Hochschule Bremerhaven
University of Applied Sciences

Fakultät II – Management und Informationssysteme
Informatik
Modul Theoretische Informatik
Prof. Dr.-Ing Henrik Lipskoch

Protokoll zu Aufgabenblatt 02: Team: ti2023_22

Von

Ekane Njoh Junior Lesage Matrikelnmr: 40128
Aguiwo II Steve Matrikelnmer: 40088

I. Aufgabe 1

Für diese Aufgabe müssen zwei reguläre Grammatiken konstruiert werden, jeweils eine links und rechte. Dazu haben wir folgendes Beispiel aus dem ersten Übungsblatt verwendet:

```
{"type" : "https://beispiel.com/Junior"  , "title" : "You should not pass
Ekane." ,  "detail" : "Lesage don't give you the permssion to acces this
file." , "instance" : "/account/123/prompt/Njoh"}
```

Für diese Aufgaben mussten wir unsere Grammatikregeln neu anpassen, weil wir sonst mehr als 15 Regeln gehabt hätten.

a. Mengenangaben

Sei $G = (\Sigma, V, P, < problem + json >)$
$\boldsymbol{A} =$: "$\boldsymbol{https://beispiel.com/Junior}$" ,
$\boldsymbol{B} =$: "$\boldsymbol{You\ should\ not\ pass\ Ekane.}$" ,
$\boldsymbol{C} =$: "$\boldsymbol{Lesage\ don't\ give\ you\ the\ permssion\ to\ acces\ this\ file.}$" ,
$\boldsymbol{D} =$: "$\boldsymbol{/account/123/prompt/Njoh}$" ,
$\boldsymbol{T} =$ "$\boldsymbol{type}$"
$\boldsymbol{T2} =$ "$\boldsymbol{title}$"
$\boldsymbol{De} =$ "$\boldsymbol{detail}$"
$\boldsymbol{It} =$ "$\boldsymbol{instance}$"

Seien $\Sigma = \{$ "{"; $\boldsymbol{T}$; $\boldsymbol{A}$; $\boldsymbol{T2}$; $\boldsymbol{B}$; $\boldsymbol{De}$; $\boldsymbol{C}$; $\boldsymbol{It}$; $\boldsymbol{D}$; "}" $\}$
Und $V = \{< problem + json >; < type >; < title >; < detail >; < instance >; < uri >; < acces > ; < denied >; < infos >; < zeichen > \}$

Dann haben wir

b. rechtsreguläre Grammatik

wir wissen aus $[Folie\ 2-1]$ Satz über reguläre Sprachen, dass sich eine rechtsreguläre Grammatik dadurch auszeichnet, dass links höchstens eine Variable gefolgt von genau einem Buchstaben steht.

P =

<problem+json>	→	{<type>
<type>	→	T<uri>
<uri>	→	A<title>
<title>	→	T2<denied>
<denied>	→	B<detail>
<detail>	→	De<acces>
<acces>	→	C<instance>
<instance>	→	It<infos>
<infos>	→	D<zeichen>
<zeichen>	→	}

c. Linksregulär Grammatik

Nach der Konvention der regulären Sprachen besitzt jede rechtsreguläre Grammatik eine linksreguläre Grammatik.

P =

<problem+json>	→	**<zeichen>}**
<zeichen>	→	**<infos>D**
<infos>	→	**<instance>It**
<instance>	→	**<acces>C**
<acces>	→	**<detail>De**
<detail>	→	**<denied>B**
<denied>	→	**<title>T2**
<title>	→	**<uri>A**
<uri>	→	**<type>T**
<type>	→	**{**

II. Aufgabe 2

Bei dieser Aufgabe geht es darum, aus den regulären Sprachen aus der ersten Aufgabe einen (deterministischen) Automaten nach Definition aus der Vorlesung.
Zusätzlich soll auch ein Fehlerzustand für illegale Zeichen und einen Fehlerzustand für legale, aber unpassende Zeichen besitzen. Dabei sollten wir uns passende illegale Zeichen ausdenken.

a. Automat für die rechtsreguläre Grammatik

Aus $[Folie\ 2-5]$ über deterministischen Automaten wissen wir, Ein (deterministischer) endlicher Automat M (DEA) ist gegeben durch ein Tupel

Sei $M = (Z, \Sigma, \delta, z0, E)\ mit$

$Z = \{z0;\ z1;\ z2;\ z3;\ z4;\ z5;\ z6;\ z7;\ z8;\ z9;\ z10;\ z11;\ z12\}$

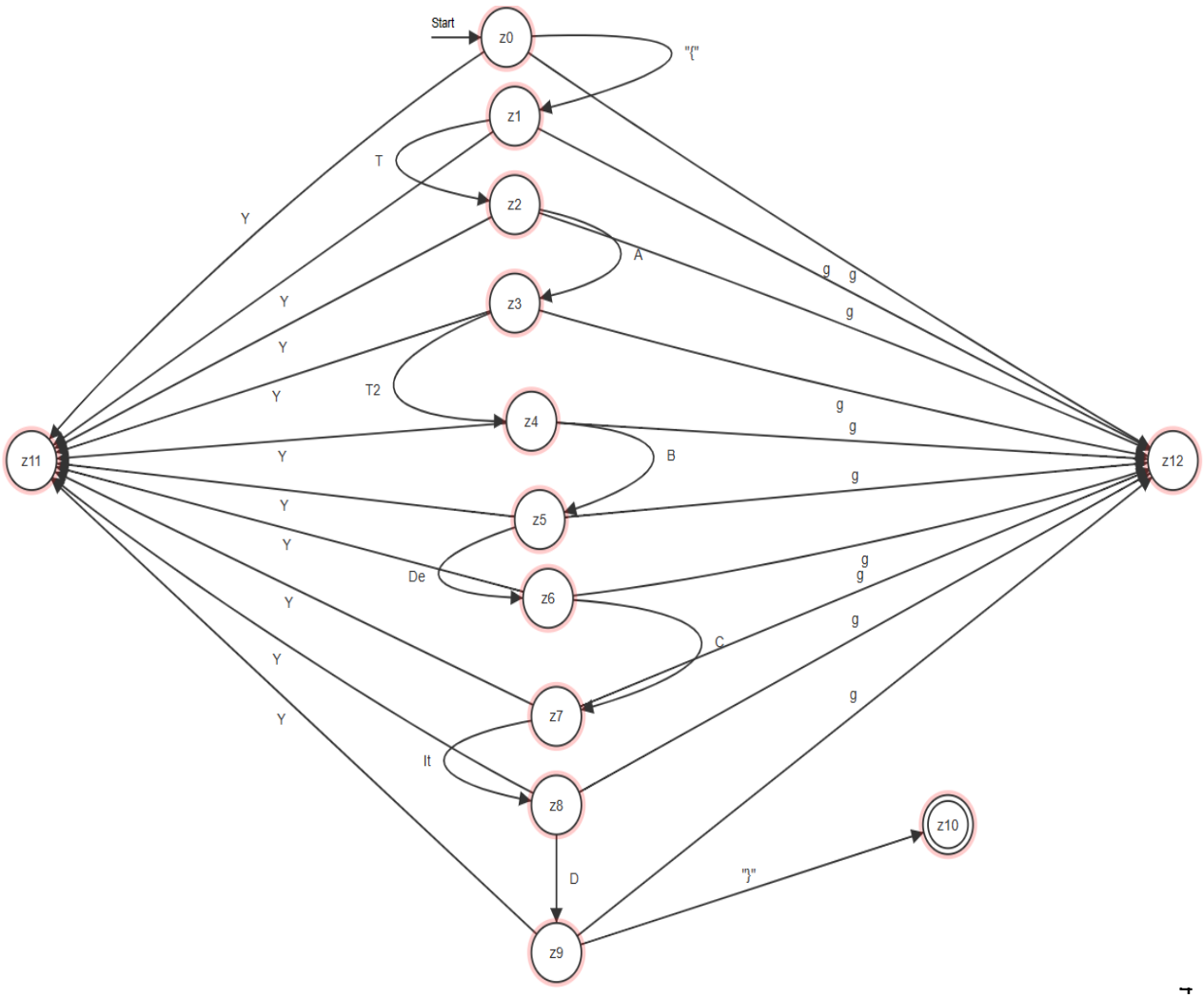
$\Sigma = \{"\{";\ T;\ A;\ T2;\ B;\ De;\ C;\ It;\ D;\ "\}";\ g;\ Y\}$

Seien auch $g \in G = \{\#;\ [\ ;\ ]\ ;\ ?\ ;*\ ;\sim;\ \&\ ;\ \S\ ;\ \%\ ;\ Moin\ ;\ !;\ Bhv\}$,

$Y = beliebig\ erlaubte\ Eingabe, die\ f\ddot{u}r\ den\ aktuellen\ Zustand\ nicht\ passt$

$$
\delta = \begin{cases}
\delta(z0,"\{") = z1; & \delta(z4,B) = z5; & \delta(z8,D) = z9; \\
\delta(z0,G) = z12; & \delta(z4,G) = z12; & \delta(z8,G) = z12; \\
\delta(z0,Y) = z11; & \delta(z4,Y) = z11; & \delta(z8,Y) = z11; \\
\delta(z1,T) = z2; & \delta(z5,De) = z6; & \delta(z9,"\}") = z1; \\
\delta(z1,G) = z12; & \delta(z5,G) = z12; & \delta(z9,G) = z1; \\
\delta(z1,Y) = z11; & \delta(z5,Y) = z11; & \delta(z9,Y) = z10; \\
\delta(z2,A) = z3; & \delta(z6,C) = z7; & \delta(z11,G) = z12; \\
\delta(z2,G) = z12; & \delta(z6,G) = z12; & \\
\delta(z2,Y) = z11; & \delta(z6,Y) = z11; & \\
\delta(z3,T2) = z4; & \delta(z7,It) = z8; & \\
\delta(z3,G) = z12; & \delta(z7,G) = z12; & \\
\delta(z3,Y) = z11; & \delta(z7,Y) = z11; &
\end{cases}
$$

$E = \{z10\}$



Bei dieser Aufgabe ist uns aufgefallen, dass es viel leichter war, den Automaten mit der rechtsregulären Grammatik zu konstruieren. Wie haben uns auch die Frage gestellt woran dies liegen könnte und haben uns nach Überlegung geeinigt, dass es sich durch die Ausdruckskraft der linksregulären Grammatik erklären lässt. Ein Automat, der mit der linksregulären Grammatik erzeugt wurde, ist total verkehrt von dem, was man sich im Regelfall vorstellen könnte. Denn das Ende ist dort der Anfang und der Anfang des Ausdrucks stellt das Ende dar.

Aufgabe 3(sed).

Sed ist ein Stromeditor (stream editor). Ein Stromeditor wird für grundlegende Texttransformationen auf einen Eingabestrom (einer Datei oder aus einer Verarbeitungskette) verwandt. Obwohl in einigen Aspekten ähnlich zu einem Editor, der Bearbeitungen nach Skript erlaubt (wie Ed), führt Sed nur einen Durchlauf über die Eingabe(n) durch und ist somit effizienter. Allerdings ist es die Fähigkeit von Sed, Text in einer Verarbeitungskette zu filtern, die ihn besonders gegenüber anderen Arten von Editoren auszeichnet.

Für unsere Übung haben wir beschlossen, einen einfachen Text zu schreiben, der mit Hilfe von 'sed' bearbeitet wird. Sodass wir einen bearbeites text am Ende bekommen. Unser Ziel bei der Verwendung von sed war, einen gegebenen Text zu ändern, ohne den Text direkt zu modifizieren, sondern dies stattdessen durch die Verwendung von sed in einem Skript zu erreichen.
Das ist unser Ursprungstext.

```
steaguiwoii@hopper:~$ cat aufgabe3.txt
moin und herzlich willkommen bei TI-2023,
ich bin Steve und mein Partner im Team ist Junior.
Wir präsentieren euch einen einfachen Text, der mit `sed` bearbeitet wird.
Am Ende erhalten wir ein bearbeitetes Ergebnis.
```

Mit diesen folgenden Skript wird dann unseren Text bearbeitet.

```
1  !#/bin/bash/sed -f
2
3  # hier wir Steve durch Aguiwo ersetzt.
4  s/Steve/Aguiwo/
5
6  # Falls s/Steve/Aguiwo/ eine erfolgreiche Ersetzung durchgeführt hat, seit die letzte Eingabezeile gelesen wurde und seit dem letzten t- Behfel
7  #dann wird zu Marke output verzweigt. Falls Marke fehlt, wird zum Ende des Skripts verzweigt.( Wenn die Substitution nicht erfolgreich war, wird der t-Befehl übersprungen, und der Code unter :output wird
   nicht ausgeführt).
8
9  t output
10
11 :output
12 # hier wird alle kleine vokale Buchstabe durch die entsprechend größe Buchstabe in der erste zeile ersetzt. Also im meinem ganzen wird dann die vokale Buchstabe a,e,i,o,u durch A,E,I,O,U .
13
14 y/aeiou/AEIOU/
15
16 # hier wird mid '4d' die vierte zeile gelöscht .
17
18 4d
19
20 # ersetze im Text 'wir' durch 'ich und mein patner'. Das bedeutet, dass jedes Mal, wenn das Wort "Wir" gefunden wird, wird es durch "mein Partner und ich" ersetzt.
21
22 s/Wir/ich und mein patner/g
23
24 #mit 2p wird der zweite zeile ausgedruckt und mit q wird die verabeitunt beendet und am Ende wird dann nur die zweite zeile ausgedruckt.
25 #2p
26 #q
27
28
29 # mit Q wird  Das Sed-Skript sofort ohne Verarbeitung weiterer Eingabe beendet
30 #Q
31
32 # mit G wird hier eine leere zeile im ganzen Text hinzugefügt.
33 G
34
35
36 #  gibt es den gerade bearbeiteten Teil des Textes aus( Das hilft, die Veränderungen zu sehen oder den Text zu überprüfen, der durch die Sed-Bearbeitung gegangen ist.)
37 p
38
39 #Gibt die aktuelle Zeilennumer aus.
40
41 =
42 #Schreibt den aktuellen Musterbereich nach Dateiname . In unserem Fall wird das in der datei (aufgabe03.txt) geschrieben.
43
44 w aufgabe03.txt
45
```


Lassen wir kurz diesen Skript in kleinen Schritt erklären .

Für die bearbeitung von unserem Text haben wir verschiedene Befehle von sed benutzt und zwar (s, t , y, d, g, p, q, G, = , w).

.Wir haben hier mit (s/Steve/Aguiwo/) die name Steve durch Aguiwo ersetzt und wir haben folgenden Ausgabe bekommen.

```
steaguiwoii@hopper:~$ cat aufgabe03.txt
moin und herzlich willkommen bei TI-2023,
ich bin Aguiwo und mein Partner im Team ist Junior.
Wir präsentieren euch einen einfachen Text, der mit `sed` bearbeitet wird.
Am Ende erhalten wir ein bearbeitetes Ergebnis.
```

. t ouput , falls die Ersetzung von Steve durch Aguiwo erfolgreich war, wird zur Marke output verzweigt. Da dies der Fall ist, wird der Code unter :output ausgeführt. Wir bekommen also das selbe ausgabe.

.1y/aeiou/AEIOU (Hier werden alle kleinen Vokale(aeiou) in der ersten Zeile durch die entsprechenden großen Vokale(AEIOU) ersetzt.)

```
mOIn Und hErzlIch wIllkOmmEn bEI TI-2023,
ich bin Aguiwo und mein Partner im Team ist Junior.
Wir präsentieren euch einen einfachen Text, der mit `sed` bearbeitet wird.
Am Ende erhalten wir ein bearbeitetes Ergebnis.
```

. 4d (d steht hier für löschen und 4 steht für die nummer der zeile , das heißt unsere vierte zeile wird gelöscht.)

```
steaguiwoii@hopper:~$ cat aufgabe03.txt
mOIn Und hErzlIch wIllkOmmEn bEI TI-2023,
ich bin Aguiwo und mein Partner im Team ist Junior.
Wir präsentieren euch einen einfachen Text, der mit `sed` bearbeitet wird.
```

. s/Wir/ich und mein Partner/g (wird wird dann durch ich und mein Patner ersetzt)

```
steaguiwoii@hopper:~$ cat aufgabe03.txt
mOIn Und hErzlIch wIllkOmmEn bEI TI-2023,
ich bin Aguiwo und mein Partner im Team ist Junior.
ich und mein patner präsentieren euch einen einfachen Text, der mit `sed` bearbeitet wird.
Am Ende erhalten wir ein bearbeitetes Ergebnis.
```

. 2p p (2p druckt die zweite Zeile des Textes aus, und q beendet die Verarbeitung sofort danach. In unserem Fall wird also nur die zweite Zeile angezeigt, und das Skript wird danach gestoppt.)

```
steaguiwoii@hopper:~$ cat aufgabe03.txt
mOIn Und hErzlIch wIllkOmmEn bEI TI-2023,
```

. G (Hier wird mit G eine leere Zeile im gesamten Text hinzugefügt.)

```
mOIn Und hErzlIch wIllkOmmEn bEI TI-2023,

ich bin Aguiwo und mein Partner im Team ist Junior.

ich und mein patner präsentieren euch einen einfachen Text, der mit `sed` bearbeitet wird.

Am Ende erhalten wir ein bearbeitetes Ergebnis.
```

.Q (Q beendet die Verarbeitung sofort, ohne den restlichen Text zu bearbeiten.)

.=(Die aktuelle Zeilennummer wird ausgegeben.)

```
steaguiwoli@hopper:~$ cat aufgabe03.txt
1
mOIn Und hErzlIch wIllkOmmEn bEI TI-2023,

2
ich bin Aguiwo und mein Partner im Team ist Junior.

3
ich und mein patner präsentieren euch einen einfachen Text, der mit `sed` bearbeitet wird.

4
Am Ende erhalten wir ein bearbeitetes Ergebnis.
```

. w (w schreibt die aktuell bearbeitete Zeile in die Datei "aufgabe03.txt". In unserem Skript wird die zweite Zeile des bearbeiteten Textes in diese Datei geschrieben.). Da Der Befehl 2p vor dem q sorgt dafür, dass nur die zweite Zeile des Textes ausgegeben wird, bevor das Skript beendet wird. Daher wird der aktuelle Musterbereich, der die zweite Zeile enthält, in die Datei "aufgabe03.txt" geschrieben, da dies der letzte bearbeitete Zustand ist, bevor das Skript endet.

```
steaguiwoli@hopper:~$ cat aufgabe03.txt
mOIn Und hErzlIch wIllkOmmEn bEI TI-2023,
```

https://www.rfc-editor.org/rfc/rfc7807

regulaere-grammatik-endlicher-automat_uni-muenster_2008 (ziemke-koeln.de)

Theoretische Informatik (hs-bremerhaven.de)

Letzter Zugriff: am 02.11.2023 um 22.26