LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Formale Sprachen und Komplexität
Theoretische Informatik für Medieninformatiker
Sommersemester 2023

FSK
Zentralübung 6

Prof. Dr. Jasmin Blanchette

Lehr- und Forschungseinheit für
Theoretische Informatik

Stand: 28. Juni 2023
Folien ursprünglich von PD Dr. David Sabel

Plan für Heute

- Primitive Rekursion und μ-Rekursion
- Berechenbarkeit
- Halteproblem, Reduktion, Satz von Rice

Wiederholung: Primitiv rekursive Funktion

Definition

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ ist **primitiv rekursiv**, wenn sie der folgenden induktiven Definition genügt:

- Jede konstante Funktion $f(x_1, \ldots, x_k) = c \in \mathbb{N}$ ist primitiv rekursiv.
- Die Projektionsfunktionen $\pi_i^k(x_1, \ldots, x_k) = x_i$ sind primitiv rekursiv.
- Die Nachfolgerfunktion $succ(x) = x + 1$ ist primitiv rekursiv.
- Komposition/Einsetzung: Wenn $g : \mathbb{N}^m \to \mathbb{N}$ und für $i = 1, \ldots, m$: $h_i : \mathbb{N}^k \to \mathbb{N}$ primitiv rekursiv sind, dann ist auch f mit $f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k))$ primitiv rekursiv.
- Primitive Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ primitiv rekursiv sind, dann ist auch f mit

  $$f(x_1, \ldots, x_k) = \begin{cases} g(x_2, \ldots, x_k), & \text{wenn } x_1 = 0 \\ h(f(x_1 - 1, x_2, \ldots, x_k), x_1 - 1, x_2, \ldots, x_k), & \text{sonst} \end{cases}$$

  primitiv rekursiv.

Wiederholung: Wir wissen bereits

- $add(x, y) = x + y$ ist primitiv rekursiv
- $mult(x, y) = x * y$ ist primitiv rekursiv
- $sub(x, y) = \begin{cases} x - y, & \text{wenn } x \geq y \\ 0, & \text{sonst} \end{cases}$
  ist primitiv rekursiv
- Vertauschen, Verdoppeln, Entfernen von Argumenten ist primitiv rekursiv
- Rekursionsabstieg muss nicht über das erste, sondern kann auch über das i-te Argument erfolgen

Aufgabe

Zeige, dass die Summenfunktion mit $sum(0) = 0$ und $sum(n) = sum(n-1) + n$ für $n > 0$ primitiv rekursiv ist.

Zur Erinnerung

- Primitive Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ primitiv rekursiv, dann ist auch f primitiv rekursiv, wobei:

$$f(x_1, \dots, x_k) = \begin{cases} g(x_2, \dots, x_k), & \text{wenn } x_1 = 0 \\ h(f(x_1 - 1, x_2, \dots, x_k), x_1 - 1, x_2, \dots, x_k), & \text{sonst} \end{cases}$$

Primitiv rekursive Definition: $sum(x) = \begin{cases} g() & \text{falls } x = 0 \\ h(sum(x-1), x-1) & \text{sonst} \end{cases}$

wobei:

- $g() = ?$
- $h(x, y) = ?$

Aufgabe

Zeige, dass die Summenfunktion mit $sum(0) = 0$ und $sum(n) = sum(n-1) + n$ für $n > 0$ primitiv rekursiv ist.

Zur Erinnerung

- Primitive Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ primitiv rekursiv, dann ist auch f primitiv rekursiv, wobei:

$$f(x_1, \ldots, x_k) = \begin{cases} g(x_2, \ldots, x_k), & \text{wenn } x_1 = 0 \\ h(f(x_1 - 1, x_2, \ldots, x_k), x_1 - 1, x_2, \ldots, x_k), & \text{sonst} \end{cases}$$

Primitiv rekursive Definition: $sum(x) = \begin{cases} g() & \text{falls } x = 0 \\ h(sum(x-1), x-1) & \text{sonst} \end{cases}$

wobei:

- $g() = 0$
- $h(x, y) = ?$

Aufgabe

Zeige, dass die Summenfunktion mit $sum(0) = 0$ und $sum(n) = sum(n-1) + n$ für $n > 0$ primitiv rekursiv ist.

Zur Erinnerung

- Primitive Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ primitiv rekursiv, dann ist auch f primitiv rekursiv, wobei:
$$f(x_1, \ldots, x_k) = \begin{cases} g(x_2, \ldots, x_k), & \text{wenn } x_1 = 0 \\ h(f(x_1 - 1, x_2, \ldots, x_k), x_1 - 1, x_2, \ldots, x_k), & \text{sonst} \end{cases}$$

Primitiv rekursive Definition: $sum(x) = \begin{cases} g() & \text{falls } x = 0 \\ h(sum(x-1), x-1) & \text{sonst} \end{cases}$

wobei:

- $g() = 0$
- $h(x, y) = add(x, add(y, 1))$

Aufgabe

Zeige, dass die Summenfunktion mit $sum(0) = 0$ und $sum(n) = sum(n-1) + n$ für $n > 0$ primitiv rekursiv ist.

Zur Erinnerung

- Primitive Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ primitiv rekursiv, dann ist auch f primitiv rekursiv, wobei:

$$f(x_1, \ldots, x_k) = \begin{cases} g(x_2, \ldots, x_k), & \text{wenn } x_1 = 0 \\ h(f(x_1 - 1, x_2, \ldots, x_k), x_1 - 1, x_2, \ldots, x_k), & \text{sonst} \end{cases}$$

Primitiv rekursive Definition: $sum(x) = \begin{cases} g() & \text{falls } x = 0 \\ h(sum(x-1), x-1) & \text{sonst} \end{cases}$

wobei:

- $g() = 0$
- $h(x, y) = add(x, add(y, 1)) = add(\pi_1^2(x, y), r(x, y))$ mit
- $r(a, b) = add(\pi_2^2(a, b), const_1(a, b))$
- $const_1(a, b) = 1$

Aufgabe

Zeige, dass

$$max(x, y) = \begin{cases} x, & \text{falls } x \geq y \\ y, & \text{sonst} \end{cases}$$

primitiv rekursiv ist.

Aufgabe

Zeige, dass

$$max(x,y) = \begin{cases} x, & \text{falls } x \geq y \\ y, & \text{sonst} \end{cases}$$

primitiv rekursiv ist.

Es gilt $max(x,y) = add(x, sub(y,x))$, denn:

- Wenn $x \geq y$, dann $sub(y,x) = 0$ und $add(x,0) = x$
- Wenn $x < y$, dann $sub(y,x) = y - x$ und $add(x, y - x) = y$
- Da add, sub und Komposition primitiv rekursiv, ist max auch primitiv rekursiv.

Aufgabe

Zeige, dass

$$absdiff(x, y) = |x - y|$$

primitiv rekursiv ist.

Aufgabe

Zeige, dass

$$absdiff(x, y) = |x - y|$$

primitiv rekursiv ist.

Beachte:

	$sub(x, y)$	$sub(y, x)$	$absdiff(x, y)$
$x \leq y$	0	$y - x$	$y - x$
$x > y$	$x - y$	0	$x - y$

Aufgabe

Zeige, dass

$$absdiff(x, y) = |x - y|$$

primitiv rekursiv ist.

Beachte:

	$sub(x, y)$	$sub(y, x)$	$absdiff(x, y)$
$x \leq y$	0	$y - x$	$y - x$
$x > y$	$x - y$	0	$x - y$

Daher $absdiff(x, y) = add(sub(x, y), sub(y, x))$.

Da *sub*, *add* und Komposition primitiv rekursiv sind, ist auch *absdiff* primitiv rekursiv.

Primitiv rekursive Prädikate

- Primitiv rekursive Funktionen $\mathbb{N}^k \to \mathbb{N}$
- Prädikate liefern eigentlich wahr oder falsch.
- Wir verwenden $P : \mathbb{N}^k \to \{0, 1\}$ für Prädikate.
- Das passt immer noch zu den primitiv rekursiven Funktionen.

Aufgabe

Zeige, dass das Prädikat

$$equal(x, y) = \begin{cases} 1, & \text{wenn } x = y \\ 0, & \text{sonst} \end{cases}$$

primitiv rekursiv ist.

Beobachtung: $equal(x, y)$ gilt g.d.w. $|x - y| = 0$

- $equal(x, y) = eq0?(absdiff(x, y))$
- wobei $eq0?(x) = \begin{cases} 1, & \text{wenn } x = 0 \\ 0, & \text{sonst} \end{cases}$
- $eq0?(x) = \begin{cases} g(), & \text{wenn } x = 0 \\ h(eq0?(x-1), x-1), & \text{wenn } x > 0 \end{cases}$

  mit ?

Aufgabe

Zeige, dass das Prädikat

$$equal(x, y) = \begin{cases} 1, & \text{wenn } x = y \\ 0, & \text{sonst} \end{cases}$$

primitiv rekursiv ist.

Beobachtung: $equal(x, y)$ gilt g.d.w. $|x - y| = 0$

- $equal(x, y) = eq0?(absdiff(x, y))$
- wobei $eq0?(x) = \begin{cases} 1, & \text{wenn } x = 0 \\ 0, & \text{sonst} \end{cases}$
- $eq0?(x) = \begin{cases} g(), & \text{wenn } x = 0 \\ h(eq0?(x-1), x-1), & \text{wenn } x > 0 \end{cases}$

  mit $g() = 1$ und $h(x, y) = 0$

Wiederholung: μ-Rekursion

Definition μ-Operator

Sei $h : \mathbb{N}^{k+1} \to \mathbb{N}$ eine (partielle oder totale) Funktion. Dann ist $(\mu h) : \mathbb{N}^k \to \mathbb{N}$ definiert als

$$(\mu h)(x_1, \ldots, x_k) = \begin{cases} n, & \text{falls } h(n, x_1, \ldots, x_k) = 0 \text{ und für} \\ & \text{alle } m < n\text{: } h(m, x_1, \ldots, x_k) \text{ ist} \\ & \text{definiert und } h(m, x_1, \ldots, x_k) > 0 \\ \text{undefiniert}, & \text{sonst} \end{cases}$$

- μ-Operator „sucht“ nach einer kleinsten Nullstelle von h.
- Wenn diese nicht existiert (entweder da h keine Nullstelle hat, oder da h undefiniert ist für Werte, die kleiner als die Nullstelle sind), dann ist auch der μ-Operator angewendet auf h undefiniert.

μ-rekursive Funktionen

Definition

Die Menge aller μ-rekursiven Funktionen sei die kleinste Menge, sodass gilt:

- Jede konstante Funktion $f(x_1, \dots, x_k) = c \in \mathbb{N}$ ist μ-rekursiv.
- Die Projektionsfunktionen $\pi_i^k(x_1, \dots, x_k) = x_i$ sind μ-rekursiv.
- Die Nachfolgerfunktion $succ(x) = x + 1$ ist μ-rekursiv.
- Komposition/Einsetzung: Wenn $g : \mathbb{N}^m \to \mathbb{N}$ und für $i = 1, \dots, m$: $h_i : \mathbb{N}^k \to \mathbb{N}$ μ-rekursiv sind, dann ist auch f mit $f(x_1, \dots, x_k) = g(h_1(x_1, \dots, x_k), \dots, h_m(x_1, \dots, x_k))$ μ-rekursiv.
- Rekursion: Wenn $g : \mathbb{N}^{k-1} \to \mathbb{N}$ und $h : \mathbb{N}^{k+1} \to \mathbb{N}$ μ-rekursiv sind, dann ist

$$f(x_1, \dots, x_k) = \begin{cases} g(x_2, \dots, x_k), & \text{wenn } x_1 = 0 \\ h(f(x_1 - 1, x_2, \dots, x_k), x_1 - 1, x_2, \dots, x_k), & \text{sonst} \end{cases}$$

  auch μ-rekursiv.
- μ-Operator: Wenn $h : \mathbb{N}^{k+1} \to \mathbb{N}$ μ-rekursiv ist, dann ist auch $f = \mu h$ μ-rekursiv.

Aufgabe

Welche Funktion wird durch (μh_i) mit

a) $h_1(x, y) = sub(x, y)$
b) $h_2(x, y) = sub(y, x)$
c) $h_3(x, y) = sub(y, mult(2, x))$
d) $h_4(x, y) = sub(y, mult(x, x))$

berechnet?

a) $h_1(x, y) = sub(x, y)$

a) $h_1(x, y) = sub(x, y)$

- (μh_1) berechnet kleinstes x, sodass $sub(x, y) = 0$

a) $h_1(x, y) = sub(x, y)$

- (μh_1) berechnet kleinstes x, sodass $sub(x, y) = 0$
- das gilt schon für $x = 0$ also $\mu h_1(y) = 0$

a) $h_1(x, y) = sub(x, y)$

- (μh_1) berechnet kleinstes x, sodass $sub(x, y) = 0$
- das gilt schon für $x = 0$ also $\mu h_1(y) = 0$

b) $h_2(x, y) = sub(y, x)$

- (μh_2) berechnet kleinstes x, sodass $sub(y, x) = 0$

a) $h_1(x, y) = sub(x, y)$

- (μh_1) berechnet kleinstes x, sodass $sub(x, y) = 0$
- das gilt schon für $x = 0$ also $\mu h_1(y) = 0$

b) $h_2(x, y) = sub(y, x)$

- (μh_2) berechnet kleinstes x, sodass $sub(y, x) = 0$
- gilt für $x = y$

a) $h_1(x, y) = sub(x, y)$

- (μh_1) berechnet kleinstes x, sodass $sub(x, y) = 0$
- das gilt schon für $x = 0$ also $\mu h_1(y) = 0$

b) $h_2(x, y) = sub(y, x)$

- (μh_2) berechnet kleinstes x, sodass $sub(y, x) = 0$
- gilt für $x = y$
- für alle $m < y$ gilt $sub(y, m) > 0$

a) $h_1(x, y) = sub(x, y)$

- (μh_1) berechnet kleinstes x, sodass $sub(x, y) = 0$
- das gilt schon für $x = 0$ also $\mu h_1(y) = 0$

b) $h_2(x, y) = sub(y, x)$

- (μh_2) berechnet kleinstes x, sodass $sub(y, x) = 0$
- gilt für $x = y$
- für alle $m < y$ gilt $sub(y, m) > 0$
- daher $(\mu h_2)(y) = y$

c) $h_3(x, y) = sub(y, mult(2, x))$

c) $h_3(x, y) = sub(y, mult(2, x))$

- finde kleinstes x sodass $y - 2x \leq 0$

c) $h_3(x, y) = sub(y, mult(2, x))$

- finde kleinstes x sodass $y - 2x \leq 0$
- gilt für $x = \lceil y/2 \rceil$

c) $h_3(x, y) = sub(y, mult(2, x))$

- finde kleinstes x sodass $y - 2x \leq 0$
- gilt für $x = \lceil y/2 \rceil$
- $(\mu h_3)(y) = \lceil y/2 \rceil$

c) $h_3(x, y) = sub(y, mult(2, x))$

- finde kleinstes x sodass $y - 2x \leq 0$
- gilt für $x = \lceil y/2 \rceil$
- $(\mu h_3)(y) = \lceil y/2 \rceil$

d) $h_4(x, y) = sub(y, mult(x, x))$

c) $h_3(x, y) = sub(y, mult(2, x))$

- finde kleinstes x sodass $y - 2x \leq 0$
- gilt für $x = \lceil y/2 \rceil$
- $(\mu h_3)(y) = \lceil y/2 \rceil$

d) $h_4(x, y) = sub(y, mult(x, x))$

- finde kleinstes x sodass $y - x * x \leq 0$

c) $h_3(x, y) = sub(y, mult(2, x))$

- finde kleinstes x sodass $y - 2x \leq 0$
- gilt für $x = \lceil y/2 \rceil$
- $(\mu h_3)(y) = \lceil y/2 \rceil$

d) $h_4(x, y) = sub(y, mult(x, x))$

- finde kleinstes x sodass $y - x * x \leq 0$
- gilt für $x = \lceil \sqrt{y} \rceil$

c) $h_3(x, y) = sub(y, mult(2, x))$

- finde kleinstes x sodass $y - 2x \leq 0$
- gilt für $x = \lceil y/2 \rceil$
- $(\mu h_3)(y) = \lceil y/2 \rceil$

d) $h_4(x, y) = sub(y, mult(x, x))$

- finde kleinstes x sodass $y - x * x \leq 0$
- gilt für $x = \lceil \sqrt{y} \rceil$
- $(\mu h_4)(y) = \lceil \sqrt{y} \rceil$

Turingberechenbarkeit

Sei $bin(n)$ die Binärdarstellung von $n \in \mathbb{N}$.

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt Turingberechenbar, falls es eine (deterministische) Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \Box, E)$ gibt, sodass für alle $n_1, \ldots, n_k, m \in \mathbb{N}$ gilt:

$$f(n_1, \ldots, n_k) = m \text{ g.d.w. } z_0 bin(n_1)\# \ldots \# bin(n_k) \vdash^* \Box \ldots \Box z_e bin(m) \Box \ldots \Box \text{ mit } z_e \in E.$$

Eine Funktion $f : \Sigma^* \to \Sigma^*$ heißt Turingberechenbar, falls es eine (deterministische) Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \Box, E)$ gibt, sodass für alle $u, v \in \Sigma^*$ gilt

$$f(u) = v \text{ g.d.w } start_M(u) \vdash^* \Box \ldots \Box z_e v \Box \ldots \Box \text{ mit } z_e \in E.$$

Zudem:

- $f(n_1, \ldots, n_k) =$ undefiniert g.d.w. $z_0 bin(n_1)\# \ldots \# bin(n_k) \not\vdash^* x z_e y$ mit $z_e \in E$
- $f(u) =$ undefiniert g.d.w. $z_0 u \not\vdash^* x z_e y$ mit $z_e \in E$

Quiz: Chomsky-Hierarchie usw.

Welche der folgenden Aussagen gelten für die von Turingmaschinen akzeptierten Sprachen L?

- A L ist Typ i-Sprache (mit $i \in \{0, 1, 2, 3\}$).
- B Das Wortproblem für alle solchen L ist entscheidbar.
- C L ist entscheidbar, d.h. die Funktion χ_L ist stets berechenbar.
- D Es gibt eine Grammatik G, die L erzeugt.
- E L ist semi-entscheidbar, d.h. die Funktion χ'_L ist stets berechenbar.

Quiz: Chomsky-Hierarchie usw.

Welche der folgenden Aussagen gelten für die von Turingmaschinen akzeptierten Sprachen L?

- A L ist Typ i-Sprache (mit $i \in \{0, 1, 2, 3\}$). $\rightarrow$ ja
- B Das Wortproblem für alle solchen L ist entscheidbar. $\rightarrow$ nein
- C L ist entscheidbar, d.h. die Funktion χ_L ist stets berechenbar. $\rightarrow$ nein
- D Es gibt eine Grammatik G, die L erzeugt. $\rightarrow$ ja
- E L ist semi-entscheidbar, d.h. die Funktion χ'_L ist stets berechenbar. $\rightarrow$ ja

Quiz: Berechenbarkeit

Welche der folgenden Funktionen $f : \mathbb{N} \to \mathbb{N}$ sind berechenbar?

A $f(x) = 3 \cdot x$

B $f(0) = 0$ und $f(x)$ ist undefiniert für $x > 0$.

C $f(x) = 1$, wenn es unendlich viele Primpaarzwillinge gibt, und $f(x) = 0$ sonst
Primzahlzwillinge sind Paare $(p, p + 2)$ sodass p und $p + 2$ prim sind, z.B. $(3, 5), (11, 13)$ usw. Es ist unbekannt, ob es unendlich viele Primzahlzwillinge gibt.

D Alle Funktionen $f : \mathbb{N} \to \mathbb{N}$ sind berechenbar.

Quiz: Berechenbarkeit

Welche der folgenden Funktionen $f : \mathbb{N} \to \mathbb{N}$ sind berechenbar?

A $f(x) = 3 \cdot x \to$ ja

B $f(0) = 0$ und $f(x)$ ist undefiniert für $x > 0$. $\to$ ja

C $f(x) = 1$, wenn es unendlich viele Primpaarzwillinge gibt, und $f(x) = 0$ sonst
Primzahlzwillinge sind Paare $(p, p+2)$ sodass p und $p+2$ prim sind, z.B. $(3, 5), (11, 13)$ usw. Es ist unbekannt, ob es unendlich viele Primzahlzwillinge gibt.
$\to$ ja

D Alle Funktionen $f : \mathbb{N} \to \mathbb{N}$ sind berechenbar. $\to$ nein

Nicht berechenbare Funktionen

Sei $B := \{f : \mathbb{N} \to \mathbb{N} \mid f \text{ ist berechenbar}\}$ und $F := \{f : \mathbb{N} \to \mathbb{N}\}$.
Dann gilt

- B ist abzählbar: Da alle Mengen im 7-Tupel endliche Mengen sind, ist klar, dass man alle Turingmaschinen nacheinander aufzählen kann (mithilfe der Gödelisierung)
- F ist überabzählbar (Beweis: gleich)
- Also muss es $f \in F$ geben, mit $f \notin B$. Diese Funktion f ist nicht berechenbar.

F ist überabzählbar

Beweis durch Widerspruch:
Wäre die Menge F abzählbar, dann gibt es eine nummerierte Folge

$$f_0, f_1, f_2, \ldots$$

die alle Funktionen erfasst ($\bigcup_{i \in \mathbb{N}} f_i = F$)

Betrachte die folgende Tabelle:

- Spaltenbeschriftung: alle Eingabezahlen j
- Zeilenbeschriftung: alle Funktionen f_i
- Einträge: $f_i(j)$ bzw. $\perp$ falls $f_i(j)$ undefiniert

	0	1	2	3	4	...
f_0	0	0	0	0	1	...
f_1	$\perp$	3	10	1	6	...
f_2	3	4	5	6	1	...
f_3	9	$\perp$	$\perp$	$\perp$	6	...
...	...	...	...	...	...	...

F ist überabzählbar (2)

Betrachte die folgende Tabelle:

- Spaltenbeschriftung: alle Eingabezahlen j
- Zeilenbeschriftung: alle Funktionen f_i
- Einträge: $f_i(j)$ bzw. $\perp$ falls $f_i(j)$ undefiniert

	0	1	2	3	4	...
f_0	0	0	0	0	1	...
f_1	$\perp$	3	10	1	6	...
f_2	3	4	5	6	1	...
f_3	9	$\perp$	$\perp$	$\perp$	6	...
...	...	...	...	...	...	...
f	$\perp$	$\perp$	$\perp$	0	...	...

Konstruiere $f(i) = \begin{cases} \perp, & \text{falls } f_i(i) \text{ definiert} \\ 0, & \text{falls } f_i(i) \text{ undefiniert} \end{cases}$

F ist überabzählbar (2)

Betrachte die folgende Tabelle:

- Spaltenbeschriftung: alle Eingabezahlen j
- Zeilenbeschriftung: alle Funktionen f_i
- Einträge: $f_i(j)$ bzw. $\perp$ falls $f_i(j)$ undefiniert

	0	1	2	3	4	...
f_0	0	0	0	0	1	...
f_1	$\perp$	3	10	1	6	...
f_2	3	4	5	6	1	...
f_3	9	$\perp$	$\perp$	$\perp$	6	...
...	...	...	...	...	...	...
f_k	$\perp$	$\perp$	$\perp$	0	...	...

Konstruiere $f(i) = \begin{cases} \perp, & \text{falls } f_i(i) \text{ definiert} \\ 0, & \text{falls } f_i(i) \text{ undefiniert} \end{cases}$

Aber da $f \in F$, gibt es k mit $f_k = f$.

F ist überabzählbar (2)

Betrachte die folgende Tabelle:

- Spaltenbeschriftung: alle Eingabezahlen j
- Zeilenbeschriftung: alle Funktionen f_i
- Einträge: $f_i(j)$ bzw. $\perp$ falls $f_i(j)$ undefiniert

	0	1	2	3	4	...
f_0	0	0	0	0	1	...
f_1	$\perp$	3	10	1	6	...
f_2	3	4	5	6	1	...
f_3	9	$\perp$	$\perp$	$\perp$	6	...
...	...	...	...	...	...	...
f_k	$\perp$	$\perp$	$\perp$	0	...	...

Konstruiere $f(i) = \begin{cases} \perp, & \text{falls } f_i(i) \text{ definiert} \\ 0, & \text{falls } f_i(i) \text{ undefiniert} \end{cases}$

Aber da $f \in F$, gibt es k mit $f_k = f$.

Aber an der Stelle k: $f_k(k) = \begin{cases} \perp, & \text{falls } f_k(k) \neq \perp \\ 0, & \text{falls } f_k(k) = \perp \end{cases}$ $\rightarrow$ **Widerspruch**

Reduktion

Wiederholung

- Reduktion der Sprache L_1 auf die Sprache L_2 ($L_1 \leq L_2$): Zeige: Es gibt totale, berechenbare Funktion f mit

$$w \in L_1 \text{ g.d.w. } f(w) \in L_2$$

- Wenn L_1 unentscheidbar ist, ist damit auch L_2 unentscheidbar (denn: wäre L_2 entscheidbar, ist mit $L_1 \leq L_2$ sicher auch auch L_1 entscheidbar).

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet } 1 \text{ bei Eingabe } 0\}$ ist unentscheidbar.

Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet 1 bei Eingabe 0}\}$ ist unentscheidbar.

Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?

Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet 1 bei Eingabe 0}\}$ ist unentscheidbar.

Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?

Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$

Schritt 2: f definieren (mit $w \in H_0$ g.d.w. $f(w) \in L$)

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet 1 bei Eingabe 0}\}$ ist unentscheidbar.
Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?
Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$
Schritt 2: f definieren (mit $w \in H_0$ g.d.w. $f(w) \in L$)
Sei f die folgende Funktion, die bei Eingabe w, eine neue Turingmaschinenbeschreibung u erstellt, sodass M_u sich wie folgt verhält:

1. M_u löscht das Band und simuliert M_w auf leerer Eingabe
2. Wenn M_w anhält (und akzeptiert), dann schreibe 1 auf das Band und akzeptiere.

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet } 1 \text{ bei Eingabe } 0\}$ ist unentscheidbar.
Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?
Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$
Schritt 2: f definieren (mit $w \in H_0$ g.d.w. $f(w) \in L$)
Sei f die folgende Funktion, die bei Eingabe w, eine neue Turingmaschinenbeschreibung u erstellt, sodass M_u sich wie folgt verhält:

1. M_u löscht das Band und simuliert M_w auf leerer Eingabe
2. Wenn M_w anhält (und akzeptiert), dann schreibe 1 auf das Band und akzeptiere.

Schritt 3: f ist total und berechenbar:

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet 1 bei Eingabe 0}\}$ ist unentscheidbar.
Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?
Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$
Schritt 2: f definieren (mit $w \in H_0$ g.d.w. $f(w) \in L$)
Sei f die folgende Funktion, die bei Eingabe w, eine neue Turingmaschinenbeschreibung u erstellt, sodass M_u sich wie folgt verhält:

1. M_u löscht das Band und simuliert M_w auf leerer Eingabe
2. Wenn M_w anhält (und akzeptiert), dann schreibe 1 auf das Band und akzeptiere.

Schritt 3: f ist total (klar) und berechenbar: Die TM M_u ist konstruierbar und (De)kodieren von w und u auf TM möglich

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet } 1 \text{ bei Eingabe } 0\}$ ist unentscheidbar.

Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?

Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$

Schritt 2: f definieren (mit $w \in H_0$ g.d.w. $f(w) \in L$)

Sei f die folgende Funktion, die bei Eingabe w, eine neue Turingmaschinenbeschreibung u erstellt, sodass M_u sich wie folgt verhält:

1. M_u löscht das Band und simuliert M_w auf leerer Eingabe
2. Wenn M_w anhält (und akzeptiert), dann schreibe 1 auf das Band und akzeptiere.

Schritt 3: f ist total (klar) und berechenbar: Die TM M_u ist konstruierbar und (De)kodieren von w und u auf TM möglich

Schritt 4: Zeige $w \in H_0$ g.d.w. $f(w) \in L$

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet 1 bei Eingabe 0}\}$ ist unentscheidbar.

Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?

Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$

Schritt 2: f definieren (mit $w \in H_0$ g.d.w. $f(w) \in L$)

Sei f die folgende Funktion, die bei Eingabe w, eine neue Turingmaschinenbeschreibung u erstellt, sodass M_u sich wie folgt verhält:

1. M_u löscht das Band und simuliert M_w auf leerer Eingabe
2. Wenn M_w anhält (und akzeptiert), dann schreibe 1 auf das Band und akzeptiere.

Schritt 3: f ist total (klar) und berechenbar: Die TM M_u ist konstruierbar und (De)kodieren von w und u auf TM möglich

Schritt 4: Zeige $w \in H_0$ g.d.w. $f(w) \in L$

Es gilt: $w \in H_0$ g.d.w. M_w hält bei leerer Eingabe
g.d.w. M_u hält bei jeder Eingabe (auch bei 0) und schreibt 1 auf das Band
g.d.w. M_u berechnet 1 bei Eingabe 0
g.d.w. $u = f(w) \in L$

Beispiel: Reduktion

Zeige $L := \{w \mid \text{TM } M_w \text{ berechnet 1 bei Eingabe 0}\}$ ist unentscheidbar.

Schritt 1: Welche unentscheidbare Sprache reduzieren wir auf L?

Wir reduzieren H_0 auf L ($H_0 \leq L$): $H_0 := \{w \mid \text{TM } M_w \text{ hält bei leerer Eingabe}\}$

Schritt 2: f definieren (mit $w \in H_0$ g.d.w. $f(w) \in L$)

Sei f die folgende Funktion, die bei Eingabe w, eine neue Turingmaschinenbeschreibung u erstellt, sodass M_u sich wie folgt verhält:

1. M_u löscht das Band und simuliert M_w auf leerer Eingabe
2. Wenn M_w anhält (und akzeptiert), dann schreibe 1 auf das Band und akzeptiere.

Schritt 3: f ist total (klar) und berechenbar: Die TM M_u ist konstruierbar und (De)kodieren von w und u auf TM möglich

Schritt 4: Zeige $w \in H_0$ g.d.w. $f(w) \in L$

Es gilt: $w \in H_0$ g.d.w. M_w hält bei leerer Eingabe
g.d.w. M_u hält bei jeder Eingabe (auch bei 0) und schreibt 1 auf das Band
g.d.w. M_u berechnet 1 bei Eingabe 0
g.d.w. $u = f(w) \in L$

Schritt 5: Schluss: Damit gilt $H_0 \leq L$. Da H_0 unentscheidbar, ist auch L unentscheidbar.

Satz von Rice

Satz von Rice

Sei $\mathcal{R}$ die Klasse aller Turingberechenbaren Funktionen. Sei $\mathcal{S}$ eine beliebige Teilmenge, sodass $\emptyset \subset \mathcal{S} \subset \mathcal{R}$. Dann ist die Sprache

$$C(\mathcal{S}) = \{w \mid \text{die von } M_w \text{ berechnete Funktion liegt in } \mathcal{S}\}$$

unentscheidbar.

Anwenden, um Unentscheidbarkeit von L zu zeigen:

- Menge S von Funktionen definieren.
- Nichttrivialität von S zeigen.
- Begründe, dass S richtig gewählt, d.h. $C(S) = L$.
- Satz von Rice zeigt dann das Resultat.

Beispiel 1: Satz von Rice

$$L = \{w \mid \text{wenn } M_w \text{ auf Eingabe 0 hält, dann hält } M_w \text{ auf Eingabe 1}\}$$

- Sei
  $S = \{f \mid f \text{ ist an den Stellen 0 und 1 definiert, oder an der Stelle 0 undefiniert}\}$.
- S ist nicht trival:

Beispiel 1: Satz von Rice

$$L = \{w \mid \text{wenn } M_w \text{ auf Eingabe 0 hält, dann hält } M_w \text{ auf Eingabe 1}\}$$

- Sei
  $S = \{f \mid f \text{ ist an den Stellen 0 und 1 definiert, oder an der Stelle 0 undefiniert}\}$.
- S ist nicht trival:
  - $\emptyset \subset S$, da $id \in S$ mit $id(x) = x$ und
  - $S \subset \mathcal{R}$, da für f, mit $f(0) = 1$ aber $f(i)$ undefiniert für $i \neq 0$, gilt $f \notin S$.

Beispiel 1: Satz von Rice

$$L = \{w \mid \text{wenn } M_w \text{ auf Eingabe 0 hält, dann hält } M_w \text{ auf Eingabe 1}\}$$

- Sei
  $S = \{f \mid f$ ist an den Stellen 0 und 1 definiert, oder an der Stelle 0 undefiniert$\}$.
- S ist nicht trival:
  - $\emptyset \subset S$, da $id \in S$ mit $id(x) = x$ und
  - $S \subset \mathcal{R}$, da für f, mit $f(0) = 1$ aber $f(i)$ undefiniert für $i \neq 0$, gilt $f \notin S$.
- $$\begin{aligned} C(S) &= \{w \mid \text{die von } M_w \text{ berechnete Funktion liegt in } S\} \\ &= \{w \mid \text{wenn } M_w \text{ auf Eingabe 0 hält, dann auch bei Eingabe 1}\} \\ &= L \end{aligned}$$
- Satz von Rice: $C(S)$ ist unentscheidbar.

Beispiel 2: Satz von Rice

$$L = \{w \mid M_w \text{ hält bei Eingabe } 1\}$$

- Sei $S = \{f \mid f(1) \text{ ist definiert}\}$.
- S ist nicht trival:

Beispiel 2: Satz von Rice

$$L = \{w \mid M_w \text{ hält bei Eingabe } 1\}$$

- Sei $S = \{f \mid f(1) \text{ ist definiert}\}$.
- S ist nicht trival:
  - $\emptyset \subset S$, da $id \in S$ mit $id(x) = x$ und
  - $S \subset \mathcal{R}$, da für f, mit $f(x) =$ undefiniert für alle x gilt $f \notin S$.

Beispiel 2: Satz von Rice

$$L = \{w \mid M_w \text{ hält bei Eingabe } 1\}$$

- Sei $S = \{f \mid f(1) \text{ ist definiert}\}$.
- S ist nicht trival:
  - $\emptyset \subset S$, da $id \in S$ mit $id(x) = x$ und
  - $S \subset \mathcal{R}$, da für f, mit $f(x) =$ undefiniert für alle x gilt $f \notin S$.
- $$\begin{aligned} C(S) &= \{w \mid \text{die von } M_w \text{ berechnete Funktion liegt in } S\} \\ &= \{w \mid M_w \text{ hält bei Eingabe } 1\} \\ &= L \end{aligned}$$
- Satz von Rice: $C(S)$ ist unentscheidbar.