

Endliche Automaten

- Modellierungskonzept mit vielen brauchbaren Eigenschaften
- schnelle Spracherkennung
- graphisch-visuelle Beschreibung
- automatische Korrektheitsbeweise
- gute Kompositionalitätseigenschaften

Aber:

- Was können endliche Automaten nicht?

Pumping-Lemma

Sei L eine von einem endlichen Automaten erkannte Sprache.

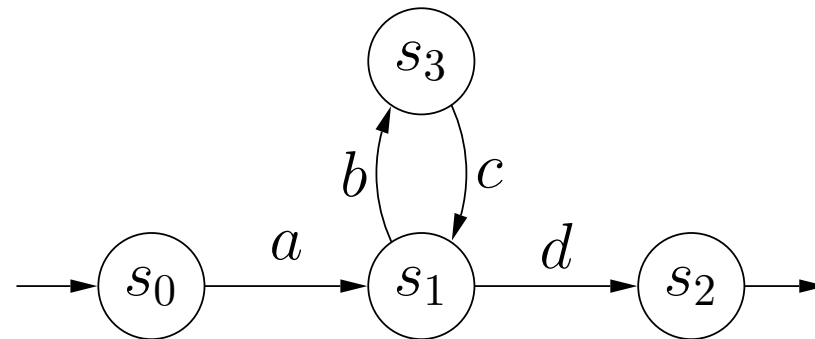
Dann existiert eine natürliche Zahl $p \in \mathbb{N}$ derart, dass jedes Wort $w \in L$ mit $length(w) \geq p$ zerlegt werden kann in drei Teilwörter $w = xyz$ mit

1. $length(xy) \leq p$
2. $y \neq \lambda$
3. $xy^iz \in L$ für alle $i \geq 0$.

Beispiel

$$L = \{a(bc)^n d \mid n \in \mathbb{N}\}$$

EA für L :



Für $p = 4$ hat jedes $w \in L$ mit $length(w) \geq 4$ die Form $a(bc)^n d$ mit $n \geq 1$. Eine **mögliche Zerlegung** von w in xyz ist $x = a$, $y = bc$, $z = (bc)^{n-1}d$, denn $length(xy) = 3 \leq 4$, $y \neq \lambda$ und $xy^i z = a(bc)^i (bc)^{n-1} d \in L \quad \forall i \in \mathbb{N}$.

Anwendung des Pumping-Lemmas

Das Pumping-Lemma kann benutzt werden, um zu zeigen, dass eine Sprache von keinem endlichen Automaten erkannt wird.

- ▶ **Ziel:** mit Hilfe des Pumping-Lemmas beweisen, dass eine Sprache L von keinem EA erkannt wird.

- ▶ **Vorgehensweise (Widerspruchsbeweis)**
 - (a) **Nimm an:** L wird von einem EA erkannt.
 - (b) **Wähle** $w \in L$ mit $length(w) \geq p$, wobei p die Konstante aus dem Pumping-Lemma ist.
 - (c) **Finde** für **jede** Zerlegung $xyz = w$ mit $y \neq \lambda$ und $length(xy) \leq p$ ein $i \in \mathbb{N}$, so dass $xy^iz \notin L$. **Falls** dies gelingt, ist der Beweis **fertig**. **Sonst** gehe zu Schritt 2.

Behauptung: Die Sprache $L_{balance} = \{a^n b^n \mid n \in \mathbb{N}\}$ wird von keinem endlichen Automaten erkannt.

Beweis (Widerspruch)

1. **Annahme:** L wird von einem endlichen Automaten erkannt. Sei p die Konstante aus dem Pumping-Lemma.
2. Sei $w = a^p b^p$ (d.h. $w \in L_{balance}$ und $length(w) \geq p$).
3. Sei $w = xyz$ mit $y \neq \lambda$ und $length(xy) \leq p$. Dann liegt das Teilwort y in der ersten Hälfte von w , d.h. $y = a^k$ für ein $k > 0$ und es gilt $xy^0z = xz = a^{p-k}b^p \notin L_{balance}$ für $k \neq 0$. (Widerspruch)

Endliche Automaten: Ausblick

Varianten endlicher Automaten

- ▶ Endliche Automaten mit λ -Übergängen
 - können aktuellen Zustand wechseln, ohne ein Zeichen zu lesen;
 - sind nützlich für die Modellierung mit endlichen Automaten;
 - sind **äquivalent** zu DEA's.

► Verallgemeinerte endliche Automaten

- lesen in jedem Schritt ein Wort statt eines Zeichens;
- sind nützlich für die Modellierung mit endlichen Automaten (“Abkürzen” möglich);
- sind **äquivalent** zu DEA's.

Minimierung endlicher Automaten

- Algorithmus, der für jeden DEA einen **äquivalenten minimalen** DEA liefert (minimale Anzahl von Zuständen), der bis auf Zustandsnamen eindeutig ist.
- Minimierungsverfahren kann benutzt werden, um zu entscheiden, ob zwei endliche Automaten äquivalent sind, d.h., ob sie dieselbe Sprache erkennen.

Kontextfreie Grammatiken

Motivation

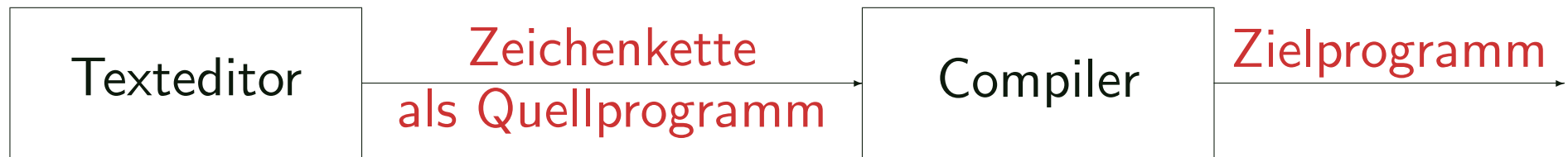
- ▶ Programme aller bekannten Programmiersprachen lassen sich nicht durch endliche Automaten erkennen (z.B. wegen der Klammerstrukturen).
- ▶ Endliche Automaten erkennen aber einzelne Konstrukte (Schlüsselwörter, Namen, . . .) und erlauben lexikalische Analyse.
- ▶ Erkennung der Gesamtprogramme im Rahmen der Syntaxanalyse auf der Basis kontextfreier Grammatiken.
- ▶ Ursprünglich von Chomsky in den 1950er Jahren eingeführt zur Beschreibung natürlicher Sprachen.

Anwendungsgebiete kontextfreier Grammatiken

- ▶ Definition und Implementierung von Programmier- und Markup-Sprachen
 - Java, C, HTML, . . .
 - Compilerbau: Parser
 - Automatische Generierung von Parsern
- ▶ DTD's in XML

Definition von Programmiersprachen

- Übliches Vorgehen beim Programmieren (im Kleinen)



- **ProgrammiererIn**: Welche Form muss ein Text bekommen, um ein Programm zu sein?
- **Computer**: Wie unterscheidet der Compiler zwischen Texten, die Programme sind und die keine Programme sind?

⇒ kontextfreie Grammatiken

Kontextfreie Grammatiken und Sprachen

Kontextfreie Grammatik: $G = (N, T, P, S)$ mit

- N : Menge nichtterminaler Zeichen,
- T : Menge terminaler Zeichen mit $N \cap T = \emptyset$,
- $P \subseteq N \times (N \cup T)^*$: endliche Menge kontextfreier Produktionen
- $S \in N$: Startsymbol

► Schreibweise für Produktionen $(A, u) \in P$: $A ::= u$

► Abkürzung für $A ::= u_1, A ::= u_2, \dots, A ::= u_k$:

$$A ::= u_1 | u_2 | \dots | u_k$$

Beispiel

$$G_{\text{bracket}_0} = (\{S\}, \{a, b\}, \{S ::= aSb \mid \lambda\}, S)$$

Ableitungen

Direkte Ableitung:

$$w = xAy \xrightarrow[p]{} xuy = w'$$

mit $w, w', x, y, u, v \in (N \cup T)^*$, $p = (A ::= u)$.

- **Schreibweise:** $w \xrightarrow[P]{} w'$,
falls P eine Menge von Produktionen ist mit $p \in P$.
- **Beispiel:** $aSb \xrightarrow[S ::= aSb]{} a^2Sb^2$

Ableitung (Iteration direkter Ableitungen)

$$w_0 \xrightarrow[p_1]{} w_1 \xrightarrow[p_2]{} \cdots \xrightarrow[p_n]{} w_n$$

für $w_0, \dots, w_n \in (N \cup T)^*$ und Produktionen p_1, \dots, p_n
($n \geq 1$)

Schreibweisen:

- $w_0 \xrightarrow[P]{} \cdots \xrightarrow[P]{} w_n$ oder $w_0 \xrightarrow[n]{P} w_n$ oder $w_0 \xrightarrow[*]{P} w_n$,
falls $p_1, \dots, p_n \in P$.
- $w \xrightarrow[*]{} w'$, falls P aus dem Kontext klar ist.

Beispiel

$$G_{\text{bracket}_0} = (\{S\}, \{a, b\}, \{S ::= aSb \mid \lambda\}, S)$$

$$\begin{array}{ccccccc} S & \longrightarrow & aSb & \longrightarrow & a^2Sb^2 & \longrightarrow & a^3Sb^3 & \longrightarrow & a^3b^3 \\ S ::= aSb & & S ::= aSb & & S ::= aSb & & S ::= \lambda & & \end{array}$$

Nullableitung

$$w \xrightarrow[P]{0} w$$

für alle $w \in (N \cup T)^*$.

Erzeugte Sprache

- $G = (N, T, P, S)$: kontextfreie Grammatik

Erzeugte Sprache

$$L(G) = \{w \in T^* \mid S \xrightarrow[P]{*} w\}$$

Beispiele kontextfreier Grammatiken

► $G_{\text{bracket}_0} = (\{S\}, \{a, b\}, \{S ::= aSb \mid \lambda\}, S)$

$$\begin{array}{l}
 S \xrightarrow[S ::= \lambda]{} \lambda, \quad S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= \lambda]{} ab \\
 S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= aSb]{} a^2Sb^2 \xrightarrow[S ::= \lambda]{} a^2b^2 \\
 S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= aSb]{} a^2Sb^2 \xrightarrow[S ::= aSb]{} a^3Sb^3 \xrightarrow[S ::= \lambda]{} a^3b^3 \\
 \dots \\
 S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= aSb]{} a^2Sb^2 \xrightarrow[S ::= aSb]{} a^3Sb^3 \xrightarrow[S ::= aSb]{} \dots \xrightarrow[S ::= \lambda]{} a^n b^n
 \end{array}$$

► $L(G_{\text{bracket}_0}) = \{a^n b^n \mid n \in \mathbb{N}\}.$

Wörter über A

$(\{ \langle word \rangle \}, A, P, \langle word \rangle)$ mit den Produktionen

$$\langle word \rangle ::= \lambda \mid x \langle word \rangle$$

für alle $x \in A$.

Erzeugung von abc :

$$\begin{aligned} \langle word \rangle &\longrightarrow a \langle word \rangle \longrightarrow ab \langle word \rangle \\ &\longrightarrow abc \langle word \rangle \longrightarrow abc \end{aligned}$$

Klammerstrukturen

► $G_{\text{bracket}_1} = (\{S\}, \{[,]\}, \{S ::= [S] \mid < S > \mid SS \mid \lambda\}, S)$

$$\begin{aligned} S &\rightarrow SS \rightarrow S[S] \rightarrow < S > [S] \rightarrow < SS > [S] \rightarrow \\ &< [S]S > [S] \rightarrow < []S > [S] \rightarrow < []S > [< S >] \rightarrow \\ &< []S > [< >] \rightarrow < [][S] > [< >] \rightarrow < [][] > [< >] \end{aligned}$$

Wörter der Form $a^{2k}b^{3l}$

► $G_{2a3b} = (\{S, A, B\}, \{S ::= AB, A ::= a^2 A \mid \lambda, B ::= b^3 B \mid \lambda\}, S)$

$$S \rightarrow AB \rightarrow a^2 AB \rightarrow a^4 AB \rightarrow a^4 B \rightarrow a^4 b^3 B \rightarrow a^4 b^3$$

oder

$$S \rightarrow AB \rightarrow Ab^3 B \rightarrow a^2 Ab^3 B \rightarrow a^2 Ab^3 \rightarrow a^4 Ab^3 \rightarrow a^4 b^3$$

$$L(G_{2a3b}) = \{a^{2k}b^{3l} \mid k, l \in \mathbb{N}\}$$

Reguläre Ausdrücke über I

$$\langle \text{reg} \rangle ::= \text{empty} \mid \text{lambda} \mid x \mid (\langle \text{reg} \rangle + \langle \text{reg} \rangle) \mid (\langle \text{reg} \rangle \circ \langle \text{reg} \rangle) \mid (\langle \text{reg} \rangle^*)$$

für alle $x \in I$.

$$\begin{aligned} \langle \text{reg} \rangle &\longrightarrow (\langle \text{reg} \rangle + \langle \text{reg} \rangle) \longrightarrow (\text{lambda} + \langle \text{reg} \rangle) \\ &\longrightarrow (\text{lambda} + (\langle \text{reg} \rangle)^*) \longrightarrow (\text{lambda} + (a)^*) \end{aligned}$$

Boolesche Ausdrücke

$$\langle \text{boolexp} \rangle ::= \text{true} \mid \text{false} \mid \langle \text{var} \rangle \mid (\neg \langle \text{boolexp} \rangle) \mid$$
$$(\langle \text{boolexp} \rangle \langle \text{boolop} \rangle \langle \text{boolexp} \rangle)$$
$$\langle \text{boolop} \rangle ::= \wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow$$
$$\langle \text{var} \rangle ::= b \langle \text{cipherseq} \rangle$$
$$\langle \text{cipherseq} \rangle ::= \langle \text{cipher} \rangle \mid \langle \text{cipher} \rangle \langle \text{cipherseq} \rangle$$
$$\langle \text{cipher} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Syntaktische Beschreibung von PASCALchen

$\langle prog \rangle ::= \langle comp \rangle$
 $\langle comp \rangle ::= \text{begin } \langle stmtlist \rangle \text{ end} \mid \text{begin end}$
 $\langle stmtlist \rangle ::= \langle stmt \rangle \mid \langle stmt \rangle ; \langle stmtlist \rangle$
 $\langle stmt \rangle ::= \langle comp \rangle \mid \langle assign \rangle \mid \langle while \rangle$
 $\langle assign \rangle ::= \langle var \rangle := \langle expr \rangle$
 $\langle while \rangle ::= \text{while } \langle var \rangle \neq \langle var \rangle \text{ do } \langle stmt \rangle$
 $\langle expr \rangle ::= 0 \mid \text{succ}(\langle var \rangle) \mid \text{pred}(\langle var \rangle)$
 $\langle var \rangle ::= X \langle nat \rangle$
 $\langle nat \rangle ::= \mid \langle one - nine \rangle \langle cipherseq \rangle$
 $\langle cipherseq \rangle ::= \lambda \mid \langle cipher \rangle \langle cipherseq \rangle$
 $\langle cipher \rangle ::= 0 \mid \langle one - nine \rangle$
 $\langle one - nine \rangle ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Kontextfreiheitslemma

Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik, $u \xrightarrow[n]{P} v$ eine Ableitung der Länge n und $u = u_1 u_2 \cdots u_k$ eine Zerlegung von u in k Teilwörter.

Dann gibt es k Ableitungen $u_i \xrightarrow[n_i]{P} v_i$, so dass $v = v_1 \cdots v_k$

und $n = \sum_{i=1}^k n_i$.