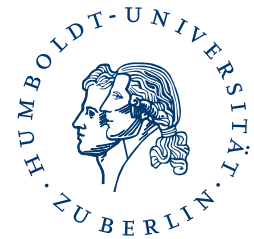
Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2016/17

Entscheidbare und semi-entscheidbare Sprachen

Definition

- Eine NTM M hält bei Eingabe x (kurz: $M(x) = \downarrow$ oder $M(x)\downarrow$), falls alle Rechnungen von $M(x)$ eine endliche Länge haben.
- Falls $M(x)$ nicht hält, schreiben wir auch kurz $M(x) = \uparrow$ oder $M(x)\uparrow$.
- 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
- $\mathrm{REG} \subsetneq \mathrm{DCFL} \subsetneq \mathrm{CFL} \subsetneq \mathrm{DCSL} \subseteq \mathrm{CSL} \subsetneq \mathrm{REC} \subsetneq \mathrm{RE}$.

Dass CSL echt in REC enthalten ist, wird in den Übungen gezeigt. Beispiele für interessante semi-entscheidbare Sprachen, die nicht entscheidbar sind, werden wir in der Vorlesung kennenlernen.

Berechenbarkeit von partiellen 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	$\cdots$
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.
- Da f_4 keine totale Funktion ist, gehört f_4 nicht zu FREC. ◁

Berechenbarkeit von charakteristischen Funktionen

Definition

Für eine Sprache $A \subseteq \Sigma^*$ sind die charakteristische Funktion χ_A und die partielle charakteristische Funktion $\hat{\chi}_A$ wie folgt definiert:

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \quad \text{und} \quad \hat{\chi}_A(x) = \begin{cases} 1, & x \in A \\ \uparrow, & x \notin A \end{cases}$$

Satz

- Eine Sprache $A \subseteq \Sigma^*$ ist genau dann entscheidbar, wenn ihre charakteristische Funktion χ_A berechenbar ist (siehe Übungen).
- Eine Sprache $A \subseteq \Sigma^*$ ist genau dann semi-entscheidbar, falls ihre partielle charakteristische Funktion $\hat{\chi}_A$ berechenbar ist.

Berechenbarkeit von charakteristischen Funktionen

Satz

Folgende Eigenschaften sind äquivalent:

1. A ist semi-entscheidbar (d.h. A wird von einer DTM akzeptiert),
2. $\hat{\chi}_A$ ist berechenbar (d.h. $\hat{\chi}_A \in \mathrm{FREC}_p$),
3. es gibt eine berechenbare partielle Funktion $f \in \mathrm{FREC}_p$ mit $A = dom(f)$ (d.h. es gibt eine DTM M mit $A = \{x \in \Sigma^* \mid M(x)\downarrow\}$).

Beweis

1 $\Rightarrow$ 2 Sei M eine DTM mit $L(M) = A$. Dann lässt sich M so modifizieren, dass sie eine 1 ausgibt (und hält), sobald sie einen Endzustand erreicht, und in eine Endlosschleife geht, sobald sie eine Konfiguration ohne Folgekonfiguration erreicht.

2 $\Rightarrow$ 3 Klar, da der Definitionsbereich $dom(\hat{\chi}_A)$ von $\hat{\chi}_A$ die Sprache A ist.

3 $\Rightarrow$ 1 Sei M eine DTM, die eine partielle Funktion f mit $dom(f) = A$ berechnet. Dann lässt sich M so modifizieren, dass sie genau dann in einen Endzustand übergeht, wenn M eine Konfiguration ohne Folgekonfiguration erreicht. □

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 ist vom Typ 0,
4. A wird von einer NTM akzeptiert,
5. A ist rekursiv aufzählbar.

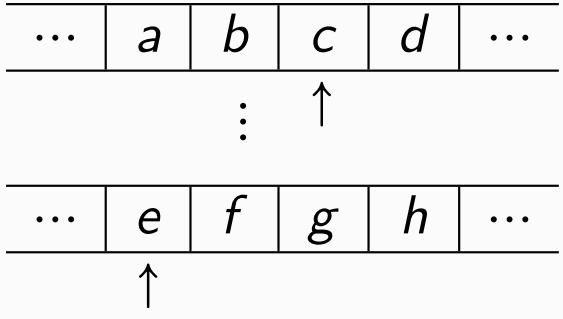
Beweis

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

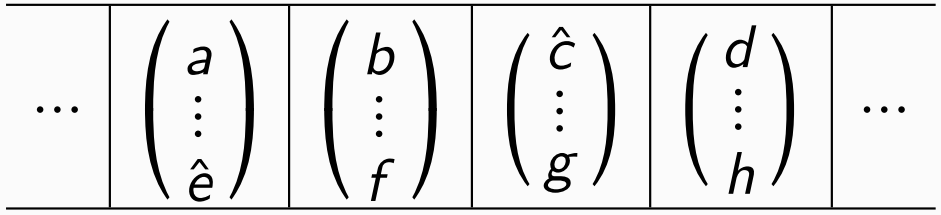
Simulation einer k-DTM durch eine 1-DTM

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.
  - 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 $L(M') = L(M)$. □

Charakterisierung der rekursiv aufzählbaren Sprachen

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

- 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):

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

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

Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von 5 $\Rightarrow$ 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. □

Charakterisierung der entscheidbaren Sprachen

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 vom Typ 0.

Beweis

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

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 mit χ_A auch $\chi_{\bar{A}}$ berechenbar, d.h. A und $\bar{A}$ sind entscheidbar und damit 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, d.h. M entscheidet A. □

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 $Z = \{q_0, \ldots, q_m\}$ (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 eine Anweisung $q_i a_j \to q_{i'} a_{j'} D$ 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 durch w_M repräsentierte natürliche Zahl $(w_M)_2$ 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 k-DTMs mit $k > 1$ (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_h die Kodierung einer DTM, die bei jeder Eingabe (sofort) hält und betrachte die Funktion

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

- 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

- Für eine DTM M sei $dom(M) = \{x \in \Sigma^* \mid M(x)\downarrow\}$ der Definitionsbereich der von M berechneten Funktion.
- Wir wissen bereits, dass RE $= \{dom(M) \mid M$ ist eine DTM$\}$ ist.
- Sei $A = (a_{w,x})_{w,x\in\{0,1\}^*}$ die durch $a_{w,x} = \chi_H(w\#x)$ def. Binärmatrix.
- Wegen

  $$a_{w,x} \;=\; \chi_H(w\#x) \;=\; \chi_{dom(M_w)}(x)$$

  repräsentiert Zeile w von A die Sprache $dom(M_w)$, d.h. die Zeilen von A repräsentieren genau die semi-entscheidbaren Binärsprachen.
- Wegen $a_{w,w} \;=\; \chi_H(w\#w) \;=\; \chi_K(w)$ repräsentiert die Diagonale von A die Sprache K.
- Da aber die komplementierte Diagonale von A mit keiner Zeile von A übereinstimmen kann, folgt $\bar{K} \notin$ RE und somit $K \notin$ REC.

Unentscheidbarkeit des speziellen Halteproblems

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

- Angenommen, die Sprache

  $$\bar{K} = \{w \mid M_w(w)\uparrow\}$$

  wäre semi-entscheidbar.
- Dann existiert eine DTM $M_{w'}$ mit $dom(M_{w'}) = \bar{K}$, d.h. es gilt

  $$M_{w'}(w)\downarrow \Leftrightarrow w \in \bar{K} \qquad (*)$$

- Folglich gilt

  $$w' \in \bar{K} \;\Leftrightarrow\; M_{w'}(w')\uparrow \underset{(*)}{\Leftrightarrow} w' \notin \bar{K} \quad \lightning \text{ (Widerspruch!)}$$

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**	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
w'	**1**	**0**	**1**	$\cdots$

□

Unentscheidbarkeit des speziellen Halteproblems

Korollar

REC $\subsetneq$ RE.

Beweis

Klar, da $K \in$ RE – 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:

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

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

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

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

Das Halteproblem H ist RE-vollständig. Es gilt nämlich

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

mittels der Reduktionsfunktion $x \mapsto w\#bin(x)$, wobei M_w eine DTM mit $dom(M_w) = \{bin(x) \mid x \in A\}$ ist. ◁

Bemerkung

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

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

Beweis

- Gelte $A \leq B$ mittels f und sei $B \in$ REC.
- Wegen $B \in$ REC 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 $B \in \mathrm{REC}$.
- 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. $A \in \mathrm{REC}$. □

Bemerkung

Der Abschluss von RE unter $\leq$ folgt analog (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 M eine DTM mit $dom(M) = A$.
- Um A auf H_0 zu reduzieren, transformieren wir x in die Kodierung w_x einer DTM M_{w_x}, die zunächst ihre Eingabe durch x ersetzt und dann $M(x)$ 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 partielle 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 partiellen Funktionen definieren wir die Sprache

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

- 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

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

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.

- Dagegen ist die Eigenschaft $\mathcal{F}' = \{\chi_H\}$ trivial, da $L_{\mathcal{F}'} = \varnothing$ ist.

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 $\overline{L}_{\mathcal{F}}$) zu reduzieren, indem wir für eine gegebene DTM M_w eine DTM $M_{w'}$ konstruieren mit

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

- 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 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 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

  $$w \in H_0 \quad \Rightarrow \quad M_{w'} \text{ berechnet } f$$
  $$w \notin H_0 \quad \Rightarrow \quad M_{w'} \text{ berechnet } u.$$
- 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 $L_{\mathcal{S}} = \varnothing$ oder $L_{\mathcal{S}} = \{0,1\}^*$ ist.

Beweis

Siehe Übungen.

Weitere Entscheidungsprobleme für Sprachklassen

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

Das Leerheitsproblem

Gegeben: Eine Sprache L aus $\mathcal{C}$.
Gefragt: Ist $L = \varnothing$?

Das Äquivalenzproblem

Gegeben: Zwei Sprachen L_1 und L_2 aus $\mathcal{C}$.
Gefragt: Gilt $L_1 = L_2$?

Das Schnittproblem

Gegeben: Zwei Sprachen L_1 und L_2 aus $\mathcal{C}$.
Gefragt: Ist $L_1 \cap L_2 = \varnothing$?

Hierbei repräsentieren wir Sprachen in $\mathcal{C} = \mathrm{REG}, \mathrm{CFL}, \mathrm{CSL}, \mathrm{RE}$ durch entsprechende Grammatiken und Sprachen in $\mathcal{C} = \mathrm{DCFL}, \mathrm{DCSL}$ durch entsprechende Akzeptoren.

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 in 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. $\triangleleft$

Das Postsche Korrespondenzproblem

Lemma

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

Beweis

- Sei $\Sigma = \{a_1, \ldots, a_m\}$ und sei $k = \max(1, \lceil \log_2(m) \rceil)$. Dann können wir a_i durch eine k-stellige Binärzahl $bin_k(a_i)$ mit dem Wert $i-1$ und ein Wort $w = w_1 \ldots w_n$ durch $bin(w) = bin_k(w_1) \ldots bin_k(w_n)$ kodieren.
- Nun folgt $\mathrm{PCP}_\Sigma \leq \mathrm{PCP}_{\{0,1\}}$ mittels der Reduktionsfunktion

$$f : \begin{pmatrix} x_1 \ldots x_k \\ y_1 \ldots y_k \end{pmatrix} \mapsto \begin{pmatrix} bin(x_1) \ldots bin(x_k) \\ bin(y_1) \ldots bin(y_k) \end{pmatrix}.$$

□

Beispiel

Sei $\Sigma = \{a, b, c\}$. Dann ist $k = \max(1, \lceil \log_2(3) \rceil) = 2$ und $bin_2(a) = 00$, $bin_2(b) = 01$ und $bin_2(c) = 10$. Somit ist

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

Das Postsche Korrespondenzproblem

Im Folgenden lassen wir im Fall $\Sigma = \{0, 1\}$ 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 $\Sigma = \{0, 1, \langle, |, \rangle\}$.
- Für ein Wort $w = w_1 \dots w_n$ sei

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

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

- Für ein Wort $w = w_1 \dots w_n$ sei

$\overleftarrow{w}$	$\overline{w}$	$\underleftarrow{w}$	$\underline{w}$
$\langle \lvert w_1 \rvert \dots \lvert w_n \rvert$	$w_1 \rvert \dots \rvert w_n \rvert$	$\langle \lvert w_1 \rvert \dots \rvert w_n$	$\lvert w_1 \rvert \dots \rvert w_n$

- Wir reduzieren MPCP mittels folgender Funktion f auf PCP_Σ:

$$f : \begin{pmatrix} x_1 & \dots & x_k \\ y_1 & \dots & y_k \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overline{x_1} & \dots & \overline{x_k} & \rangle \\ \underleftarrow{y_1} & \underline{y_1} & \dots & \underline{y_k} & \lvert\rangle \end{pmatrix}$$

- Dabei nehmen wir an, dass $(x_i, y_i) \neq (\varepsilon, \varepsilon)$ ist, da wir diese Paare im Fall $i > 1$ einfach weglassen und im Fall $i = 1$ $f(I) = \binom{0}{0}$ setzen können.

Beispiel

$$f : \begin{pmatrix} 00 & 1 & 101 & 11 \\ 001 & 11 & 0 & 1 \end{pmatrix} \mapsto \begin{pmatrix} \langle\lvert 0\rvert 0\rvert & 0\rvert 0\rvert & 1\rvert & 1\rvert 0\rvert 1\rvert & 1\rvert 1\rvert & \rangle \\ \langle\lvert 0\rvert 0\rvert 1 & \lvert 0\rvert 0\rvert 1 & \lvert 1\rvert 1 & \lvert 0 & \lvert 1 & \lvert\rangle \end{pmatrix}$$

◁

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

- Wir reduzieren MPCP mittels folgender Funktion f auf PCP_Σ:

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

- Dabei nehmen wir an, dass $(x_i, y_i) \neq (\varepsilon, \varepsilon)$ ist, da wir diese Paare im Fall $i > 1$ einfach weglassen und im Fall $i = 1$ $f(I) = \binom{0}{0}$ setzen können.
- 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

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

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

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

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 $i_1 = 1$ und $i_n = k+2$ sein, da das Lösungswort mit $\langle$ beginnen und mit $\rangle$ enden muss (und $f(I)$ nicht das Paar $(\varepsilon, \varepsilon)$ enthält).
- 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. □

Unentscheidbarkeit des PCP

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 $A \leq \mathrm{MPCP}_\Gamma$ 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 $S = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_m = w$ von w kodiert.

Unentscheidbarkeit des PCP

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

- Wir bilden $f(w)$ aus folgenden Wortpaaren:
  - $(\langle, \langle \mid S)$, „Startpaar“
  - für jede Regel $l \to r$ in P: (l, r), „Ableitungspaare“
  - für alle $a \in V \cup \Sigma \cup \{\mid\}$: (a, a), „Kopierpaare“
  - sowie das Paar $(w \mid \rangle, \rangle)$ „Abschlusspaar“

Unentscheidbarkeit des PCP

Beispiel

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

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

- Der Ableitung $\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$ entspricht dann das MPCP-Lösungswort

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

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

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

- Dieses entspricht der „parallelisierten“ Ableitung

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

◁

Unentscheidbarkeit des PCP

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

- Wir bilden $f(w)$ aus folgenden Wortpaaren:
  - $(\langle, \langle \mid S)$, „Startpaar“
  - für jede Regel $l \to r$ in P: (l, r), „Ableitungspaare“
  - für alle $a \in V \cup \Sigma \cup \{\mid\}$: (a, a), „Kopierpaare“
  - sowie das Paar $(w \mid \rangle, \rangle)$ „Abschlusspaar“
- 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. □

Unentscheidbarkeit des Schnittproblems für CFL

Das Schnittproblem für kontextfreie Grammatiken ($\mathrm{SP}_{\mathrm{Typ2}}$)

Gegeben: Zwei kontextfreie Grammatiken G_1 und G_2.

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

Satz

Das Schnittproblem für kontextfreie Grammatiken ist RE-vollständig.

Unentscheidbarkeit des Schnittproblems für CFL

Satz

Das Schnittproblem für kontextfreie Grammatiken ist RE-vollständig.

Beweis

- Das Problem $\mathrm{SP}_{\mathrm{Typ2}}$ ist semi-entscheidbar, da eine DTM systematisch nach einem Wort $x \in L(G_1) \cap L(G_2)$ suchen kann.
- Um PCP auf $\mathrm{SP}_{\mathrm{Typ2}}$ zu reduzieren, betrachten wir für eine Folge $s = (x_1, \ldots, x_k)$ von Strings $x_i \in \{0,1\}^*$ die Sprache

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

  über dem Alphabet $\Sigma = \{1, \ldots, k, \#, 0, 1\}$.
- Die Sprache L_s wird von der Grammatik $G_s = (\{A\}, \Sigma, P_s, A)$ mit der Regelmenge

  $$P_s\colon A \to 1Ax_1, \ldots, kAx_k, 1\#x_1, \ldots, k\#x_k$$

  erzeugt.

Unentscheidbarkeit des Schnittproblems für CFL

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. □

Unentscheidbarkeit des Schnittproblems 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_s, G_t) mit folgenden Regeln reduziert:

$$P_s\colon A \to 1A0,\ 2A001,\ 3A01100,\ 1\#0,\ 2\#001,\ 3\#01100$$

$$P_t\colon A \to 1A00110,\ 2A01011,\ 3A00,\ 1\#00110,\ 2\#01011,\ 3\#00$$

- Der PCP-Lösung $\alpha = (1, 3, 2, 3)$ 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 $L(G_s) \cap L(G_t)$. ◁

Unentscheidbarkeit des Schnittproblems für DCFL

Das Schnittproblem für deterministische Kellerautomaten ($\mathrm{SP}_{\mathrm{DPDA}}$)

Gegeben: Zwei DPDAs M_1 und M_2.

Gefragt: Gilt $L(M_1) \cap L(M_2) = \varnothing$?

Korollar

$\mathrm{SP}_{\mathrm{DPDA}}$ ist unentscheidbar.

Beweisidee

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

Das Äquivalenzproblem für kontextfreie Sprachen

Das Äquivalenzproblem für kontextfreie Grammatiken ($\ddot{\mathrm{A}}\mathrm{P}_{\mathrm{Typ2}}$)

Gegeben: Zwei kontextfreie Grammatiken G_1 und G_2.

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

Korollar

$\ddot{\mathrm{A}}\mathrm{P}_{\mathrm{Typ2}}$ ist unentscheidbar.

Beweisidee

- Wir reduzieren das Komplement von PCP auf $\ddot{\mathrm{A}}\mathrm{P}_{\mathrm{Typ2}}$.
- 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 $f : I \mapsto \langle G_1, G_2 \rangle$ die gewünschte Reduktion, wobei G_1 und G_2 kontextfreie Grammatiken sind mit

$$L(G_1) = \overline{L}_s \cup \overline{L}_t \text{ und } L(G_2) = \Sigma^*$$

□

Das Leerheitsproblem für DLBAs

Das Leerheitsproblem für DLBAs ($\mathrm{LP}_{\mathrm{DLBA}}$)

Gegeben: Ein DLBA M.

Gefragt: Ist $L(M) = \varnothing$?

Satz

Das Leerheitsproblem für DLBAs ist unentscheidbar.

Beweisidee

- Wir reduzieren $\overline{\mathrm{PCP}}$ auf $\mathrm{LP}_{\mathrm{DLBA}}$.
- Hierzu überführen wir eine PCP-Instanz $I = \binom{s}{t}$ in einen DLBA M mit

  $$L(M) = L_s \cap L_t.$$

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

  $$I \in \mathrm{PCP} \Leftrightarrow L_s \cap L_t \neq \varnothing \Leftrightarrow L(M) \neq \varnothing \Leftrightarrow M \notin \mathrm{LP}_{\mathrm{DLBA}}$$ □

Entscheidbare Probleme

Dagegen ist es nicht schwer,

- für eine kontextfreie Grammatik G zu entscheiden, ob mindestens ein Wort in G ableitbar ist (Leerheitsproblem $\mathrm{LP}_{\mathsf{Typ2}}$ für kontextfreie Grammatiken), und
- für eine kontextsensitive Grammatik G und ein Wort x zu entscheiden, ob x in G ableitbar ist (Wortproblem $\mathrm{WP}_{\mathsf{Typ1}}$ für kontextsensitive Grammatiken).

Satz

- Das Leerheitsproblem für kontextfreie Grammatiken ist entscheidbar.
- Das Wortproblem für kontextsensitive Grammatiken ist entscheidbar.

Beweis.

Siehe Übungen. □

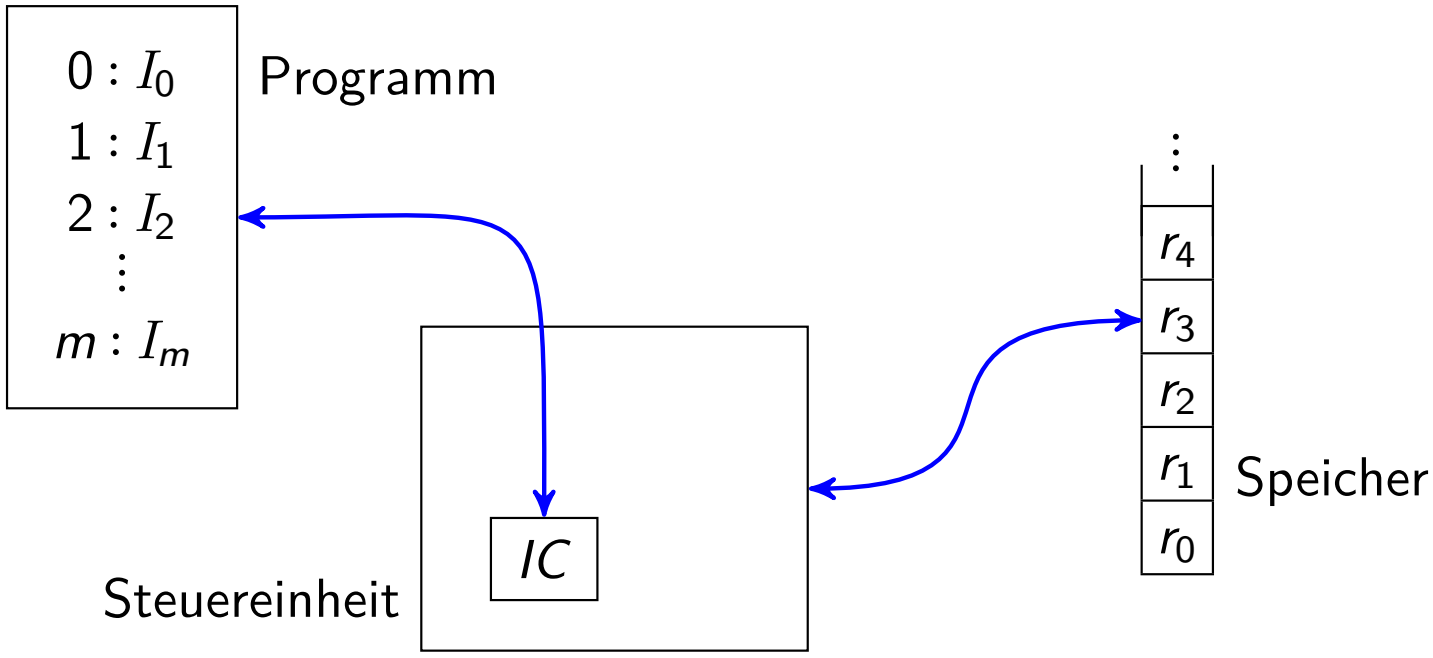
Überblick der (Un-)Entscheidbarkeitsresultate

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

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

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

Die Registermaschine (random access machine, RAM)



- führt ein Programm $P = (I_0, \ldots, I_m)$ aus, das aus einer endlichen Folge von Befehlen (instructions) I_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, $i \geq 0$, die beliebig große natürliche Zahlen aufnehmen können.

Eine Programmiersprache für RAMs

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

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

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

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
- hält P genau dann, wenn $(n_1, \ldots, n_k) \in dom(f)$ ist, und
- wenn 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** $r_1 = 0$ **THEN GOTO** 4
1 $r_1 := r_1 \dot{-} 1$
2 ~~$r_0 := r_0 + r_2$~~ **GOTO** 5
3 **GOTO** 0
4 **HALT**

5 $r_3 := r_2$
6 **IF** $r_3 = 0$ **THEN GOTO** 3
7 $r_3 := r_3 \dot{-} 1$
8 $r_0 := r_0 + 1$
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 $x_i := x_j \dot{-} c$ 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 so oft ausgeführt, wie der Wert von x_i zu Beginn der Schleife angibt.

WHILE- und LOOP-Berechenbarkeit

- Eine partielle Funktion $f : \mathbb{N}^k \to \mathbb{N} \cup \{\uparrow\}$ 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
  - hält P genau dann, wenn $(n_1, \ldots, n_k) \in dom(f)$ ist, und
  - wenn P hält, hat x_0 den Wert $f(n_1, \ldots, n_k)$.
- 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**
~~$x_0 := x_0 + x_2$;~~
$x_1 := x_1 \dot{-} 1$
END

$x_3 := x_2$;
WHILE $x_3 \neq 0$ **DO**
$x_0 := x_0 + 1$; $x_3 := x_3 \dot{-} 1$
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** ◁

Numerische Repräsentation von Wörtern

- 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 $num_\Sigma(x)$ kodieren:

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

- 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	$\ldots$

Transformation zw. Wort- und numerischen Funktionen

- Ist $\Sigma = \{0,1\}$, so lassen wir den Index weg und schreiben einfach *num* und *str* anstelle von num_Σ und str_Σ:

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

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

- 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:

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

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

Beispiel

Die Fkt. $f : (n_1, n_2) \mapsto n_1 n_2$ wird durch folgende Wortfkt. repräsentiert:

$$\hat{f}(w) = \begin{cases} str(n_1 n_2), & w = str_2(n_1, n_2), \\ \uparrow, & \text{sonst.} \end{cases}$$

w	ε	0	1	#	00	01	0#	10	11	1#	#0	#1	##	000
(n_1, n_2)	-	-	-	$(0,0)$	-	-	$(1,0)$	-	-	$(2,0)$	$(0,1)$	$(0,2)$	-	-
$n_1 n_2$	-	-	-	0	-	-	0	-	-	0	0	0	-	-
$\hat{f}(w)$	$\uparrow$	$\uparrow$	$\uparrow$	ε	$\uparrow$	$\uparrow$	ε	$\uparrow$	$\uparrow$	ε	ε	ε	$\uparrow$	$\uparrow$

w	001	00#	010	011	01#	0#0	0#1	0##	100	101	10#	...
(n_1, n_2)	-	$(3,0)$	-	-	$(4,0)$	$(1,1)$	$(1,2)$	-	-	-	$(5,0)$	...
$n_1 n_2$	-	0	-	-	0	1	2	-	-	-	0	...
$\hat{f}(w)$	$\uparrow$	ε	$\uparrow$	$\uparrow$	ε	0	1	$\uparrow$	$\uparrow$	$\uparrow$	ε	...

Äquivalenz von Turing- und GOTO-Berechenbarkeit

Satz

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

Beweis

- Sei P ein GOTO-Programm, das eine partielle Fkt. $f : \mathbb{N}^k \to \mathbb{N} \cup \{\uparrow\}$ 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 $r_0, \ldots, r_m$ beschreiben.
- Wir konstruieren eine $(m+2)$-DTM M, die
  - den Inhalt von IC in ihrem Zustand,
  - die Registerwerte $r_1, \ldots, r_m$ auf den Bändern $1, \ldots, m$ 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.

Simulation eines GOTO-Programms durch eine DTM

- Die Aufgabe von M ist es, bei Eingabe $w \in \{0,1,\#\}^*$ das Wort $str(f(n_1,\ldots,n_k))$ auszugeben, wenn $w = str_k(n_1,\ldots,n_k)$ für ein Tupel $(n_1,\ldots,n_k) \in dom(f)$ 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 $i = 2,\ldots,k$ auf das i-te Band und löscht auf dem 1. Band alle Eingabezeichen bis auf $str(n_1)$.
- Für $i = 1,\ldots,m$ 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 I 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 $str(r_0) = str(f(n_1,\ldots,n_k)) = \hat{f}(w)$ aus.

Simulation einer DTM durch ein GOTO-Programm

- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ eine DTM,die die String-Repräsentation $\hat{f}$ einer partiellen Funktion $f : \mathbb{N}^k \to \mathbb{N} \cup \{\uparrow\}$ berechnet.
- M gibt also bei Eingabe w das Wort $str(f(n_1, \ldots, n_k))$ aus, falls w die Form $w = str_k(n_1, \ldots, n_k)$ hat und $f(n_1, \ldots, n_k)$ 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.
- 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.
- Eine Konfiguration $K = uq_iv$ von M mit $u = a_{i_1} \ldots a_{i_s}$ und $v = a_{j_1} \ldots a_{j_t}$ wird wie folgt in den Registern r_0, r_1, r_2 gespeichert:
  - $r_0 = (i_1 \ldots i_s)_m$
  - $r_1 = i$
  - $r_2 = (j_t \ldots j_1)_m$

Simulation einer DTM durch ein GOTO-Programm

- Eine Konfiguration $K = uq_iv$ 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:
  - $r_0 = (i_1 \dots i_s)_m$
  - $r_1 = i$
  - $r_2 = (j_t \dots j_1)_m$
- 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.

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:
  - Im Fall $\delta(q_i, a_j) = \varnothing$ 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:

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

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

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 eines WHILE- durch ein GOTO-Programm

- Sei P ein WHILE-Programm, das $f : \mathbb{N}^k \to \mathbb{N} \cup \{\uparrow\}$ 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 $\vdots$

- 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 eines GOTO- durch ein WHILE-Programm

- Sei $P = (I_0, \ldots, I_m)$ ein GOTO-Programm, das $f : \mathbb{N}^k \to \mathbb{N} \cup \{\uparrow\}$ 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. □

Vergleich von LOOP- und WHILE-Berechenbarkeit

- 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 $a : \mathbb{N}^2 \to \mathbb{N}$, die wie folgt definiert ist

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