Semantik von RM durch Zustandsübergangsbeschreibung.
Konfiguration einer n-RM: $(s, p) \in \mathbb{N}^n \times n\text{-PROG}$.
Dabei bezeichnet $n\text{-PROG}$ die Menge aller Programme für n-RM.
Zustands- bzw. Konfigurationsübergangsfunktion:
(für n-RM mit $i \in \mathbb{N}$, $1 \le i \le n$):

$(s, succ_i) \to ((s_1, \dots, s_{i-1}, s_i+1, s_{i+1}, \dots, s_n), \varepsilon)$
$(s, pred_i) \to ((s_1, \dots, s_{i-1}, k, s_{i+1}, \dots, s_n), \varepsilon)$
mit $k = \begin{cases} s_i - 1 & \text{falls } s_i > 0 \\ 0 & \text{sonst} \end{cases}$

$(s, (p_1; p_2)) \to (s', (p_1'; p_2))$ falls $(s, p_1) \to (s', p_1')$ gilt.
$(s, (\varepsilon; p_2)) \to (s, p_2)$
$(s, while_i(p)) \to \begin{cases} (s, (p; while_i(p))) & \text{falls } s_i > 0 \\ (s, \varepsilon) & \text{sonst} \end{cases}$

Bemerkung:
Die Konfigurationen der Form (s, ε) sind terminal, d.h. es ex. keine Nachfolgekonfiguration („das Programm terminiert"). Wie gehabt definieren wir Berechnungen (endl. / unendl.) als Folgen von Konfigurationen in der $\to$-Relation.

Beispiel: RM-Programme

(1) $while_i(pred_i)$ entspricht $\underline{while}\ s_i > 0\ \underline{do}\ s_i := s_i - 1\ \underline{od}$, setzt also s_i auf Null.
$while_i(pred_i); \underbrace{succ_i; \dots; succ_i}_{k\text{-mal}}$

(2) Folgende "Funktionen" können in RM programmiert werden:
- übliche Arithmetik
- übliche Boolesche Algebra (Wahrheitswerte dargestellt durch Zahlen).

Eine Funktion $f: \mathbb{N}^n \to \mathbb{N}^m$ (partiell!) heißt RM-berechenbar, wenn es ein n-RM Programm p gibt, so dass gilt: falls $f(\underbrace{s_1, \dots, s_n}_{s}) = (\underbrace{s_1', \dots, s_m'}_{s'})$,
dann gilt

$(s, p) \to^* (s', \varepsilon)$

auch

$g: \mathbb{N}^k \to \mathbb{N}$ mit $1 \le k \le n$

und

$g(x_1, \dots, x_k) = \Pi_i^n(f(x_1, \dots, x_k, x_{k+1}, \dots, x_n))$ (es gilt $i \in \mathbb{N}$, $1 \le i \le n$)

für gegebene $x_{k+1}, \dots, x_n$ heißt auch RM-berechenbar.

Anm. TM-berechenbar $\Leftrightarrow$ RM-berechenbar

2.2. Rekursive Funktionen

Die beiden bisher betrachteten Berechnungsmodelle (TM, RM) sind „hypothetische" Maschinen, beschrieben durch Zustandsübergänge.
Jetzt betrachten wir ein stärker durch Beschreibung charakterisiertes Berechnungsmodell, rekursiv definierte Funktionen. Auch dafür können wir Algorithmen zur Auswertung angeben (vgl. Termersetzung).

2.2.1. Primitiv rekursive Funktionen

Wir betrachten n-stellige Funktionen $f: \mathbb{N}^n \to \mathbb{N}$ (total oder partiell)
Wir nennen folgende Funktionen Grundfunktionen:

$succ$	$\mathbb{N} \to \mathbb{N}$	mit $succ(n) = n+1$
$zero^{(0)}$	$\to \mathbb{N}$	konstante Nullfunktion
$zero^{(1)}$	$\mathbb{N} \to \mathbb{N}$	einstellige Nullfunktion
Π_i^n	$\mathbb{N}^n \to \mathbb{N}$	i-te Projektion, $1 \le i \le n$

Aus einer gegebenen Menge von Funktionen gewinnen wir weitere durch Komposition $g \circ [h_1, \dots, h_n]$ oder durch Anwendung des Schemas der primitiven Rekursion: Gegeben seien Funktionen
$g: \mathbb{N}^k \to \mathbb{N}$, $h: \mathbb{N}^{k+2} \to \mathbb{N}$.
Wir definieren $f: \mathbb{N}^{k+1} \to \mathbb{N}$ wie folgt:
Für alle $x_1, \dots, x_k, n \in \mathbb{N}$:

$f(x_1, \dots, x_k, \boxed{0}) = g(x_1, \dots, x_k)$
$f(x_1, \dots, x_k \boxed{n+1}) = h(x_1, \dots, x_k, n, f(x_1, \dots, x_k, n))$

Schema der primitiven Rekursion

Beispiel: $add(x,0) = x$

$add(x,n+1) = add(x,n) + 1 = h(x,n, add(x,n)) = succ(\Pi^3_3(x,n,add(x,n)))$

Dies entspricht einer induktiven Definition von f.

Fakt: Sind g,h totale Funktionen, so ist durch das Schema f eindeutig bestimmt und total.

Beweis: Induktion!

Bemerkung: Sind g und h partiell, so ist f auch eindeutig festgelegt, aber u.U. selbst partiell.

Die Menge der primitiv rekursiven Funktion (PR) ist induktiv wie folgt definiert

(1) Die Grundfunktionen sind in PR

(2) Funktionen, die durch Komposition von Funktionen aus PR gewonnen werden können, sind in PR (abgeschl. bezügl. Komp. von Fkt.?)

(3) Funktionen, die durch das Schema der prim. Rekursion über Funktionen $g,h \in PR$ gewonnen werden können, sind in PR.

Wir können, wie für die Komposition, auch für das Schema der primitiven Rekursion eine kompakte Notation einführen:

$pr : (\mathbb{N}^k \to \mathbb{N}) \times (\mathbb{N}^{k+2} \to \mathbb{N}) \to (\mathbb{N}^{k+1} \to \mathbb{N})$

mit $f = pr(g,h)$. Alle Definitionen wie oben.

Es gilt:

(1) alle arithm. Funktionen (totale Division, totale Subtraktion) sind in PR.

(2) alle Funktionen in PR sind total.

(3) alle Funktionen in PR sind TM- berechenbar.

(4) alle Funktionen in PR sind RM- berechenbar.

(5) Da alle Funktionen in PR total sind, existieren trivialerweise Funktionen, die TM- berechenbar bzw. RM- berechenbar, aber nicht in PR sind.

Wir zeigen nun, dass es eine totale Funktion gibt, die offensichtlich TM / RM- berechenbar ist, aber nicht in PR. Ackermannfunktion:

$ack : \mathbb{N}^2 \to \mathbb{N}$ zweist. Fkt auf $\mathbb{N}$

$$ack(n,m) = \begin{cases} m+1 & \text{falls } n=0 \\ ack(n-1,1) & \text{falls } n>0, m=0 \\ ack(n-1, ack(n,m-1)) & \text{sonst } (n,m>0) \end{cases}$$

verschachtelte Rek.

Feststellung:

(1) durch diese Gleichung ist ack eindeutig bestimmt und total.

(2) Wir definieren eine Schar von Funktionen $B_n : \mathbb{N} \to \mathbb{N}$ durch $B_n(m) = ack(n,m)$. Wir erhalten:

$B_0(m) = m+1$
$B_1(m) = m+2$
$B_2(m) = 2*m+3$
$B_3(m) = 2^{m+3} - 3$
$\vdots$

Alle Funktionen B_n sind primitiv rekursiv!

B_{n+1} kann durch das Schema der PR aus B_n definiert werden.

Satz: ack $\notin$ PR

Beweis (Skizze):

Durch Induktion über die Anzahl der Anwendungen des Schemas der prim. Rek. können wir zeigen: Zu jeder Funktion $g : \mathbb{N}^n \to \mathbb{N}$ existiert eine Konstante $c \in \mathbb{N}$ so dass genau $g(x_1,\dots,x_n) < ack(c, \sum_{i=1}^{n} x_i)$ für alle $x_1,\dots,x_n \in \mathbb{N}$. [Nachweis nicht hier]

Widerspruchs-Annahme: ack $\in$ PR.

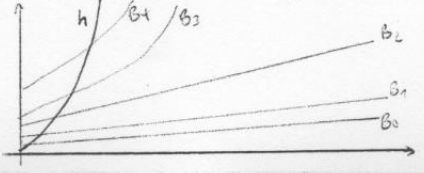
Dann ist $h : \mathbb{N} \to \mathbb{N}$ mit $h(n) = ack(n,m)$ auch in PR.

Es ex. eine Zahl $c \in \mathbb{N}$: $ack(n,m) = h(n) < ack(c,n)$ für alle $n \in \mathbb{N}$.

Mit $n = c$ erhalten wir:

$ack(c,c) = h(c) < ack(c,c)$ ↯

In anderen Worten: h wächst schneller als alle Funktionen in PR.



□

FAZIT:
(1) Es ex. eine totale Funktion, TM- berechenbar / RM- berechenbar, aber nicht in PR.
(2) ack ist offensichtlich durch Rekursion definierbar, also ist der Begriff der PR zu eng.

2.2.2. μ- rekursive Funktionen

Jetzt betrachten wir auch partielle Funktionen, verwenden alle Funktionen aus PR (und alle Konstruktionsprinzipien) und zusätzlich eine weitere Konstruktion, die der μ- Rekursion.

Beispiel: Rekursive Def.

$ulam : \mathbb{N} \to \mathbb{N}$

$$ulam(u) = \begin{cases} 1 & \text{falls } u \in 1 \\ ulam(g(u)) & \text{sonst} \end{cases}$$

$$g(u) = \begin{cases} n \div 2 & \text{falls } n \text{ gerade} \\ 3*n+1 & \text{sonst} \end{cases}$$

Feststellung: g ist primitiv rekursiv.
Frage: Terminiert ulam für alle Argumente $n \in \mathbb{N}$?
Antwort: Unbekannt!
Vermutung: Antwort ist ja!
Falls ulam immer terminiert, gilt $ulam(n) = 1$. D.h. ulam ist dann primitiv rekursiv.

23.5.2003 (Fr)
rek. def. Fkt muss nicht primitiv rekursiv sein?

Das Schema der μ- Rekursion:
Gegeben $f: \mathbb{N}^{k+1} \to \mathbb{N}$ partiell → kann partiell sein
wir definieren $\mu(f): \mathbb{N}^k \to \mathbb{N}$ durch $\mu(f)(x_1,\dots,x_k) = \min\{y \in \mathbb{N} : f(x_1,\dots,x_k,y)=0\}$ (kleinste Nullstelle)
falls $\exists y \in \mathbb{N}: f(x_1,\dots,x_k,y)=0 \wedge \forall z \in \mathbb{N}: 0 \le z \le y \Rightarrow f$ ist def. für $(x_1,\dots,x_k)$
gilt. Gilt die Aussage nicht, dann definieren wir $\mu(f)$ ist für $(x_1,\dots,x_k)$ nicht def.

Verfahren zur Berechnung von $y = \mu(f)(x_1,\dots,x_k)$:
(0) Berechne $f(x_1,\dots,x_k,0) = y_0$
  a) $y_0 = 0 \Rightarrow y = 0$
  b) $y_0 > 0$ gehe zu (1)
  c) Berechnung terminiert nicht ($f(x_1,\dots,x_k,0)$ ist nicht definiert)
  $\mu(f)$ ist nicht def. für $(x_1,\dots,x_k)$
(1) Analog für $f(x_1,\dots,x_k,1)$
⋮

Schritte 0, 1, 2, ... nebeneinander ausführbar

Achtung: Auch wenn f total ist, kann $\mu(f)$ partiell sein.
Die Menge der μ- rekursiven Funktionen MR definieren wir induktiv wie folgt:
(1) Alle primitiv rekursiven Funktionen sind in MR
(2) Funktionen, die durch Komposition oder das Schema der primitiven Rekursion aus Funktionen in MR gebildet werden, sind in MR.
(3) Falls $f \in MR$, dann $\mu(f) \in MR$.

Beispiel: μ- Rekursive Def. der partiellen Subtraktion:

$$a \dot{-} b = \begin{cases} a-b & \text{falls } a \ge b \\ \text{undef.} & \text{sonst} \end{cases}$$

Wir definieren
$a \dot{-} b = \mu(h_0)(a,b)$ mit geeignetem $h_0: \mathbb{N}^3 \to \mathbb{N}$.
Wir wählen $h_0(a,b,y) = sub(b+y,a) + sub(a,b+y)$
wobei
$$sub(a,b) = \begin{cases} a-b & \text{falls } a \ge b \\ 0 & \text{sonst} \end{cases}$$

Fälle	$sub(b+y,a) + sub(a,b+y)$		Fälle
$a \ge b$	$=0$	$=0$	$y = a-b \ge 0$ ✓
$a \ge b$		>0	$y < a-b \Leftrightarrow y+b<a$
$b > a$	>0	keine Nullstelle!	

Also entspricht die definierte Funktion der obigen Angabe!

2.2.3. Allgemeine Bemerkungen zur Rekursion

In der Informatik existieren viele Spielarten von Rekursionen zur Definition von Funktionen, Prozeduren, Methoden, Datentypen, Mengen, Formale Sprachen usw. In der rekursiven Definition einer Funktion verwenden wir Gleichungen der Form $f(x) = E$ wobei f in E auftritt
oder $f(D) = E$ mit eingeschränkten Ausdrücken D.

Beispiel: (Ackermann-Funktion) Strukturelle Rekursion, Induktive Definition

$$\begin{aligned} ack(0,0) &= 1 \\ ack(0, m+1) &= ack(0,m) + 1 \\ ack(n+1, 0) &= ack(n, 1) \\ ack(n+1, m+1) &= ack(n, ack(n+1, m)) \end{aligned}$$

2.3. Äquivalenz der Berechenbarkeitsbegriffe

Wir haben 4 Begriffe von Berechenbarkeit eingeführt:

- TM Turing-Berechenbarkeit
- RM Registermaschinen-Berechenbarkeit
- PR Primitiv rekursive Funktionen
- MR μ-Rekursive Funktionen

PR ist schwächer als TM, RM, MR. Wir zeigen nun die Äquivalenz von TM, RM, MR.

2.3.1. Äquivalenz von μ-Berechenbarkeit und TM-Berechenbarkeit

Der Logiker Kurt Gödel hat eine Idee zur Darstellung von Zeichenfolgen durch Zahlen entwickelt → Gödelisierung (1933).
Eine Gödelisierung ist eine Abb. $f: A^* \to \mathbb{N}$,
wobei A eine endliche Menge von Zeichen sei, mit folgenden Eigenschaften:

(1) f ist injektiv
(2) f ist berechenbar (TM-berechenbar)
(3) es ist berechenbar, ob eine Zahl $n \in \mathbb{N}$ im Bildbereich von f liegt (d.h. $\exists\, w \in A^*: f(w) = n$) dazu $h: \mathbb{N} \to \{0,1\} \subset \mathbb{N}$
(4) Es existiert ein Algorithmus, der zu jeder Zahl im Bildbereich das zugehörige Wort berechnet. "f ist algorithmisch umkehrbar"

Satz: TM $\cong$ MR

Beweis:

(1) $f \in MR \Rightarrow f \in TM$
Beweisidee: Für jede Grundfunktion in MR (bzw. PR) können wir eine TM angeben. (sind somit TM-berechenbar)
Das Schema der Komposition, der PR und der μ-Rekursion können wir durch TM „nachbauen". □

(2) $f \in TM \Rightarrow f \in MR$
Konfigurationen von TM entsprechen Wörtern aus A^* (mit geeignetem A).
Wir verwenden eine Gödelisierung $rep: A^* \to \mathbb{N}$ (einfach zu konstruieren).

Es zeigt sich, dass die Funktion $g: \mathbb{N} \to \mathbb{N}$ mit
$k_0 \to k_1 \iff g(rep(k_0)) = rep(k_1)$ primitiv rekursiv ist (d.h. die Nachfolgerfunktion auf Konfigurationen entspricht einer primitiv rekursiven Funktion auf den Darstellungen der Konfigurationen durch natürliche Zahlen.)
Ferner definieren wir eine primitiv rek. Funktion $h: \mathbb{N}^2 \to \mathbb{N}$ durch

$$h(n,m) = \begin{cases} 1 & \text{falls } g^m(n) \text{ einer terminalen Konfig. entspricht} \\ 0 & \text{sonst} \end{cases}$$

g m-mal angew. auf n

Die Funktion "Iteration" $it : \mathbb{N}^2 \to \mathbb{N}$
sei spezifiziert durch $it(n,0) = n$
$it(n,m+1) = it(g(n),m)$
it beschreibt die Ausführung von m Schritten der TM.

Die Funktion $tm : \mathbb{N} \to \mathbb{N}$
sei definiert durch $tm(n) = it(n, \mu(h)(n))$

Anzahl der Schritte der TM bis zur Terminierung für Eingabekonfiguration dargestellt durch n (falls TM terminiert).

- Ende Beweis (2) $f \in TM \to f \in MR$ - □

2.3.2. Äquivalenz RM / TM - Berechenbarkeit

Jede RM lässt sich in eine TM übersetzen. Dazu brauchen wir eine Darstellung der Konfiguration der RM durch eine TM:

(1) Die Registerinhalte werden auf dem TM-Band dargestellt. (z.B. Strichzahlen)
(2) Aus RM-Programm konstruieren wir den endlichen Automaten, der die TM steuert.

Zu einer gegebenen TM kann ich eine RM konstruieren, die TM simuliert.
Idee: Konfigurationen der TM werden gödelisiert und in den Registern dargestellt. Die Fahrbewegungen und Zustandsübergänge der TM werden durch Programmschritte der RM simuliert.

2.3.3. Church's These

Alonzo Church (1936):
Jede intuitiv berechenbare Funktion ist μ-rekursiv.

Bis heute nicht widerlegt - „Beweis" nicht möglich.