

Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2018/19

Definition

- Eine NTM M **hält bei Eingabe** x (kurz: $M(x) = \downarrow$ oder $M(x) \downarrow$), falls alle Rechnungen von $M(x)$ nach endlich vielen Schritten halten
- Falls $M(x)$ nicht hält, schreiben wir auch kurz $M(x) = \uparrow$ oder $M(x) \uparrow$
- Eine DTM M **entscheidet** eine Eingabe x , falls $M(x)$ hält oder eine Konfiguration mit einem Endzustand erreicht
- Eine Sprache heißt **entscheidbar**, falls sie von einer DTM M akzeptiert wird, die alle Eingaben entscheidet. Die zugehörige Sprachklasse ist

$$\text{REC} = \{L(M) \mid M \text{ ist eine DTM, die alle Eingaben entscheidet}\}$$

- Jede von einer DTM akzeptierte Sprache heißt **semi-entscheidbar**

Bemerkung

- Eine DTM M **entscheidet zwar immer alle Eingaben** $x \in L(M)$, aber eventuell nicht alle $x \in \overline{L(M)}$. Daher heißt $L(M)$ semi-entscheidbar
- Später werden wir sehen, dass $\text{RE} = \{L(M) \mid M \text{ ist eine DTM}\}$ ist

Definition

- Eine k -DTM $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ **berechnet** eine Funktion $f : \Sigma^* \rightarrow \Gamma^*$, falls M bei jeder Eingabe $x \in \Sigma^*$ in einer Konfiguration

$$K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k) \text{ mit } u_k = f(x)$$

hält (d.h. $K_x \vdash^* K$ und K hat keine Folgekonfiguration)

- Hierfür sagen wir auch, M **gibt bei Eingabe x das Wort $f(x)$ aus** und schreiben $M(x) = f(x)$
- f heißt **Turing-berechenbar** (oder einfach **berechenbar**), falls es eine k -DTM M mit $M(x) = f(x)$ für alle $x \in \Sigma^*$ gibt
- Aus historischen Gründen werden berechenbare Funktionen auch **rekursiv** (engl. *recursive*) genannt

Definition

Für eine Sprache $A \subseteq \Sigma^*$ ist die **charakteristische Funktion** $\chi_A : \Sigma^* \rightarrow \{0, 1\}$ wie folgt definiert:

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

Bemerkung

- In den Übungen wird gezeigt, dass eine Sprache A genau dann entscheidbar ist, wenn χ_A berechenbar (also rekursiv) ist
- Dies erklärt die Bezeichnung REC für die Klasse der entscheidbaren Sprachen
- Dort wird auch gezeigt, dass CSL echt in REC enthalten ist
- Beispiele für interessante semi-entscheidbare Sprachen, die nicht entscheidbar sind, werden wir noch kennenlernen
- Somit gilt $\text{REG} \subsetneq \text{DCFL} \subsetneq \text{CFL} \subsetneq \text{DCSL} \subseteq \text{CSL} \subsetneq \text{REC} \subsetneq \text{RE}$

Berechenbarkeit von partiellen Funktionen

Definition

- Eine **partielle Funktion** hat die Form $f : A \rightarrow B \cup \{\uparrow\}$, wobei $\uparrow \notin B$ ist
- Für $f(x) = \uparrow$ sagen wir auch $f(x)$ ist **undefiniert**
- Der **Definitionsbereich** (engl. *domain*) von f ist

$$\text{dom}(f) = \{x \in A \mid f(x) \neq \uparrow\}$$

- Das **Bild** (engl. *image*) von f ist

$$\text{img}(f) = \{f(x) \mid x \in \text{dom}(f)\}$$

- Eine partielle Funktion $f : A \rightarrow B \cup \{\uparrow\}$ heißt **total**, falls $\text{dom}(f) = A$ ist
- Eine partielle Funktion $f : \Sigma^* \rightarrow \Gamma^* \cup \{\uparrow\}$ heißt **berechenbar**, falls es eine DTM M mit $M(x) = f(x)$ für alle $x \in \Sigma^*$ gibt (d.h. $M(x)$ gibt für alle $x \in \text{dom}(f)$ das Wort $f(x)$ aus und hält im Fall $x \notin \text{dom}(f)$ nicht)

- Jede DTM $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ ber. eine part. Fkt. $f : \Sigma^* \rightarrow \Gamma^* \cup \{\uparrow\}$
- Die Menge $\text{dom}(f) = \{x \in \Sigma^* \mid M(x) \downarrow\}$ bezeichnen wir mit **$\text{dom}(M)$**

Wir fassen die berechenbaren Funktionen und berechenbaren partiellen Funktionen in folgenden Klassen zusammen:

$$\mathbf{FREC} = \{f \mid f \text{ ist eine berechenbare (totale) Funktion}\}$$

$$\mathbf{FREC}_p = \{f \mid f \text{ ist eine berechenbare partielle Funktion}\}$$

Dann gilt $\mathbf{FREC} \not\subseteq \mathbf{FREC}_p$.

Berechenbarkeit von Funktionen

Beispiel

- Bezeichne x^+ den **lexikografischen Nachfolger** von $x \in \Sigma^*$
- Für $\Sigma = \{0, 1\}$ ergeben sich beispielsweise folgende Werte:

x	ε	0	1	00	01	10	11	000	...
x^+	0	1	00	01	10	11	000	001	...

- Betrachte die auf Σ^* definierten partiellen Funktionen f_1, f_2, f_3, f_4 mit

$$\begin{aligned}
 f_1(x) &= 0 \\
 f_2(x) &= x \quad \text{und} \quad f_4(x) = \begin{cases} \uparrow, & x = \varepsilon \\ y, & x = y^+ \end{cases} \\
 f_3(x) &= x^+
 \end{aligned}$$

- Da f_1, f_2, f_3, f_4 berechenbar sind, gehören die totalen Funktionen f_1, f_2, f_3 zu FREC und die partielle Funktion f_4 zu FREC_p
- Da f_4 keine totale Funktion ist, gehört f_4 nicht zu FREC



Definition

Sei $A \subseteq \Sigma^*$ eine Sprache.

- Die **partielle charakteristische Funktion** von A ist $\hat{\chi}_A : \Sigma^* \rightarrow \{1\} \cup \{\uparrow\}$ mit

$$\hat{\chi}_A(x) = \begin{cases} 1, & x \in A \\ \uparrow, & x \notin A \end{cases}$$

- A heißt **rekursiv aufzählbar**, falls $A = \emptyset$ oder das Bild $\text{img}(f)$ einer (totalen) berechenbaren Funktion $f : \Gamma^* \rightarrow \Sigma^*$ ist

Satz

Folgende Eigenschaften sind für eine Sprache $A \subseteq \Sigma^*$ äquivalent:

- 1 A ist semi-entscheidbar (d.h. A wird von einer DTM akzeptiert)
- 2 A wird von einer 1-DTM akzeptiert
- 3 A ist vom Typ 0
- 4 A wird von einer NTM akzeptiert
- 5 A ist rek. aufzählbar (d.h. $A = \emptyset$ oder $A = \text{img}(f)$ für eine Fkt. $f \in \text{FREC}$)
- 6 $\hat{\chi}_A$ ist berechenbar (d.h. $\hat{\chi}_A \in \text{FREC}_p$)
- 7 es gibt eine DTM M mit $A = \text{dom}(M)$

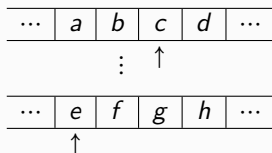
Beweis

Die Implikationen 2 \Rightarrow 3 \Rightarrow 4 werden in den Übungen gezeigt.

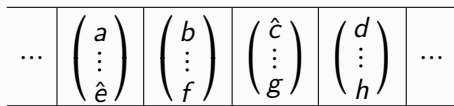
Hier zeigen wir 1 \Rightarrow 2 und 4 \Rightarrow 5 \Rightarrow 6 \Rightarrow 7 \Rightarrow 1.

Beweis von ① \Rightarrow ②: $\{L(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine 1-DTM}\}$

- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ eine k -DTM mit $L(M) = A$
- Wir konstruieren eine 1-DTM $M' = (Z', \Sigma, \Gamma', \delta', z_0, E)$ für A
- M' simuliert M , indem sie jede Konfiguration K von M der Form



durch eine Konfiguration K' folgender Form nachbildet:



Simulation einer k -DTM durch eine 1-DTM

Beweis von ① \Rightarrow ②: $\{L(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine 1-DTM}\}$

- Hierzu arbeitet M' mit dem Alphabet

$$\Gamma' = \Gamma \cup (\Gamma \cup \hat{\Gamma})^k, \text{ wobei } \hat{\Gamma} = \{\hat{a} \mid a \in \Gamma\} \text{ ist}$$

- Bei Eingabe $x = x_1 \dots x_n$ erzeugt M' zuerst die der Startkonfiguration

$$K_x = (q_0, \varepsilon, x_1, x_2 \dots x_n, \varepsilon, \sqcup, \varepsilon, \dots, \varepsilon, \sqcup, \varepsilon)$$

von M bei Eingabe x entsprechende Konfiguration

$$K'_x = q'_0 \begin{pmatrix} \hat{x}_1 \\ \hat{\sqcup} \\ \vdots \\ \hat{\sqcup} \end{pmatrix} \begin{pmatrix} x_2 \\ \sqcup \\ \vdots \\ \sqcup \end{pmatrix} \dots \begin{pmatrix} x_n \\ \sqcup \\ \vdots \\ \sqcup \end{pmatrix}$$

Simulation einer k -DTM durch eine 1-DTM

Beweis von ① \Rightarrow ②: $\{L(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine 1-DTM}\}$

- Dann simuliert M' jeweils einen Schritt von M durch folgende Sequenz von Rechenschritten:
 - Zuerst geht M' solange nach rechts, bis sie alle mit $\hat{}$ markierten Zeichen (z.B. $\hat{a}_1, \dots, \hat{a}_k$) gefunden hat
 - Diese Zeichen speichert M' in ihrem Zustand
 - Anschließend geht M' wieder nach links und realisiert dabei die durch $\delta(q, a_1, \dots, a_k)$ vorgegebene Anweisung von M
 - Dabei speichert M' den aktuellen Zustand q von M ebenfalls in ihrem Zustand
- Sobald M in einen Endzustand übergeht, wechselt M' ebenfalls in einen Endzustand und hält
- Somit gilt $L(M') = L(M)$ □

Beweis von ④ \Rightarrow ⑤: $\{L(M) \mid M \text{ ist eine NTM}\} \subseteq \{A \mid A \text{ ist rek. aufzählbar}\}$

- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ eine k -NTM und sei $A = L(M) \neq \emptyset$
- Sei $\tilde{\Gamma}$ das Alphabet $Z \cup \Gamma \cup \{\#\}$
- Wir kodieren eine Konfiguration $K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k)$ durch das Wort

$$\text{code}(K) = \#q\#u_1\#a_1\#v_1\#\dots\#u_k\#a_k\#v_k\#$$

und eine Rechnung $K_0 \vdash \dots \vdash K_t$ durch $\text{code}(K_0) \dots \text{code}(K_t)$

- Dann lassen sich die Wörter von A durch folgende Funktion $f : \tilde{\Gamma}^* \rightarrow \Sigma^*$ aufzählen (dabei ist x_0 ein beliebiges Wort in A):

$$f(w) = \begin{cases} x, & w \text{ kodiert eine akz. Rechnung } K_0 \vdash \dots \vdash K_t \text{ von} \\ & M(x), \text{ d.h. } K_0 = K_x \text{ und } K_t \in E \times (\Gamma^* \times \Gamma \times \Gamma^*)^k \\ x_0, & \text{sonst} \end{cases}$$

- Da f berechenbar ist, ist $A = \text{img}(f)$ rekursiv aufzählbar

Beweis von ⑤ \Rightarrow ⑥: $\{A \mid A \text{ ist rek. aufzählbar}\} \subseteq \{A \mid \hat{\chi}_A \in \text{FREC}_p\}$

- Sei M eine DTM, die eine Fkt. $f : \Gamma^* \rightarrow \Sigma^*$ mit $A = \text{img}(f)$ berechnet
- Dann wird $\hat{\chi}_A$ von der DTM M' berechnet, die bei Eingabe x
 - der Reihe nach für alle $w \in \Gamma^*$ das Wort $f(w)$ berechnet und
 - den Wert 1 ausgibt, sobald $f(w) = x$ ist

□

Beweis von ⑥ \Rightarrow ⑦: $\{A \mid \hat{\chi}_A \in \text{FREC}_p\} \subseteq \{\text{dom}(M) \mid M \text{ ist eine DTM}\}$

- Sei M eine DTM, die $\hat{\chi}_A$ berechnet
- Da $\text{dom}(\hat{\chi}_A) = A$ ist, folgt $A = \text{dom}(M)$

□

Beweis von ⑦ \Rightarrow ①: $\{\text{dom}(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine DTM}\}$

- Sei $A = \text{dom}(M)$ für eine DTM M
- Dann gilt $A = L(M')$ für die DTM M' , die M simuliert und genau dann in einen Endzustand übergeht, wenn M hält

□

Satz

Folgende Eigenschaften sind äquivalent:

- 1 A ist entscheidbar (d.h. A wird von einer DTM akzeptiert, die alle Eingaben entscheidet)
- 2 die charakteristische Funktion χ_A von A ist berechenbar
- 3 A wird von einer 1-DTM akzeptiert, die bei allen Eingaben hält
- 4 A wird von einer NTM akzeptiert, die bei allen Eingaben hält
- 5 A und \bar{A} sind semi-entscheidbar

Beweis

Die Äquivalenz der Bedingungen 1 bis 4 wird in den Übungen gezeigt. Hier zeigen wir nur die Äquivalenz dieser vier Bedingungen zu 5.

Beweis von ① \Rightarrow ⑤: $REC \subseteq RE \cap co-RE$

- Falls A entscheidbar ist, ist mit χ_A auch $\chi_{\bar{A}}$ berechenbar, d.h. A und \bar{A} sind entscheidbar und damit auch semi-entscheidbar

Beweis von ⑤ \Rightarrow ①: $RE \cap co-RE \subseteq REC$

- Seien M_A und $M_{\bar{A}}$ DTMs, die $\hat{\chi}_A$ und $\hat{\chi}_{\bar{A}}$ berechnen
- Betrachte die DTM M , die abwechselnd M_A und $M_{\bar{A}}$ für jeweils einen weiteren Rechenschritt simuliert und
 - in einem Endzustand hält, sobald $M_A(x)$ hält, sowie
 - in einem Nichtendzustand hält, sobald $M_{\bar{A}}(x)$ hält
- Da jede Eingabe x entweder in $dom(\hat{\chi}_A) = A$ oder in $dom(\hat{\chi}_{\bar{A}}) = \bar{A}$ enthalten ist, hält M bei allen Eingaben
- Da zudem $L(M) = A$ ist, folgt $A \in REC$

□

Kodierung (Gödelisierung) von Turingmaschinen

- Um Eigenschaften von TMs algorithmisch untersuchen zu können, müssen wir TMs als Teil der Eingabe kodieren
- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ eine 1-DTM mit
 - Zustandsmenge $Z = \{q_0, \dots, q_m\}$ (o.B.d.A. sei $E = \{q_m\}$),
 - Eingabealphabet $\Sigma = \{0, 1\}$ und
 - Arbeitsalphabet $\Gamma = \{a_0, \dots, a_l\}$,
wobei wir o.B.d.A. $a_0 = 0$, $a_1 = 1$ und $a_2 = \sqcup$ annehmen
- Dann kodieren wir eine Anweisung $q_i a_j \rightarrow q_{i'} a_{j'} D$ durch das Wort

$$\#bin(i)\#bin(j)\#bin(i')\#bin(j')\#b_D\#$$
- Dabei ist $bin(n)$ die Binärdarstellung von n und

$$b_D = \begin{cases} 0, & D = N \\ 1 & D = L \\ 10, & D = R \end{cases}$$

Kodierung von Turingmaschinen

- M lässt sich nun als ein Wort über dem Alphabet $\{0, 1, \#\}$ kodieren, indem wir die Anweisungen von M in kodierter Form auflisten
- Kodieren wir die Zeichen $0, 1, \#$ binär (z.B. $0 \mapsto 00, 1 \mapsto 01, \# \mapsto 10$), so gelangen wir zu einer Binärokodierung w_M von M
- Die durch die Binärzahl $w_M = b_n \dots b_0$ repräsentierte natürliche Zahl $(w_M)_2 = \sum_{i=0}^n b_i 2^i$ wird auch die **Gödel-Nummer** von M genannt
- M_w ist durch Angabe von w_M bzw. $(w_M)_2$ bis auf die Benennung ihrer Zustände und der Arbeitszeichen in $\Gamma \setminus \{\sqcup, 0, 1\}$ eindeutig bestimmt
- Ganz analog lassen sich auch k -DTMs mit $k > 1$ sowie NTMs binär kodieren
- Umgekehrt können wir jedem Binärstring $w \in \{0, 1\}^*$ eine DTM M_w wie folgt zuordnen (dabei ist M_0 eine beliebige, aber fest gewählte DTM):

$$M_w = \begin{cases} M, & \text{falls eine DTM } M \text{ mit } w_M = w \text{ existiert} \\ M_0, & \text{sonst} \end{cases}$$

Unentscheidbarkeit des Halteproblems

Definition

Das **Halteproblem** ist die Sprache

$$H = \left\{ w \# x \mid \begin{array}{l} w, x \in \{0, 1\}^* \text{ und} \\ \text{die DTM } M_w \text{ h\u00e4lt} \\ \text{bei Eingabe } x \end{array} \right\}$$

und das **spezielle Halteproblem** ist

$$K = \left\{ w \in \{0, 1\}^* \mid \begin{array}{l} \text{die DTM } M_w \\ \text{h\u00e4lt bei Eingabe } w \end{array} \right\}$$

χ_H	w_1	w_2	w_3	\dots
w_1	0	1	0	\dots
w_2	0	1	1	\dots
w_3	1	1	0	\dots
\vdots	\vdots	\vdots	\vdots	\ddots

χ_K				
w_1	0			
w_2		1		
w_3			0	
\vdots				\ddots

Satz

$K, H \in \text{RE}$.

Beweis von $K, H \in RE$

- Sei w_h die Kodierung einer DTM, die sofort hält, und betrachte die Funktionen $f_K : \{0, 1\}^* \rightarrow \{0, 1\}^*$ und $g_H : \{0, 1, \#\}^* \rightarrow \{0, 1, \#\}^*$ mit

$$f_K(x) = \begin{cases} w, & x \text{ ist die Binärkodierung einer haltenden Rechnung} \\ & \text{einer DTM } M_w \text{ bei Eingabe } w, \\ w_h, & \text{sonst} \end{cases}$$

und

$$g_H(x) = \begin{cases} w\#w', & x = w\#x' \text{ und } x' \text{ ist die Binärkodierung einer hal-} \\ & \text{tenden Rechnung der DTM } M_w \text{ bei Eingabe } w', \\ w_h\#\varepsilon, & \text{sonst} \end{cases}$$

- Da f_K und g_H in FREC sind und $\text{img}(f_K) = K$ sowie $\text{img}(g_H) = H$ ist, folgt $K, H \in RE$ □

Unentscheidbarkeit des speziellen Halteproblems

Satz

$K \notin \text{RE}$ und somit $\text{REC} \not\subseteq \text{RE} \neq \text{co-RE}$.

Beweisidee

- Sei $B = (b_{ij})$ die durch $b_{ij} = \chi_H(w_i \# w_j) \in \{0, 1\}$ definierte Binärmatrix
- Dann kann keine Zeile $b_{i1} b_{i2} \dots$ von B mit der invertierten Diagonalen $\bar{b}_{11} \bar{b}_{22} \dots$ von B übereinstimmen, da sonst $b_{ii} = \bar{b}_{ii}$ sein müsste
- Da die i -te Zeile von B wegen

$$b_{ij} = \chi_H(w_i \# w_j) = \chi_{\text{dom}(M_{w_i})}(w_j)$$

die Binärsprache $\text{dom}(M_{w_i}) \in \text{RE}$ und die invertierte Diagonale wegen

$$\bar{b}_{ii} = \chi_{\bar{H}(w_i \# w_i)} = \chi_{\bar{K}}(w_i)$$

die Sprache \bar{K} repräsentiert, folgt $\bar{K} \neq \text{dom}(M_{w_i})$ für alle $i \geq 1$

- Dies impliziert $\bar{K} \notin \text{RE}$, da die Gesamtheit der Zeilen von B wegen

$$\{\text{dom}(M_{w_i}) \mid i \geq 1\} = \{A \subseteq \{0, 1\}^* \mid A \in \text{RE}\}$$

die Klasse aller Binärsprachen in RE repräsentiert

Beweis von $\bar{K} \notin \text{RE}$

- Angenommen, die Sprache

$$\bar{K} = \{w \in \{0,1\}^* \mid M_w(w) \uparrow\} \quad (*)$$

wäre semi-entscheidbar

- Dann existiert eine DTM M_{w_i} mit

$$\text{dom}(M_{w_i}) = \bar{K} \quad (**)$$

- Dies führt jedoch auf einen Widerspruch:

$$w_i \in \bar{K} \stackrel{(*)}{\Leftrightarrow} M_{w_i}(w_i) \uparrow \Leftrightarrow w_i \notin \text{dom}(M_{w_i}) \stackrel{(**)}{\Leftrightarrow} w_i \notin \bar{K} \quad \text{⚡}$$

□

χ_H	w_1	w_2	w_3	...
w_1	0	1	0	...
w_2	0	1	1	...
w_3	1	1	0	...
\vdots	\vdots	\vdots	\vdots	
w_i	1	0	1	...
\vdots	\vdots	\vdots	\vdots	

Bemerkung

- Die Methode in obigem Beweis wird als **Diagonalisierung** bezeichnet
- Mit dieser Beweistechnik lässt sich auch eine Sprache in $\text{REC} \setminus \text{CSL}$ definieren (siehe Übungen)

Definition

Eine Sprache $A \subseteq \Sigma^*$ heißt auf $B \subseteq \Gamma^*$ **reduzierbar** (kurz: $A \leq B$), falls eine berechenbare Funktion $f : \Sigma^* \rightarrow \Gamma^*$ ex., so dass gilt:

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B$$

Beispiel

- Es gilt $K \leq H$ mittels $f : w \mapsto w\#w$, da für alle $w \in \{0, 1\}^*$ gilt:

$$w \in K \Leftrightarrow M_w(w) \downarrow \Leftrightarrow w\#w \in H$$

- Es gilt sogar $A \leq H$ für jede Binärsprache $A \in \text{RE}$ mittels $f : x \mapsto w\#x$, wobei w die Kodierung einer DTM M_w mit $\text{dom}(M_w) = A$ ist:

$$x \in A \Leftrightarrow M_w(x) \downarrow \Leftrightarrow w\#x \in H$$



Der Vollständigkeitsbegriff

Definition

- Eine Sprache B heißt **hart** für eine Sprachklasse \mathcal{C} (kurz: **\mathcal{C} -hart** oder **\mathcal{C} -schwer**), falls jede Sprache $A \in \mathcal{C}$ auf B reduzierbar ist:

$$\forall A \in \mathcal{C} : A \leq B$$

- Eine \mathcal{C} -harte Sprache B , die zu \mathcal{C} gehört, heißt **\mathcal{C} -vollständig**

Beispiel (H ist hart für $\{A \subseteq \{0,1\}^* \mid A \in \text{RE}\}$ und vollständig für RE)

- Wir wissen bereits, dass alle Binärsprachen in RE auf H reduzierbar sind
- Sei nun $A \in \text{RE}$ eine Sprache über einem bel. Alphabet
- Dann ist A auf die Binärsprache $A' = \{\text{bin}(x) \mid x \in A\}$ mittels der Fkt. $x \mapsto \text{bin}(x)$ reduzierbar und mit A ist auch A' in RE
- Somit folgt $A \leq A' \leq H$, was wg. der Transitivität von \leq (s. Übungen) $A \leq H$ impliziert



Abgeschlossenheit von REC unter \leq

Definition

Eine Sprachklasse \mathcal{C} heißt **unter \leq abgeschlossen**, wenn für beliebige Sprachen A, B gilt:

$$A \leq B \wedge B \in \mathcal{C} \Rightarrow A \in \mathcal{C}$$

Satz

Die Klasse REC ist unter \leq abgeschlossen.

Beweis

- Gelte $A \leq B$ mittels f und sei $B \in \text{REC}$
- Wegen $B \in \text{REC}$ ex. eine DTM M_B , die χ_B berechnet
- Betrachte folgende DTM M_A :
 - M_A berechnet bei Eingabe x zuerst den Wert $f(x)$ und
 - simuliert dann M_B bei Eingabe $f(x)$

Abgeschlossenheit von REC und RE unter \leq

Satz

Die Klasse REC ist unter \leq abgeschlossen.

Beweis.

- Gelte $A \leq B$ mittels f und sei $B \in \text{REC}$
- Dann ex. eine DTM M_B , die χ_B berechnet
- Betrachte folgende DTM M_A :
 - M_A berechnet bei Eingabe x zuerst den Wert $f(x)$ und
 - simuliert dann M_B bei Eingabe $f(x)$
- Wegen $x \in A \Leftrightarrow f(x) \in B$ ist $\chi_A(x) = \chi_B(f(x))$ und daher folgt

$$M_A(x) = M_B(f(x)) = \chi_B(f(x)) = \chi_A(x)$$
- Also berechnet M_A die Funktion χ_A , d.h. $A \in \text{REC}$ □

Bemerkung

Die Abgeschlossenheit von RE unter \leq folgt analog (siehe Übungen).

H ist nicht entscheidbar

Korollar

- $A \leq B \wedge A \notin \text{REC} \Rightarrow B \notin \text{REC}$
- $A \leq B \wedge A \notin \text{RE} \Rightarrow B \notin \text{RE}$

Beweis

Aus der Annahme, dass B entscheidbar (bzw. semi-entscheidbar) ist, folgt wegen $A \leq B$, dass dies auch auf A zutrifft (Widerspruch). \square

Bemerkung

Wegen $K \leq H$ überträgt sich somit die Unentscheidbarkeit von K auf H .

Korollar

$H \notin \text{REC}$, d.h. H ist unentscheidbar.

Das Halteproblem bei leerem Band

Definition

Das **Halteproblem bei leerem Band** ist die Sprache

$$H_0 = \left\{ w \in \{0, 1\}^* \mid \begin{array}{l} \text{die DTM } M_w \\ \text{hält bei Eingabe } \varepsilon \end{array} \right\}$$

Satz

H_0 ist RE-vollständig und somit unentscheidbar.

Beweis

- $H_0 \in \text{RE}$ folgt wegen $H_0 \leq H \in \text{RE}$ mittels der Reduktionsfunktion $w \mapsto w\#\varepsilon$

χ_H	w_1	w_2	w_3	\dots
w_1	0	1	0	\dots
w_2	0	1	1	\dots
w_3	1	1	0	\dots
\vdots	\vdots	\vdots	\vdots	\ddots

χ_{H_0}	$w_1 (= \varepsilon)$
w_1	0
w_2	0
w_3	1
\vdots	\vdots

H_0 ist RE-vollständig

Beweis

- $H_0 \in \text{RE}$ folgt wegen $H_0 \leq H \in \text{RE}$ mittels der Funktion $w \mapsto w\#\varepsilon$
- Sei $A \in \text{RE}$ und sei M_A eine DTM mit $\text{dom}(M_A) = A$
- Da jede Sprache $A \in \text{RE}$ auf eine Binärsprache in RE reduzierbar ist, können wir o.B.d.A. $A \subseteq \{0, 1\}^*$ annehmen
- Um A auf H_0 zu reduzieren, transformieren wir x in die Kodierung w_x einer DTM M_{w_x} , die ihre Eingabe ignoriert und $M_A(x)$ simuliert
- Dann gilt

$$x \in A \Leftrightarrow M_A(x) \downarrow \Leftrightarrow M_{w_x}(\varepsilon) \downarrow \Leftrightarrow w_x \in H_0$$

- Da zudem die Funktion $x \mapsto w_x$ total und berechenbar ist, folgt $A \leq H_0 \square$

Bemerkung

Derselbe Beweis zeigt, dass alle Spalten und auch die Diagonale der Matrix $B = (b_{ij})$ mit $b_{ij} = \chi_H(w_i\#w_j)$ RE-vollständige Sprachen repräsentieren.

- Oft möchte man wissen, ob die von einer DTM akz. Sprache (bzw. die von ihr ber. partielle Funktion) eine gewisse Eigenschaft hat oder nicht
- Solche Eigenschaften werden **semantisch** genannt, da sie nur vom Akzeptanz- bzw. Ein/Ausgabeverhalten der geg. DTM abhängen
- Der Satz von Rice besagt, dass so gut wie alle semantischen Eigenschaften von Turingmaschinen unentscheidbar sind
- Zum Beispiel ist das Problem unentscheidbar, ob eine gegebene DTM bei allen Eingaben hält (also eine totale Funktion berechnet)
- Formal lassen sich semantische Eigenschaften durch eine Menge \mathcal{F} von partiellen Funktionen bzw. durch eine Sprachklasse \mathcal{S} beschreiben
- Eine DTM M_w hat dann die Eigenschaft \mathcal{F} (bzw. \mathcal{S}), wenn sie eine Funktion in \mathcal{F} berechnet (bzw. eine Sprache in \mathcal{S} akzeptiert)
- Eine semantische Eigenschaft \mathcal{F} (bzw. \mathcal{S}) heißt **trivial**, falls entweder alle DTMs oder keine diese Eigenschaft haben

Definition

- Zu einer Klasse \mathcal{F} von partiellen Funktionen definieren wir die Sprache

$$L_{\mathcal{F}} = \{w \in \{0,1\}^* \mid \text{die DTM } M_w \text{ ber. eine partielle Fkt. in } \mathcal{F}\}$$

- Die Eigenschaft \mathcal{F} heißt **trivial**, wenn $L_{\mathcal{F}} = \emptyset$ oder $L_{\mathcal{F}} = \{0,1\}^*$ ist

Der Satz von Rice besagt, dass jede semantische Eigenschaft von DTMs entweder trivial oder unentscheidbar ist.

Satz (Satz von Rice)

Für jede nicht triviale Eigenschaft \mathcal{F} ist die Sprache $L_{\mathcal{F}}$ unentscheidbar.

Beispiel

- Betrachte die beiden Sprachen

$$L_1 = \{w \in \{0, 1\}^* \mid M_w(0^n) = 0^{n+1} \text{ für alle } n \geq 0\} \text{ und}$$

$$L_2 = \{w \in \{0, 1\}^* \mid M_w(x) = \hat{\chi}_K(x) \text{ für alle } x \in \{0, 1\}^*\}$$

- Dann gilt $L_i = L_{\mathcal{F}_i}$ für die Eigenschaften

$$\mathcal{F}_1 = \{f \in \text{FREC}_p \mid f(0^n) = 0^{n+1} \text{ für alle } n \geq 0\} \text{ und}$$

$$\mathcal{F}_2 = \{f \in \text{FREC}_p \mid f(x) = \hat{\chi}_K(x) \text{ für alle } x \in \{0, 1\}^*\}$$

- \mathcal{F}_1 ist nicht trivial, da die partiellen Fkten $f, u : \{0, 1\}^* \rightarrow \{0, 1\}^* \cup \{\uparrow\}$ mit $f(x) = x0$ und $u(x) = \uparrow$ für alle $x \in \{0, 1\}^*$ berechenbar sind und $f \in \mathcal{F}_1$ sowie $u \notin \mathcal{F}_1$ ist
- Auch \mathcal{F}_2 ist nicht trivial, da $\hat{\chi}_K$ wegen $K \in \text{RE}$ eine berechenbare partielle Funktion in \mathcal{F}_2 ist, während $f, u \notin \mathcal{F}_2$ sind
- Also sind die beiden Sprachen L_1 und L_2 nach dem Satz von Rice unentscheidbar

Beispiel (Fortsetzung)

- Dagegen liefert der Satz von Rice nicht die Unentscheidbarkeit folgender Sprachen:

$$L_4 = \{w \in \{0, 1\}^* \mid M_w(x) = \hat{\chi}_{\bar{K}}(x) \text{ für alle } x \in \{0, 1\}^*\} \text{ und}$$

$$L_5 = \{w \in \{0, 1\}^* \mid M_w(0^n) \text{ hält für alle } n \geq 0 \text{ nach } n \text{ Schritten}\}$$

- Es gilt $L_4 = L_{\mathcal{F}_4}$ für die Eigenschaft $\mathcal{F}_4 = \{\hat{\chi}_{\bar{K}}\}$, d.h. L_4 beschreibt zwar eine semantische Eigenschaft von DTMs, die sich nur auf deren Ein-/Ausgabeverhalten bezieht
- Da aber $\bar{K} \notin \text{RE}$ und somit $\hat{\chi}_{\bar{K}}$ nicht berechenbar ist, handelt es sich bei \mathcal{F}_4 um eine triviale Eigenschaft: $L_4 = L_{\mathcal{F}_4} = \emptyset$
- Die Sprache L_5 bezieht sich nicht nur auf das Ein-/Ausgabeverhalten von DTMs, sondern auch auf deren Laufzeit
- Daher existiert für L_5 keine Funktionenklasse \mathcal{F} mit $L_5 = L_{\mathcal{F}}$

Der Satz von Rice

Satz (Satz von Rice)

Für jede nicht triviale Eigenschaft \mathcal{F} ist die Sprache $L_{\mathcal{F}}$ unentscheidbar.

Beweisidee

- Die Idee besteht darin, H_0 auf $L_{\mathcal{F}}$ (oder auf $\bar{L}_{\mathcal{F}}$) zu reduzieren, indem wir für eine gegebene DTM M_w eine DTM $M_{w'}$ konstruieren mit

$$w \in H_0 \Leftrightarrow M_{w'} \text{ berechnet (k)eine partielle Funktion in } \mathcal{F}$$
- Hierzu lassen wir $M_{w'}$ bei Eingabe x zunächst einmal die DTM M_w bei Eingabe ε simulieren
- Falls $w \notin H_0$ ist, berechnet $M_{w'}$ also die überall undefinierte Funktion $u(x) = \uparrow$ für alle $x \in \{0, 1\}^*$
- Damit die Reduktion gelingt, müssen wir nur noch dafür sorgen, dass $M_{w'}$ im Fall $w \in H_0$ eine partielle Funktion f berechnet, die sich bzgl. der Eigenschaft \mathcal{F} von u unterscheidet, d.h. $f \in \mathcal{F} \Leftrightarrow u \notin \mathcal{F}$
- Da \mathcal{F} nicht trivial ist, ex. eine DTM M , die ein solches f berechnet

Der Satz von Rice

Satz (Satz von Rice)

Für jede nicht triviale Eigenschaft \mathcal{F} ist die Sprache $L_{\mathcal{F}}$ unentscheidbar.

Beweis

- Sei M eine DTM, die eine Funktion f mit $f \in \mathcal{F} \Leftrightarrow u \notin \mathcal{F}$ berechnet
- Betrachte die Reduktionsfunktion

$h(w) = w'$, wobei w' die Kodierung einer DTM ist, die bei Eingabe x zunächst die DTM $M_w(\varepsilon)$ simuliert und im Fall, dass $M_w(\varepsilon)$ hält, mit der Simulation von $M(x)$ fortfährt
- Dann ist $h : w \mapsto w'$ eine totale berechenbare Funktion und es gilt

$w \in H_0 \Rightarrow M_{w'} \text{ berechnet } f$

$w \notin H_0 \Rightarrow M_{w'} \text{ berechnet } u$
- Dies zeigt, dass h das Problem H_0 auf $L_{\mathcal{F}}$ (oder auf $\bar{L}_{\mathcal{F}}$) reduziert, und da H_0 unentscheidbar ist, muss auch $L_{\mathcal{F}}$ unentscheidbar sein □

Der Satz von Rice gilt auch für Eigenschaften, die das Akzeptanzverhalten einer gegebenen Turingmaschine betreffen.

Satz (Satz von Rice für Spracheigenschaften)

Für eine beliebige Sprachklasse \mathcal{S} sei

$$L_{\mathcal{S}} = \{w \in \{0, 1\}^* \mid L(M_w) \in \mathcal{S}\}.$$

Dann ist $L_{\mathcal{S}}$ unentscheidbar, außer wenn $L_{\mathcal{S}} = \emptyset$ oder $L_{\mathcal{S}} = \{0, 1\}^*$ ist.

Beweis

Siehe Übungen.

Entscheidungsprobleme für Sprachklassen

Neben dem Wortproblem sind für eine Sprachklasse \mathcal{C} auch folgende Entscheidungsprobleme interessant:

Das Leerheitsproblem ($LP_{\mathcal{C}}$)

Gegeben: Eine Sprache L aus \mathcal{C} .

Gefragt: Ist $L \neq \emptyset$?

Das Äquivalenzproblem ($\ddot{A}P_{\mathcal{C}}$)

Gegeben: Zwei Sprachen L_1 und L_2 aus \mathcal{C} .

Gefragt: Gilt $L_1 = L_2$?

Das Schnittproblem ($SP_{\mathcal{C}}$)

Gegeben: Zwei Sprachen L_1 und L_2 aus \mathcal{C} .

Gefragt: Ist $L_1 \cap L_2 \neq \emptyset$?

Hierbei repräsentieren wir Sprachen in $\mathcal{C} = \text{REG}, \text{CFL}, \text{CSL}, \text{RE}$ durch entsprechende Grammatiken und Sprachen in $\mathcal{C} = \text{DCFL}, \text{DCSL}$ durch entsprechende Akzeptoren (also DPDAs bzw. DLBAs).

Das Postsche Korrespondenzproblem

Definition

- Sei Σ ein beliebiges Alphabet mit $\# \notin \Sigma$
- Das **Postsche Korrespondenzproblem über Σ** (kurz **PCP $_{\Sigma}$**) ist:
 gegeben: n Wortpaare $(x_1, y_1), \dots, (x_n, y_n) \in \Sigma^+ \times \Sigma^+$
 gefragt: Gibt es eine Folge $s = (i_1, \dots, i_k)$ von $k \geq 1$ Indizes
 $i_j \in \{1, \dots, n\}$ mit $x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$?
- Das **modifizierte PCP über Σ** (kurz **MPCP $_{\Sigma}$**) fragt nach einer Lösung
 $s = (i_1, \dots, i_k)$ mit $i_1 = 1$
- Wir notieren eine PCP-Instanz meist in Form einer Matrix $\begin{pmatrix} x_1 \dots x_n \\ y_1 \dots y_n \end{pmatrix}$ und kodieren sie durch das Wort $x_1 \# y_1 \# \dots \# x_n \# y_n$

Beispiel

Die Instanz $I = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} = \begin{pmatrix} a & ab & caa \\ aca & bc & aa \end{pmatrix}$ besitzt wegen

$$x_1 x_3 x_2 x_3 = acaabcaa$$

$$y_1 y_3 y_2 y_3 = acaabcaa$$

die PCP-Lösung $s = (1, 3, 2, 3)$, die auch eine MPCP-Lösung ist

Das binäre PCP

Lemma

Für jedes Alphabet Σ gilt $\text{PCP}_{\Sigma} \leq \text{PCP}_{\{0,1\}}$.

Beweis

- Sei $\Sigma = \{a_0, \dots, a_{m-1}\}$ und sei $r = \max(1, \lceil \log_2(m) \rceil)$
- Wir kodieren a_i durch eine r -stellige Binärzahl $\text{bin}_r(a_i)$ mit dem Wert i und ein Wort $w = w_1 \dots w_\ell \in \Sigma^\ell$ durch $\text{bin}(w) = \text{bin}_r(w_1) \dots \text{bin}_r(w_\ell)$
- Nun folgt $\text{PCP}_{\Sigma} \leq \text{PCP}_{\{0,1\}}$ mittels der Reduktionsfunktion

$$f : \begin{pmatrix} x_1 \dots x_n \\ y_1 \dots y_n \end{pmatrix} \mapsto \begin{pmatrix} \text{bin}(x_1) \dots \text{bin}(x_n) \\ \text{bin}(y_1) \dots \text{bin}(y_n) \end{pmatrix}$$

□

Beispiel

Sei $\Sigma = \{a, b, c\}$. Dann ist $r = \max(1, \lceil \log_2(3) \rceil) = 2$ und $\text{bin}_2(a) = 00$, $\text{bin}_2(b) = 01$ und $\text{bin}_2(c) = 10$. Somit ist

$$f \begin{pmatrix} a & ab & caa \\ aca & bc & aa \end{pmatrix} = \begin{pmatrix} 00 & 0001 & 100000 \\ 001000 & 0110 & 0000 \end{pmatrix}$$

◀

Reduktion des MPCP auf das PCP

Wir schreiben für $\text{PCP}_{\{0,1\}}$ auch **PCP** (bzw. **MPCP** für $\text{MPCP}_{\{0,1\}}$).

Satz

$\text{MPCP} \leq \text{PCP}$.

Beweis

- Wir zeigen $\text{MPCP} \leq \text{PCP}_{\Sigma}$ für $\Sigma = \{0, 1, \langle, |, \rangle\}$
- Für ein Wort $w = w_1 \dots w_\ell \in \{0, 1\}^\ell$ sei

$$\begin{array}{cccc} \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} \\ \hline \langle w_1 | \dots | w_\ell \rangle & \langle w_1 | \dots | w_\ell \rangle & | w_1 | \dots | w_\ell | & w_1 | \dots | w_\ell \rangle \end{array}$$

- Wir reduzieren MPCP mittels folgender Funktion f auf PCP_{Σ} :

$$f : \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overleftarrow{x_1} & \dots & \overleftarrow{x_n} & \rangle \\ \overleftarrow{y_1} & \overleftarrow{y_1} & \dots & \overleftarrow{y_n} & | \rangle \end{pmatrix}$$

Reduktion des MPCP auf das PCP

Beweis

- Wir reduzieren MPCP mittels folgender Funktion f auf PCP_Σ :

$$f : \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overleftarrow{x_1} & \dots & \overleftarrow{x_n} & \rangle \\ \overleftarrow{y_1} & \overleftarrow{y_1} & \dots & \overleftarrow{y_n} & | \rangle \end{pmatrix}$$

Beispiel

- Betrachte die Reduktion der MPCP-Instanz

$$I = \begin{pmatrix} 00 & 1 & 101 & 11 \\ 001 & 11 & 0 & 1 \end{pmatrix} \text{ auf } f(I) = \begin{pmatrix} \langle |0|0| & 0|0| & 1| & 1|0|1| & 1|1| \rangle \\ \langle |0|0|1 & |0|0|1 & |1|1 & |0 & |1 \rangle \end{pmatrix}$$

- Dann entspricht der MPCP-Lösung $s = (1, 3, 2)$ mit dem Lösungswort

$$x_1 x_3 x_2 = 001011 = 001011 = y_1 y_3 y_2$$

für I die PCP-Lösung $s' = (1, 4, 3, 6)$ mit dem Lösungswort

$$\langle \overleftarrow{x_1} \overleftarrow{x_3} \overleftarrow{x_2} \rangle = \langle |0|0|1|0|1|1| \rangle = \langle |0|0|1|0|1|1| \rangle = \langle \overleftarrow{y_1} \overleftarrow{y_3} \overleftarrow{y_2} | \rangle$$

für $f(I)$ und umgekehrt



Beweis von $\text{MPCP} \leq \text{PCP}$

- Wir reduzieren MPCP mittels folgender Funktion f auf PCP_Σ :

$$f : \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overleftarrow{x_1} & \dots & \overleftarrow{x_n} & \rangle \\ \overleftarrow{y_1} & \overleftarrow{y_1} & \dots & \overleftarrow{y_n} & | \rangle \end{pmatrix} = \begin{pmatrix} x'_1 & x'_2 & \dots & x'_{n+2} \\ y'_1 & y'_2 & \dots & y'_{n+2} \end{pmatrix}$$

- Da jede MPCP-Lösung $s = (1, i_2, \dots, i_k)$ für I auf eine PCP-Lösung $s' = (1, i_2 + 1, \dots, i_k + 1, n + 2)$ für $f(I)$ führt, folgt

$$I \in \text{MPCP} \Rightarrow f(I) \in \text{PCP}_\Sigma$$

- Für die Rückrichtung sei $s' = (i_1, \dots, i_k)$ eine PCP-Lösung für $f(I)$
- Dann muss $i_1 = 1$ sein, da nur die beiden Einträge $x'_1 = \overleftarrow{x_1}$ und $y'_1 = \overleftarrow{y_1}$ in der ersten Spalte von $f(I)$ mit dem gleichen Zeichen beginnen
- Zudem muss $i_k = n + 2$ sein, da nur die beiden Einträge $x'_{n+2} = \rangle$ und $y'_{n+2} = | \rangle$ in der letzten Spalte von $f(I)$ mit dem gleichen Zeichen enden
- Wählen wir die Lösungsfolge s' von minimaler Länge, so gilt zudem $i_j \in \{2, \dots, n + 1\}$ für $j = 2, \dots, k - 1$
- Folglich ist $s = (i_1, i_2 - 1, \dots, i_{k-1} - 1)$ eine MPCP-Lösung für I □

Satz

PCP ist RE-vollständig und damit unentscheidbar.

Beweis.

- PCP ist semi-entscheidbar, da eine DTM systematisch nach einer Lösung suchen kann
- Um zu zeigen, dass PCP RE-hart ist, sei A eine beliebige Sprache in RE und sei $G = (V, \Sigma, P, S)$ eine Typ-0 Grammatik für A
- Wir zeigen $A \leq \text{MPCP}_\Gamma$ für $\Gamma = V \cup \Sigma \cup \{ \langle, |, \rangle \}$
- Wegen $\text{MPCP}_\Gamma \leq \text{PCP}$ folgt hieraus $A \leq \text{PCP}$

Beweis (Fortsetzung)

- Sei $G = (V, \Sigma, P, S)$ eine Typ-0 Grammatik für A
- Wir zeigen $A \leq \text{MPCP}_\Gamma$ für $\Gamma = V \cup \Sigma \cup \{\langle, |\rangle\}$
- Idee: Transformiere $w \in \Sigma^*$ in eine Instanz $f(w) = \begin{pmatrix} x_1 \dots x_n \\ y_1 \dots y_n \end{pmatrix}$, so dass für jede Indexfolge $s = (i_1, \dots, i_k)$ gilt:
 - s ist genau dann eine MPCP-Lösung für $f(w)$, wenn das zugehörige Lösungswort $z = x_{i_1} \dots x_{i_k} = y_{i_1} \dots y_{i_k}$ eine Ableitung von w in G kodiert, d.h. z hat die Form $z = \langle | \alpha_0 | \alpha_1 | \dots | \alpha_m | \rangle$ und es gilt

$$S = \alpha_0 \Rightarrow^* \alpha_1 \Rightarrow^* \dots \Rightarrow^* \alpha_m = w$$
- Konkret bilden wir $f(w)$ aus folgenden Wortpaaren:
 - $(\langle, \langle | S)$ „Startpaar“
 - für jede Regel $l \rightarrow r$ in P : (l, r) „Ableitungspaare“
 - für alle $a \in V \cup \Sigma \cup \{|\}$: (a, a) „Kopierpaare“
 - sowie das Paar $(w | \rangle, \rangle)$ „Abschlusspaar“

Unentscheidbarkeit des PCP

Beispiel

- Sei $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, \varepsilon\}, S)$ und $w = aabb$
- Die MPCP-Instanz $f(aabb)$ enthält dann die acht Wortpaare

$$f(aabb) = \left(\begin{array}{c} \langle \quad S \quad S \quad S \quad a \quad b \quad | \quad aabb| \rangle \\ \langle |S \quad aSbS \quad \varepsilon \quad S \quad a \quad b \quad | \quad \quad \rangle \end{array} \right)$$

- Der Ableitung $\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$ entspricht dann die MPCP-Lösung

$$(1, 7, 2, 7, 5, 2, 6, 4, 7, 5, 5, 4, 6, 3, 6, 4, 7, 5, 5, 3, 6, 6, 4, 7, 5, 5, 6, 6, 3, 7, 8)$$

mit dem Lösungswort

$$\begin{array}{l} \langle |S|aSbS|aaSbSbS|aaSbbS|aabbS|aabb| \rangle \\ \langle |S|aSbS|aaSbSbS|aaSbbS|aabbS|aabb| \rangle \end{array}$$

- Das kürzeste MPCP-Lösungswort für $f(aabb)$ ist

$$\begin{array}{l} \langle |S|aSbS|aaSbSb|aabb| \rangle \\ \langle |S|aSbS|aaSbSb|aabb| \rangle \end{array}$$

- Dieses entspricht der „parallelisierten“ Ableitung

$$\underline{S} \Rightarrow a\underline{S}b\underline{S} \Rightarrow^2 aa\underline{S}b\underline{S}b \Rightarrow^2 aabb$$

Beweis (Schluss)

- Wir bilden $f(w)$ aus folgenden Wortpaaren:
 - $(\langle, \langle | S)$ „Startpaar“
 - für jede Regel $l \rightarrow r$ in P : (l, r) „Ableitungspaare“
 - für alle $a \in V \cup \Sigma \cup \{\mid\}$: (a, a) „Kopierpaare“
 - sowie das Paar $(w \mid, \rangle)$ „Abschlusspaar“
- Nun lässt sich leicht aus einer Ableitung $S = w_0 \Rightarrow \dots \Rightarrow w_m = w$ von w in G eine MPCP-Lösung s für $f(w)$ mit dem Lösungswort

$$\langle \mid w_0 \mid w_1 \mid \dots \mid w_m \mid \rangle$$

angeben

- Umgekehrt lässt sich aus jeder MPCP-Lösung s für $f(w)$ auch eine Ableitung von w in G gewinnen, womit

$$w \in L(G) \Leftrightarrow f(w) \in \text{MPCP}_\Gamma$$

gezeigt ist

Das Schnittproblem für kontextfreie Grammatiken (SP_{CFL})

Gegeben: Zwei kontextfreie Grammatiken G_1 und G_2 .

Gefragt: Ist $L(G_1) \cap L(G_2) \neq \emptyset$?

Satz

Das Schnittproblem für kontextfreie Grammatiken ist RE-vollständig und somit unentscheidbar.

Satz

Das Schnittproblem für kontextfreie Grammatiken ist RE-vollständig und somit unentscheidbar.

Beweis

- Das Problem SP_{CFL} ist semi-entscheidbar, da eine DTM systematisch nach einem Wort $x \in L(G_1) \cap L(G_2)$ suchen kann
- Um PCP auf SP_{CFL} zu reduzieren, betrachten wir für eine Folge $z = (x_1, \dots, x_k)$ von Strings $x_i \in \{0, 1\}^*$ die Sprache

$$L_z = \{i_n \dots i_1 \# x_{i_1} \dots x_{i_n} \mid n \geq 1, 1 \leq i_1, \dots, i_n \leq k\}$$

über dem Alphabet $\Sigma = \{0, 1, \dots, k, \#\}$

- Die Sprache L_z wird von der Grammatik $G_z = (\{A\}, \Sigma, P_z, A)$ mit der Regelmengemenge

$$P_z: A \rightarrow 1Ax_1, \dots, kAx_k, 1\#x_1, \dots, k\#x_k$$

erzeugt

Reduktion von PCP auf das Schnittproblem für CFL

- Zu einer PCP-Instanz $I = \begin{pmatrix} x_1 \dots x_k \\ y_1 \dots y_k \end{pmatrix}$ bilden wir das Paar (G_{z_1}, G_{z_2}) , wobei $z_1 = (x_1, \dots, x_k)$ und $z_2 = (y_1, \dots, y_k)$ ist
- Dann ist $L(G_{z_1}) \cap L(G_{z_2})$ die Sprache

$$\{i_n \dots i_1 \# x_{i_1} \dots x_{i_n} \mid 1 \leq n, x_{i_1} \dots x_{i_n} = y_{i_1} \dots y_{i_n}\}$$

- Folglich ist $s = (i_1, \dots, i_n)$ genau dann eine Lösung für I , wenn $i_n \dots i_1 \# x_{i_1} \dots x_{i_n} \in L(G_{z_1}) \cap L(G_{z_2})$ ist, d.h. es gilt

$$I \in \text{PCP} \Leftrightarrow L(G_{z_1}) \cap L(G_{z_2}) \neq \emptyset$$

- Also vermittelt $f : I \mapsto (G_{z_1}, G_{z_2})$ eine Reduktion von PCP auf das Schnittproblem für CFL



Beispiel

- Die PCP-Instanz

$$I = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} = \begin{pmatrix} 0 & 001 & 01100 \\ 00110 & 01011 & 00 \end{pmatrix}$$

wird auf das Grammatikpaar (G_{z_1}, G_{z_2}) mit folgenden Regeln reduziert:

$$P_{z_1}: A \rightarrow 1A0, 2A001, 3A01100, 1\#0, 2\#001, 3\#01100$$

$$P_{z_2}: A \rightarrow 1A00110, 2A01011, 3A00, 1\#00110, 2\#01011, 3\#00$$

- Der PCP-Lösung $s = (1, 3, 2, 3)$ entspricht dann das Wort

$$\begin{aligned} 3231\#x_1x_3x_2x_3 &= 3231\#00110000101100 \\ &= 3231\#00110000101100 = 3231\#y_1y_3y_2y_3 \end{aligned}$$

im Schnitt $L(G_{z_1}) \cap L(G_{z_2})$



Das Schnittproblem für DPDAs (SP_{DPDA})

Gegeben: Zwei DPDAs M_1 und M_2 .

Gefragt: Gilt $L(M_1) \cap L(M_2) \neq \emptyset$?

Korollar

SP_{DPDA} ist RE-vollständig und daher unentscheidbar.

Beweis

Für die Sprache $L_z = \{i_n \dots i_1 \# x_{i_1} \dots x_{i_n} \mid n \geq 1, 1 \leq i_1, \dots, i_n \leq k\}$ lässt sich leicht ein DPDA M_z angeben mit $L(M_z) = L_z$ □

Das Leerheitsproblem für DLBAs

Das Leerheitsproblem für DLBAs (LP_{DLBA})

Gegeben: Ein DLBA M .

Gefragt: Ist $L(M) \neq \emptyset$?

Satz

LP_{DLBA} ist RE-vollständig und daher unentscheidbar.

Beweisidee

- Es ist leicht zu sehen, dass $LP_{DLBA} \in RE$ ist
- Wir reduzieren PCP auf LP_{DLBA}
- Hierzu überführen wir eine PCP-Instanz $I = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$ in einen DLBA M mit

$$L(M) = L_{z_1} \cap L_{z_2}$$

- Dann ist die Funktion $f : I \mapsto M$ berechenbar und es gilt

$$I \in PCP \Leftrightarrow L_{z_1} \cap L_{z_2} \neq \emptyset \Leftrightarrow L(M) \neq \emptyset \Leftrightarrow M \in LP_{DLBA} \quad \square$$

Das Äquivalenzproblem für kontextfreie Grammatiken (ÄP_{CFL})

Gegeben: Zwei kontextfreie Grammatiken G_1 und G_2 .

Gefragt: Gilt $L(G_1) = L(G_2)$?

Satz

ÄP_{CFL} ist unentscheidbar.

Beweisidee

- Wir reduzieren $\overline{\text{PCP}}$ auf ÄP_{CFL}
- Für $I = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$ gilt

$$I \notin \text{PCP} \Leftrightarrow L_{z_1} \cap L_{z_2} = \emptyset \Leftrightarrow \overline{L}_{z_1} \cup \overline{L}_{z_2} = \Sigma^*$$

- Daher vermittelt die Funktion $f : I \mapsto \langle G_1, G_2 \rangle$ die gewünschte Reduktion, wobei G_1 und G_2 kontextfreie Grammatiken sind mit

$$L(G_1) = \overline{L}_{z_1} \cup \overline{L}_{z_2} \text{ und } L(G_2) = \Sigma^*$$

Dagegen ist es nicht schwer,

- für eine kontextsensitive Grammatik G und ein Wort x zu entscheiden, ob $x \in L(G)$ ist (Wortproblem WP_{CSL})
- für eine kontextfreie Grammatik G zu entscheiden, ob $L(G) \neq \emptyset$ ist (Leerheitsproblem LP_{CFL})
- für zwei reguläre Grammatiken G_1 und G_2 zu entscheiden, ob $L(G_1) = L(G_2)$ ist (Äquivalenzproblem $\ddot{A}P_{REG}$)
- für zwei reguläre Grammatiken G_1 und G_2 zu entscheiden, ob $L(G_1) \cap L(G_2) \neq \emptyset$ ist (Schnittproblem SP_{REG})

Satz

Die Probleme WP_{CSL} , LP_{CFL} , $\ddot{A}P_{REG}$ und SP_{REG} sind entscheidbar.

Beweis.

siehe Übungen

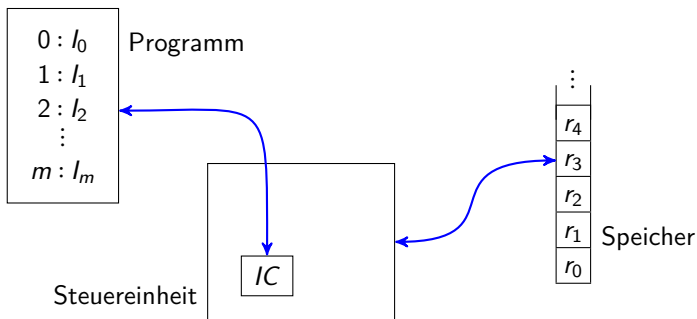


Folgende Tabelle zeigt, welche der betrachteten Entscheidungsprobleme für die verschiedenen Stufen der Chomsky-Hierarchie entscheidbar sind:

	Wort- problem $x \in L?$	Leerheits- problem $L = \emptyset?$	Äquivalenz- problem $L_1 = L_2?$	Schnitt- problem $L_1 \cap L_2 \neq \emptyset?$
REG	ja	ja	ja	ja
DCFL	ja	ja	ja ^a	nein
CFL	ja	ja	nein	nein
D CSL	ja	nein	nein	nein
CSL	ja	nein	nein	nein
RE	nein	nein	nein	nein

^aBewiesen in 1997 von Gérard Sénizergues (Univ. Bordeaux)

Die Registermaschine (random access machine, RAM) 318



- führt ein Programm $P = (l_0, \dots, l_m)$ aus, das aus einer endlichen Folge von Befehlen (**instructions**) l_j , $j = 0, \dots, m$ besteht
- hat einen Befehlszähler (**instruction counter**) IC , der die Nummer des nächsten Befehls angibt (zu Beginn ist $IC = 0$)
- verfügt über einen frei adressierbaren Speicher (**random access memory**) mit unendlich vielen Speicherzellen (Registern) r_i , $i \geq 0$, die beliebig große natürliche Zahlen aufnehmen können

In **GOTO-Programmen** sind folgende Befehle zulässig
(wobei $i, j, c \in \mathbb{N} = \{0, 1, 2, \dots\}$):

Befehl	Semantik
$r_i := r_j + c$	setzt Register r_i auf den Wert $r_j + c$
$r_i := r_j \div c$	setzt Register r_i auf den Wert $\max(0, r_j - c)$
GOTO j	setzt den Befehlszähler IC auf den Wert j
IF $r_i = c$ THEN GOTO j	setzt IC auf j , falls r_i den Wert c hat
HALT	beendet die Programmausführung

Bei Ausführung der ersten beiden Befehle wird zudem der Befehlszähler IC um eins erhöht.

GOTO-Berechenbarkeit

Definition

Eine partielle Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ heißt **GOTO-berechenbar**, falls es ein GOTO-Programm $P = (I_0, \dots, I_m)$ mit folgender Eigenschaft gibt:

- Wird P auf einer RAM mit den Werten $r_i = n_i$ für $i = 1, \dots, k$, sowie $IC = 0$ und $r_i = 0$ für $i = 0, k + 1, k + 2, \dots$ gestartet, so
- hält P genau dann, wenn $(n_1, \dots, n_k) \in \text{dom}(f)$ ist, und
- sobald P hält, hat r_0 den Wert $f(n_1, \dots, n_k)$

Beispiel

Folgendes GOTO-Programm berechnet die Funktion $f(x, y) = xy$:

0	IF $r_1 = 0$ THEN GOTO 4	5	$r_3 := r_2$
1	$r_1 := r_1 \div 1$	6	IF $r_3 = 0$ THEN GOTO 3
2	$r_0 := r_0 + r_2$ GOTO 5	7	$r_3 := r_3 \div 1$
3	GOTO 0	8	$r_0 := r_0 + 1$
4	HALT	9	GOTO 6



WHILE- und LOOP-Programme

- Die Syntax von **WHILE-Programmen** ist induktiv wie folgt definiert (wobei $i, j, c \in \mathbb{N}$):
 - Jede Wertzuweisung der Form $x_i := x_j + c$ oder $x_i := x_j \div c$ ist ein WHILE-Programm.
 - Falls P und Q WHILE-Programme sind, so auch
 - $P; Q$ und
 - **IF $x_i = c$ THEN P ELSE Q END**
 - **WHILE $x_i \neq c$ DO P END**
- Die Syntax von **LOOP-Programmen** ist genauso definiert, nur dass Schleifen der Form **LOOP x_i DO P END** an die Stelle von WHILE-Schleifen treten
- Die Semantik von WHILE-Programmen ist selbsterklärend
- Eine LOOP-Schleife **LOOP x_i DO P END** wird so oft ausgeführt, wie der Wert von x_i zu Beginn der Schleife angibt

WHILE- und LOOP-Berechenbarkeit

- Eine partielle Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ heißt **WHILE-berechenbar**, falls es ein WHILE-Programm P mit folgender Eigenschaft gibt:
 - Wird P mit den Werten $x_i = n_i$ für $i = 1, \dots, k$ gestartet, so
 - hält P genau dann, wenn $(n_1, \dots, n_k) \in \text{dom}(f)$ ist, und
 - sobald P hält, hat x_0 den Wert $f(n_1, \dots, n_k)$
- Die **LOOP-Berechenbarkeit** von f ist entsprechend definiert

Beispiel

Die Funktion $f(n_1, n_2) = n_1 n_2$ wird von dem WHILE-Programm

WHILE $x_1 \neq 0$ DO $x_0 := x_0 + x_2;$ $x_1 := x_1 \div 1$ END	{	$x_3 := x_2;$ WHILE $x_3 \neq 0$ DO $x_0 := x_0 + 1; x_3 := x_3 \div 1$ END
---	---	---

sowie von folgendem LOOP-Programm berechnet:

LOOP x_1 **DO** ~~$x_0 := x_0 + x_2$~~ **END** **LOOP** x_2 **DO** $x_0 := x_0 + 1$ **END** ◀

Satz

Eine partielle Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ ist genau dann GOTO-berechenbar, wenn sie WHILE-berechenbar ist.

- Sei P ein WHILE-Programm, das $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ berechnet
- Wir übersetzen P wie folgt in ein äquivalentes GOTO-Programm P'
- P' speichert den Variablenwert x_i im Register r_i
- Damit lassen sich alle Wertzuweisungen von P direkt in entsprechende Befehle von P' transformieren
- Eine Schleife der Form **WHILE** $x_i \neq c$ **DO** Q **END** simulieren wir durch folgendes GOTO-Programmstück:

```
 $M_1$  IF  $r_i = c$  THEN GOTO  $M_2$   
     $Q'$   
    GOTO  $M_1$ 
```

```
 $M_2$  ;
```

- Ähnlich lässt sich die Verzweigung **IF** $x_i = c$ **THEN** Q_1 **ELSE** Q_2 **END** in ein GOTO-Programmstück transformieren
- Zudem fügen wir ans Ende von P' den HALT-Befehl an

- Sei $P = (I_0, \dots, I_m)$ ein GOTO-Programm, das $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ berechnet, und sei r_z , $z > k$, ein Register, das in P nicht benutzt wird
- Wir übersetzen P wie folgt in ein äquivalentes WHILE-Programm P' :

```

 $x_z := 0;$ 
WHILE  $x_z \neq m + 1$  DO
  IF  $x_z = 0$  THEN  $P'_0$  END;
   $\vdots$ 
  IF  $x_z = m$  THEN  $P'_m$  END
END

```

- Dabei ist P'_ℓ abhängig vom Befehl I_ℓ folgendes WHILE-Programm:

I_ℓ	P'_ℓ
$r_i := r_j + c$	$x_i := x_j + c; x_z := x_z + 1$
$r_i := r_j \dot{-} c$	$x_i := x_j \dot{-} c; x_z := x_z + 1$
GOTO j	$x_z := j$
IF $r_i = c$ THEN GOTO j	IF $x_i = c$ THEN $x_z := j$ ELSE $x_z := x_z + 1$ END
HALT	$x_z := m + 1$

- Man beachte, dass P' nur eine WHILE-Schleife enthält

- Offensichtlich lässt sich jedes LOOP-Programm durch ein WHILE-Programm simulieren
- Andererseits können LOOP-Programme nur totale Funktionen berechnen, d.h. nicht jedes WHILE-Programm ist durch ein LOOP-Programm simulierbar
- Es gibt auch totale WHILE-berechenbare Funktionen, die nicht LOOP-berechenbar sind
- Eine solche Funktion kann mittels Diagonalisierung definiert werden
- Ein Beispiel für eine "natürliche" Funktion mit dieser Eigenschaft ist die **Ackermannfunktion** $a : \mathbb{N}^2 \rightarrow \mathbb{N}$, die wie folgt definiert ist

$$a(x, y) = \begin{cases} y + 1, & x = 0 \\ a(x - 1, 1), & x \geq 1, y = 0 \\ a(x - 1, a(x, y - 1)), & x, y \geq 1 \end{cases}$$

Numerische Repräsentation von Wörtern

- Als nächstes wollen wir die Äquivalenz von Turing- und GOTO-Berechenbarkeit zeigen
- Da DTMs auf Wörtern und GOTO-Programme auf Zahlen operieren, müssen wir Wörter durch Zahlen kodieren (und umgekehrt)
- Sei $\Sigma = \{a_0, \dots, a_{m-1}\}$ ein Alphabet
- Dann können wir jedes Wort $x = a_{i_1} \dots a_{i_n} \in \Sigma^*$ wie folgt durch eine natürliche Zahl kodieren:

$$\text{num}_{\Sigma}(x) = \sum_{j=0}^{n-1} m^j + \sum_{j=1}^n i_j m^{n-j} = \begin{cases} n, & m = 1 \\ \frac{m^n - 1}{m - 1} + (i_1 \dots i_n)_m, & m \geq 2 \end{cases}$$

- Da die Abbildung $\text{num}_{\Sigma} : \Sigma^* \rightarrow \mathbb{N}$ bijektiv ist, können wir umgekehrt jede natürliche Zahl n durch das Wort $\text{str}_{\Sigma}(n) = \text{num}_{\Sigma}^{-1}(n)$ kodieren

Numerische Repräsentation von Wörtern

Beispiel

- Für das unäre Alphabet $\Sigma = \{a\}$ ist $num_{\Sigma}(a^n) = n$ und $num_{\Sigma}^{-1}(n) = a^n$

w	ε	a	aa	aaa	\dots
$num_{\Sigma}(w)$	0	1	2	3	\dots

- Im Fall $\Sigma = \{a, b, c\}$ erhalten wir folgende Werte

w	ε	a	b	c	aa	ab	ac	ba	bb	bc	ca	cb	cc	aaa	\dots
$num_{\Sigma}(w)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	\dots

- Im Fall $\Sigma = \{0, 1\}$ schreiben wir kurz num und str für num_{Σ} und str_{Σ}

x	ε	0	1	00	01	10	\dots
$num(x)$	0	1	2	3	4	5	\dots

n	0	1	2	3	4	5	\dots
$str(n)$	ε	0	1	00	01	10	\dots



- Die Kodierungsfunktion $str : \mathbb{N} \rightarrow \{0, 1\}^*$ lässt sich wie folgt zu einer Kodierungsfunktion $str_k : \mathbb{N}^k \rightarrow \{0, 1, \#\}^*$ erweitern:

$$str_k(n_1, \dots, n_k) = str(n_1)\# \dots \# str(n_k)$$

- Nun können wir eine partielle Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ durch folgende partielle Wortfunktion $\hat{f} : \{0, 1, \#\}^* \rightarrow \{0, 1\}^* \cup \{\uparrow\}$ repräsentieren:

$$\hat{f}(w) = \begin{cases} str(n), & w = str_k(n_1, \dots, n_k) \text{ und } f(n_1, \dots, n_k) = n \in \mathbb{N} \\ \uparrow, & \text{sonst} \end{cases}$$

- Wir nennen \hat{f} die **String-Repräsentation** von f und f die **numerische Repräsentation** von \hat{f}

Beispiel

- Die Fkt. $f : (x, y) \mapsto x + y$ wird durch folgende Wortfkt. repräsentiert:

$$\hat{f}(w) = \begin{cases} \text{str}(x + y), & w = \text{str}_2(x, y) \\ \uparrow, & \text{sonst} \end{cases}$$

- Da f eine totale Funktion ist, gilt

$$\hat{f}(w) \neq \uparrow \Leftrightarrow w \in \text{img}(\text{str}_2),$$

also genau dann, wenn w die Form $x\#y$ mit $x, y \in \{0, 1\}^*$ hat

- Beispielsweise ist $\hat{f}(w)$ für folgende Wörter w definiert:

w	$\#$	$0\#$	$1\#$	$\#0$	$\#1$	$00\#$	$01\#$	$0\#0$	$0\#1$	$10\#$	\dots
(x, y)	$(0,0)$	$(1,0)$	$(2,0)$	$(0,1)$	$(0,2)$	$(3,0)$	$(4,0)$	$(1,1)$	$(1,2)$	$(5,0)$	\dots
$x + y$	0	1	2	1	2	3	4	2	3	5	\dots
$\hat{f}(w)$	ε	0	1	0	1	00	01	1	00	10	\dots

Satz

Eine partielle Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ ist genau dann GOTO-berechenbar, wenn ihre String-Repräsentation \hat{f} Turing-berechenbar ist.

Beweis

- Sei P ein GOTO-Programm, das eine partielle Fkt. $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ auf einer RAM R berechnet
- Dann existiert eine Zahl m , so dass P nur Register r_i mit $i \leq m$ benutzt
- Daher lässt sich eine Konfiguration von R durch Angabe der Inhalte des Befehlszählers IC und der Register r_0, \dots, r_m beschreiben
- Wir konstruieren eine $(m+2)$ -DTM M , die
 - den Inhalt von IC in ihrem Zustand,
 - die Registerwerte r_1, \dots, r_m auf den Bändern $1, \dots, m$ und
 - den Wert von r_0 auf dem Ausgabeband $m+2$ speichert
- Ein Registerwert r_i wird hierbei in der Form $str(r_i)$ gespeichert
- Band $m+1$ wird zur Ausführung von Hilfsberechnungen benutzt

- Die Aufgabe von M ist es, bei Eingabe $w \in \{0, 1, \#\}^*$ das Wort $str(f(n_1, \dots, n_k))$ auszugeben, wenn $w = str_k(n_1, \dots, n_k)$ für ein Tupel $(n_1, \dots, n_k) \in dom(f)$ ist, und andernfalls nicht zu halten
- Zuerst überprüft M , ob in w das $\#$ -Zeichen $(k - 1)$ -mal vorkommt
- Dann kopiert M die Teilwörter $str(n_i)$ für $i = 2, \dots, k$ auf das i -te Band und löscht auf dem 1. Band alle Eingabezeichen bis auf $str(n_1)$
- Für $i = 1, \dots, m$ sind nun auf Band i die Registerinhalte $r_i = n_i$ und auf Band $m + 2$ der Wert $r_0 = 0$ gespeichert
- Danach führt M das Programm P Befehl für Befehl aus
- Es ist klar, dass M jeden Befehl l in P durch eine geeignete Folge von Anweisungen simulieren kann, die die Registerinhalte und den Wert von IC entsprechend modifizieren
- Sobald P stoppt, hält auch M und gibt das auf Band $m + 2$ befindliche Wort $str(r_0) = str(f(n_1, \dots, n_k)) = \hat{f}(w)$ aus

- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ eine DTM, die die String-Repräsentation \hat{f} einer partiellen Funktion $f : \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$ berechnet
- M gibt also bei Eingabe w das Wort $str(f(n_1, \dots, n_k))$ aus, falls w die Form $w = str_k(n_1, \dots, n_k)$ hat und $f(n_1, \dots, n_k)$ definiert ist, und hält andernfalls nicht
- Wir konstruieren ein GOTO-Programm P , das bei Eingabe (n_1, \dots, n_k) die DTM M bei Eingabe $w = str_k(n_1, \dots, n_k)$ simuliert
- Wir können annehmen, dass M eine 1-DTM ist
- Sei $Z = \{q_0, \dots, q_r\}$ und $\Gamma = \{a_0, \dots, a_{m-1}\}$, wobei wir annehmen, dass $a_0 = \sqcup$, $a_1 = 0$, $a_2 = 1$ und $a_3 = \#$ ist
- Eine Konfiguration $K = uq_i v$ von M mit $u = a_{i_1} \dots a_{i_s}$ und $v = a_{j_1} \dots a_{j_t}$ wird wie folgt in den Registern r_0, r_1, r_2 gespeichert:
 - $r_0 = (i_1 \dots i_s)_m$
 - $r_1 = i$
 - $r_2 = (j_t \dots j_1)_m$

- Eine Konfiguration $K = uq_jv$ von M mit $u = a_{i_1} \dots a_{i_s}$ und $v = a_{j_1} \dots a_{j_t}$ wird wie folgt in den Registern r_0, r_1, r_2 gespeichert:
 - $r_0 = (i_1 \dots i_s)_m$
 - $r_1 = j$
 - $r_2 = (j_t \dots j_1)_m$
- P besteht aus 3 Programmteilen $P = P_1, P_2, P_3$:
 - P_1 stellt in den drei Registern r_0, r_1, r_2 die Startkonfiguration $K_w = q_0w$ von M bei Eingabe $w = str_k(n_1, \dots, n_k)$ her, d.h. P_1 berechnet in Register r_2 die Zahl $(j_t \dots j_1)_m$, wobei $w = a_{j_1} \dots a_{j_t}$ ist, und setzt r_0 und r_1 auf den Wert 0
 - P_2 überführt die in r_0, r_1, r_2 gespeicherte Konfiguration von M solange in die zugehörige Nachfolgekonfiguration bis M hält (siehe nächste Folie)
 - Danach transformiert P_3 noch den aktuellen Inhalt $(i_1 \dots i_s)_m$ von Register r_0 in die Zahl $num(a_{i_1} \dots a_{i_s})$ und hält

- Das Programmstück P_2 hat die Form

$$M_2 \quad r_3 := r_2 \text{ MOD } m$$

$$\text{IF } r_1 = 0 \wedge r_3 = 0 \text{ THEN GOTO } M_{0,0}$$

$$\vdots$$

$$\text{IF } r_1 = r \wedge r_3 = m - 1 \text{ THEN GOTO } M_{r,m-1}$$

- Die Befehle ab Position $M_{i,j}$ hängen von $\delta(q_i, a_j)$ ab:
 - Im Fall $\delta(q_i, a_j) = \emptyset$ markiert $M_{i,j}$ den Beginn von P_3
 - Im Fall $\delta(q_i, a_j) = \{(q_{i'}, a_{j'}, L)\}$ werden folgende Befehle ausgeführt:

$$M_{i,j} \quad r_1 := i'$$

$$r_2 := r_2 \text{ DIV } m$$

$$r_2 := r_2 m + j'$$

$$r_2 := r_2 m + (r_0 \text{ MOD } m)$$

$$r_0 := r_0 \text{ DIV } m$$

$$\text{GOTO } M_2$$

- Die übrigen Fälle sind ähnlich
- Makros wie die **MOD**- und **DIV**- Befehle können durch entsprechende GOTO-Programmstücke ersetzt werden □