RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 1
Rossmanith–Reidl–Sanchez 14.4.2015

Algorithmen für endliche Automaten und kontextfreie Grammatiken

Aufgabe 1

Richten Sie ein funktionierendes `git`-Repository ein, auf welches alle TeilnehmerInnen Ihrer Gruppe (und wir) zugreifen können.

Einigen Sie sich auf Standards für die Programmierung, den verwendeten Stil und die Art der Dokumentation.

Sorgen Sie dafür, daß auch alle auf die Entwicklungswerkzeuge, welche Sie verwenden wollen, Zugriff haben und alle das gesamte Projekt übersetzen können. Jeder sollte über einen Computer verfügen können, auf welchem alle Arbeitsschritte für dieses Praktikum durchgeführt werden können. Zu den Donnerstagsterminen sollten alle mithilfe eines tragbaren Computers alle Aspekte der Implementierungen der ganzen Gruppe vorführen können. Bitte sorgen Sie auch dafür, daß die Verbindung mit dem Videoprojektor im Hörsaal funktioniert.

Aufgabe 2

Erstellen Sie Klassen für Wörter und Alphabete. Ein Alphabet sollte dabei, wie üblich, eine endliche Menge von Symbolen darstellen. Die Symbole sollen einfach UTF-8-Zeichenketten sein. Erstellen Sie geeignete Methoden für die Alphabetklasse, um damit gut umgehen zu können und auch Tests bequem durchführen zu können.

Ein Alphabet könnte also zum Beispiel aus allen ASCII-Zeichen ausser `[`, `\` und zusätzlich aus `[id]`, `[mod]`, `[keyword]`, `[numeral]`, `\[` und `\\` bestehen. Beachten Sie, daß diese Strings einen präfixfreien Code darstellen, eine wichtige Eigenschaft, die Ihre Alphabetklasse (effizient) sicherstellen sollte. Überlegen Sie sich auch ein gutes Konzept, Fehler zu behandeln. Ein Fehler wäre es zum Beispiel, die Symbole `id` *und* `identifier` in einem Alphabet verwenden zu wollen.

Enthält ein Alphabet n verschiedene Symbole, dann sollten diese von 0 bis $n-1$ durchnumeriert sein: Jedes Symbol hat so eindeutig seine Nummer.

Eine Wortklasse sollte ein Wort über einem gegebenen Alphabet darstellen. Es sollte bequem möglich sein ein solches Wort zu erzeugen, zum Beispiel durch Angabe von einem String wie `ab[id][mod]x\[uvw]*` unter Verwendung des oben angegebenen Alphabets. Dieses Wort enthielte dann genau 11 Symbole.

Es sollte einfach sein, ein Wort aus einer Datei oder der Standardeingabe einzulesen und es auf schöne Weise auszugeben. Sehr wichtig für effiziente Behandlung von Wörtern ist eine Methode, welche ein Array von **int**s zurückgibt, welches die Nummern der einzelnen Symbole des Worts enthält.

Implementieren Sie die Wortklasse *immutable*, so daß ein Wort niemals verändert werden kann. Dennoch sollten Operationen wie das Aneinanderfügen von zwei Worten sehr effizient umgesetzt werden können.

Aufgabe 3 (bis 23. April)
Entwerfen und implementieren Sie ein System von Unittests, welche alle Klassen und Methoden umfangreich testen.

Aufgabe 4 (bis 23. April)
Entwerfen Sie eine Klasse für deterministische endliche Automaten. Überlegen Sie, welche einfache Methoden für die Konstruktion, Ein- und Ausgabe von DFAs geeignet sind.
Die erste wichtige Methode, welche Sie implentieren sollen, ist die Lösung des Wortproblems: Akzeptiert oder verwirft der DFA ein gegebenes Wort? Diese Methode sollte so effizient wir irgend möglich implementiert werden.

Aufgabe 5
Einigen Sie sich unter allen drei Gruppen auf ein Ein- und Ausgabeformat für Alphabete, Wörter und deterministische endliche Automaten. Dateien in diesem Format sollten dann von allen Gruppen korrekt gelesen werden können.