Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wölfl, Thorsten Engesser
Wintersemester 2015/2016

Universität Freiburg
Institut für Informatik

Übungsblatt 3

Abgabe: Freitag, 13. November 2015, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet03` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann entsprechend dem ersten Übungsblatt in Dateien in diesem Unterverzeichnis erwartet.

Beachten Sie bitte bei allen Aufgaben die *Hinweise zur Bearbeitung der Übungsaufgaben* unter der folgenden URL:

`http://gki.informatik.uni-freiburg.de/teaching/ws1516/info1/wiki/hinweise.html`

Überprüfen Sie, dass Sie alle Lösungen ins Repository hochgeladen haben (z.B. mit dem Befehl `svn status`). Überprüfen Sie auch die Webseite Ihres SVN-Unterverzeichnisses:

`https://daphne.informatik.uni-freiburg.de/ws1516/InformatikI/svn/$RZLOGIN`

Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 3.1 (Automaten; Dateien: (a) `automaton-diagram.jpg` oder `.pdf`, (b) `automaton.py`, (c) `automaton-test.txt`, (d) `automaton2.txt`; Punkte: 2+5+2+1)

Wir wollen einen Getränkeautomaten als Moore-Automaten implementieren. Wir nehmen an, dass der Getränkeautomat lediglich 50-Cent- und 1-Euro-Münzen annimmt. Ab einem Guthaben von 2,00 € werden keine weiteren Münzen mehr angenommen. Wird ein Getränk gewählt, das weniger kostet als das momentan vorhandene Guthaben, bleibt der Restbetrag erhalten (es gibt keine Geldrückgabe). Die folgenden Getränke stehen zur Verfügung: Cola für 1,50 € und Energy Drink für 2,00 €. Die Getränke können erst ausgewählt werden, wenn bereits entsprechendes Guthaben vorhanden ist. Wir nehmen an, dass Getränke unbeschränkt zur Verfügung stehen.

Für die Modellierung des Getränkeautomaten als Moore-Automaten verwenden wir insgesamt vier Eingabesymbole: zwei für den Einwurf von 50ct bzw. 1 €-Münzen (z.B. `inp50`, `inp100`) und zwei weitere für die Getränkeauswahl (z.B. `reqCoke`, `reqEnergy`). Als Ausgabealphabet verwenden wir Symbole für das Guthaben (z.B. `0ct`, `50ct`, `100ct`, ...), die beim Einwurf eines Geldstücks ausgegeben werden, sowie Symbole zur Rückgabe des Getränks (z.B. `COKE` oder `ENERGY`).

(a) Formalisieren Sie den oben beschriebenen Automaten. Überlegen Sie dazu zunächst, in welchen Zuständen der Automat sein kann und welche Nachfolgezustände jeder Zustand hat. Zeichnen Sie dann das Übergangsdiagramm für den Moore-Automaten.

Sie dürfen das Diagramm von Hand anfertigen und abfotografieren/einscannen oder auch gerne ein dafür geeignetes Zeichenprogramm verwenden. Erlaubte Dateiformate für die Abgabe sind ausschließlich JPEG- und PDF-Dateien.

(b) Implementieren Sie den Automaten in Python analog zu der Implementierung des Würfelautomaten aus der Vorlesung. Benutzen Sie dazu das auf der Übungsseite bereitgestellte Template. Eingaben und Ausgaben werden dabei direkt auf der Konsole getätigt. Ein möglicher Testlauf könnte wie folgt aussehen:

```
>>> import automaton.py
>>> automaton()
> inp50
50ct
> inp100
150ct
> inp100
250ct
> inp50
250ct
> reqCoke
COKE
> inp50
150ct
>
```

(c) Führen Sie einen ausführlichen Testlauf ihres Automaten durch, der mittels drei aufeinanderfolgender Getränkeausgaben die Funktionsweise des Automaten aufzeigt. Speichern Sie den Testlauf in der Datei `automaton-test.txt`.

(d) Überlegen Sie sich, wie Sie den Automaten anpassen müssten, um die zusätzliche Funktionalität einer Geldrückgabetaste zu implementieren. Wie viele neue Zustände müssten Sie einführen und wie würde die Übergangsfunktion dann aussehen? Abgabe in `automaton2.txt`.

Aufgabe 3.2 (Gefangenendilemma; Dateien: (a) `prisoner-description.txt`, `prisoner-diagram.{jpg|pdf}`, (b) `prisoner.py`; Punkte: 3+3+2)

In der Vorlesung wurde das wiederholte Gefangenendilemma sowie einige Beispielstrategien vorgestellt. In dieser Aufgabe geht es darum, eine Strategie Ihrer Wahl (z.B. eine der Beispielstrategien oder eine eigene Strategie) als endlichen Automaten zu implementieren. Die Ausgabefunktion soll jeweils die zu wählende Aktion (`"C"` für cooperate oder `"D"` für defect) ausgeben. Die Übergangsfunktion wählt den nächsten Zustand des Automaten in Abhängigkeit vom eigenen gegenwärtigen Zustand und der gerade gespielten Aktion des Mitspielers (ebenfalls `"C"` oder `"D"`) aus. Gehen Sie davon aus, dass jede neue Runde mit der Wahrscheinlichkeit 1/20 die letzte Runde ist.

(a) Beschreiben Sie kurz die zugrundeliegende Idee Ihrer Strategie in der Datei `prisoner-description.txt`.

Skizzieren Sie analog zur Aufgabe 3.1 den Automaten als Übergangsdiagramm (Datei: `prisoner-diagramm.jpg` oder `prisoner-diagramm.pdf`).

(b) Implementieren Sie den Moore-Automaten Ihrer Strategie. Benutzen Sie dazu das Template `prisoner.py` von der Webseite der Übungen. Sie müssen nur die beiden Funktionen `next_state(state, input)` und `output(state)` implementieren. Beachten Sie dabei, dass der Startzustand des Automaten die `0` ist. Indem Sie das Skript mittels `python3 prisoner.py` in der Konsole starten, können Sie einen vordefinierten Benchmark ausführen.

(c) Implementieren Sie eine weitere Strategie mittels der Funktionen `next_state_opponent(state)` und `output_opponent(state)`, um diese mit der in Aufgabe 3.2(b) definierten Strategie zu vergleichen. Achten Sie darauf, dass die beiden Strategien sich unterscheiden. Wenn Sie diese Teststrategie implementiert haben, wird sie beim Aufruf von `python3 prisoner.py` ebenfalls zum Benchmarken verwendet.

Wir werden die von Ihnen eingereichten Strategie-Implementierungen aus Aufgabenteil 3.2(b) paarweise gegeneinander antreten lassen. Die Implementierung mit der besten Strategie (d.h. den wenigsten gesamt akkumulierten Jahren in Gefangenschaft) erhält einen Preis.

Aufgabe 3.3 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet03` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Interessantes, etc.).