PROGRAMMIERABGABE
Abgabefrist: Mittwoch, 20.06.2018, 10 Uhr

Einführung in die theoretische Informatik

Sommersemester 2018 – Hausaufgabenblatt 9

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 neunte Hausaufgabe auf `https://www7.in.tum.de/um/courses/theo/ss2018/homework/` herunter.
Die vorimplementierten Klassen des Codegerüstes ähneln denen aus den vorigen Programmieraufgaben. Um die folgenden Aufgaben einfacher zu gestalten, wurden jedoch einige Änderungen vorgenommen.
Hinweis: Sie dürfen den Code des Templates verändern. Allerdings sollten Sie insbesondere die `main`-Methode, den Parser und die `toString`-Methoden unverändert lassen.

AUFGABE 9.1. (*AcceptPDA*)

2 Punkte

- Implementieren Sie die Methode `accept(PDA m, String w)` in der Klasse AcceptPDA. Die Methode soll für einen PDA und ein Wort `true` zurück geben, wenn der PDA das Wort akzeptiert, ansonsten `false`. Falls das Wort Buchstaben enthält, die nicht im Alphabet des PDA sind, soll ebenfalls `false` zurück geben werden.
- Laden Sie dann für Problem A (AcceptPDA) *alle* Dateien hoch, die für diese Aufgabe relevant sind.

AUFGABE 9.2. (*Palindrome*)

1 Punkt

- Implementieren Sie die Methode `printPalindromePDA()` in der Klasse Palindrome. Diese Methode soll einen String zurück geben, der einen PDA beschreibt, der die Sprache aller Palindrome über dem Alphabet $\{0, 1\}$ erkennt. Dieser String soll mit "Case #1:\n" beginnen und muss vom PDAParser gelesen werden können.
- Laden Sie dann für Problem B (Palindrome) *alle* Dateien hoch, die für diese Aufgabe relevant sind.

AUFGABE 9.3. (*PDAPrefix*)

2 Punkte

- Implementieren Sie die Methode `pdaPrefix(PDA m)` in der Klasse PDAPrefix. Die Methode soll für einen PDA m einen PDA m' zurück geben, der die Präfixsprache von m akzeptiert, also $\mathsf{L}(m') = \{u \mid \exists v \in \Sigma^* : uv \in \mathsf{L}(m)\}$.
- Laden Sie dann für Problem C (PDAPrefix) *alle* Dateien hoch, die für diese Aufgabe relevant sind.

Die folgenden Aufgaben sind Bonusaufgaben. Das heißt, Sie müssen diese nicht bearbeiten, um den vollen Notenbonus zu erhalten.

AUFGABE 9.4. (*Bonus: CNF*)

1 Punkte

- Implementieren Sie die Methode `cnf(Grammar g)` in der Klasse CNF. Die Methode soll für eine Grammatik g die nach Vorlesungsalgorithmus erzeugte Grammatik g' in Chomsky-Normalform erzeugen. **Hinweis:** Diese Aufgabe ist signifikant zeitaufwändiger als der Rest.
- Sie dürfen davon ausgehen, dass die Grammatik g eine kontextfreie Sprache erzeugt und dass auf der linken Seite der Produktionen immer genau ein Nichtterminal steht.
- Verwenden sie außerdem die Methoden `cnfLiteralName` und `cnfChainName` in `Util`, um die Namen für die neuen Nichtterminale in Schritt (1) und (2) zu erzeugen. Sie dürfen annehmen, dass diese Namen neu sind, also nicht mit den bereits existierenden (Nicht)Terminalen kollidieren.
- Laden Sie dann für Problem D (CNF) *alle* Dateien hoch, die für diese Aufgabe relevant sind.

AUFGABE 9.5. (*Bonus: CYK*)

1 Punkte

- Implementieren Sie die Methode `cyk(Grammar g, String w)` in der Klasse CYK. Die Methode soll die CYK-Tabelle, also ein `CYKTable`-Objekt, für das Wort w und die Grammatik g erzeugen.
- Sie dürfen davon ausgehen, dass die eingegebene Grammatik in Chomsky-Normalform ist und die zu testenden Worte nicht leer sind.
- Laden Sie dann für Problem E (CYK) *alle* Dateien hoch, die für diese Aufgabe relevant sind.