Theoretische Informatik II

Einheit 7.2

Universelle Maschinen

1. Standardnumerierung berechenbarer Funktionen
2. Universelle Funktion
3. Grundeigenschaften berechenbarer Funktionen

Noch Offene Fragen

- **Gibt es unentscheidbare Mengen?**
  - Unentscheidbar aber aufzählbar?
  - Nicht aufzählbar'?

Noch Offene Fragen

- **Gibt es unentscheidbare Mengen?**
  - Unentscheidbar aber aufzählbar?
  - Nicht aufzählbar'?
- **Gibt es unberechenbare Funktionen?**

Noch Offene Fragen

- **Gibt es unentscheidbare Mengen?**
  - Unentscheidbar aber aufzählbar?
  - Nicht aufzählbar'?
- **Gibt es unberechenbare Funktionen?**
- **Wie beweist man Unlösbarkeit?**
  - Kardinalitätsargument: es gibt mehr Funktionen als Programme
  - Konkretes Gegenbeispiel konstruieren

Noch Offene Fragen

- **Gibt es unentscheidbare Mengen?**
  - Unentscheidbar aber aufzählbar?
  - Nicht aufzählbar'?
- **Gibt es unberechenbare Funktionen?**
- **Wie beweist man Unlösbarkeit?**
  - Kardinalitätsargument: es gibt mehr Funktionen als Programme
  - Konkretes Gegenbeispiel konstruieren
- **Was benötigt man für diese Argumente?**
  - Präzisierung der Grundannahmen zur Berechenbarkeit
  - Nachweis, daß diese Grundannahmen erfüllt sind

Grundannahmen über Berechnungen

- **Programme und Daten sind als Zahlen codierbar**
  – Programme und Daten werden als Worte dargestellt
  – Worte, die Programme darstellen, können durchnumeriert werden
  – φ_i: Berechnete Funktion des Programms i $(\varphi_i : \mathbb{N} \to \mathbb{N})$
  – Φ_i: Rechenzeitfunktion zum Programm i $(\mathrm{domain}(\Phi_i) = \mathrm{domain}(\varphi_i))$

Grundannahmen über Berechnungen

- **Programme und Daten sind als Zahlen codierbar**
  - Programme und Daten werden als Worte dargestellt
  - Worte, die Programme darstellen, können durchnumeriert werden
  - φ_i: Berechnete Funktion des Programms i $(\varphi_i : \mathbb{N} \to \mathbb{N})$
  - Φ_i: Rechenzeitfunktion zum Programm i $(\mathrm{domain}(\Phi_i) = \mathrm{domain}(\varphi_i))$
- **Computer sind universelle Maschinen**
  - Bei Eingabe beliebiger Programme und Daten berechnen sie das Ergebnis
  - Die Funktion $u : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ mit $u(i, n) = \varphi_i(n)$ ist berechenbar

Grundannahmen über Berechnungen

- **Programme und Daten sind als Zahlen codierbar**
  - Programme und Daten werden als Worte dargestellt
  - Worte, die Programme darstellen, können durchnumeriert werden
  - φ_i: Berechnete Funktion des Programms i $(\varphi_i : \mathbb{N} \to \mathbb{N})$
  - Φ_i: Rechenzeitfunktion zum Programm i $(\mathrm{domain}(\Phi_i) = \mathrm{domain}(\varphi_i))$
- **Computer sind universelle Maschinen**
  - Bei Eingabe beliebiger Programme und Daten berechnen sie das Ergebnis
  - Die Funktion $u : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ mit $u(i, n) = \varphi_i(n)$ ist berechenbar
- **Man kann Programme effektiv zusammensetzen**
  - Die Nummer des entstehenden Programms kann berechnet werden
  - Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)} = \varphi_i \circ \varphi_j$

Grundannahmen über Berechnungen

- **Programme und Daten sind als Zahlen codierbar**
  - Programme und Daten werden als Worte dargestellt
  - Worte, die Programme darstellen, können durchnumeriert werden
  - φ_i: Berechnete Funktion des Programms i $(\varphi_i : \mathbb{N} \rightarrow \mathbb{N})$
  - Φ_i: Rechenzeitfunktion zum Programm i $(\mathrm{domain}(\Phi_i) = \mathrm{domain}(\varphi_i))$
- **Computer sind universelle Maschinen**
  - Bei Eingabe beliebiger Programme und Daten berechnen sie das Ergebnis
  - Die Funktion $u : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ mit $u(i, n) = \varphi_i(n)$ ist berechenbar
- **Man kann Programme effektiv zusammensetzen**
  - Die Nummer des entstehenden Programms kann berechnet werden
  - Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)} = \varphi_i \circ \varphi_j$
- **Rechenzeit ist entscheidbar**
  - Man kann für beliebige $i, n, t \in \mathbb{N}$ testen ob $\Phi_i(n) = t$ ist oder nicht

Codierung von Turingmaschinen

- **Codierung der Alphabete in einem Alphabet $\hat{\Gamma}$**
  - Wähle $\hat{\boldsymbol{\Gamma}} \equiv \{ \#, s_0 ... s_n, \gamma_0 ... \gamma_k, r, l, h \}$

    wobei $S = \{s_0,...,s_n\}$, $\Gamma = \{\gamma_0,...,\gamma_k\}$, $X = \{\gamma_{i_0},...,\gamma_{i_m}\} \subseteq \Gamma$, $b = \gamma_k \in \Gamma \backslash X$

Codierung von Turingmaschinen

- **Codierung der Alphabete in einem Alphabet $\hat{\Gamma}$**
  – Wähle $\hat{\mathbf{\Gamma}} \equiv \{ \#, s_0 ... s_n, \gamma_0 ... \gamma_k, r, l, h \}$
  wobei $S = \{s_0,...,s_n\}$, $\Gamma = \{\gamma_0,...,\gamma_k\}$, $X = \{\gamma_{i_0},...,\gamma_{i_m}\} \subseteq \Gamma$, $b = \gamma_k \in \Gamma \setminus X$

- **Codierung der Zustandsüberführungsfunktion δ**
  – Beschreibe $\delta(s,a) = (s',a',P)$ durch $\mathrm{code}(\delta(s,a)) \equiv s\, a\, s'\, a'\, P$
  – Beschreibe δ durch das Wort $\mathrm{code}(\delta(s_0,\gamma_0)) \ldots \mathrm{code}(\delta(s_n,\gamma_k))$

Codierung von Turingmaschinen

- **Codierung der Alphabete in einem Alphabet $\hat{\Gamma}$**
  - Wähle $\hat{\mathbf{\Gamma}} \equiv \{\, \#, s_0 ... s_n, \gamma_0 ... \gamma_k, r, l, h \,\}$

    wobei $S = \{s_0, ..., s_n\}$, $\Gamma = \{\gamma_0, ..., \gamma_k\}$, $X = \{\gamma_{i_0}, ..., \gamma_{i_m}\} \subseteq \Gamma$, $b = \gamma_k \in \Gamma \backslash X$

- **Codierung der Zustandsüberführungsfunktion δ**
  - Beschreibe $\delta(s,a) = (s', a', P)$ durch $\mathrm{code}(\delta(s,a)) \equiv s\,a\,s'\,a'\,P$
  - Beschreibe δ durch das Wort $\mathrm{code}(\delta(s_0,\gamma_0)) \ldots \mathrm{code}(\delta(s_n,\gamma_k))$

- **Codierung der Turingmaschine τ**
  - Beschreibe $\tau = (S, X, \Gamma, \delta, s_0, b)$ durch das Wort

    $$\hat{w}_\tau \equiv \gamma_{i_0} ... \gamma_{i_m} \# \mathrm{code}(\delta(s_0,\gamma_0)) \ldots \mathrm{code}(\delta(s_n,\gamma_k))$$

Codierung von Turingmaschinen

- **Codierung der Alphabete in einem Alphabet $\hat{\Gamma}$**
  - Wähle $\hat{\mathbf{\Gamma}} \equiv \{ \#, s_0 ... s_n, \gamma_0 ... \gamma_k, r, l, h \}$
    wobei $S = \{s_0,...,s_n\}$, $\Gamma = \{\gamma_0,...,\gamma_k\}$, $X = \{\gamma_{i_0},...,\gamma_{i_m}\} \subseteq \Gamma$, $b = \gamma_k \in \Gamma \setminus X$

- **Codierung der Zustandsüberführungsfunktion δ**
  - Beschreibe $\delta(s,a) = (s',a',P)$ durch $\mathrm{code}(\delta(s,a)) \equiv s\,a\,s'\,a'\,P$
  - Beschreibe δ durch das Wort $\mathrm{code}(\delta(s_0,\gamma_0)) \ldots \mathrm{code}(\delta(s_n,\gamma_k))$

- **Codierung der Turingmaschine τ**
  - Beschreibe $\tau = (S, X, \Gamma, \delta, s_0, b)$ durch das Wort
    $\hat{w}_\tau \equiv \gamma_{i_0} ... \gamma_{i_m} \# \mathrm{code}(\delta(s_0,\gamma_0)) \ldots \mathrm{code}(\delta(s_n,\gamma_k))$
  - $\boldsymbol{w_\tau}$ sei die Codierung von $\hat{w}_\tau$ im festen Alphabet $\{0,1\}$ $(\hat{\gamma}_j \in \hat{\Gamma} \mathrel{\hat{=}} \underbrace{0...0}_{j-mal}1)$

Codierung von Turingmaschinen

- **Codierung der Alphabete in einem Alphabet $\hat{\Gamma}$**
  - Wähle $\hat{\boldsymbol{\Gamma}} \equiv \{\ \#,\ s_0...s_n,\ \gamma_0...\gamma_k,\ r,\ l,\ h\ \}$
    wobei $S=\{s_0,...,s_n\}$, $\Gamma=\{\gamma_0,...,\gamma_k\}$, $X=\{\gamma_{i_0},...,\gamma_{i_m}\}\subseteq\Gamma$, $b=\gamma_k\in\Gamma\backslash X$
- **Codierung der Zustandsüberführungsfunktion δ**
  - Beschreibe $\delta(s,a)=(s',a',P)$ durch $\mathrm{code}(\delta(s,a))\ \equiv\ s\,a\,s'\,a'\,P$
  - Beschreibe δ durch das Wort $\mathrm{code}(\delta(s_0,\gamma_0))\ldots\mathrm{code}(\delta(s_n,\gamma_k))$
- **Codierung der Turingmaschine τ**
  - Beschreibe $\tau=(S,\ X,\ \Gamma,\ \delta,\ s_0,\ b)$ durch das Wort
    $$\hat{w}_\tau\ \equiv\ \gamma_{i_0}...\gamma_{i_m}\#\mathrm{code}(\delta(s_0,\gamma_0))\ldots\mathrm{code}(\delta(s_n,\gamma_k))$$
  - $\boldsymbol{w_\tau}$ sei die Codierung von $\hat{w}_\tau$ im festen Alphabet $\{0,1\}$ $(\hat{\gamma}_j\in\hat{\Gamma}\ \hat{=}\ \underbrace{0...0}_{j-mal}1)$

Viele Varianten in Details der Codierung

Numerierung von Worten

- **Bestimme lexikograpische Ordnung auf Worten**

Numerierung von Worten

- **Bestimme lexikograpische Ordnung auf Worten**
  - Worte über einem Alphabet $X = \{x_1, .., x_n\}$ können geordnet werden

$$\epsilon < x_1 < \ldots < x_n < x_1x_1 < x_1x_2 < \ldots < x_nx_n < x_1x_1x_1 < \ldots$$

Numerierung von Worten

- **Bestimme lexikograpische Ordnung auf Worten**
  - Worte über einem Alphabet $X = \{x_1, .., x_n\}$ können geordnet werden

    $$\epsilon < x_1 < \ldots < x_n < x_1x_1 < x_1x_2 < \ldots < x_nx_n < x_1x_1x_1 < \ldots$$

  - $\boldsymbol{u < v}$, falls $|u| < |v|$ oder $u = u_1...u_m,\ \ v = v_1...v_m,$

    $u_k{<}v_k$ für ein $k{\leq}m$ und $u_i{=}v_i$ für alle $i{<}k$

Numerierung von Worten

- **Bestimme lexikograpische Ordnung auf Worten**
  - Worte über einem Alphabet $X = \{x_1, .., x_n\}$ können geordnet werden

    $$\epsilon < x_1 < \ldots < x_n < x_1x_1 < x_1x_2 < \ldots < x_nx_n < x_1x_1x_1 < \ldots$$

  - $\boldsymbol{u < v}$, falls $|u| < |v|$ oder $u = u_1...u_m,\ \ v = v_1...v_m,$

    $u_k{<}v_k$ für ein $k{\leq}m$ und $u_i{=}v_i$ für alle $i{<}k$

  - Dabei $x_i{<}x_j$, falls $i{<}j$

Numerierung von Worten

- **Bestimme lexikograpische Ordnung auf Worten**
  - Worte über einem Alphabet $X = \{x_1, .., x_n\}$ können geordnet werden

    $$\epsilon < x_1 < \ldots < x_n < x_1x_1 < x_1x_2 < \ldots < x_nx_n < x_1x_1x_1 < \ldots$$

  - $\boldsymbol{u < v}$, falls $|u| < |v|$ oder $u = u_1...u_m,\ \ v = v_1...v_m,$

    $u_k{<}v_k$ für ein $k{\leq}m$ und $u_i{=}v_i$ für alle $i{<}k$
  - Dabei $x_i{<}x_j$, falls $i{<}j$

- **Numeriere gemäß der lexikograpischen Ordnung**
  - $\boldsymbol{\nu(i)}$ sei das Wort mit der Nummer i

Numerierung von Worten

- **Bestimme lexikograpische Ordnung auf Worten**
  - Worte über einem Alphabet $X = \{x_1, .., x_n\}$ können geordnet werden

    $$\epsilon < x_1 < \ldots < x_n < x_1x_1 < x_1x_2 < \ldots < x_nx_n < x_1x_1x_1 < \ldots$$

  - $\boldsymbol{u < v}$, falls $|u| < |v|$ oder $u = u_1...u_m,\ v = v_1...v_m,$

    $u_k{<}v_k$ für ein $k{\leq}m$ und $u_i{=}v_i$ für alle $i{<}k$
  - Dabei $x_i{<}x_j$, falls $i{<}j$

- **Numeriere gemäß der lexikograpischen Ordnung**
  - $\boldsymbol{\nu(i)}$ sei das Wort mit der Nummer i
  - $\nu(0){=}\epsilon$, $\nu(1){=}x_1$, $\ldots\nu(n){=}x_n$, $\nu(n{+}1){=}x_1x_1$, $\ldots\nu(n^2{+}n){=}x_nx_n$, $\ldots$

Numerierung von Worten

- **Bestimme lexikograpische Ordnung auf Worten**
  - Worte über einem Alphabet $X = \{x_1, .., x_n\}$ können geordnet werden

    $$\epsilon < x_1 < \ldots < x_n < x_1x_1 < x_1x_2 < \ldots < x_nx_n < x_1x_1x_1 < \ldots$$

  - $\boldsymbol{u < v}$, falls $|u| < |v|$ oder $u = u_1...u_m,\ \ v = v_1...v_m,$

    $u_k{<}v_k$ für ein $k{\leq}m$ und $u_i{=}v_i$ für alle $i{<}k$
  - Dabei $x_i{<}x_j$, falls $i{<}j$

- **Numeriere gemäß der lexikograpischen Ordnung**
  - $\boldsymbol{\nu(i)}$ sei das Wort mit der Nummer i
  - $\nu(0){=}\epsilon$, $\nu(1){=}x_1$, $\ldots \nu(n){=}x_n$, $\nu(n{+}1){=}x_1x_1$, $\ldots \nu(n^2{+}n){=}x_nx_n$, $\ldots$

    $\nu(i)$ entspricht der n-adischen Darstellung der Zahl i

Numerierung von Turingmaschinen

- $\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists\tau\text{:TM } w{=}w_\tau\}$ **ist entscheidbar**
  – Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt

Numerierung von Turingmaschinen

- $\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists\tau{:}\text{TM}\ w{=}w_\tau\}$ **ist entscheidbar**
  - Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt
  - Bestimme X: Menge der Symbole in w bis zum #
  - Bestimme δ: je 5 Symbole beschreiben einen Tabelleneintrag
  - Bestimme S, Γ, s_0 und b aus δ
  - Prüfe Vollständigkeit und korrekte Anordnung der Tabelle für δ

Numerierung von Turingmaschinen

- **$\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists \tau\text{:TM } w{=}w_\tau\}$ ist entscheidbar**
  - Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt
  - Bestimme X: Menge der Symbole in w bis zum #
  - Bestimme δ: je 5 Symbole beschreiben einen Tabelleneintrag
  - Bestimme S, Γ, s_0 und b aus δ
  - Prüfe Vollständigkeit und korrekte Anordnung der Tabelle für δ
  - $\hat{\Gamma}$ wird implizit identifiziert

Numerierung von Turingmaschinen

- **$\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists \tau\text{:TM } w{=}w_\tau\}$ ist entscheidbar**
  - Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt
  - Bestimme X: Menge der Symbole in w bis zum #
  - Bestimme δ: je 5 Symbole beschreiben einen Tabelleneintrag
  - Bestimme S, Γ, s_0 und b aus δ
  - Prüfe Vollständigkeit und korrekte Anordnung der Tabelle für δ
  - $\hat{\Gamma}$ wird implizit identifiziert

- **Numeriere Worte, die Turingmaschinen codieren**
  - $n_\tau(0) := \min\{j \mid \nu(j) \in \mathcal{TM}\}$ $\quad n_\tau(i{+}1) := \min\{j{>}n_\tau(i) \mid \nu(j) \in \mathcal{TM}\}$

Numerierung von Turingmaschinen

- **$\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists\tau\text{:TM } w{=}w_\tau\}$ ist entscheidbar**
  - Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt
  - Bestimme X: Menge der Symbole in w bis zum #
  - Bestimme δ: je 5 Symbole beschreiben einen Tabelleneintrag
  - Bestimme S, Γ, s_0 und b aus δ
  - Prüfe Vollständigkeit und korrekte Anordnung der Tabelle für δ
  - $\hat{\Gamma}$ wird implizit identifiziert

- **Numeriere Worte, die Turingmaschinen codieren**
  - $n_\tau(0) := \min\{j \mid \nu(j) \in \mathcal{TM}\}$ $\quad n_\tau(i{+}1) := \min\{j{>}n_\tau(i) \mid \nu(j) \in \mathcal{TM}\}$

    n_τ:$\mathbb{N}\longrightarrow\mathbb{N}$ ist berechenbar

Numerierung von Turingmaschinen

- $\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists\tau\text{:TM } w{=}w_\tau\}$ **ist entscheidbar**
  - Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt
  - Bestimme X: Menge der Symbole in w bis zum #
  - Bestimme δ: je 5 Symbole beschreiben einen Tabelleneintrag
  - Bestimme S, Γ, s_0 und b aus δ
  - Prüfe Vollständigkeit und korrekte Anordnung der Tabelle für δ
  - $\hat{\Gamma}$ wird implizit identifiziert

- **Numeriere Worte, die Turingmaschinen codieren**
  - $n_\tau(0) := \min\{j \mid \nu(j) \in \mathcal{TM}\}$ $\quad n_\tau(i{+}1) := \min\{j{>}n_\tau(i) \mid \nu(j) \in \mathcal{TM}\}$

    $\boldsymbol{n_\tau}$**:$\mathbb{N}\to\mathbb{N}$ ist berechenbar**
  - $\boldsymbol{\tau_i}$: Turingmaschine τ mit $w_\tau{=}\nu(n_\tau(i))$ "die i-te Turingmaschine"

Numerierung von Turingmaschinen

- $\boldsymbol{\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists \tau\text{:TM } w{=}w_\tau\}}$ **ist entscheidbar**
  – Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt
  – Bestimme X: Menge der Symbole in w bis zum #
  – Bestimme δ: je 5 Symbole beschreiben einen Tabelleneintrag
  – Bestimme S, Γ, s_0 und b aus δ
  – Prüfe Vollständigkeit und korrekte Anordnung der Tabelle für δ
  – $\hat{\Gamma}$ wird implizit identifiziert

- **Numeriere Worte, die Turingmaschinen codieren**
  – $n_\tau(0) := \min\{j \mid \nu(j) \in \mathcal{TM}\}$ $\quad n_\tau(i{+}1) := \min\{j{>}n_\tau(i) \mid \nu(j) \in \mathcal{TM}\}$
  $\boldsymbol{n_\tau}$**:$\mathbb{N}{\to}\mathbb{N}$ ist berechenbar**
  – $\boldsymbol{\tau_i}$: Turingmaschine τ mit $w_\tau{=}\nu(n_\tau(i))$ “die i-te Turingmaschine”
  – **Gödelnummer** der Turingmaschine τ: Zahl i mit $\tau = \tau_i$

Numerierung von Turingmaschinen

- $\mathcal{TM} = \{w \in \{0,1\}^* \mid \exists\tau{:}\mathrm{TM}\ w{=}w_\tau\}$ **ist entscheidbar**
  - Man kann testen, ob ein Wort $w \in \hat{\Gamma}^*$ ein Turingprogramm beschreibt
  - Bestimme X: Menge der Symbole in w bis zum #
  - Bestimme δ: je 5 Symbole beschreiben einen Tabelleneintrag
  - Bestimme S, Γ, s_0 und b aus δ
  - Prüfe Vollständigkeit und korrekte Anordnung der Tabelle für δ
  - $\hat{\Gamma}$ wird implizit identifiziert

- **Numeriere Worte, die Turingmaschinen codieren**
  - $n_\tau(0) := \min\{j \mid \nu(j) \in \mathcal{TM}\}$ $\quad n_\tau(i{+}1) := \min\{j{>}n_\tau(i) \mid \nu(j) \in \mathcal{TM}\}$

    $n_\tau{:}\mathbb{N}{\longrightarrow}\mathbb{N}$ **ist berechenbar**
  - τ_i: Turingmaschine τ mit $w_\tau{=}\nu(n_\tau(i))$ "die i-te Turingmaschine"
  - **Gödelnummer** der Turingmaschine τ: Zahl i mit $\tau = \tau_i$

Numerierung von Programmen ist bijektiv

Numerierung berechenbarer Funktionen

- **Berechenbare Funktionen auf Worten**
  - $\hat{\varphi}_i \equiv h_{\tau_i}$: "die von der i-ten Turingmaschine berechnete Funktion"

Numerierung berechenbarer Funktionen

- **Berechenbare Funktionen auf Worten**
  - $\hat{\varphi}_i \equiv h_{\tau_i}$: “die von der i-ten Turingmaschine berechnete Funktion”
  - $\boldsymbol{t_i}$: Schrittzahlfunktion der Turingmaschine τ_i Kap. 6, Def. E

$$t_i(w) = \begin{cases} m & \text{falls Berechnung von } \tau_i(w) \text{ in } m \text{ Schritten terminiert,} \\ \perp & \text{sonst} \end{cases}$$

Numerierung berechenbarer Funktionen

- **Berechenbare Funktionen auf Worten**
  - $\hat{\varphi}_i \equiv h_{\tau_i}$: "die von der i-ten Turingmaschine berechnete Funktion"
  - $\boldsymbol{t_i}$: Schrittzahlfunktion der Turingmaschine τ_i Kap. 6, Def. E

$$t_i(w) = \begin{cases} m & \text{falls Berechnung von } \tau_i(w) \text{ in } m \text{ Schritten terminiert,} \\ \perp & \text{sonst} \end{cases}$$

- **Berechenbare Funktionen auf Zahlen**
  - $\boldsymbol{\varphi_i} \equiv r^{-1} \circ \hat{\varphi}_i \circ r : \mathbb{N} \rightarrow \mathbb{N}$ "die i-te berechenbare Funktion"

    $r{:}\mathbb{N} \rightarrow X^*$ bijektive Repräsentation von Zahlen als Worte
  - $\boldsymbol{\Phi_i} \equiv t_i \circ r : \mathbb{N} \rightarrow \mathbb{N}$ "Schrittzahlfunktion von φ_i"

Numerierung berechenbarer Funktionen

- **Berechenbare Funktionen auf Worten**
  - $\hat{\varphi}_i \equiv h_{\tau_i}$: "die von der i-ten Turingmaschine berechnete Funktion"
  - $\boldsymbol{t_i}$: Schrittzahlfunktion der Turingmaschine τ_i Kap. 6, Def. E

$$t_i(w) = \begin{cases} m & \text{falls Berechnung von } \tau_i(w) \text{ in } m \text{ Schritten terminiert,} \\ \perp & \text{sonst} \end{cases}$$

- **Berechenbare Funktionen auf Zahlen**
  - $\boldsymbol{\varphi_i} \equiv r^{-1}\circ\hat{\varphi}_i\circ r : \mathbb{N}\to\mathbb{N}$ "die i-te berechenbare Funktion"

    $r{:}\mathbb{N}\to X^*$ bijektive Repräsentation von Zahlen als Worte
  - $\boldsymbol{\Phi_i} \equiv t_i\circ r : \mathbb{N}\to\mathbb{N}$ "Schrittzahlfunktion von φ_i"

- **Eigenschaften von $\boldsymbol{\varphi}$ und $\boldsymbol{\Phi}$**
  - φ is surjektiv, aber nicht bijektiv
  - $\text{domain}(\Phi_i) = \text{domain}(\varphi_i)$ (Φ_i terminiert auf den gleichen Eingaben wie φ_i)
  - $\{(i,n,t) \mid \Phi_i(n){=}t\}$ ist entscheidbar "Rechenzeit ist entscheidbar"

Numerierung berechenbarer Funktionen

- **Berechenbare Funktionen auf Worten**
  - $\hat{\varphi}_i \equiv h_{\tau_i}$: "die von der i-ten Turingmaschine berechnete Funktion"
  - $\boldsymbol{t_i}$: Schrittzahlfunktion der Turingmaschine τ_i Kap. 6, Def. E

$$t_i(w) = \begin{cases} m & \text{falls Berechnung von } \tau_i(w) \text{ in } m \text{ Schritten terminiert,} \\ \perp & \text{sonst} \end{cases}$$

- **Berechenbare Funktionen auf Zahlen**
  - $\boldsymbol{\varphi_i} \equiv r^{-1} \circ \hat{\varphi}_i \circ r : \mathbb{N} \to \mathbb{N}$ "die i-te berechenbare Funktion"

    $r{:}\mathbb{N} \to X^*$ bijektive Repräsentation von Zahlen als Worte
  - $\boldsymbol{\Phi_i} \equiv t_i \circ r : \mathbb{N} \to \mathbb{N}$ "Schrittzahlfunktion von φ_i"

- **Eigenschaften von φ und Φ**
  - φ is surjektiv, aber nicht bijektiv
  - $\text{domain}(\Phi_i) = \text{domain}(\varphi_i)$ (Φ_i terminiert auf den gleichen Eingaben wie φ_i)
  - $\{(i, n, t) \mid \Phi_i(n) = t\}$ ist entscheidbar "Rechenzeit ist entscheidbar"

Die Numerierung berechenbarer Funktionen ist nur surjektiv

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

- **Universelle Maschinen** Definiton H
  - τ_u ist universell, wenn $h_{\tau_u}(w_\tau, v) = h_\tau(v)$ für jede TM τ und jedes $v \in X^*$

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

- **Universelle Maschinen** Definiton H
  - τ_u ist universell, wenn $h_{\tau_u}(w_\tau, v) = h_\tau(v)$ für jede TM τ und jedes $v \in X^*$
  - Insbesondere $h_{\tau_u}(r(i), v) = h_{\tau_i}(v)$ für alle i,v (r:$\mathbb{N}\rightarrow X^*$ Zahlendarstellung)

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

- **Universelle Maschinen** Definiton H
  - τ_u ist universell, wenn $h_{\tau_u}(w_\tau, v) = h_\tau(v)$ für jede TM τ und jedes $v \in X^*$
  - Insbesondere $h_{\tau_u}(r(i), v) = h_{\tau_i}(v)$ für alle i,v (r:$\mathbb{N} \to X^*$ Zahlendarstellung)
- **Universelle Funktionen** Definiton I
  - u:$\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ ist universell, wenn $u(i, n) = \varphi_i(n)$ für alle $i, n \in \mathbb{N}$

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

- **Universelle Maschinen** Definiton H
  - τ_u ist universell, wenn $h_{\tau_u}(w_\tau, v) = h_\tau(v)$ für jede TM τ und jedes $v \in X^*$
  - Insbesondere $h_{\tau_u}(r(i), v) = h_{\tau_i}(v)$ für alle i,v (r:$\mathbb{N} \to X^*$ Zahlendarstellung)
- **Universelle Funktionen** Definiton I
  - u:$\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ ist universell, wenn $u(i, n) = \varphi_i(n)$ für alle $i, n \in \mathbb{N}$
- **Gibt es universelle Maschinen?**

Universelle Turingmaschinen

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

- **Universelle Maschinen** Definiton H
  - τ_u ist universell, wenn $h_{\tau_u}(w_\tau, v) = h_\tau(v)$ für jede TM τ und jedes $v \in X^*$
  - Insbesondere $h_{\tau_u}(r(i), v) = h_{\tau_i}(v)$ für alle i,v (r:$\mathbb{N} \to X^*$ Zahlendarstellung)
- **Universelle Funktionen** Definiton I
  - u:$\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ ist universell, wenn $u(i, n) = \varphi_i(n)$ für alle $i, n \in \mathbb{N}$
- **Gibt es universelle Maschinen?**
  - Die Numerierung n_τ ist berechenbar

Universelle Turingmaschinen

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

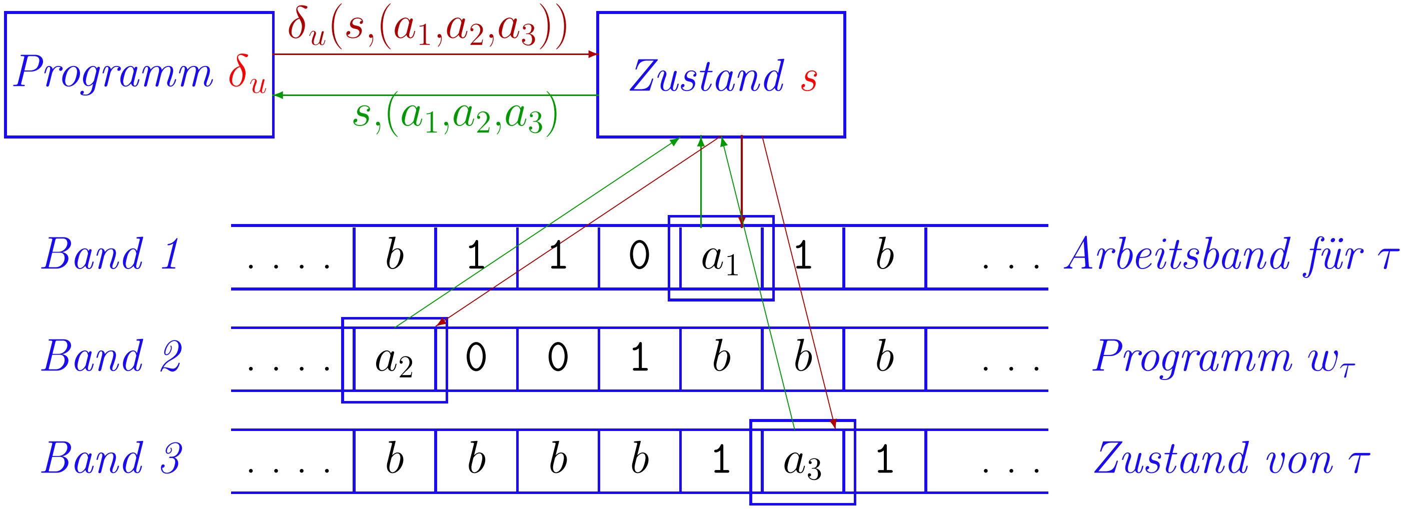
- **Universelle Maschinen** Definiton H
  - τ_u ist universell, wenn $h_{\tau_u}(w_\tau, v) = h_\tau(v)$ für jede TM τ und jedes $v \in X^*$
  - Insbesondere $h_{\tau_u}(r(i), v) = h_{\tau_i}(v)$ für alle i,v (r:$\mathbb{N} \rightarrow X^*$ Zahlendarstellung)
- **Universelle Funktionen** Definiton I
  - u:$\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ ist universell, wenn $u(i, n) = \varphi_i(n)$ für alle $i, n \in \mathbb{N}$
- **Gibt es universelle Maschinen?**
  - Die Numerierung n_τ ist berechenbar
  - Turingprogramme lassen sich simulieren

Universelle Turingmaschinen

Kann man alle Turingprogramme auf einer einzigen Maschine ausführen?

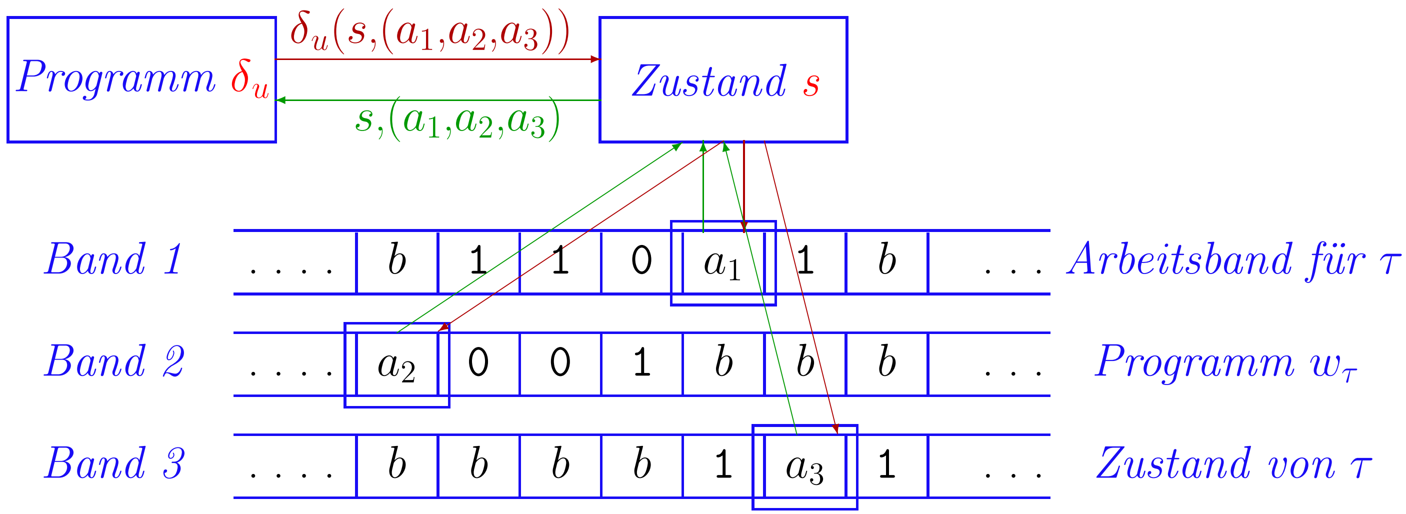
- **Universelle Maschinen** Definiton H
  - τ_u ist universell, wenn $h_{\tau_u}(w_\tau, v) = h_\tau(v)$ für jede TM τ und jedes $v \in X^*$
  - Insbesondere $h_{\tau_u}(r(i), v) = h_{\tau_i}(v)$ für alle i,v (r:$\mathbb{N} \to X^*$ Zahlendarstellung)
- **Universelle Funktionen** Definiton I
  - u:$\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ ist universell, wenn $u(i, n) = \varphi_i(n)$ für alle $i, n \in \mathbb{N}$
- **Gibt es universelle Maschinen?**
  - Die Numerierung n_τ ist berechenbar
  - Turingprogramme lassen sich simulieren
  - Baue universelle Maschine mit $\nu \circ n_\tau$ und Einzelschrittsimulation

Programmierung Universeller Turingmaschinen



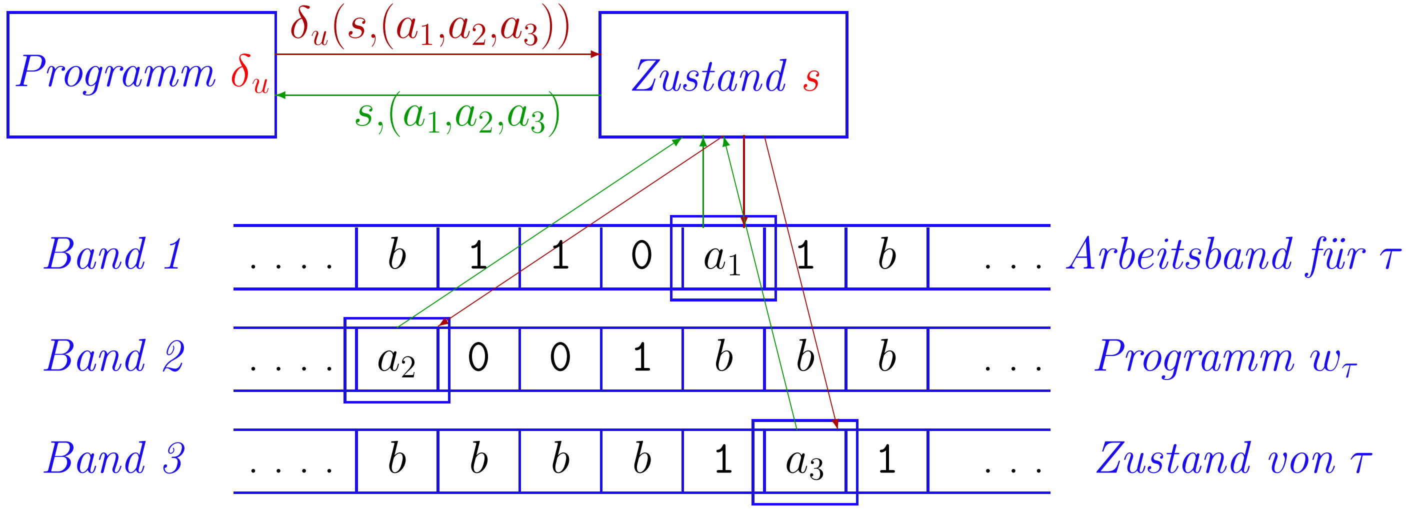
- **Benutze 3 Arbeitsbänder (+ Hilfsbänder)**
  - 1. Eingabe- und Arbeitsband der simulierten Turingmaschine τ
  - 2. w_τ: Codierung des Programms von τ
  - 3. Aktueller Zustand von τ

Programmierung Universeller Turingmaschinen



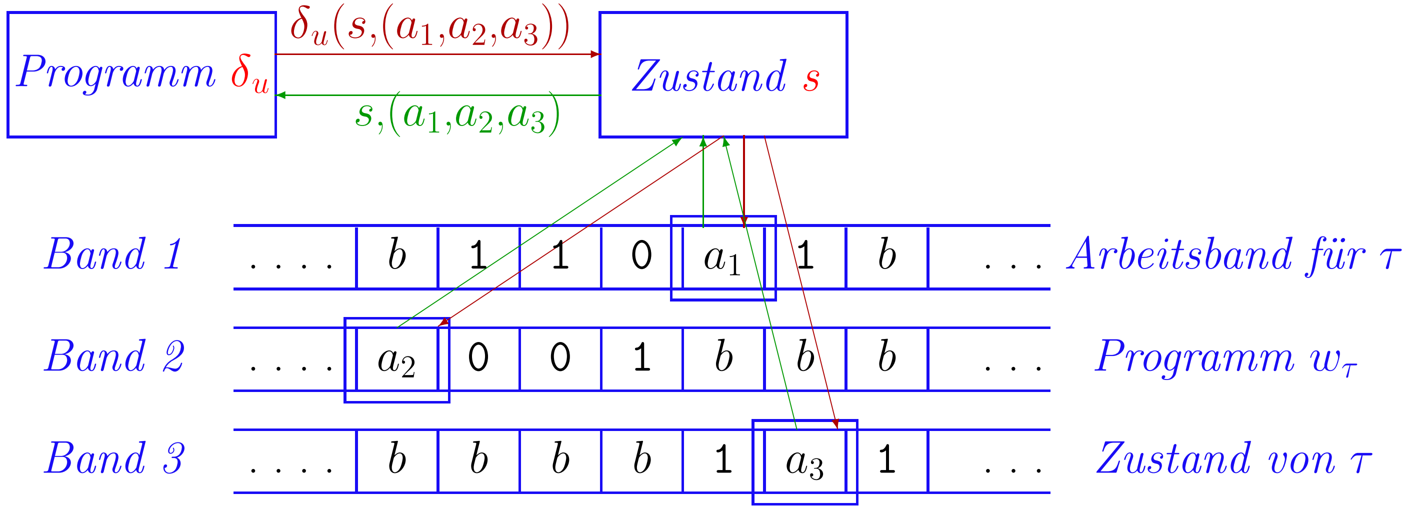
- **Benutze 3 Arbeitsbänder (+ Hilfsbänder)**
  - 1. Eingabe- und Arbeitsband der simulierten Turingmaschine τ
  - 2. w_τ: Codierung des Programms von τ
  - 3. Aktueller Zustand von τ
- **Generiere und simuliere Programm von τ**

Programmierung Universeller Turingmaschinen



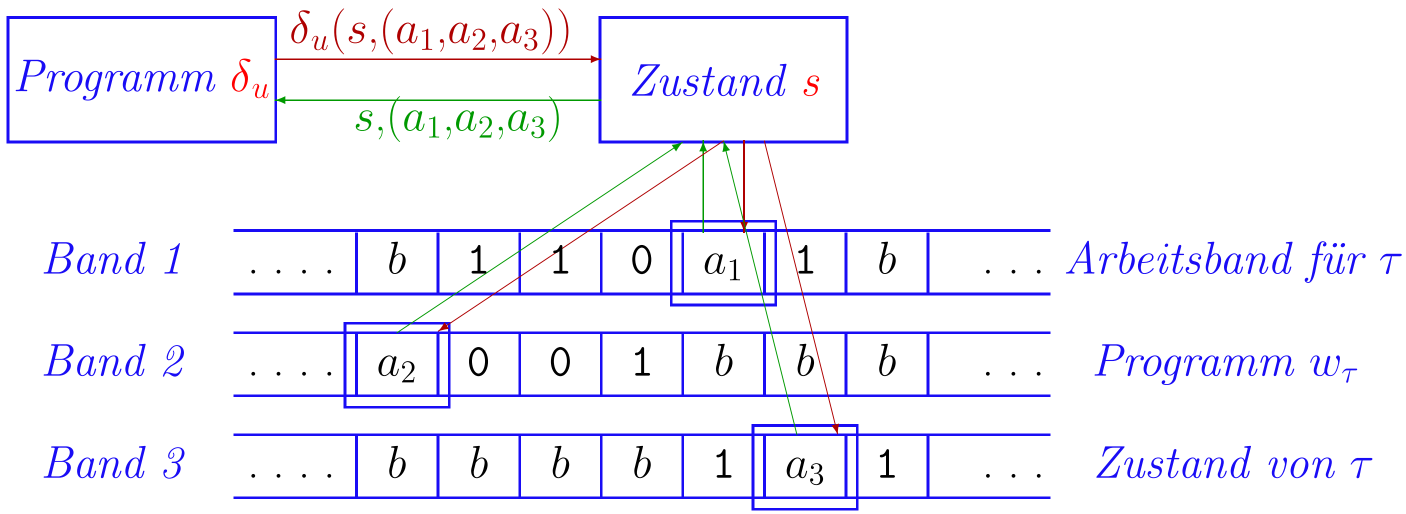
- **Benutze 3 Arbeitsbänder (+ Hilfsbänder)**
  - 1. Eingabe- und Arbeitsband der simulierten Turingmaschine τ
  - 2. w_τ: Codierung des Programms von τ
  - 3. Aktueller Zustand von τ
- **Generiere und simuliere Programm von τ**
  - Bei Eingabe (der Codierung von) i, x berechne $w_\tau = \nu(n_\tau)(i)$
  - Schreibe i auf Band 1, w_τ auf Band 2, Anfangszustand von τ auf Band 3

Programmierung Universeller Turingmaschinen



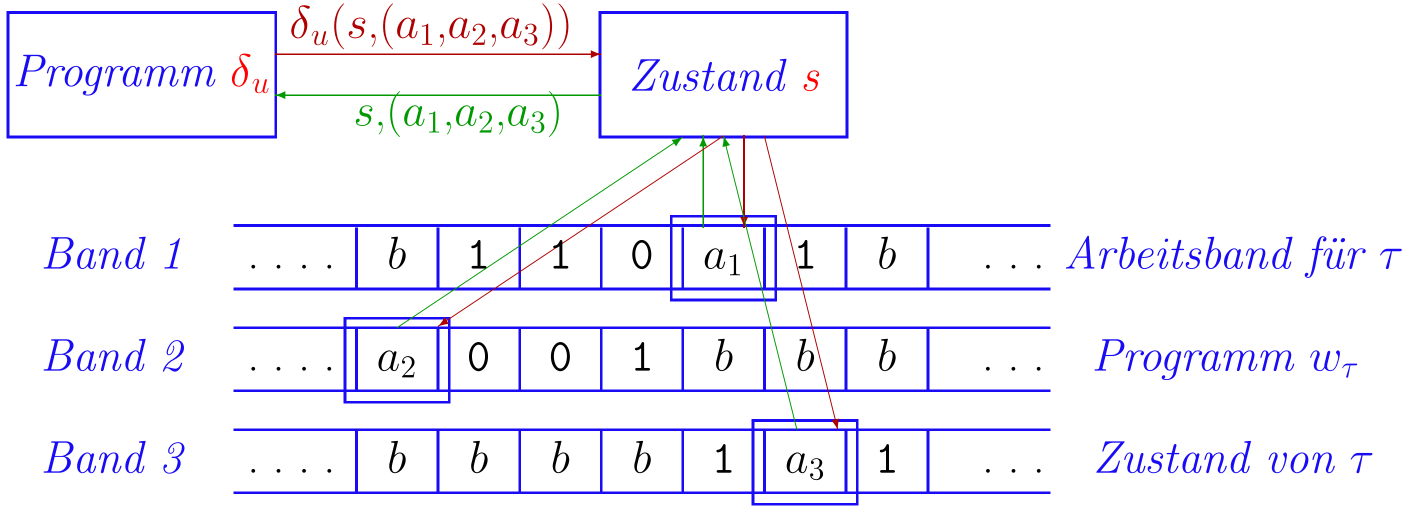
- **Benutze 3 Arbeitsbänder (+ Hilfsbänder)**
  - 1. Eingabe- und Arbeitsband der simulierten Turingmaschine τ
  - 2. w_τ: Codierung des Programms von τ
  - 3. Aktueller Zustand von τ
- **Generiere und simuliere Programm von τ**
  - Bei Eingabe (der Codierung von) i, x berechne $w_\tau = \nu(n_\tau)(i)$
  - Schreibe i auf Band 1, w_τ auf Band 2, Anfangszustand von τ auf Band 3
  - Simuliere Einzelschritte von τ gemäß Programm w_τ

Programmierung Universeller Turingmaschinen



- **Benutze 3 Arbeitsbänder (+ Hilfsbänder)**
  – 1. Eingabe- und Arbeitsband der simulierten Turingmaschine τ
  – 2. w_τ: Codierung des Programms von τ
  – 3. Aktueller Zustand von τ
- **Generiere und simuliere Programm von τ**
  – Bei Eingabe (der Codierung von) i, x berechne $w_\tau = \nu(n_\tau)(i)$
  – Schreibe i auf Band 1, w_τ auf Band 2, Anfangszustand von τ auf Band 3
  – Simuliere Einzelschritte von τ gemäß Programm w_τ
  – Bei Terminierung steht Ausgabewort auf Band 1

Programmierung Universeller Turingmaschinen



- **Benutze 3 Arbeitsbänder (+ Hilfsbänder)**
  - 1. Eingabe- und Arbeitsband der simulierten Turingmaschine τ
  - 2. w_τ: Codierung des Programms von τ
  - 3. Aktueller Zustand von τ
- **Generiere und simuliere Programm von τ**
  - Bei Eingabe (der Codierung von) i, x berechne $w_\tau = \nu(n_\tau)(i)$
  - Schreibe i auf Band 1, w_τ auf Band 2, Anfangszustand von τ auf Band 3
  - Simuliere Einzelschritte von τ gemäß Programm w_τ
  - Bei Terminierung steht Ausgabewort auf Band 1

Details z.B. in Hopcroft, Motwani, Ullman, Seite 387–389

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  - Umbenennung der Zustände von τ_2
  - Springe vom "Endzustand" von τ_1 zum Anfangszustand von τ_2

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  - Umbenennung der Zustände von τ_2
  - Springe vom "Endzustand" von τ_1 zum Anfangszustand von τ_2
  - Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit** $h_\tau = h_{\tau_1} \circ h_{\tau_2}$
  - Umbenennung der Zustände von τ_2
  - Springe vom "Endzustand" von τ_1 zum Anfangszustand von τ_2
  - Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden
  - Gödelnummer k von τ kann aus denen für τ_1 und τ_2 berechnet werden

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  - Umbenennung der Zustände von τ_2
  - Springe vom “Endzustand” von τ_1 zum Anfangszustand von τ_2
  - Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden
  - Gödelnummer k von τ kann aus denen für τ_1 und τ_2 berechnet werden
- **Kombiniere φ_i und φ_j zu φ_k mit $\varphi_k = \varphi_i \circ \varphi_j$**

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  - Umbenennung der Zustände von τ_2
  - Springe vom “Endzustand” von τ_1 zum Anfangszustand von τ_2
  - Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden
  - Gödelnummer k von τ kann aus denen für τ_1 und τ_2 berechnet werden
- **Kombiniere φ_i und φ_j zu φ_k mit $\varphi_k = \varphi_i \circ \varphi_j$**
  - Index k kann aus i und j berechnet werden

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  – Umbenennung der Zustände von τ_2
  – Springe vom "Endzustand" von τ_1 zum Anfangszustand von τ_2
  – Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden
  – Gödelnummer k von τ kann aus denen für τ_1 und τ_2 berechnet werden
- **Kombiniere φ_i und φ_j zu φ_k mit $\varphi_k = \varphi_i \circ \varphi_j$**
  – Index k kann aus i und j berechnet werden
  – Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)} = \varphi_i \circ \varphi_j$

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  - Umbenennung der Zustände von τ_2
  - Springe vom "Endzustand" von τ_1 zum Anfangszustand von τ_2
  - Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden
  - Gödelnummer k von τ kann aus denen für τ_1 und τ_2 berechnet werden
- **Kombiniere φ_i und φ_j zu φ_k mit $\varphi_k = \varphi_i \circ \varphi_j$**
  - Index k kann aus i und j berechnet werden
  - Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)} = \varphi_i \circ \varphi_j$
- **Allgemeinste Version: SMN Theorem**

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  - Umbenennung der Zustände von τ_2
  - Springe vom "Endzustand" von τ_1 zum Anfangszustand von τ_2
  - Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden
  - Gödelnummer k von τ kann aus denen für τ_1 und τ_2 berechnet werden
- **Kombiniere φ_i und φ_j zu φ_k mit $\varphi_k = \varphi_i \circ \varphi_j$**
  - Index k kann aus i und j berechnet werden
  - Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)} = \varphi_i \circ \varphi_j$
- **Allgemeinste Version: SMN Theorem**
  - Es gibt eine berechenbare totale Funktion s mit $\varphi_{s\langle m,n\rangle}(i) = \varphi_m\langle n, i\rangle$

Das Übersetzungslemma

Turingmaschinen sind effektiv kombinierbar

- **Kombiniere τ_1 und τ_2 zu τ mit $h_\tau = h_{\tau_1} \circ h_{\tau_2}$**
  - Umbenennung der Zustände von τ_2
  - Springe vom "Endzustand" von τ_1 zum Anfangszustand von τ_2
  - Programm w_τ kann aus w_{τ_1} und w_{τ_2} berechnet werden
  - Gödelnummer k von τ kann aus denen für τ_1 und τ_2 berechnet werden
- **Kombiniere φ_i und φ_j zu φ_k mit $\varphi_k = \varphi_i \circ \varphi_j$**
  - Index k kann aus i und j berechnet werden
  - Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)} = \varphi_i \circ \varphi_j$
- **Allgemeinste Version: SMN Theorem**
  - Es gibt eine berechenbare totale Funktion s mit $\varphi_{s\langle m,n\rangle}(i) = \varphi_m\langle n, i\rangle$

Technisches Resultat mit wenig eigener Bedeutung

Zusammenfassung: Kernaxiome der Berechenbareitstheorie

- **Berechenbare Funktionen sind effektiv numerierbar**
  - $\varphi_i : \mathbb{N} \to \mathbb{N}$: berechnete Funktion des Programms i
  - Φ_i: Rechenzeitfunktion zum Programm i
  - **domain(Φ_i) = domain(φ_i)**

Zusammenfassung: Kernaxiome der Berechenbarkeitstheorie

- **Berechenbare Funktionen sind effektiv numerierbar**
  - $\varphi_i : \mathbb{N} \rightarrow \mathbb{N}$: berechnete Funktion des Programms i
  - Φ_i: Rechenzeitfunktion zum Programm i
  - **domain(Φ_i) = domain(φ_i)**

- **Die Menge $\{(i, n, t) \mid \Phi_i(n) = t\}$ ist entscheidbar**

Zusammenfassung: Kernaxiome der Berechenbareitstheorie

- **Berechenbare Funktionen sind effektiv numerierbar**
  - $\varphi_i : \mathbb{N}\to\mathbb{N}$: berechnete Funktion des Programms i
  - Φ_i: Rechenzeitfunktion zum Programm i
  - **domain(Φ_i) = domain(φ_i)**
- **Die Menge $\{(i,n,t) \mid \Phi_i(n)=t\}$ ist entscheidbar**
- **Die universelle Funktion ist berechenbar** UTM Theorem
  - $u : \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ mit $u(i,n) = \varphi_i(n)$ ist berechenbar

Zusammenfassung: Kernaxiome der Berechenbarkeitstheorie

- **Berechenbare Funktionen sind effektiv numerierbar**
  - $\varphi_i : \mathbb{N} \to \mathbb{N}$: berechnete Funktion des Programms i
  - Φ_i: Rechenzeitfunktion zum Programm i
  - **domain(Φ_i) = domain(φ_i)**

- **Die Menge $\{(i, n, t) \mid \Phi_i(n) = t\}$ ist entscheidbar**

- **Die universelle Funktion ist berechenbar** $\boxed{\text{UTM Theorem}}$
  - $u : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ mit $u(i, n) = \varphi_i(n)$ ist berechenbar

- **Programme sind effektiv kombinierbar** $\boxed{\text{SMN Theorem}}$
  - Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)} = \varphi_i \circ \varphi_j$
  - Es gibt eine berechenbare totale Funktion s mit $\varphi_{s\langle m,n \rangle}(i) = \varphi_m \langle n, i \rangle$

Zusammenfassung: Kernaxiome der Berechenbareitstheorie

- **Berechenbare Funktionen sind effektiv numerierbar**
  - $\varphi_i:\mathbb{N}\to\mathbb{N}$: berechnete Funktion des Programms i
  - Φ_i: Rechenzeitfunktion zum Programm i
  - **domain(Φ_i) = domain(φ_i)**

- **Die Menge $\{(i,n,t)\mid\Phi_i(n)=t\}$ ist entscheidbar**

- **Die universelle Funktion ist berechenbar** UTM Theorem
  - $u:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ mit $u(i,n)=\varphi_i(n)$ ist berechenbar

- **Programme sind effektiv kombinierbar** SMN Theorem
  - Es gibt eine berechenbare totale Funktion h mit $\varphi_{h(i,j)}=\varphi_i\circ\varphi_j$
  - Es gibt eine berechenbare totale Funktion s mit $\varphi_{s\langle m,n\rangle}(i)=\varphi_m\langle n,i\rangle$

Alles weitere folgt aus diesen Axiomen

Kleene Normalform Theorem

Es gibt berechenbare totale Funktionen f, g und h mit $\varphi_i(n) = g(\mu f(i,n))$ und $\Phi_i(n) = h(\mu f(i,n))$

Es gibt berechenbare totale Funktionen f, g und h mit $\varphi_i(n) = g(\mu f(i,n))$ und $\Phi_i(n) = h(\mu f(i,n))$

- **Beweis**
  - Definiere $f(i,n,\langle y,t\rangle) = \begin{cases} 0 & \text{falls } \Phi_i(n){=}t \text{ und } \varphi_i(n){=}y \\ 1 & \text{sonst} \end{cases}$

Kleene Normalform Theorem

Es gibt berechenbare totale Funktionen f, g und h mit $\varphi_i(n) = g(\mu f(i,n))$ und $\Phi_i(n) = h(\mu f(i,n))$

- **Beweis**
  - Definiere $f(i, n, \langle y, t\rangle) = \begin{cases} 0 & \text{falls } \Phi_i(n){=}t \text{ und } \varphi_i(n){=}y \\ 1 & \text{sonst} \end{cases}$
  - $f{:}\mathbb{N}^3{\to}\mathbb{N}$ ist total berechenbar, da $\{(i,n,t) \mid \Phi_i(n){=}t\}$ entscheidbar ist

Kleene Normalform Theorem

Es gibt berechenbare totale Funktionen f, g und h mit $\varphi_i(n) = g(\mu f(i,n))$ und $\Phi_i(n) = h(\mu f(i,n))$

- **Beweis**
  - Definiere $f(i,n,\langle y,t\rangle) = \begin{cases} 0 & \text{falls } \Phi_i(n){=}t \text{ und } \varphi_i(n){=}y \\ 1 & \text{sonst} \end{cases}$
  - $f{:}\mathbb{N}^3{\to}\mathbb{N}$ ist total berechenbar, da $\{(i,n,t)\mid \Phi_i(n){=}t\}$ entscheidbar ist
  - Wähle $g{=}\pi^2_1$ und $h{=}\pi^2_2$

Kleene Normalform Theorem

Es gibt berechenbare totale Funktionen f, g und h mit $\varphi_i(n) = g(\mu f(i,n))$ und $\Phi_i(n) = h(\mu f(i,n))$

- **Beweis**
  - Definiere $f(i, n, \langle y, t\rangle) = \begin{cases} 0 & \text{falls } \Phi_i(n){=}t \text{ und } \varphi_i(n){=}y \\ 1 & \text{sonst} \end{cases}$
  - $f{:}\mathbb{N}^3{\to}\mathbb{N}$ ist total berechenbar, da $\{(i, n, t) \mid \Phi_i(n){=}t\}$ entscheidbar ist
  - Wähle $g{=}\pi^2_1$ und $h{=}\pi^2_2$

Kein Maschinenmodell nötig

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t)|\Phi_i(n)=t\}$ **ist entscheidbar** Rechenzeit

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t)|\Phi_i(n) = t\}$ **ist entscheidbar** Rechenzeit
  - Kernaxiom der Berechenbarkeitstheorie

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t)|\Phi_i(n)=t\}$ **ist entscheidbar** Rechenzeit
  - Kernaxiom der Berechenbarkeitstheorie
- $\{(i,n,y)|\varphi_i(n)=y\}$ **ist aufzählbar** Berechnung

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t) | \Phi_i(n) = t\}$ **ist entscheidbar** Rechenzeit
  – Kernaxiom der Berechenbarkeitstheorie
- $\{(i,n,y) | \varphi_i(n) = y\}$ **ist aufzählbar** Berechnung
  – Graph der universellen Funktion

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t) | \Phi_i(n) = t\}$ **ist entscheidbar** Rechenzeit
  – Kernaxiom der Berechenbarkeitstheorie
- $\{(i,n,y) | \varphi_i(n) = y\}$ **ist aufzählbar** Berechnung
  – Graph der universellen Funktion
- $H = \{(i,n) | \varphi_i(n) \neq \perp\}$ **ist aufzählbar** Halteproblem

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t)|\Phi_i(n) = t\}$ **ist entscheidbar** $\boxed{\text{Rechenzeit}}$
  – Kernaxiom der Berechenbarkeitstheorie
- $\{(i,n,y)|\varphi_i(n) = y\}$ **ist aufzählbar** $\boxed{\text{Berechnung}}$
  – Graph der universellen Funktion
- $H = \{(i,n)|\varphi_i(n)\neq\perp\}$ **ist aufzählbar** $\boxed{\text{Halteproblem}}$
  – Haltebereich der universellen Funktion

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t)|\Phi_i(n)=t\}$ **ist entscheidbar** — Rechenzeit
  – Kernaxiom der Berechenbarkeitstheorie
- $\{(i,n,y)|\varphi_i(n)=y\}$ **ist aufzählbar** — Berechnung
  – Graph der universellen Funktion
- $H=\{(i,n)|\varphi_i(n)\neq\perp\}$ **ist aufzählbar** — Halteproblem
  – Haltebereich der universellen Funktion
- $S=\{i|\varphi_i(i)\neq\perp\}$ **ist aufzählbar** — Selbstanwendbarkeitsproblem

Wichtige entscheidbare und aufzählbare Mengen

- $\{(i,n,t)|\Phi_i(n)=t\}$ **ist entscheidbar** Rechenzeit
  - Kernaxiom der Berechenbarkeitstheorie
- $\{(i,n,y)|\varphi_i(n)=y\}$ **ist aufzählbar** Berechnung
  - Graph der universellen Funktion
- $H=\{(i,n)|\varphi_i(n)\neq\perp\}$ **ist aufzählbar** Halteproblem
  - Haltebereich der universellen Funktion
- $S=\{i|\varphi_i(i)\neq\perp\}$ **ist aufzählbar** Selbstanwendbarkeitsproblem
  - Haltebereich von $\lambda i.u(i,i)$