Technische Universität München
Fakultät für Informatik
Prof. Dr. M. Broy

SS 2003
Übungsblatt 1
21. April 2003

Übungen zur Vorlesung Einführung in die Informatik IV

Aufgabe 1 Relationen

Gegeben sei die folgende binäre (zweistellige) Relation über der Grundmenge **N** der natürlichen Zahlen, die "Nachfolger-Relation"

$$R' = \{(n, n+1) : n \in \mathbf{N}\}$$

Wir betrachten im Folgenden die endliche Einschränkung R dieser Relation auf die Grundmenge $M = \{0, \ldots, 3\}$

(a) Stellen Sie die Relation R mit Hilfe eines Pfeildiagramms und einer Adjazenzmatrix dar.

(b) Stellen Sie die folgenden Relationen mit Hilfe von Adjazenzmatrizen dar und geben Sie jeweils an, um welche Relation es sich handelt.

Geben Sie an, welche dieser Relationen reflexiv, symmetrisch, antisymmetrisch, asymmetrisch, transitiv, irreflexiv, linkseindeutig, rechtseindeutig, linkstotal, rechtstotal, partielle Ordnungen, lineare Ordnungen, Äquivalenzrelationen, partielle und totale Funktionen sind.

(i) die konverse (inverse) Relation R^T
(ii) das Relationenprodukt $R \circ R$
(iii) die transitive Hülle R^+
(iv) reflexive und transitive Hülle R^*
(v) die symmetrische Hülle R^{sym}
(vi) die symmetrische transitive reflexive Hülle $R^{\otimes}$
(vii) **(H)** das Komplement der transitiven Hülle $(R^+)^-$
(viii) **(H)** das Komplement der reflexiven und transitiven Hülle $(R^*)^-$
(ix) **(H)** den Durchschnitt $R^* \cap (R^*)^T$
(x) **(H)** die Vereinigung $R^+ \cup (R^+)^T$

Aufgabe 2 Textersetzungssysteme, Markovalgorithmen

Ziel dieser Aufgabe ist die Einübung des Arbeitens mit Textersetzungssystemen, und zwar sowohl die Analyse als auch die Entwicklung von Systemen betreffend. Gegeben sei das folgende nichtdeterministische Textersetzungssystem T für die Manipulation von Wörtern über dem Alphabet {0,1,L,R}:

(1) `R0` → `1R`
(2) `R` → `L`
(3) `1L` → `0L`
(4) `10L` → `0R0`
(5) `0L` → `L0`

(a) Zeigen Sie, dass jede Berechnung in T für alle Eingaben terminiert?

(b) Wir interessieren uns nun für Eingabewörter der Form $w \circ \mathtt{L}$, wobei w ein Wort über dem Alphabet $\{\mathtt{0},\mathtt{1}\}$ sei. Ist die Ausgabe für alle Eingaben der genannten Form determiniert, d.h. sind die terminalen Wörter aller Berechnungssequenzen für ein Eingabewort identisch? Beweis.

Mit Hilfe von Textersetzungssystemen kann man auch formale Sprachen definieren. Wir benutzen dazu das Textersetzungssystem T.

(c) Geben Sie die Menge aller Wörter über dem Alphabet $\{\mathtt{0},\mathtt{1},\mathtt{L},\mathtt{R}\}$ an, für die eine Berechnungssequenz in T existiert, in deren terminalem Wort (mindestens einmal) $\mathtt{1}$ vorkommt.

Wir betrachten nun T unter der Markovanwendungsstrategie, d.h. die Regelanwendung wird deterministisch.

(d) Geben Sie die Berechnungssequenz für das Eingabewort $\mathtt{111L}$ an.

(e) Was lässt sich über die Länge der Berechnungssequenz für ein Eingabewort $w \circ \mathtt{L}$ mit w über $\{\mathtt{0},\mathtt{1}\}$ sagen.

[Optionale Aufgabe] Schliesslich wenden wir uns der Erstellung eines Textersetzungssystems unter Markovanwendungsstrategie zu. Die Aufgabe ist aus einer Binärzahl die entsprechende Unärzahl zu berechnen.

(f) Entwickeln Sie ein Textersetzungssystem, das jedes Wort $w \circ \mathtt{L}$ mit w über $\{\mathtt{0},\mathtt{1}\}$ abbildet in ein Wort $\mathtt{L}\underbrace{\mathtt{1}\cdots\mathtt{1}}_{k \text{ mal}}$, wobei k der Wert von w als Binärzahl ist (d.h. $\mathtt{101L}$ wird auf $\mathtt{L11111}$ abgebildet).

Tip: Bauen Sie auf dem oben angegebenen System T auf!

Aufgabe 3 (P) Reguläre Sprache, Verarbeitung strukturierter Informationen

Die Übernahme, Überprüfung und Weiterverarbeitung strukturierter Informationen repräsentiert eine typische Aufgabe der praktischen Programmierung. Als Anwendungsbeispiel hierfür soll ein Ausschnitt eines EMail-Clients entwickelt werden, der eine EMail-Nachricht aus einer gegebenen Datei einliest, hinsichtlich ihres erwarteten Aufbaus überprüft und anschließend in einer übersichtlichen Formatierung wieder auf der Konsole ausgibt.

Eine so behandelte Nachricht beinhaltet gemäß folgender Reihenfolge

- genau einen Absender, dessen Adresse nach dem Schlüsselwort **from:** angegeben wird,
- ein oder mehrere Empfänger, deren Adresse jeweils auf das Schlüsselwort **to:** folgt,
- beliebig viele (d.h. möglicherweise auch keine) Empfänger von Kopien, deren Adresse jeweils nach dem Schlüsselwort **cc:** angegeben wird,
- genau einen Titel, dessen nicht-leere Zeichenkette auf das Schlüsselwort **subject:** folgt,
- sowie den eigentlichen Nachrichtentext, der nach dem Schlüsselwort **body:** angegeben werden kann.

Hierbei entspricht eine verwendete EMail-Adresse der üblichen Konvention, setzt sich also aus zwei Bezeichnern für Name und Domäne zusammen, welche durch das Zeichen @ getrennt werden. Weiterhin ist aus Gründen der Vereinfachung anzunehmen, dass kein Schlüsselwort in einer Adresse, Titel oder Text der Nachricht verwendet wird.

Dementsprechend repräsentiert beispielsweise

```
from:    student@in.tum.de
to:      info4@mailbroy
cc:      archive@localhost
cc:      student@home.de
subject: Meine Hausaufgabe
body:    Hier also meine Lösung zu ...
```

eine nach den oben angegebenen Regeln gültige EMail-Nachricht, während etwa

```
from:    student
cc:      archive@localhost, student@home.de
subject:
body:    Hier also meine Lösung zu ...
```

zurückzuweisen ist, da die Absenderadresse keine gültige Form aufweist, der Adressat nicht aufgeführt wird, die Empfänger der Kopien nicht jeweils mit eigenem Schlüsselwort eingeleitet werden und schließlich der Nachrichtentitel nicht angegeben wird.

Implementieren Sie in Java zur Umsetzung der Aufgabenstellung

(a) eine Klasse `Message`, mit deren Attributen und Zugriffsmethoden eine gültige EMail-Nachricht vollständig repräsentiert werden kann. Darüber hinaus soll die generische `toString()`-Methode überladen werden, um eine möglichst übersichtlich formatierte Ausgabe der gesamten Nachricht zu gewährleisten.

(b) Entwickeln Sie eine Klasse `Mailer`, der zur Laufzeit eine Textdatei mit einer gültigen oder ungültigen EMail-Nachricht übergeben werden kann. Diese Datei ist einzulesen, gemäß den oben angegebenen Regeln zu überprüfen und deren Inhalt auf eine Instanz der Klasse `Message` zu übertragen. Anschließend wird die Nachricht über einen Aufruf ihrer `toString()`-Methode wieder auf der Konsole ausgegeben. Bei der Verarbeitung ungültiger Nachrichten sind aussagekräftige Fehlermeldungen anzugeben und dennoch möglichst viel Inhalt zu übernehmen.

(c) Dokumentieren Sie mindestens 5 verschiedene Testfälle mit unterschiedlichen Varianten gültiger und ungültiger Nachrichten, deren Bearbeitung eine korrekte Implementierung der Klasse `Mailer` belegt.

Hinweis: Die Java-Klassen `java.io.FileReader` und `java.io.BufferedReader` eignen sich für das zeilenweise Einlesen einer Datei, während `java.util.StringTokenizer` die Zerlegung einer Zeichenkette in einzelne Worte erleichtert. Informieren Sie sich mit Hilfe der Javadoc Online Dokumentation (z.B. unter `http://java.sun.com/j2se/1.3/docs/api/`) über den Gebrauch der oben genannten Klassen.

	R	R^T	$R \circ R$	R^+	R^*	R^{sym}
reflexiv						
symmetrisch						
antisymmetrisch						
asymmetrisch						
transitiv						
irreflexiv						
linkseindeutig						
rechtseindeutig						
linkstotal						
rechtstotal						
partielle Ordnung						
lineare Ordnung						
Äquivalenzrelation						
partielle Funktion						
totale Funktion						