Übersicht

Übersicht

Einführung

- Welche Funktionen können
  - von Computern
  - mit funktionaler Programmierung
  - mit imperativer Programmierung
  - ...

  berechnet werden?
- Stimmen diese Funktionsklassen überein?
- Was bedeutet *Berechenbarkeit* einer Funktion?
- Existieren Funktionen, die *nicht* berechnet werden können?
- Kann zu jeder Fragestellung eine Antwort berechnet werden? Gibt es *nicht*entscheidbare Probleme?

Einführung

- Maschinenmodelle und deren Ausdrucksfähigkeit
  - Beispiel: Registermaschine
  - Abstrakte Maschinen und Programmierkonzepte
  - Church'sche These
- Berechenbarkeit und Entscheidbarkeit
  - Beispiele für nichtberechenbare Funktionen
  - Beispiele für nichtentscheidbare Probleme

- Literatur: [Saake&Sattler] (Kapitel 6 und 7.1)

Übersicht

Maschinenmodelle

- Zentrale Fragestellung: *Berechenbarkeit* von Funktionen
- Lege „Spielregeln“ fest
  - Algorithmus beschrieben durch Befehle an einen Rechner
  - Definiere dazu ein abstraktes Rechnermodell
- Beispiel: Registermaschine – „ein Idealisierter Computer“
- Abstrakte Maschine –
  Allgemeines Konzept für ein Maschinenmodell
- Idee: *Abbildung* von
  - Rechnerarchitekturen,
  - Programmierparadigmen oder (-sprachen),
  - ...

  auf abstrakte Maschinen
- Mache verschiedene Konzepte damit vergleichbar

Registermaschine

- Einfaches abstraktes Rechnermodell,
  ähnlich einem „richtigen“ Computer
- Eine *Registermaschine* besteht aus Registern und Programm.
- Jedes Register *speichert* einen ganzzahligen Wert
  - B ist der Befehlszähler
  - C_0 heißt Arbeitsregister oder Akkumulator
  - $\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3, \ldots$ sind (potentiell unendlich viele) Speicherregister
- Konfiguration = momentaner Wert aller Register als Tupel

$$(\mathrm{b}, \mathrm{c}_0, \mathrm{c}_1, \ldots, \mathrm{c}_i, \ldots)$$

- Programm = (endliche) Folge von Befehlen
- Jeder Befehl $\mathtt{X}$ ändert die Konfiguration:

$$(\mathrm{b}, \mathrm{c}_0, \mathrm{c}_1, \ldots) \overset{\mathtt{X}}{\mapsto} (\mathrm{b}', \mathrm{c}_0', \mathrm{c}_1', \ldots)$$

Befehlsübersicht (1)

- Ein-/Ausgabe

Befehl	Argument	Semantik
`LOAD`	$i > 0$	$b' = b+1 \wedge c_0' = c_i \wedge c_j' = c_j$ für $j \neq 0$
`CLOAD`	$i \geqslant 0$	$b' = b+1 \wedge c_0' = i \wedge c_j' = c_j$ für $j \neq 0$
`STORE`	$i > 0$	$b' = b+1 \wedge c_i' = c_0 \wedge c_j' = c_j$ für $j \neq i$

  - Ein-/Ausgabe von/nach $C_i (i > 0)$ über Akkumulator C_0
  - Befehlszähler wird immer um 1 erhöht

- Manipulation (Es gilt jeweils $c_j' = c_j$ für $j \neq 0$.)

`ADD`	$i > 0$	$b' = b+1 \wedge c_0' = c_0 + c_i$
`CADD`	$i > 0$	$b' = b+1 \wedge c_0' = c_0 + i$
`SUB`	$i > 0$	$b' = b+1 \wedge c_0' = \begin{cases} c_0 \geqslant c_i: & c_0 - c_i \\ c_0 < c_i: & 0 \end{cases}$
`CSUB`	$i > 0$	$b' = b+1 \wedge c_0' = \begin{cases} c_0 \geqslant i: & c_0 - i \\ c_0 < i: & 0 \end{cases}$

Befehlsübersicht (2)

- Manipulation (Es gilt jeweils $c'_j = c_j$ für $j \neq 0$.)

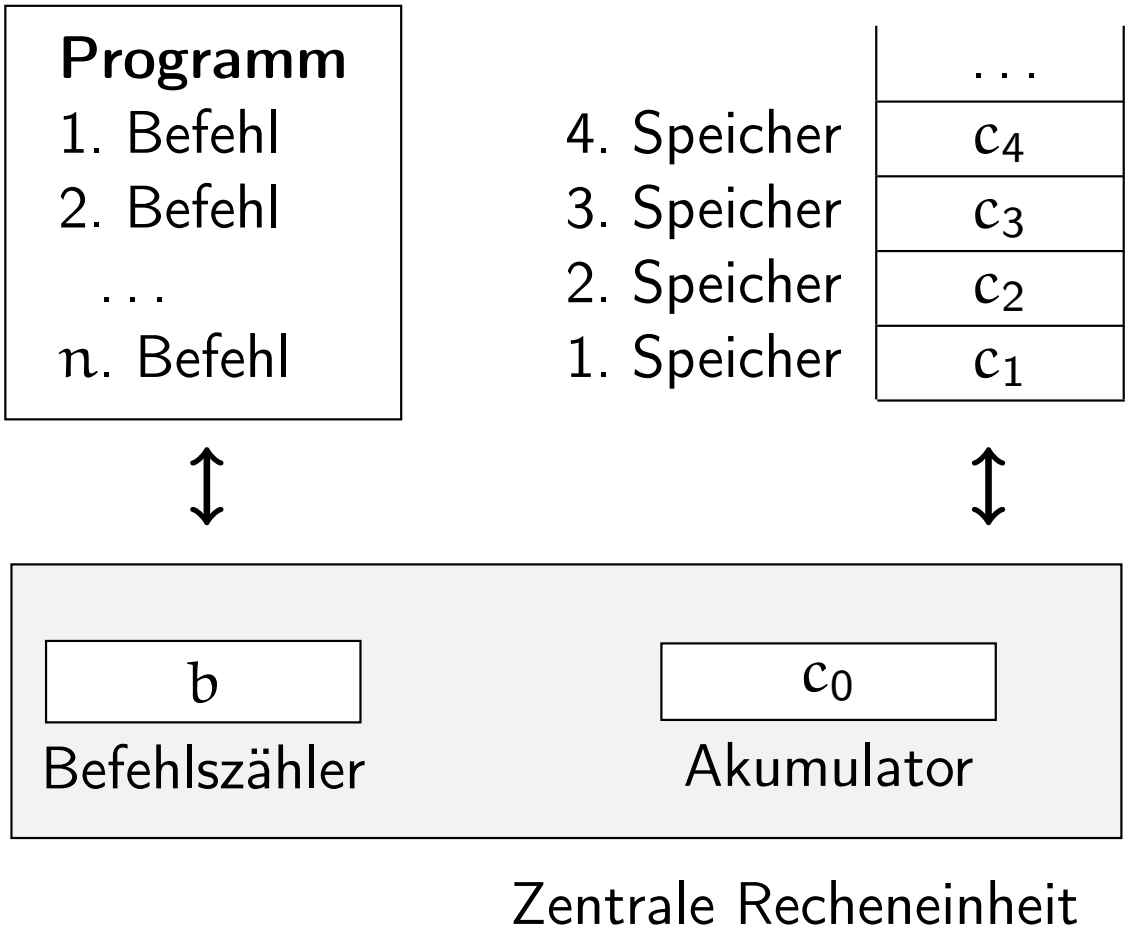
Befehl	Argument	Semantik
`MULT`	$i > 0$	$b' = b+1 \;\wedge\; c'_0 = c_0 \cdot c_i$
`CMULT`	$i > 0$	$b' = b+1 \;\wedge\; c'_0 = c_0 \cdot i$
`DIV`	$i > 0$	$b' = b+1 \;\wedge\; c'_0 = \lfloor c_0 / c_i \rfloor$
`CDIV`	$i > 0$	$b' = b+1 \;\wedge\; c'_0 = \lfloor c_0 / i \rfloor$

- Sprungbefehle (Es gilt jeweils $c'_j = c_j$ für $j \geqslant 0$.)

`GOTO`	$i > 0$	$b' = i$
`IF` $c_0 = 0$ `GOTO`	$i > 0$	$b' = \begin{cases} c_0 = 0: & i \\ c_0 \neq 0: & b+1 \end{cases}$
`END`		$b' = b$

  - Sprung (`GOTO`) und bedingter Sprung (`IF` $c_0 = 0$ `GOTO`)
  - *Stoppbefehl* `END` = Ende der Berechnung, denn Konfiguration bleibt *unverändert*

Aufbau einer Registermaschine



Beispiel: Registermaschine M_1

- Startkonfiguration $b = 1, c_0 = 0, c_1 = 32, c_2 = 5, c_3 = 0$

```
1   LOAD 1
2   DIV 2
3   MULT 2
4   STORE 3
5   LOAD 1
6   SUB 3
7   STORE 3
8   END
```

$$
\begin{aligned}
(1,0,32,5,0,\dots) &\overset{1}{\mapsto} (2,32,32,5,0,\dots)\\
&\overset{2}{\mapsto} (3,6,32,5,0,\dots)\\
&\overset{3}{\mapsto} (4,30,32,5,0,\dots)\\
&\overset{4}{\mapsto} (5,30,32,5,30,\dots)\\
&\overset{5}{\mapsto} (6,32,32,5,30,\dots)\\
&\overset{6}{\mapsto} (7,2,32,5,30,\dots)\\
&\overset{7}{\mapsto} (8,2,32,5,2,\dots)\\
&\overset{8}{\mapsto} (8,2,32,5,2,\dots)
\end{aligned}
$$

Beispiel: Registermaschine M_1

- Allgemeine Betrachtung $b = 1, c_0 = 0, c_1 = n, c_2 = m, c_3 = 0$
- Sei $q = \lfloor \frac{n}{m} \rfloor$, d.h. $n = q \cdot m + r$ mit $0 \leqslant r < m$.

```
1   LOAD 1
2   DIV 2
3   MULT 2
4   STORE 3
5   LOAD 1
6   SUB 3
7   STORE 3
8   END
```

$$
\begin{aligned}
(1, 0, n, m, 0, \dots) &\overset{1}{\mapsto} (2, n, n, m, 0, \dots) \\
&\overset{2}{\mapsto} (3, q, n, m, 0, \dots) \\
&\overset{3}{\mapsto} (4, q \cdot m, n, m, 0, \dots) \\
&\overset{4}{\mapsto} (5, q \cdot m, n, m, q \cdot m, \dots) \\
&\overset{5}{\mapsto} (6, n, n, m, q \cdot m, \dots) \\
&\overset{6}{\mapsto} (7, r, n, m, q \cdot m, \dots) \\
&\overset{7}{\mapsto} (8, r, n, m, r, \dots) \\
&\overset{8}{\mapsto} (8, r, n, m, r, \dots)
\end{aligned}
$$

- M_1 berechnet den Rest r der ganzzahligen Division n/m

Berechnete Funktion

Berechnete Funktion einer Registermaschine

Eine Registermaschine M berechnet die Funktion

$$f : \mathbb{N}_0^n \to \mathbb{N}_0^m \quad \text{mit} \quad f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m),$$

wenn es Zahlen $i_1, i_2, \ldots, i_m$ gibt, so dass M *jede* Konfiguration

$$(1, 0, x_1, x_2, \ldots, x_n, 0, 0, \ldots)$$

in eine Konfiguration $(b, c_0, c_1, \ldots)$ überführt, für die gilt

- b ist die Nummer einer `END` Anweisung, und
- $c_{i_j} = y_j$ für $1 \leqslant j \leqslant m$.

- Das Ergebnis $(y_1, \ldots, y_m)$ steht in den Registern $C_{i_1}, \ldots, C_{i_m}$.
- Beispiel: M_1 berechnet $f(x_1, x_2) = x_1 - \lfloor \frac{x_1}{x_2} \rfloor x_2$

Einfache Beispiele

- Addition $f(x_1, x_2) = x_1 + x_2$ (Ergebnis in C_3 also $i_1 = 3$.)

```
1   LOAD 1
2   ADD 2
3   STORE 3
4   END
```

- $f(x) = \begin{cases} x = 0: & 0 \\ x \neq 0: & \perp \end{cases}$

```
1   LOAD 1
2   IF c_0 = 0 GOTO 4
3   GOTO 2
4   END
```

Beispiel: Registermaschine M_2

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

- M_2 berechnet $f(x, y) = x^y$
- für Startkonfiguration $(1, 0, x, y, 0, \dots)$
- für Endkonfiguration gilt $C_3 = f(x, y)$

Beispiel: Registermaschine M_2

- Beispiel: $f(2,3) = 2^3 = 8$
- Startkonfiguration $(1,0,2,3,0,\dots)$
- $(1,0,2,3,0,\dots) \overset{1}{\mapsto} (2,1,2,3,\dots) \overset{2}{\mapsto} (3,1,2,3,1,\dots) \overset{3}{\mapsto}$
  $(4,3,2,3,1,\dots) \overset{4}{\mapsto} (5,3,2,3,1,\dots) \overset{5}{\mapsto} (6,1,2,3,1,\dots) \overset{6}{\mapsto}$
  $(7,2,2,3,1,\dots) \overset{7}{\mapsto} (8,2,2,3,2,\dots) \overset{8}{\mapsto} (9,3,2,3,2,\dots) \overset{9}{\mapsto}$
  $(10,2,2,3,2,\dots) \overset{10}{\mapsto} (11,2,2,2,2,\dots) \overset{11}{\mapsto} (4,2,2,2,2,\dots) \overset{4}{\mapsto}$
  $(5,2,2,2,2,\dots) \overset{5}{\mapsto} (6,2,2,2,2,\dots) \overset{6}{\mapsto} (7,4,2,2,2,\dots) \overset{7}{\mapsto}$
  $(8,4,2,2,4,\dots) \overset{8}{\mapsto} (9,2,2,2,4,\dots) \overset{9}{\mapsto} (10,1,2,2,4,\dots) \overset{10}{\mapsto}$
  $(11,1,2,1,4,\dots) \overset{11}{\mapsto} (4,1,2,1,4,\dots) \overset{4}{\mapsto} (5,1,2,1,4,\dots) \overset{5}{\mapsto}$
  $(6,4,2,1,4,\dots) \overset{6}{\mapsto} (7,8,2,1,4,\dots) \overset{7}{\mapsto} (8,8,2,1,8,\dots) \overset{8}{\mapsto}$
  $(9,1,2,1,8,\dots) \overset{9}{\mapsto} (10,0,2,1,8,\dots) \overset{10}{\mapsto} (11,0,2,0,8,\dots) \overset{11}{\mapsto}$
  $(4,0,2,0,8,\dots) \overset{4}{\mapsto} (12,0,2,0,8,\dots) \overset{12}{\mapsto} (12,0,2,0,8,\dots) \overset{12}{\mapsto} \dots$

Beispiel: Registermaschine M_4

```
 1   LOAD 1
 2   CSUB 1
 3   IF c_0 = 0 GOTO 19
 4   CLOAD 2
 5   STORE 2
 6   LOAD 1
 7   SUB 2
 8   IF c_0 = 0 GOTO 21
 9   LOAD 1
10   DIV 2
11   MULT 2
12   STORE 3
13   LOAD 1
14   SUB 3
15   IF c_0 = 0 GOTO 19
16   LOAD 2
17   CADD 1
18   GOTO 5
19   STORE 2
20   GOTO 23
21   CLOAD 1
22   STORE 2
23   END
```

- M_2 berechnet $f(x) = \begin{cases} x \text{ ist keine Primzahl}: & 0 \\ x \text{ ist Primzahl}: & 1 \end{cases}$
- für Startkonfiguration $(1, 0, x, 0, \dots)$

Beispiel: Registermaschine M_4

- Startkonfiguration $(1, 0, \mathbf{2}, \dots)$

  $(1,0,2,\dots) \overset{1}{\mapsto} (2,2,2,\dots) \overset{2}{\mapsto} (3,1,2,\dots) \overset{3}{\mapsto} (4,1,2,\dots) \overset{4}{\mapsto}$
  $(5,2,2,\dots) \overset{5}{\mapsto} (6,2,2,2,\dots) \overset{6}{\mapsto} (7,2,2,2,\dots) \overset{7}{\mapsto} (8,0,2,2,\dots) \overset{8}{\mapsto}$
  $(21,0,2,2,\dots) \overset{21}{\mapsto} (22,1,2,2,\dots) \overset{22}{\mapsto} (23,1,2,1,\dots) \overset{23}{\mapsto}$
  $(23,1,2,1,\dots) \overset{23}{\mapsto} \dots$

- Startkonfiguration $(1, 0, \mathbf{6}, \dots)$

  $(1,0,6,\dots) \overset{1}{\mapsto} (2,6,6,\dots) \overset{2}{\mapsto} (3,5,6,\dots) \overset{3}{\mapsto} (4,5,6,\dots) \overset{4}{\mapsto}$
  $(5,2,6,\dots) \overset{5}{\mapsto} (6,2,6,2,\dots) \overset{6}{\mapsto} (7,6,6,2,\dots) \overset{7}{\mapsto} (8,4,6,2,\dots) \overset{8}{\mapsto}$
  $(9,4,6,2,\dots) \overset{9}{\mapsto} (10,6,6,2,\dots) \overset{10}{\mapsto} (11,3,6,2,\dots) \overset{11}{\mapsto}$
  $(12,6,6,2,\dots) \overset{12}{\mapsto} (13,6,6,2,6,\dots) \overset{13}{\mapsto} (14,6,6,2,6,\dots) \overset{14}{\mapsto}$
  $(15,0,6,2,6,\dots) \overset{15}{\mapsto} (19,0,6,2,6,\dots) \overset{19}{\mapsto} (20,0,6,0,6,\dots) \overset{20}{\mapsto}$
  $(23,0,6,0,6,\dots) \overset{23}{\mapsto} (23,0,6,0,6,\dots) \overset{23}{\mapsto} \dots$

Beispiel: Registermaschine M_4

- Startkonfiguration $(1, 0, \mathbf{5}, \dots)$

$$(1,0,5,\dots) \overset{1}{\mapsto} (2,5,5,\dots) \overset{2}{\mapsto} (3,4,5,\dots) \overset{3}{\mapsto} (4,4,5,\dots) \overset{4}{\mapsto} (5,2,5,\dots) \overset{5}{\mapsto}$$
$$(6,2,5,2,\dots) \overset{6}{\mapsto} (7,5,5,2,\dots) \overset{7}{\mapsto} (8,3,5,2,\dots) \overset{8}{\mapsto} (9,3,5,2,\dots) \overset{9}{\mapsto}$$
$$(10,5,5,2,\dots) \overset{10}{\mapsto} (11,2,5,2,\dots) \overset{11}{\mapsto} (12,4,5,2,\dots) \overset{12}{\mapsto} (13,4,5,2,4,\dots) \overset{13}{\mapsto}$$
$$(14,5,5,2,4,\dots) \overset{14}{\mapsto} (15,1,5,2,4,\dots) \overset{15}{\mapsto} (16,1,5,2,4,\dots) \overset{16}{\mapsto} (17,2,5,2,4,\dots) \overset{17}{\mapsto}$$
$$(18,3,5,2,4,\dots) \overset{18}{\mapsto} (5,3,5,2,4,\dots) \overset{5}{\mapsto} (6,3,5,3,4,\dots) \overset{6}{\mapsto} (7,5,5,3,4,\dots) \overset{7}{\mapsto}$$
$$(8,2,5,3,4,\dots) \overset{8}{\mapsto} (9,2,5,3,4,\dots) \overset{9}{\mapsto} (10,5,5,3,4,\dots) \overset{10}{\mapsto} (11,1,5,3,4,\dots) \overset{11}{\mapsto}$$
$$(12,3,5,3,4,\dots) \overset{12}{\mapsto} (13,3,5,3,3,\dots) \overset{13}{\mapsto} (14,5,5,3,3,\dots) \overset{14}{\mapsto} (15,2,5,3,3,\dots) \overset{15}{\mapsto}$$
$$(16,2,5,3,3,\dots) \overset{16}{\mapsto} (17,3,5,3,3,\dots) \overset{17}{\mapsto} (18,4,5,3,3,\dots) \overset{18}{\mapsto} (5,4,5,3,3,\dots) \overset{5}{\mapsto}$$
$$(6,4,5,4,3,\dots) \overset{6}{\mapsto} (7,5,5,4,3,\dots) \overset{7}{\mapsto} (8,1,5,4,3,\dots) \overset{8}{\mapsto} (9,1,5,4,3,\dots) \overset{9}{\mapsto}$$
$$(10,5,5,4,3,\dots) \overset{10}{\mapsto} (11,1,5,4,3,\dots) \overset{11}{\mapsto} (12,4,5,4,3,\dots) \overset{12}{\mapsto} (13,4,5,4,4,\dots) \overset{13}{\mapsto}$$
$$(14,5,5,4,4,\dots) \overset{14}{\mapsto} (15,1,5,4,4,\dots) \overset{15}{\mapsto} (16,1,5,4,4,\dots) \overset{16}{\mapsto} (17,4,5,4,4,\dots) \overset{17}{\mapsto}$$
$$(18,5,5,4,4,\dots) \overset{18}{\mapsto} (5,5,5,4,4,\dots) \overset{5}{\mapsto} (6,5,5,5,4,\dots) \overset{6}{\mapsto} (7,5,5,5,4,\dots) \overset{7}{\mapsto}$$
$$(8,0,5,5,4,\dots) \overset{8}{\mapsto} (21,0,5,5,4,\dots) \overset{21}{\mapsto} (22,1,5,5,4,\dots) \overset{22}{\mapsto} (23,1,5,1,4,\dots) \overset{23}{\mapsto}$$
$$(23,1,5,1,4,\dots) \overset{23}{\mapsto} \dots$$

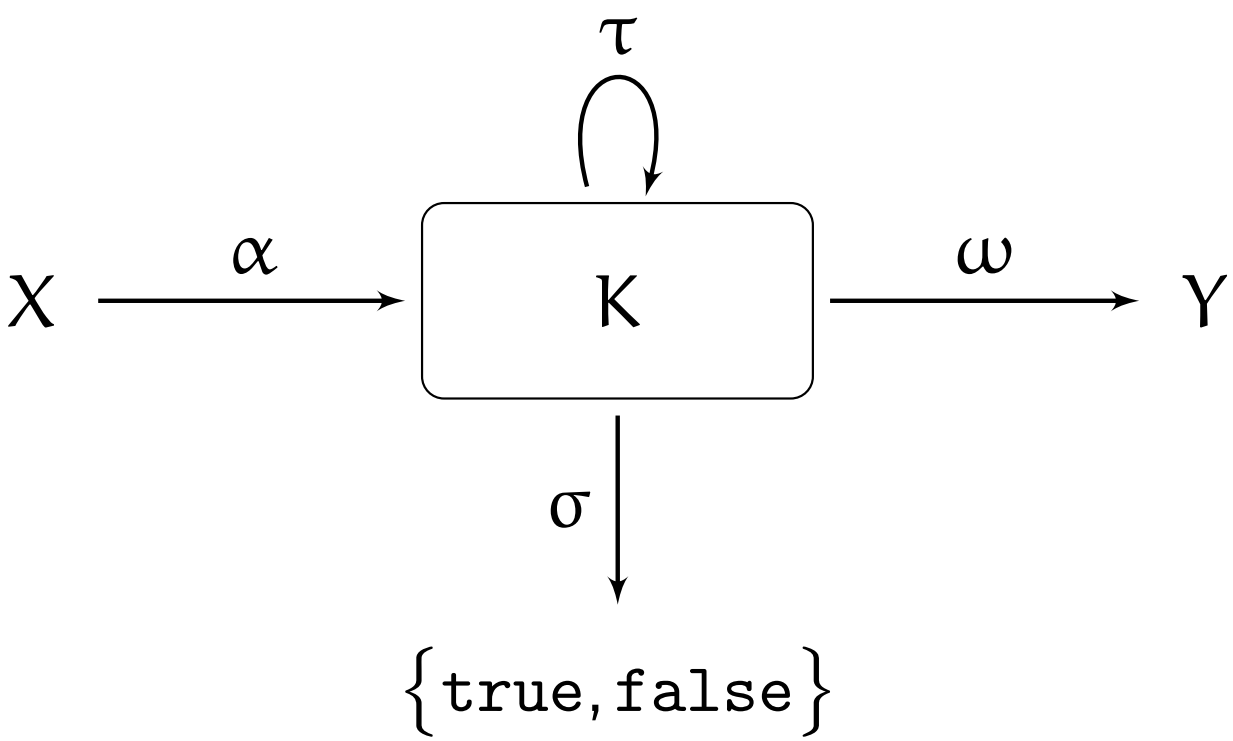
Bemerkungen

- *Imperative* Programmierung der Registermaschine
- Befehlssatz könnte weiter eingeschränkt werden z.B. Manipulation nur durch `SUCC`, `PRED`
- Registermaschine ähnelt Prozessor (CPU) eines Computers
  - $\{B \cup C_0\} \approx$ *Registersatz* (schneller Zwischenspeicher)
  - Register $C_i (i \geqslant 1) \approx$ *Hauptspeicher*
- Hauptunterschiede zu CPU
  - *Keine Möglichkeit zum Funktionsaufruf! (Keine Rekursion!)*
    - Keine Möglichkeit zum *Speichern* von $c_0 = b$
    - Keine *indirekten* Sprünge $b = c_0$
  - Keine *indirekte* Addressierung $c_{c_0} = \ldots$ („Felder“)
  - Kein *gemeinsamer* Speicher für Programm und Daten
- Mehr zu Registermaschinen in [Saake&Sattler]
  - Kommentierte Beispiele $M_1, \ldots, M_4$ (Kapitel 6.1)
  - Java-Implementierung eines Interpreters siehe (Kapitel 6.5)

Abstrakte Maschinenmodelle

- Es gibt viele weitere Modelle neben der Registermaschine, z.B.
  - Turing-Maschine
  - Markov-Algorithmen, s. [Saake&Sattler] (Kapitel 6.3 und 6.5)
- Allgemeines Modell für *deterministische* Maschinen/Algorithmen:
- Abstrakte Maschine $M = (X, Y, K, \alpha, \omega, \tau, \sigma)$
  - X Menge von Eingabewerten
  - Y Menge von Ausgabewerten
  - K Menge von Konfigurationen
  - $\alpha : X \to K$ Eingabefunktion
  - $\omega : K \to Y$ Ausgabefunktion
  - $\tau\ K \to K$ Transitionsfunktion
  - $\sigma : K \to \texttt{bool}$ Stoppfunktion (markiert Endkonfiguration)
- Endkonfigurationen zu M: $E = \{k \in K \mid \sigma(k) = \texttt{true}\}$
  - Zustände, in denen eine Berechnung *terminiert*

Abstrakte Maschine



Funktionsweise einer abstrakten Maschine

1. Eingabewert $x \in X$ bestimmt
   Anfangskonfiguration $k_0 = \alpha(x) \in K$
2. Transitionsfunktion τ führt Konfiguration k_i über in Folgekonfiguration k_{i+1}, also

$$k_1 = \tau(k_0), \quad k_2 = \tau(k_1), \ \ldots, \ k_{i+1} = \tau(k_i), \ \ldots$$

   bis zum ersten Mal eine Endkonfiguration
   $k_j \in E \Leftrightarrow \sigma(k_j) = \texttt{true}$ erreicht wird.
3. Wird eine Endkonfiguration $k_j \in E$ erreicht, dann wird der Ausgabewert $\omega(k_j) \in Y$ berechnet.

- Dabei muss nicht zwingend eine Endkonfiguration erreicht werden.
  - Dann terminiert der Algorithmus nicht.
  - D.h., das Ergebnis ist undefiniert ($\perp$).

Berechnete Funktion

- Eine abstrakte Maschine M berechnet die *partielle* Funktion

$$f_M : X \to Y$$

- Für eine Eingabe $x \in X$ gibt es zwei Möglichkeiten
  - M terminiert nicht, dann ist das Ergebnis *undefiniert* ($\perp$)
  - M terminiert, dann ist das Ergebnis der Ausgabewert $y \in Y$
- Die Laufzeit von M für die Eingabe $x \in X$ ist

$$t_M(x) \;=\; \min\left\{n \,|\, \sigma(\tau^n(\alpha(x))) = \texttt{true}\right\}$$

  - Kleinste Anzahl von Übergängen, bis M terminiert.
  - $t_M = \perp$ falls es keine solche Zahl gibt.
- Damit ist die von M berechnete Funktion

$$f_M(x) \;=\; \begin{cases} t_M(x) \neq \perp: & \omega\left(\tau^{t_M(x)}(\alpha(x))\right) \\ t_M(x) = \perp: & \perp \end{cases}$$

Abbildung auf abstrakte Maschine

- Registermaschine als Spezialfall einer abstrakten Maschine
  - $X = \mathbb{N}_0^n$, $Y = \mathbb{N}_0^m$
  - Konfigurationen der Registermaschine: K
  - τ wertet Befehle aus, σ prüft auf `END`
  - Kann Funktionsweise auf abstrakte Maschine *abbilden*
- Gleiches für Programmiersprachen
  - Funktionale Programmierung
    - K sind Terme ohne Unbestimmte
    - τ wertet Terme nach deterministischer Vorschrift aus
  - Imperative Programmierung
  - S.a. [Saake&Sattler] (Kapitel 6.2)
- *Damit folgt*:
  Alle können die gleiche Klasse von Funktionen berechnen!

Church'sche These

Church'sche These

Die Klasse der *intuitiv berechenbaren* Funktionen stimmt mit der Klasse der durch

- Imperative Programmierung, Funktionale Programmierung,
- Registermaschinen, Markov-Algorithmen, abstrakte Maschinen,
- ...

berechenbaren Funktionen überein.

- Auch *Church-Turing-These* (*Turing-berechenbare* Funktionen)
- Die These ist *nicht* beweisbar!
- Denn: Was bedeutet „intuitiv“ formal?

Universelle Programmierbarkeit

- Algorithmenmodell =
  - Programmiersprache (funktional, imperativ, . . .) oder
  - Maschinenmodell

Universelles Algorithmenmodell

Ein Algorithmenmodell heißt *universell*, wenn es alle berechenbaren Funktionen beschreiben kann.

- Synonym *universell* = *Turing-vollständig*
  - Definition über eine abstrakte Maschine
  - Modell kann alle Funktionen berechnen, die eine sog. universelle Turing-Maschine berechnen kann.
- Insbesondere Eigenschaften wie
  - Keine (prinzipielle) Beschränkung des Wertebereichs von Daten
  - Rekursion und/oder Iteration zur *bedingten* Wiederholung

Kurze Zusammenfassung

- Definition von berechenbaren Funktionen über Maschinenmodelle
  - Registermaschine
  - Abstrakte Maschine
- Abbildung von Algorithmenmodellen auf abstrakte Maschine
- Church'sche These
  - *intuitiv* berechenbar $\stackrel{!}{=}$ Maschinen-berechenbar
- *Universelle* Programmiersprache
  - Alle gängigen Programmiersprachen sind gleich mächtig.

Übersicht

- Frage
  - Gibt es Problemstellungen, für die *kein* Algorithmus zur Lösung existiert?
  - Kann man *alles* berechnen?

Definition (Berechenbarkeit)

Eine Funktion $f : \mathcal{X}^n \to \mathcal{Y}^m$ heißt *berechenbar*, wenn es einen Algorithmus gibt, der für Eingaben $x \in \mathcal{X}^n$ terminiert und $f(x) \in \mathcal{Y}^m$ (in endlicher Zeit) berechnet.

- Church'sche These:

  Die Menge der berechenbaren Funktionen entspricht allen jemals mit Computern berechenbaren Funktionen.

Existenz nichtberechenbarer Funktionen

- Wir wollen zeigen, dass es *nichtberechenbare* Funktionen gibt.
- *Idee*: Wir „zählen“ berechenbare Funktionen und Funktionen
- Zu jeder berechenbaren Funktion gibt es einen Algorithmus.
- Jeder *Algorithmus* lässt sich durch einen endlichen Text über einem festen, endlichen Alphabet beschreiben.
- *Wir wollen zeigen*: Es gibt mehr Funktionen als Algorithmen
- Vorüberlegung: *Abzählbarkeit*

Definition (Abzählbarkeit)

Eine Menge $\mathcal{M}$ heißt *abzählbar* (*unendlich*), wenn es eine bijektive Abbildung von $\mathcal{M}$ auf $\mathbb{N}$ gibt.

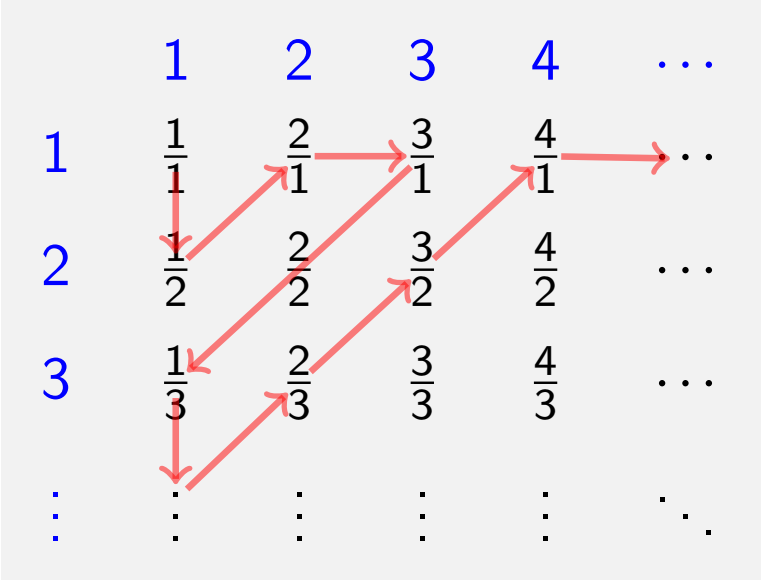
- $\mathbb{N}$ ist abzählbar.
- Die Menge der geraden Zahlen $\{2n | n \in \mathbb{N}\}$ ist abzählbar.
- Jede *unendliche* Teilmenge von $\mathbb{N}$ ist abzählbar.
- $\mathbb{Z}$ ist abzählbar: $f(n) = \begin{cases} n \text{ gerade}: & \frac{n}{2} \\ n \text{ ungerade}: & -\lfloor \frac{n}{2} \rfloor \end{cases}$

n	1	2	3	4	5	6	7	...
f(n)	0	1	-1	2	-2	3	-3	...

- Ist $\mathbb{Q}$ abzählbar?

Abzählbarkeit von $\mathbb{Q}$

- Betrachte zuerst $\frac{x}{y} \in \mathbb{Q}^+$ also Paar von natürlichen Zahlen
- Konstruiere lineare „Liste“, so dass Nachfolger eindeutig, z.B.



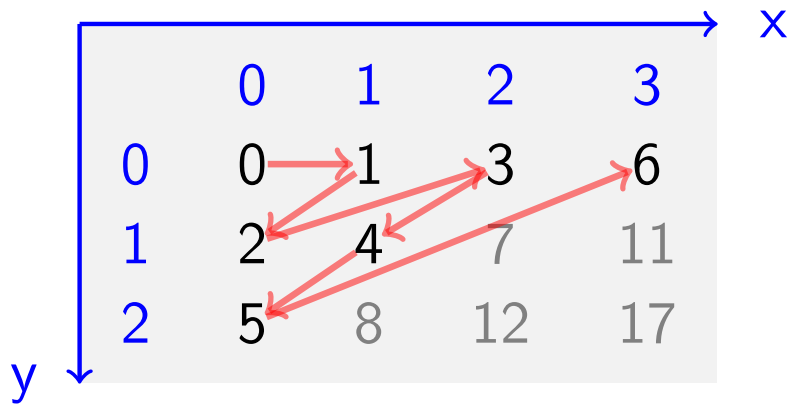
Allgemein: Abzählbarkeit von m-Tupeln

- Allgemeine Konstruktion: Cantorsche Paarungsfunktion

$$\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \text{ mit}$$

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

- Für Paare (x,y) mit $x,y \in \mathbb{N}_0$



- Menge aller m-Tupel über $\mathbb{N}$ ist abzählbar (durch $m-1$-malige Anwendung von π)

Abzählbarkeit von Texten

- Sei $\Sigma = \{a, b, \ldots\}$ mit $n = |\Sigma| \in \mathbb{N}$ ein *endliches* Alphabet
- Alle Texte über Σ in *alphabetischer* Reihenfolge

$$\Sigma^* = \{\varepsilon, a, b, \ldots, aa, ab, \ldots, ba, bb, \ldots, aaa, aab, \ldots\}$$

  - Aufzählung nach Wortlänge,
  - bei gleicher Länge lexikographisch
- Diese Reihenfolge ist *eindeutig*.
- Damit existiert eine *bijektive* Abbildung nach $\mathbb{N}$.
- Idee zur Konstruktion der Abbildung
  - Es gibt jeweils n^ℓ Worte der Länge ℓ.
  - Benötige Abbildung der Worte der Länge ℓ auf $\{1, 2, \ldots, n^\ell\}$.
- Σ^* *abzählbar* $\Rightarrow$ Menge aller Algorithmen *abzählbar*

Überabzählbare Mengen

Definition (Überabzählbare Menge)

Eine Menge heißt *überabzählbar*, wenn sie nicht abzählbar (und nicht endlich) ist.

- Anzahl Elemente ist größer als die in $\mathbb{N}$
- Jede Liste von Elementen $x_1, x_2, x_3, \ldots$ ist unvollständig
- Beispiel: Menge der reellen Zahlen $\mathbb{R}$

$\mathbb{R}$ ist überabzählbar

- Cantor'sches Diagonalargument (*Schon wieder Georg Cantor*!)
- Sei r_i eine unendliche Folge reeller Zahlen im Intervall $[0,1)$
- Seien a_{ij} die Dezimalstellen von r_i, also

$$\begin{aligned} r_1 &= 0, \underline{a_{11}}\, a_{12}\, a_{13}\, a_{14} \cdots \\ r_2 &= 0, a_{21}\, \underline{a_{22}}\, a_{23}\, a_{24} \cdots \\ r_3 &= 0, a_{31}\, a_{32}\, \underline{a_{33}}\, a_{34} \cdots \\ r_4 &= \cdots \\ &\vdots \end{aligned}$$

- Konstruiere $x = 0, x_1\, x_2\, x_3 \cdots$ aus Diagonalelementen $\underline{a_{ii}}$ mit

$$x_i = (a_{ii} + 1) \bmod 10$$

- $x \in [0,1)$ unterscheidet sich von *allen* Zahlen der Folge!
- Keine Folge enthält alle reellen Zahlen im Intervall $[0,1)$
- Intervall $[0,1)$ ist überabzählbar $\Rightarrow$ $\mathbb{R}$ ist überabzählbar

Überabzählbarkeit der Menge der Funktionen

- Wir zeigen: $\mathcal{F} = \{ f \mid f : \mathbb{N} \to \mathbb{N} \}$ überabzählbar
  - Gleiches Argument wie für $\mathbb{R}$ (Cantor'sches Diagonalargument)
- *Annahme*: $\mathcal{F}$ ist abzählbar
- Dann können wir alle $f \in \mathcal{F}$ auflisten als $F = \{f_1, f_2, \ldots\}$
- Sei nun $g : \mathbb{N} \to \mathbb{N}$ mit $g(x) = f_x(x) + 1$.
- Dann gilt für $i \in \mathbb{N}$: $\quad g(i) \neq f_i(i) \;\Rightarrow\; g \neq f_i$
- Damit gilt aber $g \in \mathcal{F}$ und $g \notin F$ ↯ Widerspruch zur Annahme!
- Menge der einstelligen Funktionen $\mathcal{F}$ ist überabzählbar
- *Die Menge der Funktionen ist überabzählbar!*

Existenz nichtberechenbarer Funktionen

- Wir haben folgendes gezeigt
  - Die Menge aller Algorithmen ist *abzählbar*.
  - Die Menge aller Funktionen ist *überabzählbar*.
- *Folgerung*: Es existieren nichtberechenbare Funktionen!
- Nichtberechenbarkeit i.a. schwerer zu zeigen als Berechenbarkeit
  - Für Berechenbarkeit genügt Angabe eines Algorithmus
  - Für Nichtberechenbarkeit zu zeigen: es gibt keinen Algorithmus
- Im folgenden zwei Beispiele
  - Halteproblem
  - Post'sches Korrespondenzproblem

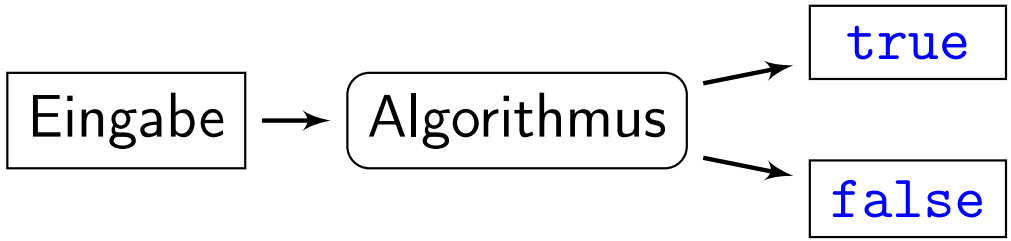
Definition (Entscheidbarkeit)

Eine Menge $A \in \Sigma^*$ heißt *entscheidbar*, wenn ihre charakteristische Funktion

$$\chi_A(\omega) = \begin{cases} \omega \in A: & 1 \\ \omega \notin A: & 0 \end{cases}$$

berechenbar ist.

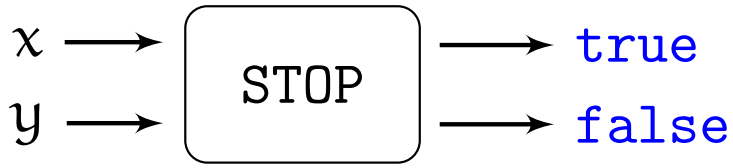
- Ein *Entscheidungsproblem* stellt sich wie folgt dar:



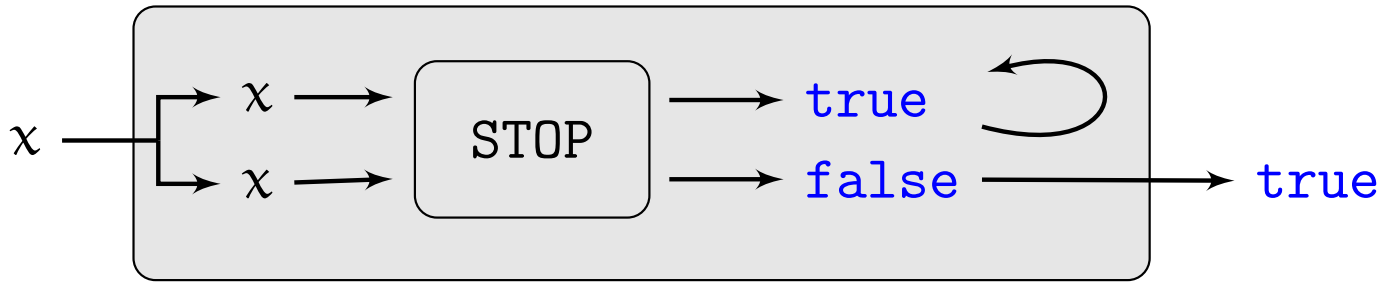
- Ein Problem heißt *entscheidbar*, wenn es einen solchen Algorithmus gibt, der für jede Eingabe terminiert (hält).

Halteproblem

- Halteproblem: *Hält Algorithmus* x *bei der Eingabe von* y*?*
- *Annahme*: Das Halteproblem ist entscheidbar.
- Dann gibt es einen Algorithmus/eine Maschine `STOP`



- Wir konstruieren eine neue Maschine `SELTSAM` wie folgt



- Hält `SELTSAM` für die Eingabe `SELTSAM`?
  - Wenn ja, dann liefert `STOP` $\Rightarrow$ Endlosschleife
  - Wenn nein, dann hält `SELTSAM` mit Ergebnis `true`

Halteproblem

- Hält `SELTSAM` für die Eingabe `SELTSAM`?
  - Dieses Problem heißt Spezielles Halteproblem
  - Das spezielle Halteproblem ist nicht entscheidbar
  - Konstruiere Widerspruch zur Annahme *`STOP` ist berechenbar*
- Somit ist auch das allgemeine Halteproblem

  Hält Algorithmus x *bei der Eingabe von* y*?*

  nicht entscheidbar!
- Formaler Beweis siehe z.B. [Saake&Sattler]
  - Ausführlich in Vorlesung *Grundlagen der Theoretischen Informatik* oder z.B. [Schöning]

Nichtenscheidebare Eigenschaften von Algorithmen

Satz von Rice

Jede nichttriviale semantische Eigenschaft von Algorithmen ist nichtentscheidbar.

- Formale Fassung siehe z.B. [Schöning]
- Beispiele
  - *Ist die berechnete Funktion total? Überall undefiniert? Injektiv? Surjektiv? Bijektiv? ...*
  - *Berechnen zwei gegebene Algorithmen die gleiche Funktion?*
  - *Ist ein gegebener Algorithmus korrekt? D.h., berechnet der die gegebene (gewünschte) Funktion?*
- Entscheidung/Nachweis jeweils nur *im Einzelfall* möglich.
- Es gibt dafür keine *allgemeine* Methode – keinen Algorithmus!

- Auch einfache Probleme können nichtentscheidbar sein ...

Post'sches Korrespondenzproblem

- Gegeben ist ein endliches Alphabet Σ und zwei gleichlange Listen von Worten über Σ

$$\begin{aligned} \alpha &= (\alpha_1, \alpha_2, \ldots, \alpha_n) \\ \beta &= (\beta_1, \beta_2, \ldots, \beta_n) \end{aligned}$$

  mit $\alpha_i, \beta_i \in \Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ und $n \geqslant 1$.

- Gesucht ist eine *Korrespondenz* in Form einer endlichen Folge $(i_1, i_2, \ldots, i_k)$, $i_j \in \{1, \ldots, n\}$ für $j = 1, 2, \ldots, k$, so dass gilt

$$\alpha_{i_1} \alpha_{i_2} \ldots \alpha_{i_k} \;=\; \beta_{i_1} \beta_{i_2} \ldots \beta_{i_k} \;.$$

Beispiel (1) zum Post'schen Korrespondenzproblem

- $\Sigma = \{0, 1\}$ und

$$\begin{aligned} \alpha &= (\,1\,,\,10111\,,\,10\,) \\ \beta &= (\,111\,,\,10\,,\,0\,) \end{aligned}$$

- Lösung: Korrespondenz $(2, 1, 1, 3)$, denn

$$\underbrace{10111}_{\alpha_2}\ \underbrace{1}_{\alpha_1}\ \underbrace{1}_{\alpha_1}\ \underbrace{10}_{\alpha_3} = \underbrace{10}_{\beta_2}\ \underbrace{111}_{\beta_1}\ \underbrace{111}_{\beta_1}\ \underbrace{0}_{\beta_3}$$

Beispiel (2) zum Post'schen Korrespondenzproblem

- $\Sigma = \{0, 1\}$ und

$$\begin{aligned} \alpha &= (\ 10\ ,\ 011\ ,\ 101\ ) \\ \beta &= (\ 101\ ,\ 11\ ,\ 011\ ) \end{aligned}$$

- Es existiert keine Korrespondenz als Lösung!
- Gäbe es eine Lösung, müsste sie mit dem Index $i_1 = 1$ beginnen

$$10 \cdots \stackrel{?}{=} 101 \cdots$$

- Nun müsste i_2 so gewählt werden, dass Wort $\alpha_{i_2} = 1 \cdots$
  - 1. Möglichkeit: $i_2 = 1$ und damit $10\ 10 \cdots \neq 101\ 101 \cdots$
  - 2. Möglichkeit: $i_2 = 3$ und damit $10\ 101 \cdots \stackrel{?}{=} 101\ 011 \cdots$
- Nun muss wieder 3 gewählt werden, also $(1, 3, 3, \ldots)$ etc.
- Konstruktion terminiert nicht!

Post'sches Korrespondenzproblem

- Wir haben lediglich zwei Beispiele betrachtet.
- Im allgemeinen gilt *(ohne Beweis)*
- *Das Post'sche Korrespondenzproblem ist nicht entscheidbar!*
  - D.h. Korrespondenzfunktion ist nicht berechenbar!
  - Gilt schon für ein eingeschränktes Alphabet $\Sigma = \{0, 1\}$!
- Korrespondenzproblem als Beispiel für ein einfaches Problem, das nicht entscheiden werden kann.
  - Es ist einfach, einen Algorithmus anzugeben, der systematisch alle Korrespondenzen testet und somit eine Lösung findet.
  - Dieser Algorithmus terminiert jedoch nicht, falls keine Lösung existiert!

Zusammenfassung

- Church'sche These: Die folgenden Mengen stimmen überein
  - Menge der *intuitiv* berechenbaren Funktionen
  - Menge der durch Maschinen berechenbaren Funktionen
- Es existieren *nichtberechenbare Funktionen*, denn
  - Menge aller berechenbaren Funktionen ist abzählbar
  - Menge aller Funktionen ist überabzählbar
- Es gibt nichtentscheidbare Mengen bzw. Probleme
  - Charakteristische Funktion nicht berechenbar
  - Beispiel: Halteproblem
  - Beispiel: Post'sches Korrespondenzproblem
- *Jede nichttriviale Eigenschaft von Algorithmen ist nichtentscheidbar.* (Rice)
- Literatur zu dieser Vorlesung: [Saake&Sattler]
  - Vertiefende Vorlesung *Grundlagen der Theoretischen Informatik*
  - Weiterführende Literatur z.B. [Schöning]