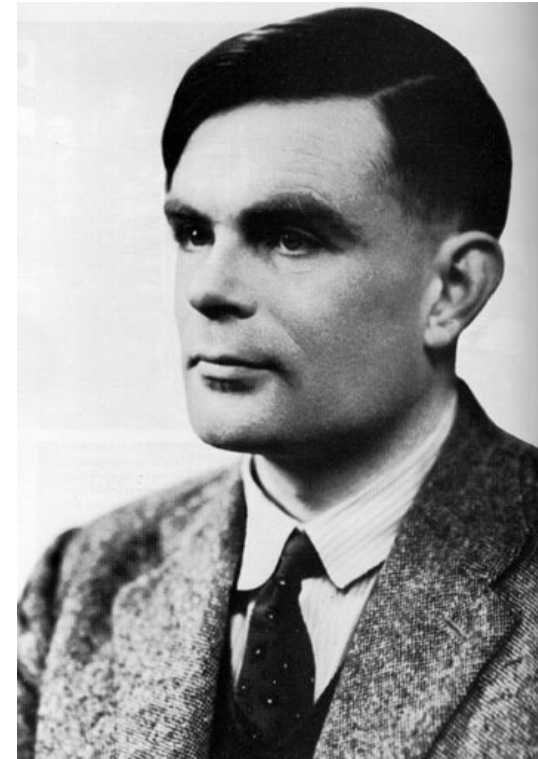


Turing

Alan M. Turing (1912 - 1954)



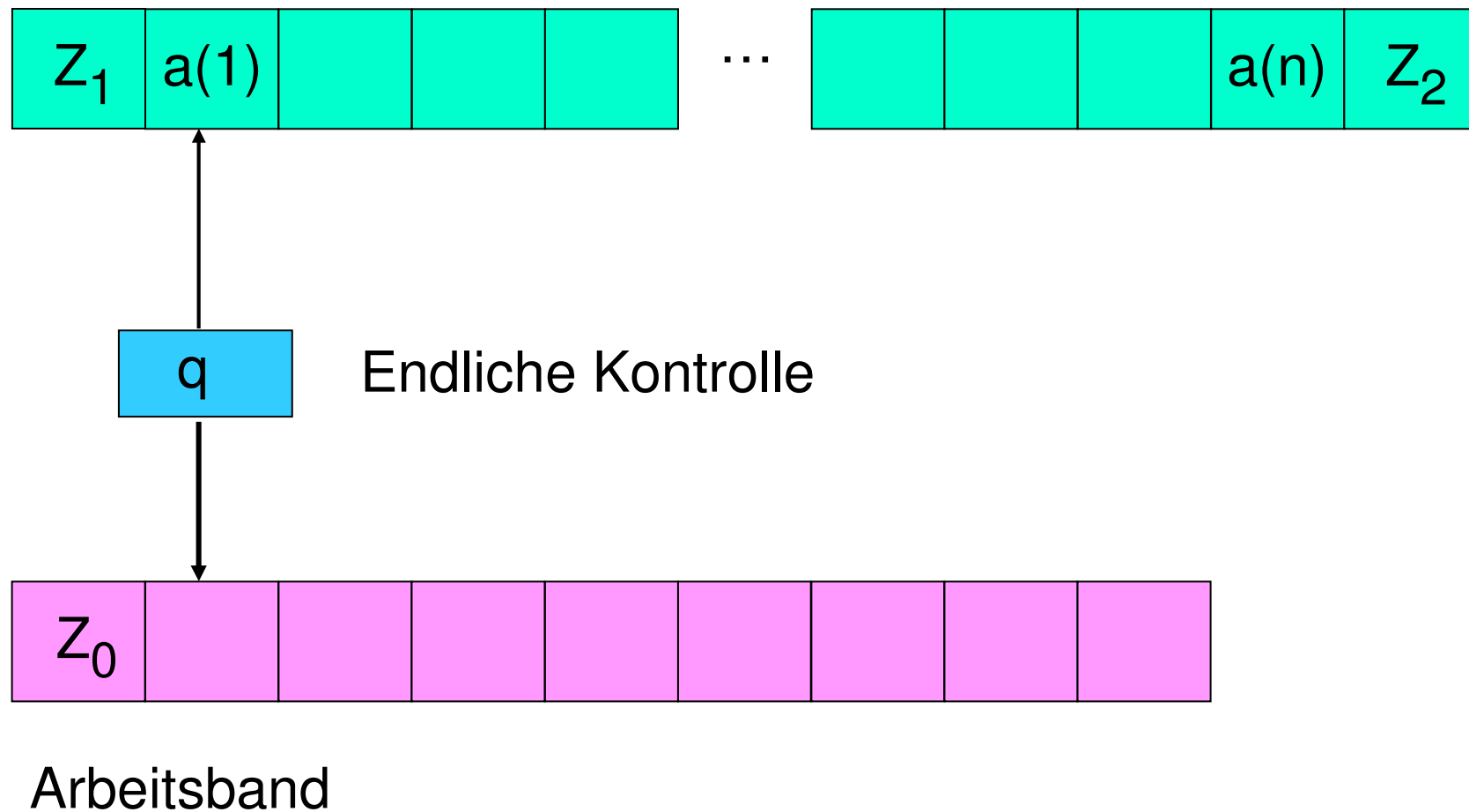
1936: On Computable Numbers, with an application to the Entscheidungsproblem.
Proc. Lond. Math. Soc. (2) 42 pp 230-265, 1936.

Turingmaschinen

Die von Alan Turing eingeführten Maschinen stellen ein Modell für universelle Berechnungen dar.

Eine **Turingmaschine** M besteht aus folgenden Komponenten: Das Eingabeband kann von links nach rechts gelesen werden. Es beinhaltet eine endliche Folge von Zeichen (das Eingabewort), wobei das Eingabewort vom Anfangssymbol Z_1 und vom Endsymbol Z_2 begrenzt ist. Das Arbeitsband kann beliebig gelesen und beschrieben werden. Die endliche Kontrolle kann einen Zustand aus einer endlichen Menge von Zuständen annehmen und steuert den Lesekopf auf dem Eingabeband und den Lese-/Schreibkopf auf dem Arbeitsband; die Kopfbewegungen sind L, R, S (links/left, rechts/right, stehenbleiben/stay).

Turingmaschinen (graphische Darstellung)



Turingmaschinen – formale Definition

$$M = (Q, T, V, \delta, q_0, Z_0, B, F)$$

Q ist eine endliche Menge von Zuständen, T das Alphabet des Eingabebandes, V das Alphabet des Arbeitsbandes, δ die Übergangsfunktion, $q_0 \in Q$ der Startzustand, $Z_0 \in V$ das linke Begrenzungssymbol auf dem Arbeitsband, $B \in V$ das Blanksymbol und $F \subseteq Q$ eine Menge von Endzuständen. Die Übergänge aus δ bestehen aus 7 Komponenten: $(q, a, X; p, Y, D(E), D(A)) \in \delta$ bedeutet, dass M im Zustand q auf dem Eingabeband das Symbol a und auf dem Arbeitsband das Symbol X liest und davon abhängig nun in den Zustand p wechselt, auf dem Arbeitsband das Symbol X mit dem Symbol Y überschreibt, auf dem Eingabeband den Lesekopf in die durch $D(E)$ beschriebene Richtung bewegt und auf dem Arbeitsband den Lese-/Schreibkopf in die durch $D(A)$ beschriebene Richtung bewegt ($D(E), D(A) \in \{L, R, S\}$).

Turingmaschinen - Konfigurationen

Eine **Konfiguration** einer (nichtdeterministischen) Turingmaschine wird durch ein Wort $Z_1 u q v Z_0 w q z$ mit $q \in Q$, $u q v \in T^* Q T^* \{Z_2\} \cup T^* \{Z_2\} Q$ und $w q z \in V^* Q \cup V^* Q V^* (V - \{B\})$ beschrieben; dabei ist das Symbol links vom ersten q das Symbol unter dem Lesekopf des Eingabebandes, das Symbol links vom zweiten q das Symbol unter dem Lese-/Schreibkopf des Arbeitsbandes.

Die **Ableitungsrelation** \Rightarrow_M auf Konfigurationen von M wird dann so definiert, dass $x \Rightarrow_M y$ genau dann gilt, wenn y eine Konfiguration beschreibt, die sich aus der Konfiguration x durch Anwendung einer durch δ definierten Transition ergibt.

Die transitive Hülle der Ableitungsrelation \Rightarrow_M wird mit \Rightarrow_M^+ , die reflexive und transitive Hülle mit \Rightarrow_M^* bezeichnet.

Der Einfachheit halber werden wir M in Zukunft weglassen.

Turingmaschinen – akzeptierte Sprache

$$M = (Q, T, V, \delta, q_0, Z_0, B, F)$$

Die von der Turingmaschine M akzeptierte Sprache $L(M)$ besteht aus genau all jenen Wörtern, bei deren Analyse M einen Endzustand erreicht, d.h., eine Konfiguration $Z_1 u q v Z_0 w q z$ mit $q \in F$.

Eine Turingmaschine M heißt **deterministisch**, wenn für alle $(q, a, X) \in Q \times T \times V$ höchstens ein Element $(q, a, X; p, Y, D(E), D(A)) \in \delta$ existiert.

Satz. Eine Sprache wird genau dann von einer deterministischen Turingmaschine akzeptiert, wenn sie von einer Typ-0-Grammatik erzeugt wird.

Turingmaschinen – Beispiel

$$M = (\{p,q,r,s,f\}, \{a,b,c\}, \{Z_0, A, B, C\}, \delta, p, Z_0, B, \{f\})$$

Die Übergangsfunktion δ enthält die folgenden Übergänge:

$$(p, a, B; p, A, R, R)$$

schreibe für eingelesenes Symbol a ein A aufs Arbeitsband

$$(p, b, B; q, B, S, L); (q, b, A; q, C, R, L)$$

überschreibe für eingelesenes Symbol b ein A auf
Arbeitsband mit C und gehe nach links

$$(q, c, Z_0; r, Z_0, S, R); (r, c, C; r, C, S, R)$$

gehe über die Symbole C auf Arbeitsband nach rechts

$$(r, c, B; s, B, S, L); (s, c, C; s, B, R, L)$$

lösche für eingelesenes Symbol c ein C auf Arbeitsband

$$(s, Z_2, Z_0; f, Z_0, S, R)$$

Übergang in Endzustand, wenn gleichzeitig mit Ende der
Eingabe auf Arbeitsband Anfang erreicht wird

$$L(M) = \{ a^n b^n c^n \mid n \geq 1 \}$$

Turingmaschinen – Varianten

- nur ein Band, d.h., Eingabe direkt auf dem Arbeitsband
- nur ein Band, allerdings nach beiden Seiten unbeschränkt
- mehrere Arbeitsbänder
- Arbeitsbänder sind in mehrere Spuren unterteilt
- mehrere Lese-/Schreibköpfe auf den Arbeitsbändern

Eine (deterministische) Turingmaschine M kann auch zur **Berechnung von Funktionen** verwendet werden, d.h., M beginnt mit dem Input auf dem Band, der Output, also der Funktionswert zum Input, ist das Ergebnis der Berechnung, wenn M hält.

Eine (nichtdeterministische) Turingmaschine M kann auch zur **Erzeugung von Wörtern** verwendet werden, d.h., M beginnt mit dem leeren Band, das Ergebnis der Berechnung ist das Wort, das am Ende einer Berechnung, wenn M hält, auf dem Band steht.

Endliche Automaten

Ein endlicher Automat ist eine Turingmaschine, die das Arbeitsband gar nicht benötigt. Die Übergänge besitzen daher die einfache Form $(q,a;p,D(E))$. Erlaubt man nur einseitige endliche Automaten, also nur $D(E) \in \{R,S\}$, dann kann man die Übergangsfunktion durch Tripel der Gestalt (q,a,p) beschreiben, wobei $a \in T \cup \{\lambda\}$; dabei ist (q,a,p) für $a \in T$ als $(q,a;p,R)$ zu interpretieren und (q,ε,p) als $(q,a;p,S)$ für ein beliebiges $a \in T$. Außerdem geht man davon aus, dass ein endlicher Automat seine Analyse auf dem ersten Symbol des Eingabewortes beginnt und in einem Endzustand akzeptiert, wenn der Lesekopf auf dem rechten Begrenzungssymbol steht, d.h., wenn das ganze Eingabewort gelesen wurde.

Kontextfreie Sprachen und Kellerautomaten

Eine Turingmaschine erfüllt die **Kellerautomatenbedingung** (Kellerautomat = **Pushdown Automaton**), wenn sie auf dem Eingabeband nie nach links gehen kann und für den Lese-/Schreibkopf auf dem Arbeitsband in jeder Konfiguration gilt, dass links davon nur Non-Blank-Symbole und rechts davon nur Blank-Symbole stehen können, d.h., jede Konfiguration muss von folgender Gestalt sein: $Z_1 u q v Z_0 w q z$ mit $q \in Q$, $u q v \in T^* Q T^* \{Z_2\} \cup T^* \{Z_2\} Q$ und $w q z \in (V - \{B\})^* (V \cup \{\epsilon, B\}) Q$.

Satz. Sei L eine formale Sprache. Dann gibt es genau dann eine Turingmaschine, welche die Kellerautomatenbedingung erfüllt und L akzeptiert, wenn L kontextfrei ist.

Linear beschränkte Automaten

Ein **linear beschränkter Automat** (linear bounded automaton) (**LBA**) ist eine Turingmaschine, die auf dem Arbeitsband nur höchstens soviel Platz verwendet wie das Eingabewort lang ist (der Platz darf sogar eine lineare Funktion der Länge des Eingabewortes sein).

Satz. Eine Sprache wird genau von einem **linear beschränkten Automaten** akzeptiert, wenn sie von einer **monotonen Grammatik** erzeugt wird.

Asymptotisches Verhalten von Funktionen

Bei der Analyse von Algorithmen beschränkt man sich oft auf die Bestimmung des Aufwands an bestimmten Operationen und berechnet nicht direkt die Laufzeit (bei Sortialgorithmen bestimmt man beispielsweise die Anzahl der notwendigen Vergleiche).

Bei Turingmaschinen ist einerseits die Anzahl der Schritte (Zeit) als auch die Anzahl der während der Analyse verwendeten Felder auf dem Arbeitsband (Speicherplatz) von Interesse. Da sich die Ergebnisse der theoretischen Untersuchungen von realen Implementierungen meist nur um konstante Werte unterscheiden, verwenden wir für die Beschreibung der Laufzeit bzw. des benötigten Speicherplatzes die folgenden Notationen (alle im Folgenden angeführten Funktionen und Konstanten nehmen wir als nichtnegativ an):

Asymptotisches Verhalten von Funktionen

$T(n) = O(f(n))$, falls es Konstanten c und m so gibt, dass $T(n) \leq cf(n)$ für alle $n \geq m$ gilt.

$T(n) = \Omega(g(n))$, falls es Konstanten c und m so gibt, dass $T(n) \geq cf(n)$ für alle $n \geq m$ gilt.

$T(n) = \theta(h(n))$, wenn $T(n) = O(h(n))$ und $T(n) = \Omega(h(n))$ gilt.

$T(n) = o(p(n))$, wenn $T(n) = O(p(n))$ und $T(n) \neq \theta(p(n))$ gilt.

Mit diesen Notationen gilt unter Anderem Folgendes:

Gilt $T_1(n) = O(f(n))$ und $T_2(n) = O(g(n))$, so gilt auch

1. $T_1(n) + T_2(n) = O(f(n) + g(n))$;
2. $T_1(n) * T_2(n) = O(f(n) * g(n))$.

Zeithierarchien

Betrachtet man das Zeitverhalten $T(n)$ von Turingmaschinen bei der Analyse von Eingabewörtern der Länge n , so ist es naheliegend, die Anzahl der Rechenschritte in Abhängigkeit von n zu betrachten und Funktionen $f(n)$ so zu finden, dass $T(n)=O(f(n))$.

Am bekanntesten sind jene Klassen von Algorithmen, die polynomiell viele Schritte einer deterministischen (P) bzw. nicht-deterministischen Turingmaschine (NP) erfordern; die entsprechenden Sprachfamilien werden üblicherweise mit P bzw. NP bezeichnet. Das Problem, ob diese beiden Klassen zusammenfallen ist wohl das bekannteste noch immer ungelöste Problem der Komplexitätstheorie.

Weitere bekannte Komplexitätsklassen werden durch logarithmische und durch exponentielle Funktionen beschrieben.

Speicherplatzhierarchien

Betrachtet man die Anzahl $S(n)$ von Feldern auf dem Arbeitsband, die eine deterministische oder eine nichtdeterministische Turingmaschine bei der Analyse von Eingabewörtern der Länge n benötigt, so erhält man Speicherplatzhierarchien.

In diesen findet man am untersten Ende die Familie der regulären Sprachen, welche konstanten Funktionen, i.e., der Ordnung $O(1)$, entsprechen (endliche Automaten brauchen ja überhaupt keinen Speicher; außerdem kann eine konstante Menge von Information in der endlichen Kontrolle gespeichert werden). Lineare Funktionen, i.e., der Ordnung $O(n)$, ergeben die linear beschränkten Automaten. Dazwischen liegen logarithmische Komplexitätsklassen, über den linear beschränkten Automaten findet man polynomielle und exponentielle Komplexitätsklassen.

Aristid LINDENMAYER

Aristid Lindenmayer (1925 - 1989)



1968: Mathematical models for cellular interaction in development.
J. Theoret. Biology, 18:280-315, 1968.

Lindenmayer-Systeme (L-Systeme)

Die vom Biologen Aristid Lindenmayer eingeführten Systeme dienten ursprünglich der Beschreibung gewisser Entwicklungsstadien bestimmter Pflanzen. Das Wesentliche dieser parallelen Grammatiken (L-Systeme) besteht in der gleichzeitigen parallelen Anwendung der Produktionen aus einer vorgegebenen Produktionen-Menge auf alle Zeichen einer Satzform.

Ein **ETOL-System** G ist ein Tupel $(V, T, P_1, \dots, P_n, w)$; dabei ist

- V ein Alphabet und
- T das Terminalalphabet;
- P_i , $1 \leq i \leq n$, sind Mengen von kontextfreien Produktionen;
- $w \in V^+$ ist das Axiom.

L-Systeme – Ableitungen, erzeugte Sprachen

Ist $G = (V, T, P_1, \dots, P_n, w)$ ein ET0L-System, dann wird die Ableitungsrelation \Rightarrow wie folgt definiert:

$v \Rightarrow w$ für Wörter $v \in V^+$ und $w \in V^*$ genau dann wenn

- $v = a_1 \dots a_k$ für Symbole $a_i \in V$, $1 \leq i \leq k$,
- $w = w_1 \dots w_k$ für Wörter $w_i \in V^*$, $1 \leq i \leq k$, und
- $a_i \rightarrow w_i \in P_j$, $1 \leq i \leq k$, für ein j mit $1 \leq j \leq n$.

Die **von G erzeugte Sprache** ist die Menge aller Wörter (Satzformen), die in beliebig vielen Schritten aus dem Axiom w abgeleitet werden können und nur aus Terminalsymbolen bestehen:

$$L(G) = \{ v \in T^* \mid w \Rightarrow^* v \}$$

L-Systeme – Varianten, Sprachfamilien

Sei $G = (V, T, P_1, \dots, P_n, w)$ ein ET0L-System:

E steht für „extended“; T steht für „tables“.

$G = (V, T, P_1, \dots, P_n, w)$ heißt

- **EPT0L-System**, wenn keine Produktionen-Menge P_i eine λ -Produktion enthält (P steht für „propagating“);
- **E0L-System**, wenn $n = 1$ (nur ein „table“);
- **EP0L-System**, wenn $n = 1$ und „propagating“;
- **T0L-System**, wenn $V = T$ („non-extended“, „pure“ system),

$G = (V, P_1, \dots, P_n, w)$;

- **PT0L-System**, wenn $V = T$ und „propagating“;
- **0L-System**, wenn $V = T$ und $n = 1$, $G = (V, P, w)$;
- **P0L-System**, wenn $V = T$, $n = 1$ und „propagating“.

Sprachfamilien: $L([E][P][T]0L)$

L-Systeme – Beispiele

- 1) Sei $G = (V, \{a \rightarrow a \mid a \in V\}, w)$ ein 0L-System, dann gilt:

$$L(G) = \{ w \}$$

- 2) Sei $G = (\{a\}, \{a \rightarrow aa\}, a)$ ein 0L-System, dann gilt:

$$L(G) = \{ a^{2^n} \mid n \geq 0 \}$$

- 3) Sei $G = (\{a, b, c\}, \{a \rightarrow abcc, b \rightarrow bcc, c \rightarrow c\}, a)$ ein 0L-System, dann gilt:

$$\{ |w| \mid w \in L(G) \} = \{ n^2 \mid n \geq 1 \}$$

- 4) Sei $G = (\{a, b\}, \{a \rightarrow b, b \rightarrow ab\}, a)$ ein 0L-System, dann gilt:

$$\{ |w| \mid w \in L(G) \} = \{ F(n) \mid n \geq 1 \}$$

Dabei sind die $F(n)$, $n \geq 1$, die Fibonacci-Zahlen, die durch $F(n+2) = F(n+1) + F(n)$ und $F(1) = F(2) = 1$ definiert sind.

Die Beispiele 2) bis 4) erzeugen nicht-kontextfreie Sprachen.

(D)0L-Systeme – Eigenschaften

Die Beispiele stellen **D0L-Systeme** dar
(D steht für „deterministisch“).

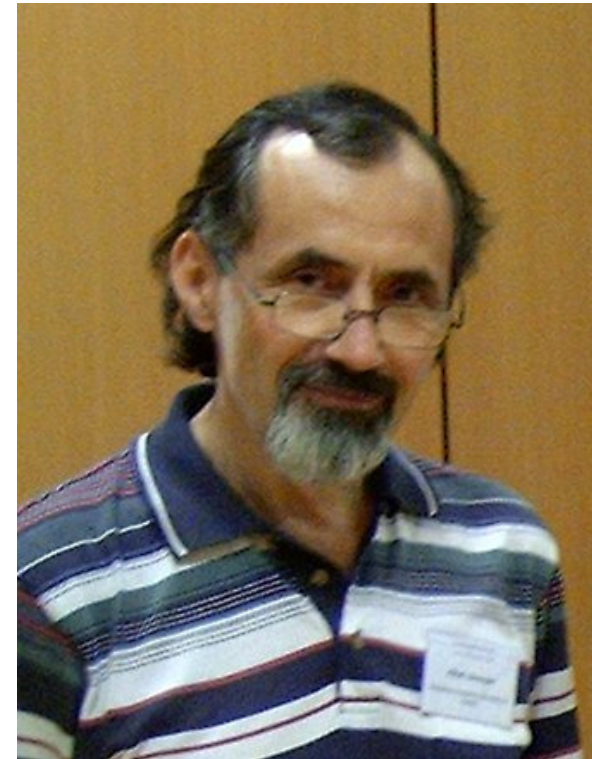
Für jedes Symbol **a** gibt es genau eine Produktion $a \rightarrow w_a$.
Die Menge der Produktionen kann in diesem Falle auch als Homomorphismus **h** auf V^* mit $h(a) = w_a$ interpretiert werden, eine Ableitung entspricht dann der wiederholten Anwendung des Homomorphismus **h**. Von besonderem Interesse ist dann die Folge der **Längen der Wörter** $h^n(w)$.

Satz. $L(0L)$ ist eine Anti-AFL, d.h., unter keiner der eine AFL definierenden Operationen abgeschlossen.

Beweis. Wie zeigen nur die Nicht-Abgeschlossenheit unter **Vereinigung**: Gem. Beispiel 1) sind $\{a\}$ und $\{aa\}$ 0L-Sprachen, aber die Vereinigung $\{a,aa\}$ kann von keinem 0L-System erzeugt werden. □

Gheorghe PĂUN

Gheorghe Păun (*1950)



Jürgen DASSOW, Gheorghe PĂUN:
Regulated Rewriting in Formal Language Theory.
Springer-Verlag, Berlin, 1989.

Kontrollmechanismen - Matrixgrammatiken

steuern die Anwendung von Produktionen in einer Grammatik.

Matrixgrammatiken

Eine Matrix ist eine endliche Folge von (kontextfreien) Produktionen $[p_1, \dots, p_k]$, die in der vorgegebenen Reihenfolge vollständig abgearbeitet werden müssen.

Matrixgrammatik $G_M = (N, T, M, A, F)$

- N ist das Alphabet der Variablen (Nonterminalsymbole),
- T ist das Alphabet der Terminalsymbole,
- M ist eine (endliche) Menge von Matrizen,
 $M = \{m(i) \mid 1 \leq i \leq n\}, m(i) = [m(i,1), \dots, m(i,n(i))]$
- $A \in (N \cup T)^+$ ist das Axiom,
- F ist eine (endliche) Menge von Produktionen, die in den Matrizen vorkommen, d.h., $F \subseteq \{m(i,j) \mid 1 \leq i \leq n, 1 \leq j \leq n(i)\}$.

Ableitungen in einer Matrixgrammatik

Matrixgrammatik $G_M = (N, T, M, A, F)$

Anwendung einer Matrix $m(i) = [m(i,1), \dots, m(i,n(i))]$ auf ein Wort w ergibt das Wort v genau dann, wenn es Wörter $w(0)$ bis $w(n(i))$ derart gibt, dass

- $w(0) = w$,
- $w(n(i)) = v$,
- für alle j mit $1 \leq j \leq n(i)$ gilt *entweder*
 - $w(j)$ ist mittels $m(i,j)$ aus $w(j-1)$ ableitbar *oder*
 - $w(j) = w(j-1)$, $m(i,j)$ ist nicht auf $w(j-1)$ anwendbar und $m(i,j)$ ist aus F .

Die von der Matrixgrammatik G_M **erzeugte Sprache** besteht aus allen Terminalwörtern, die in endlich vielen Schritten mittels der Matrizen in M aus dem Startsymbol ableitbar sind.

Matrixgrammatik $G_M = (N, T, M, A)$ ohne **ac** (also $F = \{ \}$)

ac appearance checking (Vorkommenstest)

Beispiel für Matrixgrammatik ohne ac

Matrixgrammatik $G_M = (\{L, R\}, T, M, LR)$ mit

$M = \{ [L \rightarrow a, R \rightarrow a], [L \rightarrow aL, R \rightarrow aR] \mid a \in T \}$

erzeugt die Sprache $L = \{ww \mid w \in T^+\}$

Ableitungen für $n \geq 1$:

$LR \Rightarrow^{n-1} wLwR$ für ein Wort $w \in T^{n-1}$; daraus sind dann das

- Terminalwort $wawa$ oder

- die Satzform $waLwaR$ für ein $a \in T$ ableitbar;

$waLwaR$ ist nun von der Form $w'Lw'R$ für ein $w' \in T^n$.

Beispiel für Matrixgrammatik mit ac

Matrixgrammatik $G_M = (\{A, B, F, X, Y, Z\}, \{a\}, M, XA, F)$ mit

$M = \{ [X \rightarrow X, A \rightarrow BB], [X \rightarrow Y, A \rightarrow F], [X \rightarrow Z, A \rightarrow F],$
 $[Y \rightarrow Y, B \rightarrow A], [Y \rightarrow X, B \rightarrow F], [Z \rightarrow Z, B \rightarrow a], [Z \rightarrow \varepsilon, B \rightarrow F] \}$

und $F = \{A \rightarrow F, B \rightarrow F\}$

erzeugt die Sprache $L = \{a^{2^n} \mid n \geq 1\}$.

Ableitungen für $n \geq 1$ (man beachte, dass $A^{2^0} = A^1 = A$):

$XA^{2^{n-1}} \Rightarrow^{2^{n-1}} XB^{2^n} \Rightarrow YB^{2^n} \Rightarrow^{2^n} YA^{2^n} \Rightarrow XA^{2^n}$ oder

$XA^{2^{n-1}} \Rightarrow^{2^{n-1}} XB^{2^n} \Rightarrow ZB^{2^n} \Rightarrow^{2^n} Za^{2^n} \Rightarrow a^{2^n}$

Von Matrixgrammatiken erzeugte Sprachen

Satz. Matrixgrammatiken mit ac können jede rekursiv aufzählbare Sprache erzeugen.

Über einem einelementigen Terminalalphabet können Matrixgrammatiken ohne ac nur reguläre Sprachen erzeugen.

Die Sprache $L = \{a^{2^n} \mid n \geq 1\}$ kann daher nicht von einer Matrixgrammatik ohne ac erzeugt werden, allerdings wie gezeigt von einer Matrixgrammatik mit ac .

Kontrollmechanismen

Jürgen DASSOW, Gheorghe PĂUN:
Regulated Rewriting in Formal Language Theory.
Springer-Verlag, Berlin, 1989.

graphkontrollierte Grammatiken

programmierte Grammatiken

Grammatiken mit Kontextbedingungen