M' bei Eingabe w' ?**

M' angesetzt auf w' hält

gdw. M angesetzt auf w' gibt 0 aus

gdw. $\chi_K(w') = 0$

gdw. $w' \notin K$

gdw. $M_{w'}$ angesetzt auf w' hält nicht

Unentscheidbarkeit des speziellen Halteproblems (2)

Beweis (Fortsetzung).

Konstruiere neue Maschine M' wie folgt:

- 1 Führe M auf der Eingabe w aus.
- 2 Wenn 0 auf dem Band steht: halte an.
- 3 Ansonsten: Endlosschleife.

Sei w' Codewort für M' . Wie verhält sich M' bei Eingabe w' ?

M' angesetzt auf w' hält

gdw. M angesetzt auf w' gibt 0 aus

gdw. $\chi_K(w') = 0$

gdw. $w' \notin K$

gdw. $M_{w'}$ angesetzt auf w' hält nicht

gdw. M' angesetzt auf w' hält nicht

Widerspruch! Damit ist der Beweis erbracht.



Spezielles Halteproblem: Diskussion

- Wir kennen nun ein konkretes unentscheidbares Problem.
- Das Problem ist allerdings recht künstlich:
wie oft möchte man ein Programm auf sich selbst anwenden?
- Wir werden sehen, dass man aus der Unentscheidbarkeit des speziellen Halteproblems leicht **weitere** (praktischere) Unentscheidbarkeitsergebnisse folgern kann.
- Zentraler Begriff dabei ist die **Reduktion**:
Zurückführen eines neuen Problems auf ein bereits bekanntes.

Reduzierbarkeit

Reduzierbarkeit: Definition

Definition (reduzierbar, Reduktion)

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen,
und sei $f : \Sigma^* \rightarrow \Gamma^*$ eine **totale** und **berechenbare** Funktion,
so dass für alle $x \in \Sigma^*$ gilt:

$$x \in A \quad \text{genau dann, wenn} \quad f(x) \in B.$$

Dann heisst A **auf B reduzierbar** (in Symbolen: $A \leq B$)
und f heisst **Reduktion von A auf B** .

Reduzierbarkeit: Eigenschaften (1)

Satz (Ordnungseigenschaften von Reduktionen)

Die Relation „ $\leq$ “ ist eine schwache Halbordnung:

- 1 Für alle Sprachen A gilt:
 $A \leq A$ (*Reflexivität*)
- 2 Für alle Sprachen A, B, C gilt:
Wenn $A \leq B$ und $B \leq C$, dann $A \leq C$ (*Transitivität*)

Reduzierbarkeit: Eigenschaften (1)

Satz (Ordnungseigenschaften von Reduktionen)

Die Relation „ $\leq$ “ ist eine schwache Halbordnung:

- ① Für alle Sprachen A gilt:
 $A \leq A$ (*Reflexivität*)
- ② Für alle Sprachen A, B, C gilt:
Wenn $A \leq B$ und $B \leq C$, dann $A \leq C$ (*Transitivität*)

Beweis.

zu 1.: $f(x) = x$ ist eine Reduktion von A auf A ,
da total und berechenbar und $x \in A$ gdw. $f(x) \in A$

zu 2.: $\rightsquigarrow$ Übungen



Reduzierbarkeit: Eigenschaften (2)

Satz (Reduzierbarkeit vs. (Semi-) Entscheidbarkeit)

Seien A und B Sprachen mit $A \leq B$. Dann gilt:

- ① *Wenn B entscheidbar ist, ist auch A entscheidbar.*
- ② *Wenn B semi-entscheidbar ist, ist auch A semi-entscheidbar.*
- ③ *Wenn A nicht entscheidbar ist,
dann ist auch B nicht entscheidbar.*
- ④ *Wenn A nicht semi-entscheidbar ist,
dann ist auch B nicht semi-entscheidbar.*

↪ Wir verwenden im folgenden 3., um für weitere Probleme die Unentscheidbarkeit zu zeigen.

Reduzierbarkeit: Eigenschaften (3)

Beweis.

zu 1.:

- B entscheidbar $\rightsquigarrow \chi_B$ berechenbar

Reduzierbarkeit: Eigenschaften (3)

Beweis.

zu 1.:

- B entscheidbar $\rightsquigarrow \chi_B$ berechenbar
- $A \leq B \rightsquigarrow$ es gibt Reduktion f von A auf B ;
 f total und berechenbar

Reduzierbarkeit: Eigenschaften (3)

Beweis.

zu 1.:

- B entscheidbar $\rightsquigarrow \chi_B$ berechenbar
- $A \leq B \rightsquigarrow$ es gibt Reduktion f von A auf B ;
 f total und berechenbar
- Es gilt für alle $x \in \Sigma^*$:

$$\begin{aligned}\chi_A(x) &= \begin{cases} 1 & \text{falls } x \in A \\ 0 & \text{falls } x \notin A \end{cases} \\ &= \begin{cases} 1 & \text{falls } f(x) \in B \\ 0 & \text{falls } f(x) \notin B \end{cases} \\ &= \chi_B(f(x)) = (\chi_B \circ f)(x).\end{aligned}$$

Reduzierbarkeit: Eigenschaften (3)

Beweis.

zu 1.:

- B entscheidbar $\rightsquigarrow \chi_B$ berechenbar
- $A \leq B \rightsquigarrow$ es gibt Reduktion f von A auf B ;
 f total und berechenbar
- Es gilt für alle $x \in \Sigma^*$:

$$\begin{aligned}\chi_A(x) &= \begin{cases} 1 & \text{falls } x \in A \\ 0 & \text{falls } x \notin A \end{cases} \\ &= \begin{cases} 1 & \text{falls } f(x) \in B \\ 0 & \text{falls } f(x) \notin B \end{cases} \\ &= \chi_B(f(x)) = (\chi_B \circ f)(x).\end{aligned}$$

- Komposition $\chi_B \circ f$ ist berechenbar

Reduzierbarkeit: Eigenschaften (3)

Beweis.

zu 1.:

- B entscheidbar $\rightsquigarrow \chi_B$ berechenbar
- $A \leq B \rightsquigarrow$ es gibt Reduktion f von A auf B ;
 f total und berechenbar
- Es gilt für alle $x \in \Sigma^*$:

$$\begin{aligned}\chi_A(x) &= \begin{cases} 1 & \text{falls } x \in A \\ 0 & \text{falls } x \notin A \end{cases} \\ &= \begin{cases} 1 & \text{falls } f(x) \in B \\ 0 & \text{falls } f(x) \notin B \end{cases} \\ &= \chi_B(f(x)) = (\chi_B \circ f)(x).\end{aligned}$$

- Komposition $\chi_B \circ f$ ist berechenbar

$\rightsquigarrow \chi_A$ berechenbar $\rightsquigarrow A$ entscheidbar

...

Reduzierbarkeit: Eigenschaften (4)

Beweis (Fortsetzung).

zu 1. (alternative Formulierung):

Der folgende Algorithmus berechnet $\chi_A(x)$ für Eingabe x :

$y := f(x)$

result := $\chi_B(y)$

OUTPUT(result)

Reduzierbarkeit: Eigenschaften (4)

Beweis (Fortsetzung).

zu 1. (alternative Formulierung):

Der folgende Algorithmus berechnet $\chi_A(x)$ für Eingabe x :

```
y := f(x)
result :=  $\chi_B(y)$ 
OUTPUT(result)
```

zu 2.: wie 1. mit χ' statt χ und mit *undefiniert* statt 0

Reduzierbarkeit: Eigenschaften (4)

Beweis (Fortsetzung).

zu 1. (alternative Formulierung):

Der folgende Algorithmus berechnet $\chi_A(x)$ für Eingabe x :

```
y := f(x)
result :=  $\chi_B(y)$ 
OUTPUT(result)
```

zu 2.: wie 1. mit χ' statt χ und mit *undefiniert* statt 0

zu 3.+4.: äquivalent zu 1.+2. (Kontraposition)



Weitere unentscheidbare Probleme

Allgemeines Halteproblem (1)

Definition (allgemeines Halteproblem)

Das **allgemeine Halteproblem** oder **Halteproblem** ist die Sprache

$$H = \{w\#x \in \{0, 1, \#\}^* \mid w, x \in \{0, 1\}^*, M_w \text{ angesetzt auf } x \text{ hält}\}$$

Allgemeines Halteproblem (1)

Definition (allgemeines Halteproblem)

Das **allgemeine Halteproblem** oder **Halteproblem** ist die Sprache

$$H = \{w\#x \in \{0, 1, \#\}^* \mid w, x \in \{0, 1\}^*, M_w \text{ angesetzt auf } x \text{ hält}\}$$

Satz (Unentscheidbarkeit des allgemeinen Halteproblems)

Das allgemeine Halteproblem ist nicht entscheidbar.

Intuition: wenn der Spezialfall K nicht entscheidbar ist,
kann es das allgemeinere Problem H erst recht nicht sein.

Allgemeines Halteproblem (2)

Beweis.

Wir zeigen $K \leq H$ (Erinnerung: K ist das spezielle Halteproblem)

Wir definieren $f : \{0, 1\}^* \rightarrow \{0, 1, \#\}^*$ als $f(w) := w\#w$.

f ist offensichtlich total und berechenbar, und es gilt:

$$w \in K$$

gdw. M_w angesetzt auf w hält

gdw. $w\#w \in H$

gdw. $f(w) \in H$

Allgemeines Halteproblem (2)

Beweis.

Wir zeigen $K \leq H$ (Erinnerung: K ist das spezielle Halteproblem)

Wir definieren $f : \{0, 1\}^* \rightarrow \{0, 1, \#\}^*$ als $f(w) := w\#w$.

f ist offensichtlich total und berechenbar, und es gilt:

$$w \in K$$

gdw. M_w angesetzt auf w hält

gdw. $w\#w \in H$

gdw. $f(w) \in H$

Damit ist f eine Reduktion von K auf H , und wegen der Unentscheidbarkeit von K ist auch H unentscheidbar. □

Halteproblem auf leerem Band (1)

Definition (Halteproblem auf leerem Band)

Das **Halteproblem auf leerem Band** ist die Sprache

$$H_0 = \{w \in \{0, 1\}^* \mid M_w \text{ angesetzt auf } \epsilon \text{ hält}\}.$$

Halteproblem auf leerem Band (1)

Definition (Halteproblem auf leerem Band)

Das **Halteproblem auf leerem Band** ist die Sprache

$$H_0 = \{w \in \{0, 1\}^* \mid M_w \text{ angesetzt auf } \epsilon \text{ hält}\}.$$

Satz (Unentscheidbarkeit des Halteproblems auf leerem Band)

Das Halteproblem auf leerem Band ist nicht entscheidbar.

Halteproblem auf leerem Band (2)

Beweis.

Wir zeigen $H \leq H_0$.

Halteproblem auf leerem Band (2)

Beweis.

Wir zeigen $H \leq H_0$.

Betrachte dazu die Funktion $f : \{0, 1, \#\}^* \rightarrow \{0, 1\}^*$,
die gegeben $z \in \{0, 1, \#\}^*$ das Wort $f(z)$ wie folgt berechnet:

Halteproblem auf leerem Band (2)

Beweis.

Wir zeigen $H \leq H_0$.

Betrachte dazu die Funktion $f : \{0, 1, \#\}^* \rightarrow \{0, 1\}^*$,
die gegeben $z \in \{0, 1, \#\}^*$ das Wort $f(z)$ wie folgt berechnet:

- Teste, ob z von der Form $w\#x$ mit $w, x \in \{0, 1\}^*$ ist.
- Falls nein, liefere irgendein Wort zurück, das nicht in H_0 liegt (z.B. Kodierung einer TM, die sofort in Endlosschleife geht).
- Falls ja, zerlege z in w und x .

...

Halteproblem auf leerem Band (2)

Beweis.

Wir zeigen $H \leq H_0$.

Betrachte dazu die Funktion $f : \{0, 1, \#\}^* \rightarrow \{0, 1\}^*$,
die gegeben $z \in \{0, 1, \#\}^*$ das Wort $f(z)$ wie folgt berechnet:

- Teste, ob z von der Form $w\#x$ mit $w, x \in \{0, 1\}^*$ ist.
- Falls nein, liefere irgendein Wort zurück, das nicht in H_0 liegt (z.B. Kodierung einer TM, die sofort in Endlosschleife geht).
- Falls ja, zerlege z in w und x .
- Dekodiere w zu einer TM M_2 .

...

Halteproblem auf leerem Band (3)

Beweis (Fortsetzung).

- Konstruiere eine TM M_1 , die sich wie folgt verhält:
 - falls Eingabe leer: schreibe x auf das Band und bewege Kopf auf erstes Zeichen von x (falls $x \neq \epsilon$); dann halte
 - ansonsten halte sofort

Halteproblem auf leerem Band (3)

Beweis (Fortsetzung).

- Konstruiere eine TM M_1 , die sich wie folgt verhält:
 - falls Eingabe leer: schreibe x auf das Band und bewege Kopf auf erstes Zeichen von x (falls $x \neq \epsilon$); dann halte
 - ansonsten halte sofort
- Konstruiere TM M , die erst M_1 und dann M_2 ausführt.

Halteproblem auf leerem Band (3)

Beweis (Fortsetzung).

- Konstruiere eine TM M_1 , die sich wie folgt verhält:
 - falls Eingabe leer: schreibe x auf das Band und bewege Kopf auf erstes Zeichen von x (falls $x \neq \epsilon$); dann halte
 - ansonsten halte sofort
- Konstruiere TM M , die erst M_1 und dann M_2 ausführt.
- Liefere Kodierung von M zurück.

Halteproblem auf leerem Band (3)

Beweis (Fortsetzung).

- Konstruiere eine TM M_1 , die sich wie folgt verhält:
 - falls Eingabe leer: schreibe x auf das Band und bewege Kopf auf erstes Zeichen von x (falls $x \neq \epsilon$); dann halte
 - ansonsten halte sofort
- Konstruiere TM M , die erst M_1 und dann M_2 ausführt.
- Liefere Kodierung von M zurück.

f ist total und (mit etwas Mühe) berechenbar. Es gilt:

$z \in H$ gdw. $z = w\#x$ und M_w angesetzt auf x hält
gdw. $M_{f(z)}$ hält auf leerem Band
gdw. $f(z) \in H_0$

$\rightsquigarrow H \leq H_0 \rightsquigarrow H_0$ unentscheidbar



Satz von Rice (1)

- Wir haben für eine Reihe von (verwandten) Problemen gezeigt, dass sie unentscheidbar sind:
 - spezielles Halteproblem K
 - allgemeines Halteproblem H
 - Halteproblem auf leerem Band H_0
- Viele weitere Resultate dieser Art könnten bewiesen werden.
- Stattdessen beweisen wir ein sehr viel mächtigeres Ergebnis, den **Satz von Rice**, der zeigt, dass eine sehr grosse Klasse unterschiedlicher Probleme unentscheidbar sind.
- Der Satz von Rice lässt sich informell zusammenfassen als: **jede** Frage darüber, **was** eine gegebene Turingmaschine berechnet, ist unentscheidbar.

Satz von Rice (2)

Satz (Satz von Rice)

Sei $\mathcal{R}$ die Klasse aller Turing-berechenbaren Funktionen.

*Sei S eine **beliebige** Teilmenge von $\mathcal{R}$ ausser $S = \emptyset$ oder $S = \mathcal{R}$.*

Dann ist die Sprache

$$C(S) = \{w \in \{0,1\}^* \mid \text{die von } M_w \text{ berechnete Funktion liegt in } S\}$$

unentscheidbar.

Frage: warum die Forderung $S \neq \emptyset$ und $S \neq \mathcal{R}$?

Satz von Rice (3)

Beweis.

Sei Ω die überall undefinierte Funktion.

Satz von Rice (3)

Beweis.

Sei Ω die überall undefinierte Funktion.

Fallunterscheidung:

Fall 1: $\Omega \in \mathcal{S}$

Satz von Rice (3)

Beweis.

Sei Ω die überall undefinierte Funktion.

Fallunterscheidung:

Fall 1: $\Omega \in \mathcal{S}$

Sei $q \in \mathcal{R} \setminus \mathcal{S}$ eine beliebige Turing-berechenbare Funktion ausserhalb von $\mathcal{S}$ (existiert, da $\mathcal{S} \subseteq \mathcal{R}$ und $\mathcal{S} \neq \mathcal{R}$).

Satz von Rice (3)

Beweis.

Sei Ω die überall undefinierte Funktion.

Fallunterscheidung:

Fall 1: $\Omega \in \mathcal{S}$

Sei $q \in \mathcal{R} \setminus \mathcal{S}$ eine beliebige Turing-berechenbare Funktion ausserhalb von $\mathcal{S}$ (existiert, da $\mathcal{S} \subseteq \mathcal{R}$ und $\mathcal{S} \neq \mathcal{R}$).

Sei Q eine Turingmaschine, die q berechnet.

...

Satz von Rice (4)

Beweis (Fortsetzung).

Betrachte Funktion $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$,

wobei $f(w)$ wie folgt definiert ist:

- Konstruiere TM M , die sich auf Eingabe y zunächst verhält wie M_w auf leerem Band (unabhängig von y).
- Anschliessend (falls diese Berechnung hält!) löscht M Bandinhalt, stellt Startkonfiguration von Q auf Eingabe y her und simuliert dann Q .
- $f(w)$ ist die Kodierung dieser TM M

Satz von Rice (4)

Beweis (Fortsetzung).

Betrachte Funktion $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$,

wobei $f(w)$ wie folgt definiert ist:

- Konstruiere TM M , die sich auf Eingabe y zunächst verhält wie M_w auf leerem Band (unabhängig von y).
- Anschliessend (falls diese Berechnung hält!) löscht M Bandinhalt, stellt Startkonfiguration von Q auf Eingabe y her und simuliert dann Q .
- $f(w)$ ist die Kodierung dieser TM M

f ist total und berechenbar.

...

Satz von Rice (5)

Beweis (Fortsetzung).

Welche Funktion berechnet die durch $f(w)$ kodierte TM?

Satz von Rice (5)

Beweis (Fortsetzung).

Welche Funktion berechnet die durch $f(w)$ kodierte TM?

$M_{f(w)}$ berechnet $\begin{cases} \Omega & \text{falls } M_w \text{ auf leerem Band nicht hält} \\ q & \text{sonst} \end{cases}$

Satz von Rice (5)

Beweis (Fortsetzung).

Welche Funktion berechnet die durch $f(w)$ kodierte TM?

$M_{f(w)}$ berechnet $\begin{cases} \Omega & \text{falls } M_w \text{ auf leerem Band nicht hält} \\ q & \text{sonst} \end{cases}$

Es gilt für alle Wörter $w \in \{0, 1\}^*$:

$w \in H_0 \implies M_w$ hält auf leerem Band

$\implies M_{f(w)}$ berechnet die Funktion q

$\implies$ die von $M_{f(w)}$ berechnete Funktion liegt nicht in $\mathcal{S}$

$\implies f(w) \notin C(\mathcal{S})$

Satz von Rice (6)

Beweis (Fortsetzung).

Ferner gilt:

- $w \notin H_0 \implies M_w$ hält nicht auf leerem Band
- $\implies M_{f(w)}$ berechnet die Funktion Ω
- $\implies$ die von $M_{f(w)}$ berechnete Funktion liegt in $\mathcal{S}$
- $\implies f(w) \in C(\mathcal{S})$

Satz von Rice (6)

Beweis (Fortsetzung).

Ferner gilt:

- $w \notin H_0 \implies M_w$ hält nicht auf leerem Band
- $\implies M_{f(w)}$ berechnet die Funktion Ω
- $\implies$ die von $M_{f(w)}$ berechnete Funktion liegt in $\mathcal{S}$
- $\implies f(w) \in C(\mathcal{S})$

Zusammen folgt: $w \notin H_0$ gdw. $f(w) \in C(\mathcal{S})$,
also $w \in \bar{H}_0$ gdw. $f(w) \in C(\mathcal{S})$.

Satz von Rice (6)

Beweis (Fortsetzung).

Ferner gilt:

- $w \notin H_0 \implies M_w$ hält nicht auf leerem Band
- $\implies M_{f(w)}$ berechnet die Funktion Ω
- $\implies$ die von $M_{f(w)}$ berechnete Funktion liegt in $\mathcal{S}$
- $\implies f(w) \in C(\mathcal{S})$

Zusammen folgt: $w \notin H_0$ gdw. $f(w) \in C(\mathcal{S})$,
also $w \in \bar{H}_0$ gdw. $f(w) \in C(\mathcal{S})$.

Somit ist f eine Reduktion von $\bar{H}_0$ nach $C(\mathcal{S})$.

Satz von Rice (6)

Beweis (Fortsetzung).

Ferner gilt:

- $w \notin H_0 \implies M_w$ hält nicht auf leerem Band
- $\implies M_{f(w)}$ berechnet die Funktion Ω
- $\implies$ die von $M_{f(w)}$ berechnete Funktion liegt in $\mathcal{S}$
- $\implies f(w) \in C(\mathcal{S})$

Zusammen folgt: $w \notin H_0$ gdw. $f(w) \in C(\mathcal{S})$,
also $w \in \bar{H}_0$ gdw. $f(w) \in C(\mathcal{S})$.

Somit ist f eine Reduktion von $\bar{H}_0$ nach $C(\mathcal{S})$.

Da H_0 unentscheidbar ist, ist auch $\bar{H}_0$ unentscheidbar.

Satz von Rice (6)

Beweis (Fortsetzung).

Ferner gilt:

- $w \notin H_0 \implies M_w$ hält nicht auf leerem Band
- $\implies M_{f(w)}$ berechnet die Funktion Ω
- $\implies$ die von $M_{f(w)}$ berechnete Funktion liegt in $\mathcal{S}$
- $\implies f(w) \in C(\mathcal{S})$

Zusammen folgt: $w \notin H_0$ gdw. $f(w) \in C(\mathcal{S})$,
also $w \in \bar{H}_0$ gdw. $f(w) \in C(\mathcal{S})$.

Somit ist f eine Reduktion von $\bar{H}_0$ nach $C(\mathcal{S})$.

Da H_0 unentscheidbar ist, ist auch $\bar{H}_0$ unentscheidbar.

Es folgt, dass $C(\mathcal{S})$ unentscheidbar ist.

...

Satz von Rice (7)

Beweis (Fortsetzung).

Fall 2: $\Omega \notin \mathcal{S}$

Analog zu Fall 1, aber wähle diesmal $q \in \mathcal{S}$.

Die entsprechende Funktion f ist dann eine Reduktion von H_0 auf $C(\mathcal{S})$.

Auch in diesem Fall folgt, dass $C(\mathcal{S})$ unentscheidbar ist. □

Satz von Rice (8)

Hat sich die Mühe gelohnt?

Wir wissen nun z.B. sofort, dass die folgenden (informell formulierten) Probleme alle unentscheidbar sind:

- Berechnet eine gegebene TM eine konstante Funktion?
- Berechnet eine gegebene TM eine totale Funktion (d.h. hält sie immer, und zwar in einer „korrekten“ Konfiguration)?
- Ist die Ausgabe einer gegebenen TM immer länger als ihre Eingabe?
- Berechnet eine gegebene TM die Identitätsfunktion?
- Berechnet eine gegebene TM die berechenbare Funktion f ?
- ...