Prof. T. Nipkow
M. Eberl, J. Rädle, S. Wimmer
Technische Universität München
**PROGRAMMIERABGABE** **4 Punkte**
**Abgabefrist: Dienstag, 18.06.2019, 14 Uhr**

# Einführung in die theoretische Informatik

Sommersemester 2019 – Hausaufgabenblatt 7

**Templates in anderen Sprachen**

Für die Korrektur Ihrer Programmierabgaben wird TUMjudge(`https://judge.in.tum.de/theo/public/`) verwendet. Auf dem ersten Hausaufgabenblatt wurden die für Sie relevanten Funktionen des Webinterfaces erklärt. Weitere Details finden Sie auf den Folien des Praktikums Algorithms for Programming Contests von 2017(`https://www7.in.tum.de/um/courses/theo/ss2018/materials/judge.pdf`).

**Programmierabgabe**

Aus Zeitgründen können wir diese Woche zum Zeitpunkt der Blattausgabe keine Python- und Haskell-Templates bereitstellen. Wir werden uns darum bemühen, diese noch in den kommenden Tagen (voraussichtlich Dienstag Nachmittag) nachzureichen.

**AUFGABE 7.1.** (*Automata Tutor*) 2 Punkte

Sie finden in Automata Tutor zwei neue Hausaufgaben, in denen Sie PDA's für konkrete Sprachen angeben müssen. Die Abgabefrist für diese Aufgaben ist einen Tag länger als für die Programmieraufgaben.

**AUFGABE 7.2.** (*Erreichbare Nichtterminale*) 1 Punkte

In dieser Aufgabe sollen Sie einen Algorithmus implementieren, der die Menge der erreichbaren Nichtterminale einer kontextfreien Grammatik berechnet.

- Laden Sie das Codegerüst und die Angabe für die Hausaufgabe auf `https://www21.in.tum.de/teaching/theo/SS19/materials/ha/07/templates_ha07.tar.gz` herunter.
- Lesen Sie sich die PDF-Angabe zu der Aufgabe 'CFGReachable' durch und folgen Sie den Anweisungen dort.
- Laden Sie dann für Problem A (CFGReachable) alle benötigten Dateien hoch – falls Sie eines unserer Templates benutzen, müssen Sie *alle* Dateien des Templates hochladen, nicht nur die, die Sie verändert haben.

**AUFGABE 7.3.** (*Erzeugende Nichtterminale*) 1 Punkte

In dieser Aufgabe sollen Sie einen Algorithmus implementieren, der die Menge der erzeugenden Nichtterminale einer kontextfreien Grammatik berechnet.

- Laden Sie das Codegerüst und die Angabe für die Hausaufgabe auf `https://www21.in.tum.de/teaching/theo/SS19/materials/ha/07/templates_ha07.tar.gz` herunter.
- Lesen Sie sich die PDF-Angabe zu der Aufgabe 'CFGProductive' durch und folgen Sie den Anweisungen dort.
- Laden Sie dann für Problem B (CFGProductive) alle benötigten Dateien hoch – falls Sie eines unserer Templates benutzen, müssen Sie *alle* Dateien des Templates hochladen, nicht nur die, die Sie verändert haben.

**AUFGABE 7.4.** (*Ableitungsbäume*) 1 Punkte

In dieser Aufgabe sollen Sie einen Algorithmus implementieren, der für eine gegebene kontextfreie Grammatik in Chomsky-Normalform alle Ableitungsbäume für ein bestimmtes Wort berechnet.

- Laden Sie das Codegerüst und die Angabe für die Hausaufgabe auf `https://www21.in.tum.de/teaching/theo/SS19/materials/ha/07/templates_ha07.tar.gz` herunter.
- Lesen Sie sich die PDF-Angabe zu der Aufgabe 'CFGDerivationTree' durch und folgen Sie den Anweisungen dort.
- Laden Sie dann für Problem C (CFGDerivationTree) alle benötigten Dateien hoch – falls Sie eines unserer Templates benutzen, müssen Sie *alle* Dateien des Templates hochladen, nicht nur die, die Sie verändert haben.

**AUFGABE 7.5.** (*PDA → CFG*) 1 Punkte

In dieser Aufgabe sollen Sie *exakt* die Konstruktion aus Satz 4.60 implementieren, die einen PDA in eine äquivalente CFG umwandelt.

- Laden Sie das Codegerüst und die Angabe für die Hausaufgabe auf `https://www21.in.tum.de/teaching/theo/SS19/materials/ha/07/templates_ha07.tar.gz` herunter.
- Lesen Sie sich die PDF-Angabe zu der Aufgabe 'PDAToCFG' durch und folgen Sie den Anweisungen dort.

- Laden Sie dann für Problem D (PDAToCFG) alle benötigten Dateien hoch – falls Sie eines unserer Templates benutzen, müssen Sie *alle* Dateien des Templates hochladen, nicht nur die, die Sie verändert haben.