Beispiel (Anwendung von Satz 95)

Korollar 96

Die Sprache

$$L = \{0^i 1^j 2^k;\ i = j \text{ oder } j = k\} \subset \{0, 1, 2\}^*$$

ist kontextfrei, aber nicht deterministisch kontextfrei ($\in$ CFL $\setminus$ DCFL).

Beweis:

L *wird erzeugt z.B. von der CFG mit den Produktionen*

$$S \to A \mid AB \mid B \mid C \mid CD \mid D \mid \epsilon$$

$$A \to 01 \mid 0A1 \qquad B \to 2 \mid 2B$$

$$C \to 0 \mid 0C \qquad D \to 12 \mid 1D2$$

Beispiel (Anwendung von Satz 95)

Korollar 96

Die Sprache

$$L = \{0^i 1^j 2^k;\ i = j \text{ oder } j = k\} \subset \{0,1,2\}^*$$

ist kontextfrei, aber nicht deterministisch kontextfrei ($\in$ CFL $\setminus$ DCFL).

Beweis:

Wäre $L \in$ DCFL, dann auch

$$L' := \bar{L} \cap 0^* 1^* 2^* = \{0^i 1^j 2^k;\ i \neq j \text{ und } j \neq k\}.$$

Mit Hilfe von Ogden's Lemma sieht man aber leicht, dass dies nicht der Fall ist.

Beispiel 97 (Grammatik für Arithmetische Ausdrücke)

Regeln:

$$
\begin{aligned}
S &\to A\\
A &\to E \mid A + E \mid A - E\\
E &\to P \mid E * P \mid E/P\\
P &\to (A) \mid a
\end{aligned}
$$

Wir betrachten für das Parsen einen bottom-up-Ansatz, wobei die Reduktionen von links nach rechts angewendet werden.

Beispiel 97 (Grammatik für Arithmetische Ausdrücke)

Dabei können sich bei naivem Vorgehen allerdings Sackgassen ergeben, die dann aufgrund des Backtracking zu einem ineffizienten Algorithmus führen:

Regeln:

$$S \to A$$
$$A \to E \mid A + E \mid A - E$$
$$E \to P \mid E * P \mid E/P$$
$$P \to (A) \mid a$$

Ableitung:

$$\begin{array}{ccccc} a & + & a & * & a \\ P & + & a & * & a \\ E & + & a & * & a \\ A & + & a & * & a \\ S & + & a & * & a \end{array}$$

Sackgasse!

Beispiel 97 (Grammatik für Arithmetische Ausdrücke)

... oder auch

Regeln:

$$S \to A$$
$$A \to E \mid A + E \mid A - E$$
$$E \to P \mid E * P \mid E/P$$
$$P \to (A) \mid a$$

Ableitung:

$$\begin{array}{ccccc}
a & + & a & * & a \\
P & + & a & * & a \\
E & + & a & * & a \\
A & + & a & * & a \\
A & + & P & * & a \\
A & + & E & * & a \\
 & A & & * & a \\
 & A & & * & P \\
 & A & & * & E
\end{array}$$

Sackgasse!

Beispiel 97 (Grammatik für Arithmetische Ausdrücke)

Zur Behebung des Problems führen wir für jede Ableitungsregel (besser hier: Reduktionsregel) einen Lookahead (der Länge k) ein (in unserem Beispiel $k = 1$) und legen fest, dass eine Ableitungsregel nur dann angewendet werden darf, wenn die nächsten k Zeichen mit den erlaubten Lookaheads übereinstimmen.

Beispiel 97 (Grammatik für Arithmetische Ausdrücke)

Produktion	Lookaheads (der Länge 1)
$S \to A$	ϵ
$A \to E$	$+, -,), \epsilon$
$A \to A + E$	$+, -,), \epsilon$
$A \to A - E$	$+, -,), \epsilon$
$E \to P$	beliebig
$E \to E * P$	beliebig
$E \to E/P$	beliebig
$P \to (A)$	beliebig
$P \to a$	beliebig

Beispiel 97 (Grammatik für Arithmetische Ausdrücke)

Damit ergibt sich

Produktion	Lookaheads
$S \to A$	ϵ
$A \to E$	$+, -,), \epsilon$
$A \to A + E$	$+, -,), \epsilon$
$A \to A - E$	$+, -,), \epsilon$
$E \to P$	beliebig
$E \to E * P$	beliebig
$E \to E/P$	beliebig
$P \to (A)$	beliebig
$P \to a$	beliebig

Ableitung:

$$
\begin{array}{ccccc}
a & + & a & * & a \\
P & + & a & * & a \\
E & + & a & * & a \\
A & + & a & * & a \\
A & + & P & * & a \\
A & + & E & * & a \\
A & + & E & * & P \\
A & + & & E & \\
 & & A & & \\
 & & S & &
\end{array}
$$

Definition 98

Eine kontextfreie Grammatik ist eine $LR(k)$-Grammatik, wenn man durch Lookaheads der Länge k erreichen kann, dass bei einer Reduktion von links nach rechts in jedem Schritt höchstens eine Produktion/Reduktion anwendbar ist.

Korollar 99

Jede kontextfreie Sprache, für die es eine $LR(k)$-Grammatik gibt, ist deterministisch kontextfrei.

Bemerkung:
Es gibt eine (im allgemeinen nicht effiziente) Konstruktion, um aus einer $LR(k)$-Grammatik, $k > 1$, eine äquivalente $LR(1)$-Grammatik zu machen.

Korollar 100

Die folgenden Klassen von Sprachen sind gleich:

- *die Klasse DCFL,*
- *die Klasse der $LR(1)$-Sprachen.*

4.11 $LL(k)$-Grammatiken

Beispiel 101 (Noch eine Grammatik)

Regeln:

$$
\begin{array}{rcl@{\qquad\qquad\qquad}rcl}
S & \to & A & S & \to & A \\
A & \to & E \mid E + A & A & \to & EA' \\
 & & & A' & \to & +A \mid \epsilon \\
E & \to & P \mid P * E & E & \to & PE' \\
 & & & E' & \to & *E \mid \epsilon \\
P & \to & (A) \mid a & P & \to & (A) \mid a
\end{array}
$$

Wir betrachten nun für das Parsen einen top-down-Ansatz, wobei die Produktionen in Form einer Linksableitung angewendet werden.

Beispiel 101 (Noch eine Grammatik)

Regeln:

$$S \to A$$
$$A \to E \mid E + A$$
$$E \to P \mid P * E$$
$$P \to (A) \mid a$$

Ableitung:

$$S$$
$$A$$
$$E$$
$$P$$
$$a \quad + \quad a \quad * \quad a$$

Beispiel 101 (Noch eine Grammatik)

Wir bestimmen nun für jede Produktion $A \to \alpha$ ihre Auswahlmenge, das heißt die Menge aller terminalen Präfixe der Länge $\leq k$ der von α ableitbaren Zeichenreihen. Es ergibt sich (der Einfachheit lassen wir ϵ-Produktionen zu):

$$\begin{array}{rlc}
S & \to A & \{a, (\} \\
A & \to EA' & \{a, (\} \\
A' & \to +A & \{+\} \\
A' & \to \epsilon & \{), \epsilon\} \\
E & \to PE' & \{a, (\} \\
E' & \to *E & \{*\} \\
E' & \to \epsilon & \{+,), \epsilon\} \\
P & \to (A) & \{(\} \\
P & \to a & \{a\}
\end{array}$$

Beispiel 101 (Noch eine Grammatik)

Damit ergibt sich

$$\begin{array}{rll}
S & \to A & \{a, (\} \\
A & \to EA' & \{a, (\} \\
A' & \to +A & \{+\} \\
A' & \to \epsilon & \{), \epsilon\} \\
E & \to PE' & \{a, (\} \\
E' & \to *E & \{*\} \\
E' & \to \epsilon & \{+,), \epsilon\} \\
P & \to (A) & \{(\} \\
P & \to a & \{a\}
\end{array}$$

Ableitung:

$$\begin{array}{l}
S \\
EA' \\
aE'A' \\
\vdots \\
a + PE'A' \\
\vdots \\
a + a * PE'A' \\
\vdots \\
a + a * a
\end{array}$$

Definition 102

Eine kontextfreie Grammatik ist eine $LL(k)$-Grammatik, wenn man durch Lookaheads der Länge k erreichen kann, dass bei einer top-down Linksableitung in jedem Schritt höchstens eine Produktion anwendbar ist. Eine Sprache ist $LL(k)$, wenn es dafür eine $LL(k)$-Grammatik gibt.

Korollar 103

Jede kontextfreie Sprache, für die es eine $LL(k)$-Grammatik gibt, ist deterministisch kontextfrei.

Bemerkung

Man kann zeigen, dass die Klasse der $LL(k)$-Sprachen eine echte Teilklasse der Klasse der $LL(k+1)$-Sprachen ist.

Bemerkungen:

1. Parser für $LL(k)$-Grammatiken entsprechen der Methode des rekursiven Abstiegs (recursive descent).
2. $LL(k)$ ist eine strikte Teilklasse von $LR(k)$.
3. Es gibt $L \in$ DCFL, so dass $L \notin LL(k)$ für alle k.