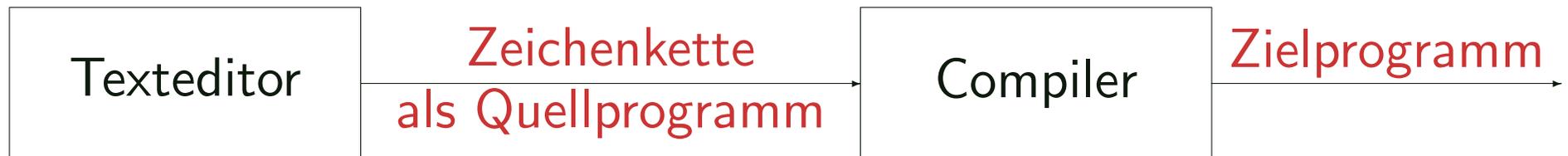


Definition von Programmiersprachen

- 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 $A ::= u_1, A ::= u_2, \dots, A ::= u_k$:

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

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

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

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

$$S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= aSb]{} a^2Sb^2 \xrightarrow[S ::= aSb]{} a^3Sb^3 \xrightarrow[S ::= \lambda]{} a^3b^3$$

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 $a^n b^n$

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

$$S \xrightarrow[S ::= \lambda]{} \lambda, \quad S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= \lambda]{} ab$$

$$S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= aSb]{} a^2 S b^2 \xrightarrow[S ::= \lambda]{} a^2 b^2$$

$$S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= aSb]{} a^2 S b^2 \xrightarrow[S ::= aSb]{} a^3 S b^3 \xrightarrow[S ::= \lambda]{} a^3 b^3$$

...

$$S \xrightarrow[S ::= aSb]{} aSb \xrightarrow[S ::= aSb]{} a^2 S b^2 \xrightarrow[S ::= aSb]{} a^3 S b^3 \xrightarrow[S ::= aSb]{} \dots \xrightarrow[S ::= \lambda]{} a^n b^n$$

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

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

► $G_{\text{bracket}_1} = (\{S\}, \{[,], <, >\}, P_{\text{bracket}_1}, S)$ mit

$$P_{\text{bracket}_1} = \{S ::= [S] \mid \langle S \rangle \mid SS \mid \lambda\}$$

$$\begin{aligned} S &\rightarrow SS \rightarrow S[S] \rightarrow \langle S \rangle [S] \rightarrow \langle SS \rangle [S] \rightarrow \\ &\langle [S]S \rangle [S] \rightarrow \langle []S \rangle [S] \rightarrow \langle []S \rangle [\langle S \rangle] \rightarrow \\ &\langle []S \rangle [\langle \rangle] \rightarrow \langle [][S] \rangle [\langle \rangle] \rightarrow \langle [][] \rangle [\langle \rangle] \end{aligned}$$

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 :

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

oder

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

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

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 $x \in I$.

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

Boolesche Ausdrücke

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

Syntaktische Beschreibung von PASCALchen

$\langle prog \rangle ::= \langle comp \rangle$

$\langle comp \rangle ::= \text{begin } \langle stmtlist \rangle \text{ end} \mid \text{begin end}$

$\langle stmtlist \rangle ::= \langle stmt \rangle \mid \langle stmt \rangle ; \langle stmtlist \rangle$

$\langle stmt \rangle ::= \langle comp \rangle \mid \langle assign \rangle \mid \langle while \rangle$

$\langle assign \rangle ::= \langle var \rangle := \langle expr \rangle$

$\langle while \rangle ::= \text{while } \langle var \rangle \neq \langle var \rangle \text{ do } \langle stmt \rangle$

$\langle expr \rangle ::= 0 \mid \text{succ}(\langle var \rangle) \mid \text{pred}(\langle var \rangle)$

⋮

$$\langle var \rangle ::= X \langle nat \rangle$$
$$\langle nat \rangle ::= | \langle one - nine \rangle \langle cipherseq \rangle$$
$$\langle cipherseq \rangle ::= \lambda | \langle cipher \rangle \langle cipherseq \rangle$$
$$\langle cipher \rangle ::= 0 | \langle one - nine \rangle$$
$$\langle one - nine \rangle ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$$

Kontextfreiheitslemma

Sei

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

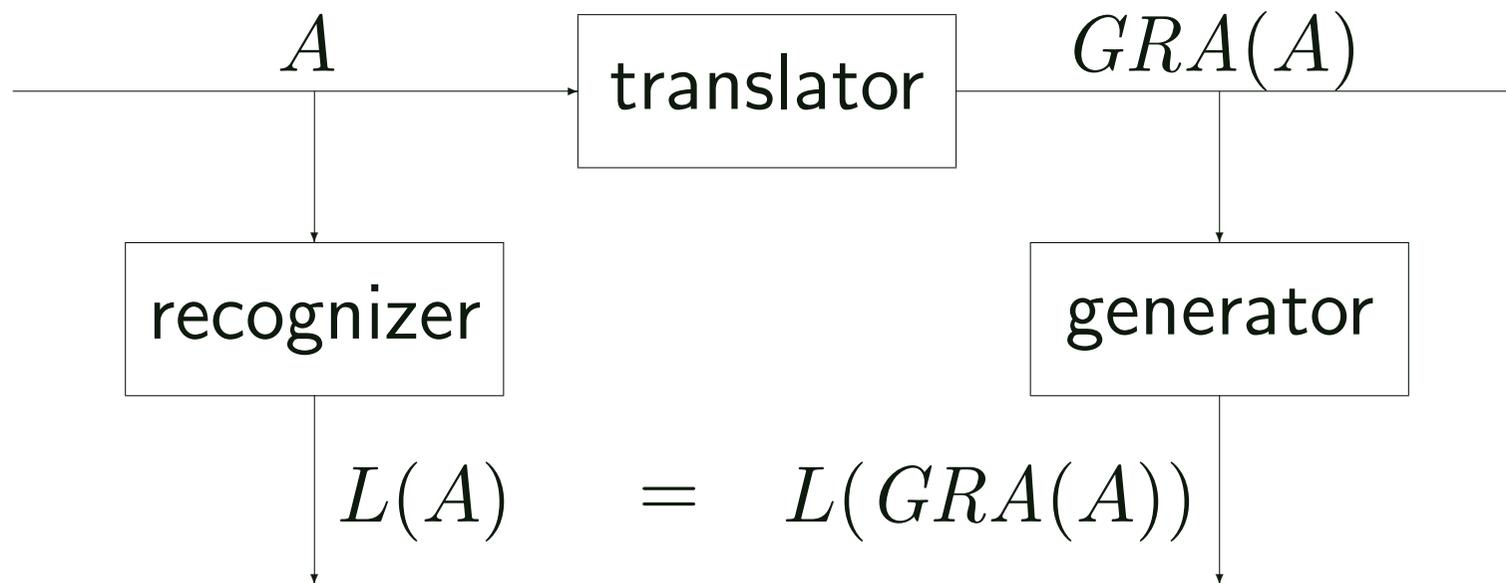
Dann gibt es Ableitungen $u_i \xrightarrow[n_i]{P} v_i$ ($i = 1, \dots, k$), so dass

$$v = v_1 \cdots v_k \text{ und } n = \sum_{i=1}^k n_i.$$

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

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