Bottom-Up-Parsing für Typ 2-Sprachen

Hans Jürgen Ohlbach

Keywords: Typ 2-Sprachen, Bottom-Up-Parsing, CYK-Algorithmus, Komplexität

Empfohlene Vorkenntnisse: Formale Sprachen, Typ 2-Sprachen

Inhaltsverzeichnis

1 Einführung **1**

2 Bottom-Up-Parsing **2**

3 Der CYK-Algorithmus **2**

1 Einführung

Die Lösung des Wortproblems für eine Sprache geschieht mit *Parsern*, die ein Wort analysieren, und entscheiden, ob es zur Sprache gehört oder nicht. Hierfür gibt es für Typ 2-Sprachen u.A. den Ansatz des *Bottom-Up-Parsings*, d.h. man startet mit dem Wort, und arbeitet rückwärts bis zur Startvariablen der entsprechenden Grammatik. Dabei ersetzt man immer rechte Seiten einer Produktion durch linke Seiten.

Wir zeigen zunächst mal die prinzipielle Idee, und dann die konkrete Umsetzung im CYK-Algorithmus.

2 Bottom-Up-Parsing

Ein erster Eindruck, wie Bottom-Up-Parsing funktioniert, gibt das folgende Beispiel.

Die Grammatik sei

$$S \to 0 \mid \ldots \mid 9 \mid (S + S) \mid (S * S)$$

Das zu parsende Wort sei $(3 + (4 * 5))$

Bottom-Up-Parsing bedeutet jetzt, dass man Regeln sucht, deren rechte Seiten im Wort zu finden sind, und diese dann durch die entsprechenden linken Seiten ersetzt. Wenn das Wort damit irgendwann auf die Startvariable schrumpft, wird das Wort akzeptiert.

Im Beispiel sieht das folgendermaßen aus:

$(3 + (4 * 5))$	Regeln:	$S \to 3, S \to 4, S \to 5$
$(S + (S * S))$	Regel:	$S \to (S * S)$
$(S + S)$	Regel:	$S \to (S + S)$
S		

Diese Idee lässt sich mehr oder weniger direkt auch auf Typ 0 und Typ 1-Sprachen anwenden.

3 Der CYK-Algorithmus

Der *CYK-Algorithmus* (von Cocke, Younger und Kasami) systematisiert das Bottom-Up-Parsing für Typ 2-Sprachen. Er braucht dazu allerdings die Chomsky-Normalform für die Grammatik.

Wir illustrieren den Algorithmus an folgendem Beispiel:

Die Grammatikregeln seien:

$$\begin{array}{lcl} S & \to & AB \mid BC \\ A & \to & BA \mid a \\ B & \to & CC \mid b \\ C & \to & AB \mid a \end{array}$$

Das zu parsende Wort sei $baaba$. Dafür bauen wir jetzt eine Dreiecksmatrix auf, deren oberste Zeile das Wort enthält. Als Gedächtnisstütze sind in der Matrix die jeweiligen Teilworte eingetragen, die die Variablen in der Zelle darunter produzieren können.

	b	a	a	b	a
1					
	ba	aa	ab	ba	
2					
	baa	aab	aba		
3					
	$baab$	$aaba$			
4					
	$baaba$				
5					

Die erste Zeile wird jetzt mit den linken Regelseiten gefüllt, die die jeweiligen Konstante erzeugen.

	b	a	a	b	a
1	B	A, C	A, C	B	A, C
	ba	aa	ab	ba	
2					
	baa	aab	aba		
3					
	$baab$	$aaba$			
4					
	$baaba$				
5					

Die zweite Zeile wird mit den linken Regelseiten gefüllt, die die Buchstabenpaare erzeugen können. Für das erste Paar ba kommen, nach der ersten Zeile, als rechte Seiten BA und BC in Frage. BC gibt es als rechte Seite der Regel $S \to BC$. BA gibt es als rechte Seite der Regel $A \to BA$. Daher wird in die erste Zelle S und A eingetragen.

Die zweite Zelle soll das Paar aa erzeugen. Laut den beiden oberhalb liegende Zellen, kämen dafür die Paare AA, AC, CA und CC in Frage. Davon gibt es aber nur CC als rechte Seite der Regel $B \to CC$. Daher wird in die Zelle B eingetragen.

Die dritte Zelle soll das Paar ab erzeugen. Die zwei Zellen der darüber liegenden Zeile schlagen die Paare AB und CB vor. Für AB als rechte Seite gibt es zwei Produktionen, $S \to AB$ und $C \to AB$. Beide linken Seiten S und C müssen eingetragen werden.

Für die letzte Zelle der zweiten Zeile ergibt sich dann analog der Eintrag S, A. Die Matrix sieht jetzt so aus:

	b	a	a	b	a
1	B	A, C	A, C	B	A, C
	ba	aa	ab	ba	
2	S, A	B	S, C	S, A	
	baa	aab	aba		
3					
	$baab$	$aaba$			
4					
	$baaba$				
5					

Die erste Zelle der 3. Zeile soll die Variablen enthalten, die das Tripel baa produzieren. Dieses Tripel kann man entweder aus b und dem Tupel aa aufbauen, oder aus dem Tupel ba und dem Buchstaben a. In den Zeilen darüber steht, aus welchen Variablen diese produziert werden können. b wird aus B produziert, und aa auch aus B. Also brauchen wir für diese Variante BB. Das gibt es aber nicht als rechte Seite. Die Alternative ist ba, welches aus S und A produziert wird, und a, welches aus A und C produziert wird. Dafür bräuchten wir also rechte Seiten SA, SC, AA AC. Keine von denen existiert. Also bleibt die Zelle leer.

Für die zweite Zelle in der dritten Reihe, d.h. die Erzeugung von aab folgen wir den roten Pfeilen, und kombinieren einmal A, C (zur Erzeugung von a) mit S, C zur Erzeugung von ab. Das gibt die Möglichkeiten AS, AC, CS und CC. Nur CC kommt als rechte Seite vor. Die linke Seite ist B. Die zweite Kombination ist B (zur Erzeugung von aa) mit B (zur Erzeugung von b). BB kommt aber als rechte Seite nicht vor. Als bleibt als einzige Möglichkeit zur Erzeugung von aab die Variable B.

Für die dritte Zelle zur Erzeugung von aba folgen wir den blauen Pfeilen und erhalten ebenfalls B.

	b	a	a	b	a
1	B	A, C	A, C	B	A, C
	ba	aa	ab	ba	
2	S, A	B	S, C	S, A	
	baa	aab	aba		
3	$-$	B	B		
	$baab$	$aaba$			
4					
	$baaba$				
5					

Um die nächste Zeile zu füllen, gibt es jeweils drei Kombinationsmöglichkeiten. $baab$ kann man aus b und aab erzeugen, aus ba und ab sowie aus baa und b. Das wird durch den roten Pfeil angedeutet. Es gibt aber keine Produktion, die eines davon erzeugt.

Um $aaba$ zu erzeugen gibt es auch wieder drei Kombinationsmöglichkeiten, die durch den blauen Pfeil angedeutet werden. Hier ergeben sich drei linke Regelseiten: S, A, C.

	b	a	a	b	a
1	B	A, C	A, C	B	A, C
	ba	aa	ab	ba	
2	S, A	B	S, C	S, A	
	baa	aab	aba		
3	$-$	B	B		
	$baab$	$aaba$			
4	$-$	S, A, C			
	$baaba$				
5					

Die letzte Zelle, zur Erzeugung des Gesamtwortes $baaba$ wird kombiniert aus den vier Möglichkeiten: 1. b und $aaba$, 2. ba und aba, 3. baa und ba, 4. $baab$ und a, welche durch den roten Pfeil angedeutet werden. Es ergeben sich S, A, C als mögliche linke Seiten. Da darunter die Startvariable S ist, ist das Wort akzeptiert.

	b	a	a	b	a
1	B	A, C	A, C	B	A, C
	ba	aa	ab	ba	
2	S, A	B	S, C	S, A	
	baa	aab	aba		
3	$-$	B	B		
	$baab$	$aaba$			
4	$-$	S, A, C			
	$baaba$				
5	S, A, C				

Das Beispiel soll deutlich machen, wie der CYK-Algorithmus ab der 3. Zeile vorgeht. Man arbeitet alle Kombinationsmöglichkeiten ab, indem man ab der ersten Zeile die Vertikale kombiniert mit der Diagonalen.

Indem man sich separat merkt, wie die einzelnen Einträge entstanden sind, kann man auch den Syntaxbaum rekonstruieren, nicht nur einen, sondern tatsächlich alle Syntaxbäume, wenn es mehrere für das Wort gibt. Das ergibt sich daraus, dass das Verfahren *alle* Varianten berechnet, wie man das Wort, und alle seine Teilwörter, erzeugen kann.

Die *Komplexität* des CYK-Algorithmus ergibt sich einmal aus der Erzeugung der Matrix: $\mathcal{O}(n^2)$, und dem Ausfüllen der Zellen. Für jede Zelle muss man die Vertikale mit der Diagonalen kombinieren, was noch einmal $\mathcal{O}(n)$ kostet, also zusammen $\mathcal{O}(n^3)$.

Für realistische Wortgrößen, z.B. zum Parsen kompletter Computerprogramme, ist das Verfahren jedoch viel zu aufwendig. Da bevorzugt man die Top-Down-Verfahren.