Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Informatik

Algorithmen & Datenstrukturen (AD)

Übung: Endliche Automaten und formale Sprachen (A3)

Themen: Automaten, formale Grammatik/Sprache, Wortproblem, EBNF, Syntaxdiagramme, EA, DEA, NEA, (State Design Pattern).
Zeitbedarf: ca. 300min.

Hansjörg Diethelm, Version 1.00 (FS 2017)

1 Automat – Maschinensteuerung (ca. 20')

1.1 Lernziel

- Ein vorgegebenes Verhalten mit Hilfe eines Automaten beschreiben, und zwar als Graph und als Tabelle.

1.2 Grundlagen

Diese Aufgabe basiert auf dem AD-Input "A31_IP_Automaten". Die Aufgabe beinhaltet keine Programmierung.

1.3 Aufgabe

Zeichnen Sie für folgende Maschine einen Automaten auf. Die Maschine wird durch Kontrollsignale gesteuert:

- Vom **Initialzustand** geht sie in den Zustand **running**, wenn das Signal *start* kommt.
- Sie bleibt im Zustand **running**, solange das Signal *okay* kommt.
- Wenn das Signal *problem* kommt, geht sie in den Zustand **repair**. Ein Signal *okay* bringt sie in den Zustand **running** zurück. Sofern es erneut ein Signal *problem* gibt, geht die Maschine in den Zustand **failure**.
- Diesen Zustand erreicht sie auch, sofern im Zustand **running** ein Signal *failure* auftritt.
- Ein Signal *reset* bringt die Maschine vom Zustand **failure** in den **Initialzustand** zurück.

2 Automat – Garagentorsteuerung (ca. 30')

2.1 Lernziele

- Das Verhalten einer Steuerung mit Hilfe eines Automaten modellieren.
- Einen adäquaten Automaten aufzeichnen.

2.2 Grundlagen

Diese Aufgabe basiert auf dem AD-Input "A31_IP_Automaten". Die Aufgabe beinhaltet keine Programmierung.

2.3 Aufgaben

Mit Hilfe eines Automaten wollen wir das Verhalten einer Garagen-Steuerung modellieren. Beim Öffnen werde das Tor motorisch seitwärts verschoben. Im Tor befinde sich zudem eine Personen-Türe. Natürlich darf sich das Tor nicht bewegen, falls diese Türe geöffnet ist.

Wir unterscheiden folgende **Zustände**:

- **zu** Tor ist zu.
- **offen** Tor ist offen.
- **öffnend** Tor ist öffnend in Bewegung.
- **schliessend** Tor ist schliessend in Bewegung.
- **stop** Bewegung ist gestoppt.

Weiter haben wir es mit folgenden *Eingaben* bzw. Signalen zu tun:

- *befehl_auf* Ausgelöst durch Fernbedienung, Schlüssel oder Ausfahrts-Sensor.
- *offenzeit_abgelaufen* Ausgelöst durch Timer.
- *tor_ist_zu* Ausgelöst durch Offen-Sensor.
- *tor_ist_offen* Ausgelöst durch Zu-Sensor.
- *türe_geöffnet* Personen-Türe im beweglichen Gargentor ist offen.
- *türe_geschlossen* Personen-Türe im beweglichen Garagentor ist zu.

a) Zeichnen Sie einen Automaten als Graphen auf, welcher eine vernünftige Garagentor-Steuerung beschreibt. Definieren Sie einen sinnvollen Startzustand.

b) Optional: Erweitern Sie die Steuerung noch mit zusätzlichen Zuständen und Eingaben.

3 Formale Grammatik (ca. 20')

3.1 Lernziele

- Den prinzipiellen Aufbau einer formalen Grammatik verstehen.
- Mit Hilfe von Produktionen Wörter erzeugen können.
- Verstehen, was es mit einer kontextsensitiven Grammatik auf sich hat.

3.2 Grundlagen

Diese Aufgabe basiert auf der Folie Beispiel einer Typ 1 Grammatik G_1 aus dem AD-Input "A32_IP_FormaleSprachen". Die Aufgabe beinhaltet keine Programmierung.

3.3 Aufgaben

a) Studieren Sie nochmals das Erzeugen des Wortes "aabbcc" aus dem Input.

b) Erzeugen Sie jetzt analog das Wort "aaabbbccc".

c) Weshalb spricht man hier von kontextsensitiv?

4 Formale Grammatik (ca. 45')

4.1 Lernziele

- Den prinzipiellen Aufbau einer formalen Grammatik verstehen.
- Mit Hilfe von Produktionen Wörter erzeugen können.
- Den Typ einer Grammatik bestimmen.
- Alternative Beschreibungsformen einsetzen.

4.2 Grundlagen

Diese Aufgabe basiert auf dem AD-Input "A32_IP_FormaleSprachen". Die Aufgabe beinhaltet keine Programmierung.

4.3 Aufgaben

Gegeben sei folgende formale Grammatik G:

$N = \{s, A, B\}$

$T = \{0, 1, 2\}$

$P = \{s \rightarrow A,\ A \rightarrow \varepsilon,\ A \rightarrow B,\ A \rightarrow 0A0,\ A \rightarrow 1A1,\ A \rightarrow 2A2,\ B \rightarrow 0,\ B \rightarrow 1,\ B \rightarrow 2\}$

s

a) Erzeugen Sie mit Hilfe der Produktionen 4 verschieden lange Wörter.

b) Versuchen Sie, die Sprache L(G) in Prosa zu definieren.

c) Von welchem Typ ist die Grammatik G?

d) Mit welchen anderen formalen Beschreibungsformen könnte man demnach G auch definieren?

e) Definieren Sie G gemäss d) mindestens noch mit einer anderen Beschreibungsform.

5 EBNF (ca. 15')

5.1 Lernziele

- EBNF korrekt interpretieren.

5.2 Grundlagen

Diese Aufgabe basiert auf dem AD-Input "A32_IP_FormaleSprachen". Die Aufgabe beinhaltet keine Programmierung.

5.3 Aufgaben

Welche der untenstehenden Wörter lassen sich nicht mit folgender EBNF-Definition erzeugen:

<Sprache> ::= <Vorspann>11<Nachspann>

<Vorspann> ::= [01{0}]

<Nachspann> ::= {0[00|11]}

a) 11

b) 01110

c) 01111

d) 01011010

e) 0111000

f) 1100

g) 0011

h) 01001111

6 Syntaxdiagramme (ca. 60')

6.1 Lernziele

- Syntaxdiagramme lesen, interpretieren und definieren.

6.2 Grundlagen

Diese Aufgabe basiert auf dem AD-Input "A32_IP_FormaleSprachen". Die Aufgabe beinhaltet keine Programmierung.

6.3 Aufgaben

a) Betrachten Sie die Folie Beispiel Syntaxdiagramm "expression" aus dem Input. Das Syntaxdiagramm definiert, wie ein "expression" aufgebaut ist. Geben Sie entsprechend ein paar zulässige "expression" an.

b) Auf der Website

http://cui.unige.ch/db-research/Enseignement/analyseinfo/JAVA/BNFindex.html

wird die Definition der Sprache Java anhand von EBNF und von Syntaxdiagrammen aufgezeigt. Es wird zwar nicht die neuste Java-Version definiert, die Website ist aber dennoch sehr illustrativ. Steigen Sie doch mal unter **index of rules** mit **type_declaration** in die Definition einer Java-Klasse bzw. eines Java-Klassentyps ein …

Weiternavigieren können Sie dann z.B. mit **class_declaration**, **field_declaration**, **method_declaration** usw.

Anmerkung: Die offizielle Java Language Specification finden Sie hier:

http://docs.oracle.com/javase/specs/jls/se8/html/index.html

c) Wir betrachten nun folgende Sprache L:

A = {0,1}

L = {0,11, 011, 110, 0110, 111101100011, ... }

Jede 1er-Gruppe besitzt eine gerade Anzahl Einer.
Jede 0er-Gruppe besitzt eine ungerade Anzahl Nullen.

Definieren Sie L mit Hilfe eines Syntaxdiagramms.

7 Wortproblem mit Hilfe eine DEA lösen (50')

7.1 Lernziele

- DEA implementieren.
- Wortproblem mit Hilfe eines DEA lösen.

7.2 Grundlagen

Diese Aufgabe basiert auf den AD-Inputs "A32_IP_FormaleSprachen" und "A33_IP_ RegulaereSprachenEndlicheAutomaten".

7.3 Aufgabe

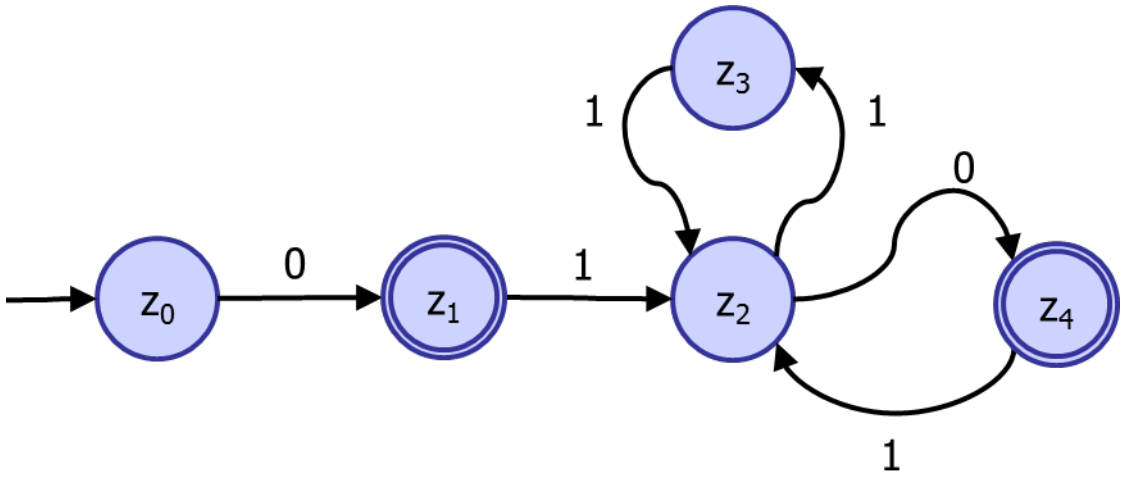
Gegeben sei folgende Sprache L:

A = {0,1}

L = {0, 010, 01110, 0111010,0101110, 0101010, ... }

Alle Wörter von L beginnen mit einer einzelnen Null. Anschliessend können Einer und Nullen abwechselnd folgen. Eine Null steht immer alleine; Einer stehen immer in einer Gruppe, deren Anzahl Einer ungerade ist. Am Ende steht immer eine einzelne Null.

Folgender DEA akzeptiert genau Wörter dieser Sprache L:



Implementieren und testen Sie eine Methode

```
public static boolean isWordLanguage(final String string)
```

welche das Wortproblem löst, indem sie obigen DEA implementiert, welcher `true` zurückliefert, falls `string` ein Wort der Sprache L ist.

8 Formale Sprache definieren (ca. 60')

8.1 Lernziele

- Eine formale Sprache definieren.
- Alternative Definitionsmöglichkeiten anwenden.

8.2 Grundlagen

Diese Aufgabe basiert auf den AD-Inputs "A32_IP_FormaleSprachen" und "A33_IP_ RegulaereSprachenEndlicheAutomaten". Die Aufgabe beinhaltet keine Programmierung.

8.3 Aufgaben

Gegeben sei folgende Sprache L:

A = {0, 1}

L = {0, 011, 01111, 0110, 011011, 011011110, ... }

Alle Wörter von L beginnen mit einer einzelnen Null. Anschliessend können Einer und Nullen abwechselnd folgen. Eine Null steht immer alleine; Einer stehen immer in einer Gruppe, deren Anzahl Einer gerade ist.

a) Definieren Sie L mit Hilfe eines Syntaxdiagrammes.

b) Definieren Sie L alternativ mit Hilfe von EBNF.

c) Auch eine Definition mit einem regulären Ausdruck ist möglich. Probieren Sie's.

d) Um was für einen Typ von Sprache haben wir es demnach bei L zu tun?

e) Entsprechend kann man die Sprache L auch mit einem endlichen Automaten (EA) definieren. Wie Sie gelernt haben, kann man dazu einen DEA oder einen NEA angeben. Zeichnen Sie also einen EA auf. Wahrscheinlich werden Sie feststellen, dass man einfacher einen NEA oder gar einen ε-NEA findet. Können Sie auch einen korrekten DEA aufzeichnen? Denken Sie daran, dass es bei einem EA mehrere Endzustände geben darf.

9 Optional: State Design Pattern (ca. 45')

9.1 Lernziele

- Das "State Design Pattern" prinzipiell verstehen.
- Verstehen, wie man Zustandsübergänge implementieren kann.

9.2 Grundlagen

Diese Aufgabe basiert auf dem AD-Input "A31_IP_Automaten" und auf folgender Quelle:

www.philipphauer.de/study/se/design-pattern/state.php

Die Aufgabe beinhaltet keine Programmierung.

9.3 Aufgaben

Studieren Sie die referenzierte Website. Schauen Sie sich insbesondere auch die Code-Fragmente an.

a) Was für Vorteile bietet das "State Design Pattern"?

b) Wie kann man die Zustandsübergänge implementieren?

c) Können Sie die UML-Diagramme lesen? Halten Sie allfällige Fragen dazu fest.