WHILE- und GOTO-Programme

Prof. Dr. David Sabel

LFE Theoretische Informatik

Letzte Änderung der Folien: 28. Juni 2022

WHILE-Programme: Syntax

WHILE-Programme werden durch die CFG (V, Σ, P, Prg) erzeugt:

$$
\begin{array}{rcl}
V & = & \{Prg, Var, Id, Const\} \\
\Sigma & = & \{\textbf{WHILE}, \textbf{LOOP}, \textbf{DO}, \textbf{END}, x, 0, \ldots, 9, ;, :=, +, -\} \\
P & = & \{Prg \;\; \to \;\; \textbf{WHILE}\ Var\ \neq 0\ \textbf{DO}\ Prg\ \textbf{END} \\
 & & \quad\;\; \mid \textbf{LOOP}\ Var\ \textbf{DO}\ Prg\ \textbf{END} \\
 & & \quad\;\; \mid Prg; Prg \\
 & & \quad\;\; \mid Var := Var + Const \\
 & & \quad\;\; \mid Var := Var - Const \\
 & & \quad Var \;\; \to x_{Id} \\
 & & \quad Const \to Id \\
 & & \quad Id \;\; \to 0 \mid 1 \mid \ldots \mid 9 \mid 1Const \mid 2Const \mid \ldots \; 9Const\}
\end{array}
$$

Beachte: WHILE-Programme **erweitern** LOOP-Programme
um das **WHILE**-Konstrukt

WHILE-Programme: Semantik (Berechnungsschritte)

Definition (Berechnungsschritt $\xrightarrow[\text{WHILE}]{}$)

Berechnungsschritt $(\rho, P) \xrightarrow[\text{WHILE}]{} (\rho', P')$, wobei ρ, ρ' Variablenbelegungen und P, P' WHILE-Programme oder ε (leeres Programm)

- $(\rho, x_i := x_j + c) \xrightarrow[\text{WHILE}]{} (\rho', \varepsilon)$ wobei $\rho' = \rho\{x_i \mapsto n\}$ und $n = \rho(x_j) + c$
- $(\rho, x_i := x_j - c) \xrightarrow[\text{WHILE}]{} (\rho', \varepsilon)$ wobei $\rho' = \rho\{x_i \mapsto n\}$ und $n = \max(0, \rho(x_j) - c)$
- $(\rho, P_1; P_2) \xrightarrow[\text{WHILE}]{} (\rho', P_1'; P_2)$ wenn $(\rho, P_1) \xrightarrow[\text{WHILE}]{} (\rho', P_1')$ und $P_1' \neq \varepsilon$
- $(\rho, P_1; P_2) \xrightarrow[\text{WHILE}]{} (\rho', P_2)$ wenn $(\rho, P_1) \xrightarrow[\text{WHILE}]{} (\rho', \varepsilon)$
- $(\rho, \textbf{LOOP } x_i \textbf{ DO } P \textbf{ END}) \xrightarrow[\text{WHILE}]{} (\rho, \underbrace{P; \ldots; P}_{\rho(x_i)\text{-mal}})$
- $(\rho, \textbf{WHILE } x_i \neq 0 \textbf{ DO } P \textbf{ END}) \xrightarrow[\text{WHILE}]{} (\rho, \varepsilon)$ wenn $\rho(x_i) = 0$
- $(\rho, \textbf{WHILE } x_i \neq 0 \textbf{ DO } P \textbf{ END}) \xrightarrow[\text{WHILE}]{} (\rho, P; \textbf{WHILE } x_i \neq 0 \textbf{ DO } P \textbf{ END})$ wenn $\rho(x_i) \neq 0$

Mit $\xrightarrow[\text{WHILE}]{}^i$ bezeichnen wir i-Schritte und mit $\xrightarrow[\text{WHILE}]{}^*$ 0 oder beliebig viele Schritte

WHILE-Berechenbarkeit

Definition (WHILE-berechenbare Funktion)

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt WHILE-berechenbar, wenn es ein WHILE-Programm P gibt, sodass

- für alle $n_1, \ldots, n_k \in \mathbb{N}$, sodass $f(n_1, \ldots, n_k)$ definiert ist:
  Für Variablenbelegung $\rho = \{x_1 \mapsto n_1, \ldots, x_k \mapsto n_k\}$ gilt $(\rho, P) \xrightarrow[\text{WHILE}]{}{}^* \ (\rho', \varepsilon)$
  und $\rho'(x_0) = f(n_1, \ldots, n_k)$.
- Falls $f(n_1, \ldots, n_k)$ nicht definiert ist, so stoppt das Programm P nicht, d. h. für alle $i \in \mathbb{N}$ gibt es ein ρ',
  sodass $(\rho, P) \xrightarrow[\text{WHILE}]{}{}^i (\rho', P')$

LOOP-berechenbar $\Longrightarrow$ WHILE-berechenbar

Satz

Jede LOOP-berechenbare Funktion ist auch WHILE-berechenbar.

Beweis:

Offensichtlich, da jedes LOOP-Programm auch ein WHILE-Programm ist, und die Semantik identisch dafür ist

LOOP durch **WHILE**

Das **LOOP**-Konstrukt ist überflüssig in WHILE:

Ersetze	**LOOP** x_i **DO** P **END**
durch	$x_m := x_i;$ **WHILE** $x_m \neq 0$ **DO** $x_m := x_m - 1; P$ **END** wobei x_m nirgends sonst vorkommt

WHILE-berechenbar $\Longrightarrow$ Turingberechenbar

Theorem 10.2.4

Jede WHILE-berechenbare Funktion ist auch Turingberechenbar.

Beweis:

- Wir haben bereits gezeigt, dass Zuweisung, Sequenz und **WHILE** durch Turingmaschinen darstellbar sind.
- Dabei liegen die Variablen x_i jeweils auf Band i einer Mehrband-TM
- **LOOP** wird vorher in **LOOP**-freies WHILE-Programm transformiert.

GOTO-Programme: Syntax

GOTO-Programme werden durch die CFG (V, Σ, P, Prg) erzeugt:

$$
\begin{aligned}
V &= \{Prg, Var, Id, Const\} \\
\Sigma &= \{\textbf{GOTO}, \textbf{HALT}, \textbf{IF}, x, 0, \ldots, 9, ;, :=, +, -\} \\
P &= \{Prg \rightarrow M_{Id} : Var := Var + Const \\
&\quad \mid M_{Id} : Var := Var - Const \\
&\quad \mid M_{Id} : \textbf{GOTO}\ M_{Id} \\
&\quad \mid M_{Id} : \textbf{IF}\ x_i = 0\ \textbf{GOTO}\ M_{Id} \\
&\quad \mid M_{Id} : \textbf{HALT} \\
&\quad \mid Prg; Prg \\
&\quad Var \rightarrow x_{Id} \\
&\quad Const \rightarrow Id \\
&\quad Id \rightarrow 0 \mid 1 \mid \ldots \mid 9 \mid 1Const \mid 2Const \mid \ldots\ 9Const\}
\end{aligned}
$$

- Befehle sind mit M_i : markiert. Wenn unwichtig, lassen wir sie weg.
- Nebenbedingung: Alle Markierungen verschieden.
- Normalisiertes GOTO-Programm: $M_1 : A_1; \ldots; M_n : A_n$
  durch konsistentes Umbenennen der Markierungen herstellbar

GOTO-Programme: Semantik (Berechnungsschritte)

Definition (Berechnungsschritt $\xrightarrow[\text{GOTO}]{P_0}$)

Sei P_0 ein GOTO-Programm, ρ eine Variablenbelegung und P ein GOTO-Programm. Dann ist ein Berechnungsschritt $\xrightarrow[\text{GOTO}]{P_0}$ durch die folgenden Fälle definiert:

- $(\rho, M_i : x_j := x_k + c; P) \xrightarrow[\text{GOTO}]{P_0} (\rho[x_j \mapsto n], P)$, wobei $n = \rho(x_k) + c$
- $(\rho, M_i : x_j := x_k + c) \xrightarrow[\text{GOTO}]{P_0} (\rho[x_j \mapsto n], \varepsilon)$, wobei $n = \rho(x_k) + c$
- $(\rho, M_i : x_j := x_k - c; P) \xrightarrow[\text{GOTO}]{P_0} (\rho[x_j \mapsto n], P)$, wobei $n = \max(0, \rho(x_k) - c)$
- $(\rho, M_i : x_j := x_k - c) \xrightarrow[\text{GOTO}]{P_0} (\rho[x_j \mapsto n], \varepsilon)$, wobei $n = \max(0, \rho(x_k) - c)$
- $(\rho, M_i : \textbf{GOTO } M_k) \xrightarrow[\text{GOTO}]{M_1:A_1;\ldots;M_n:A_n} (\rho, M_k : A_k; \ldots; M_n : A_n)$ wenn $1 \leq i \leq n$
- $(\rho, M_i : \textbf{GOTO } M_k; P) \xrightarrow[\text{GOTO}]{M_1:A_1;\ldots;M_n:A_n} (\rho, M_k : A_k; \ldots; M_n : A_n)$ wenn $1 \leq i \leq n$
- ...

GOTO-Programme: Semantik (Berechnungsschritte, Forts.)

- ...
- $(\rho, M_i : \textbf{IF } x_r = 0 \textbf{ THEN GOTO } M_k; P) \xrightarrow[\text{GOTO}]{M_1:A_1;\ldots;M_n:A_n} (\rho, M_k : A_k; \ldots; M_n : A_n)$ wenn $1 \leq i \leq n$ und $\rho(x_r) = 0$
- $(\rho, M_i : \textbf{IF } x_r = 0 \textbf{ THEN GOTO } M_k) \xrightarrow[\text{GOTO}]{M_1:A_1;\ldots;M_n:A_n} (\rho, M_k : A_k; \ldots; M_n : A_n)$ wenn $1 \leq i \leq n$ und $\rho(x_r) = 0$
- $(\rho, M_i : \textbf{IF } x_r = 0 \textbf{ THEN GOTO } M_k; P) \xrightarrow[\text{GOTO}]{P_0} (\rho, P)$ wenn $\rho(x_r) \neq 0$
- $(\rho, M_i : \textbf{IF } x_r = 0 \textbf{ THEN GOTO } M_k) \xrightarrow[\text{GOTO}]{P_0} (\rho, \varepsilon)$ wenn $\rho(x_r) \neq 0$
- $(\rho, M_i : \textbf{HALT}; P) \xrightarrow[\text{GOTO}]{P_0} (\rho, M_i : \textbf{HALT})$
- $(\rho, \varepsilon) \xrightarrow[\text{GOTO}]{P_0} (\rho, \varepsilon)$

Wir schreiben $\xrightarrow[\text{GOTO}]{P_0}^{*}$ für 0 oder beliebig viele Berechnungsschritte und $\xrightarrow[\text{GOTO}]{P_0}^{i}$ für genau i Berechnungsschritte (mit dem Programm P_0).

GOTO-Berechenbarkeit

Definition (GOTO-berechenbare Funktion)

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt GOTO-berechenbar, wenn es ein normalisiertes GOTO-Programm P gibt, sodass

- für alle $n_1, \ldots, n_k \in \mathbb{N}$, sodass $f(n_1, \ldots, n_k)$ definiert ist:
  Für Variablenbelegung $\rho = \{x_1 \mapsto n_1, \ldots, x_k \mapsto n_k\}$ gilt:
  $(\rho, P) \xrightarrow[\text{GOTO}]{P}{}^{*} (\rho', \textbf{HALT})$ und $\rho'(x_0) = f(n_1, \ldots, n_k)$.
- Falls $f(n_1, \ldots, n_k)$ nicht definiert ist, so stoppt das Programm P nicht, d.h. für alle $i \in \mathbb{N}$ gibt es ρ' und P' sodass: $(\rho, P) \xrightarrow[\text{GOTO}]{P}{}^{i} (\rho', P')$

WHILE-berechenbar $\Longrightarrow$ GOTO-berechenbar

Jedes WHILE-Programm ist durch ein GOTO-Programm simulierbar:

- Klar für $x_j := x_k \pm c$ und $P_1; P_2$
- **LOOP**-Schleife durch **WHILE**-Schleifen ersetzen
- **WHILE** $x_i \neq 0$ **DO** P **END** kann durch

$$\begin{array}{ll} M_1: & \textbf{IF } x_i = 0 \textbf{ THEN GOTO } M_j \\ \ldots & P'; \\ M_{j-1}: & \textbf{GOTO } M_1; \\ M_j: & \ldots \end{array}$$

  simuliert werden, wobei P' das GOTO-Programm zu P ist.
- **HALT** am Ende einfügen

WHILE-berechenbar $\Longrightarrow$ GOTO-berechenbar (2)

Sei P_W WHILE-Programm und P_G das beschriebene GOTO-Programm.

Dann gilt:

- Falls $(\rho, P_W) \xrightarrow[\text{WHILE}]{}^{*} (\rho', \varepsilon)$, dann $(\rho, P_G) \xrightarrow[\text{GOTO}]{P_G}^{*} (\rho'', \textbf{HALT})$ mit $\rho'(x_0) = \rho''(x_0)$.

- Falls $\forall i \in \mathbb{N} : (\rho, P_W) \xrightarrow[\text{WHILE}]{}^{i} (\rho_i, P_{W,i})$ dann auch

  $\forall i \in \mathbb{N} : (\rho, P_G) \xrightarrow[\text{GOTO}]{P_G}^{i} (\rho'_i, P_{G,i})$.

Beweisbar durch Induktion über die gegebenen Sequenzen.

Damit folgt:

Jede WHILE-berechenbare Funktion ist auch GOTO-berechenbar

GOTO-berechenbar $\Longrightarrow$ WHILE-berechenbar

Sei $M_1 : A_1; \ldots; M_n : A_n$ ein normalisiertes GOTO-Programm, sodass x_M nicht darin vorkommt. Passendes WHILE-Programm:

$x_M := 1;$
WHILE $x_M \neq 0$ **DO**
  IF $x_M = 1$ **THEN** P_1 **END**;
  $\ldots$
  IF $x_M = n$ **THEN** P_n **END**;
END

wobei P_i das folgende Programm ist:

- $x_j := x_k \pm c; x_M := x_M + 1$, falls $A_i = x_j := x_k \pm c$
- $x_M := j$, falls $A_i =$ **GOTO** M_j
- **IF** $x_k = 0$ **THEN** $x_M := j$ **ELSE** $x_M := x_M + 1$ **END**, falls $A_i =$ **IF** $x_k = 0$ **THEN GOTO** M_j
- $x_M := 0$, falls $A_i =$ **HALT**

GOTO-berechenbar $\Longrightarrow$ WHILE-berechenbar (2)

Es gilt:

- $(\rho, M_1 : A_1; \ldots; M_n : A_n) \xrightarrow[\text{GOTO}]{M_1:A_1;\ldots;M_n:A_n}{}^{*} (\rho', \textbf{HALT})$
  g.d.w. $(\rho, P) \xrightarrow[\text{WHILE}]{}{}^{*} (\rho'', \varepsilon)$ wobei $\rho'(x_0) = \rho''(x_0)$
- Falls $\forall i \in \mathbb{N} : (\rho, M_1 : A_1; \ldots; M_n : A_n) \xrightarrow[\text{GOTO}]{M_1:A_1;\ldots;M_n:A_n}{}^{i} (\rho_i, P_i)$, dann auch
  $\forall i \in \mathbb{N} : (\rho, P) \xrightarrow[\text{WHILE}]{}{}^{i} (\rho_i', P_i')$

Damit folgt:

Jede GOTO-berechenbare Funktion ist auch WHILE-berechenbar.

GOTO-berechenbar $\iff$ WHILE-berechenbar

Theorem 10.4.1

WHILE- und GOTO-Berechenbarkeit sind äquivalente Begriffe. D.h. jede WHILE-berechenbare Funktion ist auch GOTO-berechenbar und umgekehrt ist jede GOTO-berechenbare Funktion auch WHILE-berechenbar.

Die Übersetzungen zeigen auch:

Satz

WHILE-Programme benötigen nur eine WHILE-Schleife.

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (1)

Ziel:

- GOTO-Programm simuliert die Turingmaschine

Idee:

- Stelle TM-Konfiguration durch Zahlen dar
- Zahlen werden in Programmvariablen abgelegt.
- Es werden am Ende nur 3 Programmvariablen benutzt!

Darstellung von Worten als natürliche Zahlen

- Sei Bandalphabet $\Gamma = \{a_1, \ldots, a_m\}$
- Zeichen a_i wird mit Index i identifiziert
- D.h. $a_{i_1} \cdots a_{i_n} \in \Gamma^*$ äquivalent zur Folge $(i_1 \ldots i_n)$
- Sei $b > m = |\Gamma|$
- $a_{i_1} \ldots a_{i_n}$: Zahl im Stellenwertsystem zur Basis b
  (beachte: 0-wertige Ziffer nicht in $a_1, \ldots, a_m$)

$$\text{Dezimalwert von } a_{i_1} \cdots a_{i_n}\text{: } (i_1 \cdots i_n)_b = \sum_{j=1}^{n} i_j \cdot b^{n-j}$$

Umkehrung: Berechne $(i_1 \cdots i_n)$ aus einer natürlichen Zahl:

- Teile wiederholt mit Rest durch Basis b
- Liste der Reste: $r_0, \ldots, r_m$.
- Ergibt $(r_m \cdots r_0)$ zur Basis b und repräsentiert daher das Wort $a_{r_m} \cdots a_{r_0}$.

Beispiel

Sei $\Gamma = \{\square, 0, 1\}$, sodass $a_1 = \square, a_2 = 0, a_3 = 1$.

Betrachte $\square 110 \square$, d.h. $a_1 a_3 a_3 a_2 a_1$, sei $b = 4$

Identifiziere $\square 110 \square$ mit (13321) und $(13321)_b = (13321)_4 =$
$1 * 4^4 + 3 * 4^3 + 3 * 4^2 + 2 * 4^1 + 1 * 4^0 = 256 + 3 * 64 + 3 * 16 + 2 * 4 + 1 * 1 = 505$

Umgekehrt ergibt

$$\begin{array}{rcll} 505/4 & = & 126 & \text{Rest } 1 \\ 126/4 & = & 31 & \text{Rest } 2 \\ 31/4 & = & 7 & \text{Rest } 3 \\ 7/4 & = & 1 & \text{Rest } 3 \\ 1/4 & = & 0 & \text{Rest } 1 \end{array}$$

und daher ergibt dies die Zahl (13321) zur Basis b, welche das Wort $a_1 a_3 a_3 a_2 a_1 = \square 110 \square$ repräsentiert.

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (2)

TM-Konfiguration im GOTO-Programm:

Konfiguration der TM mit $\Gamma = \{a_1, \ldots, a_m\}$:

$$a_{i_1} \cdots a_{i_n} z_k a_{j_1} \cdots a_{j_m}$$

Im GOTO-Programm durch 3 Variablen x_p, x_z, x_s dargestellt:

$$\begin{aligned} x_p &= (i_1 \cdots i_n)_b \\ x_z &= k \\ x_s &= (j_m \cdots j_1)_b \end{aligned}$$

Beachte: x_s verwendet umgekehrte Reihenfolge der Ziffern

GOTO-Programme für Transitionen

Für $\delta(z_k, a_{j_1}) = (z_l, a_{j'}, Q)$ mit $Q \in \{N, L, R\}$ sei $P(k, j_1)$ das zugehörige GOTO-Programm.

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (3)

Fall $\delta(z_k, a_{j_1}) = (z_l, a_{j'}, L)$

Transition ist

$$a_{i_1} \cdots a_{i_n} z_k a_{j_1} \cdots a_{j_m} \vdash a_{i_1} \cdots a_{i_{n-1}} z_l a_{i_n} a_{j'} a_{j_2} \cdots a_{j_m},$$

falls $n > 0$. Bezüglich der Zahlendarstellung gilt daher:

- x_z muss auf den Wert l aktualisiert werden
- x_p muss von $(i_1 \cdots i_n)_b$ zu $(i_1 \cdots i_{n-1})_b$ aktualisiert werden
- x_s muss von $(j_m \cdots j_1)_b$ zu $(j_m \cdots j_2 j' i_n)_b$ aktualisiert werden

Das Programm $P(k, j_1)$ dazu

$x_z := l$; // neuer Zustand z_l

$x_s := x_s \text{ div } b$; // Abschneiden der letzten Stelle $(j_m \cdots j_1) \to (j_m \cdots j_2)$

$x_s := b * x_s + j'$; // j' als letzte Stelle hinzufügen $(j_m \cdots j_2) \to (j_m \cdots j_2 j')$

$x_s := b * x_s + (x_p \bmod b)$; // $i_n = x_p \bmod b$ am Endes hinzufügen $(j_m \cdots j_2 j') \to (j_m \cdots j_2 j' i_n)$

$x_p := x_p \text{ div } b$// Abschneiden der letzten Stelle $(i_1 \cdots i_n) \to (i_1 \cdots i_{n-1})$

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (4)

Fall $\delta(z_k, a_{j_1}) = (z_l, a_{j'}, L)$

Für den Grenzfall

$$z_k a_{j_1} \cdots a_{j_m} \vdash z_l \square a_{j_1} \cdots a_{j_m}$$

setze zuerst $x_p := r$; wobei $\square = a_r$ und führe dann das Programm für den allgemeinen Fall aus.

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (5)

Fall $\delta(z_k, a_{j_1}) = (z_l, a_{j'}, N)$

Transition ist

$$a_{i_1} \cdots a_{i_n} z_k a_{j_1} \cdots a_{j_m} \vdash a_{i_1} \cdots a_{i_n} z_l a_{j'} a_{j_2} \cdots a_{j_m}$$

- x_z muss auf den Wert l aktualisiert werden
- x_p bleibt unverändert.
- x_s muss von $(j_m \cdots j_1)_b$ zu $(j_m \cdots j_2 j')_b$ aktualisiert werden

Das Programm $P(k, j_1)$ sei dann:

$$\begin{aligned} x_z &:= l; \\ x_s &:= x_s \text{ div } b; \\ x_s &:= b * x_s + j' \end{aligned}$$

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (6)

Fall $\delta(z_k, a_{j_1}) = (z_l, a_{j'}, R)$

Transition ist

$$a_{i_1} \cdots a_{i_n} z_k a_{j_1} \cdots a_{j_m} \vdash a_{i_1} \cdots a_{i_n} a_{j'} z_l a_{j_2} \cdots a_{j_m},$$

falls $m > 1$. Bezüglich der Zahlendarstellung gilt daher:

- x_z muss auf den Wert l aktualisiert werden
- x_p muss von $(i_1 \cdots i_n)_b$ zu $(i_1 \cdots i_n j')_b$ aktualisiert werden
- x_s muss von $(j_m \cdots j_1)_b$ zu $(j_m \cdots j_2)_b$ aktualisiert werden

Das Programm $P(k, j_1)$ sei dann:

$$\begin{aligned} &x_z := l; \\ &x_p := b * x_p + j'; \\ &x_s := x_s \text{ div } b \end{aligned}$$

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (7)

Fall $\delta(z_k, a_{j_1}) = (z_l, a_{j'}, R)$

Für den Fall $m = 1$ gilt

$$a_{i_1} \cdots a_{i_n} z_k a_{j_1} \vdash a_{i_1} \cdots a_{i_n} a'_j z_l \square$$

In diesem Fall sei $P(k, j_1)$ das Programm

$$\begin{aligned} x_z &:= l; \\ x_p &:= b * x_p + j'; \\ x_s &:= r \end{aligned}$$

wobei $\square = a_r$.

Beachte, dass wir die Sonderfälle abfragen können, durch testen, ob x_p bzw. x_s gleich zu 0 ist.

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (8)

Simulation der TM durch das folgende GOTO-Programm:
(Eingaben in $x_1, \ldots, x_k$, Ausgabe in x_0)

$$M_1 : P_1;$$
$$M_2 : P_2;$$
$$M_3 : P_3$$

- P_1 erzeugt für die Eingabe in $x_1, \ldots, x_k$ die Binärdarstellung und dann die Darstellung der TM-Konfiguration in x_p, x_z, x_s
- P_3 verwendet die Darstellung der Endkonfiguration, um die Ausgabe in der Ausgabevariablen x_0 zu erzeugen.
- P_2: Nächste Folie

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (9)

Programmteil P_2 hat die Form:

M_2 : $x_a := x_s \bmod b$;
IF $(x_z = 1)$ **and** $(x_a = 1)$ **THEN GOTO** $M_{1,1}$;
IF $(x_z = 1)$ **and** $(x_a = 2)$ **THEN GOTO** $M_{1,2}$;
...
IF $(x_z = q)$ **and** $(x_a = m)$ **THEN GOTO** $M_{q,m}$

$M_{1,1}$: $P'_{1,1}$;
GOTO M_2;

$M_{1,2}$: $P'_{1,2}$;
GOTO M_2;

...

$M_{q,m}$: $P'_{q,m}$
GOTO M_2

Programm $P'_{i,j}$ ist **GOTO** M_3, wenn z_i Endzustand ist und $P(i,j)$ sonst.

Turingberechenbar $\Longrightarrow$ GOTO-berechenbar (10)

Insgesamt zeigt das GOTO-Programm:

Satz 10.5.2

GOTO-Programme können Turingmaschinen simulieren. Jede Turingberechenbare Funktion ist auch GOTO-berechenbar.

Fazit

Theorem

Turingberechenbarkeit, WHILE-Berechenbarkeit und GOTO-Berechenbarkeit sind äquivalente Begriffe.