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

Technische Universität München
PROGRAMMIERABGABE **7 Punkte**
Abgabefrist: Freitag, 24.05.2018, 10 Uhr

Einführung in die theoretische Informatik

Sommersemester 2018 – Hausaufgabenblatt Lösungsskizze 5

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`).

Achtung: Nachdem auf diesem Blatt 7 Punkte erreicht werden können, haben Sie eine verlängerte Abgabefrist bis Freitag 25.05.2018.
Laden Sie das Codegerüst für die fünfte Hausaufgabe auf `https://www7.in.tum.de/um/courses/theo/ss2018/homework/` herunter.
Die vorimplementierten Klassen des Codegerüstes sind dieselben wie bei der vorigen Programmieraufgabe. Machen Sie sich erneut mit den Klassen State, Transition, EpsilonNFA, NFA, DFA und Parser vertraut.
Hinweis: Sie dürfen den Code des templates verändern. Allerdings sollten Sie main-Methode und parser so lassen, wie sie sind.

AUFGABE 5.1. (*IsFinite*)

2 Punkte

- Implementieren Sie die Methode `isFinite(DFA d)` in der Klasse IsFinite. Die Methode soll für einen DFA d `true` zurück geben, wenn die Sprache des DFA endlich ist, also $|L(d)| < \infty$, ansonsten `false`.
- Laden Sie dann für Problem A (isFinite) *alle* Dateien hoch, die für diese Aufgabe relevant sind, also State, Transition, DFA, NFA, EpsilonNFA, Parser und IsFinite.

AUFGABE 5.2. (*IsEmpty*)

1 Punkt

- Implementieren Sie die Methode `isEmpty(DFA d)` in der Klasse IsEmpty. Die Methode soll für einen DFA d `true` zurück geben, wenn die Sprache des DFA leer ist, also $L(d) = \emptyset$, ansonsten `false`.
- Laden Sie dann für Problem B (isEmpty) *alle* Dateien hoch, die für diese Aufgabe relevant sind, also State, Transition, DFA, NFA, EpsilonNFA, Parser und IsEmpty.

AUFGABE 5.3. (*Intersection*)

2 Punkte

- Implementieren Sie die Methode `intersection(DFA d1, DFA d2)` in der Klasse Intersection. Die Methode soll für zwei DFAs $d1$ und $d2$ einen DFA zurück geben, der die Schnittmenge der Sprachen von $d1$ und $d2$ akzeptiert, also $L(d) = L(d1) \cap L(d2)$.
  Wenden Sie die Produktkonstruktion an.
  Sie dürfen davon ausgehen, dass die Alphabete der beiden Automaten gleich sind.
  Hinweis: Ihren Code für diese Aufgabe können Sie vermutlich für Problem D (Operations) wiederverwenden und erweitern. Die Klasse java.util.function.BinaryOperator könnte nützlich sein.
- Laden Sie dann für Problem C (Intersection) *alle* Dateien hoch, die für diese Aufgabe relevant sind, also State, Transition, DFA, NFA, EpsilonNFA, Parser und Intersection.
- Wie bei der Potenzmengenkonstruktion wird Ihr Ergebnis wieder auf Sprachgleichheit getestet. Deswegen beginnt der Input mit der Anzahl der Zeilen und der Output mit "Case 1:". Beides wird von der main-methode bereits übernommen.

AUFGABE 5.4. (*Operations*)

1 Punkt

- Implementieren Sie in der Klasse Operations die Methoden
  – `union(DFA d1, DFA d2)`
  – `setminus(DFA d1, DFA d2)`
  – `xor(DFA d1, DFA d2)`
  Die Methoden soll für zwei DFAs $d1$ und $d2$ einen DFA zurück geben, der
  – die Vereinigung der Sprachen von $d1$ und $d2$ akzeptiert, also $L(d) = L(d1) \cup L(d2)$.
  – die Mengendifferenz der Sprachen von $d1$ und $d2$ akzeptiert, also $L(d) = L(d1) \setminus L(d2)$.
  – die symmetrische Differenz der Sprachen von $d1$ und $d2$ akzeptiert, also $L(d) = L(d1) \Delta L(d2) = (L(d1) \setminus L(d2)) \cup (L(d2) \setminus L(d1))$.

Wenden Sie die Produktkonstruktion an.
Sie dürfen davon ausgehen, dass die Alphabete der beiden Automaten gleich sind.
Hinweis: Sie können Ihren Code Problem C (Intersection) wiederverwenden und erweitern. Die Klasse java.util.function.BinaryOperator könnte nützlich sein.

- Laden Sie dann für Problem D (operations) *alle* Dateien hoch, die für diese Aufgabe relevant sind, also State, Transition, DFA, NFA, EpsilonNFA, Parser und Operations.

AUFGABE 5.5. (*Equivalent*) 1 Punkt

- Implementieren Sie in der Klasse Equivalent die Methode `equivalent(DFA d1, DFA d2)` Die Methoden soll für zwei DFAs $d1$ und $d2$ `true` zurück geben, wenn die beiden dieselbe Sprache akzeptieren, also $L(d1) = L(d2)$, ansonsten false.
  Sie dürfen davon ausgehen, dass die Alphabete der beiden Automaten gleich sind.
  Hinweis: Es ist sinnvoll, Ihren Code aus den vorherigen Problemen wieder zu verwenden.
- Laden Sie dann für Problem E (Equivalent) *alle* Dateien hoch, die für diese Aufgabe relevant sind, also State, Transition, DFA, NFA, EpsilonNFA, Parser und Equivalent.