
Auflösen von Mehrdeutigkeiten in kontextfreien Grammatiken

Lennart Protte

17.05.2024

Mehrdeutige Grammatik

Grammatik

$$G(N, T, P, S)$$

$$N = \{E\}$$

$$T = \{+, *, \mathbf{num}\}$$

$$S = \{E\}$$

$$P = \{$$

$$E \rightarrow E + E \mid E * E \mid \mathbf{num}$$

$$\}$$

Ableitungen

$$E \rightarrow E + E$$

$$\rightarrow \mathbf{num} + E$$

$$\rightarrow \mathbf{num} + \mathbf{num}$$

$$E \rightarrow E + E$$

$$\rightarrow E + \mathbf{num}$$

$$\rightarrow \mathbf{num} + \mathbf{num}$$

Definition[Vasudevan and Tratt, 2013]

Eine Grammatik ist mehrdeutig, wenn sie mehrere Ableitungen für ein Wort zulässt.

Definition der Chomsky-Normalform

Definition[Watrous, 2020]

Eine kontextfreie Grammatik ist in der Chomsky-Normalform, wenn jede Produktionsregel eine der folgenden Formen hat:

$$A \rightarrow BC$$

$$A \rightarrow a$$

$$S \rightarrow \varepsilon$$

Hierbei sind $A, B, C \in N$, $S \in$ Startsymbol und $a \in T$.
Für B und C gilt $B, C \neq S$.

Chonsky Normal Form Beispiel

CFG

$$\begin{aligned} G(N, T, P, S) \\ N &= \{E\} \\ T &= \{+, *, \mathbf{num}\} \\ S &= \{E\} \\ P &= \{ \\ E &\rightarrow E + E \mid E * E \mid \mathbf{num} \\ &\} \end{aligned}$$

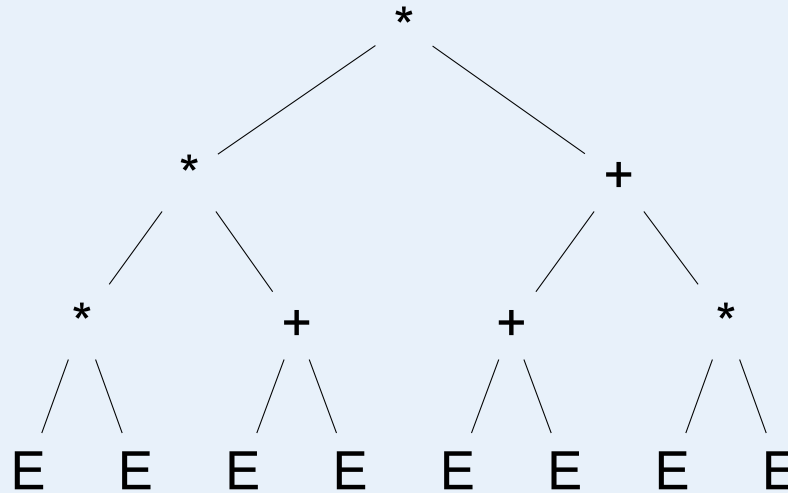
→

CNF

$$\begin{aligned} G'(N, T, P, S) \\ N &= \{S, E, H_0, H_1, H_2, H_3\} \\ T &= \{+, *, \mathbf{num}\} \\ S &= \{S\} \\ P &= \{ \\ S &\rightarrow H_0 E \mid H_1 E \mid \mathbf{num} \\ E &\rightarrow H_0 E \mid H_1 E \mid \mathbf{num} \\ H_0 &\rightarrow E H_2 \\ H_1 &\rightarrow E H_3 \\ H_2 &\rightarrow + \\ H_3 &\rightarrow * \\ &\} \end{aligned}$$

Baum von G'

Beispielhafter AST von G'



Bewertung[Watrous, 2020]

Jeder, aus einer CNF erzeugter Baum, ist ein Binärbaum.

Definition von Mustern

Definition[Vasudevan and Tratt, 2013]

Gegeben ist das Tupel $(S, \alpha \circ \beta, \gamma)$ mit:

S = Das Startsymbol der Produktion

$\alpha \circ \beta$ = zwei Nichtterminale aus G mit \circ als Operator

γ = Die Ableitung

Ob α oder β abgeleitet wird,

wird durch ein \bullet vor dem jeweiligen Nichtterminal gekennzeichnet.

Beispiel

So kann $E \Rightarrow E + E \Rightarrow \mathbf{num} + E \Rightarrow \mathbf{num} + \mathbf{num}$

durch $(E, \bullet E + E, \mathbf{num})$ und $(E, E + \bullet E, \mathbf{num})$ beschrieben werden.

Tabellen für Ordnung und Assoziativität

Aus der Ordnung $\alpha_1 > \alpha_2$ ergibt sich:

$>$	$E ::= E\alpha_2 E$
$E ::= E\alpha_1 E$	$(E, \bullet E\alpha_1 E, E\alpha_2 E)$ $(E, E\alpha_1 \bullet E, E\alpha_2 E)$

[Vasudevan and Tratt, 2013]

Aus α_1, α_2 sind links assoziativ folgt:

links	$E ::= E\alpha_1 E$	$E ::= E\alpha_2 E$
$E ::= E\alpha_1 E$	$(E, E\alpha_1 \bullet E, E\alpha_1 E)$	$(E, E\alpha_1 \bullet E, E\alpha_2 E)$
$E ::= E\alpha_2 E$	$(E, E\alpha_2 \bullet E, E\alpha_1 E)$	$(E, E\alpha_2 \bullet E, E\alpha_2 E)$

[Vasudevan and Tratt, 2013]

Beispiel Algorithmus

Definition der Grammatik:

$$E ::= E * E \text{ (links)} > E + E \text{ (links)} | \mathbf{num}$$

Operator	Vorrang	Assoziativität
*	2	links assoziativ
+	1	links assoziativ

Beispiel

So wird $2 + 3 * 4$ gemäß der Definierten Assoziativität und der Vorrangsregeln eindeutig als $2 + (3 * 4)$ ausgewertet, was korrekt ist.

Beispiel Algorithmus

Muster zur Beseitigung der Mehrdeutigkeit

$$\begin{aligned} &(E, E * \bullet E, E * E) \\ &(E, \bullet E * E, E + E) \\ &(E, E + \bullet E, E * E) \\ &(E, \bullet E + E, E + E) \end{aligned}$$

Nach der Erstellung von neuen Nichtterminalen

$$\begin{aligned} &G(N, T, P, S) \\ &N = \{E, E_1, E_2, E_3, E_4\} \\ &T = \{+, *, \mathbf{num}\} \\ &S = \{E\} \\ &P = \{ \\ &\quad E \rightarrow E_1 * E_2 | E_3 + E_4 | \mathbf{num} \\ &\} \end{aligned}$$

Beispiel Algorithmus

Nach der Erstellung von neuen Produktionsregeln

$$\begin{aligned} G(N, T, P, S) \\ N &= \{E, E_1, E_2, E_3, E_4\} \\ T &= \{+, *, \mathbf{num}\} \\ S &= \{E\} \\ P &= \{ \\ E &\rightarrow E_1 * E_2 \mid E_3 + E_4 \mid \mathbf{num} \\ E_1 &\rightarrow E_1 * E_2 \mid \mathbf{num} \\ E_2 &\rightarrow E_3 + E_4 \mid \mathbf{num} \\ E_3 &\rightarrow E_1 * E_2 \mid \mathbf{num} \\ E_4 &\rightarrow E_3 + E_4 \mid \mathbf{num} \\ &\} \end{aligned}$$


Nach dem Prüfen von verschachtelten Fällen

$$\begin{aligned} G(N, T, P, S) \\ N &= \{E, E_1, E_2, E_3, E_4\} \\ T &= \{+, *, \mathbf{num}\} \\ S &= \{E\} \\ P &= \{ \\ E &\rightarrow E_1 * E_2 \mid E_3 + E_4 \mid \mathbf{num} \\ E_1 &\rightarrow E_1 * E_5 \mid \mathbf{num} \\ E_2 &\rightarrow E_3 + E_4 \mid \mathbf{num} \\ E_3 &\rightarrow E_5 * E_2 \mid \mathbf{num} \\ E_4 &\rightarrow E_3 + E_4 \mid \mathbf{num} \\ E_5 &\rightarrow \mathbf{num} \\ &\} \end{aligned}$$

- [Vasudevan and Tratt, 2013] Vasudevan, N. and Tratt, L. (2013).
Detecting ambiguity in programming language grammars.
In Erwig, M., Paige, R. F., and Van Wyk, E., editors, *Software Language Engineering*, pages 157–176, Cham. Springer International Publishing.
- [Watrous, 2020] Watrous, J. (2020).
Parse trees, ambiguity, and chomsky normal form.
<https://cs.uwaterloo.ca/~watrous/ToC-notes/ToC-notes.08.pdf>.
[Online; accessed 17-April-2024].