Berechenbarkeit und Komplexität
6. Vorlesung

Prof. Dr. Dietrich Kuske

FG Automaten und Logik, TU Ilmenau

Sommersemester 2023

Mehrband-Turingmaschine

Weitere Beispielfunktionen (neben Nachfolger und Ω) lassen sich leichter mit „Turing-Maschinen mit mehreren Bändern“ berechnen. Wir werden aber als erstes zeigen, daß sich diese Mehrband-Turingmaschinen durch eine einfache TM simulieren lassen.

Mehrband-Turingmaschine

- Eine Mehrband-Turingmaschine besitzt k ($k \geq 1$) Bänder mit k unabhängigen Köpfen, aber nur eine Steuereinheit.
- Aussehen der Übergangsfunktion:

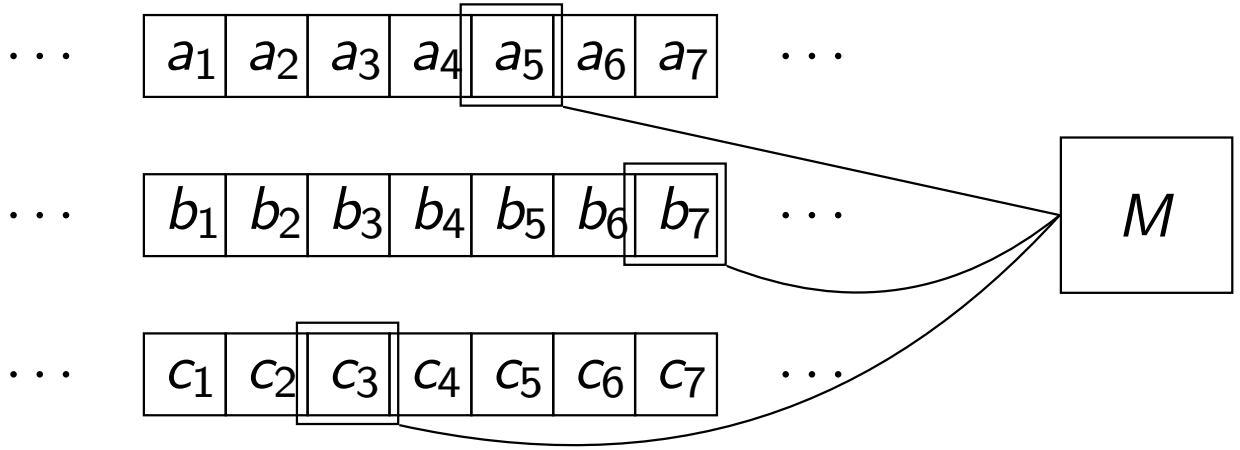
$$\delta: Z \times \Gamma^k \to \left(Z \times \Gamma^k \times \{L, N, R\}^k\right)$$

  (ein Zustand, k Bandsymbole, k Bewegungen)
- Die Ein- und Ausgabe stehen jeweils auf dem ersten Band. Zu Beginn und am Ende (in einer akzeptierenden Haltekonfiguration) sind die restlichen Bänder leer.

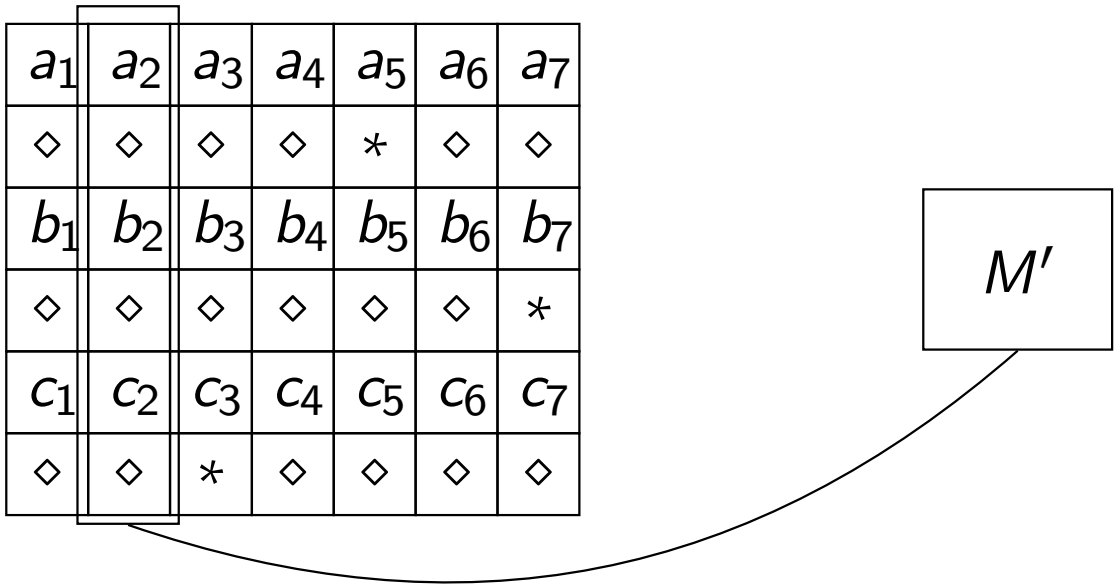
Satz

Zu jeder Mehrband-Turingmaschine M gibt es eine (Einband-)Turingmaschine M', die dieselbe Funktion berechnet.

Beweisidee: Wir beginnen mit der Darstellung einer typischen Konfiguration einer Mehrband-Turingmaschine.



Simulation mittels Einband-Turingmaschine durch Erweiterung des Alphabets: Wir fassen die übereinanderliegenden Bandeinträge zu einem Feld zusammen und markieren die Kopfpositionen auf jedem Band durch $*$.



Neues Bandalphabet:

$$\Gamma' = \Sigma \uplus \{\square\} \uplus \left(\Gamma \times \{*, \diamond\}\right)^k$$

Bedeutung:

- Mit Hilfe von Symbolen von Σ wird die Eingabe dargestellt. Diese wird dann in einem ersten Durchlauf in die „Mehrband-Kodierung“ umgewandelt.
- Ein Alphabetsymbol der Form $(a, *, b, \diamond, c, *, \dots) \in \left(\Gamma \times \{*, \diamond\}\right)^k$ hat die Bedeutung:

  Entsprechende Felder der Bänder 1, 2, 3 usw. sind mit $a, b, c, \dots$ belegt,

  1. und 3. Kopf sind anwesend. ($*$: Kopf anwesend, $\diamond$: Kopf nicht anwesend)

Problem: Eine Einband-Turingmaschine hat nur einen Kopf und der kann nur an einer Stelle stehen $\leadsto$ Simulation eines Schritts der Mehrband-Turingmaschine durch mehrere Schritte.

Simulation:

- Zu Beginn der Simulation eines Schritts steht der Kopf der Einband-Turingmaschine M' links von allen $*$-Markierungen.
- Dann läuft der Kopf nach rechts, überschreitet alle $*$-Markierungen und merkt sich die jeweils gelesenen Zeichen. (Dazu benötigt man viele Zustände.)
- Dann läuft der Kopf wieder zurück nach links und führt alle notwendigen Änderungen aus. □

Beispiel

Die Funktion + ist Turing-berechenbar.

Beweisidee: Wir verwenden eine 3-Band-Turingmaschine, deren 1. Band am Anfang den Inhalt $\mathrm{bin}(x)\#\mathrm{bin}(y)$ enthält.

- 1. Schritt: schreibe $\mathrm{bin}(x)$ auf 2., $\mathrm{bin}(y)$ auf 3. und 0 auf 1. Band, danach stehen die Köpfe am rechten Rand der jeweiligen Inschriften
- 2. Schritt: lese 2. und 3. Band von rechts nach links, führe dabei die Addition aus und schreibe Ergebnis auf 1. Band; lösche dabei 2. und 3. Band. □

Beispiel

Die Funktion $\cdot$ ist Turing-berechenbar.

Beweisidee: Wir verwenden eine 4-Band-Turingmaschine, deren 1. Band am Anfang den Inhalt $\mathrm{bin}(x)\#\mathrm{bin}(y)$ enthält.

- 1. Schritt: schreibe $\mathrm{bin}(x)$ auf 2., $\mathrm{bin}(y)$ auf 3. und 0 auf 1. Band, danach stehen die Köpfe am rechten Rand der jeweiligen Inschriften
- 2. Schritt: in jedem Unterschritt
  1. bewegt sich 2. Kopf eine Stelle nach links (bis er $\square$ liest),
  2. wird (falls 2. Kopf eine 1 liest) die Summe von 1. und 3. Band auf 4. Band geschrieben,
  3. wird Inhalt des 4. Bandes auf 1. Band kopiert und
  4. wird eine 0 an Beschriftung des 3. Bandes angehängt. $\square$

Beispiel

Die Funktionen mod und div sind Turing-berechenbar.

Beweisidee: auch hier wird das schriftliche Rechnen aus der Schulmathematik auf einer festen Anzahl von Bändern durchgeführt. □

Seien $f, g: \mathbb{N} \to \mathbb{N}$ Turing-berechenbare Funktionen. Ist auch

$$\mathrm{subst}(f;g): \mathbb{N} \to \mathbb{N}: n \mapsto \begin{cases} f\big(g(n)\big) & \text{falls } g(n) \text{ \& } f\big(g(n)\big) \text{ definiert sind} \\ \text{undef.} & \text{sonst} \end{cases}$$

Turing-berechenbar?

Konstruktionsversuch einer TM für $\mathrm{subst}(f;g)$**:** die neue TM M simuliert zunächst die TM M_g für g und, nachdem diese geendet hat, die TM M_f für f.

Problem: Was passiert, wenn $g(n)$ nicht definiert ist? Nach Definition erreicht M_g dann keine Haltekonfiguration aus $\square^* E \Sigma^* \square^*$.

hierfür gibt es zwei Möglichkeiten:

1. M_g erreicht keine Haltekonfiguration („terminiert nicht“) (okay)
2. M_g terminiert in einer Haltekonfiguration, die nicht zu $\square^* E \Sigma^* \square^*$ gehört (problematisch)

Im zweiten Fall könnte dann die TM M_f durchaus in eine akzeptierende Haltekonfiguration gelangen, die neue TM M berechnet also einen Wert bei Eingabe von n, obwohl $f\big(g(n)\big)$ nicht definiert ist. Also berechnet M nicht die partielle Funktion $\mathrm{subst}(f; g)$!

Satz

Sei $g: \Sigma^* \dashrightarrow \Sigma^*$ eine Turing-berechenbare partielle Funktion.
Dann wird g von einer TM M berechnet, für die gilt:

$$\forall x \in \Sigma^* \; \forall k \text{ Haltekonfiguration: } z_0 x \square \vdash_M^* k \Longrightarrow k \in \square^* E \Sigma^* \square^* .$$

Beweis: Da g Turing-berechenbar ist, existiert eine TM M, die g berechnet.
Ändere M wie folgt:

- neuer „Fehlerzustand“ $\perp$ mit $\delta(\perp, a) = (\perp, a, R)$ („im Fehlerzustand läuft die Maschine immer weiter nach rechts“)
- Wenn $\delta(z, a) = (z, a, N)$ für ein $z \notin E$, so setze $\delta(z, a) = (\perp, a, R)$.
- $\widetilde{\square}$ neues Bandsymbol. Laß M immer $\widetilde{\square}$ an Stelle von $\square$ schreiben (beim Lesen macht M keinen Unterschied zwischen $\square$ und $\widetilde{\square}$).
  Ergebnis: alle Konfigurationen k mit $z_0 x \square \vdash_M^* k$ gehören zu
  $$\left(Z \cup \Gamma \setminus \{\square\} \cup \{\widetilde{\square}\}\right)^* \{\varepsilon, \square\}$$

- Wenn $\delta(z, a) = (z, a, N)$ und $z \in E$, so starte neue TM, die testet, ob Konfiguration zu $\widetilde{\sqcup}^* E \Sigma^* \widetilde{\sqcup}^* \sqcup^*$ gehört und die in diesem Fall alle $\widetilde{\sqcup}$ durch $\sqcup$ ersetzt (ansonsten wechselt sie in Fehlerzustand $\perp$) $\square$

Der analoge Satz gilt natürlich für Turing-berechenbare partielle Funktionen $\mathbb{N}^k \dashrightarrow \mathbb{N}$.

Satz

Sind $f\colon \mathbb{N}^j \dashrightarrow \mathbb{N}$ und $g_1, g_2, \ldots, g_j\colon \mathbb{N}^k \dashrightarrow \mathbb{N}$ Turing-berechenbar, so auch die partielle Funktion $\mathrm{subst}(f; g_1, g_2, \ldots, g_k)\colon \mathbb{N}^k \dashrightarrow \mathbb{N}$.

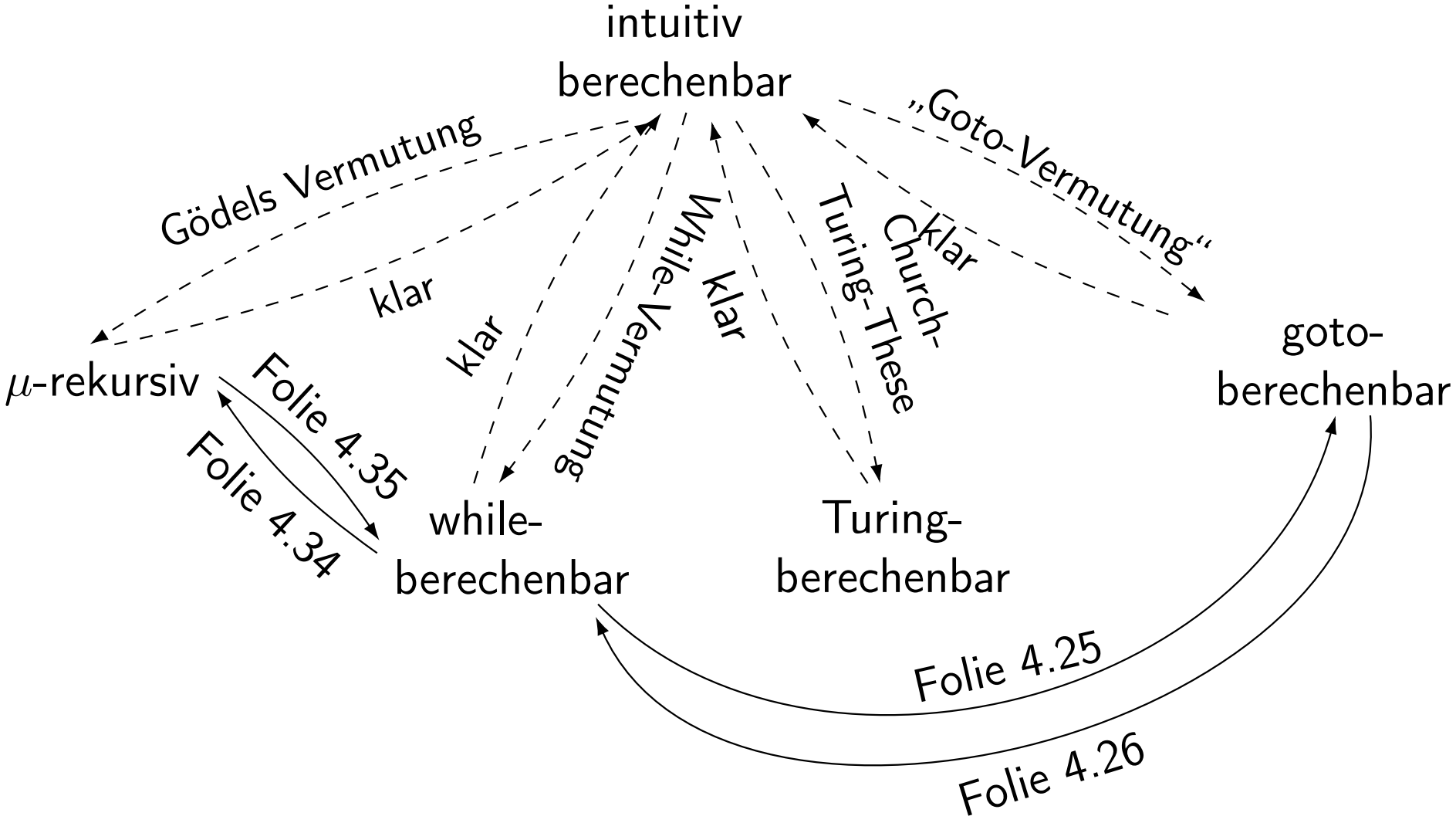
Beweisidee: vgl. Folie 6.10 und verwende Mehrband-TM. □

Beispiel

Ist $p \in \mathbb{N}[x_1, \ldots, x_k]$ Polynom mit natürlichen Koeffizienten, so ist die Funktion $(n_1, \ldots, n_k) \mapsto p(n_1, \ldots, n_k)$ Turing-berechenbar.

Wir wollen jetzt das folgende Bild vervollständigen, indem wir zeigen, daß

- jede while-berechenbare Funktion auch Turing-berechenbar und
- jede Turing-berechenbare Funktion auch goto-berechenbar ist.



Lemma

Jede while-berechenbare partielle Funktion $\mathbb{N}^k \dashrightarrow \mathbb{N}$ ist Turing-berechenbar.

Beweisidee: Wir verwenden eine Mehrband-Turingmaschine, bei der auf jedem Band der Wert einer Variable in Binärdarstellung gespeichert wird: k Variablen $\rightsquigarrow$ k Bänder

- zunächst wird die Eingabe $\mathrm{bin}(n_1)\#\mathrm{bin}(n_2)\#\cdots\mathrm{bin}(n_k)$ vom ersten Band auf die Bänder $1, \ldots, k$ verteilt.
- Simulation des While-Programms $x_i := c$ mit $c \in \{0, 1\}$: überschreibe Band i mit 0 bzw. mit 1
- Simulation des While-Programms $x_i := x_j \pm c$ mit $c \in \{0, 1\}$: kopiere Inhalt des Bandes j auf Band i, addiere bzw. subtrahiere c

- Simulation des While-Programms $P_1; P_2$:

  Seien M_i TM, die P_i simulieren mit Anfangszustand $z_{0,i}$ und Endzuständen E_i.

  Satz auf Folie 6.12: o.E. hält M_1 nur in akzeptierenden Haltekonfigurationen, d.h. höchstens dann, wenn Zustand zu E_1 gehört und alle Köpfe auf dem Beginn des Bandinhalts stehen.

  Simulation von $P_1; P_2$ durch „disjunkte Vereinigung“ dieser TMen mit folgender Änderung:
  gilt $z_e \in E_1$ und $\delta_1(z_e, \overline{a}) = \big(z_e, \overline{a}, (N, N, \ldots, N)\big)$, so ersetze dies durch $\delta(z_e, \overline{a}) = \big(z_{0,2}, \overline{a}, (N, N, \ldots, N)\big)$ (d.h. starte die Maschine M_2).

- Simulation des While-Programms `while` $x_i \neq 0$ `do` P `end`:

  Sei M TM, die P simuliert mit Anfangszustand z_0 und Endzuständen E.

  Satz auf Folie 6.12: o.E. hält M nur in akzeptierenden Haltekonfigurationen, d.h. höchstens dann, wenn Zustand zu E gehört und alle Köpfe auf dem Beginn des Bandinhalts stehen.

  Ändere diese Maschine wie folgt:
  gilt $z_e \in E$ und $\delta_1(z_e, \overline{a}) = \big(z_e, \overline{a}, (N, \ldots, N)\big)$, so ersetze dies durch $\delta(z_e, \overline{a}) = \big(z_{0,2}, \overline{a}, (N, \ldots, N)\big)$, wobei $z_{0,2}$ Anfangszustand einer TM ist, die Band i auf Inhalt 0 testet. Ist dies der Fall, so akzeptiert sie. Ansonsten geht sie in Zustand z_0 über (und startet damit die TM M erneut). □

Im obigen Beweis haben wir die Variablenbelegungen (d.h. k-Tupel natürlicher Zahlen) durch die Binärdarstellung als Tupel von Wörtern über $\{0,1\}$ auffassen können, die Turing-Maschine hat dann diese Wörter bearbeitet.

Bei der Übersetzung von Turing-Maschinen in Goto-Programme gibt es ein grundsätzliches Problem: Wir müssen eine Bandbelegung, d.h. ein Wort über einem Alphabet Γ, in den Variablen des Goto-Programms speichern. Grundidee: o.E. ist Γ eine Menge $\{0,1,\ldots,b-1\}$. Eine Konfiguration, d.h. ein Wort über $\Gamma \cup Z$, kann dann als Darstellung einer Zahl zur Basis b betrachtet werden.

Lemma

Jede Turing-berechenbare partielle Funktion $\mathbb{N}^k \dashrightarrow \mathbb{N}$ ist goto-berechenbar.

Beweis:

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine Turingmaschine, die die partielle Funktion $f: \mathbb{N}^k \dashrightarrow \mathbb{N}$ berechnet.

Sei o.B.d.A. $\Gamma = \{0, \ldots, m-1\}$, $\square = 0$ und $Z = \{m, m+1, \ldots, m+n\}$.

Für $a_1 a_2 \cdots a_p \in \Gamma^*$ bezeichne

$$(a_1 a_2 \cdots a_p)_m = \sum_{i=1}^{p} a_i \cdot m^{i-1}$$

den Wert von $a_1 a_2 \cdots a_p$ in der Darstellung zur Basis m (niederwertigste Ziffer ganz links).

Eine Konfiguration $a_1 \cdots a_p z b_1 \cdots b_q$ wird durch das Tripel

$$\big((a_p \cdots a_1)_m, z, (b_1 \cdots b_q)_m\big) \in \mathbb{N} \times \{m, \ldots, m+n\} \times \mathbb{N}$$

repräsentiert.

Solch ein Tripel wird im folgenden in den drei Variablen x, z, y gespeichert.

Simulation von Turingmaschinenoperationen:

Kopf liest Zeichen: $a := y \bmod m$

Zeichen b aufs Band schreiben: $y := (y \operatorname{div} m) \cdot m + b$

Kopf nach links: $y := y \cdot m + (x \bmod m);\ x := x \operatorname{div} m$

Kopf nach rechts: $x := x \cdot m + (y \bmod m);\ y := y \operatorname{div} m$

Bemerkung: Die verwendeten Funktionen mod, $\cdot$ etc. sind loop- und damit goto-berechenbar.

Initialisierung:

- die Turingmaschine startet mit dem Wort $\text{bin}(n_1)\#\text{bin}(n_2)\#\cdots\#\text{bin}(n_k)$ auf dem Band
- das Goto-Programm startet mit den Werten $n_1, n_2, \ldots, n_k$ in den Variablen $x_1, x_2, \ldots, x_k$
- für die dargestellte Repräsentation muß das Goto-Programm den Wert

$$\big(\text{bin}(n_1)\#\text{bin}(n_2)\#\cdots\#\text{bin}(n_k)\big)_m$$

berechnen und in die Variable y schreiben (Übung für $k = 2$ und $m = 5$). In $\text{bin}(n_i)$ sei hier die Binärziffer 0 durch das Symbol $2 \in \Gamma \setminus \{\square\}$ repräsentiert. Dies ist notwendig, da $\square$ bereits mit der 0 identifiziert wurde.
Weiterhin muß das Goto-Programm $x := 0$ und $z := z_0$ setzen.

Termination

- das Goto-Programm terminiert mit einem Wert in der Variablen y
- dies bedeutet, daß die TM mit dem Wort w mit $(w)_m = y$ terminiert
- für die dargestellte Repräsentation muß das Goto-Programm die Zahl n mit

$$\mathrm{bin}(n) = w$$

berechnen und in die Variable x_1 schreiben (Übung für $m = 5$).

Simulation der Berechnung:

$$
\begin{aligned}
A_i &= \texttt{if } (z \in E) \texttt{ then } j; \\
A_{i+1} &= a := y \bmod m; \qquad \text{(Zeichen einlesen)} \\
A_{i+2} &= \texttt{if } (z = m) \texttt{ and } (a = 0) \texttt{ then } i_{m,0}; \\
A_{i+3} &= \texttt{if } (z = m) \texttt{ and } (a = 1) \texttt{ then } i_{m,1}; \\
&\vdots \\
A_{i_{m,0}} &= \ldots \text{ (Aktion } \delta(m,0) \text{ simulieren)} \\
&\quad \texttt{goto } i; \\
A_{i_{m,1}} &= \ldots \text{ (Aktion } \delta(m,1) \text{ simulieren)} \\
&\quad \texttt{goto } i; \\
&\vdots \\
A_j &= \ldots \text{ (führe Termination aus)}
\end{aligned}
$$

Ein kleiner Ausflug - Zählermaschinen

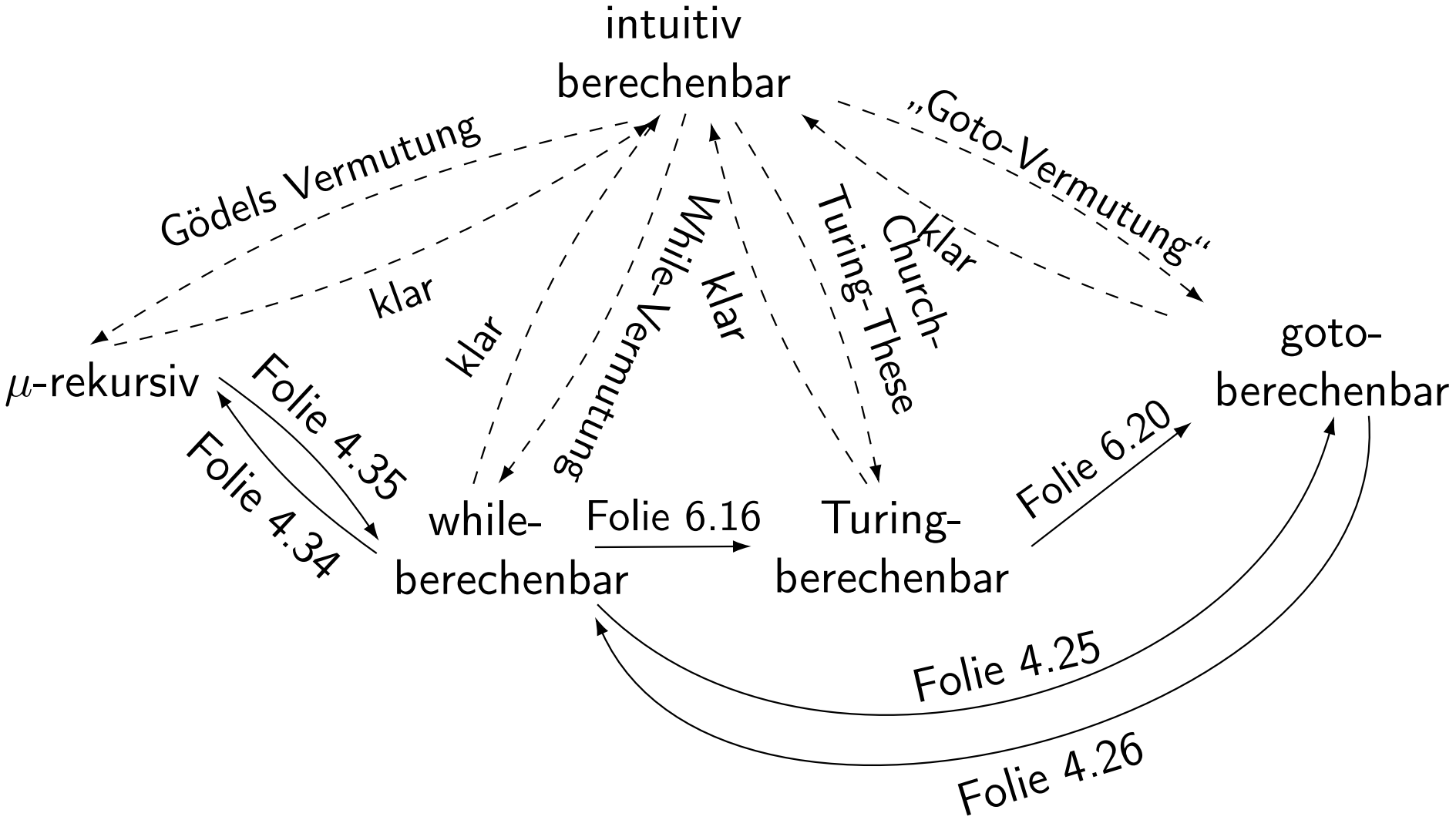
Jede Turing-berechenbare Funktion $\mathbb{N} \dashrightarrow \mathbb{N}$ ist durch ein Goto-Programm berechenbar, das

- nur die Variablen x_1, x_2 und x_3 und
- nur die Wertzuweisungen $x_i := x_i \pm 1$ (für $i \in \{1, 2, 3\}$) verwendet.

Diese eingeschränkten Programme heißen 3-Zähler-Maschinen.

Ist $f : \mathbb{N} \dashrightarrow \mathbb{N}$ Turing-berechenbar, so ist $\mathbb{N} \dashrightarrow \mathbb{N} : 2^n \mapsto 2^{f(n)}$ durch eine 2-Zähler- oder Minsky-Maschine berechenbar (aber $n \mapsto 2^n$ kann nicht durch eine Minsky-Maschine berechnet werden).

Zusammenfassung



(U^+) vier äquivalente Vorschläge, den Begriff „intuitiv berechenbar“ zu formalisieren: while-berechenbar, μ-rekursiv, goto-berechenbar und Turing-berechenbar (daneben gibt es viele weitere wie λ-Kalkül, Random Access Machine, Markov-Algorithmen, ...)

(K^+) viele Funktionen sind while-berechenbar, μ-rekursiv, goto-berechenbar bzw. Turing-berechenbar

(A^+) die vier Klassen erfüllen viele Abschlußeigenschaften

+ Turings Analyse des intuitiven Berechnens führt zum Begriff der Turing-Berechenbarkeit
+ trotz 85-jähriger Suche sind keine Gegenbeispiele für die Church-Turing-These gefunden worden

Damit haben wir wirklich sehr gute Gründe, der While-Vermutung (Folie 4.6), Gödels Vermutung (Folie 4.13), der „Goto-Vermutung“ bzw. der Church-Turing-These (Folie 5.7) zuzustimmen.

Ab jetzt sprechen wir von „berechenbar“, wenn wir „Turing-berechenbar“ meinen.

Es bleibt die Frage nach einer konkreten Funktion, die nicht algorithmisch (d.h. von einer TM) berechenbar ist; bzw. die Frage nach einer konkreten Sprache, deren Wortproblem nicht algorithmisch (d.h. von einer TM) lösbar ist (vgl. Folie 1.11).

Zusammenfassung 6. Vorlesung

in dieser Vorlesung neu

- Äquivalenz der Turing- und While-Berechenbarkeit, d.h. Äquivalenz von While-, Goto- und Gödels Vermutung mit der Church-Turing-These

kommende Vorlesung

- Entscheidung von Problemen mittels Turingmaschinen (oder anderen Algorithmen)
- Probleme, die nicht von Turingmaschinen gelöst werden können (und damit von keinem Algorithmus)