Beispiel 89 (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 89 (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:

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

Beispiel 89 (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:

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

Beispiel 89 (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 89 (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 89 (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 90

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 91

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 92

Die folgenden Familien von Sprachen sind gleich:

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

Beispiel 93 (Noch eine Grammatik)

Regeln:

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

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

Beispiel 93 (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 93 (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):

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

Beispiel 93 (Noch eine Grammatik)

Damit ergibt sich

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

Ableitung:

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

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.

5. Kontextsensitive und Typ-0-Sprachen

5.1 Turingmaschinen

Turingmaschinen sind das grundlegende Modell, das wir für Computer/Rechenmaschinen verwenden. Es geht auf Alan Turing (1912–1954) zurück.

Definition 94

Eine nichtdeterministische Turingmaschine (kurz TM oder NDTM) wird durch ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ beschrieben, das folgende Bedingungen erfüllt:

1. Q ist eine endliche Menge von Zuständen.
2. Σ ist eine endliche Menge, das Eingabealphabet.
3. Γ ist eine endliche Menge, das Bandalphabet.
4. $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R, N\})$ ist die Übergangsfunktion.
5. $q_0 \in Q$ ist der Startzustand.
6. $\square \in \Gamma \setminus \Sigma$ ist das Leerzeichen.
7. $F \subseteq Q$ ist die Menge der (akzeptierenden) Endzustände.

Eine Turingmaschine heißt deterministisch, falls gilt

$$|\delta(q, a)| \leq 1 \qquad \text{für alle } q \in Q, a \in \Gamma.$$

Erläuterung:
Intuitiv bedeutet $\delta(q,a) = (q',b,d)$ bzw. $\delta(q,a) \ni (q',b,d)$:
Wenn sich M im Zustand q befindet und unter dem Schreib-/Lesekopf das Zeichen a steht, so geht M im nächsten Schritt in den Zustand q' über, schreibt an die Stelle des a's das Zeichen b und bewegt danach den Schreib-/Lesekopf um eine Position nach rechts (falls $d = R$), links (falls $d = L$) bzw. lässt ihn unverändert (falls $d = N$).

Beispiel 95

Es soll eine TM angegeben werden, die eine gegebene Zeichenreihe aus $\{0,1\}^+$ als Binärzahl interpretiert und zu dieser Zahl 1 addiert. Folgende Vorgehensweise bietet sich an:

1. Gehe ganz nach rechts bis ans Ende der Zahl. Dieses Ende kann durch das erste Auftreten eines Leerzeichens gefunden werden.
2. Gehe wieder nach links bis zur ersten 0 und ändere diese zu einer 1. Ersetze dabei auf dem Weg alle 1en durch 0.

Also:

$$\begin{aligned} \delta(q_0,0) &= (q_0,0,R) & \qquad \delta(q_1,1) &= (q_1,0,L) \\ \delta(q_0,1) &= (q_0,1,R) & \qquad \delta(q_1,0) &= (q_f,1,N) \\ \delta(q_0,\Box) &= (q_1,\Box,L) & \qquad \delta(q_1,\Box) &= (q_f,1,N) \end{aligned}$$

Damit ist $Q = \{q_0, q_1, q_f\}$ und $F = \{q_f\}$.

Definition 96

Eine Konfiguration einer Turingmaschine ist ein Tupel $(\alpha, q, \beta) \in \Gamma^* \times Q \times \Gamma^*$.

Das Wort $w = \alpha\beta$ entspricht dem Inhalt des Bandes, wobei dieses rechts und links von w mit dem Leerzeichen $\square$ gefüllt sei. Der Schreib-/Lesekopf befindet sich auf dem ersten Zeichen von $\beta\square^\infty$.

Die Startkonfiguration der Turingmaschine bei Eingabe $x \in \Sigma^*$ entspricht der Konfiguration

$$(\epsilon, q_0, x)\,,$$

d.h. auf dem Band befindet sich genau die Eingabe $x \in \Sigma^*$, der Schreib-/Lesekopf befindet sich über dem ersten Zeichen der Eingabe und die Maschine startet im Zustand q_0.

Je nach aktuellem Bandinhalt und Richtung $d \in \{L, R, N\}$ ergibt sich bei Ausführung des Zustandsübergangs $\delta(q, \beta_1) = (q', c, d)$ folgende Änderung der Konfiguration:

$$(\alpha_1 \cdots \alpha_n, q, \beta_1 \cdots \beta_m) \rightarrow \begin{cases} (\alpha_1 \cdots \alpha_n, q', c\beta_2 \cdots \beta_m) & \text{falls } d = N, \\ & n \geq 0, m \geq 1 \\ (\epsilon, q', \square c\beta_2 \cdots \beta_m) & \text{falls } d = L, \\ & n = 0, m \geq 1 \\ (\alpha_1 \cdots \alpha_{n-1}, q', \alpha_n c\beta_2 \cdots \beta_m) & \text{falls } d = L, \\ & n \geq 1, m \geq 1 \\ (\alpha_1 \cdots \alpha_n c, q', \square) & \text{falls } d = R, \\ & n \geq 0, m = 1 \\ (\alpha_1 \cdots \alpha_n c, q', \beta_2 \cdots \beta_m) & \text{falls } d = R, \\ & n \geq 0, m \geq 2 \end{cases}$$

Der Fall $m = 0$ wird mittels $\beta_1 = \square$ abgedeckt.

Definition 97

Die von einer Turingmaschine M akzeptierte Sprache ist

$$L(M) = \{x \in \Sigma^*;\ (\epsilon, q_0, x) \to^* (\alpha, q, \beta) \text{ mit } q \in F, \alpha, \beta \in \Gamma^*\}$$