Theoretische Informatik I

Einheit 2.5

Grammatiken

1. Arbeitsweise
2. Klassifizierung
3. Beziehung zu Automaten

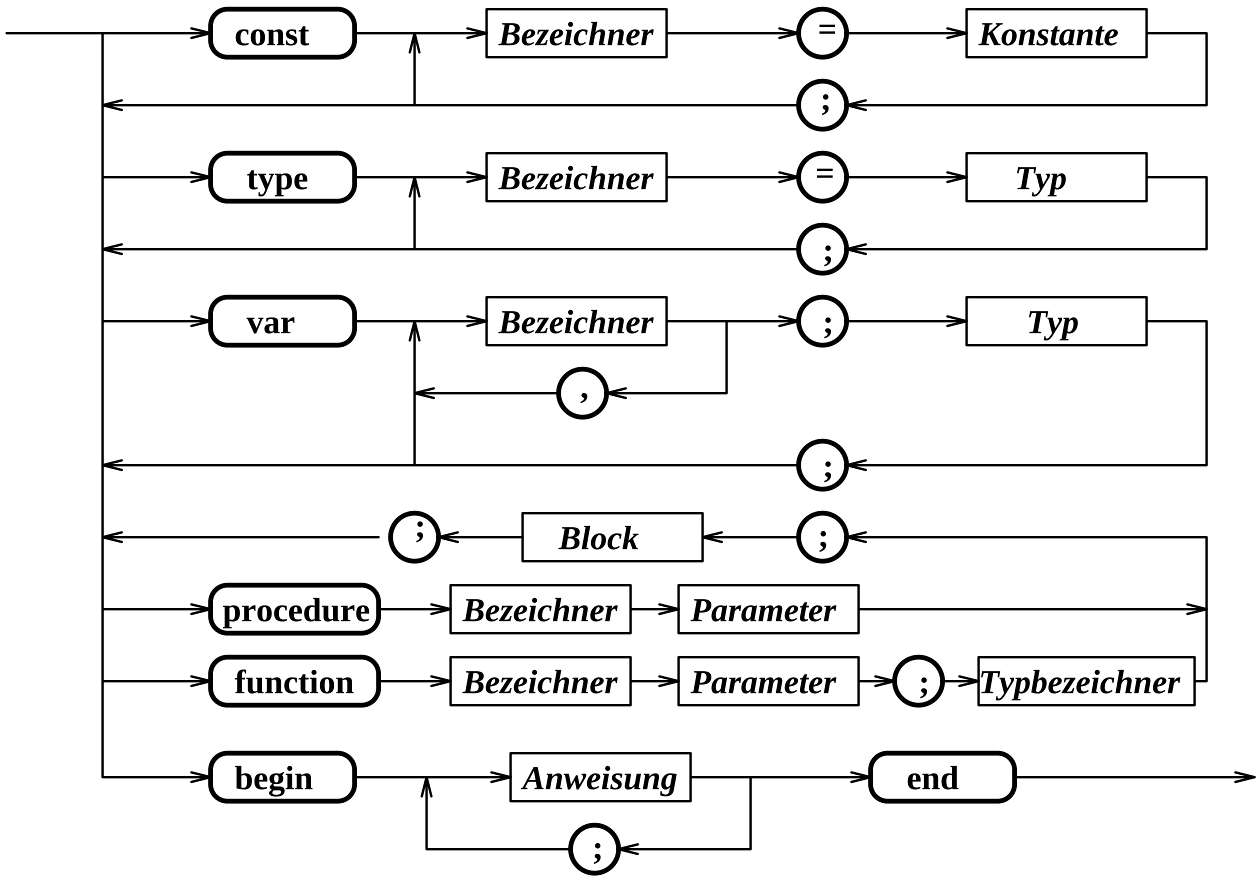
Beschreibung des Aufbaus von Sprachen

- **Mathematische Mengennotation**
  - Beschreibung durch Eigenschaften der Worte (Prädikate)
  - Extrem flexibel, nicht notwendig "berechenbar"
- **Endliche Automaten**
  - Beschreibung der Verarbeitung von Sprachen
  - Schwerpunkt ist Erkennen korrekter Worte
- **Reguläre Ausdrücke**
  - Beschreibung der Struktur der Worte
- **Grammatiken**
  - Beschreibung des Aufbaus von Sprachen durch Produktionsregeln
  - Auch für komplexere Strukturen
  - Gängig für die Beschreibung von Programmiersprachen

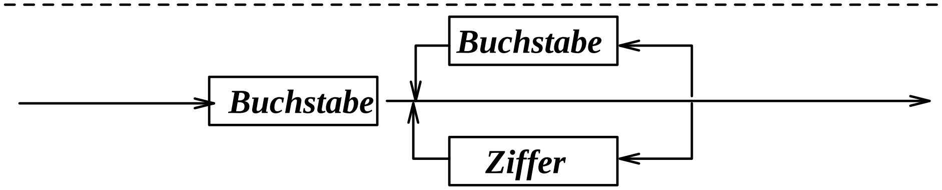
PASCAL Grammatik

Programm

Block



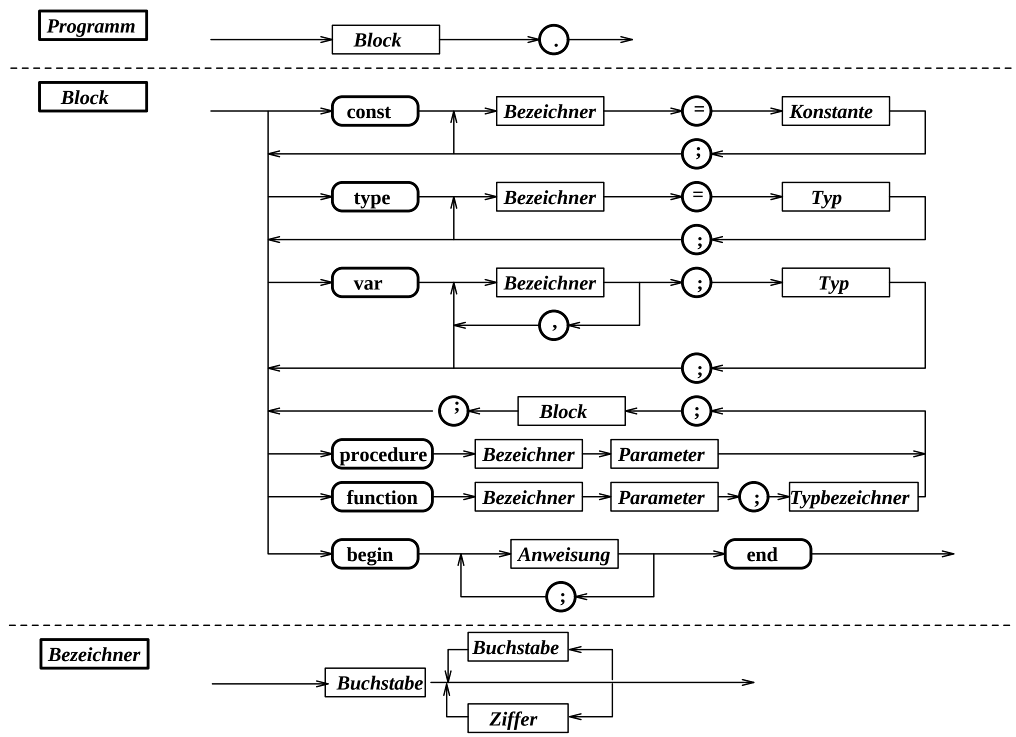
Bezeichner



Komponenten von Gramatiken

- **Alphabet der Sprache (Terminalsymbole)**
  - Symbole, aus denen die erzeugten Worte bestehen sollen
  - Bei Programmiersprachen meist ASCII-Symbole ohne Kontrollzeichen
- **Hilfsalphabet (Variablen)**
  - Beschreiben die syntaktischen Kategorien der Sprache
  - Bei PASCAL z.B. Programm, Block, Bezeichner, Anweisung, ...
  - Andere Bezeichnung: Nichtterminale Symbole
- **Regeln zur Erzeugung von Worten (Produktionen)**
  - Erklären wie syntaktischen Kategorien aufgebaut sind
  - Erklären Erzeugung von Worten der Sprache in den einzelnen Kategorien
  - z.B. "Ein Programm besteht aus einem Block gefolgt vom Symbol ."
- **Startsymbol**
  - Erklärt welche syntaktische Kategorie beschrieben werden soll

Grammatiken – mathematisch präzisiert



Eine Grammatik ist ein 4-Tupel $G = (V, T, P, S)$ mit

- T endliches **Terminalalphabet**
- V endliches **Hilfsalphabet** mit $V \cap T = \emptyset$
- $P \subseteq \Gamma^+ \times \Gamma^*$ endliche Menge der **Produktionen** (wobei $\Gamma = V \cup T$)

  Schreibweise für Produktionen: $\boldsymbol{l \longrightarrow r} \in P \equiv (l, r) \in P$
- $S \in V$ **Startsymbol**

Arbeitsweise: Produktion von Worten der Zielsprache

- $G_1 = (\{S\},\ \{0,1\},\ P,\ S)$ **mit** $P = \{S \to S1,\ S \to S0,\ S \to \epsilon\}$

  Erzeugung von Worten:

  $S \to \epsilon$

  $S \to S0 \to \mathbf{0}$

  $S \to S0 \to S10 \to S010 \to S0010 \to \mathbf{0010}$

  – Nur Worte über dem Terminalalphabet sind von Interesse

  – ϵ, 0, 0010 gehören zur erzeugten Sprache

  – S, $S0$, $S10$, $S010$, $S0010$ sind nur “Zwischenschritte”

- $G_2 = (\{S, A, B, C\},\ \{0,1\},\ P,\ S)$ **mit**
  $P = \{S \to B,\ S \to CA0,\ A \to BBB,\ B \to C1,\ B \to 0,\ CC1 \to \epsilon\}$

  Ableitungen:

  $S \longrightarrow B \longrightarrow \mathbf{0}$ $\surd$

  $S \longrightarrow B \longrightarrow C1$ Erfolglos, kein Wort der Zielsprache erreichbar

  $S \longrightarrow CA0 \longrightarrow CBBB0 \longrightarrow CC1BB0 \longrightarrow BB0 \longrightarrow 0B0 \longrightarrow \mathbf{000}$ $\surd$

Arbeitsweise von Grammatiken – präzisiert

- **Ableitungsrelation** $\longrightarrow\ \subseteq \Gamma^+ \times \Gamma^*$
  - $\boldsymbol{w} \longrightarrow \boldsymbol{z} \;\equiv\; \exists x, y \in \Gamma^*.\ \exists l \to r \in P.\ w = x\, l\, y \;\wedge\; z = x\, r\, y$

    Anwendung von Produktionen auf Worte

- **Erweiterte Ableitungsrelation** $\xrightarrow{*}\ \subseteq \Gamma^+ \times \Gamma^*$
  - $\boldsymbol{w} \xrightarrow{\mathbf{0}} \boldsymbol{z} \;\equiv\; w = z$
  - $\boldsymbol{w} \xrightarrow{\boldsymbol{n+1}} \boldsymbol{z} \;\equiv\; \exists u \in \Gamma^*.\ w \longrightarrow u \;\wedge\; u \xrightarrow{n} z$
  - $\boldsymbol{w} \xrightarrow{*} \boldsymbol{z} \;\equiv\; \exists n \in \mathbb{N}.\ w \xrightarrow{n} z$
  - Grammatik durch optionalen Index G ($\xrightarrow{*}_G$) spezifizierbar

- **Von G erzeugte Sprache**
  - Menge der Terminalworte, die aus S abgeleitet werden können

$$\boldsymbol{L(G)} \;\equiv\; \{w \in T^* \mid S \xrightarrow{*} w\}$$

GRAMMATIK FÜR $L = \{0^k 1^l \mid k \leq l\}$

- $G_3 = (\{S\},\ \{0,1\},\ P,\ S)$ **mit** $P = \{S \to S1,\ S \to 0S1,\ S \to \epsilon\}$

- **Zeige $L(G_3) = L$ per Induktion über Länge der Ableitung**
  - Ableitungen der Länge 0 liefern keine Terminalworte
  - Zeige: $\forall l \in \mathbb{N}.\ \forall w \in \{0,1\}^*.\ S \xrightarrow{l+1} w \Leftrightarrow (\exists k \leq l.\ w = 0^k 1^l)$

- **Basisfall**
  - $S \xrightarrow{1} w \;\Leftrightarrow\; (S \to w) \in P \;\Leftrightarrow\; w = \epsilon \;\Leftrightarrow\; \exists k \leq 0.\ w = 0^k 1^0$ ✓

- **Induktionsschritt**
  - Es gelte $\forall w \in \{0,1\}^*.\ S \xrightarrow{l+1} w \Leftrightarrow (\exists k \leq l.\ w = 0^k 1^l)$
  - $$\begin{aligned} & S \xrightarrow{l+2} v \\ \Leftrightarrow\ & S \to S1 \xrightarrow{l+1} v \ \vee\ S \to 0S1 \xrightarrow{l+1} v \\ \Leftrightarrow\ & \exists w \in \{0,1\}^*.\ \ S \xrightarrow{l+1} w \ \wedge\ (v = w1 \vee v = 0w1) \\ \Leftrightarrow\ & \exists w \in \{0,1\}^*. \exists k \leq l.\ \ w = 0^k 1^l \ \wedge\ (v = w1 \vee v = 0w1) && \text{(Annahme)} \\ \Leftrightarrow\ & \exists k \leq l.\ \ v = 0^k 1^{l+1} \ \vee\ v = 0^{k+1} 1^{l+1} \\ \Leftrightarrow\ & \exists k \leq (l+1).\ \ v = 0^k 1^{l+1} \end{aligned}$$ ✓

Klassifizierung von Grammatiken

- **allgemein (Typ 0):** keine Einschränkung an die Produktionen
- **kontextsensitiv (Typ 1)**
  - nur Regeln der Form $x\,A\,y \to x\,z\,y$ oder $S \to \epsilon$ $\quad (x, y, z \in \Gamma^*,\ A \in V, z \neq \epsilon)$

    ($S \to \epsilon$ nur erlaubt, wenn S nicht rechts in einer anderen Regel auftaucht)
- **expansiv**
  - nur Regeln der Form $x \to z$ mit $|x| \leq |z|$, oder $S \to \epsilon$ $\quad (x \in \Gamma^+, z \in (\Gamma - \{S\})^+)$
- **kontextfrei (Typ 2)**
  - nur Regeln der Form $A \to z$ $\quad (z \in \Gamma^*,\ A \in V)$
- **linear**
  - nur Regeln der Form $A \to \epsilon$ oder $A \to u\,B\,v$ $\quad (A, B \in V, u, v \in T^*)$
- **rechtslinear (Typ 3)**
  - nur Regeln der Form $A \to \epsilon$ oder $A \to v\,B$ $\quad (A, B \in V, v \in T)$
- **linkslinear**
  - nur Regeln der Form $A \to \epsilon$ oder $A \to B\,v$ $\quad (A, B \in V, v \in T)$

Beispiele für Grammatikklassen

- **kontextsensitiv**: Regeln $x\,A\,y \rightarrow x\,z\,y$ oder $S \rightarrow \epsilon$
- **expansiv**: Regeln $x \rightarrow z$ mit $|x| \leq |z|$, oder $S \rightarrow \epsilon$
- **kontextfrei**: Regeln $A \rightarrow z$
- **linear**: Regeln $A \rightarrow \epsilon$ oder $A \rightarrow u\,B\,v$
- **rechtslinear**: Regeln $A \rightarrow \epsilon$ oder $A \rightarrow v\,B$
- **linkslinear**: Regeln $A \rightarrow \epsilon$ oder $A \rightarrow B\,v$

- $\boldsymbol{G_1 = (\{S\},\ \{0,1\},\ P,\ S)}$ **mit** $\boldsymbol{P = \{S \rightarrow S1,\ S \rightarrow S0,\ S \rightarrow \epsilon\}}$
  – linkslinear, kontextfrei, nicht expansiv, nicht kontextsensitiv

- $\boldsymbol{G_2 = (\{S, A, B, C\},\ \{0,1\},\ P,\ S)}$ **mit**
  $\boldsymbol{P = \{S \rightarrow B,\ S \rightarrow CA0,\ A \rightarrow BBB,\ B \rightarrow C1,\ B \rightarrow 0,\ CC1 \rightarrow \epsilon\}}$
  – allgemein (keine anderen Bedingungen erfüllt)

- $\boldsymbol{G_3 = (\{S\},\ \{0,1\},\ P,\ S)}$ **mit** $\boldsymbol{P = \{S \rightarrow S1,\ S \rightarrow 0S1,\ S \rightarrow \epsilon\}}$
  – kontextfrei, nicht expansiv, nicht kontextsensitiv

Zusammenhang zwischen Grammatiken und Automaten

- **Automaten verarbeiten Eingabeworte**
  - Jedes Symbol wird in einem Schritt abgearbeitet
  - Symbol bestimmt, ob Automat im Zustand bleibt oder wechselt
- **Grammatiken erzeugen Worte**
  - Hilfssymbole werden im Endeffekt in Terminalworte umgewandelt
  - Nichtlineare Grammatiken erzeugen mehrere Symbole gleichzeitig
  - Ableitungen in rechts-/linkslinearen Grammatiken erzeugen pro Schritt ein Terminalsymbol und verwenden jeweils nur ein Hilfssymbol
- **Kann man umwandeln?**
  - Gibt es zu jedem DEA eine äquivalente rechtslineare Grammatik?
  - Gibt es zu jeder rechtslinearen Grammatik einen äquivalenten DEA?

Umwandlung von DEAs in Typ-3 Grammatiken

Für jeden DEA A gibt es eine Typ-3 Grammatik G mit $L(G) = L(A)$

- **Gegeben DEA $A = (Q, \Sigma, \delta, q_0, F)$**
  - Wandle Abarbeitung von Symbolen in Erzeugung durch Grammatik um
  - Setze $G := (Q, \Sigma, P, q_0)$ mit $P = \{q \rightarrow aq' \mid \delta(q,a) = q'\} \cup \{q \rightarrow \epsilon \mid q \in F\}$
  - G ist per Konstruktion rechtslinear, also vom Typ 3

- **Zeige $L(G) = L(A)$**

$$
\begin{aligned}
& w \in L(G) \\
\Leftrightarrow\ & S \xrightarrow{*} w = w_1..w_n \\
\Leftrightarrow\ & \exists q_1,..,q_n \in Q.\ q_0 \longrightarrow w_1 q_1 \longrightarrow w_1 w_2 q_2 \longrightarrow \dots w_1..w_n q_n \longrightarrow w_1..w_n \\
\Leftrightarrow\ & \exists q_1,..,q_n \in Q.\ \delta(q_0,w_1){=}q_1 \wedge \delta(q_1,w_2){=}q_2 \wedge \dots \delta(q_{n-1},w_n){=}q_n \wedge q_n \in F \\
\Leftrightarrow\ & \exists q_n \in F.\ \hat{\delta}(q_0,w) = q_n \\
\Leftrightarrow\ & w \in L(A)
\end{aligned}
$$

$\surd$

Umwandlung von Typ-3 Grammatiken in NEAs

Für jede Typ-3 Grammatik G gibt es einen NEA A mit $L(A) = L(G)$

- **Gegeben Grammatik $G = (V, T, P, S)$**
  - Wandle Erzeugung von Symbolen in Abarbeitung durch Automaten um
  - Setze $A := (V, T, \delta, S, F)$ mit $\delta(X, a) = \{X' \mid X \to aX' \in P\}$
    und $F = \{X \in V \mid X \to \epsilon \in P\}$

- **Zeige $L(A) = L(G)$**

$$
\begin{aligned}
& w = w_1..w_n \in L(A) \\
\Leftrightarrow\ & \exists X_n \in F.\ X_n \in \hat{\delta}(S, w) \\
\Leftrightarrow\ & \exists X_1, .., X_n \in V.\ X_1 \in \delta(S, w_1)\ \wedge\ \dots\ X_n \in \delta(X_{n-1}, w_n)\ \wedge\ X_n \in F \\
\Leftrightarrow\ & \exists X_1, .., X_n \in V.\ S \longrightarrow w_1 X_1 \longrightarrow w_1 w_2 X_2 \longrightarrow \dots w_1..w_n X_n \longrightarrow w_1..w_n \\
\Leftrightarrow\ & S \xrightarrow{*} w \\
\Leftrightarrow\ & w \in L(G)
\end{aligned}
$$

$\checkmark$

Sprachklassen

- **Typ-0 Sprachen**
  - Sprachen der Form $L = L(G)$ für eine beliebige Grammatik G
- **Typ-1 Sprachen (kontextsensitive Sprachen)**
  - Sprachen der Form $L = L(G)$ für eine kontextsensitive Grammatik G
  - L ist kontextsensitiv g.d.w. $L = L(G)$ für eine expansive Grammatik G
- **Typ-2 Sprachen (kontextfreie Sprachen)**
  - Sprachen der Form $L = L(G)$ für eine kontextfreie Grammatik G
- **lineare Sprachen**
  - Sprachen der Form $L = L(G)$ für eine lineare Grammatik G
- **Typ-3 Sprachen (reguläre Sprachen)**
  - Sprachen der Form $L = L(G)$ für eine rechtslineare Grammatik G
  - L ist regulär g.d.w. $L = L(G)$ für eine linkslineare Grammatik G
- $\boldsymbol{\mathcal{L}_i \equiv \{\, L \mid L \text{ ist Sprache vom Typ } i\}}$

Die Chomsky Hierarchie

$$\boldsymbol{\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0}$$

- **Wichtige Vertreter**
  - $\mathcal{L}_2 - \mathcal{L}_3$: $\{0^n 1^n \mid n \in \mathbb{N}\}$
  - $\mathcal{L}_1 - \mathcal{L}_2$: $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$
  - $\mathcal{L}_0 - \mathcal{L}_1$: $\{w_i \in \{0,1\}^* \mid$ Das Programm mit Codierung w_i hält bei Eingabe w_i $\}$
- **Zugehörige Automatenmodelle**
  - $\mathcal{L}_0$: Turingmaschine
  - $\mathcal{L}_1$: linear platzbeschränkte nichtdeterministische Turingmaschine
  - $\mathcal{L}_2$: nichtdeterministischer endlicher Automat mit Kellerspeicher
  - $\mathcal{L}_3$: endlicher Automat

Mehr in zukünftigen Vorlesungen

Errata

In der Erstversion dieser Folien gab es leider viele Tippfehler

- Folie 7, drittletzte Zeile war $k \geq l$ statt $k \leq l$
- Folie 8: Rechts-/Linkslineare Grammatiken
  – Ursprünglich: Regeln der Form $A \rightarrow \epsilon$ oder $A \rightarrow v\, B$ $(A, B \in V, v \in T^*)$
  Diese Definition ist zwar äquivalent zu der Bedingung $v \in T$,
  (Ersetze die Regel $A \rightarrow v_1..v_k\, B$ durch $A \rightarrow v_1 B_1$, $B_1 \rightarrow v_2 B_2$, $B_{k-1} \rightarrow v_k B$, B_i neu)
  ist aber komplizierter und wird in der Literatur nicht benutzt
- Folie 9, G_1 und G_3 ist nicht expansiv oder kontextsensitiv
  Ich habe das falsch ausgedrückt. Jede kontextfreie Sprache ist auch Sprache einer expansiven oder kontextsensitiven Grammatik, aber wegen der Randbedingungen für S ist nicht jede kontextfreie Grammatik auch kontextsensitiv oder expansiv
- Folie 11 und 12
  – Anstatt $\hat{\delta}$ stand ursprünglich nur δ
- Folie 12
  – Der Korrektheitsbeweis $L(A) = L(G)$ war ziemlich verhunzt
  – Ich hatte mit paste & copy einen Korrektheitsbeweis für einen DEA erzeugt, obwohl die Grammatik in einen NEA umgewandelt wird.