Prof. T. Nipkow
M. Eberl, J. Rädle, S. Wimmer
Technische Universität München
PROGRAMMIERABGABE **4 Punkte**
Abgabefrist: Montag, 20.05.2019, 14 Uhr

Einführung in die theoretische Informatik

Sommersemester 2019 – Hausaufgabenblatt 3

Programmierabgabe

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`).

AUFGABE 3.1. (*Automata Tutor*)

Im Automata Tutor finden Sie wieder bepunktete Hausaufgaben. Beachten Sie, dass Sie für diese Aufgaben weiterhin nur fünf Versuche haben. Die Aufgaben finden Sie unter den Kategorien "English to Regular Expression" und "Regular Expression to NFA". Beachten Sie, dass der AutomataTutor automatisch $c \mid \varepsilon$ als $c?$ abkürzt. 2 Punkte

AUFGABE 3.2. (*Operationen auf regulären Ausdrücken*)

In dieser Aufgabe implementieren Sie einge rekursive Prozeduren auf regulären Ausdrücken. 2 Punkte

- Lesen Sie sich die PDF-Angabe zu der Aufgabe 'RegexOperations' durch (zu finden unter 'course/problems' auf TUMjudge). **Achtung**: Lassen Sie sich nicht von den automatisch angehängten Ein-/Ausgabepaaren verwirren, da dort Unicode-Zeichen fehlen. Das Archiv für das Codegerüst enthält die richtigen Paare.
- Laden Sie das Codegerüst für die erste Hausaufgabe auf `https://www21.in.tum.de/teaching/theo/SS19/materials/ha/03/templates_ha03.tar.gz` herunter.
- Betrachten Sie die abstrakte Basisklasse `Regex` und ihre konkreten Kindklassen. Sie sollten die Attribute, insbesondere die dafür gewählten Datentypen, verstehen, bevor Sie fortfahren.
- Implementieren Sie alle mit `TODO` markierten Methoden in den Kindklassen von `Regex`.
- Laden Sie dann für Problem A (RegexOperations) 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 3.3. (*Regulärer Ausdruck zu ε-NFA*)

In dieser Aufgabe implementieren Sie die aus der Vorlesung bekannte Prozedur zur Übersetzung eines regulären Ausdrucks in einen ε-NFA, der dieselbe Sprache akzeptiert. 1 Punkt

- Lesen Sie sich die PDF-Angabe zu der Aufgabe 'RegexToEpsilonNFA' durch (zu finden unter 'course/problems' auf TUMjudge) **Achtung**: Lassen Sie sich nicht von den automatisch angehängten Ein-/Ausgabepaaren verwirren, da dort Unicode-Zeichen fehlen. Das Archiv für das Codegerüst enthält die richtigen Paare. Dort finden Sie auch graphische Versionen der Ausgabe als `.pdf`-Dateien.
- Laden Sie das Codegerüst für die erste Hausaufgabe auf `https://www21.in.tum.de/teaching/theo/SS19/materials/ha/03/templates_ha03.tar.gz` herunter.
- Betrachten Sie die abstrakte Klasse `Regex`, ihre konkreten Kindklassen und die Klasse `EpsilonNFA`. Sie sollten insbesondere den Konstruktor von `EpsilonNFA` und die in den Attributen von `EpsilonNFA` verwendeten Datentypen verstehen.
- Implementieren Sie alle mit `TODO` markierten Methoden in den Kindklassen von `Regex`.
- Laden Sie dann für Problem B (RegexToNFA) 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 3.4. (*Superstring-DFA*)

Ziel dieser Aufgabe ist es den minimalen DFA zu konstruieren, der die *Superstring-Sprache* zu einem Wort w akzeptiert (d.h. die Sprache aller Wörter, die w enthalten). 1 Punkt

- Lesen Sie sich die PDF-Angabe zu der Aufgabe 'SuperStringDFA' durch (zu finden unter 'course/problems' auf TUMjudge)
- Laden Sie das Codegerüst für die erste Hausaufgabe auf `https://www21.in.tum.de/teaching/theo/SS19/materials/ha/03/templates_ha03.tar.gz` herunter. Dort finden Sie auch graphische Versionen der Ausgabe als `.pdf`-Dateien.
- Betrachten Sie die Klassen `EpsilonNFA` und `DFA`. Sie sollten insbesondere den Konstruktor von `DFA` und die in den Attributen von `EpsilonNFA` verwendeten Datentypen verstehen.
- Implementieren Sie alle mit `TODO` markierten Methoden in der Klasse `SuperstringDFA`.
- Laden Sie dann für Problem C (SuperStringDFA) alle benötigten Dateien hoch – falls Sie eines unserer Templates benutzen, müssen Sie *alle* Dateien des Templates hochladen, nicht nur die, die Sie verändert haben.