Matrixgrammatiken – Definition I

Definition:

i) Eine Matrixgrammatik ist ein Quintupel $G = (N, T, M, S, F)$, wobei

— N, T und S wie bei einer kontextfreien Grammatik spezifiziert sind,

— $M = \{m_1, m_2, \ldots m_n\}$ eine endliche Menge von endlichen Folgen von kontextfreien Regeln ist (d. h. für $1 \leq i \leq n$ gilt

$$m_i = (A_{i,1} \to w_{i,1}, A_{i,2} \to w_{i,2}, \ldots, A_{i,r_i} \to w_{i,r_i})$$

mit gewissen $r_i \geq 1$, $A_{i,j} \in N$, $w_{i,j} \in (N \cup T)^*$, $1 \leq j \leq r_i$) und

— F eine Teilmenge der Regeln ist, die in den m_i, $1 \leq i \leq n$, vorkommen.

Matrixgrammatiken – Definition II

ii) Es sei $m = (A_1 \to w_1, A_2 \to w_2, \ldots, A_r \to w_r) \in M$. Wir sagen, dass durch m aus $x \in V^+$ das Wort $y \in V^*$ abgeleitet wird (und schreiben dafür $x \Longrightarrow_m y$), wenn es Wörter $x_1, x_2, \ldots, x_{r+1}$ so gibt, dass

— $x = x_1$ und $y = x_{r+1}$ gelten und

— für $0 \leq i \leq r-1$ entweder $x_i = x_i' A_i x_i''$ und $x_{i+1} = x_i' w_i x_i''$ gelten oder A_i in x_i nicht vorkommt und $x_{i+1} = x_i$ und $A_i \to w_i \in F$ gelten.

iii) Die von G erzeugte Sprache $L(G)$ besteht aus allen Wörtern $z \in T^*$, für die es eine Ableitung

$$S \Longrightarrow_{m_{i_1}} w_1 \Longrightarrow_{m_{i_2}} w_2 \Longrightarrow_{m_{i_3}} \ldots \Longrightarrow_{m_{i_t}} = w_t = z$$

mit $t \geq 1$ und $m_{i_j} \in M$ für $1 \leq j \leq t$ gibt.

Matrixgrammatiken – Normalform

Definition: Eine Matrixgrammatik $G = (N, T, M, S, F)$ ist in Normalform, wenn die folgenden Bedingungen erfüllt sind:

- $N = N_1 \cup N_2 \cup \{S, Z, \#\}$, $S, Z \notin N_1 \cup N_2$, $N_1 \cap N_2 = \emptyset$
- jede Matrix von M hat eine der folgenden Formen
  — $(S \to XA)$ mit $X \in N_1$, $A \in N_2$,
  — $(X \to Y, A \to w)$ mit $X, Y \in N_1$, $A \in N_2$, $w \in (N_2 \cup T)^*$,
  — $(X \to Y, A \to \#)$ mit $X \in N_1$, $Y \in N_1 \cup \{Z\}$, $A \in N_2$,
  — $(Z \to \lambda)$,
- es gibt nur eine Matrix der Form $(S \to XA)$ in M,
- F besteht aus allen Regeln der Form $A \to \#$ mit $A \in N_2$.

Matrixgrammatiken – Resultate

$\mathcal{L}(MAT)$ bezeichne die Menge aller Sprachen, die von Matrixgrammatiken erzeugt werden können.

Satz: $\mathcal{L}(MAT) = \mathcal{L}(RE)$.

Satz: Für jede rekursiv-aufzählbare Sprache L gibt es eine Matrixgrammatik G in Normalform so, dass $L(G) = L$ gilt.

Grammatiksysteme – Definition

Definition: i) Ein Grammatiksystem mit n Komponenten ist ein $(n+3)$-Tupel $G = (N, T, P_1, P_2, \ldots, P_n, S)$, wobei
— N, T und S wie bei einer kontextfreien Grammatik spezifiziert sind, und
— $P_1, P_2, \ldots, P_n$ endliche Mengen von kontextfreien Regeln sind.

ii) Wir sagen, dass durch P_i, $1 \leq i \leq n$, aus x das Wort y abgeleitet wird (geschrieben als $x \Longrightarrow^t_{P_i} y$), wenn $x \Longrightarrow^*_{P_i} y$ gilt (d. h. y kann aus x durch eine Ableitung gewonnen werden, in der nur Regeln aus P_i benutzt werden) und keine Regel aus P_i auf y anwendbar ist.

iii) Die vom Grammatiksystem G erzeugte Sprache $L(G)$ besteht aus allen Wörtern $z \in T^*$, die durch eine Ableitung der Form

$$S \Longrightarrow^t_{P_{i_1}} w_1 \Longrightarrow^t_{P_{i_2}} w_2 \Longrightarrow^t_{P_{i_3}} \ldots \Longrightarrow^t_{P_{i_s}} w_s = z$$

mit gewissen $t \geq 1$, $1 \leq i_j \leq n$, $1 \leq j \leq s$, erzeugt werden können.

Grammatiksysteme – Resultate

$\mathcal{L}_n(CF)$ bezeichne die Menge aller Sprachen, die von Grammatiksystemen mit n Komponenten erzeugt werden können.

Satz:

i) $\mathcal{L}(CF) = \mathcal{L}_1(CF) = \mathcal{L}_2(CF)$.

ii) For any $n \geq 3$, $\mathcal{L}_n(CF) = \mathcal{L}_3(CF)$.

Registermaschine – Definition I

i) Eine Registermaschine besteht aus den Registern B, C_0, $, C_1, C_2, \ldots, C_n, \ldots$ und einem Programm.
B heißt Befehlszähler, C_0 heißt Arbeitsregister oder Akkumulator, und jedes der Register C_n, $n \geq 1$, heißt Speicherregister.

Jedes Register enthält als Wert eine Zahl aus $\mathbf{N}_0$.

ii) Unter einer Konfiguration der Registermaschine verstehen wir das unendliche Tupel $(b, c_0, c_1, \ldots, c_n, \ldots)$, wobei
das Register B die Zahl b enthält, und
für $n \geq 0$ das Register C_n die Zahl c_n enthält.

iii) Das Programm ist eine endliche Folge von Befehlen.

Registermaschine – Definition II

Liste der zugelassenen Befehle und der von ihnen bewirkte Änderung der Konfiguration $(b, c_o, c_1, \ldots c_n, \ldots)$ in die Konfiguration $(b', c'_0, c'_1, \ldots, c'_n, \ldots)$ (wobei $u' = u$ für die nicht angegebenen Komponenten gilt)

Ein- und Ausgabebefehle:

`LOAD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_0 = c_i$
`ILOAD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_0 = c_{c_i}$
`CLOAD` i ,	$i \in \mathbf{N}_0$	$b' = b + 1$	$c'_0 = i$
`STORE` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_i = c_0$
`ISTORE` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c'_{c_i} = c_0$

Sprungbefehle:

`GOTO` i ,	$i \in \mathbf{N}$	$b' = i$
`IF` $c_0 = 0$ `GOTO` i ,	$i \in \mathbf{N}$	$b' = \begin{cases} i & \text{falls } c_0 = 0 \\ b + 1 & \text{sonst} \end{cases}$

Registermaschine – Definition III

Arithmetische Befehle:

`ADD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 + c_i$
`CADD` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 + i$
`SUB` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \begin{cases} c_0 - c_i & \text{für } c_0 \geq c_i \\ 0 & \text{sonst} \end{cases}$
`CSUB` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \begin{cases} c_0 - i & \text{für } c_0 \geq i \\ 0 & \text{sonst} \end{cases}$
`MULT` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 * c_i$
`CMULT` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = c_0 * i$
`DIV` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \lfloor c_0 / c_i \rfloor$
`CDIV` i ,	$i \in \mathbf{N}$	$b' = b + 1$	$c_0' = \lfloor c_0 / i \rfloor$

Stopbefehl:

`END`

Registermaschine – induzierte Funktion

Definition:
Sei M eine Registermaschine. Die von M induzierte Funktion $f_M : \mathbf{N}^n \to \mathbf{N}$ ist wie folgt definiert:
$f(x_1, x_2, \ldots x_n) = y$ gilt genau dann, wenn M ausgehend von der Konfiguration $(1, 0, x_1, x_2, \ldots, x_n, 0, 0, \ldots)$ die Konfiguration $(b, c_0, y, c_2, c_3, \ldots)$ für gewisse $b, c_0, c_2, c_3, \ldots$ erreicht und der b-te Befehl des Programm `END` ist.

Registermaschine – Resultate

Satz: Die Menge der von Registermaschinen induzierten Funktion ist die Menge der partiell-rekursiven Funktionen.

Satz: Für jede partiell-rekursive Funktion f gibt es eine Registermaschine,

- die f induziert,
- die nur die Befehle
  — $a+$ mit $c'_a = c_a + 1$ und $b' = b + 1$,
  — $a–(k)$ mit $c'_a = c_a - 1$ und $b' = b + 1$ für $c_a > 0$,
  $c'_a = c_a$ und $b' = k$ für $c_a = O$,
  — `END`
  enthält,
- `END` nur als letzten Befehl des Programms benutzt und
- das Resultat in einem Register a enthält, auf das kein Befehl $a–(k)$, $k \geq 1$, angewendet wurde.

Multimengen

Multimenge M over U – Abbildung von U in $\mathbf{N}_0 \cup \{\infty\}$
$M(x)$ – Multiplizität von x

M endlich, wenn $M(x) \in \mathbf{N}_0$ für alle $x \in U$ gilt.

Die Mächtigkeit von endlichem M ist $\#(M) = \sum_{x \in U} M(x)$

Identifikation von endlichem M mit Wort $w_M \in U^*$ mit $\#_x(w_m) = M(x)$,
w_M nur eindeutig bis auf Permutation der Buchstaben

Zu jedem $w \in U^*$ gibt es genau eine endliche Multimenge M mit $w = w_M$

$\#(M) = |w_M|$

Längenmengen von Sprachen

Für eine Sprache L und eine Familie X von Grammatiken definieren wir

$$\begin{aligned} N(L) &= \{n \mid n = |w| \text{ for some } w \in L\}, \\ N(X) &= \{N(L) \mid L \in \mathcal{L}(X)\}. \end{aligned}$$

Satz : i) $N(REG) = N(CF) \subset N(CS) \subset N(RE)$.
ii) Eine Menge $M \subseteq \mathbf{N}_0$ gehrt genau dann zu $N(CF)$, wenn es Zahlen $r, s, q_1, q_2, \ldots q_r, p_1, p_2, \ldots, p_s \in \mathbf{N}_0$ so gibt, dass

$$p \geq 1, \quad q_1 < q_2 < \ldots < q_r < p_1 < p_2 < \ldots < p_s,$$

$$M = \{q_1, q_2, \ldots, q_r\} \cup \bigcup_{i=1}^{s} \{p_i + np \mid n \in \mathbf{N}_0\}$$

gelten.

Membransystem – Definition I

Definition:

i) Ein Membransystem mit m Membranen ist ein $(2m+3)$- Tupel

$$\Gamma = (V, \mu, w_1, w_2, \ldots w_m, R_1, R_2, \ldots R_m, i),$$

wobei

— V ein Alphabet (der in den Membranen vorkommenden Objekte) ist,

— μ ein Membranstruktur (von m Membranen) ist,

— w_j für $1 \leq j \leq m$ ein Wort/eine (endliche) Multimenge über V ist (initialer Inhalt der Membran j),

— R_j für $1 \leq j \leq m$ eine endliche Menge von Regeln ist (anwendbar auf Wörter in der Membran j),

— i eine natürliche Zahl mit $1 \leq i \leq m$ und die Membran i einfach ist (Ausgabemembran).

Membransystem – Definition II

ii) Eine Konfiguration von Γ ist ein m-Tupel von Wörtern/Multimengen.

Es seien $C = (u_1, u_2, \ldots, u_m)$ und $C' = (u'_1, u'_2, \ldots, u'_m)$ zwei Konfigurationen. Wir sagen, dass C genau dann in C' transformiert wird (geschrieben als $C \vdash C'$), wenn C' aus C durch maximal parallele Anwendung von Regeln aus R_i auf u_i, $1 \leq i \leq m$, entsteht.

iii) Eine Konfiguration $C = (u_1, u_2, \ldots, u_m)$ heißt genau dann Stoppkonfiguration, wenn für $1 \leq i \leq m$ keine Regel aus R_i mehr anwendbar ist.

iv) Die vom Membransystem Γ erzeugte Sprache $L(\Gamma)$ ist die Menge aller Zahlen n, für die eine Stoppkonfiguration $C = (u_1, u_2, \ldots, u_m)$ mit $|u_i| = n$ existiert.

Spezielle Membransysteme

Ein Buchstabe c aus V heißt Katalyst, wenn alle Regeln, in denen c vorkommt von der Form $ca \to cw$ mit $a \in V$ und $w \in (V \times Tar)^*$.

Eine Regel $u \to w$ mit $u \in V^+$ und $w \in (V \times Tar)^*$ heißt
— *nichtkooperierend*, wenn $u \in V$ gilt,
— *kooperierend*, wenn $|u| \geq 2$ gilt,
— *katalytisch*, wenn $u = ca$ und $w = cw'$ für einen Katalysator c und gewisse $a \in V$, $w' \in V^*$ gelten.

Ein Membransystem heißt
— *nichtkooperierend*, wennn alle seine Regeln nichtkooperierend sind,
— *katalytisch*, wenn alle seine Regeln nichtkooperierend oder katalytisch sind,
— *kooperierend*, wenn es eine kooperierende und nichtkatalytische Regel enthält.

Membransysteme – Resultate I

$\mathcal{L}_n(P, nco)$, $\mathcal{L}_n(P, cat)$ und $\mathcal{L}_n(P, coo)$ bezeichnen die Mengen von Sprachen, die von nichtkooperierenden, katalytischen und kooperierenden Membransystemen mit höchstens n Membranen erzeugt werden knnen.

Für $X \in \{nco, cat, coo\}$ setzen wir: $\mathcal{L}_*(P, X) = \bigcup_{n \geq 1} \mathcal{L}_n(P, X)$

Fakt: Für $X \in \{nco, cat, coo\}$ gilt
$\mathcal{L}_1(P, X) \subseteq \mathcal{L}_2(P, X) \subseteq \mathcal{L}_3(P, X) \subseteq \ldots \subseteq \mathcal{L}_n(P, X) \subseteq \ldots \subseteq \mathcal{L}_*(P, X)$.

Lemma: Für $X \in \{nco, cat, coo\}$ und $n \geq 2$ gilt $\mathcal{L}_1(P, X) \subseteq \mathcal{L}_2(P, X) = \mathcal{L}_n(P, X) = \mathcal{L}_*(P, X)$.

Satz: Für $n \geq 1$ gilt $\mathcal{L}_1(P, nco) = \mathcal{L}_n(P, nco) = \mathcal{L}_*(P, nco) = N(CF)$.

Satz: Für $n \geq 1$ gilt $\mathcal{L}_1(P, coo) = \mathcal{L}_n(P, coo) = \mathcal{L}_*(P, coo) = N(RE)$.

Membransysteme – Resultate II

Definition: Es seien $\Gamma = (V, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m, i)$ ein Membransystem mit m Membranen und $o \in V$ ein ausgezeichnetes Element. Falls die Sprache des Membransystems

$$\Gamma_n = (V, \mu, w_1 o^n, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m, i)$$

für jedes n genau ein Element a_n enthält, so sagen wir, dass Γ die Funktion $f : \mathbf{N}_0 \to \mathbf{N}_0$ mit $f(n) = a_n$ berechnet.

Satz: Für jede partiell-rekursive Funktion $f : \mathbf{N}_0 \to \mathbf{N}_0$ gibt es ein katalytisches Membransystem mit zwei Membranen, das f berechnet.

Satz: Für $n \geq 2$ gilt
$\mathcal{L}_1(P, cat) \subset \mathcal{L}_2(P, cat) = \mathcal{L}_n(P, cat) = \mathcal{L}_*(P, cat) = N(RE)$.

Membransysteme mit Symport/Antiport-Regeln I

Definition: i) Ein Membransystem mit Symport/Antiport-Regeln und m Membranen ist ein Konstrukt

$$\Gamma = (V, \mu, E, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots R_m, i),$$

wobei V, μ, $w_1, w_2, \ldots w_m$, $R_1, R_2, \ldots, R_m$ and i wie bei einem Membransystem spezifiziert sind, E eine Teilmenge von V ist und R_j für $1 \leq j \leq m$ eine endliche Menge von Regeln der Form (x, in) oder (x, out) oder $(x, out; , y, in)$ mit $x, y \in V^+$ ist.

ii) Eine Konfiguration eines Membransystems mit m Membranen und Symport/Antiport-Regeln ist ein m-Tupel $C = (u_1, u_2, \ldots, u_m)$ von Wörtern/Multimengen über V. Die Umgebung enthält die Multimenge Z mit $Z(x) = \infty$ für $x \in E$ und $Z(x) = 0$ für $x \notin E$.

Membransysteme mit Symport/Antiport-Regeln II

Es seien j, $1 \leq j \leq m$, eine Membran, j' die eindeutige Membran, die j enthält, und $C = (u_1, u_2, \ldots, u_m)$ eine Konfiguration. Die Anwendung der Regel (x, in) aus R_j auf $C = (u_1, u_2, \ldots, u_m)$ besteht im Entnehmen der Multimenge x aus $u_{j'}$ und Hinzufügen zu u_j; die Anwendung von (x, out) besteht in der Entnahme von x aus c_j und dem Einfügen von x in $u_{j'}$; die Anwendung von $(x, out; y, in)$ besteht in der gleichzeitigen Anwendung von (x, out) und (y, in) wie beschrieben. Wenn j die äußere Membrane ist, so übernimmt die Umgebung die Rolle von j'.

Die Transformation einer Konfiguration C in eine Konfiguration C' (geschrieben als $C \vdash C'$) erfolgt durch maximal parallele Anwendung der Regeln aus $\bigcup_{j=1}^{m} R_j$ auf die Komponenten von C.

Eine Konfiguration C heißt Stoppkonfiguration, wenn keine Regeln aus $\bigcup_{j=1}^{m} R_j$ auf C anwendbar ist.

Membransysteme mit Symport/Antiport-Regeln III

iii) die von einem Membransystem Γ mit Symport/Antiport-Regeln erzeugte Sprache $L(\Gamma)$ ist die Menge aller natrlichenm Zahlen n, für die es eine Stoppkonfiguration $C = (u_1, u_2, \ldots, u_m)$ mit $|u_i| = n$ gibt.

Satz: Für jede Menge $L \in N(RE)$ gibt es ein Membransystem Γ mit Symport/Antiport-Regeln derart, dass $L(\Gamma) = L$ gilt.