Prof. J. Křetínský
M. Weininger, T. Meggendorfer

Technische Universität München

PROGRAMMIERABGABE **5+2 Punkte**

Abgabefrist: Montag, 25.06.2018, 10 Uhr

Einführung in die theoretische Informatik

Sommersemester 2018 – Hausaufgabenblatt 10

Programmierabgabe

Für die Korrektur Ihrer Programmierabgaben wird TUMjudge(`https://judge.in.tum.de/theo/public/`) verwendet. Auf dem ersten Hausaufgabenblatt werden 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`).

Laden Sie das Codegerüst für die zehnte Hausaufgabe auf `https://www7.in.tum.de/um/courses/theo/ss2018/homework/` herunter. Die `toDot()` Methode in `TuringMachine` produziert eine graphische Repräsentation der TM im Dot Format (`http://graphviz.org/`), dies kann beim debugging helfen.
Hinweis: Sie dürfen den Code des Templates verändern. Allerdings sollten Sie insbesondere die `main`-Methoden, Parser und `toString`-Methoden unverändert lassen.

AUFGABE 10.1. (*Simulate*) 1 Punkt

- Implementieren Sie die Methode `simulate(TuringMachine<S> tm, String word, int bound)` in der Klasse `Simulate`. Diese Methode soll die gegebene Turing Maschine `tm` auf dem Wort `word` für maximal `bound` Schritte simulieren und das Ergebnis zurückgeben, also `Result.ACCEPTED` falls die Maschine das Wort innerhalb des Schritt-Limits akzeptiert, `Result.STUCK` falls keine Transition mehr möglich ist und `Result.RUNNING` falls die Maschine nach `bound` Schritten noch läuft.
- Laden Sie dann für Problem A (Simulate) *alle* Dateien hoch, die für diese Aufgabe relevant sind.

AUFGABE 10.2. (*Rectangle*) 2 Punkt

- Implementieren Sie die Methode `rectangleMachine()` in der Klasse `Rectangle`. Die Methode soll eine Turing Maschine erzeugen, die die Rechteck-Sprache $\{\rightarrow^i\uparrow^j\leftarrow^i\downarrow^j \mid i,j>0\}$ auf dem Alphabet `>, ^, <, v` erkennt. Die Maschine darf maximal 15 Zustände verwenden und für Worte der Länge $n \leq 200$ maximal 1000000 Schritte für die Entscheidung benötigen.
  Hinweis: Die Musterlösung verwendet 8 Zustände und weniger als 10000 Schritte.
- Laden Sie dann für Problem B (Rectangle) *alle* Dateien hoch, die für diese Aufgabe relevant sind.

AUFGABE 10.3. (*ToEpsilon*) 2 Punkte

- Implementieren Sie die Methode `toEpsilon(TuringMachine<S> tm)` in der Klasse `ToEpsilon`. Die Methode soll, falls möglich, für eine *right-moving* Turing Maschine einen ε-NFA konstruieren, der die selbe Sprache erkennt, andernfalls soll `null` zurückgegeben werden. *Right moving* bedeutet, dass die Turing Maschine `tm` nur R Transitionen hat, also auf dem Band nur nach rechts geht.
- Sie dürfen davon ausgehen, dass die Turing Maschine keine ausgehendene Transitionen in Final-Zuständen hat.
- Laden Sie dann für Problem C (ToEpsilon) *alle* Dateien hoch, die für diese Aufgabe relevant sind.

Die folgenden Aufgabe ist eine Bonusaufgabe. Das heißt, Sie müssen diese nicht bearbeiten, um den vollen Notenbonus zu erhalten. **Hinweis:** Diese Aufgabe ist signifikant zeitaufwändiger als der Rest.

AUFGABE 10.4. (*Bonus: PrimeTM*) 2 Punkte

- Implementieren Sie die Methode `primeMachine()` in der Klasse `PrimeTM`. Die Methode soll eine Turing Maschine erzeugen, die die Sprache $\{a^p \mid p \text{ prim}\}$ erkennt. Die Maschine darf maximal 30 Zustände verwenden und für Worte der Länge $n \leq 50$ maximal 2500000 Schritte für die Entscheidung benötigen.
  Hinweis: Die Musterlösung verwendet 15 Zustände und weniger als 500000 Schritte.
- Laden Sie dann für Problem D (PrimeTM) *alle* Dateien hoch, die für diese Aufgabe relevant sind.