Wir haben den Berechenbarkeitsbegriff bisher auf der primitiven Rekursion und dem $\mu-$Operator aufgebaut. Wir haben gezeigt, daß im Rahmen dieser Theorie Funktionen definiert werden, die nur partiell berechenbar sind in dem Sinne, daß die Theorie keine Methoden an die Hand gibt, diese Einschränkung der Berechenbarkeit zu beheben. Wir haben darüberhinaus gezeigt, daß dieser Mangel grundsätzlicher Natur ist, daß er nämlich auch in jeder Erweiterung des Berechenbarkeitsbegriffs auftreten wird.

Wir sind aber durchaus nicht sicher, daß wir mit der Definition der μ-rekursiven Funktionen alle im intuitiven Sinne berechenbaren Funktionen eingefangen haben. Daß das so ist, darin besteht heute Einigkeit. Wir müssen uns aber im Rahmen der Vorlesung diese Einsicht noch verschaffen.

Es besteht wohl Konsens darin, daß man jede berechenbare Funktion auf unseren Computern programmieren kann, soweit man von ihrer Endlichkeit absieht. Wir werden deshalb das idealisierte Konzept der Maschine präzisieren und anschließend zeigen, daß jede auf diesen Maschinen berechenbare Funktion μ-rekursiv ist, oder besser gesagt, auf eine solche Funktion reduziert werden kann.

Wir betrachten zunächst **Maschinen mit endlichem Speicher**.

Sei

$$K_1 := \{\xi : \{x_1, \dots, x_n\} \longrightarrow \mathbb{N}\}$$

die Menge der Belegungen der Variablen $x_1, \dots, x_n$ mit Werten aus $\mathbb{N}$. K_1 ist für uns die Menge der möglichen Speicherinhalte. Weiter sei $K_2 := [0 : N] \subset \mathbb{N}$; K_2 fassen wir als die Menge der Operationen der Maschine auf.

Wir nennen

$$K = K_1 \times K_2$$

die Konfigurationsmenge der Maschine.

$$K_e := \{(\xi, 0) \mid \xi \in K_1\}$$

heißt die Menge der **Endkonfigurationen**,

$$K_a := \{(\xi, 1) \mid \xi \in K_1\}$$

die Menge der **Anfangskonfigurationen**.

Die Funktion der Maschine wird durch eine Abbildung

$$\Delta : K \longrightarrow K$$

gegeben. Wir verlangen

$$\Delta(k) = k \text{ für } k \in K_e.$$

In einer realen Maschine wird durch jedes $\xi \in K_2$ eine spezielle Operation veranlaßt. Beispiele dafür sind

$$\begin{aligned} x_1 &:= x_2 * x_3, \\ \text{while } x_1 = 0 \text{ do } x_2 &:= x_3 * x_2; x_1 := x_1 - 1 \text{ od}, \end{aligned}$$

oder

$$x_i := x_{\xi(x_j)}.$$

In den beiden ersten Fällen ist klar, daß es sich um μ-rekursive Funktionen handelt. In dem dritten Fall ist das nicht so offensichtlich. Ist n fest, kann man die indizierte Zuweisung auf eine n-fache Fallabfrage zurückführen. Somit ist diese Zuweisung für unsere Maschine, die mit festem n rechnet, μ-rekursiv. Ist n variabel, dann benötigen wir die Zeitvariable t. Nach dem t-ten Rechenschritt steht im Speicher ξ_t.

Wir haben

$$\xi_0 = \text{ Eingabe}$$

bei der Anwendung des indizierten Befehles

$$\xi_{t+1}(x_i) := \begin{cases} \xi_t(x_i) \text{ für } i \neq \xi_t(x_j) \\ \xi_t(\xi_t(x_j)) \text{ sonst.} \end{cases}$$

Also diese Zuweisung ist eine primitiv-rekursive Funktion. Wir sehen also, daß die Operationen einer realen Maschine alle μ-rekursiv sind.

Wir können also davon ausgehen, daß sich Δ durch eine endliche Fallunterscheidung mittels primitiv-rekursiven Prädikaten ($\xi(x_i) \gtreqless 0$) und Funktion aus $\mathbb{P}$ definieren läßt. Also können wir annehmen, daß $\Delta \in \mathbb{P}$ gilt.

Definition: $\mathfrak{M} := (K, K_a, K_e, \Delta)$ heißt $\mathbb{P}$-Maschine $:\iff \Delta \in \mathbb{P}$.

$$\mathfrak{M} \text{ heißt } \mu\mathbb{P}\text{-}Maschine :\iff \Delta \in \mu\mathbb{P}.$$

Eine **Berechnung** von $\mathfrak{M}$ ist eine Folge

$$(\xi_1, \zeta_1). \xrightarrow{\Delta} (\xi_2, \zeta_2) \xrightarrow{\Delta} \ldots \xrightarrow{\Delta} (\xi_m, \zeta_m),$$

Eine Berechnung heißt regulär $:\iff \zeta_1 = 1 \;\&\; \zeta_m = 0$.

Wir definieren weiter

$$T(\xi) := \begin{cases} min\{m \mid (\xi,1) \xrightarrow{\Delta} (\xi_1,\zeta_1) \longrightarrow \dots \\ \qquad\qquad \longrightarrow (\xi_m,\zeta_m) \text{ ist regulär}\} \\ \infty \text{ falls das Minimum nicht existiert.} \end{cases}$$

T heißt **Laufzeitfunktion** von $\mathfrak{M}$.

Die Funktion

$$f : D \longrightarrow \mathbb{N}^j, D \subset \mathbb{N}^k$$

heißt $\mathfrak{M}$-**berechenbar** $:\iff$ Für

$\xi(x_i) = \alpha_i$ für $i = 1, \dots, k$
und $\xi(x_i) = 0$ für $i = k+1, \dots, n$
und $\alpha \in D$ ist $T(\xi) \neq \infty$ und es gilt
$f(\alpha) = (\xi_m(x_1), \dots, \xi_m(x_j))$ für $m = T(\xi)$.

Satz: Jede mit einer $\mu\mathbb{P}$-Maschine berechenbare Funktion ist μ-rekursiv.

Beweis: Wir definieren

$$F(\xi, 0) := (\xi, 1),$$
$$F(\xi, t+1) := \Delta(F(\xi, i)).$$

Es gilt also $\Delta^{(m)} = F(\xi, m)$. F wird durch eine primitive Rekursion auf einer μ-rekursiven Funktion erzeugt und ist also μ-rekursiv.

Die Funktion T ist μ-rekursiv, denn F ist total und es gilt

$$T = \mu F.$$

Die μ-rekursive Funktion

$$F(\xi, T(\xi))$$

liefert den Endzustand der Maschine. $f(\alpha)$ erhalten wir durch die Projektion von K_1 auf die ersten j Komponenten. Also ist f μ-rekursiv.

Wir erhalten auf die gleiche Weise

Satz: Ist f durch eine $\mathbb{P}$-Maschine berechenbar, dann ergibt sich f durch die einmalige Anwendung des μ-Operators auf eine primitiv-rekursive Funktion und eine anschließende Projektion.

Wir zeigen nun, daß sich jede μ-rekursive Funktion durch eine einmalige Anwendung des μ-Operators und eine nachfolgende Projektion berechnen läßt. Das ergibt sich unmittelbar aus dem folgenden

Satz: Jede μ-rekursive Funktion läßt sich mittels einer $\mathbb{P}$-Maschine berechnen.

Beweis: Ist f μ-rekursiv, dann gibt es eine Folge von Definitionen von Funktionen, die aufbauend auf E durch die Anwendung der zugelassenen Operationen f definieren. Wir führen den Beweis induktiv nach der Länge dieser Konstruktion. O.b.d.A. dürfen wir annehmen, daß jede Maschine nach dem Start mit $\zeta = 1$ nicht mehr in einen Anfangszustand zurückkehrt.

Die Aussage des Satzes gilt trivialerweise für die Funktionen aus E. Wir nehmen an, daß die Behauptung richtig ist für alle Funktionen, deren Konstruktion höchstens l Schritte erfordert. Daraus leiten wir ab, daß die Behauptung auch für $l + 1$ gilt.

(1) Wir betrachten zunächst die Operation

$$f := \tilde{f} \times \overline{f}.$$

Für $\tilde{f}$ und $\overline{f}$ gebe es eine $\mathbb{P}$-Maschine $\tilde{\mathfrak{M}}$ bzw. $\overline{\mathfrak{M}}$, die $\tilde{f}$ bzw. $\overline{f}$ berechnet. Wir definieren

$$\mathfrak{M} := \tilde{\mathfrak{M}} \times \overline{\mathfrak{M}}$$

durch die folgenden Ausdrücke

$$\begin{array}{rcl} K_1 & := & \tilde{K}_1 \times \tilde{K}_2 \times \overline{K}_1 \times \overline{K}_2, \\ K_2 & := & b_2^{-1}(\tilde{K}_2 \times \overline{K}_2), \\ \Delta(\xi, \zeta) & := & ((\tilde{\Delta}(\tilde{\xi}, \tilde{\zeta}), \overline{\Delta}(\overline{\xi}, \overline{\zeta})), b_2^{-1}(\tilde{\zeta}_1, \overline{\zeta}_1)). \end{array}$$

Dabei sei $\tilde{\zeta}_1 := \tilde{\Delta}_2(\tilde{\xi}, \tilde{\zeta})$ und $\bar{\zeta}_1 := \bar{\Delta}_2(\bar{\xi}, \bar{\zeta})$. Damit ist $\mathfrak{M}$ eine $\mathbb{P}$-Maschine, da das nach Voraussetzung für $\tilde{\mathfrak{M}}$ und $\overline{\mathfrak{M}}$ gilt.

Wegen

$$b_2^{-1}(\tilde{\zeta}, \overline{\zeta}) = 0 \iff \tilde{\zeta} = 0 \ \& \ \overline{\zeta} = 0$$

hält die Maschine genau dann an, wenn $\tilde{\mathfrak{M}}$ und $\overline{\mathfrak{M}}$ angehalten haben.

Durch eine geeignete Projektion von K_1 erhalten wir $f(\xi) = (\tilde{f}(\tilde{\xi}), \overline{f}(\overline{\xi}))$.

(2) Wir betrachten nun

$$\tilde{f}:\tilde{D}_1 \longrightarrow \tilde{D}_2, \overline{f}:\overline{D}_1 \longrightarrow \overline{D}_2$$

mit $\tilde{D}_2 = \overline{D}_1$ und $f := \overline{f} \circ \tilde{f}$.

Wir definieren zunächst zu vorgegebenen Maschinen $\mathfrak{M}$ eine Maschine $\mathfrak{M}_c$, die primitiv-rekursive Funktionen berechnet. Man erhält $\mathfrak{M}_c$ aus $\mathfrak{M}$, indem man $\mathfrak{M}$ so abändert, daß die Operation von $\mathfrak{M}$ in Abhängigkeit von einem neu eingeführten Parameter r gerade r-mal ausgeführt wird.

Wir setzen

$$K_{c1} := K \times \mathbb{N}^2, K_{c2} := \{0,1,2\}$$

und

$$\Delta_c(\xi,\zeta,\varrho,r,\zeta') := \begin{cases} (\xi,\zeta,\varrho,r,\zeta') \text{ für } \zeta' = 0 \\ (\xi,\zeta,r,r,2) \text{ für } \zeta' = 1 \\ (\Delta(\xi,\zeta),\varrho-1,r,2) \text{ für } \varrho \neq 0 \\ (\xi,\zeta,0,r,0) \text{ für } \varrho = 0. \end{cases}$$

$\mathfrak{M}_c$ ist eine $\mathbb{P}$-Maschine und berechnet Funktionen in $\mathbb{P}$, da das Anhalten durch den beschränkten μ-Operator bewirkt wird.

Seien nun $\tilde{\mathfrak{M}}$ und $\overline{\mathfrak{M}}$ Maschinen, die $\tilde{f}$ bzw. $\overline{f}$ berechnen. Wir bilden $\tilde{\mathfrak{M}}_c$ und $\overline{\mathfrak{M}}_c$ und kombinieren diese Maschinen so, daß die neue Maschine $\overline{f} \circ \tilde{f}$ berechnet.

Die neue Maschine setzt zunächst $\tilde{\mathfrak{M}}_c$ in einen Anfangszustand, indem sie $\tilde{\zeta} := 1$ ausführt und $\tilde{r}$ und $\overline{r}$ je auf einen vorgegebenen Wert setzt. Die Maschine $\overline{\mathfrak{M}}_c$ setzt die neue Maschine in einen Endzustand, indem sie $\overline{\zeta} := 0$ ausführt und $\overline{r}$ den gleichen Wert zuweist wie $\tilde{r}$. Nun läuft $\tilde{\mathfrak{M}}_c$. Sobald $\tilde{\mathfrak{M}}_c$ in den Endzustand kommt, wird das Resultat $\tilde{f}_r(\zeta)$ von $\tilde{\mathfrak{M}}_c$ an $\overline{\mathfrak{M}}_c$ übergeben und $\overline{\mathfrak{M}}_c$ durch $\zeta := 1$ gestartet. Nach r Rechenschritten erreicht $\overline{\mathfrak{M}}_c$ einen Endzustand. Falls $\overline{\zeta} = \tilde{\zeta} = 0$ ist, wird die Berechnung beendet. Es ist nun $\overline{f}_r(\tilde{f}_r(\xi)) = f(\xi)$.

Ist die Bedingung $\tilde{\zeta} = \overline{\zeta} = 0$ nicht erfüllt, erhöht die neue Maschine in ihren Komponenten $\tilde{\mathfrak{M}}_c$ und $\overline{\mathfrak{M}}_c$ r um 1 und startet den eben beschriebenen Prozeß aufs Neue. Die Maschine hält für alle $\xi \in D$ nach endlicher Zeit an. Man erhält $f(\xi)$ durch eine geeignete Projektion des Speicherinhaltes.

Die eben skizzierte Maschine ist eine $\mathbb{P}$-Maschine, da die Abfragen $\tilde{\zeta} = 0$ und $\overline{\zeta} = 0$ durch die beschränkten Abfragen auf $\varrho = 0$ ersetzt werden.

(3) Die primitive Rekursion behandelt man ebenso, indem man sie nicht durch rekursive Prozeduraufrufe ausführt, sondern iterativ, d. h. der Reihe nach $f(\alpha, 0), f(\alpha, 1), \ldots, f(\alpha, x)$ berechnet. Es kommt im Vergleich zu (2) hinzu, daß man das Resultat von $\overline{\mathfrak{M}}$ an $\tilde{\mathfrak{M}}$ übergeben muß und diesen Zyklus $\xi(x)$-mal betreibt, wenn die primitive Rekursion über die Variable x läuft.

(4) Fall: μ-Operator.

Sei also $\mathfrak{M}$ eine $\mathbb{P}$-Maschine, die

$$f : \mathbb{N}^{k+1} \longrightarrow \mathbb{N}$$

berechnet. Wir konstruieren eine Maschine $\tilde{\mathfrak{M}}$, die $g := \mu f$ berechnet wie folgt.

Die Maschine $\tilde{\mathfrak{M}}$ verhält sich weitgehend wie $\mathfrak{M}$. Zum Start „rettet“sie nur $\xi(x_1, \ldots, x_k)$ in neue Variablen $(y_1, \ldots, y_k)$ und setzt $\xi(x_{k+1}) = 0$. Die Variable ζ von $\mathfrak{M}$ schlagen wir nun den Speichervariablen zu. D. h. wir definieren

$$\begin{aligned} \tilde{K}_1 &:= K_1 \times K_2 \times K_1'; \\ K_1' &:= \{\eta : \{y_1, \ldots, y_{k+1}\} \longrightarrow \mathbb{N}\}. \end{aligned}$$

Wir setzen

$$\tilde{K}_2 := \{0, 1, 2\}.$$

Im Anfangszustand $\tilde{\zeta} = 1$ setzt $\tilde{\mathfrak{M}}$ $\zeta := 1$ und $\eta(y_1, \ldots, y_k) := \eta(x_1, \ldots, x_k)$, $\eta(x_{k+1}) := \xi(x_{k+1}) := 0$. Danach geht $\tilde{\zeta}$ in den Zustand 2 über.

In diesem Zustand verhält sich $\tilde{\mathfrak{M}}$ wie $\mathfrak{M}$, solange wie $\zeta \neq 0$ ist. Ist $\zeta = 0$, $\xi(x_1) \neq 0$, dann setzt $\tilde{\mathfrak{M}}$

$$\begin{aligned} \xi'(x_1, \ldots, x_k) &:= \eta(y_1, \ldots, y_k), \\ \xi(x_{k+1}) &:= \eta(y_{k+1}) + 1 \\ \eta'(y_{k+1}) &:= \eta(y_{k+1}) + 1, \\ \zeta &:= 1. \end{aligned}$$

Diese „Runden“ laufen solange, bis $\xi(x_1) = 0$ auftritt. Nun wird $\zeta := 0$ gesetzt. Gibt man $\eta(y_{k+1})$ aus, dann erhält man $g(\xi(x_1, \ldots, x_k))$.

$\tilde{\mathfrak{M}}$ ist wieder eine $\mathbb{P}$-Maschine. Also ist g durch eine $\mathbb{P}$-Maschine berechenbar.

Aus (1) bis (4) ergibt sich nun unser Satz.

Es bleibt die Behandlung von Berechnungen auf $\mathbb{N}^*$.