Hochschule RheinMain
Prof. Dr. Steffen Reith
Jan Reinhard

9. Übungsblatt

Der CYK-Algorithmus kann das Wortproblem für kontextfreie Sprachen „schnell“ (Polynomialzeit) lösen und ist nach seinen Erfindern Cocke, Younger und Kasami benannt.

Sei $x = a_1 \dots a_n$, dann ist $x_{i,j}$ das Teilwort von x, das an *Position i startet* und die *Länge j* hat.

Die grundlegende Idee des CYK-Algorithmus ist, dass für jede kontextfreie Sprache L eine Grammatik G in Chomsky-Normalform mit $L = L(G)$ existiert. Es gilt:

Worte der Länge 1: Es kommt nur eine Regel der Form $A \to a$ in Frage um das Wort zu erzeugen.

Worte der Länge > 1: Wenn $A \overset{*}{\vdash} x$ gilt, dann wurde im ersten Schritt eine Regel der Form $A \to BC$ angewendet. Sei nun $x = a_1 \dots a_n$, dann wird ein Teil von x aus B erzeugt und das Reststück aus C. Es muss also ein $1 \leq k \leq n$ geben, sodass sich der Syntaxbaum aus Abbildung 1 ergibt. Mit Hilfe dieser Idee können wir das Wortproblem für die Länge n auf zwei Probleme der Länge k und $n-k$ zurückführen. Dabei ist allerdings das k (also der Index an dem das Wort in zwei Teile zerlegt wird) zu diesem Zeitpunkt unbekannt. Aus diesem Grund müssen alle Möglichkeiten von 1 bis $n-1$ untersucht werden.

Um das Problem der richtigen „Unterteilungsposition“ k zu lösen, verwenden wir die Methode des *dynamischen Programmierens*:

Der CYK-Algorithmus (siehe Algorithmus 1 auf Seite 3) verwendet eine zweidimensionale Matrix $T[1 \dots n][1 \dots n]$, wobei aber nur die obere Dreiecksmatrix zum Einsatz kommt. Im Tabelleneintrag $T[i][j]$ sind alle die Nichtterminale notiert, aus denen $x_{i,j}$ abgeleitet werden kann. Offensichtlich gilt dann $x = a_1 \dots a_n \in L(G)$ gdw. $S \in T[1][n]$.

Lösen Sie nun die folgenden Aufgaben:

1. Gegeben sei die Grammatik $G = (\{a, (,), +, *\}, \{E\}, E, P)$, wobei $P = \{E \to E + E, E \to E * E, E \to (E), E \to a\}$. Geben Sie zwei *unterschiedliche* Wege an, mit denen das Wort $a{+}a{*}a$ aus dem Startsymbol E abgeleitet werden kann und zeichnen Sie die dazugehörigen Syntaxbäume, die dann auch eine unterschiedliche Struktur haben müssen. Welche Problematik steckt hinter diesen strukturellen Unterschieden?

2. Gegeben sei die Sprache $L_1 =_{\text{def}} \{0^p \mid p \text{ ist Primzahl}\}$. Zeigen Sie, dass L_1 nicht kontextfrei ist.

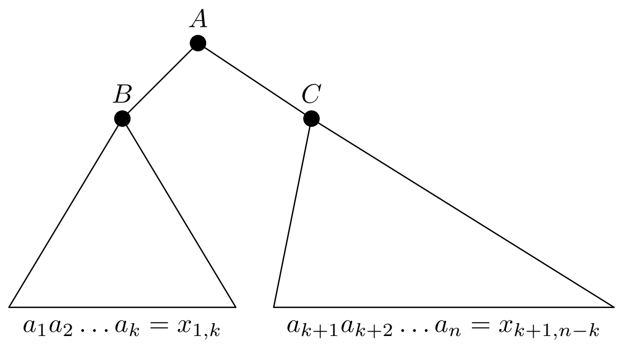

Abbildung 1: Struktur eines Syntaxbaums bei Anwendung einer Regel $A \to BC$

3. Sei die Grammatik $G = (\{a, b\}, \{S, A, B, C\}, S, P)$ mit

$$P = \left\{ \begin{array}{ll} S \to AB, & A \to BA, \\ S \to BC, & A \to a, \\ B \to CC, & C \to AB, \\ B \to b, & C \to a \end{array} \right\}$$

   gegeben. Überprüfen Sie mit Hilfe des CYK-Algorithmus, ob das Wort *baaba* in $L(G)$ enthalten ist. Welche Laufzeit hat der CYK-Algorithmus?

Besprechung in den Übungen ab dem 28.6.2021.

Data: Grammatik $G = (\Sigma, N, P, S)$, Wort $x = a_1, \ldots, a_n$ und Länge n
Result: `true` falls $x \in L(G)$ und `false` sonst

for *($i = 1$; $i \leq n$; i++)* **do**
　　　　`/* Worte der Länge 1 */`
　$T[i][1] = \{A \in N \mid A \to a_i \ \in P\}$;
end
　　　　`/* Bearbeite alle möglichen Längen */`
for *($j = 2$; $i \leq n$; j++)* **do**
　　　　`/* Bearbeite alle möglichen Startpositionen */`
　for *($i = 1$; $i \leq n + 1 - j$; i++)* **do**
　　　　`/* Initialisiere die Mengenvariable */`
　　$T[i][j] = \emptyset$;
　　　`/* Teste mögliche Zerlegungen für Worte der Länge` j `an Position` i `*/`
　　for *($k = 1$; $k \leq j - 1$; k++)* **do**
　　　$T[i][j] = T[i][j] \cup \{A \in N \mid A \to BC \in P, B \in T[i][k]$ und $C \in T[i+k][j-k]\}$;
　　end
　end
end
　　　　`/* Teste` $x \overset{?}{\in} L$ `(`$\triangleq$ `das Wort` x `kann aus` S `erzeugt werden) */`
if *($S \in T[1][n]$)* **then**
　return `true`;
else
　return `false`;
end

Algorithmus 1: Der CYK-Algorithmus