Auflösen von Mehrdeutigkeiten in kontextfreien Grammatiken

Lennart Protte

17.05.2024

Mehrdeutige Grammatik

Grammatik

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

Definition[Vasudevan and Tratt, 2013]

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

Ableitungen

$$
\begin{aligned}
E &\rightarrow E + E\\
&\rightarrow E + E * E\\
&\rightarrow \textbf{num} + E * E\\
&\rightarrow \textbf{num} + \textbf{num} * E\\
&\rightarrow \textbf{num} + \textbf{num} * \textbf{num}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow E + E\\
&\rightarrow \textbf{num} + E\\
&\rightarrow \textbf{num} + E * E\\
&\rightarrow \textbf{num} + \textbf{num} * E\\
&\rightarrow \textbf{num} + \textbf{num} * \textbf{num}
\end{aligned}
$$

Definition der Chomsky-Normalform

Definition[Watrous, 2020]

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

$$A \to BC$$
$$A \to a$$
$$S \to \varepsilon$$

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

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

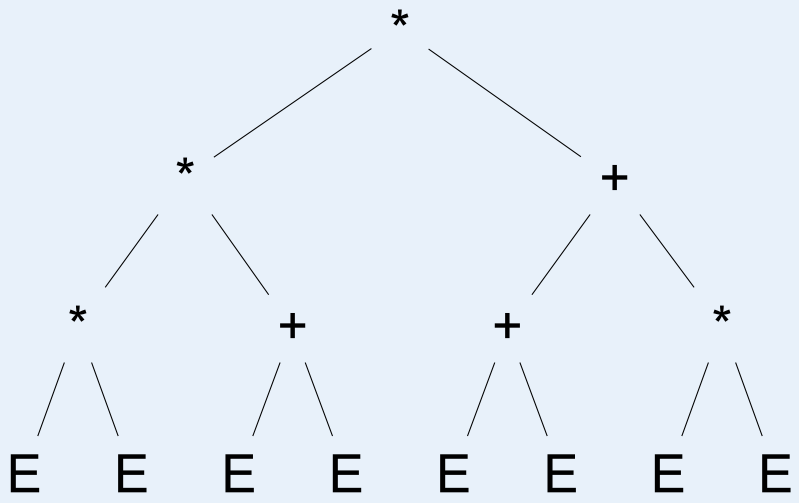
$\longrightarrow$

CNF

$$
\begin{aligned}
&G'(N, T, P, S)\\
&N = \{S, E, H_0, H_1, H_2, H_3\}\\
&T = \{+, *, \textbf{num}\}\\
&S = \{S\}\\
&P = \{\\
&S \rightarrow H_0E | H_1E | \textbf{num}\\
&E \rightarrow H_0E | H_1E | \textbf{num}\\
&H_0 \rightarrow EH_2\\
&H_1 \rightarrow EH_3\\
&H_2 \rightarrow +\\
&H_3 \rightarrow *\\
&\}
\end{aligned}
$$

Baum von G'

Beispielhafter AST von G'



Bewerkung[Watrous, 2020]

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

Lösen von Mehrdeutigkeiten bei Operatoren

Probleme der Chomsky Normal Form

- Keine Aussage über die Mehrdeutigkeit einer Grammatik
- Für komplexe Sprachen werden viele Produktionen benötigt

Methode zur Beseitigung von Mehrdeutigkeiten

- Durch Operatorvorrang und Operatorassoziativität können alle Mehrdeutigkeiten beseitigt werden
- Die Grammatik kann weiterhin kontextfrei bleiben

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 $E \Rightarrow E + E \Rightarrow \textbf{num} + E \Rightarrow \textbf{num} + \textbf{num}$
durch $(E, \bullet E + E, \textbf{num})$ und $(E, E + \bullet E, \textbf{num})$ 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:

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

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

$$(E, E * \bullet E, E * E)$$
$$(E, \bullet E * E, E + E)$$
$$(E, E + \bullet E, E * E)$$
$$(E, \bullet E + E, E + E)$$

Nach der Erstellung von neuen Nichtterminalen

$$G(N, T, P, S)$$
$$N = \{E, E_1, E_2, E_3, E_4\}$$
$$T = \{+, *, \textbf{num}\}$$
$$S = \{E\}$$
$$P = \{$$
$$E \rightarrow E_1 * E_2 | E_3 + E_4 | \textbf{num}$$
$$\}$$

Beispiel Algorithmus

Nach der Erstellung von neuen Produktionsregeln

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

Beispiel aufgelöster Produktionsregel

Für beispielsweise $E + E + E$ gibt es nur noch eine Möglichkeit:

$$
\begin{aligned}
E &\to E_3 + E_4\\
&\to E_3 + E_3 + E_4
\end{aligned}
$$

Das Muster $(E, \bullet E + E, E + E)$ verhindert die Alternative.

Beispiel Algorithmus

Nach dem Prüfen von verschachtelten Fällen

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

Bemerkung

- Die Vorrangs und Assoziativitätsregeln sind in der Produktion umgesetzt.
- Die Grammatik ist nun eindeutig.
- Die Grammatik ist frei von Zyklen

Quellen

[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].