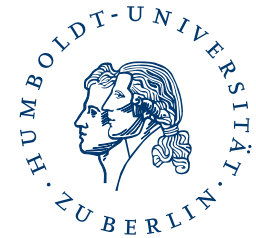
Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2012/13

Entscheidbare und semi-entscheidbare Sprachen

Definition

- Eine NTM M hält bei Eingabe x (kurz: $M(x) = \downarrow$), falls alle Rechnungen von $M(x)$ eine endliche Länge haben.
- Falls $M(x)$ nicht hält, schreiben wir auch kurz $M(x) = \uparrow$.
- Eine NTM M entscheidet eine Eingabe x, falls $M(x)$ hält oder eine Konfiguration mit einem Endzustand erreichen kann.
- Eine Sprache $L \subseteq \Sigma^*$ heißt entscheidbar, falls eine DTM M mit $L(M) = L$ existiert, die jede Eingabe $x \in \Sigma^*$ entscheidet.
- Jede von einer DTM M erkannte Sprache heißt semi-entscheidbar.

Bemerkung

- Die von M akzeptierte Sprache $L(M)$ heißt semi-entscheidbar, da M zwar alle Eingaben $x \in L$ entscheidet (aber eventuell nicht alle $x \in \bar{L}$).
- Später werden wir sehen, dass genau die Typ-0 Sprachen semi-entscheidbar sind.

Berechnung von Funktionen

Definition

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

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

  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.

Berechenbarkeit von partiellen Funktionen

Definition

- Eine partielle Funktion hat die Form $f : \Sigma^* \to \Gamma^* \cup \{\uparrow\}$.
- Für $f(x) = \uparrow$ sagen wir auch $f(x)$ ist undefiniert.
- Der Definitionsbereich (engl. *domain*) von f ist

  $$dom(f) = \{x \in \Sigma^* \mid f(x) \neq \uparrow\}.$$

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

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

- f heißt total, falls $dom(f) = \Sigma^*$ ist.
- Eine k-DTM $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ berechnet f, falls $M(x)$ für alle $x \in dom(f)$ das Wort $f(x)$ ausgibt und für alle $x \notin dom(f)$ keine Ausgabe berechnet (d.h. $M(x) = \uparrow$).

Wir fassen die entscheidbaren Sprachen und die (partiellen) berechenbaren Funktionen in folgenden Klassen zusammen:

$$\mathrm{REC} = \{L(M) \mid M \text{ ist eine DTM, die jede Eingabe entscheidet}\},$$
$$\mathrm{FREC} = \{f \mid f \text{ ist eine berechenbare (totale) Funktion}\},$$
$$\mathrm{FREC}_p = \{f \mid f \text{ ist eine berechenbare partielle Funktion}\}.$$

Dann gilt:

- $\mathrm{FREC} \subsetneq \mathrm{FREC}_p$ und
- $\mathrm{REG} \subsetneq \mathrm{DCFL} \subsetneq \mathrm{CFL} \subsetneq \mathrm{DCSL} \subseteq \mathrm{CSL} \subsetneq \mathrm{REC} \subsetneq \mathrm{RE}$.

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	$\cdots$
x^+	0	1	00	01	10	11	000	001	$\cdots$

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

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

- 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. ◁

Berechenbarkeit von charakteristischen Funktionen

Satz

- Eine Sprache $A \subseteq \Sigma^*$ ist genau dann entscheidbar, wenn ihre charakteristische Funktion $\chi_A : \Sigma^* \to \{0,1\}$ berechenbar ist:

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

- Eine Sprache $A \subseteq \Sigma^*$ ist genau dann semi-entscheidbar, falls ihre partielle charakteristische Funktion $\hat{\chi}_A : \Sigma^* \to \{1, \uparrow\}$ berechenbar ist:

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

Beweis

Siehe Übungen.

Charakterisierung der rekursiv aufzählbaren Sprachen

Definition

Eine Sprache $A \subseteq \Sigma^*$ heißt rekursiv aufzählbar, falls $A = \varnothing$ oder das Bild $img(f)$ einer berechenbaren Funktion $f : \Gamma^* \to \Sigma^*$ ist.

Satz

Folgende Eigenschaften sind äquivalent:

1. A ist semi-entscheidbar (d.h. A wird von einer DTM akzeptiert),
2. A wird von einer 1-DTM akzeptiert,
3. A wird von einer 1-NTM akzeptiert,
4. A ist vom Typ 0,
5. A wird von einer NTM akzeptiert,
6. A ist rekursiv aufzählbar.

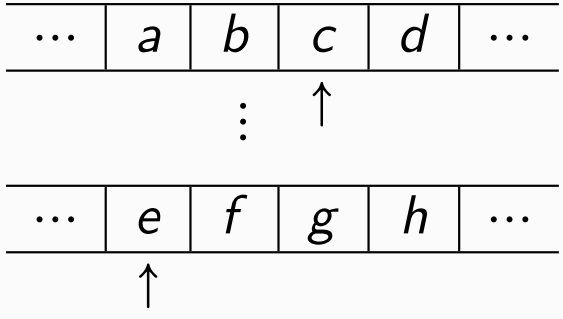
Beweis

Die Implikation 2 $\Rightarrow$ 3 ist klar. Die Implikationen 3 $\Rightarrow$ 4 $\Rightarrow$ 5 werden in den Übungen gezeigt. Hier zeigen wir 1 $\Rightarrow$ 2 und 5 $\Rightarrow$ 6 $\Rightarrow$ 1.

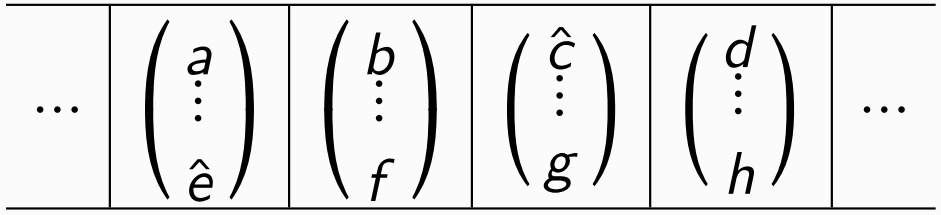
Simulation einer k-DTM durch eine 1-DTM

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

- 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 ❶ $\Rightarrow$ ❷: $\{L(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine 1-DTM}\}$

- Das heißt, M' arbeitet mit dem Alphabet

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

- und erzeugt bei Eingabe $x = x_1 \dots x_n \in \Sigma^*$ 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

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

Simulation einer k-DTM durch eine 1-DTM

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

- 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. $\hat{a}_1, \ldots, \hat{a}_k$) gefunden hat.
  - Diese Zeichen speichert M' in ihrem Zustand.
  - Anschließend geht M' wieder nach links und realisiert dabei die durch $\delta(q, a_1, \ldots, a_k)$ vorgegebene Anweisung von M.
  - Den aktuellen Zustand q von M speichert M' ebenfalls in ihrem Zustand.
- Sobald M in einen Endzustand übergeht, wechselt M' ebenfalls in einen Endzustand und hält.
- Nun ist leicht zu sehen, dass $L(M') = L(M)$ ist. □

Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von ❺ ⇒ ❻: $\{L(M) \mid M \text{ ist eine NTM}\} \subseteq \{L \mid L \text{ ist rek. aufzählbar}\}$

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

  $$code(K) = \#q\#u_1\#a_1\#v_1\# \ldots \#u_k\#a_k\#v_k\#$$

  und eine Rechnung $K_0 \vdash \cdots \vdash K_t$ durch $code(K_0) \ldots code(K_t)$.
- Dann lassen sich die Wörter von A durch folgende Funktion $f : \tilde{\Gamma}^* \to \Sigma^*$ aufzählen (dabei ist x_0 ein beliebiges Wort in A):

$$f(x) = \begin{cases} y, & x \text{ kodiert eine Rechnung } K_0 \vdash \cdots \vdash K_t \text{ von } M \text{ mit} \\ & K_0 = K_y \text{ und } K_t \in E \times (\Gamma^* \times \Gamma \times \Gamma^*)^k \\ x_0, & \text{sonst.} \end{cases}$$

- Da f berechenbar ist, ist $A = img(f)$ rekursiv aufzählbar. □

Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von 6 $\Rightarrow$ 1: $\{L \mid L \text{ ist rek. aufzählbar}\} \subseteq \{L(M) \mid M \text{ ist eine DTM}\}$

- Sei $f : \Gamma^* \to \Sigma^*$ eine Funktion mit $A = img(f)$ und sei M eine k-DTM, die f berechnet.
- Betrachte folgende $(k+1)$-DTM M', die bei Eingabe x
  - auf dem 2. Band der Reihe nach alle Wörter y in Γ^* erzeugt,
  - für jedes y den Wert $f(y)$ durch Simulation von $M(y)$ berechnet, und
  - ihre Eingabe x akzeptiert, sobald $f(y) = x$ ist. $\square$

Charakterisierung der entscheidbaren Sprachen

Satz

A ist genau dann entscheidbar, wenn A und $\bar{A}$ semi-entscheidbar sind, d.h. REC = RE $\cap$ co-RE.

Beweis.

- Falls A entscheidbar ist, ist auch $\bar{A}$ entscheidbar, d.h. A und $\bar{A}$ sind dann auch semi-entscheidbar.
- Für die Rückrichtung seien $f_1, f_2 : \Gamma^* \to \Sigma^*$ Turing-berechenbare Funktionen mit $img(f_1) = A$ und $img(f_2) = \bar{A}$.
- Wir betrachten folgende DTM M, die bei Eingabe x für jedes $y \in \Gamma^*$ die beiden Werte $f_1(y)$ und $f_2(y)$ bestimmt und im Fall
  - $f_1(y) = x$ in einem Endzustand hält,
  - $f_2(y) = x$ in einem Nichtendzustand hält.
- Da jede Eingabe x entweder in $img(f_1) = A$ oder in $img(f_2) = \bar{A}$ enthalten ist, hält M bei allen Eingaben. □

Kodierung (Gödelisierung) von Turingmaschinen

- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ eine 1-DTM mit
  - Zustandsmenge $Z = \{q_0, \ldots, q_m\}$ (o.B.d.A. sei $E = \{q_m\}$),
  - Eingabealphabet $\Sigma = \{0, 1, \#\}$ und
  - Arbeitsalphabet $\Gamma = \{a_0, \ldots, a_l\}$, wobei wir o.B.d.A. $a_0 = \sqcup$, $a_1 = 0$, $a_2 = 1$ und $a_3 = \#$ annehmen.
- Dann können wir jede Anweisung der Form $q_i a_j \to q_{i'} a_{j'} D$ durch das Wort

  $$\#bin(i)\#bin(j)\#bin(i')\#bin(j')\#b_D\#$$

  kodieren.
- 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. $0 \mapsto 00$, $1 \mapsto 11$, $\# \mapsto 10$), so gelangen wir zu einer Binärkodierung w_M von M.
- Die Binärzahl w_M wird auch die Gödel-Nummer von M genannt.
- M_w ist durch Angabe von w bis auf die Benennung ihrer Zustände und Arbeitszeichen eindeutig bestimmt.
- Ganz analog lassen sich auch DTMs mit einer beliebigen Anzahl von Bändern (sowie NTMs, Konfigurationen oder Rechnungen von TMs) binär kodieren.
- Umgekehrt können wir jedem Binärstring $w \in \{0, 1\}^*$ eine DTM M_w wie folgt zuordnen:

$$M_w = \begin{cases} M, & \text{falls eine DTM } M \text{ mit } w_M = w \text{ existiert,} \\ M_0, & \text{sonst (dabei sei } M_0 \text{ eine beliebige DTM).} \end{cases}$$

Unentscheidbarkeit des Halteproblems

Definition

- Das Halteproblem ist die Sprache

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

- Das spezielle Halteproblem ist

$$K = \left\{ w \in \{0,1\}^* \;\middle|\; \begin{array}{l} \text{die DTM } M_w \\ \text{hält bei Eingabe } w \end{array} \right\}$$

χ_H	x_1	x_2	x_3	$\cdots$
w_1	0	1	0	$\cdots$
w_2	0	1	1	$\cdots$
w_3	1	1	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

χ_K				
w_1	**0**			
w_2		**1**		
w_3			**0**	
$\vdots$				$\ddots$

Satz

$K \in \text{RE} \setminus \text{co-RE}$.

Semi-Entscheidbarkeit des speziellen Halteproblems

Beweis von $K \in \mathrm{RE}$

- Sei w_0 die Kodierung einer DTM, die bei jeder Eingabe (sofort) hält und betrachte die Funktion

$$f(x) = \begin{cases} w, & x \text{ ist Kodierung einer haltenden Rechnung einer DTM } M_w \text{ bei Eingabe } w, \\ w_0, & \text{sonst.} \end{cases}$$

- Da f berechenbar und $img(f) = K$ ist, folgt $K \in \mathrm{RE}$. □

Bemerkung

Ganz ähnlich lässt sich $H \in \mathrm{RE}$ zeigen.

Unentscheidbarkeit des speziellen Halteproblems

Beweisidee

- Da $K \in \text{RE}$ ist, gibt es in der Matrixdarstellung von χ_H eine Zeile (sprich DTM M), die mit der Diagonalen der Matrix übereinstimmt.
- Beispielsweise können wir für M eine DTM wählen, die die partielle charakteristische Funktion $\hat{\chi}_K$ von K berechnet.
- Wäre auch $\bar{K} \in \text{RE}$, so würde eine DTM $\hat{M}$ existieren, so dass die zugehörige Zeile in der Matrix invers zur Zeile von M und damit zur Diagonalen ist.
- Beispielsweise können wir für $\hat{M}$ eine DTM wählen, die die partielle charakteristische Funktion $\hat{\chi}_{\bar{K}}$ von $\bar{K}$ berechnet.
- Da in keiner Matrix eine Zeile existieren kann, die invers zur Diagonalen ist, führt dies auf den gewünschten Widerspruch.

Unentscheidbarkeit des speziellen Halteproblems

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

- Angenommen, die Sprache

$$\bar{K} = \{w \mid M_w(w) \text{ hält nicht}\} \qquad (*)$$

wäre semi-entscheidbar.

- Dann existiert eine DTM $\hat{M}$, die die partielle charakteristische Funktion $\hat{\chi}_{\bar{K}}$ von $\bar{K}$ berechnet, d.h. es gilt

$$\hat{M}(w) \text{ hält} \Leftrightarrow w \in \bar{K} \qquad (**)$$

- Für die Kodierung $\hat{w}$ von $\hat{M}$ folgt dann aber

$$\hat{w} \in \bar{K} \overset{(*)}{\Leftrightarrow} M_{\hat{w}}(\hat{w}) \text{ hält nicht} \overset{(**)}{\Leftrightarrow} \hat{w} \notin \bar{K} \lightning \text{ (Widerspruch!)}$$

□

χ_H	x_1	x_2	x_3	$\cdots$
w_1	**0**	1	0	$\ldots$
w_2	0	**1**	1	$\ldots$
w_3	1	1	**0**	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
$\hat{w}$	**1**	**0**	**1**	$\cdots$

Unentscheidbarkeit des speziellen Halteproblems

Korollar

REC $\subsetneq$ RE.

Beweis

Klar, da $K \in \text{RE} - \text{REC}$. $\square$

Der Reduktionsbegriff

Definition

Eine Sprache $A \subseteq \Sigma^*$ heißt auf $B \subseteq \Gamma^*$ reduzierbar (kurz: $A \leq B$), falls eine berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ 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:

$$\begin{aligned} w \in K &\Leftrightarrow M_w \text{ ist eine DTM, die bei Eingabe } w \text{ hält} \\ &\Leftrightarrow w\#w \in H. \end{aligned}$$

◁

Abschluss von REC unter $\leq$

Definition

- Eine Sprachklasse $\mathcal{C}$ heißt unter $\leq$ abgeschlossen, wenn für alle Sprachen A, B gilt:

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

Satz

Die Klassen REC und RE sind unter $\leq$ abgeschlossen.

Beweis

- Gelte $A \leq B$ mittels f und sei $B \in$ REC.
- Dann ex. eine DTM M, die χ_B berechnet.
- Betrachte folgende DTM M':
  - M' berechnet bei Eingabe x zuerst den Wert $f(x)$ und
  - simuliert dann M bei Eingabe $f(x)$.

Abschluss 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, die χ_B berechnet.
- Betrachte folgende DTM M':
  - M' berechnet bei Eingabe x zuerst den Wert $f(x)$ und
  - simuliert dann M bei Eingabe $f(x)$.
- Wegen $x \in A \Leftrightarrow f(x) \in B$ ist $\chi_A(x) = \chi_B(f(x))$ und daher folgt

$$M'(x) = M(f(x)) = \chi_B(f(x)) = \chi_A(x).$$

- Also berechnet M' die Funktion χ_A, d.h. $A \in \text{REC}$. □

Bemerkung

Der Abschluss von RE unter $\leq$ folgt analog (siehe Übungen).

Der Vollständigkeitsbegriff

Definition

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

$$\forall L \in \mathcal{C} : L \leq A.$$

- Eine $\mathcal{C}$-harte Sprache A, die zu $\mathcal{C}$ gehört, heißt $\mathcal{C}$-vollständig.

Beispiel

Das Halteproblem H ist RE-vollständig. Es gilt nämlich

- $H \in \mathrm{RE}$ und
- $\forall L \in \mathrm{RE} : L \leq H$

mittels der Reduktionsfunktion $x \mapsto w\#x$, wobei w die Kodierung einer DTM M_w ist, die $\hat{\chi}_L$ berechnet. ◁

Bemerkung

Auch das spezielle Halteproblem K ist RE-vollständig (siehe Übungen).

H ist nicht entscheidbar

Korollar

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

Beweis

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

Bemerkung

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

Korollar

$H \notin \mathrm{REC}$.

Das Halteproblem bei leerem Band

Definition

Das Halteproblem bei leerem Band ist die Sprache

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

Satz

H_0 ist RE-vollständig.

Beweis

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

χ_H	x_1	x_2	x_3	$\cdots$
w_1	0	1	0	$\cdots$
w_2	0	1	1	$\cdots$
w_3	1	1	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

χ_{H_0}	x_1 $(= \varepsilon)$
w_1	0
w_2	0
w_3	1
$\vdots$	$\vdots$

H_0 ist RE-vollständig

Beweis

- $H_0 \in \mathrm{RE}$ folgt wegen $H_0 \leq H \in \mathrm{RE}$ mittels der Reduktionsfunktion $w \mapsto w\#\varepsilon$.
- Sei $A \in \mathrm{RE}$ und sei w die Kodierung einer DTM, die $\hat{\chi}_A$ berechnet. Um A auf H_0 zu reduzieren, transformieren wir $x \in \{0,1\}^*$ in die Kodierung w_x einer DTM M, die zunächst ihre Eingabe durch x ersetzt und dann $M_w(x)$ simuliert. Dann gilt

$$x \in A \iff w_x \in H_0$$

und somit $A \leq H_0$ mittels der Reduktionsfunktion $x \mapsto w_x$.

Korollar

$H_0 \notin \mathrm{REC}$.

Der Satz von Rice

Frage

- Kann man einer beliebig vorgegebenen DTM ansehen, ob die von ihr berechnete Funktion eine gewisse Eigenschaft hat?
- Kann man beispielsweise entscheiden, ob eine gegebene DTM eine totale Funktion berechnet?

Antwort

Nein (es sei denn, die fragliche Eigenschaft ist trivial, d.h. keine oder jede DTM berechnet eine Funktion mit dieser Eigenschaft).

Der Satz von Rice

Definition

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

$$L_{\mathcal{F}} = \left\{ w \in \{0,1\}^* \;\middle|\; \text{die DTM } M_w \text{ berechnet eine Funktion in } \mathcal{F} \right\}.$$

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

Der Satz von Rice besagt, dass $L_{\mathcal{F}}$ nur für triviale Eigenschaften entscheidbar ist.

Satz (Satz von Rice)

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

Der Satz von Rice

Beispiel

- Die Sprache

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

ist unentscheidbar.

- Dies folgt aus dem Satz von Rice, da die Eigenschaft

$$\mathcal{F} = \{f \mid \{0\}^* \subseteq dom(f) \wedge f(0^n) = 0^{n+1} \text{ für alle } n \geq 0\}$$

nicht trivial und $L = L_{\mathcal{F}}$ ist.

- $\mathcal{F}$ ist nicht trivial, da z.B. die berechenbare partielle Funktion

$$f(x) = \begin{cases} 0^{n+1}, & x = 0^n \text{ für ein } n \geq 0 \\ \uparrow, & \text{sonst} \end{cases}$$

in $\mathcal{F}$ und die konstante Funktion $g(x) = 0$ nicht in $\mathcal{F}$ enthalten ist.

Der Satz von Rice

Satz (Satz von Rice)

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

Beweis

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

  $$w \in H_0 \Leftrightarrow M_{w'} \text{ berechnet (k)eine Funktion in } \mathcal{F}.$$

- 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 mit $u(x) = \uparrow$ 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 Funktion f berechnet, die genau dann die Eigenschaft $\mathcal{F}$ hat, wenn u sie nicht hat.
- Da $\mathcal{F}$ nicht trivial ist, gibt es eine DTM M, die eine solche Funktion f berechnet.

Beweis (Schluss)

- Da $\mathcal{F}$ nicht trivial ist, gibt es eine DTM M, die eine solche Funktion 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

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

Der Satz von Rice für Akzeptoren

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 $L_{\mathcal{S}} \in \{\varnothing, \{0,1\}^*\}$ ist.

Beweis

Siehe Übungen.

Das Postsche Korrespondenzproblem (PCP)

Definition

- Sei Σ ein beliebiges Alphabet mit $\# \notin \Sigma$.
- Das Postsche Korrespondenzproblem über Σ (kurz PCP_Σ) ist:
  gegeben: k Paare $(x_1, y_1), \ldots, (x_k, y_k)$ von Wörtern über Σ.
  gefragt: Gibt es eine Folge $\alpha = (i_1, \ldots, i_n)$, $n \geq 1$, von Indizes $i_j \in \{1, \ldots, k\}$ mit $x_{i_1} \ldots x_{i_n} = y_{i_1} \ldots y_{i_n}$?
- Das modifizierte PCP über Σ (kurz MPCP_Σ) fragt nach einer Lösung $\alpha = (i_1, \ldots, i_n)$ mit $i_1 = 1$.
- Wir notieren eine PCP-Instanz meist in Form einer Matrix $\binom{x_1 \ldots x_k}{y_1 \ldots y_k}$ und kodieren sie durch das Wort $x_1 \# y_1 \# \ldots \# x_k \# y_k$.

Beispiel

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

$x_1 x_3 x_2 x_3 = acaaabcaa$

$y_1 y_3 y_2 y_3 = acaaabcaa$

die PCP-Lösung $\alpha = (1, 3, 2, 3)$, die auch eine MPCP-Lösung ist. ◁

Das Postsche Korrespondenzproblem

Lemma

Für jedes Alphabet Σ gilt $\mathrm{PCP}_\Sigma \leq \mathrm{PCP}_{\{a,b\}}$.

Beweis

- Sei $\Sigma = \{a_1, \ldots, a_m\}$. Für ein Zeichen $a_i \in \Sigma$ sei $\hat{a}_i = ab^{i-1}$ und für ein Wort $w = w_1 \ldots w_n \in \Sigma^*$ mit $w_i \in \Sigma$ sei $\hat{w} = \hat{w}_1 \ldots \hat{w}_n$.
- Dann folgt $\mathrm{PCP}_\Sigma \leq \mathrm{PCP}_{\{a,b\}}$ mittels der Reduktionsfunktion

$$f : \begin{pmatrix} x_1 \ldots x_k \\ y_1 \ldots y_k \end{pmatrix} \mapsto \begin{pmatrix} \hat{x}_1 \ldots \hat{x}_k \\ \hat{y}_1 \ldots \hat{y}_k \end{pmatrix}.$$

□

Beispiel

Sei $\Sigma = \{0, 1, 2\}$. Dann ist $\hat{0} = a$, $\hat{1} = ab$ und $\hat{2} = abb$. Somit ist

$$f\begin{pmatrix} 0 & 01 & 200 \\ 020 & 12 & 00 \end{pmatrix} = \begin{pmatrix} a & aab & abbaa \\ aabba & ababb & aa \end{pmatrix}.$$

Das Postsche Korrespondenzproblem

Im Folgenden lassen wir im Fall $\Sigma = \{a, b\}$ den Index weg und schreiben einfach PCP (bzw. MPCP).

Satz

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

Beweis

- Wir zeigen $\mathrm{MPCP} \leq \mathrm{PCP}_\Sigma$ für $\Sigma = \{a, b, \langle, |, \rangle\}$.
- Für ein Wort $w = w_1 \dots w_n$ sei

$\overleftarrow{w}$	$\overleftarrow{w}$	$\overleftarrow{w}$	$\overleftarrow{w}$
$\langle w_1 \vert \dots \vert w_n \vert$	$\langle w_1 \vert \dots \vert w_n$	$\vert w_1 \vert \dots \vert w_n$	$w_1 \vert \dots \vert w_n \vert$

Das Postsche Korrespondenzproblem

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

- Wir zeigen $\mathrm{MPCP} \leq \mathrm{PCP}_\Sigma$ für $\Sigma = \{a, b, \langle, |, \rangle\}$.
- Für ein Wort $w = w_1 \dots w_n$ sei

$\overleftarrow{w}$	$\overrightarrow{w}$	$\overleftarrow{w}$	$\overrightarrow{w}$
$\langle w_1 \vert \dots \vert w_n \vert$	$w_1 \vert \dots \vert w_n \vert$	$\langle w_1 \vert \dots \vert w_n$	$\vert w_1 \vert \dots \vert w_n$

- Wir reduzieren MPCP mittels folgender Funktion f auf PCP_Σ:

$$f : \begin{pmatrix} x_1 & \dots & x_k \\ y_1 & \dots & y_k \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overrightarrow{x_1} & \dots & \overrightarrow{x_k} & \rangle \\ \overleftarrow{y_1} & \overrightarrow{y_1} & \dots & \overrightarrow{y_k} & \vert\rangle \end{pmatrix}$$

Beispiel

$$f : \begin{pmatrix} aa & b & bab & bb \\ aab & bb & a & b \end{pmatrix} \mapsto \begin{pmatrix} \langle a\vert a\vert & a\vert a\vert & b\vert & b\vert a\vert b\vert & b\vert b\vert & \rangle \\ \langle a\vert a\vert b & \vert a\vert a\vert b & \vert b\vert b & \vert a & \vert b & \vert\rangle \end{pmatrix}$$

◁

Das Postsche Korrespondenzproblem

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

- Wir zeigen $\mathrm{MPCP} \leq \mathrm{PCP}_\Sigma$ für $\Sigma = \{a, b, \langle, |, \rangle\}$.
- Für ein Wort $w = w_1 \dots w_n$ sei

$\overleftarrow{w}$	$\overrightarrow{w}$	$\overleftarrow{w}$	$\overrightarrow{w}$
$\langle w_1 \vert \dots \vert w_n \vert$	$w_1 \vert \dots \vert w_n \vert$	$\langle w_1 \vert \dots \vert w_n$	$\vert w_1 \vert \dots \vert w_n$

- Wir reduzieren MPCP mittels folgender Funktion f auf PCP_Σ:

$$f : \begin{pmatrix} x_1 & \dots & x_k \\ y_1 & \dots & y_k \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overrightarrow{x_1} & \dots & \overrightarrow{x_k} & \rangle \\ \overleftarrow{y_1} & \overrightarrow{y_1} & \dots & \overrightarrow{y_k} & \vert\rangle \end{pmatrix}$$

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

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

Das Postsche Korrespondenzproblem

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

- Für die umgekehrte Implikation sei $\alpha' = (i_1, \ldots, i_n)$ eine PCP-Lösung für

$$f(I) = \begin{pmatrix} \overleftarrow{x_1} & \overrightarrow{x_1} & \ldots & \overrightarrow{x_k} & \rangle \\ \overleftarrow{y_1} & \overrightarrow{y_1} & \ldots & \overrightarrow{y_k} & |\rangle \end{pmatrix}.$$

- Dann muss $i_1 = 1$ sein, da $(\overleftarrow{x_1}, \overleftarrow{y_1})$ das einzige Paar in $f(I)$ ist, bei dem beide Komponenten mit demselben Buchstaben anfangen.
- Zudem muss $i_n = k + 2$ sein, da nur das Paar $(\rangle, |\rangle)$ mit demselben Buchstaben aufhört.
- Wählen wir α' von minimaler Länge, so ist $i_j \in \{2, \ldots, k+1\}$ für $j = 2, \ldots, n-1$.
- Dann ist aber

$$\alpha = (i_1, i_2 - 1, \ldots, i_{n-1} - 1)$$

eine MPCP-Lösung für I. □

Das Postsche Korrespondenzproblem

Satz

PCP ist RE-vollständig und damit unentscheidbar.

Beweis.

- Es ist leicht zu sehen, dass $\mathrm{PCP} \in \mathrm{RE}$ ist.
- Sei $A \in \mathrm{RE}$ und sei $M = (Z, \Sigma, \Gamma, \delta, z_0, E)$ eine 1-DTM für A.
- Wir zeigen $A \leq \mathrm{MPCP}_{\Sigma'}$ für $\Sigma' = \Gamma \cup Z \cup \{\langle, |, \rangle\}$.
- Wegen $\mathrm{MPCP}_{\Sigma'} \leq \mathrm{PCP}$ folgt hieraus $A \leq \mathrm{PCP}$.

Beweisidee für die Reduktion $A \leq \mathrm{MPCP}_{\Sigma'}$:

Transformiere eine Eingabe $w \in \Sigma^*$ in eine Instanz $g(w) = \binom{x_1 \ldots x_k}{y_1 \ldots y_k}$, so dass $\alpha = (i_1, \ldots, i_n)$ genau dann eine MPCP-Lösung für $g(w)$ ist, wenn das zugehörige Lösungswort $x_{i_1} \ldots x_{i_n} = y_{i_1} \ldots y_{i_n}$ eine akzeptierende Rechnung von $M(w)$ kodiert. Dann gilt

$$w \in A \Leftrightarrow g(w) \in \mathrm{MPCP}_{\Sigma'}.$$

Das Postsche Korrespondenzproblem

Beweis von $A \leq \mathrm{MPCP}_{\Sigma'}$

Wir bilden $g(w)$ aus folgenden Wortpaaren:

1. $(\langle, \langle | z_0 w)$, „Startregel“
2. für alle $a \in \Gamma \cup \{|\}$: (a, a), „Kopierregeln“
3. für alle $a, a', b \in \Gamma$, $z, z' \in Z$:

$$
\begin{array}{lll}
(za, z'a'), & \text{falls } \delta(z,a) = (z', a', N), & \\
(za, a'z'), & \text{falls } \delta(z,a) = (z', a', R), & \\
(bza, z'ba'), & \text{falls } \delta(z,a) = (z', a', L), & \\
(|za, |z'\sqcup a'), & \text{falls } \delta(z,a) = (z', a', L), & \text{„Überführungsregeln“} \\
(bz|, z'ba'|), & \text{falls } \delta(z,\sqcup) = (z', a', L), & \\
(z|, z'a'|), & \text{falls } \delta(z,\sqcup) = (z', a', N), & \\
(z|, a'z'|), & \text{falls } \delta(z,\sqcup) = (z', a', R), &
\end{array}
$$

4. für alle $e \in E$, $a \in \Gamma$: (ae, e), (ea, e), „Löschregeln“
5. sowie für alle $e \in E$: $(e|\rangle, \rangle)$. „Abschlussregeln“

Das Postsche Korrespondenzproblem

Beispiel

- Betrachte die 1-DTM $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ mit $Z = \{q_0, \ldots q_4\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b, A, B, \sqcup\}$, $E = \{q_4\}$ und den Anweisungen

$$\begin{array}{lllll} \delta\colon & q_0 a \to q_1 AR \ (1) & q_1 a \to q_1 aR \ (2) & q_1 B \to q_1 BR \ (3) & q_1 b \to q_2 BL \ (4) \\ & q_2 a \to q_2 aL \ (5) & q_2 B \to q_2 BL \ (6) & q_2 A \to q_0 AR \ (7) & q_0 B \to q_3 BR \ (8) \\ & q_3 B \to q_3 BR \ (9) & q_3 \sqcup \to q_4 \sqcup N \ (10) & & \end{array}$$

- M akzeptiert die Eingabe ab wie folgt:

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{(10)}{\vdash} ABq_4 \sqcup$$

- Die MPCP-Instanz $g(ab)$ enthält für $u \in \Gamma$ die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle \mid q_0 ab)$	$(u, u), (\mid, \mid)$	$(q_4 u, q_4), (uq_4, q_4)$	$(q_4 \mid \rangle, \rangle)$

sowie folgende Überführungsregeln:

Das Postsche Korrespondenzproblem

Beispiel

Anweisung		zugehörige Überführungsregeln
$q_0 a \to q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 a \to q_1 aR$	(2)	$(q_1 a, aq_1)$
$q_1 B \to q_1 BR$	(3)	$(q_1 B, Bq_1)$
$q_1 b \to q_2 BL$	(4)	$(uq_1 b, q_2 uB)$, $(\vert q_1 b, \vert q_2 \sqcup B)$
$q_2 a \to q_2 aL$	(5)	$(uq_2 a, q_2 ua)$, $(\vert q_2 a, \vert q_2 \sqcup a)$
$q_2 B \to q_2 BL$	(6)	$(uq_2 B, q_2 uB)$, $(\vert q_2 B, \vert q_2 \sqcup B)$
$q_2 A \to q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \to q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 B \to q_3 BR$	(9)	$(q_3 B, Bq_3)$
$q_3 \sqcup \to q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup)$, $(q_3 \vert, q_4 \sqcup \vert)$

Beispiel

- Die MPCP-Instanz $g(ab)$ enthält für $u \in \Gamma$ die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle \mid q_0 ab)$	(u, u), $(\mid, \mid)$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4 \mid \rangle, \rangle)$

  sowie u.a. folgende Überführungsregeln:

$$
\begin{array}{lll}
q_0 a \to q_1 A R & (1) & (q_0 a, A q_1) \\
q_1 b \to q_2 B L & (4) & (u q_1 b, q_2 u B),\ (\mid q_1 b, \mid q_2 \sqcup B) \\
q_2 A \to q_0 A R & (7) & (q_2 A, A q_0) \\
q_0 B \to q_3 B R & (8) & (q_0 B, B q_3) \\
q_3 \sqcup \to q_4 \sqcup N & (10) & (q_3 \sqcup, q_4 \sqcup),\ (q_3 \mid, q_4 \sqcup \mid)
\end{array}
$$

- Der akzeptierenden Rechnung

$$
q_0 ab \underset{(1)}{\vdash} A q_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} A q_0 B \underset{(8)}{\vdash} AB q_3 \sqcup \underset{(10)}{\vdash} AB q_4 \sqcup
$$

  von $M(ab)$ entspricht dann das MPCP-Lösungswort

$$
\langle \mid q_0 ab \mid A q_1 b \mid q_2 AB \mid A q_0 B \mid AB q_3 \mid AB q_4 \sqcup \mid A q_4 \sqcup \mid q_4 \sqcup \mid q_4 \mid \rangle
$$
$$
\langle \mid q_0 ab \mid A q_1 b \mid q_2 AB \mid A q_0 B \mid AB q_3 \mid AB q_4 \sqcup \mid A q_4 \sqcup \mid q_4 \sqcup \mid q_4 \mid \rangle
$$

◁

Beweis von $A \leq \mathrm{MPCP}_{\Sigma'}$

- Nun lässt sich leicht aus einer akzeptierenden Rechnung

  $$K_0 = z_0 w \vdash K_1 \vdash \cdots \vdash K_t = uev$$

  mit $e \in E$ und $u, v \in \Gamma^*$ eine MPCP-Lösung mit einem Lösungswort der Form

  $$\langle \,|\, K_0 \,|\, K_1 \,|\, \ldots \,|\, K_t \,|\, K_{t+1} \,|\, \ldots \,|\, K_{t+|K_t|-1} \,|\, \rangle$$

  angeben, wobei K_{t+i} aus K_t durch Löschen von i Zeichen in der Nachbarschaft von e entsteht.
- Umgekehrt lässt sich aus jeder MPCP-Lösung auch eine akzeptierende Rechnung von M bei Eingabe w gewinnen, womit

  $$w \in L(M) \Leftrightarrow g(w) \in \mathrm{MPCP}_{\Sigma'}$$

  gezeigt ist. □

Das Postsche Korrespondenzproblem

Satz

PCP ist RE-vollständig und damit unentscheidbar.

Einfacherer Beweis über Typ-0 Grammatiken

- Sei $A \in \mathrm{RE}$ und sei $G = (V, \Sigma, P, S)$ eine Typ-0 Grammatik für A.
- Wir zeigen $A \leq \mathrm{MPCP}_\Gamma$ für $\Gamma = V \cup \Sigma \cup \{\langle, |, \rangle\}$.
- Wegen $\mathrm{MPCP}_\Gamma \leq \mathrm{PCP}$ folgt hieraus $A \leq \mathrm{PCP}$.

Beweisidee für die Reduktion $A \leq \mathrm{MPCP}_\Gamma$:

Transformiere eine Eingabe $w \in \Sigma^*$ in eine Instanz $f(w) = \binom{x_1 \ldots x_k}{y_1 \ldots y_k}$, so dass $\alpha = (i_1, \ldots, i_n)$ genau dann eine MPCP-Lösung für $f(w)$ ist, wenn das zugehörige Lösungswort $x_{i_1} \ldots x_{i_n} = y_{i_1} \ldots y_{i_n}$ eine Ableitung $S = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_m = w$ kodiert. Dann gilt

$$w \in A \Leftrightarrow f(w) \in \mathrm{MPCP}_\Gamma.$$

Das Postsche Korrespondenzproblem

Beweis von $A \leq \mathrm{MPCP}_\Gamma$

- Wir bilden $f(w)$ aus folgenden Wortpaaren:
  - $(\langle, \langle \mid S)$, „Startregel“
  - für jede Regel $l \to r$ in P: (l, r), „Ableitungsregeln“
  - für alle $a \in V \cup \Sigma \cup \{\mid\}$: (a, a), „Kopierregeln“
  - sowie das Paar $(w \mid \rangle, \rangle)$ „Abschlussregel“
- Nun lässt sich leicht aus einer Ableitung $S = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_m = w$ von w in G eine MPCP-Lösung mit dem Lösungswort

  $$\langle \mid \alpha_0 \mid \alpha_1 \mid \ldots \mid \alpha_m \mid \rangle$$

  angeben.
- Umgekehrt lässt sich aus jeder MPCP-Lösung auch eine Ableitung von w in G gewinnen, womit

  $$w \in L(M) \Leftrightarrow f(w) \in \mathrm{MPCP}_\Gamma$$

  gezeigt ist. □

Das Postsche Korrespondenzproblem

Beispiel

- Betrachte die Grammatik $G = (\{S\}, \{a, b\}, \{S \to aSbS, \varepsilon\}, S)$ und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die MPCP-Instanz $f(aabb)$ enthält dann für $u \in \{S, a, b, |\}$ die Wortpaare

Startregel	Ableitungsregeln	Kopierregeln	Abschlussregel
$(\langle, \langle \mid S)$	$(S, aSbS), (S, \varepsilon)$	(u, u)	$(aabb \mid \rangle, \rangle)$

- Obiger Ableitung entspricht dann folgendes MPCP-Lösungswort für $f(aabb)$:

$$\langle \mid S \mid aSbS \mid aaSbSbS \mid aaSbbS \mid aabbS \mid aabb \mid \rangle$$
$$\langle \mid S \mid aSbS \mid aaSbSbS \mid aaSbbS \mid aabbS \mid aabb \mid \rangle$$

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

$$\langle \mid S \mid aSbS \mid aaSbSb \mid aabb \mid \rangle$$
$$\langle \mid S \mid aSbS \mid aaSbSb \mid aabb \mid \rangle$$

- Dieses entspricht der „parallelisierten“ Ableitung

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

◁

Das Schnittproblem für CFL ist unentscheidbar

Das Schnittproblem für kontextfreie Grammatiken

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.

Beweis

- Es ist leicht zu sehen, dass das Problem semi-entscheidbar ist.
- Um PCP auf dieses Problem zu reduzieren, betrachte für eine Folge $s = (x_1, \ldots, x_k)$ von Strings $x_i \in \{a, b\}^*$ die Sprache

$$L_s = \{i_n \ldots i_1 x_{i_1} \cdots x_{i_n} \mid n \geq 1, 1 \leq i_1, \ldots, i_n \leq k\}.$$

- L_s wird von der Grammatik $G_s = (\{S\}, \{a, b, 1, \ldots, k\}, P_s, S)$ erzeugt mit

$$P_s\colon S \to 1Sx_1, \ldots, kSx_k, 1x_1, \ldots, kx_k$$

Das Schnittproblem für CFL ist unentscheidbar

Reduktion von PCP auf das Schnittproblem für CFL

- Zu einer PCP-Instanz $I = \binom{x_1 \ldots x_k}{y_1 \ldots y_k}$ bilden wir das Paar (G_s, G_t), wobei $s = (x_1, \ldots, x_k)$ und $t = (y_1, \ldots, y_k)$ ist.
- Dann ist $L(G_s) \cap L(G_t)$ die Sprache

$$\{i_n \ldots i_1 x_{i_1} \ldots x_{i_n} \mid 1 \leq n,\ x_{i_1} \ldots x_{i_n} = y_{i_1} \ldots y_{i_n}\}.$$

- Folglich ist $\alpha = (i_1, \ldots, i_n)$ genau dann eine Lösung für I, wenn $i_n \ldots i_1 x_{i_1} \ldots x_{i_n} \in L(G_s) \cap L(G_t)$ ist.
- Also vermittelt $f : I \mapsto (G_s, G_t)$ eine Reduktion von PCP auf das Schnittproblem für CFL. □

Das Schnittproblem für CFL ist unentscheidbar

Beispiel

- Die PCP-Instanz

$$I = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} = \begin{pmatrix} a & aab & abbaa \\ aabba & ababb & aa \end{pmatrix}$$

wird auf das Grammatikpaar (G_s, G_t) mit folgenden Regeln reduziert:

$$P_s\colon S \to 1Sa,\ 2Saab,\ 3Sabbaa,\ 1a,\ 2aab,\ 3abbaa,$$

$$P_t\colon S \to 1Saabba,\ 2Sababb,\ 3Saa,\ 1aabba,\ 2ababb,\ 3aa.$$

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

$$3231x_1x_3x_2x_3 = 3231aabbaaaababbaa = 3231aabbaaaababbaa = 3231y_1y_3y_2y_3$$

im Schnitt $L(G_s) \cap L(G_t)$. ◁

Das Schnitt- und das Inklusionsproblem für DCFL

Das Inklusionsproblem für DPDAs

Gegeben: Zwei DPDAs M_1 und M_2.

Gefragt: Gilt $L(M_1) \subseteq L(M_2)$?

Korollar

1. Das Schnittproblem für DPDAs ist RE-vollständig.
2. Das Inklusionsproblem für DPDAs ist co-RE-vollständig.

Beweisidee.

Die kontextfreie Grammatik G_s in obigem Beweis lässt sich leicht in DPDAs M_s und $\overline{M}_s$ überführen mit $L(M_s) = L(G_s)$ und $L(\overline{M}_s) = \overline{L(G_s)}$ (siehe Übungen).

Weitere Unentscheidbarkeitsresultate für CFL

Korollar

Für kontextfreie Grammatiken sind folgende Probleme unentscheidbar:

1. Ist $L(G) = \Sigma^*$? (Ausschöpfungsproblem)
2. Ist $L(G_1) = L(G_2)$? (Äquivalenzproblem)
3. Ist G mehrdeutig? (Mehrdeutigkeitsproblem)

1 Das Ausschöpfungsproblem für CFL ist co-RE-vollständig

Wir reduzieren das Komplement von PCP auf das Ausschöpfungsproblem für kontextfreie Grammatiken. Es gilt

$$I \notin \mathrm{PCP} \Leftrightarrow L_s \cap L_t = \varnothing \Leftrightarrow \overline{L}_s \cup \overline{L}_t = \Sigma^*.$$

Daher vermittelt die Funktion $f : I \mapsto G$, wobei G eine kontextfreie Grammatik mit

$$L(G) = \overline{L}_s \cup \overline{L}_t$$

ist, die gewünschte Reduktion. □

Weitere Unentscheidbarkeitsresultate für CFL

Korollar

Für kontextfreie Grammatiken sind folgende Probleme unentscheidbar:

1. Ist $L(G) = \Sigma^*$? (Ausschöpfungsproblem)
2. Ist $L(G_1) = L(G_2)$? (Äquivalenzproblem)
3. Ist G mehrdeutig? (Mehrdeutigkeitsproblem)

2 Das Äquivalenzproblem für CFL ist co-RE-vollständig

Wir reduzieren das Ausschöpfungsproblem für CFL auf das Äquivalenzproblem für CFL.

Dies leistet beispielsweise die Reduktionsfunktion

$$f : G \mapsto (G, G_{all}),$$

wobei G_{all} eine kontextfreie Grammatik mit $L(G_{all}) = \Sigma^*$ ist. □

Weitere Unentscheidbarkeitsresultate für CFL

3 Das Mehrdeutigkeitsproblem ist RE-vollständig

- Wir reduzieren PCP auf dieses Problem.
- Hierzu überführen wir eine PCP-Instanz $I = \binom{x_1 \ldots x_k}{y_1 \ldots y_k}$ in die Grammatik

$$G = (\{S, A, B\}, \{a, b, 1, \ldots, k\}, P_1 \cup P_2 \cup \{S \to A, S \to B\}, S)$$

mit den Regeln

$$P_1\colon A \to 1Ax_1, \ldots, kAx_k, 1x_1, \ldots, kx_k,$$

$$P_2\colon B \to 1By_1, \ldots, kBy_k, 1y_1, \ldots, ky_k.$$

- Da alle von A oder B ausgehenden Ableitungen eindeutig sind, ist G genau dann mehrdeutig, wenn es ein Wort $w \in L(G)$ gibt mit

$$S \Rightarrow A \Rightarrow^* w \quad \text{und} \quad S \Rightarrow B \Rightarrow^* w.$$

- Wie wir gesehen haben, ist dies genau dann der Fall, wenn I eine PCP-Lösung hat. □

Ein Unentscheidbarkeitsresultat für DCSL

Das Leerheitsproblem für DLBAs

Gegeben: Ein DLBA M.
Gefragt: Ist $L(M) = \varnothing$?

Satz

Das Leerheitsproblem für DLBAs ist co-RE-vollständig.

Beweis.

- Wir reduzieren das Ausschöpfungsproblem für CFL auf das Leerheitsproblem für DLBAs.
- Eine kontextfreie Grammatik G lässt sich wie folgt in einen DLBA M mit $L(M) = \overline{L(G)}$ überführen:
  - Bestimme zunächst einen DLBA M mit $L(M) = L(G)$.
  - Konstruiere daraus einen DLBA $\overline{M}$ mit $L(\overline{M}) = \overline{L(M)}$.
- Dann gilt $L(G) = \Sigma^* \Leftrightarrow L(M) = \varnothing$, d.h. die Funktion $f : G \mapsto \overline{M}$ berechnet die gewünschte Reduktion. □

Entscheidbare Probleme

Dagegen ist es nicht schwer,

- für eine kontextfreie Grammatik G zu entscheiden, ob mindestens ein Wort in G ableitbar ist (Leerheitsproblem für CFL), und
- für eine kontextsensitive Grammatik G und ein Wort x zu entscheiden, ob x in G ableitbar ist (Wortproblem für CSL).

Satz

- Das Leerheitsproblem für CFL ist entscheidbar.
- Das Wortproblem für CSL ist entscheidbar.

Beweis.

Siehe Übungen. □

Überblick der (Un-)Entscheidbarkeitsresultate

In folgender Tabelle fassen wir nochmals zusammen, wie schwierig die betrachteten Entscheidungsprobleme für die verschiedenen Stufen der Chomsky-Hierarchie sind.

	Wort-problem $x \in L$?	Leerheits-problem $L = \varnothing$?	Aus-schöpfung $L = \Sigma^*$?	Äquivalenz-problem $L_1 = L_2$?	Inklusions-problem $L_1 \subseteq L_2$	Schnitt-problem $L_1 \cap L_2 \neq \varnothing$?
REG	ja	ja	ja	ja	ja	ja
DCFL	ja	ja	ja	ja[a]	nein	nein
CFL	ja	ja	nein	nein	nein	nein
DCSL	ja	nein	nein	nein	nein	nein
CSL	ja	nein	nein	nein	nein	nein
RE	nein	nein	nein	nein	nein	nein

[a] Bewiesen in 1997 von Géraud Sénizergues (Univ. Bordeaux).

Die arithmetische Hierarchie

Frage

Wie kann man den Grad der Unentscheidbarkeit von unentscheidbaren Problemen messen?

Definition

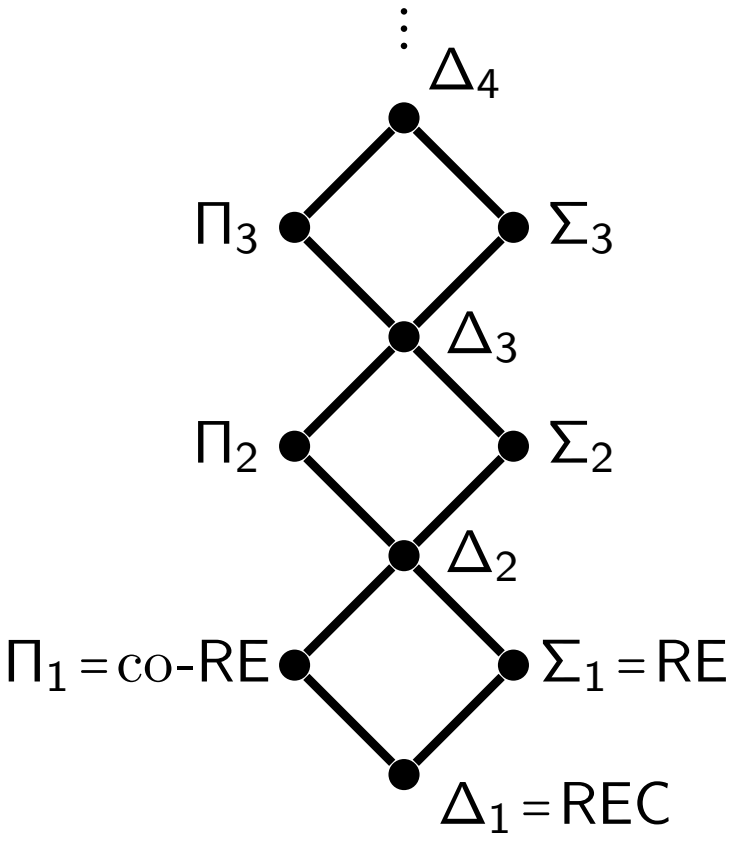
- Sei $A \subseteq \Sigma^*$ eine Sprache und $\mathcal{K}$ eine Sprachklasse. Dann ist

$$\exists A = \{x \in \Sigma^* \mid \exists y \in \{0,1\}^* : x\#y \in A\}$$

und

$$\exists \mathcal{K} = \mathcal{K} \cup \{\exists A \mid A \in \mathcal{K}\}.$$

- Weiter sei $\Sigma_0 = \mathrm{REC}$, sowie $\Pi_i = \text{co-}\Sigma_i$, $\Delta_i = \Sigma_i \cap \Pi_i$ und $\Sigma_{i+1} = \exists \Pi_i$ für $i \geq 0$.
- Die Klassen $\Sigma_i, \Pi_i, \Delta_i$, $i \geq 0$, bilden die Stufen der arithmetischen Hierarchie.



Die arithmetische Hierarchie

Proposition

- $\Sigma_1 = \text{RE}$, $\Pi_1 = \text{co-RE}$.
- $\Delta_0 = \Sigma_0 = \Pi_0 = \Delta_1 = \text{REC}$.
- $\Sigma_i \cup \Pi_i \subseteq \Delta_{i+1}$ für $i \geq 0$.

Bemerkung

Mittels Diagonalisierung kann man zeigen, dass $\Sigma_i \neq \Pi_i$ für alle $i \geq 1$ gilt. Dies impliziert, dass die Inklusionen $\Sigma_i \subseteq \Delta_{i+1} \subseteq \Sigma_{i+1}$ und $\Pi_i \subseteq \Delta_{i+1} \subseteq \Pi_{i+1}$ für alle $i \geq 1$ echt sind.

Die arithmetische Hierarchie

Beweis von $\Sigma_1 \subseteq \text{RE}$

Sei $A \in \Sigma_1$ und sei $A = \exists B$ für eine Sprache $B \in \text{REC}$.
Dann wird A von einer DTM akzeptiert, die bei Eingabe x systematisch nach einem String $y \in \{0,1\}^*$ mit $x\#y \in B$ sucht. Folglich ist $A \in \text{RE}$. □

Beweis von $\text{RE} \subseteq \Sigma_1$

Sei $A \subseteq \Sigma^*$ eine Sprache in RE und sei $A = L(M)$ für eine DTM M.
Dann ist $A = \exists B$ für die Sprache

$$B = \{x\#y \mid x \in \Sigma^* \text{ und } y \text{ kodiert eine akz. Rechnung von } M(x)\}.$$

Da B entscheidbar ist, folgt $A \in \exists\text{REC}$. □

Beweis von $\Pi_1 = \text{co-RE}$

Folgt wegen $\Pi_1 = \text{co-}\Sigma_1$ direkt aus $\Sigma_1 = \text{RE}$. □

Beweis von $\Delta_0 = \Sigma_0 = \Pi_0 = \Delta_1 = \text{REC}$

Klar, da $\text{REC} = \text{co-REC}$ und $\text{REC} = \text{RE} \cap \text{co-RE}$. □

Beweis von $\Pi_i \subseteq \Sigma_{i+1}$ und $\Sigma_i \subseteq \Pi_{i+1}$

Klar, da $\Pi_i \subseteq \exists\Pi_i = \Sigma_{i+1}$ und somit $\Sigma_i = \text{co-}\Pi_i \subseteq \text{co-}\Sigma_{i+1} = \Pi_{i+1}$. □

Beweis von $\Sigma_i \subseteq \Sigma_{i+1}$

Dies zeigen wir induktiv.

$i = 0$: Es gilt $\Sigma_0 = \text{REC} \subseteq \text{RE} = \Sigma_1$.

$i - 1 \rightsquigarrow i$: Nach IV gilt $\Sigma_{i-1} \subseteq \Sigma_i$. Folglich ist $\Pi_{i-1} \subseteq \Pi_i$. Dies wiederum impliziert $\exists\Pi_{i-1} \subseteq \exists\Pi_i$, also $\Sigma_i \subseteq \Sigma_{i+1}$. □

Beweis von $\Sigma_i, \Pi_i \subseteq \Delta_{i+1}$

Klar, da $\Delta_{i+1} = \Sigma_{i+1} \cap \Pi_{i+1}$. □

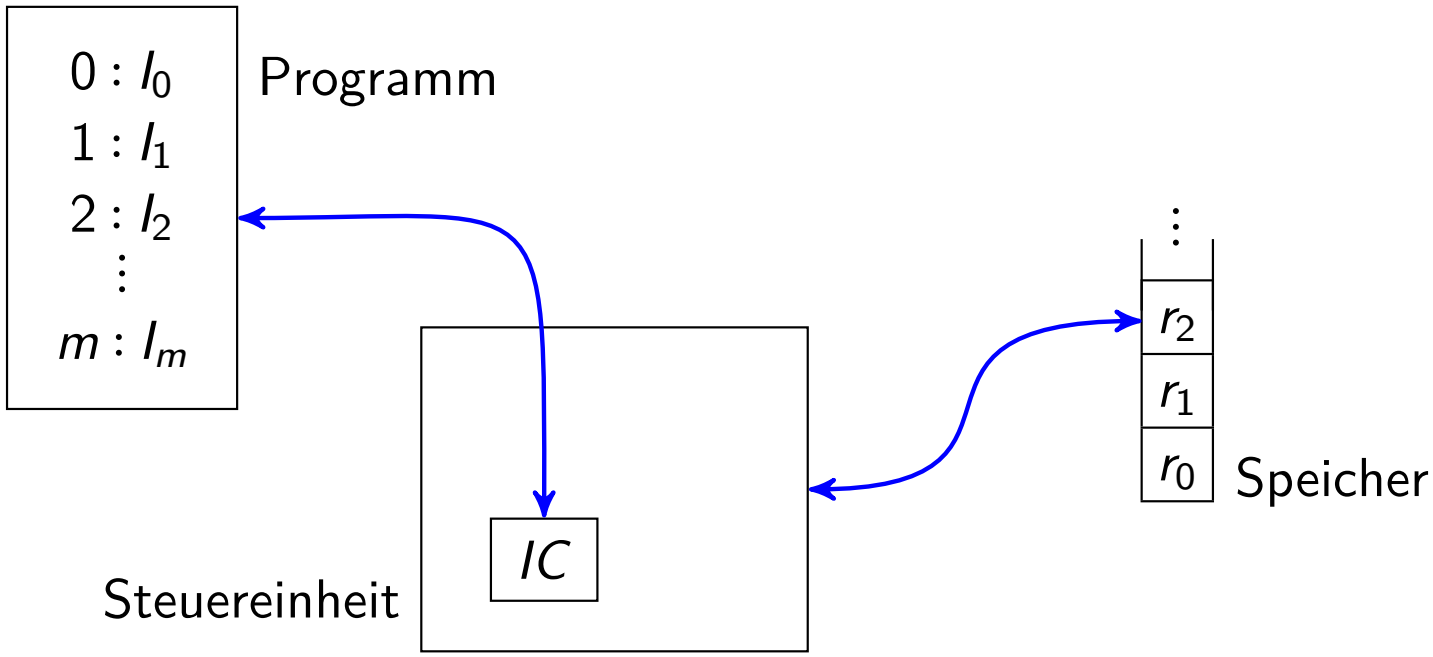
Überblick der (Un-)Entscheidbarkeitsresultate

Die betrachteten Entscheidungsprobleme für die verschiedenen Stufen der Chomsky-Hierarchie lassen sich nun wie folgt in die arithmetische Hierarchie einordnen.

	Wort-problem $x \in L$?	Leerheits-problem $L = \varnothing$?	Aus-schöpfung $L = \Sigma^*$?	Äquivalenz-problem $L_1 = L_2$?	Inklusions-problem $L_1 \subseteq L_2$	Schnitt-problem $L_1 \cap L_2 \neq \varnothing$?
REG	REC	REC	REC	REC	REC	REC
DCFL	REC	REC	REC	REC	co-RE	RE
CFL	REC	REC	co-RE	co-RE	co-RE	RE
DCSL	REC	co-RE	co-RE	co-RE	co-RE	RE
CSL	REC	co-RE	co-RE	co-RE	co-RE	RE
RE	RE	co-RE	Π_2	Π_2	Π_2	RE

Tatsächlich sind alle betrachteten Entscheidungsprobleme sogar vollständig für die angegebenen Sprachklassen.

Die Registermaschine (random access machine, RAM)



- führt ein Programm $P = l_0, \ldots, l_m$ aus, das aus einer endlichen Folge von Befehlen (instructions) l_i 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, die beliebig große natürliche Zahlen aufnehmen können.

Eine Programmiersprache für RAMs

In GOTO-Programmen sind folgende Befehle zulässig ($i, j, c \in \mathbb{N}$):

Befehl	Semantik
$r_i := r_j + c$	setzt Register r_i auf den Wert $r_j + c$
$r_i := r_j \dot{-} c$	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

Falls nichts anderes angegeben ist, wird zudem IC auf den Wert $IC + 1$ gesetzt.

GOTO-Berechenbarkeit

Definition

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

Wird P auf einer RAM mit den Werten $r_i = n_i$ für $i = 1, \ldots, k$, sowie $IC = 0$ und $r_i = 0$ für $i = 0, k+1, k+2, \ldots$ gestartet, so gilt:

- P hält genau dann, wenn $(n_1, \ldots, n_k) \in dom(f)$ ist, und
- falls P hält, hat r_0 nach Beendigung von P den Wert $f(n_1, \ldots, n_k)$.

Beispiel

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

```
0  IF r1 = 0 THEN GOTO 4
1  r1 := r1 ∸ 1
2  r0 := r0 + r2 GOTO 5
3  GOTO 0
4  HALT
5  r3 := r2
6  IF r3 = 0 THEN GOTO 3
7  r3 := r3 ∸ 1
8  r0 := r0 + 1
9  GOTO 6
```

◁

DTMs und GOTO-Programme sind gleichmächtig

- Da DTMs auf Wörtern und GOTO-Programme auf Zahlen operieren, müssen wir Wörter durch Zahlen (und umgekehrt) kodieren.
- Sei $\Sigma = \{a_0, \ldots, a_{m-1}\}$ ein Alphabet. Dann können wir jedes Wort $x = a_{i_1} \ldots a_{i_n} \in \Sigma^*$ durch eine natürliche Zahl $num_\Sigma(w)$ kodieren:

$$num_\Sigma(x) = \sum_{j=0}^{n-1} m^j + \sum_{j=1}^{n} i_j m^{n-j} = \frac{m^n - 1}{m-1} + (i_1 \ldots i_n)_m.$$

- Da die Abbildung $num_\Sigma : \Sigma^* \to \mathbb{N}$ bijektiv ist, können wir umgekehrt jede natürliche Zahl n durch das Wort $str_\Sigma(n) = num_\Sigma^{-1}(n)$ kodieren.

Beispiel

Für das Alphabet $\Sigma = \{a, b, c\}$ erhalten wir folgende Kodierung:

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

Äquivalenz von Turing- und GOTO-Berechenbarkeit

- Ist $\Sigma = \{0,1\}$, so lassen wir den Index weg und schreiben einfach *num* und *str* anstelle von num_Σ und str_Σ.
- Zudem erweitern wir die Kodierungsfunktion $str : \mathbb{N} \to \{0,1\}^*$ zu einer Kodierungsfunktion $str_k : \mathbb{N}^k \to \{0,1,\#\}^*$ wie folgt:

$$str_k(n_1,\ldots,n_k) = str(n_1)\#\ldots\#str(n_k).$$

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

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

- Es ist klar, dass f durch $\hat{f}$ eindeutig bestimmt ist. Wir nennen f die numerische Repräsentation von $\hat{f}$.

Satz

Sei f die numerische Repräsentation einer partiellen Funktion $\hat{f}$.
Dann ist f genau dann GOTO-berechenbar, wenn $\hat{f}$ berechenbar ist.

Simulation eines GOTO-Programms durch eine DTM

- Sei $\hat{f}$ eine partielle Funktion, deren numerische Repräsentation f von einem GOTO-Programm P auf einer RAM R berechnet wird.
- Dann existiert eine Zahl k', so dass P nur Register r_i mit $i \leq k'$ benutzt.
- Daher lässt sich eine Konfiguration von R durch Angabe der Inhalte des Befehlszählers IC und der Register $r_0, \ldots, r_{k'}$ beschreiben.
- Wir konstruieren eine $(k'+2)$-DTM M, die
  - den Inhalt von IC in ihrem Zustand,
  - die Registerwerte $r_1, \ldots, r_{k'}$ auf den Bändern $1, \ldots, k'$ und
  - den Wert von r_0 auf dem Ausgabeband $k'+2$ speichert.
- Ein Registerwert r_i wird hierbei in der Form $str(r_i)$ gespeichert.
- Band $k'+1$ wird zur Ausführung von Hilfsberechnungen benutzt.

Simulation eines GOTO-Programms durch eine DTM

- Die Aufgabe von M ist es, bei Eingabe $w = str_k(n_1, \ldots, n_k)$ das Wort $str(f(n_1, \ldots, n_k))$ auszugeben, wenn $(n_1, \ldots, n_k) \in dom(f)$ ist, und andernfalls nicht zu halten.
- Zuerst kopiert M die Teilwörter $str(n_i)$ für $i = 2, \ldots, k$ auf das i-te Band und löscht auf dem 1. Band alle Eingabezeichen bis auf $str(n_1)$.
- Da das leere Wort den Wert $num(\varepsilon) = 0$ kodiert, sind nun auf den Bändern $1, \ldots, k'$ und auf Band $k' + 2$ die der Startkonfiguration von R bei Eingabe $(n_1, \ldots, n_k)$ entsprechenden Registerinhalte gespeichert.
- Danach führt M das Programm P Befehl für Befehl aus.
- Es ist leicht zu sehen, dass sich jeder Befehl I in P durch eine Folge von Anweisungen realisieren lässt, die die auf den Bändern gespeicherten Registerinhalte bzw. den im Zustand von M gespeicherten Wert von IC entsprechend modifizieren.
- Sobald P stoppt, hält auch M und gibt das Wort $str(r_0) = str(f(n_1, \ldots, n_k))$ aus.

Simulation einer DTM durch ein GOTO-Programm

- Sei $f : \mathbb{N}^k \to \mathbb{N} \cup \{\uparrow\}$ eine partielle Funktion und sei $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ eine DTM mit Eingabealphabet $\Sigma = \{0, 1, \#\}$, die die zugehörige Wortfunktion $\hat{f} : \Sigma^* \to \{0,1\}^* \cup \{\uparrow\}$ berechnet.
- M gibt also bei Eingabe $str_k(n_1, \ldots, n_k)$ das Wort $str(f(n_1, \ldots, n_k))$ aus, falls $f(n_1, \ldots, n_k)$ definiert ist, und hält andernfalls nicht.
- Wir konstruieren ein GOTO-Programm P, das bei Eingabe $(n_1, \ldots, n_k)$ die DTM M bei Eingabe $w = str_k(n_1, \ldots, n_k)$ simuliert und im Fall, dass $M(w)$ hält, den Wert $num(M(w)) = f(n_1, \ldots, n_k)$ berechnet.
- Wir können annehmen, dass M eine 1-DTM ist.
- Sei $Z = \{q_0, \ldots, q_r\}$ und $\Gamma = \{a_0, \ldots, a_{m-1}\}$, wobei wir annehmen, dass $a_0 = \sqcup$, $a_1 = 0$, $a_2 = 1$ und $a_3 = \#$ ist.

Simulation einer DTM durch ein GOTO-Programm

- Eine Konfiguration $K = uq_iv$ von M wird wie folgt in den Registern r_0, r_1, r_2 gespeichert. Sei $u = a_{i_1} \dots a_{i_n}$ und $v = a_{j_1} \dots a_{j_{n'}}$.
  - $r_0 = (i_1 \dots i_n)_m$,
  - $r_1 = i$,
  - $r_2 = (j_{n'} \dots j_1)_m$.
- P besteht aus 3 Programmteilen $P = P_1, P_2, P_3$.
- P_1 berechnet in Register r_2 die Zahl $(j_n \dots j_1)_m$, wobei $(j_1, \dots, j_n)$ die Indexfolge der Zeichen von $str_k(n_1, \dots, n_k) = a_{j_1} \dots a_{j_n}$ ist.
- Die übrigen Register setzt P_1 auf den Wert 0.
- P_1 stellt also die Startkonfiguration $K_w = q_0w$ von M bei Eingabe $w = str_k(n_1, \dots, n_k)$ in den Registern r_0, r_1, r_2 her.
- Anschließend führt P_2 eine schrittweise Simulation von M aus.
- Hierzu überführt P_2 solange die in r_0, r_1, r_2 gespeicherte Konfiguration von M in die zugehörige Nachfolgekonfiguration, bis M hält.

Simulation einer DTM durch ein GOTO-Programm

- Das Programmstück P_2 hat die Form

M_2 $r_3 := r_2$ **MOD** m
IF $r_1 = 0 \wedge r_3 = 0$ **THEN GOTO** $M_{0,0}$
$\vdots$
IF $r_1 = r \wedge r_3 = m-1$ **THEN GOTO** $M_{r,m-1}$

- Die Befehle ab Position $M_{i,j}$ hängen von $\delta(q_i, a_j)$ ab. Wir betrachten exemplarisch den Fall $\delta(q_i, a_j) = \{(q_{i'}, a_{j'}, L)\}$:

$M_{i,j}$ $r_1 := i'$
$r_2 := r_2$ **DIV** m
$r_2 := r_2 m + j'$
$r_2 := r_2 m + (r_0$ **MOD** $m)$
$r_0 := r_2$ **DIV** m
GOTO M_2

- Im Fall $\delta(q_i, a_j) = \varnothing$ erfolgt ein Sprung an den Beginn von P_3.
- P_3 transformiert den Inhalt $r_0 = (j_1 \dots j_n)_m$ von Register r_0 in die Zahl $num(a_{j_1} \dots a_{j_n})$ und hält.

WHILE- und LOOP-Programme

- Die Syntax von WHILE-Programmen ist induktiv wie folgt definiert ($i, j, c \in \mathbb{N}$):
  - Jede Wertzuweisung der Form $x_i := x_j + c$ oder $x_i := x_j \dot{-} c$ 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 sooft ausgeführt, wie der Wert von x_i zu Beginn der Schleife angibt.

WHILE- und LOOP-Berechenbarkeit

- Eine partielle Funktion $f : \mathbb{N}^k \to \mathbb{N} \cup \{\uparrow\}$ heißt WHILE-berechenbar, falls es ein WHILE-Programm P mit folgender Eigenschaft gibt:

  Wird P mit den Werten $x_i = n_i$ für $i = 1, \ldots, k$ gestartet, so gilt:

  - P hält genau dann, wenn $(n_1, \ldots, n_k) \in dom(f)$ ist, und
  - falls P hält, hat x_0 nach Beendigung von P den Wert $f(n_1, \ldots, n_k)$.
- Die LOOP-Berechenbarkeit von f ist entsprechend definiert.

Beispiel

Die Funktion $f(x_1, x_2) = x_1 x_2$ wird von dem WHILE-Programm

WHILE $x_1 \neq 0$ **DO**
  $x_0 := x_0 + x_2$;
  $x_1 := x_1 \dot{-} 1$
END

sowie von folgendem LOOP-Programm berechnet:

LOOP x_1 **DO** $x_0 := x_0 + x_2$ **END**

◁

Äquivalenz von WHILE- und GOTO-Berechenbarkeit

Satz

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

Simulation von WHILE- durch GOTO-Programme

- Sei P ein WHILE-Programm, das f 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 $\vdots$

- 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.

Simulation von GOTO- durch WHILE-Programme

- Sei $P = (I_0, \ldots, I_m)$ ein GOTO-Programm, das f 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'_j abhängig vom Befehl I_j folgendes WHILE-Programm:
  - $r_i := r_k + c$: $x_i := x_k + c$; $x_z := x_z + 1$,
  - $r_i := r_k \dot{-} c$: $x_i := x_k \dot{-} c$; $x_z := x_z + 1$,
  - **GOTO** k: $x_z := k$,
  - **IF** $r_i = c$ **THEN GOTO** k:
    IF $x_i = c$ **THEN** $x_z := k$ **ELSE** $x_z := x_z + 1$ **END**
  - **HALT**: $x_z := m + 1$.
- Man beachte, dass P' nur eine WHILE-Schleife enthält.

Vergleich von LOOP- und WHILE-Berechenbarkeit

- Es ist leicht zu sehen, dass sich jedes LOOP-Programm durch ein WHILE-Programm simulieren lässt.
- Offensichtlich können LOOP-Programme nur totale Funktionen berechnen.
- Daher kann nicht jedes WHILE-Programm durch ein LOOP-Programm simuliert werden.
- Mittels Diagonalisierung lässt sich eine totale WHILE-berechenbare Funktion f angeben, die nicht LOOP-berechenbar ist.
- Ein bekanntes Beispiel einer totalen WHILE-berechenbaren Funktion, die nicht LOOP-berechenbar ist, ist die Ackermannfunktion $a(x, y)$, die induktiv wie folgt definiert ist:

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