185.263 **Automaten und Formale Sprachen**

VO 2.0 + UE 1

Rudolf Freund, Marion Oswald

Übersicht

Grundlegende Definitionen:

- elementare Mengentheorie
- Arten von Beweisen
- Graphen

Formale Sprachen

- Grammatiken
- endliche Automaten, reguläre Mengen
- Pumping Lemmas
- Kellerautomaten
- Normalformen
- CHOMSKY-Hiearchie
- Abschlusseigenschaften von Sprachfamilien
- Turingmaschinen
- regulierte Grammatiken
- Lindenmayer-Systeme (parallele Grammatiken)

Elementare Mengentheorie - Cantor

GEORG CANTOR (1845 - 1918)

„*Eine Menge ist eine Zusammenfassung von bestimmten wohlunterschiedenen Objekten unseres Denkens oder unserer Anschauung (welche die Elemente der Menge genannt werden) zu einem Ganzen.*“

1874: Über eine Eigenschaft des Inbegriffs aller reellen algebraischen Zahlen.
J. reine angew. Math., 77, 258-262

Elementare Mengentheorie

Eine Menge ist eine Gruppe von Objekten, die als Einheit repräsentiert werden.

Mengen können beliebige Typen von Objekten beinhalten, inklusive Zahlen, Symbole, und auch andere Mengen.

Die Objekte einer Menge bezeichnet man als Elemente.

Um einige Konzepte zu veranschaulichen, verwenden wir Venn-Diagramme. Dabei werden Mengen als Kreise dargestellt.

Mengentheorie - Zugehörigkeit

Mengen können formal auf verschiedene Arten beschrieben werden.

Eine Möglichkeit besteht darin, die Elemente aufzulisten:

{7,21,57} enthält die Elemente 7,21,57

Notation:

$\in$ ... Mengen-Zugehörigkeit $7 \in \{7,21,57\}$

$\notin$ ... Nicht-Zugehörigkeit $5 \notin \{7,21,57\}$

Mengentheorie - (Multi-) Mengen

Die Reihenfolge der Elemente und auch deren Wiederholung in einer Menge ist nicht von Bedeutung.

$$\{7,7,57,7,21\} = \{7,21,57\}$$

Bemerkung:

{7} und {7,7} sind identische Mengen, aber unterschiedliche Multimengen (multisets).

Mengentheorie – Kardinalität von Mengen

Eine unendliche Menge enthält unendlich viele Elemente.

Beispiel:

Menge der natürlichen Zahlen: $\mathbf{N} = \{0, 1, 2, 3,...\}$

Menge der natürlichen Zahlen $\geq$ k: $\mathbf{N_k} = \{k, k+1, k+2,...\}$

Menge der natürlichen Zahlen zwischen k und m: [k..m]

Besteht eine Menge aus den Elementen $a_1,...,a_n$ für $n \geq 1$, dann schreibt man $A = \{a_1,...,a_n\}$ (endliche Menge).

Die Anzahl der Elemente einer endlichen Menge bezeichnet man als Kardinalität, geschrieben als card(M).

Die Menge, die keine Elemente enthält, wird als leere Menge bezeichnet und geschrieben als { } oder Ø.

Beispiel: $\text{card}(A) = n$ $\text{card}(\emptyset) = 0$

Mengentheorie – Mengenvorschrift

Mengen können auch durch eine Mengenvorschrift angegeben werden:

$\{ x \mid E(x) \}$	Menge der Objekte x für die E(x) gilt
$\{ x \in A \mid E(x) \}$	Menge der Objekte aus der Menge A für die E(x) gilt
$\{ x \in A \mid E(x) \}$	ist äquivalent zu $\{ x \mid E(x)$ und $x \in A \}$

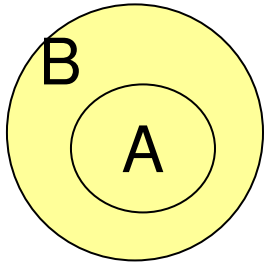
Beispiel: $\{ n \mid n = m^2$ für $m \in \mathbf{N} \}$

Mengentheorie - (echte) Teilmengen

Für zwei Mengen A und B gilt:

A ist eine Teilmenge von B, wenn jedes Element von A auch ein Element von B ist.

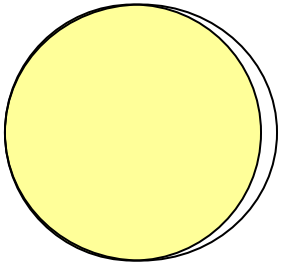
$A \subseteq B$



A ist eine echte Teilmenge von B, wenn A eine Teilmenge von B ist, die nicht gleich B ist.

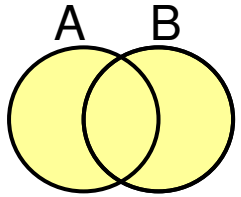
$A \subset B$

$A = B$ wenn $A \subseteq B$ und $B \subseteq A$



Mengentheorie – Operationen auf Mengen

Vereinigung: $A \cup B := \{ x \mid x \in A \text{ oder } x \in B \}$



Vereinigung endlich vieler Mengen $M_1,..,M_n$

$$\cup_{i \in \{1,...,n\}} M_i$$

Vereinigung abzählbar unendlich vieler Mengen $M_1,M_2,...$

$$\cup_{i \in N_1} M_i \quad \text{bzw.} \quad \cup_{i \in N_1} M_i$$

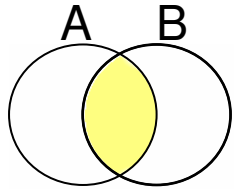
$$\cup_{i \in N_1} M_i = \{ x \mid x \in M_i \text{ für ein } i \in N_1 \}$$

Für beliebige Mengen A, sodass M_a für jedes $x \in A$ eine Menge ist, definiert man

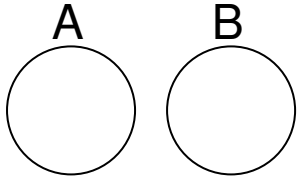
$$\cup_{i \in A} M_a = \{ x \mid x \in M_a \text{ für ein } a \in A \}$$

Mengentheorie – Operationen auf Mengen

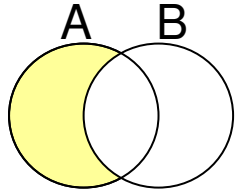
Durchschnitt: $A \cap B := \{ x \mid x \in A \text{ und } x \in B \}$



analog $\cap_{i \in \{1,\ldots,n\}} M_i$, $\cap_{i \in N_1} M_i$, $\cap_{i \in A} M_a$

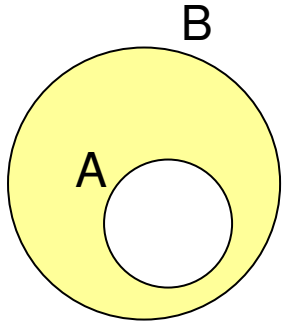


A und B sind disjunkt wenn $A \cap B = \varnothing$



Differenzmenge: $A - B := \{ x \mid x \in A \text{ und } x \notin B \}$

Gilt $A \subseteq B$, so nennt man $B - A$ das Komplement von A (bezüglich B), geschrieben $\overline{A}$



Sequenzen und Tupel

Eine Sequenz von Objekten ist eine geordnete Liste von Objekten.

Beispiel: (7,21,57) ist eine andere Sequenz als (7,57,21) oder auch (7,7,21,57)

Endliche Sequenzen werden auch Tupel genannt.
Eine Sequenz mit k Elementen wird k-Tupel genannt.
((7,21,57) ist also ein 3-Tupel oder Tripel; ein 2-Tupel wird auch Paar genannt)

Mengen und Tupel können auch Elemente von anderen Mengen und Tupeln sein.

Mengentheorie – Operationen auf Mengen

Potenzmenge von A:

Menge aller Teilmengen von A 2^A bzw. $\wp(A)$

Beispiel:

$A = \{ 0,1 \}$ $2^A = \{ \{ \}, \{0\}, \{1\}, \{0,1\} \}$

Kartesisches Produkt (Kreuzprodukt):

$A \times B := \{ x \mid x = (a,b) \text{ mit } a \in A \text{ und } b \in B \}$

$M_1 \times \ldots \times M_n := \{ x \mid x = (x_1,\ldots,x_n) \text{ mit } x_i \in M_i \text{ für } 1 \leq i \leq n \}$

$card(A) = n, card(B) = m$

$card(2^A) = 2^n$ $card(A \times B) = nm$

Relationen und Funktionen

Relation R auf S: $R \subseteq S \times S$

(Definitionsbereich (domain) × Wertebereich (range))

Menge von Paaren (a,b), wobei eine Beziehung zwischen a und b aus S besteht oder nicht besteht.
Schreibweise: aRb

Eigenschaften von Relationen

1. Reflexiv wenn aRa für alle a aus S gilt
2. Irreflexiv wenn aRa für alle a aus S falsch ist
3. Transitiv wenn aRc aus aRb und bRc folgt
4. Symmetrisch wenn bRa aus aRb folgt
5. Asymmetrisch wenn aRb impliziert, dass bRa nicht gilt.

Beispiel:

Ordnungsrelation < auf der Menge der ganzen Zahlen ist transitiv und asymmetrisch (und daher irreflexiv)

Relationen und Funktionen

Äquivalenzrelation:
Relation, die reflexiv, symmetrisch und transitiv ist.

Wichtige Eigenschaft einer Äquivalenzrelation R auf S:
S wird durch R in disjunkte, nichtleere Äquivalenzklassen unterteilt.
$S = S_1 \cup S_2 \cup \ldots$, wobei für i und j mit $i \neq j$ gilt:

1. $S_i \cap S_j = \{\}$
2. Für jedes a und b aus S_i ist aRb wahr
3. Für jedes a aus S_i und b aus S_j ist aRb falsch

Beispiel: Kongruenz modulo einer ganzen Zahl m

$i \equiv j \pmod{m}$ falls i und j ganze Zahlen sind mit der Eigenschaft, dass i - j durch m teilbar ist.

Relationen und Funktionen

Hüllen von Relationen

E..Menge von Eigenschaften von Relationen über S.
E-Hülle von R ist die kleinste Relation R‘, die alle Paare von R enthält und die Eigenschaften aus E besitzt

Transitive Hülle von R, geschrieben R^+:

1. Falls (a,b) in R ist, so ist (a,b) auch in R^+
2. Falls (a,b) und (b,c) in R^+ sind, so ist (a,c) auch in R^+
3. Nichts ist in R^+, außer es folgt aus 1. und 2.

Reflexive und Transitive Hülle von R, geschrieben R^*:

$R^+ \cup \{(a,a) \mid a \in S\}$

Beispiel: Sei R = { (1,2),(2,2),(2,3) } Relation auf {1,2,3}

R^+ = { (1,2),(2,2),(2,3),(1,3) }

R^* = { (1,1),(1,2),(1,3),(2,2),(2,3),(3,3) }

Relationen und Funktionen

Unter einer Funktion oder Abbildung $f: X \to Y$ einer Menge X in eine Menge Y (festgelegt durch $f \subseteq X \times Y$) versteht man eine Vorschrift, die jedem Element x von X ein eindeutig bestimmtes Element y aus Y zuordnet: $f(x)=y$
(f(x) ist der Funktionswert von f an Stelle x)

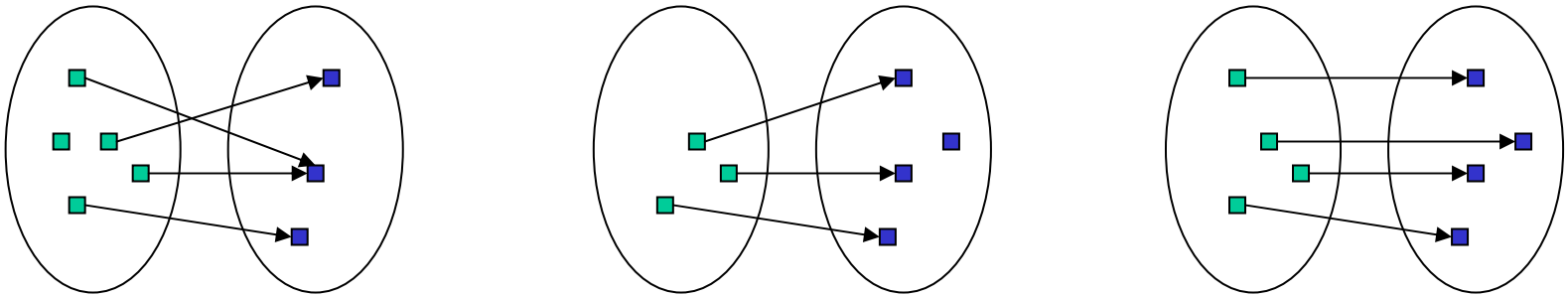
Funktionswerte von f (*range*):
$rng(f) = \{ y \mid (x,y) \in f \text{ für ein } x \in X\}$

$f: X \to Y$ und $g: X \to Y$ heißen gleich, symbolisch $f \equiv g$,
wenn $f(x) = g(x)$ für alle $x \in X$.

Relationen und Funktionen

Man nennt f

- surjektiv von X auf Y, wenn rng(f) =Y
- injektiv wenn für beliebige $x_1, x_2 \in X$ aus $f(x_1) = f(x_2)$ auch $x_1 = x_2$ folgt.
- bijektiv wenn f sowohl surjektiv als auch injektiv ist.



Beweise

Ein Beweis ist ein überzeugendes logisches Argument, dass eine Aussage wahr ist.

Beweis durch Konstruktion

Viele Sätze behaupten die Existenz von bestimmten Typen von Objekten. Ein Weg, solche Sätze zu beweisen, besteht darin, zu zeigen, wie man diese Objekte konstruiert.

Indirekter Beweis (Beweis durch Widerspruch)

Wir nehmen an, dass der Satz wahr ist, und zeigen dann, dass diese Annahme zu einer offensichtlich falschen Konsequenz, einem Widerspruch, führt.

Beweise - Gauss

CARL FRIEDRICH GAUSS (1777-1855)

Die Lehrer von Gauss waren sehr erstaunt, als dieser bereits im Alter von sieben Jahren die Zahlen von 1 bis 100 im Handumdrehen summieren konnte.

Er erkannte nämlich sofort, dass diese Summe aus 50 Zahlenpaaren bestand, die jeweils 101 ergaben.

Beweise: Induktion

Um eine Aussage A(n) für alle $n \in \mathbf{N}$ (bzw. für alle $n \in \mathbf{N}_k$) zu beweisen: (A(n) hängt von der ganzen Zahl $n \geq k$ ab.)

Induktionsbasis:
A(m) ist für $m = k$ richtig.

Induktionshypothese:
Nehme an, A(m) gilt für $m = n$.

Induktionsbehauptung:
Zeige, A(m) gilt dann auch für $m = n+1$.

Induktionsprinzip:
Dann ist die Aussage A(m) für alle $m \geq k$ richtig.

Beweise: Induktion

Beispiel:

Aussage: für alle positiven natürlichen Zahlen gilt

$$1+2+...+n = (n\ (n+1))/2$$

Induktionsbasis:

A(n) ist offensichtlich für n = 1 richtig: $1 = (1*2)/2$

Induktionshypothese:

$1 + 2 +...+ n = (n\ (n+1))/2$ Umformen ergibt:

Induktionsbehauptung:

$1 + 2 +...+ n + n + 1 = ((n + 1)\ (n+1 + 1))/2$

Einsetzen und Umformen ergibt:

$1 + 2 +...+ n + n + 1 = (n\ (n+1))/2 + n + 1 =$

$(n+1)((n/2)+1) = (n+1)((n+2)/2) = ((n + 1)\ (n+1 + 1))/2$

Somit haben wir mittels Induktion bewiesen:

$1+2+...+n = (n\ (n+1))/2$ gilt für alle natürlichen Zahlen $n \geq 1$.

Beweise: Induktion - Aufgaben

Aufgabe INDA: Zeigen Sie mittels Induktion, dass für alle positiven natürlichen Zahlen n gilt:

$1^3+2^3+...+n^3 = ((n\ (n+1))/2)^2$

Aufgabe INDB**: Zeigen Sie mittels Induktion nach n, dass für alle positiven natürlichen Zahlen n und k gilt:

$1^k+2^k+...+n^k = ((n\ (n+1))/2)\ p_k(n)$

wobei $p_k(x)$ ein Polynom vom Grad k-1 in x ist.

Aufgabe INDC: Zeigen Sie mittels Induktion nach n, dass eine Zahl k so gewählt werden kann, dass für alle positiven natürlichen Zahlen $n \geq k$ gilt:

$n!\ \geq 2^n$ ($n! = n * (n-1) \ldots 2 * 1$)

Beweise: Generalisierung

Kann man für eine beliebige natürlich Zahl n (≥k) zeigen, dass die Aussage A(n) gilt, dann gilt sie für alle natürlichen Zahlen n (≥k) .

Beispiel: Es gibt beliebig große Primzahllücken.

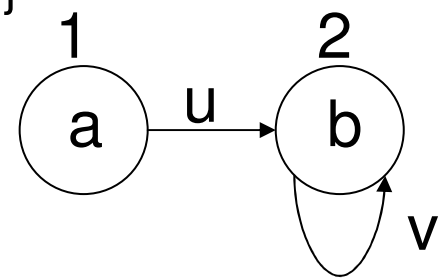
Wir nehmen eine beliebige natürlich Zahl n ≥ 1 und zeigen, dass es n-1 aufeinanderfolgende Zahlen gibt, die alle keine Primzahlen sind: Betrachte
$n! + k$ für $k = 2,\ldots, n$
Klarerweise gilt $k/(n!+k)$, da k ein Faktor von n! ist. q.e.d.

Graphen

Seien V und W endliche Mengen. Ein markierter gerichteter Graph g über V und W ist ein Tripel (K,E,L) wobei
- K die Menge der Knoten
- $E \subseteq K \times K \times W$ die Menge der Kanten
- $L: K \rightarrow V$ die Knotenmarkierungsfunktion ist.

Ein Element $(x,w,y) \in E$ stellt eine gerichtete Kante vom Knoten x zum Knoten y mit Markierung w dar.

Beispiel: $g = (\{1,2\}, \{(1,u,2),(2,v,2)\}, \{(1,a),(2,b)\})$ ist ein markierter gerichteter Graph über $W \supseteq \{a,b\}$ und $V \supseteq \{1,2\}$

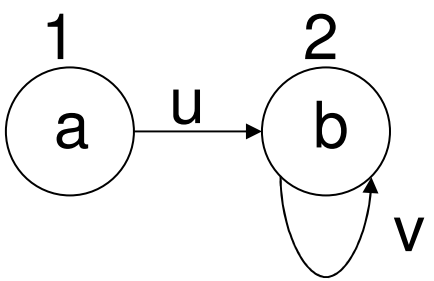


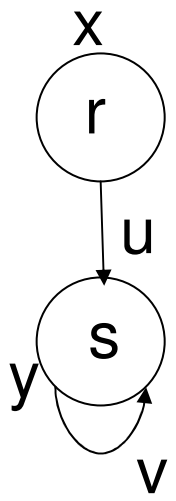
Übergangsmatrix:

δ	u	v
1	{2}	{ }
2	{ }	{2}

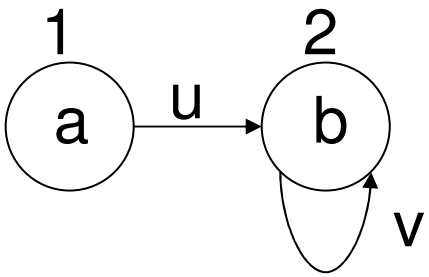
Graphen: Isomorphie und Äquivalenz

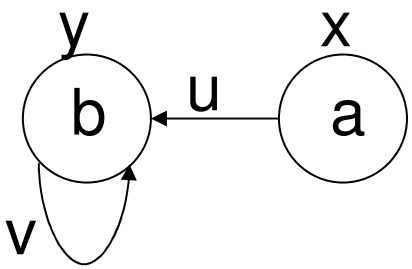
Zwei markierte gerichtete Graphen g = (K,E,L) und g‘= (K‘,E‘,L‘) über V und W heißen isomorph, wenn es eine bijektive Abbildung f: K $\rightarrow$K‘ gibt, sodass für alle k,l $\in$ K und w $\in$ W gilt:
(k,w,l) $\in$ E $\leftrightarrow$ (f(k),w,f(l)) $\in$ E‘.





Gilt außerdem L(k)=L‘(f(k)) für alle k $\in$ K, so heißen g und g‘ äquivalent.





Mächtigkeit von Mengen

Zwei Mengen A und B heißen gleichmächtig, wenn es eine bijektive Abbildung von A nach B gibt.

Jede Menge, die gleichmächtig mit **N** ist, heißt eine abzählbare Menge.

Jede unendliche, nicht abzählbare Menge heißt überabzählbar.

Beispiel:
Die Menge aller reellen Zahlen ist gleichmächtig mit der Menge der reellen Zahlen in jedem beliebigen Intervall
(a,b) = { x | x reell und a < x < b}.
Beide Mengen sind nicht abzählbar.

Diagonalverfahren

Sei $M = \{0,1\}^{\mathbf{N}}$ die Menge aller Funktionen $f: \mathbf{N} \rightarrow \{0,1\}$.
Dann ist M nicht abzählbar.

Cantorsches Diagonalverfahren

Angenommen, es gibt eine bijektive Funktion $g: \mathbf{N} \rightarrow M$. Sei dann die Funktion $h: \mathbf{N} \rightarrow \{0,1\}$ folgendermaßen definiert: $h(n) = ([g(n)]\,(n) + 1 \pmod 2)$, i.e.,

$$h(n) = \begin{cases} 1 & \text{falls } [g(n)](n) = 0 \\ 0 & \text{falls } [g(n)](n) = 1 \end{cases}$$

Dann ist h eine Funktion von $\mathbf{N}$ in $\{0,1\}$, die somit in M sein müsste, allerdings wurde h so definiert, dass für kein $n \in \mathbf{N}$ $h = g(n)$ sein kann, denn für jedes $n \in \mathbf{N}$ gilt $h(n) \rightarrow [g(n)](n)$, was aber $h \notin M$ bedeutet.
Somit ergibt sich aber ein Widerspruch zur Annahme, dass die Menge M abgezählt werden kann. □

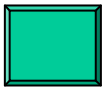
Cantorsches Diagonalverfahren

	g(1) $\neq$ h	g(2) $\neq$ h	…	g(n) $\neq$ h	…
1	[g(1)](1) $\neq$ h(1)	[g(2)](1)	…	[g(n)](1)	…
2	[g(1)](2)	[g(2)](2) $\neq$ h(2)	…	[g(n)](2)	…
…	…	…	…	…	…
n	[g(1)](n)	[g(2)](n)	…	[g(n)](n) $\neq$ h(n)	…
…	…	…	…	…	…

Folgerung

Sei A eine abzählbar unendliche Menge. Dann ist $2^{\mathbf{A}}$ überabzählbar.

Sei $A = \{x_{\mathbf{n}} \mid n \in \mathbf{N}\}$ und für jede Teilmenge $M \subseteq A$ die Funktion $\chi_{\mathbf{M}}: \mathbf{N} \rightarrow \{0,1\}$ durch
$\chi_{\mathbf{M}} = 0$, falls $x_{\mathbf{n}} \notin M$ und
$\chi_{\mathbf{M}} = 1$, falls $x_{\mathbf{n}} \in M$,
definiert; $\chi_{\mathbf{M}}$ ist die *charakteristische Funktion* bezüglich A. Dann ist durch $g: 2^{\mathbf{A}} \rightarrow \{0,1\}^{\mathbf{N}}$ mit $g(m) = \chi_{\mathbf{M}}$ eine bijektive Abbildung zwischen $2^{\mathbf{A}}$ und $\{0,1\}^{\mathbf{N}}$ definiert und $2^{\mathbf{A}}$ nach dem vorigen Satz somit ebenfalls überabzählbar.□



Formale Sprachen

AXEL THUE (1863 - 1922)

„*The further removed from usefulness or practical application the more important* “

1906: Über unendliche Zeichenreihen.
Norske Vid. Selsk. Skr., I Mat. Nat. Kl., Kristiana 7, 1-22

Formale Sprachen - Symbole und Wörter

Ein Alphabet ist eine endliche Menge von Symbolen.

z.B.: $\Sigma_1 = \{0,1\}$
$\Sigma_2 = \{a,...,z\}$
$\Sigma_3 = \{0,1,a,b,c\}$

Ein Wort über einem Alphabet ist eine endliche Folge von Symbolen über diesem Alphabet.

z.B.: 01001 ist ein Wort über Σ_1
abbeb ist ein Wort über Σ_2

Formale Sprachen - Wörter

Ist w ein Wort über Σ, dann ist die Länge von w, in Zeichen |w|, die Anzahl der Symbole, die w enthält.
z.B.: $\Sigma = \{0,1\}$ w = 0101 |w| = 4

Hat ein Wort w über Σ die Länge n, dann schreiben wir $w = a_1 a_2 ... a_n$ wobei jedes $a_i \in \Sigma$.

Das Wort mit der Länge 0 heißt Leerwort, geschrieben ε oder λ, d.h. $|\varepsilon| = 0$.

Formale Sprachen - Konkatenation

Haben wir ein Wort x der Länge n und ein Wort y der Länge m, dann ist die Konkatenation von x und y das Wort, das man durch Hintereinanderschreiben von x und y erhält: $x \cdot y = xy \quad |xy| = n+m$

Um ein Wort mit sich selbst mehrere Male zu verketten, benützen wir folgende Notation (Potenzbildung):

$$w^k = \underbrace{w \cdot w \cdot \ldots \cdot w}_{k}$$

$$w^0 = \varepsilon \qquad w^n = w \cdot w^{n-1}$$

Formale Sprachen - Sprache

Σ^* ist die Menge aller Wörter über Σ
$\Sigma^+ = \Sigma^* - \{\varepsilon\}$

Eine formale Sprache ist eine beliebige Teilmenge L von Σ^*.

Es gilt: $L \subseteq \Sigma^*$

$\Sigma^* = \cup_{n \in N} \Sigma^n$ wobei $\Sigma^n = \{x_1 x_2 ... x_n \mid x_i \in A \text{ und } 0 \leq i \leq n\}$

$\Sigma^+ = \cup_{n \in N_1} \Sigma^n$

Formale Sprachen

Sei $w \in \Sigma^*$ und $w = xuy$ für Wörter $x,u,y \in \Sigma^*$.
Dann heißt x Präfix, u Teilwort und y Suffix von w.

Für $w \in \Sigma^*$ und $a \in \Sigma$ bezeichnen wir mit $|w|_a$ die Anzahl der Symbole a in w.

Sei $w = a_1a_2...a_{n-1}a_n$ aus Σ^*. Dann ist $w^r = a_n\, a_{n-1} \ldots a_2a_1$ das Spiegelbild von w.

Ein Wort $w \in \Sigma^*$ heißt Palindrom, wenn $w = w^r$ gilt.

Operationen auf Sprachen

Seien A und B Sprachen.

Vereinigung: $A \cup B = \{ x \mid x \in A \text{ oder } x \in B \}$

Konkatenation: $A \cdot B = \{ xy \mid x \in A \text{ und } x \in B \}$

Stern: $A^* = \{x_1 x_2 ... x_k \mid x_i \in A \text{ und } 0 \leq i \leq k\}$

Beispiel:

$\Sigma = \{a,b,...,z\}$.

Wenn A = { good, bad} und B = { girl, boy } dann ist

$A \cup B$ = { good, bad, boy, girl }}

$A \cdot B$ = {goodgirl, goodboy, badgirl, badboy,}

A^* = { ε , good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad,...}

Operationen auf Sprachen - Rechenregeln

$$A \cup B = B \cup A$$

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

$$A \cdot (B \cup C) = A \cdot B \cup A \cdot C$$

$$(B \cup C) \cdot A = B \cdot A \cup C \cdot A$$

$$\{\varepsilon\} \cdot A = A$$

$$A \cdot \{\varepsilon\} = A$$

$$(A \cup \{\varepsilon\})^* = A^*$$

$$(A^*)^* = A^*$$

$$A \cdot A^* = A^+$$

$$A^* \cdot A = A^+$$

$$A^+ \cup \{\varepsilon\} = A^*$$

Die Menge der formalen Sprachen über einem Alphabet T bildet daher einen nichtkommutativen Semiring mit der Vereinigung als Addition und dem neutralen Element {} sowie mit der Konkatenation als Multiplikation und dem Einheitselement {ε}.

Formale Sprachen - Induktive Definition

Sind eine Menge B und Operatoren auf Teilmengen von B gegeben, so kann man sich natürlich fragen, ob die Anwendung der Operatoren auf diese Teilmengen von B wieder Teilmengen von B ergibt.

Sei B eine Menge und $f:B^n \rightarrow B$ eine Funktion. Eine Menge $A \subseteq B$ heißt abgeschlossen unter f, wenn gilt: aus $x_1,...,x_n \in A$ folgt $f(x_1,...,x_n) \in A$

Schema der induktiven Definition:
A ist die kleinste Menge für die gilt:
(1) $A_0 \in A$
(2) Wenn $x_1,...,x_n \in A$, dann $f(x_1,...,x_n) \in A$.

Komponenten: Grundmenge, Abschlusseigenschaft, Minimalitätsbedingung

Formale Sprachen - Induktive Definition: Beispiele

Beispiel:

Die Menge der Palindrome ist die kleinste Menge, für die gilt:

(1) ε ist ein Palindrom.

(2) Für jedes Symbol a ist a ein Palindrom.

(3) Ist a ein Symbol und x ein Palindrom dann ist auch axa ein Palindrom.

Beispiel:

Die Menge der wohlgeformten Klammerausdrücke (WKA) über dem Alphabet { [,] } ist die kleinste Menge, für die gilt:

(1) ε ist ein WKA.

(2) Ist w ein WKA, dann ist auch [w] ein WKA.

(3) Sind w und v WKA, dann ist auch wv ein WKA.

Induktive Definition: Reguläre Mengen

Die Menge $\boldsymbol{L}_{\mathbf{reg}}(\Sigma)$ regulärer Mengen über Σ ist die kleinste Menge, sodass

(1) $\emptyset, \{a\} \in \Sigma$ für alle $a \in \Sigma$.

(2) Wenn A und B $\in \boldsymbol{L}_{\mathbf{reg}}(\Sigma)$, dann sind auch

$A \cup B, AB, A^* \in \boldsymbol{L}_{\mathbf{reg}}(\Sigma)$.

Um die Mengen in $\boldsymbol{L}_{\mathbf{reg}}$ formal zu beschreiben, verwendet man folgende Notation:

Die Menge REG(Σ) der regulären Ausdrücke über Σ ist die kleinste Menge, sodass

(1) $\emptyset, \boldsymbol{a} \in \mathrm{REG}(\Sigma)$ für alle $a \in \Sigma$

(2) Wenn r und s $\in$ REG(Σ), dann sind auch

$(r + s), (r \cdot s), (r^*) \in \mathrm{REG}(\Sigma)$.

Grammatiken

Das fundamentale Modell zur Beschreibung von formalen Sprachen durch Erzeugungsmechanismen sind Grammatiken.

Eine Grammatik ist ein Quadrupel (N,T,P,S) wobei
- N das Alphabet der Nonterminale (Variablen)
- T das Alphabet der Terminalsymbole
- P eine Menge von Produktionen
- $S \in N$ das Startsymbol ist.

Üblicherweise ist $N \cap T = \{\}$
Wir definieren $V := N \cup T$

$P \subseteq V^+ \times V^*$
d.h., jede Produktion p aus P ist der Gestalt $p = (\alpha, \beta)$ mit $\alpha \in V^+$ und $\beta \in V^*$.

Anstelle von (α, β) schreiben wir auch $\alpha \rightarrow \beta$.

Grammatiken: Ableitung

Sei G = (N,T,P,S) eine Grammatik.

Ein Wort $w \in V^*$ heißt ableitbar in G aus dem Wort $v \in V^+$, in Symbolen $v \underset{G}{\Rightarrow} w$, falls Wörter $x,y \in V^*$ existieren, sodass $v = x\,\alpha y$ und $w = x\,\beta y$ für eine Produktion $(\alpha, \beta) \in P$ gilt.

Durch $\underset{G}{\Rightarrow}$ wird eine Relation über V* definiert.

$\overset{*}{\underset{G}{\Rightarrow}}$ Reflexive und transitive Hülle von $\underset{G}{\Rightarrow}$

$\underset{G}{\Rightarrow}$ Ableitung in einem Schritt

$\overset{n}{\underset{G}{\Rightarrow}}$ Ableitung in n Schritten

Ist G eindeutig aus dem Zusammenhang erkennbar, so schreiben wir $\Rightarrow$ statt $\underset{G}{\Rightarrow}$ etc.

Grammatiken: erzeugte Sprache

Sei G = (N,T,P,S) eine Grammatik.

Gilt $S \overset{*}{\underset{G}{\Rightarrow}} w$ für ein Wort $w \in V^*$, so nenn man w Satzform.

Menge aller in n Schritten ableitbaren Satzformen:
$SF(G,n) = \{w \in V^* \mid S \overset{n}{\underset{G}{\Rightarrow}} w \}$

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

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

Grammatiken: Beispiel 1

Beispiel:

$G_1 = (\{ S \}, \{a\}, \{S \rightarrow \varepsilon , \ S \rightarrow aS\}, S)$

$L(G_1) = \{ \varepsilon \} \cup \{ a^n \mid n \geq 1 \} = \{a\}^*$

Alle in G möglichen Ableitungen sind von der Gestalt

$S \Rightarrow \varepsilon \quad$ bzw. $\quad S \Rightarrow aS \Rightarrow^n a^{n+1}S \Rightarrow a^{n+1}$

für ein $n \in \mathbf{N}$.

Grammatiken: Beispiel 2

Beispiel:

$G_2 = (\{S\}, \{a,b\}, \{ S \rightarrow aSb, S \rightarrow \varepsilon \}, S)$

$L(G_2) = \{ a^n b^n \mid n \in \mathbf{N} \}$

Alle in G möglichen Ableitungen sind von der Gestalt

$S \Rightarrow^n a^nSb^n \Rightarrow a^nb^n$ für alle $n \in \mathbf{N}$.

Formaler Beweis mittels natürlicher Induktion:

Menge aller Satzformen nach genau n Schritten:

$SF(G_2,n) = \{ a^nSb^n , a^{n-1}b^{n-1} \}$

Grammatiken: Beispiel 2 (Induktion)

$G_2 = (\{S\}, \{a,b\}, \{ S \rightarrow aSb, S \rightarrow \varepsilon \}, S)$

Formaler Beweis mittels natürlicher Induktion.
Menge aller Satzformen nach genau n ≥ 1 Schritten:

$SF(G_2, n) = \{ a^n S b^n , a^{n-1} b^{n-1} \}$

Induktionsbasis: $SF(G_2,1) = \{ a^1 S b^1, \varepsilon \}$

Induktionshypothese: $SF(G_2,n) = \{ a^n S b^n , a^{n-1} b^{n-1} \}$

Induktionsbehauptung: $SF(G_2,n+1) = \{ a^{n+1} S b^{n+1} , a^n b^n \}$

Beweis: Das Wort $a^{n-1} b^{n-1}$ ist terminal und daher nicht mehr weiter ableitbar. Aus $a^n S b^n$ ist ableitbar:

mittels $S \rightarrow aSb$: $a^{n+1} S b^{n+1}$
mittels $S \rightarrow \varepsilon$: $a^n b^n$

□

Grammatiken: Beispiel 3

Beispiel:

$G_3 = (\{S,A,C\}, \{a,b,c\}, P_3, S)$ wobei

$P_3 = \{ S \rightarrow abc,\ S \rightarrow aAbc,\ A \rightarrow aAbC,\ A \rightarrow abC,$
$Cb \rightarrow bC,\ Cc \rightarrow cc\}$

$L(G_3) = \{a^n b^n c^n \mid n \in \mathbf{N}_1 \}$

Alle in G möglichen Ableitungen sind von der Gestalt

$S \Rightarrow abc$

bzw. für $n \geq 2$:

$S \Rightarrow aAbc \Rightarrow^{n-2} a^{n-1}A(bC)^{n-2}bc \Rightarrow a^nA(bC)^{n-1}bc \Rightarrow^* a^n b^n c^n$

Grammatik-Typen

Die vorhergehenden Beispiele zeigen, dass zur Erzeugung bestimmter formaler Sprachen Produktionen mit wachsender Komplexität benötigt werden.

Aufgrund dieser Komplexität der Produktionen können wir verschiedene Typen von Grammatiken definieren.

Typ-i-Grammatiken

Sei G=(N,T,P,S) eine Grammatik.
Dann heißt G auch unbeschränkte Grammatik (Typ-0)

Gilt für alle Produktionen $(\alpha, \beta) \in P$

- $|\alpha| \leq |\beta|$ so heißt G monoton;
- $\alpha = uAv$ und $\beta = uwv$ für ein $A \in N$, $w \in V^+$ und $u,v \in V^*$

  so heißt G kontextsensitiv (Typ-1) (kommt S nicht auf der rechten Seite einer Produktion vor, so ist auch $S \rightarrow \varepsilon$ erlaubt);
- $A \rightarrow \beta$ für ein $A \in N$, so heißt G kontextfrei (Typ-2);
- $A \rightarrow aB$ oder $A \rightarrow \varepsilon$ für $A,B \in N$ und $a \in T$, so heißt G regulär (Typ-3).

Erzeugte Sprachen

Eine formale Sprache heisst rekursiv aufzählbar, monoton (kontextsensitv), kontextfrei bzw. regulär, wenn sie von einer Typ-0-, Typ-1-, Typ-2-, bzw. Typ-3-Grammatik erzeugt wird.

Aufgrund der Definition können wir nun die einzelnen Sprachen aus den vorigen Beispielen klassifizieren:

Es ergibt sich, dass
$L(G_1)$ regulär
$L(G_2)$ kontextfrei und
$L(G_3)$ monoton ist.

Zwei Grammatiken G und G‘ heißen äquivalent wenn
$L(G) = L(G')$

Äquivalenz von kontextsensitiven und monotonen Grammatiken

Nach Definition ist jede kontextsensitive Grammatik auch eine monotone Grammatik.
Es gilt allerdings auch die Umkehrung:

Satz. Zu jeder monotonen Grammatik kann man eine äquivalente kontextsensitive Grammatik konstruieren.

Aufgabe MONA*: Beweisen Sie obigen Satz.

Grenzen der Berechenbarkeit

Im folgenden wollen wir kurz auf die Möglichkeiten des Berechnungsmodells der Grammatiken eingehen und dessen Grenzen aufzeigen.

Sei Σ ein beliebiges Alphabet.
Σ^* ist abzählbar.
2^{Σ^*} ist überabzählbar.

Es bleibt nur zu zeigen, dass Σ^* abzählbar ist.
Aber Σ^n ist für jedes $n \in \mathbf{N}$ endlich:
$\text{card}(\Sigma^0) = \text{card}(\varepsilon) = 1$; für $n>0$ gilt: $\text{card}(\Sigma^n) = (\text{card}(\Sigma))^n$

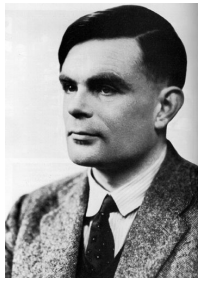
Grenzen der Berechenbarkeit

Die Menge aller formaler Sprachen $L \subseteq \Sigma^*$ ist also überabzählbar, doch nur eine abzählbare Menge davon ist von einer Grammatik erzeugbar.

Die Menge aller formalen Sprachen $L \subseteq \Sigma^*$, die von einer Grammatik erzeugt werden können, ist abzählbar.

Gibt es aber vielleicht andere Modelle von Generierungs- oder Analysemechanismen, durch die mehr formale Sprachen als durch Typ-0-Grammatiken beschrieben werden können?

Grenzen der Berechenbarkeit



1936 erfanden gleichzeitig Alan Turing die Turingmaschinen und Alonzo Church den λ-Kalkül, um den Begriff des Algorithmus bzw. der berechenbaren Funktionen zu formalisieren, und beide Modelle erwiesen sich als gleichwertig. Auch alle anderen seither entwickelten Modelle zur Formalisierung des Begriffs Algorithmus erwiesen sich als nicht mächtiger als Typ-0-Grammatiken bzw. Turingmaschinen. Die folgende These wird daher allgemein akzeptiert:

These von Turing

Gibt es ein endlich beschreibbares Verfahren zur exakten Spezifizierung einer formalen Sprache L, so gibt es eine Typ-0-Grammatik, die L erzeugt bzw. eine Turingmaschine, die L akzeptiert.

Sprachfamilien

Jeder der vorgestellten Grammatiktypen definiert auch eine *Familie formaler Sprachen*:

Sei $i \in \{0,1,2,3\}$ und Σ ein Alphabet.
Dann wird die Menge aller formaler Sprachen $L \subseteq \Sigma^*$, die von einer Grammatik vom Typ i erzeugt werden können, mit $\boldsymbol{L}_i(\Sigma)$ bezeichnet. Die Familie der formalen Sprachen, die von einer Typ-i-Grammatik erzeugt werden können, bezeichnen wir mit $\boldsymbol{L}_i$.

Sprachfamilien (rekursive Sprachen)

Sei Σ ein Alphabet. Eine formale Sprache $L \subseteq \Sigma^*$ heißt genau dann rekursiv, wenn sowohl $L \in \boldsymbol{L_0}$ als auch $\Sigma^*\text{-}L \in \boldsymbol{L_0}$ gilt. Die Menge aller rekursiven Sprachen über Σ wird mit $\boldsymbol{L_{rek}}(\Sigma)$, die Familie aller rekursiven Sprachen mit $\boldsymbol{L_{rek}}$ bezeichnet.

Eine formale Sprache L ist also genau dann rekursiv, wenn sowohl die Sprache L selbst als auch ihr Komplement Σ^*-L rekursiv aufzählbar sind. Damit bildet $\boldsymbol{L_{rek}}$ aber auch die größte Sprachfamilie, für die das Problem $w \in L$ für alle $w \in \Sigma^*$ entscheidbar ist.

Das Problem $w \in L$ ist für rekursive Sprachen entscheidbar.

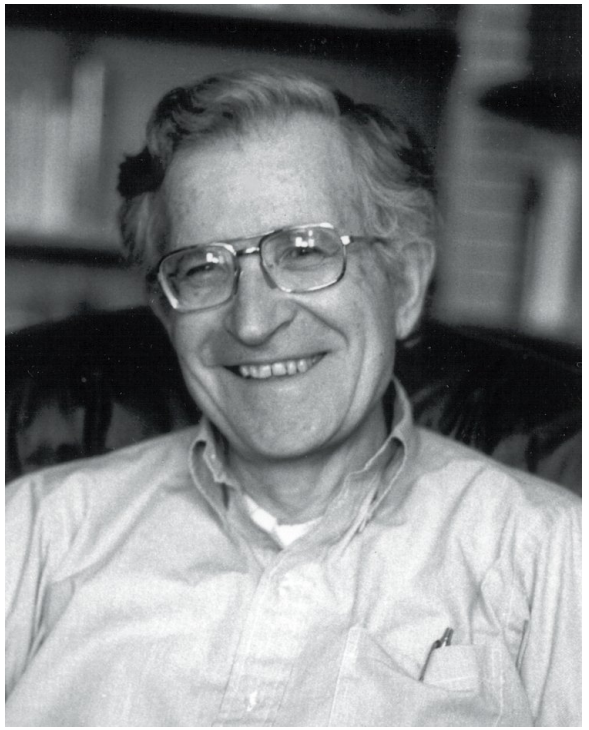
Entscheidbarkeit des Wortproblems für rekursive Sprachen

Das Problem $w \in L$ ist für rekursive Sprachen entscheidbar.

Betrachte Grammatik G mit L(G) = L und Grammatik G‘ mit $L(G') = \Sigma^* - L$. Berechne für n = 1,2,…
SF(G,n) und SF(G‘,n). Nach Definition von G und G‘ muss es ein n so geben, dass
$w \in SF(G,n)$ (d.h., $w \in L$) oder
$w \in SF(G',n)$ (d.h., $w \in \Sigma^* - L$).

Die Chomsky Hierarchie

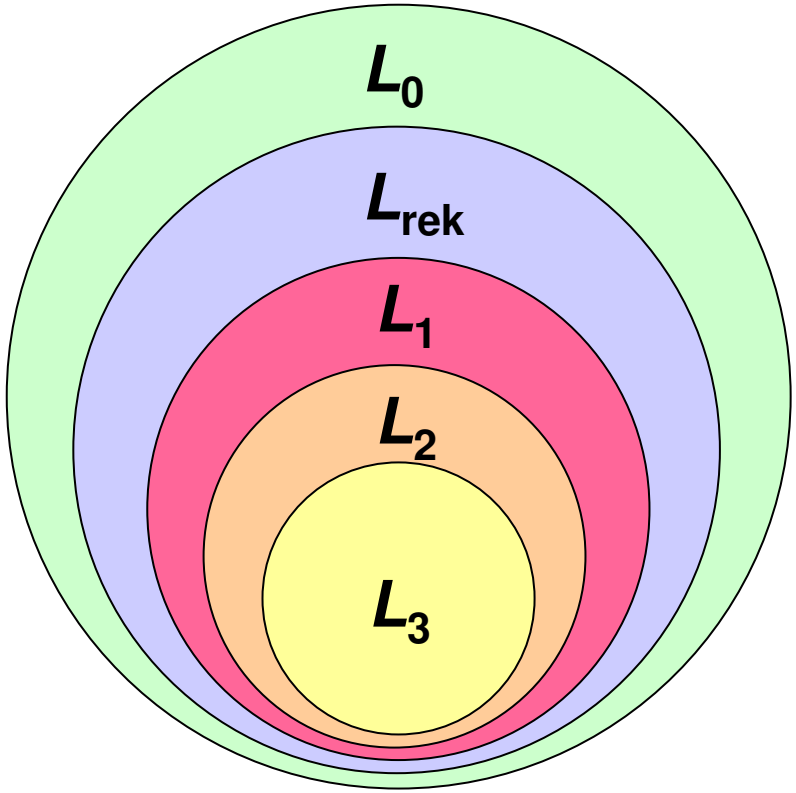
NOAM CHOMSKY (*1928)



1959 : On certain formal properties of grammars.
Information and Control 2 (1959), 137-167

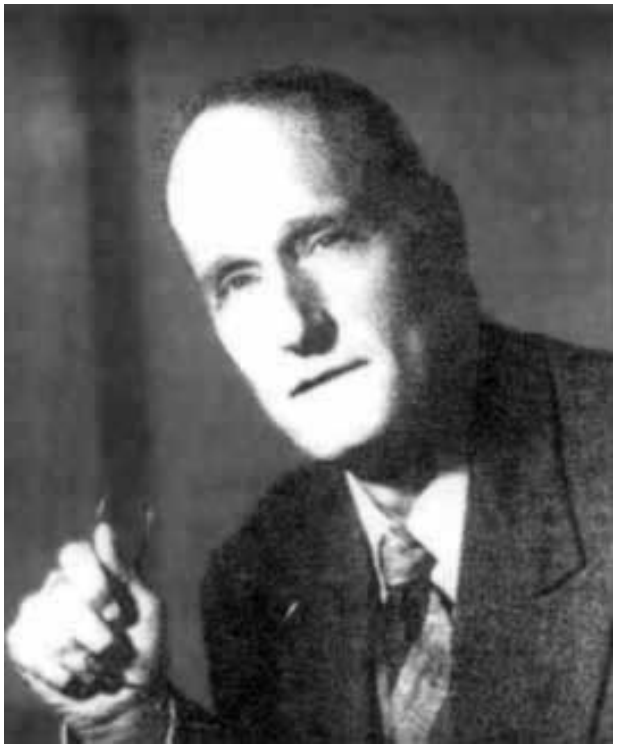
Die Chomsky Hierarchie

$$L_3 \subset L_2 \subset L_1 \subset L_{rek} \subset L_0$$



Endliche Automaten - Kleene

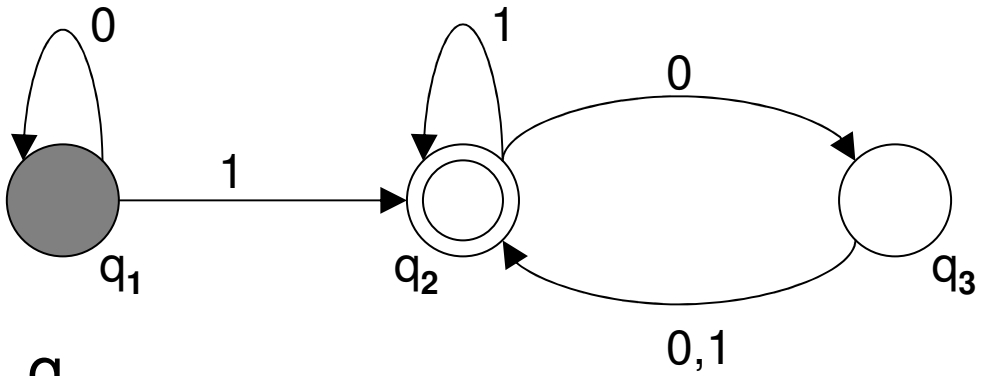
STEPHEN KLEENE (1909 - 1994)



1956: Representation of events in nerve nets and finite automata.
In: C.E. Shannon und J. McCarthy (eds.), Automata studies,
Princeton Univ. Press, 3-42

Endliche Automaten

Zustandsdiagramm:

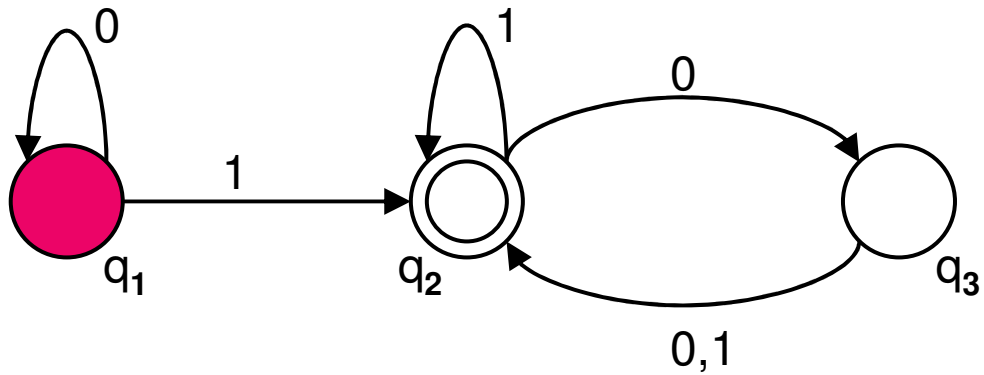


Drei Zustände: q_1 , q_2 , q_3
Startzustand ●: q_1
Endzustand ◎: q_2
Transitionen (Übergänge): Pfeile

Eingabe: Wort
Ausgehend vom Startzustand liest der Automat M von links nach rechts Symbol für Symbol. Nach dem Lesen eines Symbols geht M in den nächsten Zustand über, indem er entlang der mit diesem Symbol markierten Kante geht. Nach dem Lesen des letzten Symbols wird der „Output“ erzeugt: Befindet sich M in einem Endzustand, wird das Wort akzeptiert; ansonsten wird das Wort nicht akzeptiert.

Endliche Automaten: Beispiel

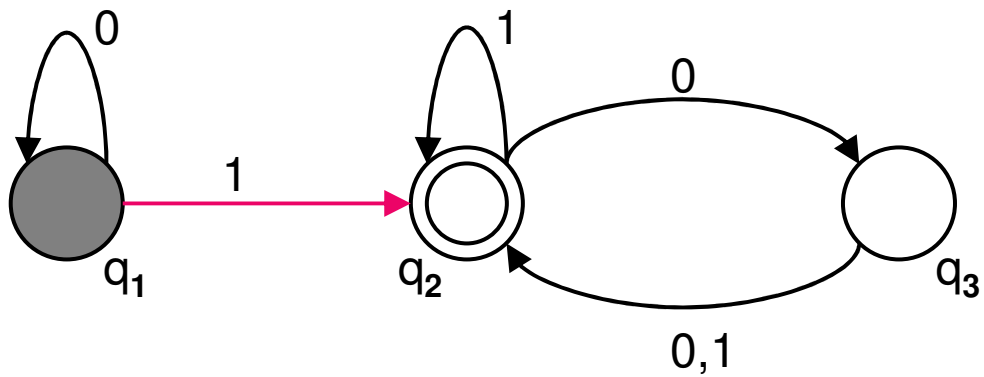
Eingabewort: 1101



Start in Zustand q_1

Endliche Automaten: Beispiel

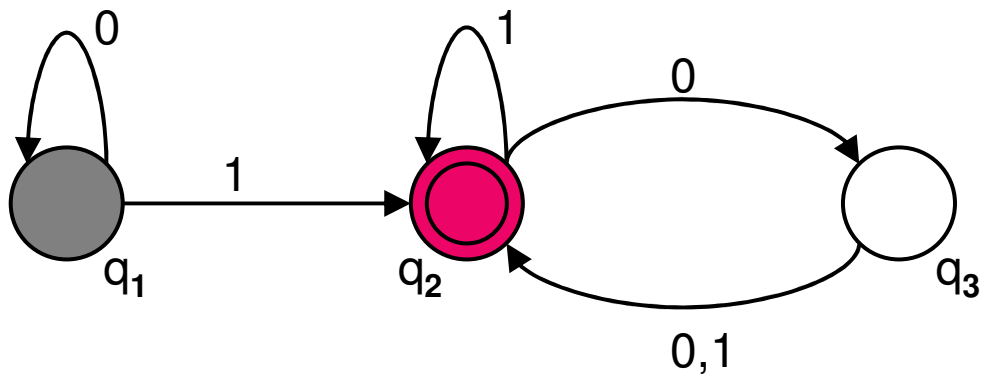
Wort: **1**101



Lies 1 und folge der mit 1 markierten Kante

Endliche Automaten: Beispiel

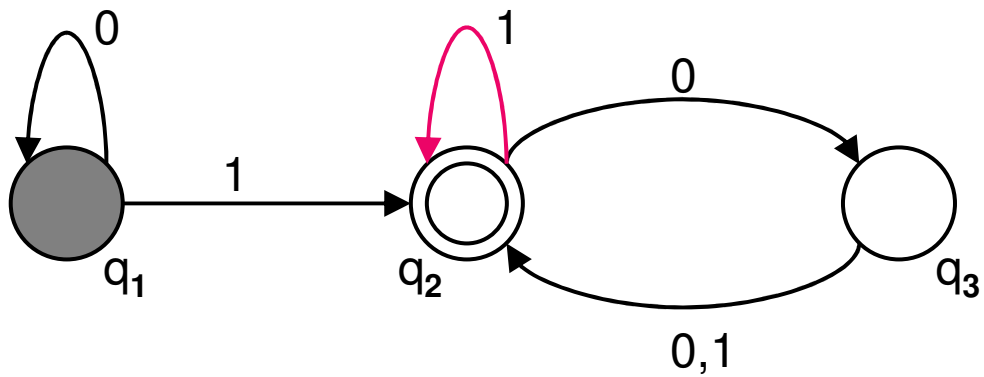
Wort: 1101



Lies 1 und folge der mit 1 markierten Kante zum Zustand q_2

Endliche Automaten: Beispiel

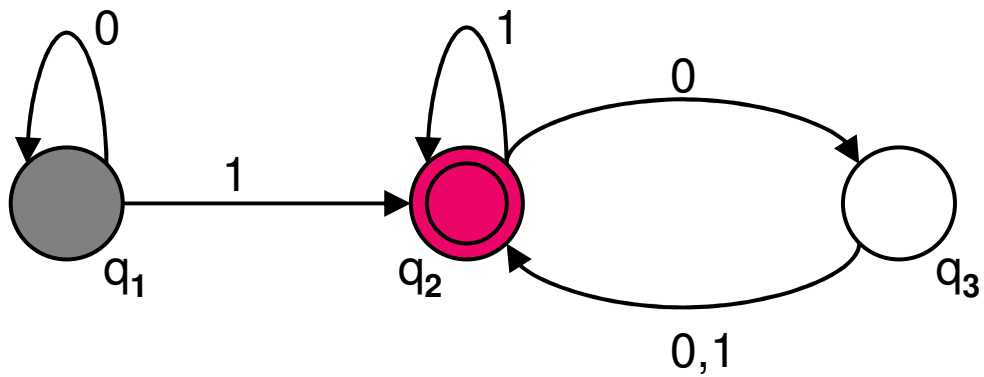
Wort: 1101



Lies 1 und folge der mit 1 markierten Kante

Endliche Automaten: Beispiel

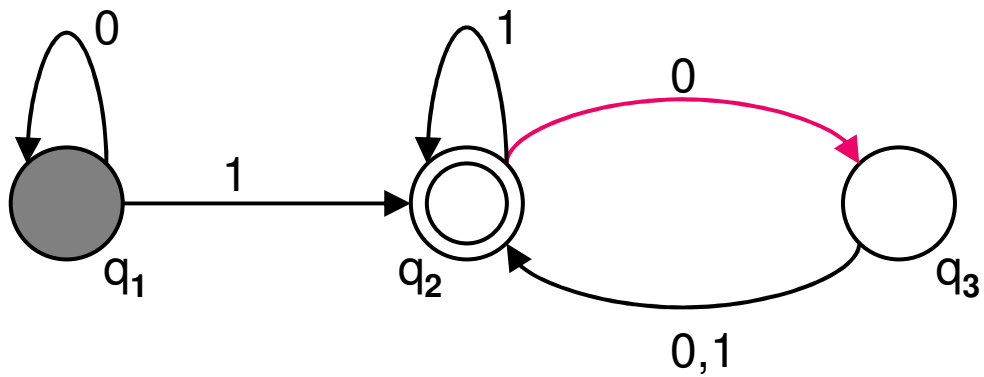
Wort: 1101



Lies 1 und folge der mit 1 markierten Kante zum Zustand q_2

Endliche Automaten: Beispiel

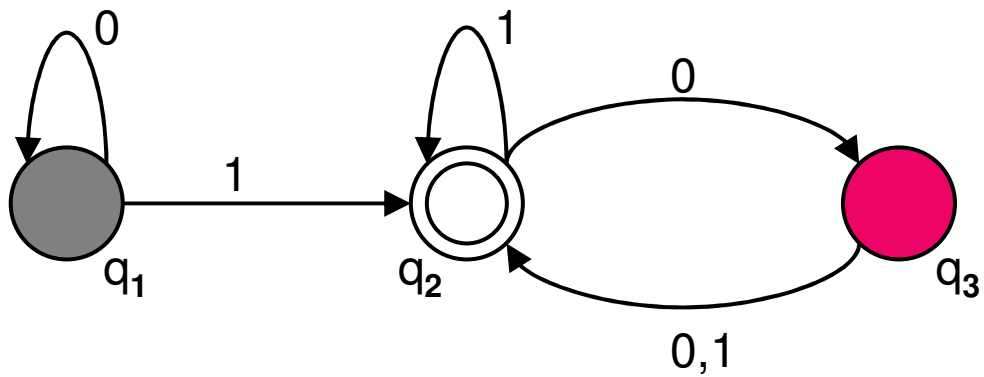
Wort: 1101



Lies 0 und folge der mit 0 markierten Kante

Endliche Automaten: Beispiel

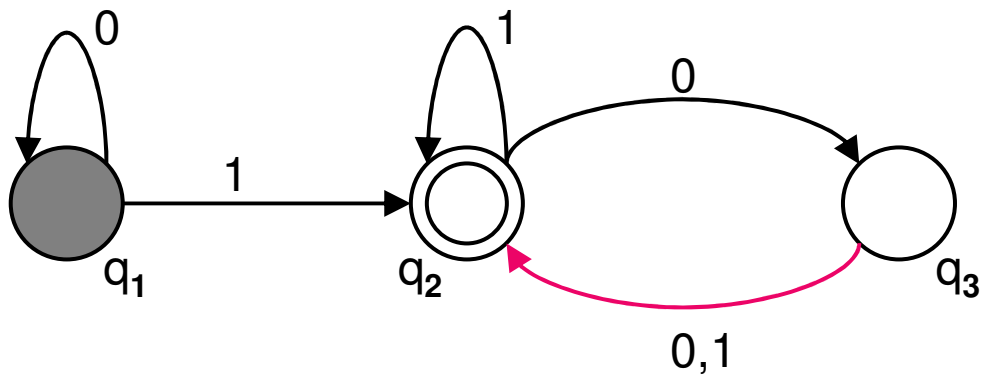
Wort: 1101



Lies 0 und folge der mit 0 markierten Kante zum Zustand q_3

Endliche Automaten: Beispiel

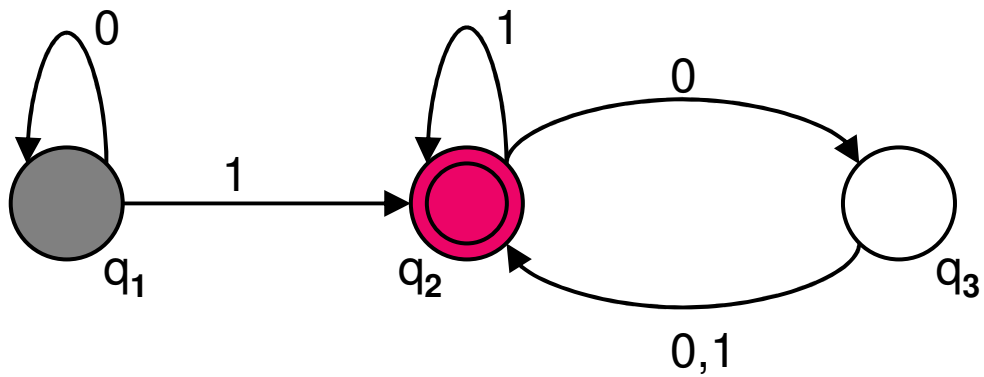
Wort: 1101



Lies 1 und folge der mit 1 markierten Kante

Endliche Automaten: Beispiel

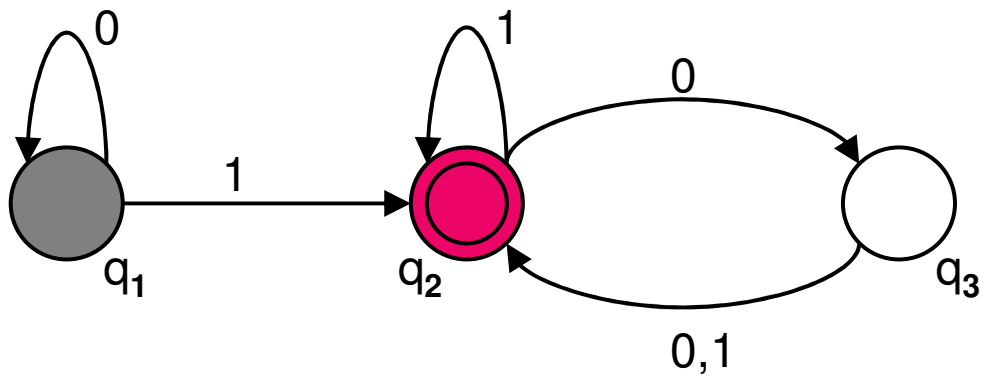
Wort: 1101



Lies 1 und folge der mit 1 markierten Kante zum Zustand q_2

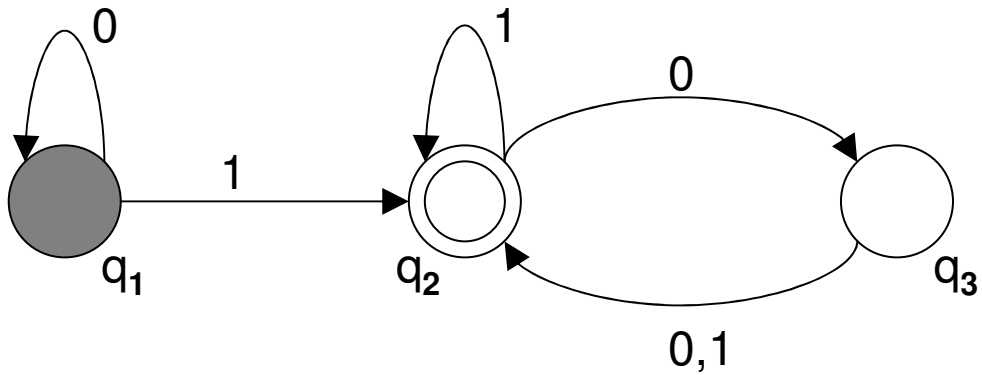
Endliche Automaten: Beispiel

Wort: 1101



Das Wort 1101 wird akzeptiert, da sich der Automat am Ende des Eingabewortes in einem Endzustand befindet.

Endliche Automaten: Beispiel



Akzeptierte Wörter:
1, 01, 11, 01010101, ...

Wörter, die mit 1 enden

Aber auch:
100, 0100, 110000, ...

Wörter, die mit einer geraden Anzahl von 0en nach der letzten 1 enden.

Nicht akzeptierte Wörter:
0, 10, 101000, ...

Akzeptierte Sprache: {0}*{1}({1}*{00,01})*{1}*

Endliche Automaten: formale Definition

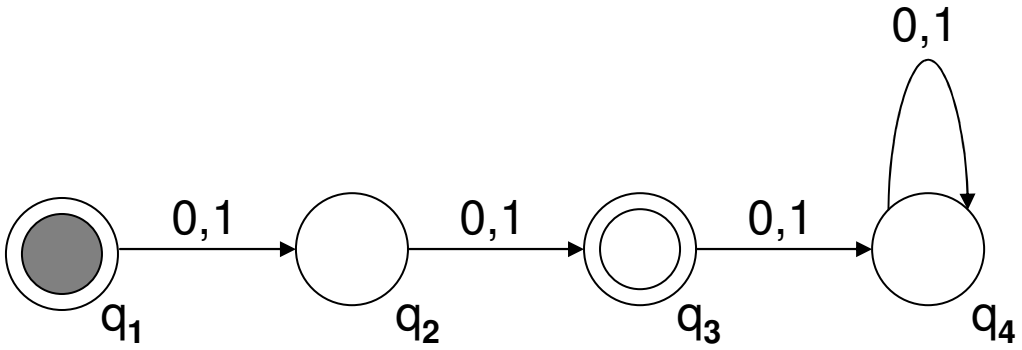
Ein deterministischer endlicher Automat (DEA) ist ein 5-Tupel

$$(Q, \Sigma, \delta, q_0, F)$$

wobei

- Q eine endliche Menge von Zuständen,
- Σ das Alphabet,
- $\delta: Q \times \Sigma \rightarrow Q$ die Transitionsfunktion,
- q_0 der Startzustand und
- $F \subseteq Q$ eine Menge von Endzuständen ist.

Endliche Automaten: Falle



Akzeptierte Sprache:
{ ε ,00,01,10,11}

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA und $q \in Q - F$ mit $\delta(q,a)=q$ für alle $a \in \Sigma$; dann heißt q Falle.

Um die Übersichtlichkeit zu erhöhen, können Fallen bei der Beschreibung endlicher Automaten weggelassen werden (allerdings nur, wenn ausdrücklich erlaubt).

Endliche Automaten: formale Definition

Um das Verhalten eines DEA auf einer Zeichenkette formal zu beschreiben, erweitern wir die Übergangsfunktion δ auf beliebige Wörter aus Σ^*:

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Dann definieren wir die erweiterte Übergangsfunktion $\delta^*: Q \times \Sigma \rightarrow Q$ folgendermaßen:

$\delta^*(q, \varepsilon) = q$

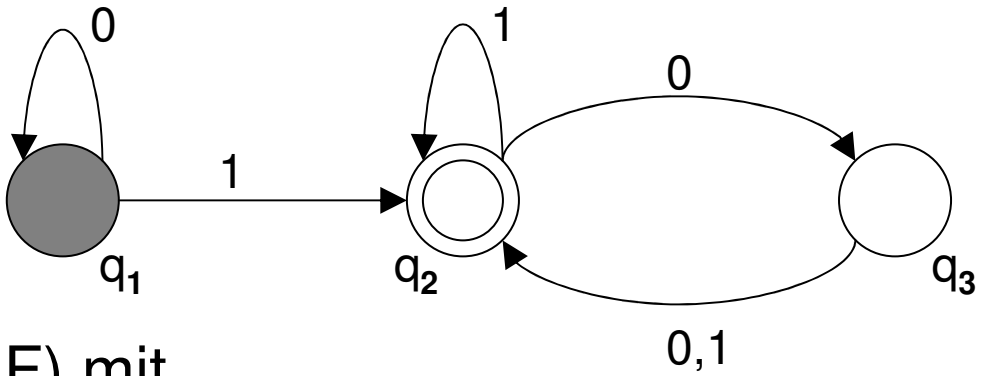
$\delta^*(q,aw) = \delta^*(\delta(q,a),w)$ für alle $q \in Q$, $w \in \Sigma^*$, $a \in \Sigma$

Eine Zeichenkette $w \in \Sigma^*$ heißt vom DEA $M = (Q, \Sigma, \delta, q_0, F)$ akzeptiert, falls $\delta^*(q_0,w) = p$ für einen Zustand $p \in F$ gilt.

Die von M akzeptierte Sprache, bezeichnet mit L(M), ist die Menge $\{ w \in \Sigma^* \mid \delta^*(q_0,w) \in F \}$.

Endliche Automaten: Beispiel

Beispiel:



M = (Q, Σ, δ, q_1 , F) mit
Q = { q_1, q_2 , q_3 }
Σ = { 0,1 }
δ ist gegeben durch die Übergangsmatrix
q_1 Startzustand
F = {q_2}

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

$\delta^*(q_1,1101) = \delta^*(\delta(q_1,1),101) = \delta^*(q_2,101) = \delta^*(\delta(q_2,1),01)$
$\delta^*(q_2,01) = \delta^*(\delta(q_2,0),1) = \delta^*(q_3,1) = \delta^*(q_2,\varepsilon) = q_2$

L(M) = {0}*{1}({1}*{00,01})*{1}*

Minimalautomat

Graphen 

Zu jeder regulären Sprache L kann man effektiv einen DEA M mit einer minimalen Anzahl von Zuständen konstruieren, der bis auf die Umbenennung der Zustände eindeutig ist.

Nondeterminismus

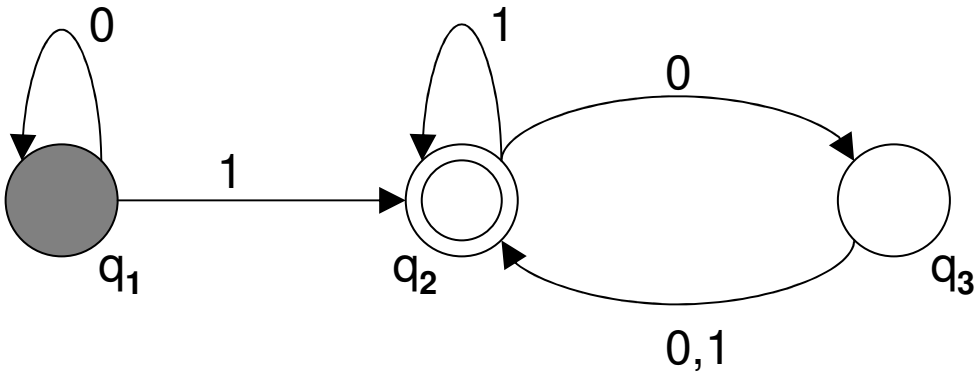
MICHAEL O. RABIN (*1931)

DANA SCOTT (*1932)

1959: *Finite Automata and Their Decision Problem*
IBM J. Research and Development, 3:114-125

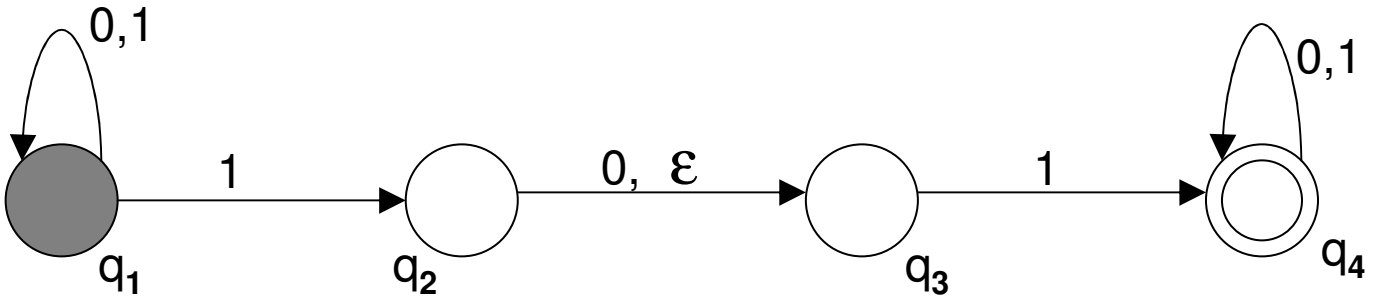
1976: Turing-Preis für Informatik

Nondeterminismus



DEA

Von einem Zustand aus gibt es mit ein und demselben Eingabesymbol genau einen Folgezustand.



NEA

Übergänge sind auch mit ε möglich (ε-Übergänge)
Von einem Zustand aus kann es mit ein und demselben Eingabesymbol mehrere Folgezustände geben.

NEA

Ein nichtdeterministischer endlicher Automat (NEA)
ist ein 5-Tupel $(Q, \Sigma, \delta, q_0, F)$

wobei

- Q eine endliche Menge von Zuständen,
- Σ das Alphabet,
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$ die Transitionsfunktion
- q_0 der Startzustand und
- $F \subseteq Q$ eine Menge von Endzuständen ist.

Äquivalenz von NEA und DEA

Zu jedem nicht-deterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

NEA: $M = (Q, \Sigma, \delta, q_0, F)$

DEA: $M = (Q', \Sigma, \delta', q'_0, F')$ wobei

$$Q' = 2^Q$$

$$\delta'(q',a) = \bigcup_{q \in q'} \delta^*(q,a) \qquad \text{für alle } q' \in Q', a$$

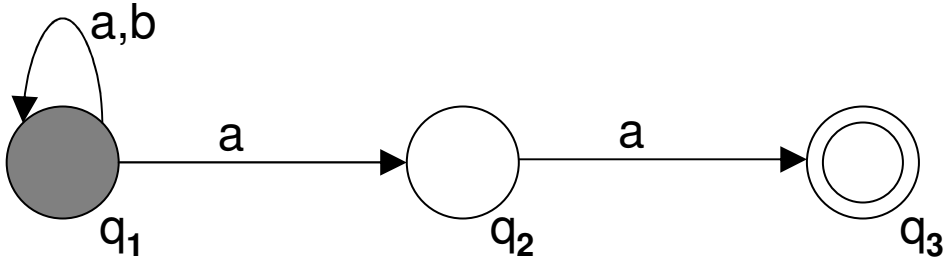
$$q'_0 = \{ q_0 \}$$

$$F' = \begin{cases} \{q' \in Q' \mid q' \cap F \neq 0 \} \cup \{q'_0\} & \text{falls } \varepsilon \in L(A) \\ \{q' \in Q' \mid q' \cap F \neq 0 \} & \text{sonst} \end{cases}$$

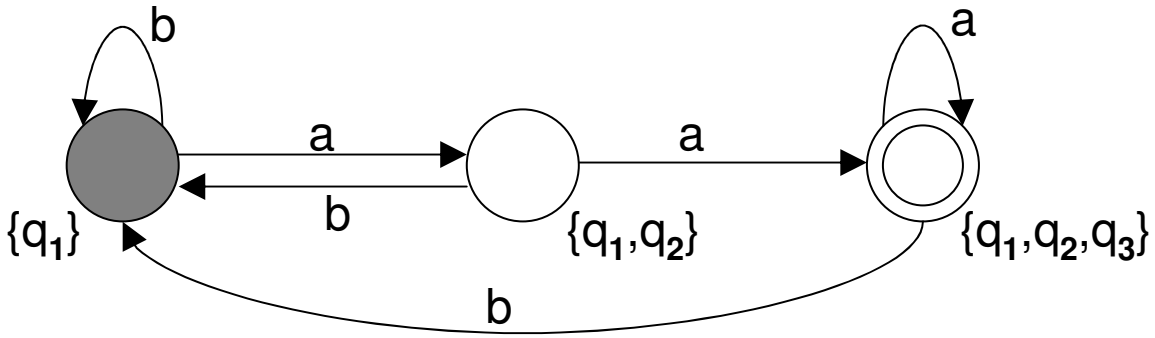
□

von NEA zu DEA: Beispiel

$L(M) = \{ waa \mid w \in \{a,b\}^* \}$

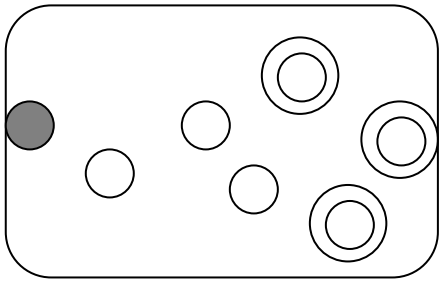


	δ'	a	b
SZ	$\{q_1\}$	$\{q_1,q_2\}$	$\{q_1\}$
	$\{q_1,q_2\}$	$\{q_1,q_2,q_3\}$	$\{q_1\}$
EZ	$\{q_1,q_2,q_3\}$	$\{q_1,q_2,q_3\}$	$\{q_1\}$

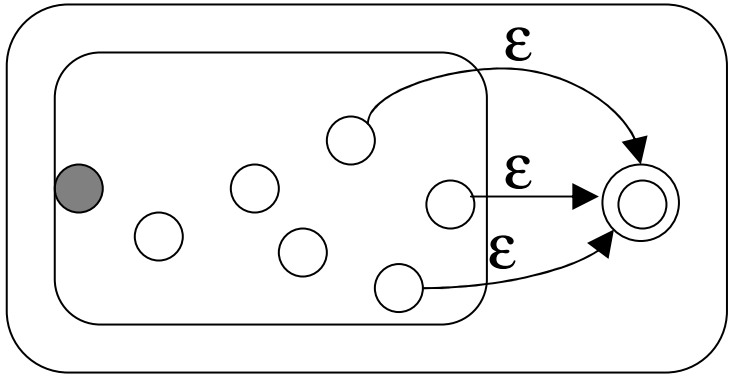


NEA mit einem Endzustand

Zu jedem NEA M gibt es einen äquivalenten NEA M‘ mit card(F) = 1.



M



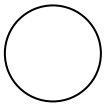
M‘

□

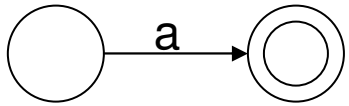
EA und reguläre Sprachen

Zu jeder regulären Sprache R gibt einen endlichen Automaten M sodass R = L(M)

L = {}

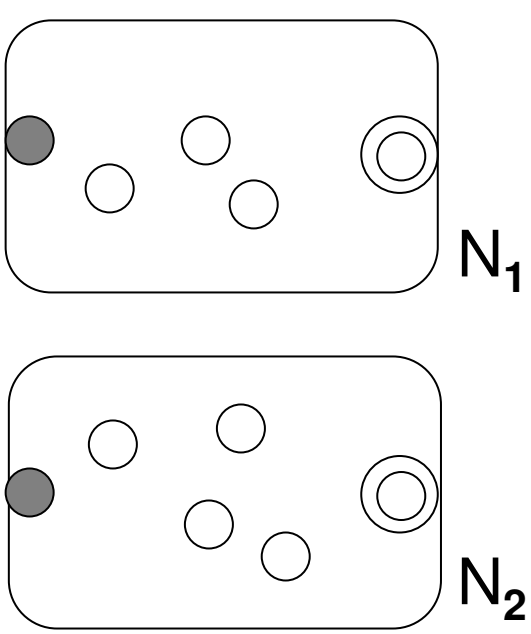


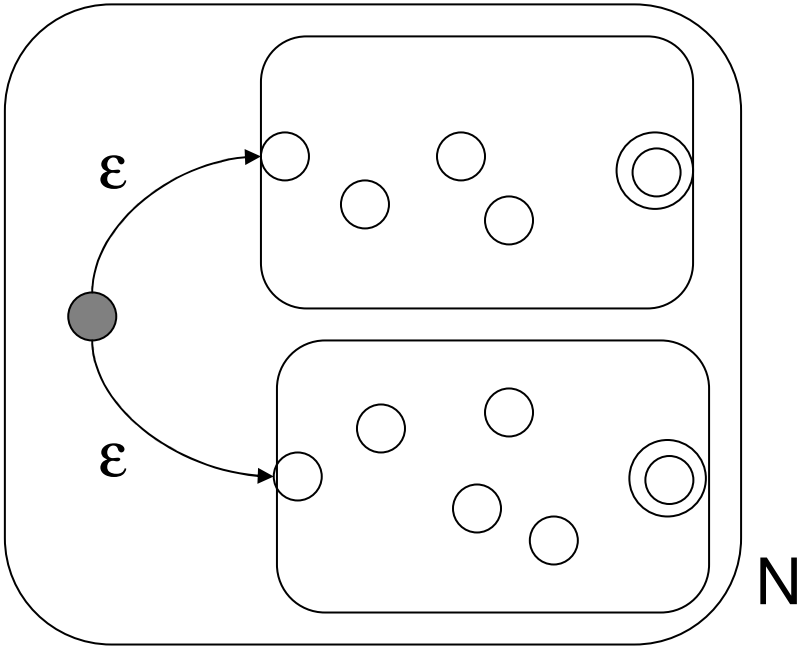
L = {a} für a $\in$ Σ



EA und reguläre Sprachen

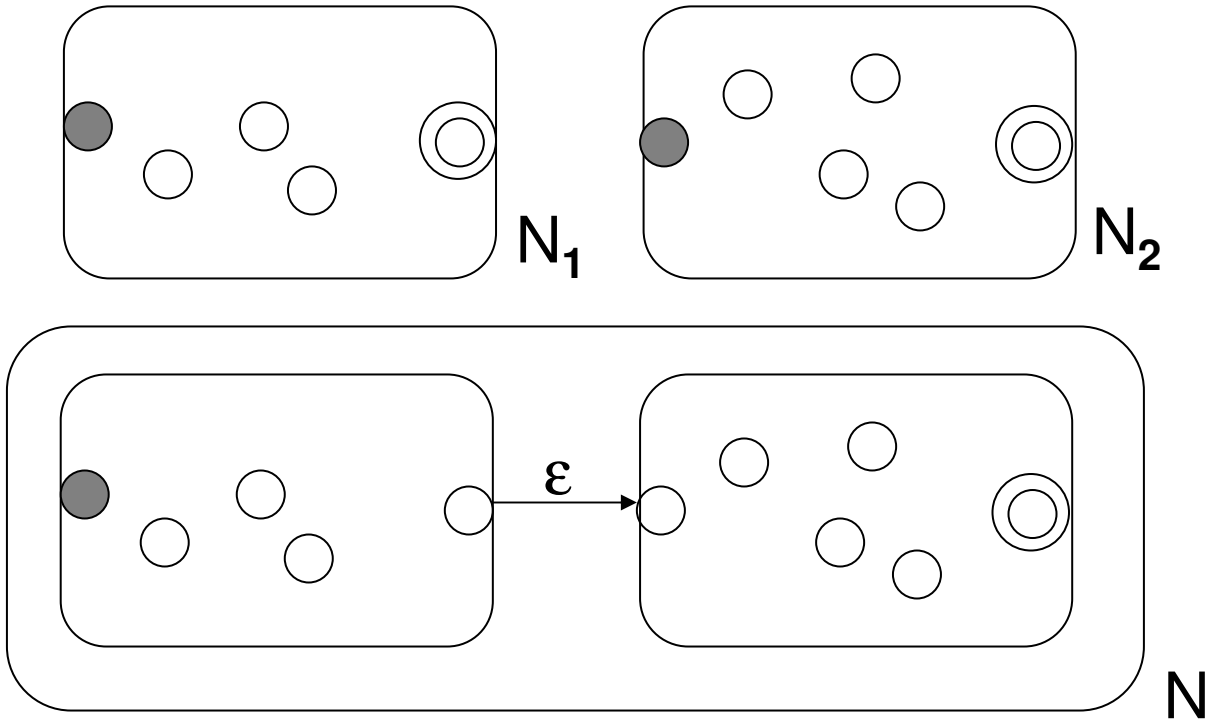
$L = L_1 \cup L_2$





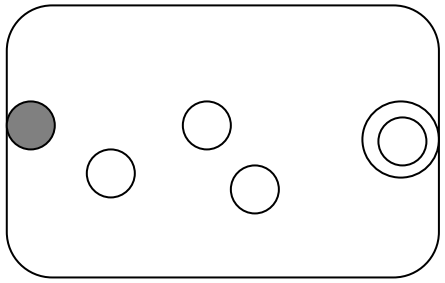
EA und reguläre Sprachen

$L = L_1 \cdot L_2$

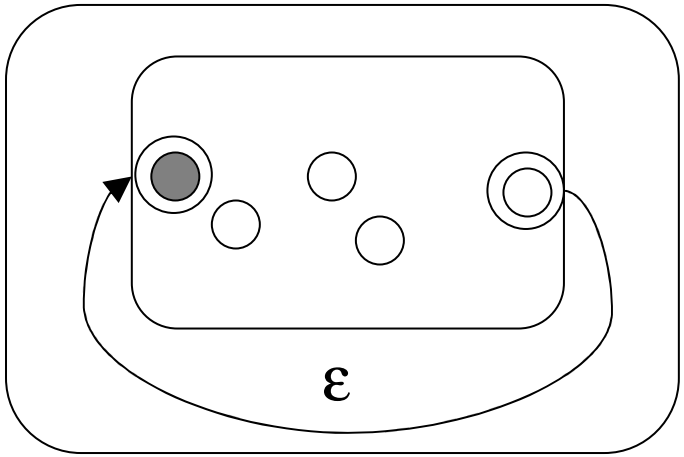


EA und reguläre Sprachen

$L = (L_1)^*$



N_1



N

□

Von DEA zu REG

Wird eine Sprache von einem DEA akzeptiert, dann ist sie regulär.

$M = (\{ q_i \mid 1 \leq i \leq n\}, \Sigma, \delta, q_0, F)$

R^k_{ij} Menge aller Wörter, mit denen man von q_i nach q_j gelangt, ohne einen Zwischenzustand mit Index größer als k zu durchlaufen.

$$R^0_{ij} = \begin{cases} \{a \in \Sigma \mid \delta(q_i,a) = q_j\} & \text{für } i \neq j \\ \{a \in \Sigma \mid \delta(q_i,a) = q_j\} \cup \{\varepsilon\} & \text{für } i = j \end{cases}$$

$$R^k_{ij} = R^{k-1}_{ij} \cup R^{k-1}_{ik} \cdot (R^{k-1}_{kk})^* \cdot R^{k-1}_{kj} \quad \text{für } k > 0$$

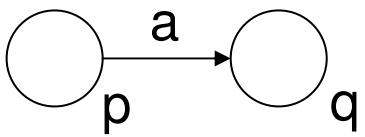
$$L(M) = \bigcup_{q_j \in F} R^n_{ij}$$

□

Von DEA zu REG Grammatik

Zu jedem DEA M gibt es eine reguläre Grammatik G sodass L(M) = L(G).

$M = (Q, \Sigma, \delta, q_0, F)$ $G = (Q, \Sigma, P, q_0)$



$p \rightarrow aq$

$p,q \in Q$ und $a \in \Sigma$

$\delta(p,a) = q$

$p \rightarrow \varepsilon$

□

Reguläre Sprachen: Zusammenfassung

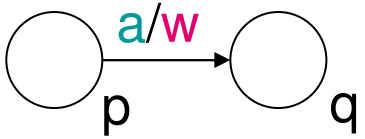
Beschreibungsmethoden für reguläre Sprachen:

Reguläre Mengen
Reguläre Ausdrücke
Reguläre Grammatiken
DEA
NEA

Verallgemeinerte Sequentielle Maschinen

generalized sequential machines

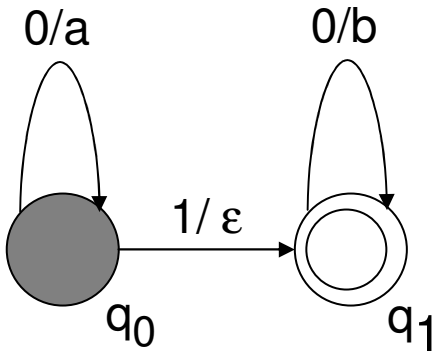
Endliche Automaten mit Ausgabe, d.h. mit jedem Eingelesenen Symbol wird ein Wort ausgegeben.
Die bei der erfolgreichen Analyse von Eingabewörtern aus einer Sprache L erzeugten Ausgabewörter bilden die durch die gsm-Abbildung erzeugte Sprache.



Beim Übergang vom Zustand p in den Zustand q wird ein Symbol a gelesen und das Wort w ausgegeben.

gsm - Beispiele

gsm M bildet $\{0^n10^n \mid n \geq 0\}$ auf $\{a^nb^n \mid n \geq 0\}$ ab.



$M = (\{q_0,q_1\},\{0,1\},\{a,b\}, \delta, q_0,\{q_1\})$

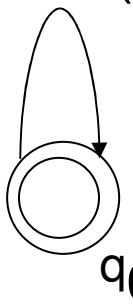
$\delta(q_0,0) = (q_0,a)$

$\delta(q_0,1) = (q_1, \varepsilon)$

$\delta(q_1,0) = (q_1,b)$

Jeder Homomorphismus h ist eine gsm-Abbildung:

a/h(a) für alle a aus T



$M = (\{q_0\},T,T', \delta, q_0,\{q_0\})$

$\delta(q_0,0) = (q_0,h(a))$ für alle a aus T

gsm - Aufgaben

Konstruieren Sie eine gsm M, welche

A) $\{0^{2n}10^{3n} \mid n \geq 1\}$ auf $\{a^n b^n \mid n \geq 1\}$,

B) $\{0^{2n}10^{3n}10^{4n} \mid n \geq 0\}$ auf $\{a^{4n}b^{3n-1}c^{64n+16} \mid n \geq 0\}$,

C) $\{(a^3b^5)^{2n}c^{3n} \mid n \geq 0\}$ auf $\{a^{16n}b^{81n} \mid n \geq 1\}$,

abbildet.

Eigenschaften regulärer Sprachen

Seien Σ und Γ zwei Alphabete.

Ein Homomorphismus ist eine Funktion h: $\Sigma \rightarrow \Gamma^*$

Erweiterung auf Wörter über Σ:

$h(s_1...s_n) = h(s_1)...h(s_n)$ wobei $s_1,...,s_n$ Symbole aus Σ sind.

Homomorphes Bild einer Sprache L:

$h(L) = \{ h(w) \mid wL\}$

$L_{reg}(\Sigma)$ ist abgeschlossen gegenüber

- Vereinigung, Konkatenation, Stern
- Plus-Operator ($A^+ = AA^*$)
- Komplement bzgl. Σ^* (Konstruiere DFA und vertausche End- und Nichtendzustände)
- Durchschnitt ($A \cap B = \overline{\overline{A} \cup \overline{B}}$)
- Differenz ($A - B = A \cap \overline{B}$)
- Homomorphismen

Grenzen der Regularität

Um die Mächtigkeit von endlichen Automaten zu verstehen, muss man auch ihre Grenzen kennen.

Sei z.B. $B = \{0^n 1^n \mid n \geq 0\}$ Gibt es einen DEA für B?

Es sieht so aus, als müsste sich die Maschine die Anzahl der Nullen merken. Nachdem ebendiese Anzahl aber nicht limitiert ist, müsste sich der DEA eine unbeschränkte Anzahl von Möglichkeiten merken können. Mit einer endlichen Anzahl von Zuständen ist das aber nicht möglich!

Reicht das als Beweis? **NEIN!**

Nur weil es so aussieht, als ob eine Sprache unbegrenzten Speicher brauchen würde, ist das nicht notwendigerweise so!

Grenzen der Regularität

In der Tat ist $B = \{ 0^n1^n \mid n \geq 0 \}$ nicht regulär.
ABER: Andere Sprachen scheinen auch eine unbegrenzte Anzahl von Möglichkeiten zu verlangen, und doch sind sie regulär:

Seien C und D Sprachen über $\Sigma = \{0,1\}$

C = { w | w enthält gleich viele Symbole 0 wie Symbole 1 }

D = { w | die Teilworte 01 und 10 kommen gleich oft in w vor}

Auf den ersten Blick scheint eine akzeptierende Maschine in beiden Fällen unbegrenzt „zählen“ zu müssen, daher könnte man annehmen, dass keine der beiden Sprachen regulär sei.

Wie angenommen, ist C tatsächlich nicht regulär,
D überraschenderweise aber sehr wohl!

Schubfachprinzip

LEJEUNE DIRICHLET (1805-1859)

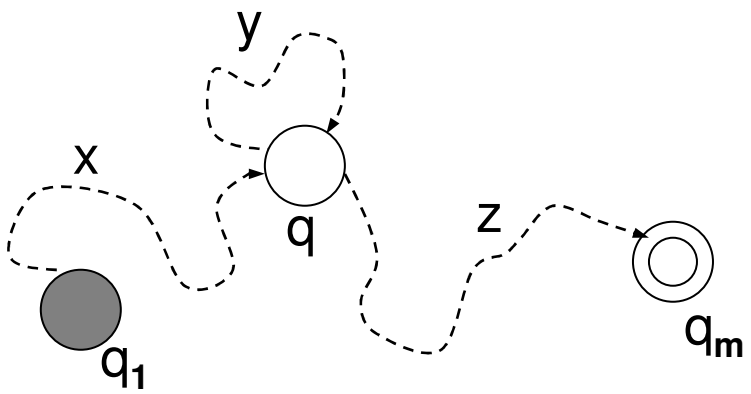
Vorlesungen über Zahlentheorie
(prepared for publication by Dedekind, first edition 1863)

Schubfachprinzip:
Bei Verteilung von n+1 Gegenständen auf n Schubfächer müssen in mindestens einem Schubfach zwei Gegenstände landen. (pigeonhole principle)

Grenzen der Regularität

(Unendliche) Reguläre Sprachen haben eine spezielle Eigenschaft:
Jedes Wort ab einer bestimmten Länge kann „aufgepumpt“ werden, d.h. jedes solche Wort enthält ein Teilstück, das beliebig oft wiederholt werden kann, wobei die resultierenden Wörter noch immer in derselben Sprache liegen.

Deterministischer endlicher Automat habe m Zustände.
Gilt für ein von M akzeptiertes Wort w, dass $|w| \geq m$, so muss mindestens ein Zustand mehr als einmal durchlaufen werden.
(Schubfachprinzip) Zerlege w in xyz:
y gehe von q nach q ; diese Schleife kann ausgelassen bzw. beliebig oft wiederholt werden, d.h. xy^iz wird für alle i ebenfalls von M akzeptiert !



Pumping Lemma für reguläre Sprachen 1

Sei L eine unendliche reguläre Sprache. Dann gibt es eine (nur von L abhängige) Schranke $m > 0$ so, dass für jedes Wort w in L mit $|w| \geq m$ Wörter x,y,z so existieren, dass

$w = xyz$

$|y| > 0$ und

$|xy| \leq m$

sowie

$w(i) = xy^iz$ für alle $i \geq 0$ ebenfalls in L liegt.

Pumping Lemma für reguläre Sprachen

$\forall\ L \in \boldsymbol{L_3}$	für jede reguläre Sprache L

$\exists\ m \in \mathbf{N}$	gibt es eine natürliche Zahl m
$\forall\ w \in L\ (\ \lvert w\rvert \geq m\ )$	für jedes Wort w aus L (mind. m lang)
$\exists\ xyz = w$	gibt es Teilwörter x,y,z von w so, dass
$\lvert y\rvert > 0$	das Teilwort y nicht leer ist,
$\lvert xy\rvert \leq m$	die ersten zwei Teilwörter sind nicht länger als m sowie
$\forall\ i \in \mathbf{N}$	für jede natürliche Zahl i
$xy^iz \in L$	ist die i-fach gepumpte Version in L.

Pumping Lemma für reguläre Sprachen

$\exists \quad L \notin \boldsymbol{L_3}$	es gibt nicht-reguläre Sprachen L

$\neg$

$\exists \quad m \in \mathbf{N}$	gibt es eine natürliche Zahl m
$\forall\, w \in L \;(\; \lvert w \rvert \geq m \;)$	für jedes Wort w aus L (mind. m lang)
$\exists \quad xyz = w$	gibt es Teilwörter x,y,z von w so, dass
$\lvert y \rvert > 0$	das Teilwort y nicht leer ist,
$\lvert xy \rvert \leq m$	die ersten zwei Teilwörter sind nicht länger als m sowie
$\forall\, i \in \mathbf{N}$	für jede natürliche Zahl i
$xy^i z \in L$	ist die i-fach gepumpte Version in L.

Pumping Lemma für reguläre Sprachen

$\exists \quad L \notin \boldsymbol{L_3}$	es gibt nicht-reguläre Sprachen L

$\forall \; m \in \mathbf{N}$	für alle natürlichen Zahl m
$\exists \; w \in L \; (\lvert w \rvert \geq m)$	existiert ein Wort w aus L (mind. m lang)
$\forall$ x,y,z mit xyz = w	für alle Zerlegungen von w in xyz, wo
$\lvert y \rvert > 0$	das Teilwort y nicht leer ist und
$\lvert xy \rvert \leq m$	die ersten zwei Teilwörter nicht länger als m sind,
$\exists \; i \in \mathbf{N}$	existiert eine natürliche Zahl i so, dass
$xy^i z \notin L$	die i-fach gepumpte Version nicht in L.

Pumping Lemma - Beispiel

$L = \{ 0^n 1^n \mid n \geq 0 \}$

Beweis durch Widerspruch:

Angenommen L sei regulär.

Für beliebiges m wähle $w = 0^m 1^m$

(offensichtlich gilt $w \in L$ und $|w| \geq m$).

Betrachte beliebige Zerlegungen von w in xyz mit

$|xy| \leq m$ und $|y| > 0$: xy kann nur aus Nullen bestehen.

Wähle ein i so, dass xy^iz nicht von der Gestalt $0^n 1^n$ ist:

Für alle i gilt, dass $xy^iz = 0^{m+(i-1)\,|y|}\, 1^m$ ist, d.h. für

$i = 0$ und alle $i > 1$ ist $m+(i-1)\,|xy| \neq m$ und

xy^iz daher nicht von der Gestalt $0^n 1^n$! □

Pumping Lemma für reguläre Sprachen 2

Sei L eine unendliche reguläre Sprache. Dann gibt es eine (nur von L abhängige) Schranke $m > 0$ so, dass für jedes Wort w in L mit $|w| \geq m$ und jedes Teilwort v von w mit $|v| \geq m$ und $w = svt$ für Wörter $s,t \in \Sigma^*$ gilt:

$v = xyz$

$|xy| \leq m$ und

$|y| > 0$

sowie

$w(i) = sxy^izt$ für alle $i \geq 0$ ebenfalls in L liegt.

Diese zweite Variante des Pumping Lemmas ist eine Verallgemeinerung der ersten Variante, bei der ein beliebiges Teilwort ausreichender Länge ausgewählt werden kann, von dem nun wieder ein Teilwort beschränkter Länge beliebig aufgepumpt werden kann.

Pumping Lemma für reguläre Sprachen - Beispiele

Zeigen Sie mit einer der beiden Variante des Pumping Lemmas für reguläre Sprachen, dass die folgenden formalen Sprachen nicht regulär sind.
In welchem Fall wird die allgemeinere Variante benötigt?

Aufgabe PLR A:

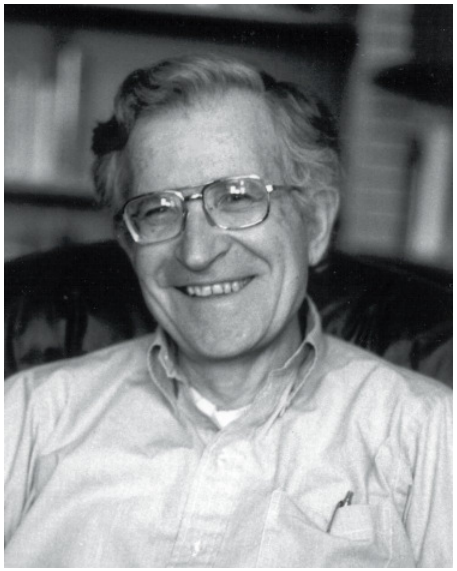
$L = \{ 0^n 1^m \mid m \geq n \}$

Aufgabe PLR B:

$L = \{ 0^n 1^m \mid n \geq m \}$

Noam CHOMSKY, Sheila GREIBACH

Noam CHOMSKY (*1928)



Sheila GREIBACH (*1939)

Normalformen für kontextfreie Grammatiken

Grammatik $G = (N,T,P,S)$

GREIBACH Normalform:

$A \rightarrow aw, \quad w \in N^*$

Erweiterte GREIBACH Normalform:

$A \rightarrow aw, \quad w \in (N \cup T)^*$

CHOMSKY Normalform:

$A \rightarrow BC,\ A \rightarrow a, \quad A,B,C \in N,\ a \in T$

$S \rightarrow \varepsilon$ ist nur dann erlaubt, wenn S nicht auf der rechten Seite einer Produktion vorkommt.

Homomorphismen auf kontextfreien Sprachen

Beweis:

Sei Grammatik $G = (N,T,P,S)$ eine Typ-2-Grammatik, ohne Beschränkung der Allgemeinheit in CHOMSKY-Normalform, und $h: T^* \rightarrow W^*$ ein Homomorphismus mit $T \cap W = \{\}$.

Konstruiere nun ein kontextfreie Grammatik G‘, $G' = (N \cup T', W, P', S')$ mit

$$P' = P - \{A \rightarrow a \mid A \rightarrow a \in P, A \in N, a \in T\} \cup \{A \rightarrow h(a) \mid A \rightarrow a \in P, A \in N, a \in T\}.$$

Klarerweise gilt auf Grund der Konstruktion $L(G') = h(L(G))$. □

Ableitungen in kontextfreien Grammatiken

„normale“ Ableitung $\Rightarrow$:
ein beliebiges Nonterminal wird ersetzt.

Links-Ableitung $\Rightarrow_L$:

das in der Satzform am weitesten links vorkommende Nonterminal wird ersetzt

Rechts-Ableitung $\Rightarrow_R$:

das in der Satzform am weitesten rechts vorkommende Nonterminal wird ersetzt

Parallel-Ableitung $\Rightarrow_{||}$:

alle in der Satzform vorkommenden Nonterminale werden gleichzeitig ersetzt

Alle Ableitungsvarianten ergeben dieselbe Sprache!

Bäume

Ein (geordneter, markierter gerichteter) Baum über V und W ist ein Graph $g = (K,E,L)$ über V und W, der folgende Bedingungen erfüllt:

1) $W = [1..n]$ für ein $n \in N_1$.
2) Es gibt genau einen ausgezeichneten Knoten p_0 (Wurzel), der keinen Vorgänger hat. Außerdem gibt es von der Wurzel aus zu jedem anderen Knoten q von g einen Pfad $(p_0,e_1,p_1,\ldots,p_{k-1},e_k,q)$ der Länge $k \geq 1$, $(p_i,e_{i+1},p_{i+1}) \in E$, $0 \leq i < k$, $q = p_k$.
3) Jeder von der Wurzel verschiedene Knoten hat genau einen Vorgänger.
4) Jeder Knoten ohne Nachfolger heißt Blatt.
5) Ist p kein Blatt, so sind die Nachfolger von p geordnet (die Kanten tragen die Bezeichnungen 1 bis k für ein k).

Ableitungsbäume für kontextfreie Grammatiken

Sei G = (N,T,P,S) eine kontextfreie Grammatik. Ein Baum g = (K,E,L) über $V = N \cup T \cup \{\varepsilon\}$ und W = [1..n] heißt Ableitungsbaum für G, wenn Folgendes gilt:

1) Ist p_0 die Wurzel von g, so gilt $L(p_0) = S$.
2) Ist p kein Blatt, so muss $L(p) \in N$ gelten.
3) Ist p ein Blatt mit $L(p) = \varepsilon$, so ist p der einzige Nachfolger seines Vorgängers.
4) Ist $<(p,i,q_i)>i \in [1..k]$ die geordnete Menge der von p mit L(p) = A wegführenden Kanten, so ist $A \rightarrow L(q_1)\ldots L(q_k)$ eine Produktion in P.

A-Baum für G: für Wurzel gilt $L(p_0) = A$.

Ableitungsbaum ist ein S-Baum.

Front eines Ableitungsbaumes

Sei G = (N,T,P,S) eine kontextfreie Grammatik und
g = (K,E,L) ein A-Baum für G.

Wir definieren nun eine Ordnungsrelation auf den Pfaden von g: Seien $P(j) = (p(j,0),e(j,1),p(j,1),\ldots,e(j,k_j),p(j,k_j))$ für $j \in \{1,2\}$ zwei voneinander verschiedene Pfade in g, die in der Wurzel beginnen (i.e., $p(1,0) = p(2,0) = p_0$) und zu einem Blatt von g führen, dann definieren wir $P(1) < P(2)$ genau dann, wenn es ein $m \geq 1$ so gibt, dass $e(1,i) = e(2,i)$ für alle $1 \leq i < m$ und $e(1,m) \neq e(2,m)$.

Betrachten wir nun alle derartigen Pfade in g, so sind diese wohlgeordnet und sind p(1),…,p(k) die Blätter dieser Pfade, so ist die Front von g durch L(p(1))…L(p(k)) definiert.

Ableitungen und Ableitungsbäume

Sei G = (N,T,P,S) eine kontextfreie Grammatik und g = (K,E,L) ein A-Baum für G sowie $w \in (N \cup T)^*$.

Dann gilt $A \Rightarrow_G w$ genau dann, wenn es einen A-Baum für G mit Front w gibt.

Jeder Linksableitung in G kann man eindeutig einen Ableitungsbaum zuordnen. Gibt es zwei verschiedene Linksableitungen in G für ein Wort w, so sind die entsprechenden Ableitungsbäume nicht äquivalent.

Eindeutigkeit, (inhärente) Mehrdeutigkeit

Sei G = (N,T,P,S) eine kontextfreie Grammatik.

G heißt eindeutig, wenn es zu jedem in G ableitbaren Terminalwort genau eine Linksableitung in G gibt.
Ansonsten heißt G mehrdeutig.

Eine kontextfreie Sprache heißt inhärent mehrdeutig, wenn jede Grammatik, die L erzeugt, mehrdeutig ist.

Beispiel: Die kontextfreie Sprache $L = L(1) \cup L(2)$ mit
$L(1) = \{a^n b^n c^m \mid n,m \in \mathbf{N}\}$ und
$L(2) = \{a^n b^m c^m \mid n,m \in \mathbf{N}\}$
ist inhärent mehrdeutig.

Bemerkung: $L(1) \cap L(2) = \{a^n b^n c^n \mid 1 \leq n\}$.

Pumping Lemma für kontextfreie Sprachen

Sei L eine unendliche kontextfreie Sprache. Dann gibt es eine (nur von L abhängige) Schranke $m > 0$ so, dass für jedes Wort z in L mit $|z| \geq m$ Wörter $u,x,v,y,w \in \Sigma^*$ so existieren, dass gilt:

$z = uxvyw$,

$|xvy| \leq m$ und

$|xy| > 0$

sowie

$z(i) = ux^ivy^iw$ für alle $i \geq 0$ ebenfalls in L liegt.

Pumping Lemma für kontextfreie Sprachen – Beweis 1

Sei G = (N,T,P,S) eine kontextfreie Grammatik, die L erzeugt, k = card(N) und m := 2^k.

Wegen |z| ≥ m (= 2^k) muss jeder Ableitungsbaum für z einen Pfad mit einer Länge von mindestens k+1 haben. So ein Pfad hat aber mindestens k+2 Knoten, wobei alle bis auf den letzten mit einem Nonterminal markiert sind. Also muss es mindestens ein Nonterminal A aus N geben, das in diesem Pfad mindestens zweimal als Markierung eines Knotens vorkommt (Schubfachprinzip!).

Pumping Lemma für kontextfreie Sprachen – Beweis 2

Sei nun $(p_0,e_1,\ldots,p_l)$ so ein Pfad maximaler Länge in einem Ableitungsbaum von z, d.h., $l \geq k$.

Dann kann man in diesem Pfad zwei Knoten p_{v1} und p_{v2} so auswählen, dass Folgendes gilt:

1) $0 < v2 - v1 \leq k$ und $v1 \geq l-k-1$;
2) $L(p_{v1}) = L(p_{v2}) = A$ für ein $A \in N$ und
   $L(p_j) \neq A$ für alle $j \in \{i \mid v1 < i \leq l\} - \{v2\}$.

Der A-Baum T‘ mit der Wurzel p_{v1} repräsentiert die Ableitung eines Teilwortes z‘ von z mit einer Länge von höchstens 2^k, da es lt. Voraussetzung nur Pfade mit einer maximalen Länge von k+1 geben kann; z‘ ist somit die Front von T‘; bezeichnet man die Front des vom Knoten p_{v2} ausgehenden A-Baumes mit v, so kann man z‘ = xvy für gewisse Wörter x,v,y schreiben.

Pumping Lemma für kontextfreie Sprachen – Beweis 3

Da G (außer eventuell $S \rightarrow \varepsilon$) keine ε-Produktionen enthält und $v2 \neq v1$ gilt, muss $|xy| \geq 1$ gelten, d.h.:

$A \Rightarrow^* xAy$ und $A \Rightarrow^* v$,
wobei $|xvy| \leq 2k = m$ und $|xy| \geq 1$.

$A \Rightarrow^* x^iAy^i \Rightarrow^* x^ivy^i$ für alle $i \geq 0$.

Offensichtlich gibt es nun noch Wörter u und w so, dass man $z = uxvyw$ schreiben kann, d.h., man erhält
$S \Rightarrow^* ux^ivy^iw$ für alle $i \geq 0$. □

Korollare zum Pumping Lemma für kontextfreie Sprachen

Korollar A.

Sei $L = \{a^{f(m)} \mid m \in \mathbf{N}\}$ eine formale Sprache und $f: N \to N$ eine monoton wachsende Funktion über den natürlichen Zahlen derart, dass für jedes $c \in \mathbf{N}$ ein $k \in \mathbf{N}$ mit $f(k+1) > f(k)+c$ existiert. Dann ist L nicht kontextfrei.

Korollar B.

Sei $L = \{a^{f(m)} \mid m \in \mathbf{N}\}$ eine formale Sprache und $f: N \to N$ eine monoton wachsende Funktion über den natürlichen Zahlen derart, dass für ein $d > 0$
$f(k+1) > f(k) + dk$ für alle $k \in \mathbf{N}$ gilt.
Dann ist L nicht kontextfrei.

Beispiele zum Pumping Lemma für kontextfreie Sprachen

Bemerkung: Korollar B folgt direkt aus Korollar A.

Beispiel. $L = \{a^p \mid p \text{ prim}\}$
ist nach Korollar A nicht kontextfrei, da es beliebig große Primzahllücken gibt.

Aufgabe PL2A. $L = \{a^{f(m)} \mid m \in \mathbf{N}\}$ ist nicht kontextfrei für:

1) $f(m) = m^d$, $d \geq 2$,
2) $f(m) = k^m$.

Aufgabe PL2B. L ist nicht kontextfrei für:

1) $L = \{a^n b^n c^n \mid n \geq 1\}$,
2) $L = \{ww \mid w \{0,1\}^*\}$.

Abschlusseigenschaften kontextfreier Sprachen

Die Familie der kontextfreien Sprachen ist weder gegenüber Durchschnitt noch gegenüber Komplement abgeschlossen.

Die Sprachen L(1) und L(2) mit
$L(1) = \{a^n b^n c^m \mid n,m \in \mathbf{N}\}$ und
$L(2) = \{a^n b^m c^m \mid n,m \in \mathbf{N}\}$
sind kontextfreie Sprachen.

$L(1) \cap L(2) = \{a^n b^n c^n \mid 1 \leq n\}$ ist aber nicht kontextfrei.

Außerdem gilt
$L(1) \cap L(2) = \{a,b,c\}^* - ((\{a,b,c\}^* - L(1)) \cup (\{a,b,c\}^* - L(2)))$.
Wäre also $\boldsymbol{L_2}$ gegenüber Komplement abgeschlossen,
dann wäre $\boldsymbol{L_2}$, da gegenüber Vereinigung abgeschlossen,
auch gegenüber Durchschnitt abgeschlossen; Widerspruch!

Kontextfreie Sprachen aus {a}*

Jede kontextfreie Sprache über einem einelementigen Alphabet ist regulär.

Die Korollare A und B zum Pumping Lemma für kontextfreie Sprachen würden somit schon aus dem Pumping Lemma für reguläre Sprachen ableitbar sein.

Charakterisierung regulärer Sprachen aus {a}*:
Für jede reguläre Sprache L aus {a}* gibt es natürliche Zahlen $d,m \geq 0$ sowie $c(k)$, $1 \leq k \leq m$, derart, dass

$$L = \bigcup_{1 \leq k \leq m} \{a^{dx+c(k)} \mid x \in \mathbf{N}\}.$$

Aufgabe: Wie schauen die entsprechenden Minimalautomaten aus?

Kellerautomaten (pushdown automata)

Wie wir gleich sehen werden, sind Kellerautomaten das Gegenstück zu den kontextfreien Grammatiken, d.h., zu einer formalen Sprache L gibt es genau dann eine kontextfreie Grammatik G, die L erzeugt, wenn es einen Kellerautomaten M gibt, der L akzeptiert.

Ein Kellerautomat ist eine Maschine M mit einer endlichen Kontrolle, einem Eingabeband und einem Keller; dies ist ein spezielles Arbeitsband, auf dem M in jedem Schritt nur das oberste Symbol lesen und durch ein beliebiges Wort über dem Kelleralphabet ersetzen kann. Am Beginn des Kellerbandes steht das Kellergrundsymbol Z_0. Dieses kann im letzten Schritt gelöscht werden, d.h., M akzeptiert durch leeren Keller.
Die zweite Akzeptierungsart ist wie bei endlichen Automaten, dass M durch Endzustand akzeptiert.

Kellerautomaten (pushdown automata) – Def.

Ein (nichtdeterministischer) Kellerautomat (KA) ist ein Septupel $M = (Q,T,\Gamma,\delta,q_0,Z_0,F)$, wobei
Q die Menge der Zustände,
T das Eingabealphabet,
Γ das Kelleralphabet,
$\delta: Q \times (T \cup \{\varepsilon\}) \times \Gamma \rightarrow \wp(Q \times \Gamma^*)$ die Überführungsfunktion,
$q_0 \in Q$ der Startzustand,
$Z_0 \in \Gamma$ das Kellergrundsymbol und
$F \subseteq Q$ eine Menge von Endzuständen ist.

$(p,y) \in \delta(q,a,X)$ bedeutet, dass, wenn M im Zustand q auf dem Eingabeband das Terminalsymbol a und auf dem Kellerband das Symbol X liest, dann in den Zustand p übergeht und statt X das Wort y auf das Kellerband schreibt.

Kellerautomaten - Konfigurationen

Eine Konfiguration eines (nichtdeterministischen) Kellerautomaten ist ein Tripel (q,w,y) mit $q \in Q$, $w \in T^*$ und $y \in \{\varepsilon\} \cup \{Z_0\}(\Gamma - \{Z_0\})^*$.

Die Ableitungsrelation $\Rightarrow_M$ auf Konfigurationen von M ist dann wie folgt definiert:

$(q,aw,zX) \Rightarrow_M (p,w,zy)$ genau dann, wenn $(p,y) \in \delta(q,a,X)$,

wobei $p,q \in Q$, $a \in T \cup \{\varepsilon\}$, $w \in T^*$, $X \in \Gamma$, $z,y \in \Gamma^*$.

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

Der Einfachheit halber werden wir M in Zukunft weglassen.

Kellerautomaten – akzeptierte Sprache

Sei $M = (Q,T,\Gamma,\delta,q_0,Z_0,F)$ ein (nichtdeterministischer) KA.

Die von M durch Endzustand akzeptierte Sprache ist

$$L(M) = \{w \in T^* \mid (q_0,w,Z_0) \Rightarrow^* (p,\varepsilon,z) \text{ für ein } p \in F \text{ und ein } z \in \Gamma^*\}.$$

Die von M durch leeren Keller akzeptierte Sprache ist

$$N(M) = \{w \in T^* \mid (q_0,w,Z_0) \Rightarrow^* (p,\varepsilon,\varepsilon) \text{ für ein } p \in Q\}.$$

Die von Kellerautomaten durch Endzustand akzeptierten Sprachen und die durch leeren Keller akzeptierten Sprachen ergeben dieselbe Sprachfamilie, nämlich die der kontextfreien Sprachen.

Kellerautomaten – Beispiel

Sei $M = (\{q_0,q_1,q_f\},\{a,b\},\{Z_0,A\},\delta,q_0,Z_0,\{q_f\})$ ein KA mit

$\delta(q_0,a,Z_0) = (q_0,Z_0A)$,

$\delta(q_0,a,A) = (q_0,AA)$,

$\delta(q_0,b,A) = (q_1,\varepsilon)$,

$\delta(q_1,b,A) = (q_1,\varepsilon)$,

$\delta(q_1,\varepsilon,Z_0) = (q_f,\varepsilon)$.

Die vom Kellerautomaten M durch Endzustand akzeptierte Sprache und die durch leeren Keller akzeptierte Sprache ist dieselbe kontextfreie Sprache:
$L(M) = N(M) = \{a^n b^n \mid n \in N\}$.

Kellerautomaten – Beispiel, Tabelle

Die Übergangsfunktion von M kann in folgender Tabelle anschaulich dargestellt werden:

δ		a	b	ε
q_0	Z_0	$\{(q_0,Z_0A)\}$		
	A	$\{(q_0,AA)\}$	$\{(q_1, \varepsilon)\}$	
q_1	Z_0			$\{(q_f, \varepsilon)\}$
	A		$\{(q_1, \varepsilon)\}$	
q_f	Z_0			
	A			

Kontextfreie Sprachen und Kellerautomaten

Sei L eine kontextfreie Sprache. Dann gibt es einen KA M mit N(M) = L.

Beweis.

Sei G = (N,T,P,S) eine kontextfreie Grammatik in GREIBACH-Normalform mit L(G) = L.

Dann definieren wir den KA M = ({q},T,N,δ,q,S,{q}) mit N(M) = L wie folgt:

$\delta(q,a,A) = \{(q,w^R) \mid A \rightarrow aw \in P\}$

für alle $a \in T$ und alle $A \in N$

sowie noch, falls $\varepsilon \in L$, dann auch $\delta(q,\varepsilon,S) = \{\{(q,\varepsilon)\}$.

M simuliert Linksableitungen in G. □

Kellerautomaten und kontextfreie Sprachen

Sei $M = (Q,T,\Gamma,\delta,q_0,Z_0,F)$ ein (nichtdeterministischer) KA.

Dann ist die von M durch leeren Keller akzeptierte Sprache

$N(M) = \{w \in T^* \mid (q_0,w,Z_0) \Rightarrow^* (p,\varepsilon,\varepsilon) \text{ für ein } p \in Q\}$.

kontextfrei.

Beweis.

Konstruiere kontextfreie Grammatik $G = (N,T,P,S)$ mit
$N = Q \times \Gamma \times Q \cup \{S\}$ und den folgenden Produktionen in P:

1) $S \rightarrow [q_0,Z_0,q]$ für alle $q \in Q$.

2) $[q,A,q_{m+1}] \rightarrow a\ \Pi_{i \in [1..m]}\ [q_i,B_i,q_{i+1}]$

für alle $m \in \mathbf{N}$, $a \in T \cup \{\varepsilon\}$, $q,q_1,\ldots,q_{m+1} \in Q$,
$A,B_1,\ldots,B_m \in \Gamma$, $(q_1, (\Pi_{i \in [1..m]}\ B_i)^R) \in \delta(q,a,A)$. □

Normalformen für kontextfreie Grammatiken

Grammatik $G = (N,T,P,S)$

GREIBACH Normalform:

$A \rightarrow aw, \quad w \in N^*$

Erweiterte GREIBACH Normalform:

$A \rightarrow aw, \quad w \in (N \cup T)^*$

CHOMSKY Normalform:

$A \rightarrow BC,\ A \rightarrow a, \quad A,B,C \in N,\ a \in T$

$S \rightarrow \varepsilon$ ist nur dann erlaubt, wenn S nicht auf der rechten Seite einer Produktion vorkommt.

Normalformen für Grammatiken

Grammatik $G = (N,T,P,S)$

Normalform für monotone Grammatiken:

$A \rightarrow BC,\ AD \rightarrow BC,\ A \rightarrow a, \quad A,B,C,D \in N,\ a \in T$

$S \rightarrow \varepsilon$ ist nur dann erlaubt, wenn S nicht auf der rechten Seite einer Produktion vorkommt.

Normalform für unbeschränkte Grammatiken:

$A \rightarrow BC,\ AD \rightarrow BC,\ A \rightarrow a, \quad A,B,C,D \in N,\ a \in T \cup \{ \varepsilon \}$

Varianten von regulären Grammatiken

Grammatik $G = (N,T,P,S)$

Normalform für reguläre Grammatiken:

$A \rightarrow aB,\ A \rightarrow a,\quad A,B \in N,\ a \in T$

$S \rightarrow \varepsilon$ ist nur dann erlaubt, wenn S nicht auf der rechten Seite einer Produktion vorkommt.

„Maximalvariante“ für reguläre Grammatiken:

$A \rightarrow wB,\ A \rightarrow w,\quad A,B \in N,\ w \in T^*$

CHOMSKY - Hierarchie

Grammatik G = (N,T,P,S); betrachte Normalformen:

Normalform für unbeschränkte Grammatiken:

$A \rightarrow BC,\ AD \rightarrow BC,\ A \rightarrow a,\quad A,B,C,D \in N,\ a \in T \cup \{ \varepsilon \}$

Normalform für monotone Grammatiken:

$A \rightarrow BC,\ AD \rightarrow BC,\ A \rightarrow a,\quad A,B,C,D \in N,\ a \in T$

CHOMSKY Normalform:

$A \rightarrow BC,\ A \rightarrow a,\quad A,B,C \in N,\ a \in T$

Normalform für reguläre Grammatiken:

$A \rightarrow aB,\ A \rightarrow a,\quad A,B \in N,\ a \in T$

CHOMSKY-Hierarchie: $L_3 \subset L_2 \subset L_1 \subset L_0$

$L_3 \subset L_2 \subset L_1 \subset L_{rek} \subset L_0$

Bedeutung der ε - Produktionen

Grammatik G = (N,T,P,S):

Normalform für unbeschränkte Grammatiken:

$A \to BC$, $AD \to BC$, $A \to a$, $A,B,C,D \in N$, $a \in T \cup \{ \varepsilon \}$

Normalform für monotone Grammatiken:

$A \to BC$, $AD \to BC$, $A \to a$, $A,B,C,D \in N$, $a \in T$

Unterschied: ε-Produktionen der Gestalt $A \to \varepsilon$

Eine ε-Produktion der Gestalt $A \to \varepsilon$ reicht bereits:

Zu jeder Typ-0-Sprache $L \subseteq T^*$ gibt es eine monotone Sprache $L' \subseteq (T \cup \{e\})^*$ derart, dass gilt:

- Für jedes Wort $w \in L$ existiert ein Wort $we^n \in L'$ für ein $n \in N$.
- Jedes Wort $v \in L'$ ist von der Gestalt we^n für ein $w \in L$ und ein $n \in N$.

Homomorphismen auf monotonen Sprachen

Beweis:

Sei Grammatik $G = (N,T,P,S)$ eine Typ-0-Grammatik, ohne Beschränkung der Allgemeinheit in Normalform.

Konstruiere dazu ein monotone Grammatik G‘,

$G' = (N \cup \{S',E\}, T \cup \{e\}, P', S')$ mit den folgenden Produktionen in P‘:

$S' \rightarrow Se$, $ED \rightarrow DE$ für alle $D \in N$, $Ee \rightarrow ee$,

$A \rightarrow E$ für alle Produktionen $A \rightarrow \varepsilon$ in P,

$A \rightarrow BC$, $AD \rightarrow BC$, $A \rightarrow a$, $A,B,C,D \in N$, $a \in T$,

für alle derartigen Produktionen in P. □

$h(L(G')) = L$ für den Homomorphismus

$h: (T \cup \{e\})^* \rightarrow T^*$ mit $h(a) = a$, $a \in T$, und $h(e) = \varepsilon$.

Analog dazu gilt $L(G'') = L$ für

$G'' = (N \cup \{S',E,e\}, T, P' \cup \{e \rightarrow \varepsilon\}, S')$.

Abgeschlossenheit gegenüber Homomorphismen

Aufgabe HOMA:
Zeigen Sie, dass alle Sprachfamilien der CHOMSKY-Hierarchie gegenüber ε -freien Homomorpismen abgeschlossen sind.

Aufgabe HOMB:
Zeigen Sie, dass die Sprachfamilien der CHOMSKY-Hierarchie $\boldsymbol{L_3}$, $\boldsymbol{L_2}$ und $\boldsymbol{L_0}$ gegenüber beliebigen Homomorpismen abgeschlossen sind, $\boldsymbol{L_1}$ und $\boldsymbol{L_{rek}}$ hingegen nicht.

Sprachfamilien – (volle) Trios

Sprachfamilie: nichttriviale Menge formaler Sprachen
(enthält zumindest eine nichtleere Sprache)

TRIO:

Abgeschlossen gegenüber $\cap R$, h^{-1}, $h_{-\varepsilon}$
(ε -freier Homom.)

full (volles) TRIO:

Abgeschlossen gegenüber $\cap R$, h^{-1}, h

(volle) Abstrakte Sprachfamilien

(full) abstract family of languages: (f)AFL

AFL:

TRIO und abgeschlossen gegenüber $\cup$, $\bullet$, $^+$

full AFL:

volles TRIO und abgeschlossen gegenüber $\cup$, $\bullet$, *

$\cup$ oder $\bullet$ folgt bereits aus den jeweils 5 anderen Eigenschaften!

Abschlusseigenschaften von Sprachfamilien

	TRIO	fTRIO	AFL	fAFL	$L_3 \subset$	$L_2 \subset$	$L_1 \subset$	L_0
$\cup$			+	+	+	+	+	+
$\bullet$			+	+	+	+	+	+
$+$			+	+	+	+	+	+
$*$				+	+	+	+	+
$\cap R$	+	+	+	+	+	+	+	+
$h_{-\varepsilon}$	+	+	+	+	+	+	+	+
h		+	+	+	+	+	-	+
h^{-1}	+	+	+	+	+	+	+	+
$gsm_{-\varepsilon}$	+	+	+	+	+	+	+	+
gsm		+		+	+	+	-	+
gsm^{-1}	+	+	+	+	+	+	+	+
Kompl.					+	-		-

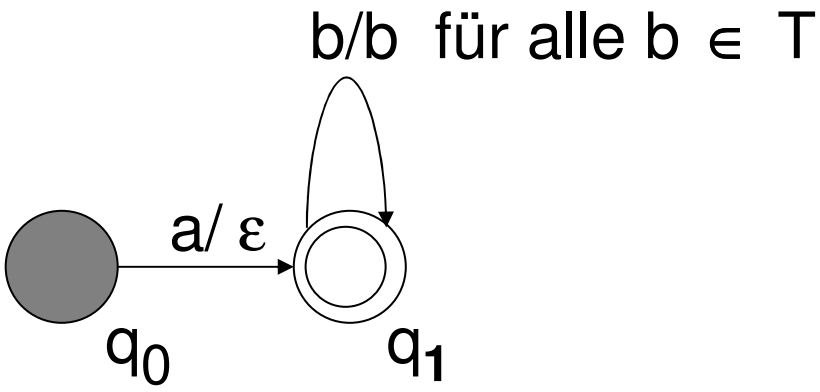
Quotient von Sprachen

Quotient von Sprachen L/M

$L/M = \{ w \mid \text{es gibt ein } u \in M \text{ sodass } wu \in L \}$

$L\backslash M = \{ w \mid \text{es gibt ein } u \in M \text{ sodass } uw \in L \}$

Ist F eine Sprachfamilie, die gegenüber gsm-Abbildungen abgeschlossen ist, so ist für jede Sprache $L \subseteq T^*$ aus F auch $L\backslash\{a\}$ für jedes $a \in T$ aus F.
Beweis:



Weitere Operationen auf Sprachen

$INIT(L) = \{ w \mid wu \in L \}$

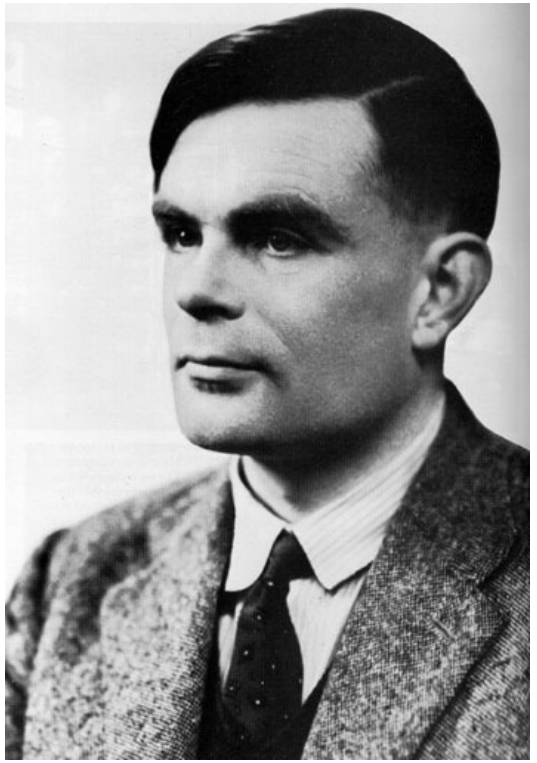
$FIN(L) = \{ w \mid uw \in L \}$

$SUB(L) = \{ w \mid xwu \in L \}$

Aufgabe AFLA: Zeigen Sie, dass jede Sprachfamilie, die gegenüber gsm-Abbildungen abgeschlossen ist, auch gegenüber den Operationen INIT, FIN und SUB abgeschlossen ist.

Turing

Alan M. Turing (1912 - 1954)



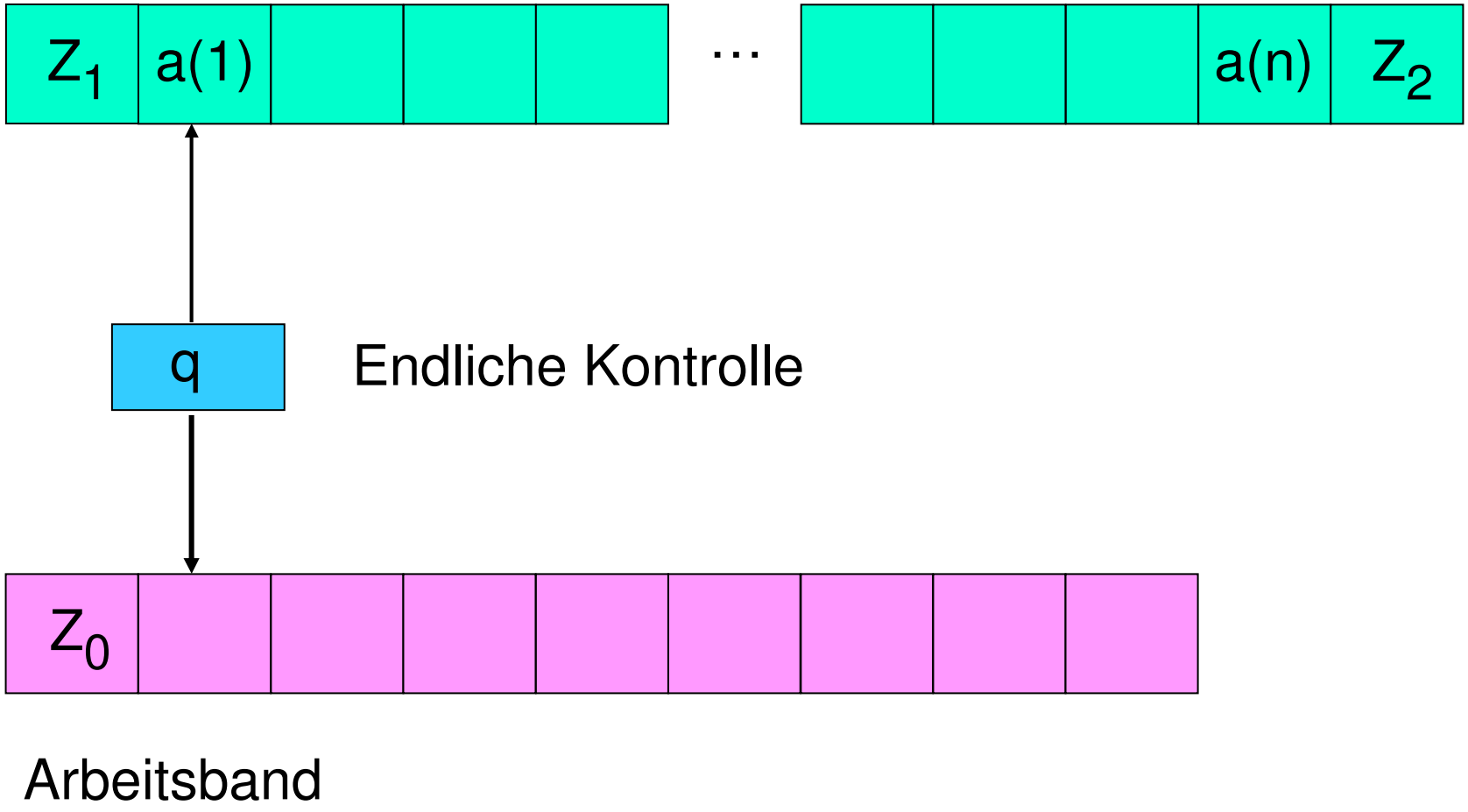
1936: On Computable Numbers, with an application to the Entscheidungsproblem.
Proc. Lond. Math. Soc. (2) 42 pp 230-265, 1936.

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 – 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 Anaylse M einen Endzustand erreicht.

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 – 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.

Kellerautomaten

Im Wesentlichen ist ein Kellerautomat eine Turingmaschine, die das Arbeitsband als sogenannten Keller verwendet, d.h., der Kellerautomat darf bei jedem Schritt nur das oberste Symbol lesen und dieses dann löschen bzw. durch ein endliches Wort über dem sogenannten Kelleralphabet ersetzen. Das linke Begrenzungssymbol des Arbeitsbandes Z_0 heißt Kellergrundsymbol. Der Kellerautomat funktioniert wie ein Stack nach dem Last-in-first-out-(LIFO)Prinzip.

Wir fordern, dass Kellerautomaten in Normalform sind, d.h., dass sie bei dem Übergang in den (einzigen) Endzustand das rechte Begrenzungssymbol Z_2 des Eingabebandes lesen und im Keller gleichzeitig das Kellergrundsymbol löschen, was bedeutet, dass das Eingabewort durch Endzustand und leeren Keller akzeptiert wird.

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 Sortieralgorithmen 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 git 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.

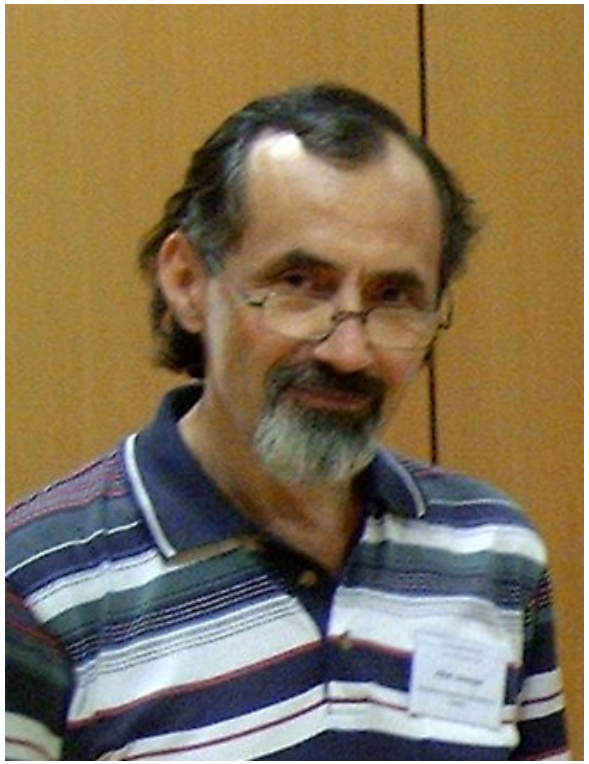
Weitere bekannte Komplexitätsklassen werden durch logarithmische und durch exponentielle Funktionen beschrieben.

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.

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 $[p_1,\ldots,p_k]$, 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,
  $M = \{m(i) \mid 1 \leq i \leq n\}$, $m(i) = [m(i,1),\ldots,m(i,n(i))]$
- $A \in (N \cup T)^+$ 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),…,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≤j≤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

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

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

$M = \{ [X\to X, A\to BB] , [X\to Y, A\to F], [X\to Z, A\to F],$

$[Y\to Y, B\to A], [Y\to X, B\to F], [Z\to Z, B\to a], [Z\to \varepsilon , B\to F] \}$

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

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.

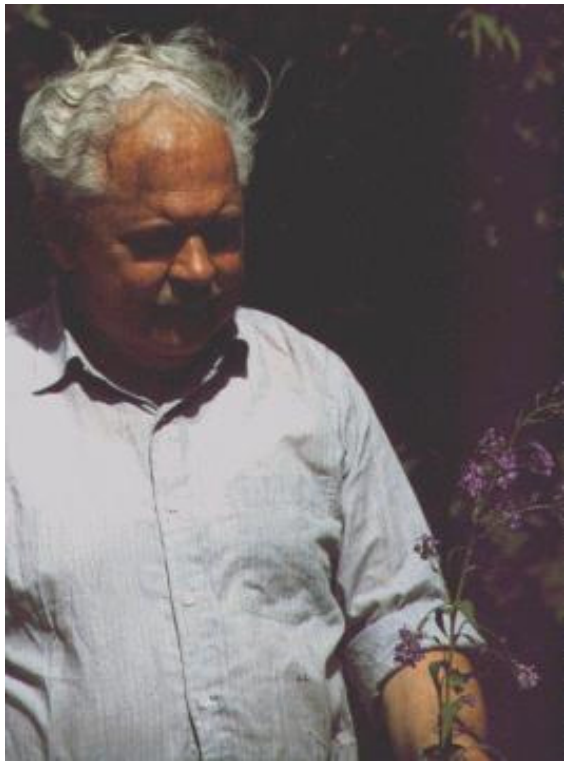
graphkontrollierte Grammatiken

programmierte Grammatiken

Grammatiken mit Kontextbedingungen

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 ET0L-System G ist ein Tupel $(V,T,P_1,\ldots,P_n,w)$; 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 $G = (V,T,P_1,\ldots,P_n,w)$ 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

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

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

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

$G = (V,T,P_1,\ldots,P_n,w)$ 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), $G = (V,P_1,\ldots,P_n,w)$;
- 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 $G = (V,\{a \to a \mid a \in V\},w)$ ein 0L-System, dann gilt:

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

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

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

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

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

4) Sei $G = (\{a,b\},\{a \to b,\ b \to ab\ \},a)$ 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 $a \to w_a$.
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. □