Theoretische Informatik I

Einheit 2.4

Grammatiken

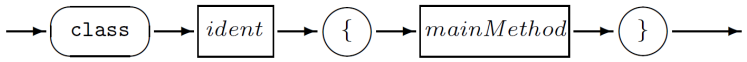
1. Arbeitsweise
2. Klassifizierung
3. Beziehung zu Automaten

Beschreibungsformen für Sprachen

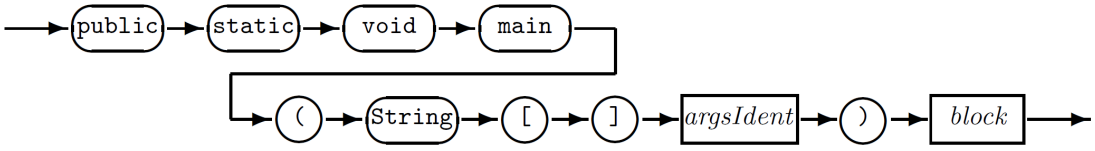
- **Mathematische Mengennotation**
  - Prädikate beschreiben Eigenschaften der Wörter
  - Extrem flexibel, nicht notwendig “berechenbar”
- **Endliche Automaten**
  - Beschreibung der Verarbeitung von Sprachen
  - Schwerpunkt ist Erkennen korrekter Wörter
- **Reguläre Ausdrücke**
  - Beschreibung der Struktur der Sprache
- **Grammatiken**
  - Produktionsregeln beschreiben Aufbau der Wörter
  - Auch für komplexere Strukturen als reguläre Sprachen
  - Gängig für die Beschreibung von Programmiersprachen

Beispiel: Auszug der Grammatik von JAVA

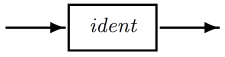
program



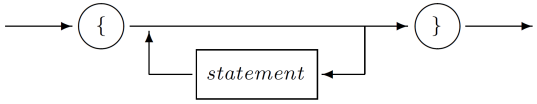
mainMethod



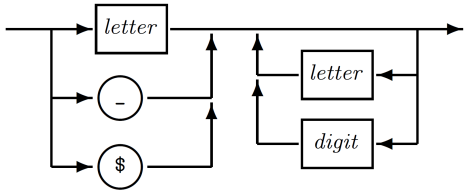
argsIdent



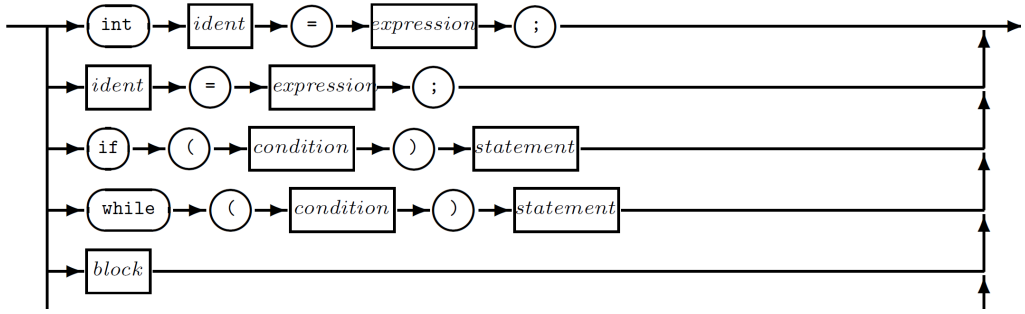
block



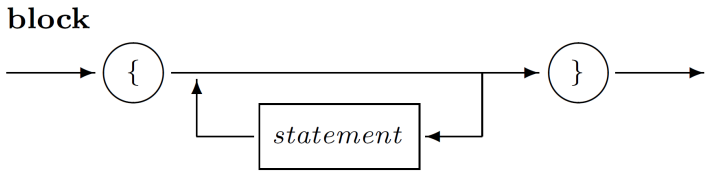
ident



statement

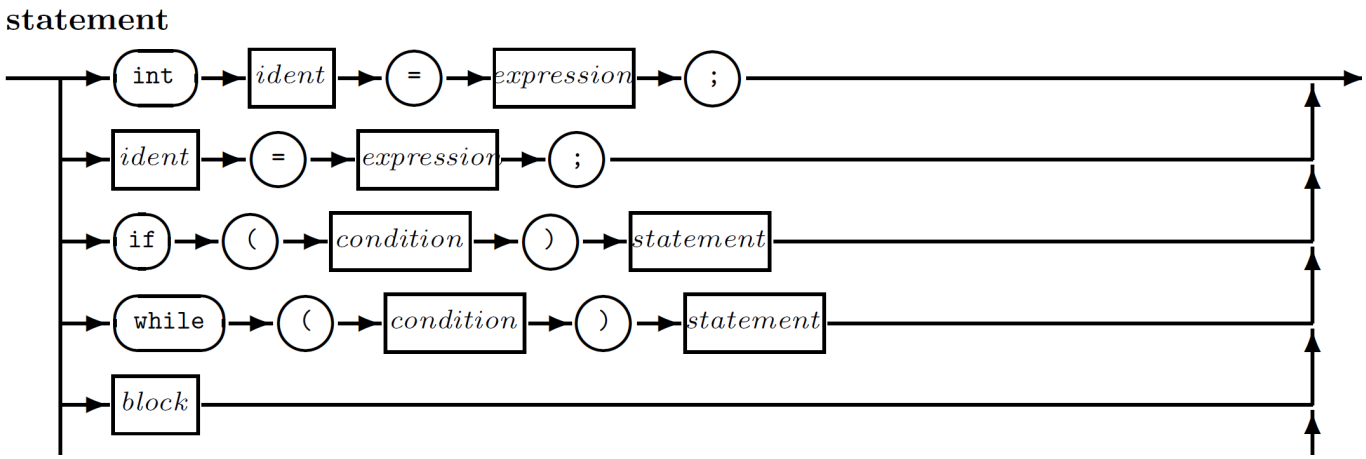


Komponenten von Grammatiken



- **Terminalsymbole: Alphabet der Sprache**
  - Symbole, aus denen die erzeugten Wörter bestehen sollen
  - Bei Programmiersprachen meist ASCII-Symbole ohne Kontrollzeichen
- **Variablen: Hilfsalphabet für Verarbeitung**
  - Beschreiben die syntaktischen Kategorien der Sprache
  - Bei JAVA z.B. *program, mainMethod, argsIdent, block, statement* ...
  - Andere Bezeichnung: Nichtterminale Symbole
- **Produktionen: Regeln zur Erzeugung von Wörtern**
  - Erklären wie syntaktischen Kategorien aufgebaut sind
  - Erklären Erzeugung von Wörtern der Sprache in einzelnen Kategorien
  - z.B. "Ein Block beginnt mit { gefolgt von einem Statement und }"
- **Startsymbol**
  - Erklärt welche syntaktische Kategorie beschrieben werden soll

Grammatiken – mathematisch präzisiert



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

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

  Schreibweise für Produktionen: $l \rightarrow r \in P$ statt $(l, r) \in P$
- $S \in V$ **Startsymbol**

Arbeitsweise: Produktion von Wörtern der Zielsprache

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

  Erzeugung von Wörtern:

  $S \to \epsilon$

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

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

  – Nur Wörter 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:

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

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

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

Arbeitsweise von Grammatiken – präzisiert

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

    Anwendung von Produktionen auf Wörter

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

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

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

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 Terminalwörter
  - Zeige: $\forall l \in \mathbb{N}.\ \forall w \in \{0,1\}^*.\ S \xrightarrow{l+1} w \Leftrightarrow (\exists k \leq l.\ w = 0^k 1^l)$
- **Induktionsanfang** $l{=}0$**:**
  - $S \xrightarrow{1} w \ \Leftrightarrow\ (S \to w) \in P \ \Leftrightarrow\ w = \epsilon \ \Leftrightarrow\ \exists k \leq 0.\ w = 0^k 1^0$ ✓
- **Induktionsannnahme: Für ein beliebiges** $l' \in \mathbb{N}$ **sei gezeigt**
  - $\forall w' \in \{0,1\}^*.\ S \xrightarrow{l'+1} w' \Leftrightarrow (\exists k \leq l'.\ w' = 0^k 1^{l'})$
- **Induktionsschritt: Es sei** $l = l'{+}1$**. Dann gilt für alle** $w \in \{0,1\}^*$

$$
\begin{aligned}
& S \xrightarrow{l+1} w \\
\Leftrightarrow\ & S \to S1 \xrightarrow{l} w \ \vee\ S \to 0S1 \xrightarrow{l} w \\
\Leftrightarrow\ & \exists w' \in \{0,1\}^*.\ \ S \xrightarrow{l'+1} w' \ \wedge\ (w = w'1 \vee w = 0w'1) && (l = l'{+}1) \\
\Leftrightarrow\ & \exists w' \in \{0,1\}^*. \exists k \leq l'.\ \ w' = 0^k 1^{l'} \ \wedge\ (w = w'1 \vee w = 0w'1) && \text{(Annahme)} \\
\Leftrightarrow\ & \exists k \leq l'.\ \ w = 0^k 1^{l'+1} \ \vee\ w = 0^{k+1} 1^{l'+1} \\
\Leftrightarrow\ & \exists k \leq (l'{+}1).\ \ w = 0^k 1^{l'+1} \ \Leftrightarrow\ \exists k \leq l.\ \ w = 0^k 1^l
\end{aligned}
$$

✓

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, $\Gamma = V \cup T$)
- **expansiv**
  - nur Regeln der Form $x \to z$ mit $|x| \leq |z|$, oder $S \to \epsilon$ $\quad (x, z \in \Gamma^+)$

    ($S \to \epsilon$ nur erlaubt, wenn S nicht rechts in einer anderen Regel auftaucht)
- **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 $A \to a\,B$ $\quad (A, B \in V,\ a \in T)$

    Manche Bücher: 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 $A \to B\,a$ $\quad (A, B \in V,\ a \in T)$

Beispiele für Grammatikklassen

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

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

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

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

- $G_4 = (\{S, A, B, C\},\ \{a, b, c\},\ P,\ S)$ **mit** $P = \{S \to aCBC,$
  $S \to aBC,\ CB \to BC,\ aB \to ab,\ bB \to bb,\ bC \to bc,\ cC \to cc\}$
  – expansiv, nicht kontextfrei, nicht kontextsensitiv

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

$$\mathcal{L}_i \equiv \{\, L \mid L \text{ ist Sprache vom Typ } i \}$$

Typ-3 Sprachen vs. reguläre Sprachen

Wie hängen Grammatiken und Automaten zusammen?

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

$\mapsto$ $\mathcal{L}_3 = \{ L \mid L \text{ ist regulär} \}$

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 $P = \{q \to aq' \mid \delta(q, a) = q'\} \cup \{q \to \epsilon \mid q \in F\}$

    Bildlich ausgedrückt: $P = \{q \to aq' \mid \text{das Diagramm für } \delta \text{ enthält } q \xrightarrow{a} q'\} \cup \{q \to \epsilon \mid q \in F\}$
  - G ist per Konstruktion rechtslinear, also vom Typ 3
- **Zeige $L(G) = L(A)$**

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

✓

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 NEA um
  - Setze $A := (V, T, \delta, S, F)$ mit $\delta(X, a) = \{X' \mid X \to aX' \in P\}$
    und $F = \{X \in V \mid X \to \epsilon \in P\}$

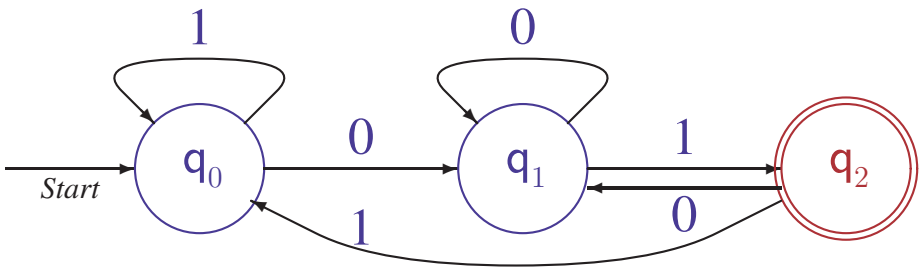
- **Zeige $L(A) = L(G)$**

$$
\begin{aligned}
& w = w_1..w_n \in L(A) \\
\Leftrightarrow\ & \exists X_n \in F. \quad (S, w_1..w_n) \vdash^* (X_n, \epsilon) \\
\Leftrightarrow\ & \exists X_1, .., X_n \in V. \ (S, w_1..w_n) \vdash (X_1, w_2..w_n) \vdash ... \vdash (X_n, \epsilon) \wedge X_n \in F \\
\Leftrightarrow\ & \exists X_1, .., X_n \in V. \ S \longrightarrow w_1 X_1 \longrightarrow ... \longrightarrow w_1..w_n X_n \longrightarrow w_1..w_n \\
\Leftrightarrow\ & S \xrightarrow{*} w \\
\Leftrightarrow\ & w \in L(G)
\end{aligned}
$$

✓

Umwandlungen am Beispiel

- **Konvertiere DEA für** `(0+1)*01`
  - $A = (\{ q_0, q_1, q_2 \}, \{0,1\}, \delta, q_0, \{q_2\})$ mit



- **Erzeugte Grammatik**
  - $G = (\{ q_0, q_1, q_2 \}, \{0,1\}, P, q_0)$ mit

    $P = \{q_0 \to 1q_0,\ q_0 \to 0q_1,\ q_1 \to 1q_2,\ q_1 \to 0q_1,\ q_2 \to 1q_0,\ q_2 \to 0q_1,\ q_2 \to \epsilon\}$

- **Umwandlung von G in einen NEA**
  - Transformation erzeugt ursprünglichen Automaten

Die Chomsky Hierarchie

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

- **Wichtige Vertreter der Klassen**
  - $\mathcal{L}_2 - \mathcal{L}_3$: $\{0^n 1^n \mid n \in \mathbb{N}\}$
  - $\mathcal{L}_1 - \mathcal{L}_2$: $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$
  - $\mathcal{L}_0 - \mathcal{L}_1$: $\{w \in \{0,1\}^* \mid$ *w ist Code eines Programms, das bei Eingabe w anhält* $\}$
- **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