Rekursive Funktionen – Basisfunktionen

- die nullstellige Funktion Z, die den konstanten Wert 0 liefert,
- die Funktion $S : \mathbf{N} \to \mathbf{N}$, bei der jeder natürlichen Zahl ihr Nachfolger zugeordnet wird,
- die Funktion $P : \mathbf{N} \to \mathbf{N}$, bei der jede natürliche Zahl $n \geq 1$ auf ihren Vorgänger und die 0 auf sich selbst abgebildet wird,
- die Funktionen $P_i^n : \mathbf{N}^n \to \mathbf{N}$, die durch

$$P_i^n(x_1, x_2, \ldots, x_n) = x_i$$

definiert sind.

Rekursive Funktionen – Operationen

- *Kompositionsschema*:
  Für eine m-stellige Funktion g und m n-stellige Funktionen $f_1, f_2, \ldots, f_m$ definieren wir die n-stellige Funktion f vermöge

$$f(x_1, x_2, \ldots, x_n) = g(f_1(x_1, \ldots, x_n), f_2(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)).$$

- *Rekursionsschema*:
  Für fixierte natürliche Zahlen $x_1, x_2 \ldots, x_n$, eine n-stellige Funktion g und eine $(n+2)$-stellige Funktion h definieren wir die $(n+1)$-stellige Funktion f vermöge

$$\begin{aligned} f(x_1, x_2, \ldots, x_n, 0) &= g(x_1, x_2, \ldots, x_n), \\ f(x_1, x_2, \ldots, x_n, y+1) &= h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y)). \end{aligned}$$

Rekursive Funktionen – Definition und Charakterisierung

Definition:
Eine Funktion $f : \mathbf{N}^n \to \mathbf{N}$ heißt primitiv-rekursiv, wenn sie mittels endlich oft wiederholter Anwendung von Kompositions- und Rekursionsschema aus den Basisfunktionen erzeugt werden kann.

Satz:
Eine Funktion f ist genau dann primitiv-rekursiv, wenn sie **LOOP**-berechenbar ist.

Primitiv-rekursive Funktionen – Beispiele I

a) f mit $f(n) = S(S(n))$
f' mit $f'(n) = S(f(n)) = S(S(S(n)))$
f bzw. f' ordnen jeder natürlichen Zahl ihren zweiten bzw. dritten Nachfolger zu

b) $P(S(x))$ ist primitiv-rekursiv
$P(S(x)) = x$
$id : \mathbf{N} \to \mathbf{N}$ mit $id(x) = x$ ist primitiv-rekursiv,

Primitiv-rekursive Funktionen – Beispiele II

c) Addition natürlicher Zahlen
$add(x, 0) = id(x)$,
$add(x, y + 1) = S(P_3^3(x, y, add(x, y)))$

Multiplikation natürlicher Zahlen
$mult(x, 0) = Z(x)$,
$mult(x, y + 1)$
$= add(P_1^3(x, y, mult(x, y)), P_3^3(x, y, mult(x, y)))$

d) $sum(0) = Z(0) = 0$, $sum(y + 1) = S(add(y, sum(y)))$
definieren

$$sum(y) = \sum_{i=0}^{y} i = \frac{y(y+1)}{2}$$

μ-Operator

Für eine $(n+1)$-stellige Funktion h definieren wir die n-stellige Funktion f wie folgt:
$f(x_1, x_2, \ldots, x_n) = z$ gilt genau dann, wenn die folgenden Bedingungen erfüllt sind:
- $h(x_1, x_2 \ldots, x_n, y)$ ist für alle $y \leq z$ definiert,
- $h(x_1, x_2, \ldots, x_n, y) \neq 0$ für $y < z$,
- $h(x_1, x_2, \ldots, x_n, z) = 0$.

Bezeichnung: $f(x_1, \ldots, x_n) = (\mu y)[h(x_1, \ldots, x_n, y) = 0]$, $f = (\mu y)[h]$

Beispiele:

a) $(\mu y)[add(x,y)] = \begin{cases} 0 & \text{für } x = 0 \\ nicht\ definiert & \text{sonst} \end{cases}$

b) Für $h(x,y) = |9x^2 - 10xy + y^2|$ gilt $(\mu y)[h(x,y)] = id$

Partiell-rekursive Funktionen

Definition:
Eine Funktion $f : \mathbf{N}^n \to \mathbf{N}$ heißt partiell-rekursiv, wenn sie mittels endlich oft wiederholter Anwendung von Kompositionsschema, Rekursionsschema und μ-Operator aus den Basisfunktionen erzeugt werden kann.

Satz:
Eine Funktion ist genau dann partiell-rekursiv, wenn sie **LOOP**/**WHILE**-berechenbar ist.

Folgerung:
Es gibt eine totale Funktion, die nicht partiell-rekursiv ist.

Registermaschine – Definition I

i) Eine Registermaschine besteht aus den Registern $B,\ C_0,\ , C_1, C_2, \ldots, C_n, \ldots$ und einem Programm.
B heißt Befehlszähler, C_0 heißt Arbeitsregister oder Akkumulator, und jedes der Register C_n, $n \geq 1$, heißt Speicherregister.

Jedes Register enthält als Wert eine Zahl aus $\mathbf{N}_0$.

ii) Unter einer Konfiguration der Registermaschine verstehen wir das unendliche Tupel $(b, c_0, c_1, \ldots, c_n, \ldots)$, wobei
das Register B die Zahl b enthält,
für $n \geq 0$ das Register C_n die Zahl c_n enthält.

iii) Das Programm ist eine endliche Folge von Befehlen.

Registermaschine – Definition II

Liste der zugelassenen Befehle und der von ihnen bewirkte Änderung der Konfiguration $(b, c_o, c_1, \ldots c_n, \ldots)$ in die Konfiguration $(b', c'_0, c'_1, \ldots, c'_n, \ldots)$ (wobei $u' = u$ für die nicht angegebenen Komponenten gilt)

Ein- und Ausgabebefehle:

`LOAD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_0 = c_i$
`ILOAD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_0 = c_{c_i}$
`CLOAD` i ,	$i \in \mathbf{N}_0$	$b' = b + 1$	$c'_0 = i$
`STORE` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_i = c_0$
`ISTORE` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_{c_i} = c_0$

Sprungbefehle:

`GOTO` i , $\quad i \in \mathbf{N} \quad b' = i$

`IF` $c_0 = 0$ `GOTO` i , $\quad i \in \mathbf{N} \quad b' = \begin{cases} i & \text{falls } c_0 = 0 \\ b+1 & \text{sonst} \end{cases}$

Registermaschine – Definition III

Arithmetische Befehle:

`ADD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 + c_i$
`CADD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 + i$
`SUB` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \begin{cases} c_0 - c_i & \text{für } c_0 \geq c_i \\ 0 & \text{sonst} \end{cases}$
`CSUB` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \begin{cases} c_0 - i & \text{für } c_0 \geq i \\ 0 & \text{sonst} \end{cases}$
`MULT` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 * c_i$
`CMULT` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 * i$
`DIV` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \lfloor c_0 / c_i \rfloor$
`CDIV` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \lfloor c_0 / i \rfloor$

Stopbefehl:

`END`

Registermaschine – induzierte Funktion

Definition:
Sei M eine Registermaschine. Die von M induzierte Funktion $f_M : \mathbf{N}^n \longrightarrow \mathbf{N}$ ist wie folgt definiert:
$f(x_1, x_2, \ldots x_n) = y$ gilt genau dann, wenn M ausgehend von der Konfiguration $(1, 0, x_1, x_2, \ldots, x_n, 0, 0, \ldots)$ die Konfiguration $(b, c_0, y, c_2, c_3, \ldots)$ für gewisse $b, c_0, c_2, c_3, \ldots$ erreicht und der b-te Befehl des Programm `END` ist.

Registermaschine – Beispiel 1

Registermaschine M_1 mit Programm

```
 1   CLOAD 1              8   LOAD 2
 2   STORE 3              9   CSUB 1
 3   LOAD 2              10   STORE 2
 4   IF c_0 = 0 GOTO 12  11   GOTO 4
 5   LOAD 3              12   LOAD 3
 6   MULT 1              13   STORE 1
 7   STORE 3             14   END
```

$$f_{M_1}(x,y) = 1 \cdot \underbrace{x \cdot x \cdot \ldots \cdot x}_{y \ mal} = x^y$$

Registermaschine – Beispiel 2

Registermaschine M_2 mit Programm

1	LOAD 1	8	LOAD 1
2	IF $c_0 = 0$ GOTO 12	9	CSUB 1
3	LOAD 2	10	STORE 1
4	CADD 1	11	GOTO 1
5	STORE 2	12	LOAD 3
6	ADD 3	13	STORE 1
7	STORE 3	14	END

$f_{M_2}(n) = \sum_{i=1}^{n} i$

Registermaschinen versus LOOP/WHILE-Programme I

Satz:
Zu jedem **LOOP**/**WHILE**-Programm Π gibt es eine Registermaschine M derart, dass $f_M = \Phi_{\Pi,1}$ gilt.

Beweis: (Induktion über die Tiefe)

$x_i := 0$ wird simuliert durch	$x_i := S(x_j)$ wird simuliert durch
1 CLOAD 0	1 LOAD j
2 STORE 1	2 CADD 1
3 END	3 STORE i
	4 END

Registermaschinen versus LOOP/WHILE-Programme II

$\Pi = \Pi_1; \Pi_2$

M_i – Registermaschine mit $f_{M_i} = \Phi_{\Pi_i,1}$, $i \in \{1,2\}$,
Programm P_i von M_i bestehe aus r_i Befehlen
$p_{i,j}$ sei der j-te Befehl von P_i
$p_{i,r_i} =$ `END` und dies einziger Stopp-Befehl in P_i

$q_{j,2} = p_{j,2}$, falls $p_{j,2}$ kein Sprungbefehl
sonst entstehe $q_{j,2}$ aus $p_{j,2}$ durch Erhöhung der Sprungadresse um $r_1 - 1$

1	$p_{i,1}$	r_1	$q_{1,2}$
2	$p_{r_1-1,1}$	r_1+1	$q_{2,2}$
...	...	...	...
r_1-1	$p_{r_1-1,1}$	r_1+r_2-1	$q_{r_2,2}$

berechnet $\Phi_{\Pi,1}$

Registermaschinen versus LOOP/WHILE-Programme III

$\Pi' =$ **WHILE** $x_i \neq 0$ **BEGIN** Π **END**

M – Registermaschine M mit $f_M = \Phi_{\Pi,1}$
$p_1, p_2, \ldots, p_r$ Befehle von M
$p_r =$ `END` einziger Stoppbefehl

q_i entstehe aus p_i durch Erhöhung aller Befehlsnummer um 2 (sowohl Nummern der Befehle als auch Nummern der Sprungadressen)

1	`LOAD` i	...	...	
2	`IF` $c_0 = 0$ `GOTO` $r+3$	$r+1$	q_{r-1}	berechnet $\Phi_{\Pi',1}$
3	q_1	$r+2$	`GOTO` 1	
4	q_2	$r+3$	`END`	

Registermaschinen versus k-Band-TURING-Maschinen

$dec(n)$ – Dezimaldarstellung von n

Satz:
Sei M eine Registermaschine M mit $f_M : \mathbf{N}^n \to \mathbf{N}$. Dann gibt es eine 3-Band-TURING-Maschine M', deren Eingabealphabet außer den Ziffern 0,1,2,. . . 9 noch das Trennsymbol $\#$ und das Fehlersymbol F enthält und deren induzierte Funktion

$$f_{M'}(w) = \begin{cases} dec(f_M(m_1, m_2, \ldots, m_n)) & w = dec(m_1)\#dec(m_2)\ldots\#dec(m_n) \\ F & \text{sonst} \end{cases}$$

gilt (auf einer Eingabe, die einem Zahlentupel entspricht, verhält sich M' wie M und gibt bei allen anderen Eingaben eine Fehlermeldung).

Zusammenfassung

Satz:
Für eine Funktion f sind die folgenden Aussagen gleichwertig:

- f ist durch ein **LOOP**/**WHILE**-Programm berechenbar.
- f ist partiell-rekursiv.
- f ist durch eine Registermaschine berechenbar.
- f ist bis auf Konvertierung der Zahlendarstellung durch eine TURING-Maschine berechenbar.
- f ist bis auf Konvertierung der Zahlendarstellung durch eine k-Band-TURING-Maschine berechenbar.

Komplexität von Registermaschinen

Beschränkung der (indirekten) arithmetischen Befehle auf Addition und Subtraktion

Definition:
Es sei M eine Registermaschine. Die (uniforme) Komplexität von M auf $(x_1, x_2, \ldots, x_n)$, bezeichnet durch $t_M(x_1, x_2, \ldots, x_n)$ ist als Anzahl der Schritte definiert, die M bis zum Erreichen des `END`-Befehls auf der Eingabe $(x_1, x_2, \ldots, x_n)$ ausführt.

Definition:
Es sei M eine Registermaschine. Wir sagen, dass M $t(n)$-zeitbeschränkt ist, wenn

$$t_M(x_1, x_2, \ldots, x_n) \leq t(n)$$

für jede Eingabe $(x_1, x_2, \ldots, x_n)$ gilt.

Registermaschinen versus TURING-Maschinen

Satz:
i) Jede $t(n)$-zeitbeschränkte Registermaschine kann durch eine $O((t(n))^3)$-zeitbeschränkten 3-Band-TURING-Maschine simuliert werden.
ii) Jede $t(n)$-zeitbeschränkte Registermaschine kann durch eine $O((t(n))^6)$-zeitbeschränkten TURING-Maschine simuliert werden.

Lemma:
Es sei $k \geq 2$. Jede $t(n)$-zeitbeschränkte k-Band-TURING-Maschine kann durch eine $O(t(n))$-zeitbeschränkte k-Band-TURING-Maschine simuliert werden, die keine Zelle links der Eingabe betritt/verändert.

Satz:
Jede $t(n)$-zeitbeschränkte k-Band-TURING-Maschine mit $k \geq 2$ kann durch eine $O(t(n))$-zeitbeschränkte Registermaschine simuliert werden.