Motivation – natürliche Sprachen

(Satz) $\longrightarrow$ (Substantivphrase)(Verbphrase)
(Satz) $\longrightarrow$ (Substantivphrase)(Verbphrase)(Objektphrase)

(Substantivphrase) $\longrightarrow$ (Artikel)(Substantiv)

(Verbphrase) $\longrightarrow$ (Verb)(Adverb)

(Substantiv) $\longrightarrow$ Hund (Substantiv) $\longrightarrow$ Banane

(Artikel) $\longrightarrow$ der (Artikel) $\longrightarrow$ ein

(Verb) $\longrightarrow$ geht (Verb) $\longrightarrow$ singt

(Adverb) $\longrightarrow$ langsam

Motivation – natürliche Sprachen

$$
\begin{aligned}
(\text{Satz}) &\Longrightarrow (\text{Substantivphrase})(\text{Verbphrase}) \\
&\Longrightarrow (\text{Substantivphrase})(\text{Verb})(\text{Adverb}) \\
&\Longrightarrow (\text{Substantivphrase})\ \text{geht}\ (\text{Adverb}) \\
&\Longrightarrow (\text{Substantivphrase})\ \text{geht langsam} \\
&\Longrightarrow (\text{Artikel})(\text{Substantiv})\ \text{geht langsam} \\
&\Longrightarrow \text{der}\ (\text{Substantiv})\ \text{geht langsam} \\
&\Longrightarrow \text{der Hund geht langsam}
\end{aligned}
$$

$$(\text{Satz}) \Longrightarrow \ldots \Longrightarrow \text{ein Banane singt langsam}$$

Motivation – Programmiersprachen

(unsigned integer) $\rightarrow$ (digit) | (digit){digit}
(unsigned real) $\rightarrow$ (unsigned integer).(digit){digit} |
(unsigned integer)E(scale factor)
(scale factor) $\rightarrow$ (unsigned integer) | (sign) (unsigned integer)
(digit) $\rightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
(sign) $\rightarrow$ + | –

(unsigned real) $\Longrightarrow$ (unsigned integer)E(scale factor)
$\Longrightarrow$ (digit){digit}E(scale factor)
$\Longrightarrow$ 3{digit}E(scale factor)
$\Longrightarrow$ 314E(scale factor) $\Longrightarrow$ 314E(sign)(unsigned integer)
$\Longrightarrow$ 314E–(unsigned integer) $\Longrightarrow$ 314E–(digit)
$\Longrightarrow$ 314E–2

Regelgrammatik – Definition

Definition: Eine Regelgrammatik (oder kurz Grammatik) ist ein Quadrupel

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

wobei

- N und T endliche, disjunkte Alphabete sind ($V = N \cup T$),
- P eine endliche Teilmenge von $(V^* \setminus T^*) \times V^*$ ist, und
- $S \in N$ gilt.

Regelgrammatik – Ableitung und Sprache

Definition: Sei $G = (N, T, P, S)$ eine Regelgrammatik.
Wir sagen, dass aus dem Wort $\gamma \in V^+$ das Wort $\gamma' \in V^*$ erzeugt wird, wenn

$$\gamma = \gamma_1 \alpha \gamma_2, \quad \gamma' = \gamma_1 \beta \gamma_2, \quad \alpha \longrightarrow \beta \in P$$

für gewisse $\gamma_1, \gamma_2 \in V^*$ gelten.

Schreibweise: $\gamma \Longrightarrow \gamma'$

$\Longrightarrow^*$ — reflexiver und transitiver Abschluss von $\Longrightarrow$

Definition: Für eine Grammatik $G = (N, T, P, S)$ definieren wir die von G erzeugte Sprache $L(G)$ durch

$$L(G) = \{w : w \in T^* \; und \; S \Longrightarrow^* w\}.$$

Regelgrammatik – Beispiele I

$G_1 = (\{S, A, B\}, \{a, b\}, \{p_1, p_2, p_3, p_4, p_5\}, S)$
$p_1 = S \to AB,\ p_2 = A \to aA,\ p_3 = A \to \lambda,\ p_4 = B \to Bb,\ p_5 = B \to \lambda$
$L(G_1) = \{a^n b^m : n \geq 0, m \geq 0\}$

$G_2 = (\{S\}, \{a, b\}, \{S \longrightarrow aSb, S \longrightarrow ab\}, S)$
$L(G_2) = \{a^n b^n : n \geq 1\}$

$G_3 = (\{S, A\}, \{a, b\}, \{S \longrightarrow \lambda, S \longrightarrow aS, S \longrightarrow Sb\}, S)$
$L(G_3) = \{a^n b^m : n \geq 0, m \geq 0\}$

$G_4 = (\{S, A\}, \{a, b\}, P_4, S)$
$P_4 = \{S \longrightarrow \lambda, S \longrightarrow aS, S \longrightarrow a, S \longrightarrow A, A \longrightarrow bA, A \longrightarrow b\}$
$L(G_4) = \{a^n b^m : n \geq 0, m \geq 0\}$

Regelgrammatik – Beispiele II

$G_i = (\{S, A, B, B', B''\}, \{a, b, c\}, P_i, S)$ für $i \in \{5, 6\}$

P_5	P_6
	$p_0 = S \to abc$
$p_1 = S \to ABA$	$p_1 = S \to aABbA$
$p_2 = AB \to aAbB'$	$p_2 = AB \to aAbB'$
$p_3 = AB \to abB''$	$p_3 = AB \to abB''$
$p_4 = B'b \to bB'$	$p_4 = B'b \to bB'$
$p_5 = B''b \longrightarrow bB''$	$p_5 = B''b \longrightarrow bB''$
$p_6 = B'A \longrightarrow BAc$	$p_6 = B'A \longrightarrow BAc$
$p_7 = B''A \longrightarrow c$	$p_7 = B''A \longrightarrow cc$
$p_8 = bB \longrightarrow Bb$	$p_8 = bB \longrightarrow Bb$

$L(G_5) = L(G_6) = \{a^n b^n c^n \mid n \geq 1\}$

Regelgrammatik – Beispiele III

$G_7 = (N, T, P, S)$ mit

$$\begin{aligned} N &= \{S\}, \\ T &= \{x, y, z, +, -, \cdot, :, (,)\}, \\ P &= \{S \longrightarrow (S + S), S \longrightarrow (S - S), S \longrightarrow (S \cdot S), S \longrightarrow (S : S), \\ &\qquad S \longrightarrow x, S \longrightarrow y, S \longrightarrow z\}. \end{aligned}$$

$L(G_7)$ besteht aus allen exakt geklammerten arithmetischen Ausdrücken mit den Variablen x, y, z

Regelgrammatik – Beispiele IV

$G_8 = (\{A, I, J\}, T, P, A)$ mit

$$
\begin{aligned}
T &= \{S, P, \mathbf{LOOP}, \mathbf{WHILE}, \mathbf{BEGIN}, \mathbf{END}, :=, \neq, ;, (,) \\
&\quad 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x, [,]\}, \\
P &= \{A \to x[I] := 0,\ A \to x[I] := x[I],\ A \to x[I] := S(x[I]), \\
&\quad A \to x[I] := P(x[I]),\ A \to A; A,\ A \to \mathbf{LOOP}\ x[I]\ \mathbf{BEGIN}\ A\ \mathbf{END}, \\
&\quad A \to \mathbf{WHILE}\ x[I] \neq 0\ \mathbf{BEGIN}\ A\ \mathbf{END}\} \\
&\quad \cup \{I \to Jx,\ J \to Jx \mid x \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}\} \\
&\quad \cup \{I \to x,\ J \to x \mid x \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}\}
\end{aligned}
$$

$L(G_8)$ besteht aus allen **LOOP**/**WHILE**-Programmen

Typen von Regelgrammatiken

Definition: Sei $G = (N, T, P, S)$ eine Regelgrammatik. Wir sagen

a) G ist monoton, wenn für alle Regeln $\alpha \longrightarrow \beta \in P$ die Bedingung $|\alpha| \leq |\beta|$ erfüllt ist, wobei als Ausnahme $S \longrightarrow \lambda$ zugelassen ist, wenn $|\beta'|_S = 0$ für alle Regeln $\alpha' \longrightarrow \beta' \in P$ gilt,

b) G ist kontextabhängig, wenn alle Regeln in P von der Form $uAv \longrightarrow uwv$ mit $u, v \in V^*$, $A \in N$ und $w \in V^+$ sind, wobei als Ausnahme $S \longrightarrow \lambda$ zugelassen ist, wenn $|\beta'|_S = 0$ für alle Regeln $\alpha' \longrightarrow \beta' \in P$ gilt,

c) G ist kontextfrei, wenn alle Regeln in P von der Form $A \longrightarrow w$ mit $A \in N$ und $w \in V^*$ sind,

d) G ist regulär, wenn alle Regeln in P von der Form $A \longrightarrow wB$ oder $A \longrightarrow w$ mit $A, B \in N$ und $w \in T^*$ sind.

Typen von Sprachen

Definition: Eine Sprache L heißt monoton (bzw. kontextabhängig, kontextfrei oder regulär), wenn es eine monotone (bzw. kontextabhängige, kontextfreie oder reguläre) Grammatik G mit $L = L(G)$ gibt.

$\mathcal{L}(REG)$	–	Menge der regulären Sprachen
$\mathcal{L}(CF)$	–	Menge der kontextfreien Sprachen
$\mathcal{L}(CS)$	–	Menge der kontextabhängigen Sprachen
$\mathcal{L}(MON)$	–	Menge der monotonen Sprachen
$\mathcal{L}(RE)$	–	Menge der von Regelgrammatiken erzeugbaren Sprachen

Lemma:
$\mathcal{L}(CS) \subseteq \mathcal{L}(MON) \subseteq \mathcal{L}(RE)$ und $\mathcal{L}(REG) \subseteq \mathcal{L}(CF) \subseteq \mathcal{L}(RE)$

Normalformen I

Lemma:
Zu jeder Regelgrammatik $G = (N, T, P, S)$ kann eine Regelgrammatik $G' = (N', T, P', S)$ so konstruiert werden, dass alle Regeln aus P' von der Form

$$\alpha \longrightarrow \beta \text{ oder } A \longrightarrow a \text{ mit } \alpha, \beta \in (N')^*,\ A \in N',\ a \in T$$

sind und $L(G) = L(G')$ gilt. Ist außerdem G eine monotone, kontextabhängige bzw. kontextfreie Grammatik, so ist auch G' monoton, kontextabhängig bzw. kontextfrei.

Normalformen II

Satz:
Zu jeder monotonen Grammatik $G = (N, T, P, S)$ kann eine monotone Grammatik $G' = (N', T, P', S')$ so konstruiert werden, dass jede Regel aus P' von einer der Formen

$$A \longrightarrow BC,\ A \longrightarrow B,\ AB \longrightarrow CB,\ AB \longrightarrow AC,\ B \longrightarrow a \text{ oder } S' \longrightarrow \lambda$$

mit $A \in N',\ B, C \in N' \setminus \{S'\},\ a \in T$ ist und $L(G) = L(G')$ gilt.

Folgerung: $\mathcal{L}(MON) = \mathcal{L}(CS)$.

Normalformen III

Lemma:
Zu jeder kontextfreien Grammatik $G = (N, T, P, S)$ existiert eine kontextfreie Grammatik $G' = (N', T, P', S)$ derart, dass
i) P' keine Regel der Form $A \longrightarrow \lambda$ mit $A \neq S$ enthält,
ii) $|w|_S = 0$ für alle Regeln $A \longrightarrow w \in P'$ gilt, und
iii) $L(G) = L(G')$ ist.

Folgerung: $\mathcal{L}(CF) \subseteq \mathcal{L}(MON)$.

$(\mathcal{L}(REG) \subseteq \mathcal{L}(CF) \subseteq \mathcal{L}(CS) = \mathcal{L}(MON) \subseteq \mathcal{L}(RE))$

Normalformen IV

Lemma:
Zu jeder kontextfreien Grammatik $G = (N, T, P, S)$ kann eine kontextfreie Grammatik $G' = (N, T, P', S)$ so konstruiert werden, dass P' keine Regel der Form $A \longrightarrow B$ mit $A, B \in N$ enthält und $L(G) = L(G')$ gilt.

Satz (CHOMSKY-Normalform):
Zu jeder kontextfreien Grammatik $G = (N, T, P, S)$ kann eine kontextfreie Grammatik $G' = (N', T, P', S)$ so konstruiert werden, dass P' nur Regeln der Form

$$A \longrightarrow BC \quad \text{und} \quad A \longrightarrow a \quad \text{mit} \quad A, B, C \in N',\ a \in T$$

enthält, wobei $S \longrightarrow \lambda$ als Ausnahme zugelassen ist, falls S in keiner rechten Seite einer Regel aus P' vorkommt, und $L(G) = L(G')$ gilt.

Normalformen V

Satz:
Zu jeder regulären Grammatik $G = (N, T, P, S)$ kann eine reguläre Grammatik $G' = (N', T, P', S)$ so konstruiert werden, dass P' nur Regeln der Form

$$A \longrightarrow aB \quad \text{und} \quad A \longrightarrow a \quad \text{mit} \quad A, B \in N',\ a \in T$$

enthält, wobei $S \longrightarrow \lambda$ als Ausnahme zugelassen ist, falls P' keine Regel der Form $A \longrightarrow aS$ enthält, und $L(G) = L(G')$ gilt.

Schleifensätze I

Satz (Schleifensatz / Pumping-Lemma für reguläre Sprachen):
Sei L eine reguläre Sprache. Dann gibt es eine (von L abhängige) Konstante k derart, dass es zu jedem Wort $z \in L$ mit $|z| \geq k$ Wörter u, v, w gibt, die den folgenden Eigenschaften genügen:

i) $z = uvw$,
ii) $|uv| \leq k$, $|v| > 0$, und
iii) $uv^iw \in L$ für alle $i \geq 0$.

Lemma: $L = \{a^n b^n : n \geq 1\} \in \mathcal{L}(CF) \setminus \mathcal{L}(REG)$.

Schleifensätze II

Satz (Schleifensatz / Pumping-Lemma für kontextfreie Sprachen):
Sei L eine kontextfreie Sprache. Dann gibt es eine (von L abhängige) Konstante k derart, dass es zu jedem Wort $z \in L$ mit $|z| \geq k$ Wörter u, v, w, x, y gibt, die folgenden Eigenschaften genügen:

i) $z = uvwxy$,
ii) $|vwx| \leq k$, $|v| + |x| > 0$, und
iii) $uv^iwx^iy \in L$ für alle $i \geq 0$.

Lemma: $L = \{a^nb^nc^n : n \geq 1\} \in \mathcal{L}(MON) \setminus \mathcal{L}(CF)$.

Satz: $\mathcal{L}(REG) \subset \mathcal{L}(CF) \subset \mathcal{L}(CS) = \mathcal{L}(MON) \subseteq \mathcal{L}(REG)$.

Akzeptierende TURING-Maschine

Definition:

Eine akzeptierende TURING-Maschine M ist ein Sechstupel

$$M = (X, Z, z_0, Q, \delta, F)\,,$$

wobei (X, Z, z_0, Q, δ) eine TURING-Maschine ist und $F \subseteq Q$ gilt.
Die von M akzeptierte Sprache $T(M)$ wird durch

$$T(M) = \{w : w \in X^*,\ (\lambda, z_0, w) \models^* (v_1, q, v_2) \text{ für ein } q \in F\}$$

definiert.

Akzeptierende TURING-Maschine – Beispiel I

$$M_1' = (\{a,b\}, \{z_0, z_a, z_b, q_a, q_b\}, z_0, \{q_a, q_b\}, \delta', \{q_a\})$$

δ	z_0	z_a	z_b
$*$	$(q, *, N)$	(q_a, a, N)	(q_b, b, N)
a	$(z_a, *, R)$	(z_a, a, R)	(z_b, a, R)
b	$(z_b, *, R)$	(z_a, b, R)	(z_b, b, R)

$$T(M_1') = \{aw \mid w \in \{a,b\}^*\}\,.$$

Akzeptierende TURING-Maschine – Beispiel II

$$M_2' = (\{a, b\}, \{z_0, z_1, q\}, z_0, \{q\}, \delta, \{q\})$$

δ	z_0	z_1
$*$	$(z_0, *, N)$	$(q, *, N)$
a	(z_1, a, R)	(z_0, a, R)
b	(z_1, b, R)	(z_0, b, R)

$$T(M_2') = \{w : w \in \{a, b\}^*, |w| \; ungerade\}.$$

Eine Normalform für akzeptierende TURING-Maschinen

Lemma:
Zu jeder akzeptierenden TURING-Maschine M gibt es eine akzeptierende TURING-Maschine M', deren Menge der Stopzustände mit der Menge der akzeptierenden Zustände übereinstimmt und für die $T(M) = T(M')$ gilt. Dabei kann die Menge der Stopzustände von M' einelementig gewählt werden.

Satz:
Eine Sprache wird genau dann von einer TURING-Maschine akzeptiert, wenn sie Definitionsbereich einer TURING-berechenbaren Funktion ist.

Rekursive Sprachen

Definition: Eine Sprache $L \subseteq X^*$ heißt rekursiv, falls es eine akzeptierende TURING-Maschine $M = (X, Z, z_0, Q, \delta, F)$ gibt, die auf jeder Eingabe stoppt und L akzeptiert.

Satz: Eine Sprache $L \subseteq X^*$ ist genau dann rekursiv, wenn sowohl L als auch $X^* \setminus L$ von TURING-Maschinen akzeptiert werden

Satz: Für eine rekursive Menge L ist die charakteristische Funktion

$$\varphi_L(x) = \begin{cases} 0 & x \notin L \\ 1 & x \in L \end{cases}$$

von L algorithmisch berechenbar.

Satz: Die Menge der rekursiven Sprachen ist echt in der Menge der von TURING-Maschinen akzeptierbarten Sprachen enthalten.

TURING-Maschinen versus Regelgrammatiken

Lemma:
Zu jeder TURING-Maschine M gibt es eine Regelgrammatik G mit $L(G) = T(M)$.

Lemma:
Zu jeder Regelgrammatik G gibt es eine nichtdeterministische TURING-Maschine M mit $T(M) = L(G)$.

Satz:
Die folgenden Aussagen sind äquivalent:

i) L wird von einer Regelgrammatik erzeugt.
ii) L wird von einer deterministischen TURING-Maschine akzeptiert.
iii) L wird von einer nichtdeterministischen TURING-Maschine akzeptiert.

Nichtdeterministische TURING-Maschinen I

Definition: Eine nichtdeterministische TURING-Maschine M ist ein Quintupel

$$M = (X, Z, z_0, Q, \tau, F),$$

wobei X, Z, z_0, Q und F wie bei einer (akzeptierenden deterministischen) TURING-Maschine definiert sind und τ eine Funktion

$$\tau : (Z \setminus Q) \times (X \cup \{*\}) \to 2^{Z \times (X \cup \{*\}) \times \{R,N,L\}}$$

ist.

Nichtdeterministische TURING-Maschinen II

$(w_1, z, xw_2) \models (w_1', z', w_2')$, falls $(z', x', r) \in \tau(z, x)$ existiert und
$(w_1, z, w_2) \models (w_1', z', w_2')$ bei einer (deterministischen) TURING-Maschine
mit $(z', x', r) = \delta(z, x)$ gilt

Definition: Es sei $M = (X, Z, z_0, Q, \delta)$ eine nichtdeterministische TURING-Maschine. Die von M akzeptierte Sprache $T(M)$ definieren wir durch

$$T(M) = \{w : w \in X^*,\ (\lambda, z_0, w) \models^* (v_1, q, v_2) \text{ für ein } q \in F\}.$$

Nichtdeterministische TURING-Maschinen – Beispiel

$$M = (\{a, b\}, \{z_0, z_{0,2}z_{1,2}, z_2, z_2', z_2'', q\}, z_0, \{q\}, \tau, \{q\})$$

$$\begin{aligned}
\tau(z_0, x) &= \{(z_2, x, N), (z_3, x, N)\} \quad \text{für } x \in \{a, b\}, \\
\tau(z_i, a) &= \{(z_i', a, R)\} \quad \text{für } i \in \{2, 3\}, \\
\tau(z_i', a) &= \{(z_i'', a, R)\} \quad \text{für } i \in \{2, 3\}, \\
\tau(z_i'', a) &= \{(z_i'', a, R)\}, \\
\tau(z_i, *) &= \{(z_i, *, N)\} \quad \text{für } i \in \{2, 3\}, \\
\tau(z_i', *) &= \{(q, *, N)\} \quad \text{für } i \in \{2, 3\}, \\
\tau(z_i'', *) &= \{(z_{0,i}, *, L)\} \quad \text{für } i \in \{2, 3\}
\end{aligned}$$

Nichtdet. TURING-Maschinen – Beispiel - Fortsetzung

$$
\begin{aligned}
\tau(z_{0,2}, a) &= \{(z_{1,2}, b, L)\}, \quad \tau(z_{1,2}, a) = \{(z_{0,2}, a, L)\}, \\
\tau(z_{j,2}, b) &= \{(z_{i,2}, b, L)\} \quad \text{für } j \in \{0, 1\}, \\
\tau(z_{0,3}, a) &= \{(z_{1,2}, b, L)\}, \quad \tau(z_{1,3}, a) = \{(z_{2,3}, b, L)\}, \\
\tau(z_{2,3}, a) &= \{(z_{0,3}, a, L)\}, \\
\tau(z_{j,3}, b) &= \{(z_{j,3}, b, L)\} \quad \text{für } j \in \{0, 1, 2\}, \\
\tau(z_{0,i}, *) &= \{(z_i, *, R)\} \quad \text{für } i \in \{2, 3\}, \\
\tau(z_{j,i}, *) &= \{(z_{j,i}, *, N)\} \quad \text{für } j \in \{1, 2\} \ , i \in \{2, 3\}
\end{aligned}
$$

$$T(M) = \{a^m : m = 2^n \text{ oder } m = 3^n \text{ für ein } n \geq 0\}$$

Linear beschränkter Automat

Definition:
Ein linear beschränkter Automat ist eine nichtdeterministische TURING-Maschine $M = (X, Z, z_0, Q, \delta, F)$, deren Kopf sich während der Abarbeitung der Eingabe $w \in X^*$ höchstens über $|w| + 2$ verschiedenen Zellen befindet.

Satz:
Eine Sprache ist genau dann kontextabhängig, wenn sie von einem linear beschränkten Automaten akzeptiert werden kann.

Endlicher Automat – Definition

Definition: i) Ein endlicher Automat ist ein Quintupel

$$\mathcal{A} = (X, Z, z_0, F, \delta),$$

wobei
- X und Z Alphabete sind,
- $z_0 \in Z$ und $F \subseteq Z$ gelten,
- δ eine Funktion von $Z \times X$ in Z ist.

ii) Die Erweiterung δ^* von δ auf $Z \times X^*$ definieren wir durch

$\delta^*(z, \lambda) = z$ und $\delta^*(z, wx) = \delta(\delta^*(z, w), x)$ für $w \in X^*,\ x \in X$.

iii) Die durch $\mathcal{A}$ akzeptierte Wortmenge definieren wir durch

$$T(\mathcal{A}) = \{w : w \in X^*, \delta^*(z_0, w) \in F\}.$$

Endlicher Automat – Beispiel

$\mathcal{A} = (X, Z, z_0, F, \delta)$ mit $X = \{a, b, c\}$

$Z = \{z_0, z_1, z_2, z_3\}$,

$F = \{z_2\}$,

$$\delta(z,x) = \begin{cases} z_1 & \text{für } z = z_0, x = a \\ z_2 & \text{für } z = z_1, x = a \\ z_0 & \text{für } z \in \{z_0, z_2\}, x = c \\ z_3 & \text{sonst} \end{cases}$$

$$T(\mathcal{A}) = \{c^{n_1}aac^{n_1}aac^{n_2}aa \dots c^{n_k}aa \mid k \geq 1, n_1 \geq 0, n_j \geq 1, 2 \leq j \leq k\}$$

Nichtdeterministischer endlicher Automat

Definition:

i) Ein nichtdeterministischer endlicher Automat ist ein Quintupel $\mathcal{A} = (X, Z, z_0, F, \delta)$, wobei für X, Z, z_0, F die Bedingungen wie beim endlichen Automaten gelten und δ eine Funktion von $Z \times X$ in die Menge der Teilmengen von Z ist.

ii) Wir definieren $\delta^*(z, \lambda) = \{z\}$ für $z \in Z$, und für $w \in X^*$, $x \in X$ und $z \in Z$ gelte $z' \in \delta^*(z, wx)$ genau dann, wenn es einen Zustand $z'' \in \delta^*(z, w)$ mit $z' \in \delta(z'', x)$ gibt.

iii) Die von $\mathcal{A}$ akzeptierte Wortmenge ist durch

$$T(\mathcal{A}) = \{w : \delta^*(z_0, w) \cap F \neq \emptyset\}$$

definiert.

Endliche Automaten versus reguläre Sprachen

Satz:
Die beiden folgenden Aussagen sind für eine Sprache L äquivalent:
i) L wird von einem (deterministischen) endlichen Automaten akzeptiert.
ii) L wird von einem nichtdeterministischen endlichen Automaten akzeptiert.

Satz:
Für eine Sprache L sind die folgenden Aussagen äquivalent.
i) L ist regulär.
ii) L wird von einem (deterministischen) endlichen Automaten akzeptiert.
iii) L wird von einem nichtdeterministischen endlichen Automaten akzeptiert.

Beispiel für Konstruktion des Automaten aus Grammatik I

$$
\begin{aligned}
G &= (\{S, A, B\}, \{a, b\}, P, S) \\
P &= \{S \to \lambda, S \to aA, S \to a, S \to b, S \to bB, A \to a, \\
&\qquad A \to b, A \to aA, A \to bB, B \to bB, B \to bB, B \to b\} \\
G' &= (\{S, A, B, \$\}, \{a, b\}, P', S) \\
P' &= \{S \to \lambda, S \to aA, S \to a\$, S \to b\$, S \to bB, A \to a\$, \\
&\qquad A \to b\$, A \to aA, A \to bB, B \to bB, B \to b\$, \$ \to \lambda\}.
\end{aligned}
$$

nichtdeterministischer endlicher Automat: $\mathcal{B} = (\{a, b\}, \{S, A, B, \$\}, S, \{S, \$\}, \delta)$

$\delta(S, a) = \delta(A, a) = \{A, \$\}$, $\quad \delta(S, b) = \delta(A, b) = \delta(B, b) = \{B, \$\}$,
$\delta(B, a) = \delta(\$, a) = \delta(\$, b) = \emptyset$.

Beispiel für Konstruktion des Automaten aus Grammatik II

deterministischer endlicher Automat: $\mathcal{B}' = (\{a,b\}, \{S,A,B,\$\}, \{S\}, \{S,\$\}, \delta')$

$$
\begin{aligned}
\{A,\$\} &= \delta'(\{S\},a) = \delta'(\{A\},a) = \delta'(\{S,A\},a) = \delta'(\{S,B\},a) \\
&= \delta'(\{A,B\},a) = \delta'(\{S,A,B\},a), \\
\emptyset &= \delta'(\{B\},a) = \delta'(\emptyset,a) = \delta'(\emptyset,b) \\
\{B,\$\} &= \delta'(\{S\},b) = \delta'(\{A\},b) = \delta'(\{B\},b) = \delta'(\{S,A\},b) \\
&= \delta'(\{S,B\},b) = \delta'(\{A,B\},b) = \delta'(\{S,A,B\},b), \\
\delta'(U \cup \{\$\},x) &= \delta'(U,x) \cup \{\$\} \quad \text{für } U \subseteq \{S,A,B\},\ x \in \{a,b\}
\end{aligned}
$$

Kellerautomat – Definition 1

Definition:
Ein Kellerautomat ist ein Sechstupel

$$\mathcal{M} = (X, Z, \Gamma, z_0, F, \delta),$$

wobei
- X das Eingabealphabet ist,
- Z die endliche Menge von Zuständen ist,
- Γ das Bandalphabet ist,
- $z_0 \in Z$ und $F \subseteq Z$ gelten,
- δ eine Funktion von $Z \times X \times (\Gamma \cup \{\#\})$ in die Menge der endlichen Teilmengen von $Z \times \{R, N\} \times \Gamma^*$ ist,
  wobei $\# \notin \Gamma$, R und N zusätzliche Symbole sind.

Kellerautomat – Definition 2

Definition: Eine Konfiguration K des Kellerautomaten $\mathcal{M}$ ist ein Tripel $(w, z, \alpha\#)$ mit $w \in X^*$, $z \in Z$ und $\alpha \in \Gamma^*$.

Der Übergang von einer Konfiguration K_1 in die nachfolgende Konfiguration K_2 (geschrieben als $K_1 \models K_2$) wird wie folgt beschreiben: Für $x \in X, v \in X^*, z \in Z$, $z' \in Z, \gamma \in \Gamma, \beta \in \Gamma^* \alpha \in \Gamma^*$ gilt

$$\begin{array}{lll}
(xv, z, \gamma\alpha\#) \models (v, z', \beta\alpha\#), & \text{falls} & (z', R, \beta) \in \delta(z, x, \gamma), \\
(xv, z, \gamma\alpha\#) \models (xv, z', \beta\alpha\#), & \text{falls} & (z', N, \beta) \in \delta(z, x, \gamma), \\
(xv, z, \#) \models (v, z', \beta\#), & \text{falls} & (z', R, \beta) \in \delta(z, x, \#), \\
(xv, z, \#) \models (xv, z', \beta\#), & \text{falls} & (z', N, \beta) \in \delta(z, x, \#).
\end{array}$$

Definition: Sei $\mathcal{M}$ ein Kellerautomat. Die von $\mathcal{M}$ akzeptierte Sprache definieren wir durch

$$T(\mathcal{M}) = \{w : (w, z_0, \#) \models^* (\lambda, q, \#) \text{ für ein } q \in F\}.$$

Kellerautomat – Beispiele

$\mathcal{M} = (X, Z, \Gamma, z_0, F, \delta)$

$X = \{a, b\},\ \Gamma = \{a\}$,
$Z = \{z_0, z_1, z_2\},\ F = \{z_1\}$,

$\delta(z_0, a, \#) = \{(z_0, R, aa)\}$,
$\delta(z_0, a, a) = \{(z_0, R, aaa)\}$,
$\delta(z_0, b, a) = \{(z_1, R, \lambda)$,
$\delta(q, b, a) = \{(z_1, R, \lambda)\}$,
$\delta(z, x, \gamma) = \{(z_2, R, \gamma)\}$
in allen anderen Fällen

$T(\mathcal{M}) = \{a^n b^{2n} : n \geq 1\}$

$\mathcal{M}' = (X, Z', \Gamma', z_0', F', \delta')$

$X = (\{a, b\},\ \Gamma' = \{S, a, b\}$,
$Z' = \{z_0', z_1', z_2'\},\ F' = \{z_1'\}$

$\delta'(z_0', x, \#) = \{(z_1', N, S)\}$ für $x \in \{a, b\}$,
$\delta'(z_1', x, S) = \{(z_1', N, aSbb), (z_1', Nabb)\}$
für $x \in \{a, b\}$,
$\delta'(z_1', x, x) = \{(z_1', R, \lambda)\}$ für $x \in \{a, b\}$
$\delta'(z, x, \gamma) = \{(z_2', R, \lambda)\}$
in allen weiteren Fällen

$T(\mathcal{M}') = \{a^n b^{2n} : n \geq 1\}$

Kellerautomaten versus kontextfreie Spachen

Lemma:
Für jede kontextfreie Sprache L gibt es einen Kellerautomaten $\mathcal{M}$ mit $T(\mathcal{M}) = L$.

Lemma:
Zu jedem Kellerautomaten $\mathcal{M}$ gibt es eine kontextfreie Grammatik G mit $L(G) = T(\mathcal{M})$.

Satz:
Die beiden folgenden Aussagen sind für eine Sprache L äquivalent:
i) L ist eine kontextfreie Sprache.
ii) $L = T(\mathcal{M})$ gilt für einen Kellerautomaten $\mathcal{M}$.

Algebraische Operationen

Definition: Es seien L, L_1, L_2 Sprachen über einem Alphabet X. Wir definieren dann das Produkt von L_1 und L_2 durch

$$L_1 \cdot L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}.$$

Weiterhin setzen wir

$$L^0 = \{\lambda\} \quad \text{und} \quad L^{n+1} = L^n \cdot L \quad \text{für} n \geq 0$$

und definieren den KLEENE-Abschluss (oder KLEENE-$*$) von L bzw. den positiven KLEENE-Abschluss (oder KLEENE-$+$) von L durch

$$L^* = \bigcup_{n \geq 0} L^n \quad \text{bzw.} L^+ = \bigcup_{n \geq 1} L^n .$$

Algebraische Operationen – Beispiel 1

$$L = \{ab, ac\} \quad \text{und} \quad L' = \{ab^n a : n \geq 1\}$$

$$\begin{aligned}
L \cdot L = L^2 &= \{abab, abac, acab, acac\}, \\
L \cdot L' &= \{abab^n a : n \geq 1\} \cup \{acab^n a : n \geq 1\}, \\
(L')^3 &= \{ab^i aab^j aab^k a : i \geq 1, j \geq 1, k \geq 1\}, \\
L^* &= \{ax_1 ax_2 \ldots ax_r : r \geq 1, x_i \in \{b, c\}, 1 \leq i \leq r\} \cup \{\lambda\}, \\
(L')^+ &= \{ab^{s_1} aab^{s_2} a \ldots ab^{s_t} a : t \geq 1, s_j \geq 1, 1 \leq j \leq t\}.
\end{aligned}$$

Algebraische Operationen – Beispiel 2

X – Alphabet
X^n – Menge aller Wörter über X der Länge n,
X^* – Menge aller Wörter über X (einschließlich λ)

$$x \in X, \quad \{x\}^* = \{x^n : n \geq 0\}, \quad \{x\}^+ = \{x^n : n \geq 1\} = \{x\}\{x\}^*$$

$$\{c^{n_1}aac^{n_2}aa \ldots c^{n_k}aa \mid k \geq 1, n_1 \geq 0, n_i \geq 1, 2 \leq i \leq k\}$$
$$= \{c\}^*\{a\}\{a\}(\{c\}^+\{a\}\{a\})^* = \{c\}^*\{a\}\{a\}(\{c\}\{c\}^*\{a\}\{a\})^*$$

$$\{a^n b^m : n \geq 1, m \geq 2\} = \{a\}^+\{b\}\{b\}^+.$$

R – Menge aller Wörter über dem Alphabet X mit mindestens einem Buchstaben aus $Y \subseteq X$
$R = \bigcup_{x \in Y} X^*\{x\}X^*$

Abschluss unter Operationen

Satz:
Wenn L und L' reguläre Sprachen sind, so sind auch die Sprachen

i) $L \cup L'$,
ii) $L \cap L'$,
iii) $V^* \setminus L$ (wobei $L \subseteq V^*$ gilt),
iv) $L \cdot L'$,
v) L^+ und L^*

regulär.

Reguläre Ausdrücke – Definition

Definition: Reguläre Ausdrücke über einem Alphabet X sind induktiv wie folgt definiert:

1. $\emptyset$, λ und x mit $x \in X$ sind reguläre Ausdrücke.

2. Sind R_1, R_2 und R reguläre Ausdrücke, so sind auch $(R_1 + R_2)$, $(R_1 \cdot R_2)$ und R^* reguläre Ausdrücke.

3. Ein Ausdruck ist nur dann regulär, wenn dies aufgrund von 1. oder 2. der Fall ist.

Menge zu einem regulären Ausdruck

Definition:
Die einem regulären Ausdruck U über dem Alphabet X zugeordnete Menge $M(U)$ ist induktiv durch die folgenden Festlegungen definiert:

- $M(\emptyset) = \emptyset$, $M(\lambda) = \{\lambda\}$ und $M(x) = \{x\}$ für $x \in X$,
- Sind R_1, R_2 und R reguläre Ausdrücke, so gelten

$$\begin{aligned} M((R_1 + R_2)) &= M(R_1) \cup M(R_2), \\ M((R_1 \cdot R_2)) &= M(R_1) \cdot M(R_2), \\ M(R^*) &= (M(R))^* . \end{aligned}$$

Reguläre Ausdrücke – Beispiel

	Ausdruck	Menge
R_1	$= a$	$\{a\}$
R_2	$= b$	$\{b\}$
R_3	$= c$	$\{c\}$
R_1'	$= (R_1 \cdot R_1) = (a \cdot a)$	$\{a\}\{a\} = \{a^2\}$
R_2'	$= R_2^* = b^*$	$\{b\}^* = \{b^m : m \geq 0\}$
R	$= (R_2' + R_1') = (b^* + (a \cdot a))$	$\{b^m : m \geq 0\} \cup \{a^2\}$
R_3'	$= R_3^* = c^*$	$\{c\}^* = \{c^n : n \geq 0\}$
R_3''	$= (R_3 \cdot R_3') = (c \cdot c^*)$	$\{c\}\{c^n : n \geq 0\} = \{c^n : c \geq 1\}$
R'	$= (R \cdot R_3'')$	$(\{b^m : m \geq 0\} \cup \{a^2\}) \cdot \{c^n : n \geq 1\} =$
	$= ((b^* + (a \cdot a)) \cdot (c \cdot c^*))$	$\{b^m c^n : m \geq 0, n \geq 1\} \cup \{a^2 c^n : n \geq 1\}$

Satz von KLEENE

Satz:
Eine Sprache L ist genau dann regulär, wenn es einen regulären Ausdruck R mit $M(R) = L$ gibt.

Satz:
Eine Sprache $L \subseteq X$ ist genau dann regulär, wenn sie in endlich vielen Schritten mittels der Operationen Vereinigung, Produkt und KLEENE-Abschluss aus den Mengen $\emptyset$, $\{\lambda\}$ und $\{x\}$ für $x \in X$ erzeugt werden kann.

Entscheidungsprobleme bei formalen Sprachen

Leerheitsproblems für kontextfreie Sprachen:

Gegeben:	kontextfreie Grammatik G	oder	Gegeben:	Kellerautomat $\mathcal{M}$
Frage:	Ist $L(G)$ leer ?		Frage:	Ist $T(\mathcal{M})$ leer ?

Endlichkeitsproblem: Gegeben: Grammatik G
Frage: Ist $L(G)$ endlich ?

Äquivalenzproblem Gegeben: Grammatiken G_1 und G_2
Frage: Gilt $L(G_1) = L(G_2)$?

Mitgliedsproblem Gegeben: Grammatik $G = (N, T, P, S)$ und Wort $w \in T^*$
Frage : Ist w in $L(G)$ enthalten ?

Mitgliedsproblem – Resultate

Satz: Das Mitgliedsproblem ist für (beliebige) Regelgrammatiken unentscheidbar.

Satz: Das Mitgliedsproblem ist für monotone (oder kontextsensitive) Grammatiken entscheidbar.

Satz: i) Das Mitgliedsproblem ist für kontextfreie Grammatiken $G = (N, T, P, S)$ in CHOMSKY-Normalform in der Zeit $O(\#(P) \cdot |w|^3\})$ entscheidbar.
ii) Das Mitgliedsproblem ist für kontextfreie Grammatiken $G = (N, T, P, S)$ in der Zeit $O(\#(N) \cdot \#(P) \cdot |w|^3)$ entscheidbar.

Satz: Für eine reguläre Grammatik $G = (N, T, P, S)$ und ein Wort w ist in der Zeit $O(k(G) \cdot \#(N) \cdot |w|)$ entscheidbar, ob $w \in L(G)$ gilt.

Satz: $\mathcal{L}(MON) \subset \mathcal{L}(RE)$

Endlichkeits–, Leerheits– und Äquivalenzproblem – Resultate

Satz: Das Leerheits- und das Endlichkeitsproblem sind für beliebige Regelgrammatiken und monotone (kontextsensitive) Grammatiken unentscheidbar.

Satz: Für kontextfreie Grammatiken sind das Leerheits- und Endlichkeitsproblem in der Zeit $O(\#(V) \cdot k(G))$ entscheidbar.

Satz: Das Äquivalenzproblem ist für kontextfreie Grammatiken unentscheidbar.

Satz: Das Äquivalenzproblem für reguläre Grammatiken $G_1 = (N_1, T, P_1, S_1)$ und $G_2 = (N_2, T, P_2, S_2)$ ist in der Zeit $O(n \cdot k^2)$ mit $n = max\{\#(N_1), \#(N_2)\}$ und $k = max\{k(G_1), k(G_2)\}$ entscheidbar.

Beweis der Unentscheidbarkeit des Äquivalenzproblems bei kontextfreien Sprachen

$G_1 = (N, T \cup \{c\}, P, S)$ und $G_2 = (N \cup \{S', S''\}, T \cup \{c\}, P \cup P', S')$

$N = \{S, S_u, S_r, S_l\}$,

$P = \{S \to xSx : x \in T\} \cup \{S \to xS_l, S \to S_r x\} \cup \{S \to xS_u y : x, y \in T, x \neq y\}$
$\cup \{S_u \to xS_u y : x, y \in T\} \cup \{S_u \to S_r, S_u \to S_l\}$
$\cup \bigcup_{x \in T} \{S_l \to xS_l, S_r \to S_r x\}$
$\cup \{S_u \to c, S_l \to c, S_r \to c\},$

$P' = \{S' \to S, S' \to S''\} \cup \bigcup_{i=1}^{n} \{S'' \to u_i S'' v_i^R, S'' \to u_i c v_i^R\}$

$L(G_1) = \{\alpha c \beta^R : \alpha, \beta \in T^+, \alpha \neq \beta\}$

$L(G_2) = L(G_1) \cup \{u_{i_1} u_{i_2} \dots u_{i_k} c v_{i_k} v_{i_{k-1}} \dots v_{i_1} : k \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq k\}$

Entscheidbarkeit bei formalen Sprachen – Zusammenfassung

	$\mathcal{L}(REG)$	$\mathcal{L}(CF)$	$\mathcal{L}(CS)$	$\mathcal{L}(RE)$
Mitgliedsproblem	$+$	$+$	$+$	$-$
Leerheitsproblem	$+$	$+$	$-$	$-$
Endlichkeitsproblem	$+$	$+$	$-$	$-$
Äquivalenzproblem	$+$	$-$	$-$	$-$