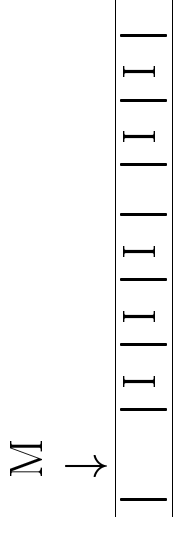


## Maschinenmodelle

## Die Turingmaschine:



**Maschinen:** *informatische* Modelle für Berechenbarkeit.

**Basisfunktionen + Operatoren:** *mathematische* Modelle der Berechenbarkeit.

- *M*: Lese/Schreibkopf.  
Bewegung: ein Feld nach links/rechts.  
Schreiben: *I* oder Blank.

## Funktionsweise der Turingmaschine:

- *Band*: unendlich lang und in unendlich viele Felder unterteilt.
- *Bandalphabet*:  $X = \{I, B\}$  wobei  $B$  für Blank steht und das leere Feld symbolisiert.
- *Operationen*:
  - $d_I$ : beschreibe Feld mit  $I$ .
  - $d_B$ : lösche Symbol, i.e. beschreibe Feld mit  $B$ .
  - $L$ : gehe ein Feld nach links.
  - $R$ : gehe ein Feld nach rechts.
  - $t$ : testet ob Feld leer ist (ob  $B$  im Feld steht); Ausgabe ist **t** oder **f**.
- *Programm*: endliche Vorschrift zur Kombination der obigen Operationen.

## Beispiel: Addition

```
1: if  $t$  then 2 else 3   while  $t$  do  $R$ ;  
2:  $R$  then 1             while  $\neg t$  do  $R$ ;  
3: if  $t$  then 5 else 4    $d_I$ ;  
4:  $R$  then 3             while  $\neg t$  do  $R$ ;  
5:  $d_I$  then 6            $L$ ;  
6: if  $t$  then 8 else 7    $d_B$ .  
7:  $R$  then 6  
8:  $L$  then 9  
9:  $d_B$  then 0
```

Sprung zum Label 0: Termination.

**Rechenvorgang:** für Beispiel.

*Input:*

$$B^{-\infty} III B I I B^{\infty}.$$

Mit Stellung des Kopfes:

$$B^{\infty} \overset{\downarrow}{B} III B I I B^{\infty}.$$

$$B^{\infty} \overset{\downarrow}{B} III B I I B^{\infty} \Rightarrow B^{\infty} \overset{\downarrow}{I} I I B I I B^{\infty} \Rightarrow$$

$$B^{\infty} I \overset{\downarrow}{I} I B I I B^{\infty} \Rightarrow B^{\infty} I I \overset{\downarrow}{I} B I I B^{\infty} \Rightarrow$$

$$B^{\infty} I I I \overset{\downarrow}{B} I I B^{\infty} \Rightarrow B^{\infty} I I I \overset{\downarrow}{I} I I B^{\infty} \Rightarrow$$

$$B^{\infty} I I I I \overset{\downarrow}{I} I B^{\infty} \Rightarrow B^{\infty} I I I I I \overset{\downarrow}{I} B^{\infty} \Rightarrow$$

$$B^{\infty} I I I I I I \overset{\downarrow}{B} B^{\infty} \Rightarrow B^{\infty} I I I I I I \overset{\downarrow}{I} B^{\infty} \Rightarrow$$

$$B^{\infty} I I I I I \overset{\downarrow}{B} B^{\infty}.$$

**die Registermaschine:**

Verallgemeinerter Abacus.

- unendlich viele Register,
- beliebig viele Speichereinheiten/Register,
- Einheit speichern,
- Einheit löschen

1	• • •
2	• •
3	
...	

# Operationen:

1.  $A_i$ : eine Einheit (zusätzlich) in das  $i$ -te Register speichern.
2.  $S_i$ : eine Einheit aus dem  $i$ -ten Register entfernen.
3.  $t_i$ : ein Prädikat welches genau dann wahr ist wenn das  $i$ -te Register leer ist.

# Additionsprogramm für Registermaschine:

Speichere Inhalt von 2. (zusätzlich) in 1. Register

- 1 : if  $t_2$  then 0 else 2
- 2 :  $A_1$  then 3
- 3 :  $S_2$  then 1.

Resultat auf der Eingabe 3 und 2:

1	● ● ● ● ●
2	
3	
...	

## allgemeine Maschinendefinition:

Eine *Maschine* ist ein 5-Tupel  $(S, O, T, \delta, \beta)$  mit folgenden Eigenschaften:

- (M1)  $S$  ist eine Menge (von Speicherzuständen).
- (M2)  $O$  ist eine Menge von Funktionen  $S \rightarrow S$  (Funktionsbefehle).
- (M3)  $T$  ist eine Menge von Prädikaten auf  $S$  (Testbefehle).
- (M4)  $\delta: I \rightarrow S$  ( $I$ : Inputmenge,  $\delta$ : Inputfunktion).
- (M5)  $\beta: S \rightarrow J$  ( $J$ : Outputmenge,  $\beta$ : Outputfunktion).

## • Die Turingmaschine:

$$\begin{aligned} X &= \{I, B\} \text{ (Alphabet),} \\ S &= \{z \mid z \in X^{\mathbf{Z}}, z(i) = Bf.ü.\}, \\ O &= \{d_I, d_B, L, R\}, \\ T &= \{t\}, \end{aligned}$$

wobei gilt:

$$\begin{aligned} d_I(z) &= z', \text{ wobei} \\ z'(0) &= I, z'(i) = z(i) \text{ für } i \neq 0, \\ d_B(z) &= z', \text{ wobei} \\ z'(0) &= B, z'(i) = z(i) \text{ für } i \neq 0, \\ L(z) &= z', \text{ wobei} \\ z'(i) &= z(i-1) \text{ für } i \in \mathbf{Z}, \\ R(z) &= z', \text{ wobei} \\ z'(i) &= z(i+1) \text{ für } i \in \mathbf{Z}, \\ t(z) &= \mathbf{t} \text{ genau dann wenn } z(0) = B. \end{aligned}$$

Eingabe/Ausgabe:

Für uns wichtig:  $\mathbb{N}^m$  für  $m \geq 1$ .

- $I = \mathbb{N}^m$ :  $\delta_m: \mathbb{N}^m \rightarrow S$ ,
- $J = \mathbb{N}^n$ :  $\beta_n: S \rightarrow \mathbb{N}^n$ .

$$\delta_m(x_1, \dots, x_m) =$$

$$B^{-\infty} \overset{\downarrow}{B} I^{x_1} B I^{x_2} \dots B I^{x_m} B^\infty,$$

$$\beta_n(B^{-\infty} u \overset{\downarrow}{v} I^{y_1} B I^{y_2} \dots B I^{y_n} B w B^\infty) = (y_1, \dots, y_n)$$

für  $u, w \in X^*, v \in X$ .

Dabei steht z.B.

$$B^{-\infty} \overset{\downarrow}{B} I^{x_1} B I^{x_2} \dots B I^{x_n} B^\infty$$

für folgendes  $z \in S$ :

$$z(i) = B \quad \text{für } i \leq 0,$$

$$z(i) = I \quad \text{für } 0 < i \leq x_1,$$

$$z(x_1 + 1) = B$$

$$z(i) = I \quad \text{für}$$

$$x_1 + 1 < i \leq x_1 + x_2 + 1,$$

$$z(x_1 + x_2 + 2) = B,$$

.....

$$z(i) = I \quad \text{für}$$

$$(\sum_{j=1}^{n-1} x_j) + n - 1 < i \leq \sum_{j=1}^n x_j + n - 1$$

$$z(i) = B \quad \text{für } i \geq (\sum_{j=1}^n x_j) + n.$$

Die Turingmaschine mit  $I = \mathbb{N}^m$  und  $J = \mathbb{N}^n$  wird als  $\text{TM}_n^m$  bezeichnet.

Turingmaschinen auf Strings:

- $X = \Sigma \cup \{B\}$  für
- $\Sigma = \{x_1, \dots, x_r\}$ .

Input–Output Mengen:

$$I = J = \Sigma^*.$$

$$\begin{aligned} O &= \{L, R\} \cup \{d_x \mid x \in X\}, \\ T &= \{t_x \mid x \in X\}, \end{aligned}$$

$$\begin{aligned} \delta(y) &= B^{-\infty} \overset{\downarrow}{B} y B^{\infty} \quad \text{für } y \in \Sigma^*, \\ \beta(u \overset{\downarrow}{v} y B w) &= y \quad \text{für } y \in \Sigma^*. \end{aligned}$$

Dabei gilt

$$\begin{aligned} d_x(z) &= z' \quad \text{mit } z'(0) = x, z'(i) = z(i) \quad \text{für } i \neq 0 \\ t_x(z) &= \mathbf{t} \iff z(0) = x. \end{aligned}$$

# • **Die Registermaschine (RM):** Sei $\mathbb{N}_+ = \mathbb{N} - \{0\}$ .

$$S = \{r \mid r \in \mathbb{N}^{\mathbb{N}_+}, \sum_{i \geq 1} r(i) < \infty\}.$$

$$\begin{aligned} O &= \{A_i, S_i \mid i \in \mathbb{N}_+\}, \\ T &= \{t_i \mid i \in \mathbb{N}_+\}, \end{aligned}$$

wobei gilt

$$\begin{aligned} A_i(r) &= r' \quad \text{mit} \\ r'(i) &= r(i) + 1, \quad r'(j) = r(j) \quad \text{für } j \neq i, \\ S_i(r) &= r' \quad \text{mit} \\ r'(i) &= r(i) - 1, \quad r'(j) = r(j) \quad \text{für } j \neq i, \\ t_i(r) &= \mathbf{t} \iff r(i) = 0. \end{aligned}$$

Input-Output für  $I = \mathbb{N}^m$  und  $J = \mathbb{N}^n$ :

$\delta_m(x_1, \dots, x_m) = r$  mit  
 $r(i) = x_i$  für  $1 \leq i \leq m$  und  $r(j) = 0$  für  $j > m$ .

$\beta_n(r) = (r(1), \dots, r(n))$ .

Registermaschine mit  $I = \mathbb{N}^m$  und  $J = \mathbb{N}^n$ :  $\text{RM}_n^m$

## Programme:

Sei  $M = (S, O, T, \delta, \beta)$  eine Maschine. Ein String der Gestalt

- $a = r: f$  then  $p$

für  $r, p \in \mathbb{N}$  und  $f \in O$  heißt *Funktionsanweisung* zu  $M$ . Dabei heißt  $r$  die *Kennmarke* von  $a$  und  $p$  die *Sprungmarke* von  $a$ .

Ein String

- $b = r: \text{if } t \text{ then } p \text{ else } q$

heißt *Testanweisung* zu  $M$ .  $r$  heißt *Kennmarke* von  $b$ ,  $p$  und  $q$  heißen *Sprungmarken* von  $b$ .

Eine *Anweisung* ist eine Funktions- oder Testanweisung.

$P = (A, n)$  ist ein *Programm* zu  $M$  wenn folgendes gilt:

(P1)  $n$  ist eine natürliche Zahl, die *Startmarke*.

(P2)  $A$  ist eine endliche Menge von Anweisungen.

(P3) verschiedene Anweisungen in  $A$  haben verschiedene Kennmarken.

*Endmarke*: Sprungmarke aber nicht Kennmarke.

Bezeichnungen:  $KM(P)$ ,  $SM(P)$ ,  $EM(P)$ .

Bedingung (P3): Determinismus.

Ohne (P3): nondeterministisches Programm.

### Beispiel:

Sei  $M = \text{RM}_1^2$  und  $P = (A, 1)$  für

$A = \{1: \text{if } t_2 \text{ then } 0 \text{ else } 2,$

2:  $A_1$  then 3,

3:  $S_2$  then 1}

Dann ist  $P$  ein Programm zu  $M$ .

$P$ : zwei Funktions-, eine Testanweisung.

1: Startmarke.

0: Endmarke.

$$KM(P) = \{1, 2, 3\},$$

$$SM(P) = \{0, 1, 2, 3\},$$

$$EM(P) = \{0\}.$$

**Zusammensetzung** von Programmen:

$$\begin{aligned} P_1 &= (A_1, p_1), \\ P_2 &= (A_2, p_2). \end{aligned}$$

Gilt

$$KM(A_1) \cap KM(A_2) = \emptyset$$

dann sind

$$(A_1 \cup A_2, p_1), (A_1 \cup A_2, p_2)$$

auch Programme.

Gilt außerdem

$$\begin{aligned} EM(P_1) &= \{p_2\} \text{ und} \\ SM(P_2) \cap KM(P_1) &= \emptyset \end{aligned}$$

so ist

$$(A_1 \cup A_2, p_1)$$

die *Zusammensetzung* von  $P_1$  und  $P_2$ .

**Rechenprozesse:**

*Konfigurationen:*

Sei  $M = (S, O, T, \delta, \beta)$ .

Eine *Konfiguration* ist ein Paar aus  $\mathbb{N} \times S$ .

Ist  $(n, s)$  eine Konfiguration und  $n$  die Startmarke eines Programmes  $P$  dann heißt  $(n, s)$  *Anfangskonfiguration* für  $P$ .

Ist  $n \in EM(P)$  dann heißt  $(n, s)$  eine *Endkonfiguration* von  $P$ . die Menge aller Endkonfigurationen von  $P$  wird mit  $EK(P)$  bezeichnet.

- Rechenprozess =  
Folge von Konfigurationen.

## Rechenschrittfunktion:

Sei  $P = (A, n)$  ein Programm zur Maschine  $M$ .

Wir definieren eine Funktion  $RS_P$  (die *Rechenschrittfunktion* zu  $P$ ) welche Konfigurationen in Konfigurationen überführt:

- $RS_P(m, s) = (m, s)$  wenn  $m \notin KM(P)$ ,
- $RS_P(m, s) = (p, f(s))$  wenn  $m: \underline{f \text{ then } p \in A}$ ,
- $RS_P(m, s) = (p, s)$  wenn  $m: \underline{\text{if } t \text{ then } p \text{ else } q \in A} \text{ und } t(s) = \mathbf{t}$ ,
- $RS_P(m, s) = (q, s)$  wenn  $m: \underline{\text{if } t \text{ then } p \text{ else } q \in A} \text{ und } t(s) = \mathbf{f}$ .

## iterierte Rechenschrittfunktion:

$RSI_P: \mathbb{N} \times S \times \mathbb{N} \rightarrow \mathbb{N} \times S$ .

$$RSI_P(m, s, 0) = (m, s),$$

$$RSI_P(m, s, n+1) = RSI_P(RSI_P(m, s, n))$$

Ist  $P = (A, n)$  und  $(n, s)$  Anfangskonfiguration für  $P$  dann

- $RSI_P(n, s, m)$  = die Konfiguration nach  $m$  Programmschritten.

Rechenprozess auf  $(n, s)$  terminiert wenn ein  $m$  existiert mit

$$RSI_P(n, s, m) \in EK(P).$$

Formal definieren wir das *Terminationsprädikat*  $T$ :

$$T(P, n, s) \iff (\exists m) RSI_P(n, s, m) \in EK(P)$$

### *Nicht-terminierende Programme:*

Sei  $P = (A, 1)$  das Programm zur Turingmaschine mit

$$A = \{1: d_B \text{ then } 1\}.$$

Dann gilt offensichtlich

- $RSI_P(1, s, m) = (1, s)$  für  $m \in \mathbb{N}$ ,
- 1 ist nicht Endmarke

$$T(P, 1, s) = \mathbf{f} \text{ für alle } s \in S.$$

- Programme definieren *partielle* Funktionen.
- definiert wenn das Programm terminiert.
- Rechenzeit ist partielle Funktion.

### **Rechenzeit:**

$$M = (S, O, T, \delta, \beta).$$

$$P = (A, n).$$

$t_P: I \rightarrow \mathbb{N}$  (*Rechenzeit von P*):

$$t_P(x) =$$

$$\min\{m \mid RSI_P(n, \delta(x), m) \in EK(P)\}.$$

$t_P$  ist im allgemeinen nur eine partielle Funktion auf  $I$ . Für den Definitionsbereich gilt:

$$D(t_P) = \{x \mid x \in I, T(P, n, \delta(x)) = \mathbf{t}\}.$$

### Rechenendschrittfunktion:

$M = (S, O, T, \delta, \beta)$  und  
 $\delta: I \rightarrow S, \beta: S \rightarrow J$ .

Die *Rechenendschrittfunktion*  $ReS_P$  von  
 $P = (A, n)$  ist dann eine partielle Funkti-  
on  $I \rightarrow J$  definiert als:

$$ReS_P(x) = (\beta \circ J_2)(RSI_P(n, \delta(x), t_P(x))).$$

wobei  $J_2(p, s) = s$  für alle Konfiguratio-  
nen  $(p, s)$  gilt.

Da  $\delta, \beta, I_2$  und  $RSI_P$  totale Funktionen  
sind gilt

$$D(ReS_P) = D(t_P).$$

### Maschinen-berechenbare (partielle) Funktionen:

$M = (S, O, T, \delta, \beta)$  mit  
 $\delta: I \rightarrow S$ .

Die Menge

$$F(M) = \{ReS_P \mid P \text{ Programm zu } M\}$$

heißt die Klasse der

- *M-berechenbaren partiellen Funktionen*.

Die Menge

$$F^t(M) = \{f \mid f \in F(M), D(f) = I\}$$

heißt die Klasse der

- *M-berechenbaren totalen Funktionen*.

$$plus \in F^t(TM_1^2) \cap F^t(RM_1^2).$$

$M = \text{RM}_1^2$  und  $P = (A, 1)$  für

$A = \{1: \text{if } t_2 \text{ then } 0 \text{ else } 2,$   
            $2: A_1 \text{ then } 3,$   
            $3: S_2 \text{ then } 1\}$

Sei  $x = (3, 2)$ .

Dann ist  $\delta(x) = (3, 2, 0, \dots)$ .

Anfangskonfiguration für  $P$ :

$$k_0 = (1, (3, 2, 0, \dots)).$$

Konfigurationsfolge:

$$\begin{aligned} k_0 &= (1, (3, 2, 0, \dots)), & k_1 &= (2, (3, 2, 0, \dots)), \\ k_2 &= (3, (4, 2, 0, \dots)), & k_3 &= (1, (4, 1, 0, \dots)), \\ k_4 &= (2, (4, 1, 0, \dots)), & k_5 &= (3, (5, 1, 0, \dots)), \\ k_6 &= (1, (5, 0, 0, \dots)), & k_7 &= (0, (5, 0, \dots)). \end{aligned}$$

$$\begin{aligned} k_0 &= (1, (3, 2, 0, \dots)), & k_1 &= (2, (3, 2, 0, \dots)), \\ k_2 &= (3, (4, 2, 0, \dots)), & k_3 &= (1, (4, 1, 0, \dots)), \\ k_4 &= (2, (4, 1, 0, \dots)), & k_5 &= (3, (5, 1, 0, \dots)), \\ k_6 &= (1, (5, 0, 0, \dots)), & k_7 &= (0, (5, 0, \dots)). \end{aligned}$$

$k_7$ : Endkonfiguration.

$$t_P(3, 2) =$$

$$\min\{i \mid RSI_P(k_0, i) \in EK(P)\} = 7.$$

Zur Ausgabe dient damit die Konfiguration  
on  $k_7$ .

$$\beta_1(5, 0, \dots) = 5 \text{ und } J_2(k_7) = (5, 0, \dots)$$

$$ReS_P(2, 3) = (\beta_1 \circ J_2)(k_7) = \beta_1(J_2(k_7)) =$$

**Achtung:**

- $ReS_P = plus$ .
- $ReS_P(3, 2) = ReS_P(2, 3)$ .
- $t_P(3, 2) \neq t_P(2, 3)!$