66115
Frühjahr
2012

Theoretische Informatik / Algorithmen (vertieft)

Aufgabenstellungen mit Lösungsvorschlägen

Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht

Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Aufgabe 3 [Kontextfrei aber nicht regulär]

Beweisen Sie, dass folgende Sprache kontextfrei, aber nicht regulär ist.

$$C = \{ a^n b^m \mid n \geq m \geq 1 \}$$

Nachweis Kontextfrei über Grammatik

$G = (\{S\}, \{a, b\}, P, S)$

$$P = \Big\{ \quad S \rightarrow aSb \mid aS \mid ab \quad \Big\}$$

- Regel 1: aSb
- Regel 2: aS
- Regel 3: ab

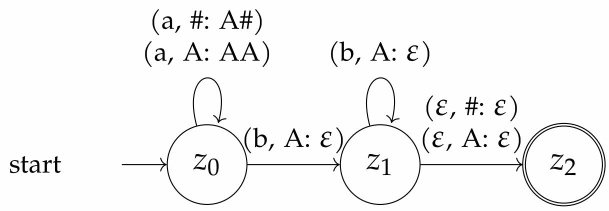
ab**:** $S \xrightarrow{3} ab$

$a^n b$**:** $S \xrightarrow[n-1]{2} a^{n-1}S \xrightarrow{3} a^{n-1}ab$

$a^n b^m$**:** $S \xrightarrow[m-1]{1} a^{m-1}Sb^{m-1} \xrightarrow[n-(m-1)]{2} a^{n-1}Sb^{m-1} \xrightarrow{3} a^n b^m$

$\Rightarrow L(G) = C$

Nachweis Kontextfrei über Kellerautomat



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aji151myg

Nachweis: C nicht regulär

C sei regulär

$\Rightarrow$ Pumping-Lemma für C erfüllt
j sei die Pumping-Zahl ($j \in \mathbb{N}$)
$\omega \in C$: $\omega = a^j b^j$
$\omega = uvw$
Dann gilt:

- $|v| \geq 1$
- $|uv| \leq j$
- $uv^i w \in C$ für alle $i \in \mathbb{N}_0$

In uv können nur a's vorkommen
$\Rightarrow$ In v muss mindestens ein a vorkommen
$\Rightarrow uv^0 w = a^l (a^{j-l})^0 b^j$ $((a^{j-l})^0 = \varepsilon)$
$\Rightarrow$ In ω' sind nur l viele a's, Da $l < j$, $\omega' \notin C$,
$\Rightarrow$ Widerspruch zur Annahme
$\Rightarrow$ C nicht regulär

Aufgabe 4 [Nonterminale: SAB, Terminale: ab]

Gegeben ist die kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit $\Sigma = \{a, b\}$, $N = \{S, A, B\}$ und

P = $\Bigg\{$

$$
\begin{aligned}
S &\rightarrow A \\
S &\rightarrow B \\
A &\rightarrow aAb \\
B &\rightarrow AA \\
B &\rightarrow bBa \\
A &\rightarrow a
\end{aligned}
$$

$\Bigg\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gr3rgt2vg

Geben Sie eine äquivalente Grammatik in Chomsky-Normalform an.

Lösungsvorschlag

Kann auch so geschrieben werden:

P = $\Bigg\{$

$$
\begin{aligned}
S &\rightarrow A \mid B \\
A &\rightarrow aAb \mid a \\
B &\rightarrow AA \mid bBa
\end{aligned}
$$

$$\}$$

(a) **Elimination der ε-Regeln**

— *Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.*

∅ Nichts zu tun

(b) **Elimination von Kettenregeln**

— *Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.*

$$\mathrm{P} = \{$$

$$\begin{aligned} S &\to aAb \mid a \mid AA \mid bBa \\ A &\to aAb \mid a \\ B &\to AA \mid bBa \end{aligned}$$

$$\}$$

(c) **Separation von Terminalzeichen**

— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.*

$$\mathrm{P} = \{$$

$$\begin{aligned} S &\to T_aAT_b \mid a \mid AA \mid T_bBT_a \\ A &\to T_aAT_b \mid a \\ B &\to AA \mid T_bBT_a \\ T_a &\to a \\ T_b &\to b \end{aligned}$$

$$\}$$

(d) **Elimination von mehrelementigen Nonterminalketten**

— *Alle Produktionen der Form $A \to B_1B_2 \ldots B_n$ werden in die Produktionen $A \to A_{n-1}B_n, A_{n-1} \to A_{n-2}B_{n-1}, \ldots, A_2 \to B_1B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.*

$$\mathrm{P} = \{$$

$$
\begin{aligned}
S &\to T_aC \mid a \mid AA \mid T_bD \\
A &\to T_aC \mid a \\
B &\to AA \mid T_bD \\
T_a &\to a \\
T_b &\to b \\
C &\to AT_b \\
D &\to BT_a
\end{aligned}
$$

}

Aufgabe 7 [Kürzeste Kreise]

Mit der Länge eines Pfads oder eines Kreises bezeichnen wir die Anzahl der Kanten, aus denen der Pfad bzw. der Kreis besteht. Bekanntlich kann man Breitensuche verwenden, um für zwei gegebene Knoten s und ? die Länge eines kürzesten s-t-Wegs zu berechnen. Im folgenden geht es um die Berechnung kürzester Kreise.

(a) Für einen Graphen G und einen Knoten v von G berechnet KK(G,v) (siehe Abbildung 1) die Länge des kürzesten Kreises in G, der durch v geht.

Analysieren Sie die Laufzeit von KK in Abhängigkeit von der Anzahl n der Knoten von G, von der Anzahl m der Kanten von G und vom Grad deg(v) des übergebenen Knotens v.

(b) Wenn man den Algorithmus KK für jeden Knoten eines Graphen G aufruft, kann man die Länge eines kürzesten Kreises in G berechnen. Welche Laufzeit hat der resultierende Algorithmus in in Abhängigkeit von n und m?

(c) Geben Sie einen Algorithmus KKschnell(G, v) an, der in O(n +m) Zeit die Länge des kürzesten Kreises in G berechnet, der durch v geht. Argumentieren Sie, warum ihr Algorithmus korrekt ist.

Abbildung 1
vum pam aba ee aan mn a sr lee
KK(ungerichteter UNBEWICHLELEN rapı
ıL= 2 Adi): =weV | v,w € E 8 foreach w € Adjlvj do
4 | Sei G′ der Graph G ohne die Kante v, w.
5 Sei £ die Länge eines kürzesten v-w-Wegs in G′. 6 if 2< L then
7 | Lex
s return L