Tutorium
Automaten und Formale Sprachen
Grammatiken

Christopher Blöcker, B. Sc.
inf9900@fh-wedel.de

SS 2012

Beispiel: Bezeichner in PASCAL (BNF)

<BEZEICHNER> ::= <BUCHSTABE> (<BUCHSTABE> | <ZIFFER>)*
<BUCHSTABE> ::= 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N'
| 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z
<ZIFFER> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

Definition : Grammatik

Eine Grammatik ist ein 4-Tupel.

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S})$$

mit

- $\mathcal{N}$ Endliche, nicht-leere Menge der Nicht-Terminale
- $\mathcal{T}$ Endliche, nicht-leere Menge der Terminale
  $\mathcal{N} \cap \mathcal{T} = \emptyset$
- $\mathcal{R}$ Menge der Produktionen mit
  $\mathcal{R} \subseteq (\mathcal{N} \cup \mathcal{T})^+ \times (\mathcal{N} \cup \mathcal{T})^*$
- $\mathcal{S}$ Das Startsymbol mit $\mathcal{S} \in \mathcal{N}$

$\mathcal{L} = \{w \in \Sigma^* \mid w = a^n b^n, n \in \mathbb{N}\}$

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S})$$

mit

$$\begin{array}{lcl} \mathcal{N} & = & \{S\} \\ \mathcal{T} & = & \{a, b\} \\ \mathcal{R} & : & S \to \epsilon \\ & & S \to aSb \\ \mathcal{S} & = & S \end{array}$$

Anmerkung

Regeln mit gleicher linker Seite können zusammengefasst werden

$$S \to \epsilon,\ S \to aSb \ \Rightarrow\ S \to \epsilon \mid aSb.$$

Die Chomsky-Hierarchie

Noam Chomsky hat den Begriff der Chomsky-Hierarchie geprägt und die folgenden 4 Sprachklassen definiert:

- Typ-3 : Reguläre Sprachen $\mathcal{L}_3$
- Typ-2 : Kontext-Freie Sprachen $\mathcal{L}_2$
- Typ-1 : Kontext-Sensitive Sprachen $\mathcal{L}_1$
- Typ-0 : Rekursiv Aufzählbare Sprachen $\mathcal{L}_0$

Es gilt:

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

Jede dieser Sprachklassen lässt sich durch Grammatiken des jeweiligen Types erzeugen. Dabei unterliegen die Produktionen gewissen Restriktionen.

Anmerkung

Es gibt (überabzählbar viele) Sprachen, die **nicht einmal** rekursiv aufzählbar sind!

Typ-3 Grammatik (Reguläre Sprachen)

Sei

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S}).$$

$\mathcal{G}$ ist vom Typ-3, wenn die Produktionen folgende Form haben:

$$\begin{aligned} A &\rightarrow \epsilon \\ A &\rightarrow a \\ A &\rightarrow aB \end{aligned}$$

mit

- $A, B \in \mathcal{N}$
- $a \in \mathcal{T}$

Die hier gezeigte Grammatik ist rechtslinear.
(Der Ableitungsbaum wächst nach rechts.)

Typ-3 Grammatik (Reguläre Sprachen)

Zu jeder rechtslinearen Typ-3-Grammatik gibt es eine linkslineare Grammatik, die dieselbe Sprache erzeugt.
Rechts- und linkslineare Regeln dürfen nicht vermischt werden, sonst können auch Sprachen erzeugt werden, die nicht vom Typ-3 sind.

$\mathcal{L} = \{w \in \Sigma^* \mid w = a^n b^n, n \in \mathbb{N}\}$

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S})$$

mit

$$\begin{aligned} \mathcal{N} &= \{S\} \\ \mathcal{T} &= \{a, b\} \\ \mathcal{R} &: S \to aA \mid \epsilon \\ &\quad\; A \to Sb \\ \mathcal{S} &= S \end{aligned}$$

Typ-2 Grammatik (Kontext-freie Sprachen)

Sei

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S}).$$

$\mathcal{G}$ ist vom Typ-2, wenn die Produktionen folgende Form haben:

$$A \quad \rightarrow \quad w$$

mit

- $A \in \mathcal{N}$
- $w \in (\mathcal{N} \cup \mathcal{T})^*$

Typ-1 Grammatik (Kontext-sensitive Sprachen)

Sei

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S}).$$

$\mathcal{G}$ ist vom Typ-1, wenn die Produktionen folgende Form haben:

$$uAv \quad \rightarrow \quad u\phi v$$

mit

- $A \in \mathcal{N}$
- $u, v \in (\mathcal{N} \cup \mathcal{T})^*$
- $\phi \in (\mathcal{N} \cup \mathcal{T})^+$

Typ-0 Grammatik (Rekursiv aufzählbare Sprachen)

Sei

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S}).$$

$\mathcal{G}$ ist vom Typ-0, wenn die Produktionen folgende Form haben:

$$u \quad \rightarrow \quad v$$

mit

- $u \in (\mathcal{N} \cup \mathcal{T})^+$
- $v \in (\mathcal{N} \cup \mathcal{T})^*$

Erzeugte Sprache

Sei

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S}).$$

Die von $\mathcal{G}$ erzeugte Sprache ist

$$\mathcal{L}(\mathcal{G}) = \{w \in \Sigma^* \mid \mathcal{S} \overset{*}{\rightarrow} w\}.$$

"$\overset{*}{\rightarrow}$" bezeichnet eine Folge von Ableitungsschritten.

Definition : Ableitungsschritt

Ein Ableitungsschritt ist das Anwenden einer Produktion auf eine Satzform, z.B.:

- $r_1 : A \rightarrow aBc$
- $w = xyz\underline{A}zyx$

$w \overset{r_1}{\rightarrow} w'$ mit $w' = xyz\underline{aBc}zyx$.

Definition: Mehrdeutigkeit

Eine Grammatik ist mehrdeutig, wenn es ein Wort gibt, für das unterschiedliche Ableitungsbäume existieren.

Ob eine Grammatik mehrdeutig ist, ist **im Allgemeinen nicht entscheidbar**. Es existiert also kein Algorithmus, der zu jeder Grammatik entscheiden kann, ob sie mehrdeutig ist.

Dennoch ist es unter Vorgabe einer konkreten Grammatik möglich, zu entschieden, ob diese mehrdeutig ist oder nicht.

Nicht-Determinismus in Grammatiken

Grammatiken sind nicht-deterministisch, da die angewendeten Produktionsregeln zufällig ausgewählt werden.

Soll eine Grammatik deterministisch sein, so darf es immer nur eine Produktionsregel zur Auswahl geben.

Chomsky Normalform (CNF)

Eine Grammatik $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S})$ ist in Chomsky Normalform, wenn alle Produktionen folgende Form haben:

$$\begin{aligned} A &\rightarrow BC \\ A &\rightarrow a \end{aligned}$$

Jede Grammatik vom Typ 2 kann in die Chomsky Normalform überführt werden. Dasselbe gilt wegen $\mathcal{L}_3 \subsetneq \mathcal{L}_2$ auch für Grammatiken vom Typ 3.

Liegt eine Grammatik in CNF vor, so kann mit Hilfe des CYK-Algorithmus das Wortproblem relativ leicht entschieden werden.

Das Wortproblem

Das Wortproblem bezeichnet die Aufgabe, von einem gegebenen Wort w zu entscheiden, ob es zu einer Sprache L gehört oder nicht, also

$$w \stackrel{?}{\in} L.$$

Das Wortproblem ist für Sprachen der Typen 3, 2, 1 entscheidbar, für Sprachen vom Typ 0 im Allgemeinen aber nicht.

Vereinbarung

Im folgenden soll gelten:

Großbuchstaben $A, B, \ldots$ bezeichnen Nichtterminale mit $A, B, \ldots \in \mathcal{N}$.

Die Kleinbuchstaben a, b, c bezeichnen Terminale mit $a, b, c \in \mathcal{T}$.

Die Kleinbuchstaben u, v bezeichnen Satzformen mit $u, v \in (\mathcal{N} \cup \mathcal{T})^*$

Hinweis

Bei der Transformation einer Grammatik nach CNF sollte in der hier beschriebenen Reihenfolge vorgegangen werden.

1. Elimination von ϵ-Produktionen
2. Elimination von Einheitsproduktionen
3. Transformation der Produktionen in korrekte Form

Da bei der Elimination von ϵ-Produktionen Einheitsproduktionen entstehen können, sollte dieser Schritt durchgeführt werden, bevor die Einheitsproduktionen eliminiert werden.

Außerdem: Grammatiken in CNF können nicht das leere Wort erzeugen.

Elimination von ϵ-Produktionen

- Auswählen einer ϵ-Produktion
- Anwenden der Produktion in allen anderen Produktionen und Aufnehmen der entstehenden Regeln in die Grammatik
- Löschen der ausgewählten ϵ-Produktion

$$\mathcal{R} \leftarrow [\mathcal{R} \cup \{A \rightarrow uv \mid (A, uBv), (B, \epsilon) \in \mathcal{R}\}] \setminus \{(B, \epsilon)\}$$

Es können auch Regeln existieren, die mehrfache Vorkommen von Nichtterminalen enthalten, die nach ϵ abgeleitet werden können. Dann muss jede mögliche Kombination von ϵ-Ableitungen durchgeführt werden. Aus

$$\begin{aligned} A &\rightarrow BCDCE \mid CC \\ C &\rightarrow \epsilon \end{aligned}$$

wird dann

$$A \rightarrow BCDCE \mid BDCE \mid BCDE \mid BDE \mid CC \mid C \mid \epsilon.$$

Vorgriff auf Compilerbau: Ein Nichtterminal A mit $A \overset{*}{\rightarrow} \epsilon$ wird als „nullable“ bezeichnet.

Elimination von Einheitsproduktionen

- Auswählen einer Einheitsproduktion
- Einsetzen aller möglichen rechten Seiten und Aufnehmen der entstehenden Regeln in die Grammatik
- Löschen der ausgewählten Einheitsproduktion

$$\mathcal{R} \leftarrow [\mathcal{R} \cup \{A \rightarrow u \mid (A, B), (B, u) \in \mathcal{R}\}] \setminus \{(A, B)\}$$

Aus

$$\begin{aligned} A &\rightarrow B \\ B &\rightarrow CD \mid EFG \mid a \mid bc \end{aligned}$$

wird dann

$$A \rightarrow CD \mid EFG \mid a \mid bc.$$

Transformation von Produktionen

- Produktionen der Form $A \to a$ müssen nicht verändert werden
- Für Produktionen der Form $A \to bc$ werden zwei neue Nichtterminale eingefügt mit

$$\begin{aligned} A &\to BC \\ B &\to b \\ C &\to c \end{aligned}$$

- Für Produktionen der Form $A \to uv$ mit $|u| = 1, |v| > 1$ wird eine neue Variable eingefügt mit
  1. Wenn $u \in \mathcal{N}$

$$\begin{aligned} A &\to uB \\ B &\to v \end{aligned}$$

  2. Wenn $u \in \mathcal{T}$

$$\begin{aligned} A &\to UB \\ B &\to v \\ U &\to u \end{aligned}$$

  Die Regel $A \to uv$ wird dann gelöscht.

Vereinbarung

Im Folgenden bezeichne

- $\mathcal{L}_{i,D}$ die Menge der Deterministischen Typ-i-Sprachen
- $\mathcal{L}_{i,N}$ die Menge der Nicht-Deterministischen Typ-i-Sprachen

Reguläre Sprachen (Typ-3)

$$\mathcal{L}_{3,D} = \mathcal{L}_{3,N}$$

Dies ist leicht einzusehen, da zu jedem **NEA** ein äquivalenter **DEA** angegeben werden kann.

Kontext-freie Sprachen (Typ-2)

$$\mathcal{L}_{2,D} \subsetneq \mathcal{L}_{2,N}$$

NKA sind mächtiger als **DKA**.

Kontext-Sensitive Sprachen (Typ-1)

$$\mathcal{L}_{1,D} \overset{?}{\subseteq} \mathcal{L}_{1,N}$$

Es ist bisher nicht bekannt, ob die Inclusion echt oder unecht ist.

Rekursiv Aufzählbare Sprachen (Typ-0)

$$\mathcal{L}_{0,D} = \mathcal{L}_{0,N}$$

NDTM können durch **DTM** simuliert werden, was im Allgemeinen exponentielle Zeit benötigt.

Rekursiv Aufzählbare Sprachen $\mathcal{L}_{re}$

Sei $\mathcal{L} \subseteq \Sigma^*, x \in \mathcal{L}, \mathcal{M}$ eine **TM** mit $\mathcal{L}_{\mathcal{M}} = \mathcal{L}$.
$\mathcal{L}$ ist Rekursiv Aufzählbar, wenn gilt

$$\mathcal{M}(x) = \begin{cases} 1 & \text{wenn } x \in \mathcal{L} \\ \nearrow & \text{sonst} \end{cases}$$

$\mathcal{M}$ **erkennt** $\mathcal{L}$.

Rekursive Sprachen $\mathcal{L}_r$

Sei $\mathcal{L} \subseteq \Sigma^*, x \in \mathcal{L}, \mathcal{M}$ eine **TM** mit $\mathcal{L}_{\mathcal{M}} = \mathcal{L}$.
$\mathcal{L}$ ist Rekursiv wenn gilt

$$\mathcal{M}(x) = \begin{cases} 1 & \text{wenn } x \in \mathcal{L} \\ 0 & \text{wenn } x \notin \mathcal{L} \end{cases}$$

$\mathcal{M}$ **entscheidet** $\mathcal{L}$.