der Rechnung von M bei der Eingabe w ist.

M definiert für jedes $r \in \mathbb{N}$ die Fkt. $f_M^r : \mathbb{N}_0^r \to \mathbb{N}_0$ wobei

$$f_M^r(x_1, x_2, \dots, x_r) = \begin{cases} bin^{-1}(y) & \text{falls M bei Eingabe } bin(x_1)\#\dots\#bin(x_r) \text{ mit Inschrift } y \in \{0,1\}^+ \text{ auf Band 1 anhält} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Eine Funktion $f : \Sigma^* \to \Sigma^*$ ($f : \mathbb{N}_0^r \to \mathbb{N}_0$) heißt <u>turingberechenbar</u>, falls eine TM M mit $f = f_M$ ($f = f_M^r$) existiert.

<u>Ziel:</u> Beweis der Gleichheit der Klasse der μ-rekursiven und der Klasse der turingberechenbaren Fkten.

<u>Satz 3.5</u>

Jede μ-rekursive Funktion ist turingberechenbar.

<u>Beweis:</u>

<u>Lemma 3.3</u>

Die Nachfolgerfunktion, die konstanten Funktionen und die Projektionen können durch reguläre Einband-TM berechnet werden.

Beweis:

Übung

□

Lemma 3.4

Seien $f: \mathbb{N}_0^r \to \mathbb{N}_0$ und $g_i: \mathbb{N}_0^m \to \mathbb{N}_0$, $1 \le i \le r$ Funktionen, die durch reguläre k-Band-TM $F, G_1, \dots, G_r$ berechnet werden. Sei h diejenige Fkt., die durch Substitution der g_i in f entsteht. Dann gibt es eine reguläre $(k+2)$-Band TM H, die h berechnet.

Beweis:
Idee:
- endliche Kontrolle von H enthält
  - endlichen Kontrollen von $F, G_1, \dots, G_r$
- H läßt nacheinander $G_1, G_2, \dots, G_r$ laufen und speichert die Ergebnisse
- Danach läßt H die TM F auf den zwischengespeicherten Ergebnisse als Eingabe laufen.

Durchführung:

reguläre $(k+2)$-Band TM G_i', $1 \le i \le r$, wobei

(1) G_i' kopiert den Inhalt von Band 1 auf Band 3 (Band 3 := Band 1)

(2) G_i' läßt die k-Band TM G_i auf den Bändern $3, 4, \ldots, k+2$ laufen.

(3) Der Inhalt von Band 2 wird wie folgt modifiziert:

$$\text{Band } 2 := \begin{cases} \text{Band } 3 & \text{falls } i=1 \\ \text{Band } 2 \# \text{Band } 3 & \text{falls } i>1 \end{cases}$$

(4) Die Bänder $3, 4, \ldots, k+2$ werden gelöscht, d.h., die Inhalte der Bandquadrate >0 werden mit ␣ überschrieben.

- H läßt nacheinander $G_1', G_2', \ldots, G_r'$ laufen. Dabei gilt:

$$\text{Startkonfiguration}(G_i') = \begin{cases} \text{Startkonf. } H & \text{falls } i=1 \\ \text{Endkonf. } G_{i-1}' & \text{falls } i>1 \end{cases}$$

- Danach läßt H nun F auf den Bändern $2, 3, \ldots, k+1$ laufen, wobei

$$\text{Startkonf } F = \text{Endkonf } G_r'.$$

- H löscht Band 1, kopiert Band 2 auf Band 1 und fährt L/S-Köpfe auf Bandquadrat 1 der korrespondierenden Bänder

Korrektheit ✓

□

Lemma 3.5

Seien $g : \mathbb{N}_0^r \to \mathbb{N}_0$ und $f : \mathbb{N}_0^{r+2} \to \mathbb{N}_0$ Fkten, die von regulären k-Band TM G und F berechnet werden. Werde die Fkt. h durch primitive Rekursion aus g und f definiert. Dann gibt es eine reguläre $(k+4)$-Band TM H, die h berechnet.

Beweis:

- endl. Kontrolle von H enthält
  - endl. Kontrollen von G und F.
- Bänder 1 und 4 dienen als Zähler
  - Band 1 enthält Anzahl der noch durchzuführenden Schritte bzgl. F
  - Band 4 " Anzahl der bereits durchgeführten Schritte bzgl. F
  - Band 2 enthält stets
    $\text{bin}(x_1)\#\text{bin}(x_2)\#\ldots\#\text{bin}(x_r)$
  - Band 3 enthält

$$\text{bin}(h(i, x_1, \ldots, x_r))$$

zuletzt berechneter Fkt-Wert.

- Falls H mit Eingabe

$$\text{bin}(n)\#\text{bin}(x_1)\#\ldots\#\text{bin}(x_r)$$

gestartet wird, dann

steht nach i-ten Durchlauf der while-Schleife auf

Band 1	$bin(n-i)$
Band 2	$bin(x_1)\#\ldots\#bin(x_r)$
Band 3	$bin(h(i,x_1,\ldots,x_r))$
Band 4	$bin(i)$

$\rightsquigarrow$

(1) Kopiere den hinter dem ersten # stehenden Inhalt von Band 1 auf Band 2

(2) Lösche Band 1 ab dem ersten #;

(3) Band 3 := Band 2;

(4) H lässt G auf den Bändern $3, 4, \ldots, k+2$ laufen;

(5) Band 4 := 0;

(6) Lösche Bänder $5, 6, \ldots, k+2$;

(7) while Band 1 $\neq$ 0
do
Band 5 := Band 4;
Band 5 := Band 5 # Band 3;
Band 5 := Band 5 # Band 2;
Laß F auf den Bändern $5, 6, \ldots, k+4$ laufen;
Band 3 := Band 5;
Lösche die Bänder $5, 6, \ldots, k+4$;

eigentlich eine for schleife

Band 1 := Band 1 − 1;

Band 4 := Band 4 + 1

od;

(8) Band 1 := Band 3;

(9) Setze alle L/S-Köpfe auf Bandquadrat 1.

Korrektheit ✓

□

Lemma 3.6

Sei $f: \mathbb{N}_0^{r+1} \to \mathbb{N}_0$ eine Fkt., die durch eine reguläre k-Band-TM F berechnet wird. Dann gibt es eine reguläre $(k+3)$-Band-TM M, die μf berechnet.

Beweis:

endl. Kontrolle von M enthält endl. Kontrolle von F

M führt while-Schleife derart durch, dass stets folgende Invariante erfüllt ist.

Invariante:

M, gestartet mit Eingabe $bin(x_1)\#\ldots\# bin(x_r)$

$\rightsquigarrow$

Nach i-ten Durchlauf der while-Schleife steht auf

Band 1	$bin(f(i-1, x_1, \ldots, x_r))$
Band 2	$bin(x_1)\#\ldots\# bin(x_r)$
Band 3	$bin(i)$

$\rightsquigarrow$

(1) Band 2 := Band 1;
(2) Band 1 := 1;
(3) Band 3 := 0;
(4) <u>while</u> Band 1 $\neq$ 0
<u>do</u>
Band 4 := Band 3;
Band 4 := Band 4 # Band 2;
M läßt F auf den Bändern 4, ..., k+3 laufen;
Band 1 := Band 4;
Lösche Bänder 4, 5, ..., k+3;
Band 3 := Band 3 + 1
<u>od</u>;
(5) Band 1 := Band 3 − 1; $\Longleftarrow$ erkläre warum!
(6) Setze alle L/S-Köpfe auf Bandquadrat 1.

Korrektheit ✓

□

Lemmata 3.3 - 3.6 $\Longrightarrow$

Jede μ-rekursive Fkt. kann durch Hintereinanderausführung von endlich vielen Turingprogrammen vom obigen Typ berechnet werden.

Für den Beweis der anderen Richtung ist folgendes Lemma hilfreich

Lemma 3.7

Jede turingberechenbare Funktion kann von einer regulären Einband-TM berechnet werden.

Beweis:

gegeben: beliebige k-Band-TM M

Ziel: Konstruktion einer regulären 1-Band TM M', die M simuliert.

Kopf 1					↓			1. Spur
Band 1	\$	a	a	b	c	b		2. Spur
Kopf 2		↓						3. Spur
Band 2	\$	b	a	a	c			4. Spur

⋮

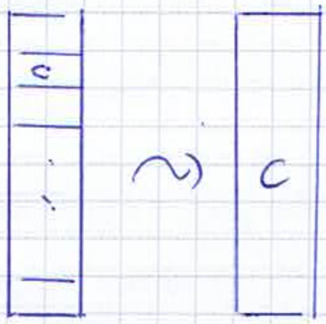
Kopf k						↓		
Band k	\$	c	a	b	b	a		2k. Spur

Band von M'.

Simulation eines Rechenschrittes:

- M' besucht alle Bandquadrate, in denen L/S-Kopfmarkierung steht und speichert zugehöriges Symbol in endlicher Kontrolle.

- Danach wird Rechenschritt auf offensichtliche Art und Weise durchgeführt (erklären)
- Nach Beendigung der Simulation ersetzt M' für jedes Bandquadrat seinen Inhalt durch den Inhalt seiner 2. Spur.



- M' fährt L/S-Kopf auf erstes Bandquadrat.

Bemerkung:

Konfiguration k einer 1-Band-TM als String:

Pos L/S-Kopf.

$$k = ¢\, a_1 a_2 \cdots a_{k-1}\, q\, a_k \cdots a_t$$

Satz 3.6

Sei $f: \mathbb{N}_0^r \to \mathbb{N}_0$ turingberechenbar. Dann ist f auch μ-rekursiv.

Beweis:

f turingberechenbar $\Longrightarrow$

$\exists$ reguläre 1-Band TM $M = (Q, \Sigma, \delta, q_0, F)$, die f berechnet.

Ziel:
Definition von μ-rekursiven Funktionen, die
die Arbeitsweise von M simulieren.

Schwierigkeit

TM operieren auf Zeichenreihen

$\Longleftrightarrow$

μ-rekursive Funktionen operieren auf Zahlen.

$\leadsto$

Wir benötigen Kodierung von Zeichenreihen
durch Zahlen.

D.h. injektive Abbildung

$$\varphi : (Q \cup \Sigma)^* \to \mathbb{N}_0$$

gewünschte Eigenschaft:
Einfache Operationen auf Zeichenreihen sollen
durch einfache primitiv rekursiven Fkte
auf Zahlen simuliert werden können.

Seien

- $K = uqv$, $u,v \in \Sigma^*, q \in Q$

eine Konfiguration von M.

- $\Delta(K)$ die Nachfolgerkonfiguration von K

Idee:

- Konstruktion einer primitiv rekursiven Fkt., die aus der Kodierung einer beliebigen Konfiguration von M die Kodierung der eindeutig bestimmten Nachfolgerkonfiguration konstruiert.
- Konstruktion einer primitiv rekursiven Fkt., die für eine beliebige natürliche Zahl feststellt, ob diese Kodierung einer Endkonfiguration ist.

$\rightsquigarrow$

Lemma 3.8

Es gibt primitiv rekursive Fkten $\tilde{\Delta}: \mathbb{N}_0 \to \mathbb{N}_0$ und $END: \mathbb{N}_0 \to \mathbb{N}_0$, so dass für alle Konfigurationen K von M

a) $\tilde{\Delta}(\varphi(K)) = \varphi(\Delta(K))$ und

b)
$$END(\varphi(K)) = \begin{cases} 0 & \text{falls } K \text{ Endkonfiguration} \\ 1 & \text{sonst.} \end{cases}$$

Vor Spezifikation von φ und Beweis des Lemmas 5.8 beweisen wir den Satz zu Ende.

Beobachtung:

- $\tilde{\Delta}$ bestimmt aus $\varphi(K)$ die Kodierung der eindeutig bestimmten Nachfolgerkonfiguration von K.

- Wir benötigen auch für beliebige $n > 0$ die Kodierung derjenigen Konfiguration K', die aus K nach genau n Schritten von M entsteht.

<u>Idee:</u> einmalige Anwendung von primitiver Rekursion auf $\tilde{\Delta}$

Sei

$D: \mathbb{N}_0^2 \to \mathbb{N}_0$, wobei

$D(0,x) = x$ und
$D(n+1,x) = \tilde{\Delta}(D(n,x))$ für $n \geq 0$.

Mittels Induktion kann einfach bewiesen werden, dass $D(n, +(K))$ die gewünschte Kodierung liefert.

<u>Beobachtung:</u>

- Wir benötigen die Anzahl der Rechenschritte, die M, gestartet in der Konfiguration K, durchführt.

$\rightsquigarrow$ (Anwendung des μ-Operators).

Sei

$A: \mathbb{N}_0 \to \mathbb{N}_0$, wobei

$$A(x) = \begin{cases} \min\{i \mid D(i,x) \text{ ist Kodierung einer Endkonfiguration}\} & \text{falls solches } i \text{ ex.} \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

Sei

$$g : \mathbb{N}_0^2 \to \mathbb{N}_0 \quad \text{mit}$$
$$g(i,x) = END(D(i,x))$$

Dann gilt:

$$A = \mu g.$$

$\Rightarrow$ A ist μ-rekursiv.

Also gilt:

$$A(\varphi(k)) = \begin{cases} \#\text{ Schritte die } M \text{ gestartet in Konfiguration } k \text{ durchführt} & \text{falls definiert} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

und

$$D(A(\varphi(k)), \varphi(k)) =$$

$$\begin{cases} \text{Kodierung der Endkonfiguration in die } M \text{, gestartet in } k \text{ gelangt} & \text{falls } A(\varphi(k)) \text{ definiert} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Ziel:

Definition von primitiv rekursiven Funktionen, die

- für gegebenes Element des Def.bereiches die Kodierung der korrespondierenden Start-konfiguration berechnet bzw.

- aus der Kodierung einer Endkonfiguration den korrespondierenden Funktionswert berechnet. (Endkonfiguration ist dann $\underline{\underline{\$ q \, bin(f(x))}}$) !!!

<u>Lemma 3.9</u>

Es gibt primitiv rekursive Funktionen $E: \mathbb{N}_0^r \to \mathbb{N}_0$ und $F: \mathbb{N}_0 \to \mathbb{N}_0$, so dass $\forall\, x, x_i \in \mathbb{N}_0$, $1 \leq i \leq r$ und alle $q \in Q$

a) $E(x_1, x_2, \ldots, x_r) = \psi(\$ q_0 \, bin(x_1) \# \ldots \# bin(x_r))$
und

b) $F(\psi(\$ q \, bin(x))) = x$.

Wegen

$$f(x_1, x_2, \ldots, x_r) = F\Big(D\big(A(E(x_1, x_2, \ldots, x_r)), E(x_1, x_2, \ldots, x_r)\big)\Big)$$

folgt dann die Behauptung des Satzes.

Offen sind noch

- Definition von ψ,
- der Beweis von Lemma 5.8 und
- der Beweis von Lemma 5.9.

<u>Definition von ψ:</u>

Sei $|Q \cup \Sigma| = p$.

Idee:

Interpretation eines Strings über $Q \cup \Sigma$ als $(p+1)$-äre Zahl y nebst Abbildung dieser Zahl auf die korrespondierende Zahl in $\mathbb{N}_0$.

$\rightsquigarrow$ Erläutern, warum dies eine gute Idee ist!

Sei $Q \cup \Sigma = \{a_1, a_2, \ldots, a_p\}$. Definiere

$$\psi : (Q \cup \Sigma)^* \to \mathbb{N}_0 \quad \text{durch}$$

$$\psi(\varepsilon) = 0$$

$$\psi(a_i) = i \qquad a_i \in Q \cup \Sigma$$

$$\psi(v_1 v_2 \ldots v_s) = \sum_{j=1}^{s} \psi(v_j)(p+1)^{s-j} \qquad v_j \in Q \cup \Sigma$$

Simulation von einfachen Operationen auf $(Q \cup \Sigma)^*$ durch primitiv rekursive Funktionen:

Lemma 3.10

Es gibt primitiv rekursive Funktionen L, CONCAT, PREFIX, SUFFIX, FIRST, LAST, SELECT, so dass $\forall\, b, c \in (Q \cup \Sigma)^*$ und $i \in [0..|b|]$ gilt:

a) $L(\psi(b)) = |b|$.

b) $\mathrm{CONCAT}(\psi(b), \psi(c)) = \psi(bc)$.

c) $\mathrm{PREFIX}(\psi(b), i) = \begin{cases} \psi(b_1 b_2 \cdots b_i) & \text{falls } i > 0 \\ \psi(\varepsilon) & \text{falls } i = 0. \end{cases}$

d) $\mathrm{SUFFIX}(\psi(b), i) = \begin{cases} \psi(b_i \dots b_{|b|}) & \text{falls } i > 0 \\ \psi(\varepsilon) & \text{falls } i = 0. \end{cases}$

e) $\mathrm{FIRST}(\psi(b)) = \psi(b_1).$

f) $\mathrm{LAST}(\psi(b)) = \psi(b_{|b|}).$

g) $\mathrm{SELECT}(\psi(b), i) = \begin{cases} \psi(b_i) & \text{falls } i > 0 \\ \psi(\varepsilon) & \text{falls } i = 0 \end{cases}$

Beweis: (durch Angabe der prim. rek. Fkten)

a) $L(x) = \min\{m \mid m \leq x \wedge (p+1)^m > x\}.$

b) $\mathrm{CONCAT}(x,y) = x(p+1)^{L(y)} + y.$

c) $\mathrm{PREFIX}(x,i) = \begin{cases} x \text{ div } (p+1)^{L(x)-i} & \text{falls } i > 0 \\ 0 & \text{falls } i = 0. \end{cases}$

d) $\mathrm{SUFFIX}(x,i) = \begin{cases} x \bmod (p+1)^{L(x)-i+1} & \text{falls } i > 0 \\ 0 & \text{falls } i = 0. \end{cases}$

e) $\mathrm{FIRST}(x) = \mathrm{PREFIX}(x,1).$

f) $\mathrm{LAST}(x) = \mathrm{SUFFIX}(x, L(x)).$

g) $\mathrm{SELECT}(x,i) = \begin{cases} \mathrm{FIRST}(\mathrm{SUFFIX}(x,i)) & \text{falls } i > 0 \\ 0 & \text{falls } i = 0. \end{cases}$

Beweis der primitiven Rekursivität dieser Fkten als Übung.

$\square$

Beweis des Lemmas 3.8:

Sei

$$\psi(Q) = \{ \psi(q) \mid q \in Q \}.$$

Das Prädikat $P: x \in \psi(Q)$ ist nur für endlich viele x wahr.

Satz 5.2 $\Rightarrow$ P ist primitiv rekursiv

Schreibweise:

$$x_{(i)} = \text{SELECT}(x, i)$$

$$[x, y] = \text{CONCAT}(x, y)$$

$$[x, y, z] = [[x, y], z]$$

Sei

$$q(x) = \min \{ i \mid i < L(x) \wedge x_{(i)} \in \psi(Q) \}$$

Interpretation:
(Kodierung einer
x Konfiguration $\Rightarrow$ $q(x)$ Pos L/S-Kopf.

D.h.

$$x = \psi(uqv), \quad u, v \in \Sigma^*, \; q \in Q$$

$\Rightarrow$

$$q(x) = |u| \cancel{+1} +1.$$

Ziel: $x = \psi(u \overbrace{bqa}^{w} v)$

Konstruktion der Kodierungen $\psi(u)$, $\psi(v)$, $\psi(bqa)$.

Seien

$$u(x) = \text{PREFIX}(x, q(x) \dot- 2),$$

$$v(x) = \text{SUFFIX}(x, q(x) + 2) \quad \text{und}$$

$$\overset{\$}{\overset{\shortparallel}{x_1}}\ \underset{u}{x_2} \quad \underbrace{x_{r-2}\ \overset{q}{\overset{\shortparallel}{x_{r-1}}}\ x_r,} \quad {}_{v}\ x_t \qquad q(x) = r-1$$

$$w(x) = [x_{(q(x) \dot- 1)}, x_{(q(x))}, x_{(q(x)+1)}].$$

Die Fkt.en q, u, v, w sind primitiv rekursiv.

Ferner gilt: für

$$x = \psi(ubqav) \quad \text{mit}$$

$u \in \Sigma^*$, $b \in \Sigma$ ~~$\cup \{\varepsilon\}$, wobei $b \neq \varepsilon$ falls $u \neq \varepsilon$~~
$q \in Q$, $a \in \Sigma$ und $v \in \Sigma^*$:

$$u(x) = \psi(u), \quad v(x) = \psi(v) \quad \text{und} \quad w(x) = \psi(bqa)$$

Ziel:

Definition einer prim. rek. Fkt. $\tilde{\delta}$, die der Übergangsfkt. δ entspricht.

Sei

$$y = \psi(bqa) \quad \text{mit } b \in \Sigma \cup \{\varepsilon\}, q \in Q, a \in \Sigma.$$

Definiere

Bem.:
Das \$ steht das linke Bandende mark. ok

$$\tilde{\delta}(y) := \begin{cases} \psi(bcq') & \text{falls } \delta(q,a) = (q', c, +1) \\ \psi(bq'c) & \text{falls } \delta(q,a) = (q', c, 0) \\ \psi(q'bc) & \text{falls } \delta(q,a) = (q', c, -1) \end{cases}$$

Für alle andere Fälle definiere $\tilde{\delta}(y) = 0$.

$\Rightarrow$

Nur für endlich viele y gilt $\tilde{\delta}(y) \neq 0$.

$\Rightarrow$ Satz 3.2 $\hat{\delta}$ ist primitiv rekursiv.

Definition von $\tilde{\Delta}$ und END:

Sei $x \in \mathbb{N}_0$.

$$\tilde{\Delta}(x) := [u(x), \tilde{\delta}(w(x)), v(x)] \text{ und}$$

$$\text{END}(x) := \begin{cases} 0 & \text{falls } x_{(q(x))} \in \{\psi(e) \mid e \in F\} \\ 1 & \text{sonst} \end{cases}$$

Konstruktion $\Rightarrow$ $\tilde{\Delta}$ und END prim. rek.

$\square$

Beweis des Lemma 3.8:

a) Gesucht prim. rek. Fkt. $E: \mathbb{N}_0^r \to \mathbb{N}_0$ mit

$$E(x_1, x_2, \ldots, x_r) = \psi(\$ q_0 \, bin(x_1) \# \ldots \# bin(x_r)).$$

Zwischenziele:

- Konstruktion einer primitiv rekursiven Funktion $B : \mathbb{N}_0^2 \to \mathbb{N}_0$, die, gegeben $i, x \in \mathbb{N}_0$, das i.-letzte Bit in der Binärdarstellung von x berechnet.

Sei

$$B(i,x) := \left(x \bmod 2^i\right) \operatorname{div} 2^{i-1}.$$ ✓

- Betrachte

$$P(n,x) := \sum_{i=1}^{n} \varphi(B(i,x)) \cdot (p+1)^{i-1}$$

Dann gilt:

$$P(L(\varphi(\mathrm{bin}(x))), x) = \varphi(\mathrm{bin}(x)).$$

<u>Frage</u>: Ist P primitiv rekursiv?

<u>Beobachtung</u>:

$$P(n+1, x) = \varphi(B(n+1, x)) \cdot (p+1)^{n+1-1} + P(n,x)$$

$\Rightarrow$

Wir erhalten P aus B mittels primitiver Rekursion.

$\Rightarrow$

P ist primitiv rekursiv.

Wir erhalten E mit Hilfe der Funktion P durch endlich viele Anwendungen von CONCAT.
Es gilt:

$$E(x_1, x_2, \ldots, x_r) =$$

$$[\varphi(\$), \varphi(q_0), P(L(\varphi(\mathrm{bin}(x_1))), x_1), \varphi(\#), \ldots, \varphi(\#), P(L(\varphi(\mathrm{bin}(x_r))), x_r)].$$

P, L und CONCAT prim. rek.

$\Longrightarrow$ E prim. rek.

b) Gesucht prim. rek. Fkt. $F : \mathbb{N}_0 \to \mathbb{N}_0$ mit

$$F(\varphi(\$ q\, \mathrm{bin}(x))) = x.$$

Sei

$$G : \mathbb{N}_0 \to \mathbb{N}_0$$

definiert durch

$$G(y) = \mathrm{Suffix}(y, q(y) + 1)$$

Dann gilt für $y = \varphi(\$ q\, \mathrm{bin}(x))$

$$G(y) = \varphi(\mathrm{bin}(x)).$$

<u>Ziel</u>:

Konstruktion einer prim. rek. Fkt.

$H : \mathbb{N}_0 \to \mathbb{N}_0$ mit $H(\varphi(\mathrm{bin}(x))) = x.$

Dann gilt: $F(\varphi(\$q\,bin(x))) = H(G(\varphi(\$q\,bin(x))))$

O.B.d.A. seien $a_1 = 0$ und $a_2 = 1$.

$\Rightarrow$

$\varphi(0) = 1$ und $\varphi(1) = 2$.

Die gesuchte Funktion H ist dann definiert durch:

Länge der kodierten Binärzahl.

$$H(x) = \sum_{i=1}^{L(\cancel{bin}(x))} ((x \bmod (p+1)^i \operatorname{div} (p+1)^{i-1}) - 1) \cdot 2^{i-1}$$

Definition $\Rightarrow$ H ist primitiv rekursiv.

Korrektheit als Übung

$\square$

Churchsche These (1936)

Die Klasse der im intuitiven Sinn berechenbaren Funktionen ist gleich der Klasse der μ-rekursiven Funktionen.

etwas erzählen

Churchsche These: (1936)

Die Klasse der im intuitiven Sinn berechenbaren Funktionen ist gleich der Klasse der μ-rekursiven Funktionen.

etwas erzählen

3.4 Entscheidbarkeit

Σ endl. Alphabet. Eine Sprache $L \subseteq \Sigma^*$ heißt rekursiv oder auch entscheidbar, falls ihre charakteristische Funktion $c_L : \Sigma^* \to \{0,1\}$ mit

$$c_L(x) = \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{sonst} \end{cases}$$

turingberechenbar ist.

L heißt rekursiv aufzählbar, falls die partielle Fkt. c_L^* mit

$$c_L^*(x) = \begin{cases} 1 & \text{falls } x \in L \\ \text{undefiniert} & \text{sonst} \end{cases}$$

turingberechenbar ist.

Berechnet eine TM M für eine Sprache L die Funktion c_L oder c_L^*, dann schreiben wir $L(M) = L$. Wir sagen, M akzeptiert x genau dann, wenn $x \in L(M)$.

Ziel:

Lösung Halteproblem für Turingmaschinen

Formale Definition:

- $\Gamma = \{0, 1, \#\}$

Ziel:

beliebige TM $M = (Q, \Sigma, \delta, q_0, F)$ als String über Γ ausdrücken.

- Seien Elemente von $Q \cup \Sigma$ derart mit 0 beginnend durchnummeriert, dass
  - die ersten $|Q| = p$ Zahlen die Zustände bezeichnen (d.h. $Q = \{q_0, q_1, \dots, q_{p-1}\}$), wobei q_0 der Startzustand und $F = \{q_{p-|F|}, \dots, q_{p-1}\}$ die Endzustände sind und
  - $\Sigma = \{a_p, a_{p+1}, \dots, a_r\}$

$\delta(q_i, a_j) = (q_{i'}, a_{j'}, c)$ wird durch den String

$$\#\#\, bin(i) \# bin(j) \# bin(i') \# bin(j') \# bin(m),$$

wobei

$$m = \begin{cases} 0 & \text{falls } c = -1 \\ 1 & \text{falls } c = 0 \\ 2 & \text{falls } c = +1 \end{cases}$$

kodiert.

$\rightsquigarrow$

M kann wie folgt kodiert werden:

$$\#\#\#\#\, bin(0)\#\ldots\# bin(p-1)\#\#\#\, bin(p)\#\ldots$$

$$\ldots\# bin(r)\#\#\#\, bin(p-|F|)\#\ldots\# bin(p-1)\#\,\Delta\#\#\#\#$$

wobei Δ die Strings bzgl. δ in beliebiger Reihenfolge enthält.

Bezeichne $\langle M\rangle$ die oben definierte Kodierung von M.

Beliebige Eingabe $w = a_{i_1} a_{i_2} \ldots a_{i_n}$ von M kann durch

$$bin(i_1)\# bin(i_2)\#\ldots\# bin(i_n)\#\#$$

kodiert werden.

Bezeichne $\langle w\rangle$ diese Kodierung.

Das <u>Halteproblem für TM</u> ist definiert durch die Sprache:

$$H = \{x \in \{0,1,\#\}^* \mid x = \langle M\rangle\langle w\rangle \text{ für eine TM } M\text{, die auf Eingabe } w \text{ hält}\}.$$

<u>Ziel</u>:

Beweis, dass H nicht entscheidbar.

Idee:

(1) Definition einer eingeschränkten Version H_e von H und Beweis deren Unentscheidbarkeit.

(2) Beweis, dass Entscheidbarkeit von H die Entscheidbarkeit von H_e impliziert.

Kodierung der Strings über $\{0,1,\#\}$ durch Strings über $\{0,1\}$:

$$\varphi : \{0,1,\#\} \to \{0,1\}^2 \text{, mit}$$

$$\varphi(0) = 00 \quad , \quad \varphi(1) = 01 \quad , \quad \varphi(\#) = 11$$

Erweiterung von φ auf $\{0,1,\#\}^+$ durch

$$\varphi(a_1 a_2 \dots a_n) = \varphi(a_1)\varphi(a_2)\dots\varphi(a_n).$$

eingeschränkte Halteproblem H_e für TM:

$$H_e = \{ x \in \{0,1\}^+ \mid x = \varphi(\langle M \rangle) \text{ für eine TM } M \text{ und } M \text{ hält auf } x \}$$

Satz 3.7

H_e ist nicht entscheidbar.

Beweis:

Annahme: H_e ist entscheidbar.

$\Rightarrow$

$\exists$ TM M, die c_{H_e} berechnet.

Konstruiere TM M', die auf Eingaben in $\{0,1\}^*$ wie folgt operiert:

- endl. Kontrolle von M' enthält endl. Kontrolle von M.
- Falls M bei Eingabe x den Wert 0 ausgeben würde, dann hält M' an.
- Falls M bei Eingabe x den Wert 1 ausgeben würde, dann gerät M' in eine Endlosschleife, hält also nicht an.

Betrachte das Verhalten von M' bei der Eingabe $\varphi(\langle M'\rangle)$. Es gilt:

M' hält auf $\varphi(\langle M'\rangle)$ $\Leftrightarrow$ M auf $\varphi(\langle M'\rangle)$ gibt 0 aus

$\Leftrightarrow$ $\varphi(\langle M'\rangle) \notin H_e$

$\Leftrightarrow$ M' hält nicht auf $\varphi(\langle M'\rangle)$.

Also ist Annahme falsch. D.h., H_e unentscheidbar.

Korollar 3.2

H ist nicht entscheidbar.

Beweis:

Betrachte $x \in \{0,1\}^*$ mit $x = \varphi(\langle M \rangle)$ für eine TM M. Es gilt

$$x \in H_e \iff \langle M \rangle \langle x \rangle \in H.$$

Da $\langle M \rangle \langle x \rangle$ einfach aus x konstruierbar ist, würde aus der Entscheidbarkeit von H auch die Entscheidbarkeit von H_e folgen. $\blacksquare$