Varianten endlicher Automaten

Endliche Automaten mit λ-Übergängen

- können aktuellen Zustand wechseln, ohne ein Zeichen zu lesen;
- sind praktisch (vereinfachen oft die Modellierung mit endlichen Automaten);
- sind äquivalent zu DEA's.

Verallgemeinerte endliche Automaten

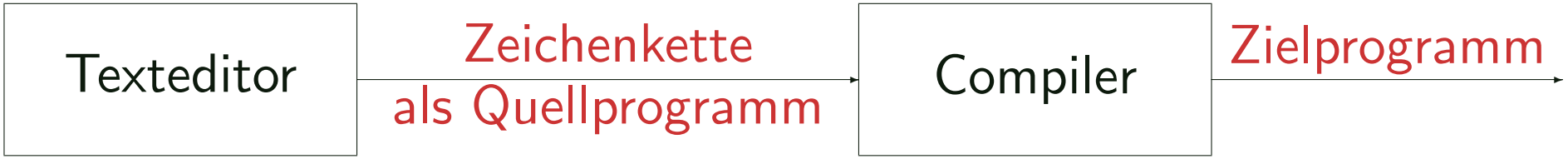
- lesen in jedem Schritt ein Wort statt eines Zeichens;
- sind nützlich für die Modellierung mit endlichen Automaten ("Abkürzen" möglich);
- sind äquivalent zu DEA's.

Minimale endliche Automaten

- Jeder DEA A kann in einen äquivalenten DEA $Min(A)$ übersetzt werden, dessen Anzahl von Zuständen minimal ist. $Min(A)$ ist bis auf Zustandsnamen eindeutig.
- Minimierungsverfahren kann benutzt werden, um zu entscheiden, ob zwei endliche Automaten äquivalent sind, d.h., ob sie dieselbe Sprache erkennen.

Definition von Programmiersprachen

- Übliches Vorgehen beim Programmieren (im Kleinen)



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

Grenzen endlicher Automaten

- 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.

Kontextfreie Grammatiken

Kontextfreie Grammatiken

- sind eine Teilklasse der Chomsky-Grammatiken, die N. Chomsky in den 1950er Jahren ursprünglich zur Beschreibung natürlicher Sprachen eingeführt hat,
- formalisieren die Syntax von Programmier- und Markup-Sprachen (wie z.B. Java, C, HTML, LaTeX, . . .),
- dienen als Eingabe für Parsergeneratoren (wie z.B. yacc oder JavaCC),
- beschreiben den strukturellen Aufbau von Dokumenten

Definition

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 $A ::= u_1, A ::= u_2, \ldots A ::= u_k$:

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

Beispiel

$$G_{bracket_0} = (\{S\}, \{a, b\}, \{S ::= aSb \mid \lambda\}, S)$$

Ableitungen

Direkte Ableitung:

$$w = xAy \underset{p}{\longrightarrow} xuy = w'$$

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

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

Ableitung (Iteration direkter Ableitungen)

$$w_0 \underset{p_1}{\longrightarrow} w_1 \underset{p_2}{\longrightarrow} \cdots \underset{p_n}{\longrightarrow} w_n$$

für $w_0, \ldots, w_n \in (N \cup T)^*$ und Produktionen $p_1, \ldots, p_n$

Schreibweisen:

- $w_0 \underset{P}{\longrightarrow} \cdots \underset{P}{\longrightarrow} w_n$ oder $w_0 \overset{n}{\underset{P}{\longrightarrow}} w_n$ oder $w_0 \overset{*}{\underset{P}{\longrightarrow}} w_n$, falls $p_1, \ldots, p_n \in P$.

- $w \overset{*}{\longrightarrow} w'$, falls P aus dem Kontext klar ist.

Beispiel

$$G_{bracket_0} = (\{S\}, \{a, b\}, \{S ::= aSb \mid \lambda\}, S)$$

$$S \underset{S ::= aSb}{\longrightarrow} aSb \underset{S ::= aSb}{\longrightarrow} a^2Sb^2 \underset{S ::= aSb}{\longrightarrow} a^3Sb^3 \underset{S ::= \lambda}{\longrightarrow} a^3b^3$$

Nullableitung

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

für alle $w \in (N \cup T)^*$.

Erzeugte Sprache

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

Erzeugte Sprache

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

Die von kontextfreien Grammatiken erzeugten Sprachen heißen **kontextfreie Sprachen**.

Beispiel: Wörter der Form $a^n b^n$

- $G_{bracket_0} = (\{S\}, \{a, b\}, \{S ::= aSb \mid \lambda\}, S)$

$$S \underset{S::=\lambda}{\longrightarrow} \lambda, \quad S \underset{S::=aSb}{\longrightarrow} aSb \underset{S::=\lambda}{\longrightarrow} ab$$
$$S \underset{S::=aSb}{\longrightarrow} aSb \underset{S::=aSb}{\longrightarrow} a^2Sb^2 \underset{S::=\lambda}{\longrightarrow} a^2b^2$$
$$S \underset{S::=aSb}{\longrightarrow} aSb \underset{S::=aSb}{\longrightarrow} a^2Sb^2 \underset{S::=aSb}{\longrightarrow} a^3Sb^3 \underset{S::=\lambda}{\longrightarrow} a^3b^3$$
$$\ldots$$
$$S \underset{S::=aSb}{\longrightarrow} aSb \underset{S::=aSb}{\longrightarrow} a^2Sb^2 \underset{S::=aSb}{\longrightarrow} a^3Sb^3 \underset{S::=aSb}{\longrightarrow} \cdots \underset{S::=\lambda}{\longrightarrow} a^nb^n$$

- $L(G_{bracket_0}) = \{a^n b^n \mid n \in \mathbb{N}\}.$

Wörter über A

$(\{<word>\}, A, P, <word>)$ mit den Produktionen

$$\boxed{<word> ::= \lambda \mid x<word>}$$

für alle $x \in A$.

Erzeugung von abc:

$$<word> \longrightarrow a<word> \longrightarrow ab<word>$$
$$\longrightarrow abc<word> \longrightarrow abc$$

Klammerstrukturen

- $G_{bracket_1} = (\{S\}, \{[,], <, >\}, P_{bracket_1}, S)$ mit

$$P_{bracket_1} = \{S ::= [S] \mid <S> \mid SS \mid \lambda\}$$

$$S \rightarrow SS \rightarrow S[S] \rightarrow < S > [S] \rightarrow < SS > [S] \rightarrow$$
$$< [S]S > [S] \rightarrow < []S > [S] \rightarrow < []S > [< S >] \rightarrow$$
$$< []S > [<>] \rightarrow < [][S] > [<>] \rightarrow < [][] > [<>]$$

Wörter der Form $a^{2k}b^{3l}$

- $G_{2a3b}= (\{S, A, B\}, \{a, b\}, P_{2a3b}, S)$ mit

$$P_{2a3b} = \{S ::= AB, A ::= a^2A \mid \lambda, B ::= b^3B \mid \lambda\}$$

- Erzeugung von a^4b^3:

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

oder

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

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

Reguläre Ausdrücke über I

$$<reg> ::= empty \mid lambda \mid x \mid (<reg> + <reg>) \mid (<reg> \circ <reg>) \mid (<reg>^*)$$

für alle $x \in I$.

$$<reg> \longrightarrow (<reg> + <reg>) \longrightarrow (lambda + <reg>) \longrightarrow (lambda + (<reg>)^*) \longrightarrow (lambda + (a)^*)$$

Boolesche Ausdrücke

$$
\begin{aligned}
&<boolexp> ::= \ true \mid false \mid \\
&\qquad\qquad <var> \ \mid (\neg <boolexpr>) \mid \\
&\qquad\qquad (<boolexpr><boolop><boolexpr>) \\
&<boolop> ::= \wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow \\
&<var> ::= b<cipherseq> \\
&<cipherseq> ::= \ <cipher> \ \mid \\
&\qquad\qquad <cipher><cipherseq> \\
&<cipher> ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Syntaktische Beschreibung von PASCALchen

$$
\begin{aligned}
&<prog> ::= <comp> \\
&<comp> ::= \text{begin } <stmtlist> \text{ end} \mid \text{begin end} \\
&<stmtlist> ::= <stmt> \mid <stmt>; <stmtlist> \\
&<stmt> ::= <comp> \mid <assign> \mid <while> \\
&<assign> ::= <var> := <expr> \\
&<while> ::= \text{while } <var> \neq <var> \text{ do } <stmt> \\
&<expr> ::= 0 \mid \text{succ}(<var>) \mid \text{pred}(<var>) \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
&<var> ::= X <nat> \\
&<nat> ::= \mid <one-nine><cipherseq> \\
&<cipherseq> ::= \lambda \mid <cipher><cipherseq> \\
&<cipher> ::= 0 \mid <one-nine> \\
&<one-nine> ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Kontextfreiheitslemma

Sei

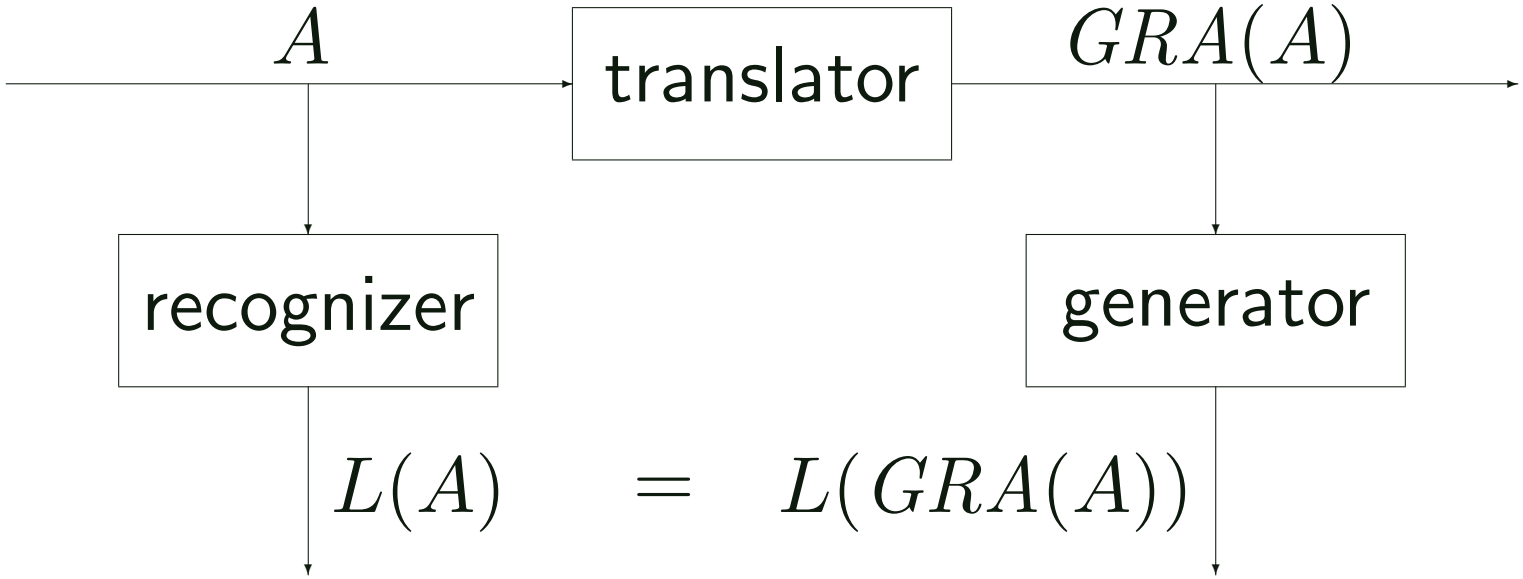
- $G = (N, T, P, S)$ eine kontextfreie Grammatik,
- $u \xrightarrow[P]{n} v$ eine Ableitung und
- $u = u_1 u_2 \cdots u_k$ eine Zerlegung von u.

Dann gibt es Ableitungen $u_i \xrightarrow[P]{n_i} v_i$ $(i = 1, \ldots, k)$, so dass $v = v_1 \cdots v_k$ und $n = \sum_{i=1}^{k} n_i$.

Fragestellungen

- Sind kfG's kompositional?
- Wie hängen endliche Automaten und kfG's zusammen?
- Ist für kfG's das Wortproblem schnell lösbar? (Wird in Theo 2 behandelt.)
- Was können kfG's nicht?

Übersetzung endlicher Automaten in (rechtslineare) Grammatiken



Konstruktion von $GRA(A)$

Sei $A = (Z, I, d, s_0, F)$ ein NEA.

$GRA(A) = (Z, I, P_A, s_0)$ mit
$P_A = \{s ::= xs' \mid s' \in d(s, x)\} \cup \{s'' ::= \lambda \mid s'' \in F\}$

Satz
$L(A) = L(GRA(A))$.