

Hochschule Bremerhaven  
University of Applied Sciences

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

**Protokoll zu Aufgabenblatt 04: Team: ti2023\_22**

**Von**

**Ekane Njoh Junior Lesage**

Matrikelnmr: 40128

**Aguiwo II Steve**

Matrikelnmer: 40088

## Inhaltsverzeichnis

I. Aufgabe 1 (Pumping-Lemma für reguläre Sprachen) .....	2
a. Definierte Grammatik für RFC 7807.....	3
b. Anwendung am ersten Beispiel.....	3
c. Anwendung am zweiten Beispiel.....	4
d. Anwendung am dritten Beispiel.....	5
e. Anwendung am vierten Beispiel.....	6
II. Aufgabe 2 .....	7
a. Definition der Produktionsregeln.....	7
b. 8	
c. 10	
d. Definition der Produktionsregeln.....	<b>Fehler! Textmarke nicht definiert.</b>
III. Literaturverzeichnis .....	10

### I. Aufgabe 1 (Pumping-Lemma für reguläre Sprachen)

Bei dieser Aufgabe geht es darum für jedes der vier Beispiele aus dem ersten Übungsblatt zu zeigen, das für diese das Pumping-Lemma gilt. Hierfür zeigen wir für jedes der vier Wörter  $z_j, j = 1, 2, 3, 4$  :

- Dass es eine Zahl  $n_j$  gibt,
- eine Zerlegung  $z_j = u_j v_j w_j$  existiert
- sodass alle drei Bedingungen des Pumping-Lemmas gelten

Zur Lösung dieser Aufgabe ist es notwendig uns erstmal an unsere Grammatik erinnern.

### a. Definierte Grammatik für RFC 7807

Es ist  $G = (\Sigma, V, P, problem + json)$  Dabei betrachten wir erstmal die Menge  $V$ , die alle unserer Variablen enthält.

$\Sigma = \{A; B; C; D; E; F; G; H; I; J; K; L; M; N; O; P; Q; R; S; T; U; V; W; X; Y; Z; a; b; c; d; e; f; g; h; i; j; k; l; m; n; o; p; q; r; s; t; u; v; w; x; y; z; 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; ", " ; "-"; "_"; "https://"; "."; "("; ")" ; "["; "]" ; "{"; "}" ; " "; "!" ; "/" ; "\"; ":" ; "\" ; "\" }$

$V = \{problem + json; type; title; detail; instance; string; uri; char; tld\}$

$P = \left\{ \begin{array}{l} \langle problem + json \rangle ::= \{ \langle type \rangle , \langle title \rangle , \langle detail \rangle , \langle instance \rangle \} \\ \langle type \rangle ::= \langle char \rangle \langle uri \rangle \langle string \rangle \\ \langle string \rangle ::= \langle char \rangle | "." | "(" | ")" | "[" | "]" | "{" | "}" | "!" | "/" | "\" | "?" | "@" | ":" | "\"" | \langle char \rangle \langle string \rangle \\ \langle char \rangle ::= A|B|C| \dots |Z|a|b|c| \dots |z|0|1|2|3| \dots |9| - | "_" | \langle char \rangle \\ \langle uri \rangle ::= "https://" \langle char \rangle "." \langle tld \rangle \langle string \rangle \langle char \rangle \\ \langle tld \rangle ::= A|B|C| \dots |Z|a|b|c| \dots |z|0|1|2|3| \dots |9| \langle tld \rangle \\ \langle title \rangle ::= \langle char \rangle \langle string \rangle \\ \langle detail \rangle ::= \langle char \rangle \langle string \rangle \\ \langle instance \rangle ::= \langle char \rangle \langle string \rangle \end{array} \right\}$

Abkürzungen:

$A = "type" :$

$B = "title" :$

$C = "detail" :$

$D = "instance" :$

Sei  $L = \{ w \mid w \text{ ist eine Zeichenfolge, die den definierten Produktionsregeln entspricht} \}$

Anmerkung: bei allen Beispielen wird die Annahme gemacht, dass die Sprache regulär ist.

### b. Anwendung am ersten Beispiel

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

Nehmen wir das Wort `/account/123/prompt/Njoh` für diesen Fall (Es wird hier als ein Wort betrachtet, weil es von einer einzigen Variablen erzeugt wird). Dieses Wort wird in unseren Produktionsregel durch eine einzige Variable erzeugt, daher

Die Bedingungen für das Pumping-Lemma sind folgende:

Hier wird angenommen:  $n_1 = 20$

$\exists n_1, \text{ sodass } \forall z_1 \in L, \text{ mit } |z_1| \geq n_1:$

$z_1$  lässt sich in  $z_1 = u_1 v_1 w_1$  und

- $|u_1| \geq 1$
- $|u_1 v_1| \leq n$
- $\forall i = 0, 1, 2, \dots : u_1 v_1^i w_1 \in L$

$z_1 = \text{"account/123/prompt/Njoh"}$

Zerlegen wir in  $z_1 = u_1 v_1 w_1$ , so erhalten wir folgendes:

$u_1 = \text{"account/123/"}$ , mit  $|u_1| \geq 1$

$v_1 = \text{"prompt"}$ , mit

$w_1 = \text{"Njoh"}$

Nun können wir für verschiedene Werte von  $i$  prüfen, ob das Pumping-Lemma für dieses Beispiel gilt.

Für  $i = 0$  bekommen wir  $z_1 = u_1 v_1^0 w_1 = u_1 w_1 \rightarrow z_1 = \text{"account/123/Njoh"}$

Für  $i = 1$  bekommen wir  $z_1 = u_1 v_1^1 w_1 \rightarrow z_1 = \text{"account/123/prompt/Njoh"}$

Für  $i = 2$  bekommen wir  $z_1 = u_1 v_1^2 w_1 \rightarrow z_1 = \text{"account/123/prompt/prompt/Njoh"}$

Daraus können wir schließen, dass  $\forall i = 1, 2, 3, 4, \dots$  bleibt  $z_1$  in unserer Sprache enthalten, weil es in jeden geprüften Fällen Produktionsregeln entspricht. Daher gilt auch das Pumping-Lemma für dieses Beispiel.

Dass das Pumping-Lemma für unsere Sprache gilt, dies nicht, dass es unbedingt regulär ist, weil unsere Sprache folgende Bedingung für reguläre Sprachen nicht erfüllt:  $w_2 \in \Sigma \cup \Sigma V$ , sodass links genau eine Variable steht und rechts genau ein Buchstabe gefolgt von höchstens einer Variablen. Unsere Sprache lässt sich eher zu den kontextfreien Sprachen klassifizieren.

Es wäre dennoch möglich die Sprache regulär werden zu lassen, indem Änderungen an die Produktionsregeln vorgenommen werden. Neue Regeln könnten der Form sein:

$$A ::= eB \text{ oder } A ::= Bd$$

### c. Anwendung am zweiten Beispiel

```
{ "type" : "https://hp.com/Steve" , "title" : "Aguiwo II." , "detail" : "Ekane Njoh ist nicht eingetragen." , "instance" : "account/Lesage/mgsa/Njoh" }
```

Nehmen wir das Wort **"Ekane Njoh ist nicht eingetragen."** für diesen Fall (wird ebenfalls als ein einziges Wort angesehen, denn es entsteht durch eine Variable und zwar die Variable <string>).

Die Bedingungen für das Pumping-Lemma sind folgende:

Angenommen  $n_2 = 10$

$\exists n_2$ , sodass  $\forall z_1 \in L$ , mit  $|z_2| \geq n$ :

$z_2$  lässt sich in  $z_2 = u_2 v_2 w_2$  und

- $|u_2| \geq 1$
- $|u_2 v_2| \leq n$

- $\forall i = 0, 1, 2, \dots : u_2 v_2^i w_2 \in L$

$z_2 = \text{"Ekane Njoh ist nicht eingetragen."}$

Zerlegen wir in  $z_2 = u_2 v_2 w_2$ , so erhalten wir folgendes:

$u_2 = \text{"Ekane Njoh"}$ , mit  $|u_2| \geq 1$

$v_2 = \text{ist}$ , mit

$w_2 = \text{"nicht eingetragen."}$

Nun können wir für verschiedene Werte von  $i$  prüfen, ob das Pumping-Lemma für dieses Beispiel gilt.

Für  $i = 0$  bekommen wir  $z_2 = u_2 v_2^0 w_2 = u_2 w_2 \rightarrow z_2 = \text{"Ekane Njoh nicht eingetragen."}$

Für  $i = 1$  bekommen wir  $z_2 = u_2 v_2^1 w_2 \rightarrow z_2 = \text{"Ekane Njoh ist nicht eingetragen."}$

Für  $i = 2$  bekommen wir  $z_2 = u_2 v_2^2 w_2 \rightarrow z_2 = \text{"Ekane Njoh ist ist nicht eingetragen."}$

Daraus können wir schließen, dass  $\forall i = 1, 2, \dots$  bleibt  $z_1$  in unserer Sprache enthalten, weil es in jeden geprüften Fällen Produktionsregeln entspricht. Daher gilt auch das Pumping-Lemma für dieses Beispiel.

#### d. Anwendung am dritten Beispiel

```
{ "type" : "https://Steve.123/Aguiwo" , "title" : "Junior hat bald Geburtstag." ,  
  "detail" : "TI macht Spaß." , "instance" : " " /Lesage/1234/localhost/moin" }
```

Betrachten wir das Wort `"https://Steve.123/Aguiwo"` als Beispiel. Das Pumping-Lemma legt folgende Bedingungen fest:

Angenommen  $n_3 = 20$

$\exists n_3$ , sodass  $\forall z_3 \in L$ , mit  $|z_3| \geq n$ :

$z_3$  lässt sich in  $z_3 = u_3 v_3 w_3$  und

- $|u_3| \geq 1$
- $|u_3 v_3| \leq n_3$
- $\forall i = 0, 1, 2, \dots : u_3 v_3^i w_3 \in L$

$z_3 = \text{"https://Steve.123/Aguiwo"}$

Zerlegen wir in  $z_2 = u_2 v_2 w_2$ , so erhalten wir folgendes:

$u_3 = \text{"https://"}$ , mit  $|u_3| \geq 1$

$v_3 = \text{Steve}$ , mit

$w_3 = \text{123/Aguiwo"}$

Nun können wir für verschiedene Werte von  $i$  prüfen, ob das Pumping-Lemma für dieses Beispiel gilt.

Für  $i = 0$  bekommen wir  $z_3 = u_3 v_3^0 w_3 = u_3 w_3 \rightarrow z_3 = \text{"https://.123/Aguiwo"}$

Für  $i = 1$  bekommen wir  $z_3 = u_3 v_3^1 w_3 \rightarrow z_3 = \text{"https://Steve.123/Aguiwo"}$

Für  $i = 2$  bekommen wir  $z_3 = u_3 v_3^2 w_3 \rightarrow z_3 = \text{"https://SteveSteve.123/Aguiwo"}$

Für  $i = 3$  bekommen wir  $z_3 = u_3 v_3^3 w_3 \rightarrow z_3 = \text{"https://SteveSteveSteve.123/Aguiwo"}$

Es wird sofort auffällig, dass das Wort  $z_3$  für  $i = 0$  nicht mehr in der Sprache enthalten ist, weil es die Produktionsregeln widerspricht. Somit bestätigt dieses Beispiel, dass die Sprache nicht regulär ist.

Anmerkung: Es heißt allerdings nicht, dass Pumping-Lemma für das Beispiel gelten könnte. Es besteht die Möglichkeit, die Produktionsregeln so anzupassen, dass die Sprache regulär wäre und somit das Pumping-Lemma gelten würde.

#### e. Anwendung am vierten Beispiel

```
{"type" : "https://lib.iso/Njoh" , "title" : "failed to call Steve." ,  
"detail" : "can not reach Aguiwo." , "instance" : " /log/error/Steve/9875" }
```

Betrachten wir das Wort "instance" als Beispiel im Kontext des Pumping-Lemmas. Das Pumping-Lemma stellt bestimmte Anforderungen an Wörter in einer Sprache, um zu überprüfen, ob diese Sprache regulär ist.

Angenommen  $n_4 = 10$

$\exists n_4$ , sodass  $\forall z_4 \in L$ , mit  $|z_4| \geq n$ :

$z_4$  lässt sich in  $z_4 = u_4 v_4 w_4$  und

- $|u_4| \geq 1$
- $|u_4 v_4| \leq n_4$
- $\forall i = 0, 1, 2, \dots : u_4 v_4^i w_4 \in L$

$z_4 = \text{"instance"}$

Zerlegen wir in  $z_4 = u_4 v_4 w_4$ , so erhalten wir folgendes:

$u_4 = \text{"inst"}$ , mit  $|u_4| \geq 1$

$v_4 = \text{"a"}$ , mit

$w_4 = \text{"nce"}$

Nun können wir für verschiedene Werte von  $i$  prüfen, ob das Pumping-Lemma für dieses Beispiel gilt.

Für  $i = 0$  bekommen wir  $z_4 = u_4 v_4^0 w_4 = u_4 w_4 \rightarrow z_4 = \text{"instnce"}$

Für  $i = 1$  bekommen wir  $z_4 = u_4 v_4^1 w_4 \rightarrow z_4 = \text{"instance"}$

Für  $i = 2$  bekommen wir  $z_4 = u_4 v_4^2 w_4 \rightarrow z_4 = \text{"instaance"}$

Das Wort  $z_4$  widerspricht für  $i = 0$  & und  $\forall i \geq 2$  den Produktionsregeln, weil nicht mehr in der Sprache enthalten ist. Die Produktionsregeln legen fest, dass  $z_4$  in der Sprache fest definiert ist. Es werden daher keine Änderungen an diesem Wort zugelassen.

Anmerkung: Es heißt allerdings nicht das Pumping-Lemma für das Beispiel gelten könnte. Es besteht die Möglichkeit, die Produktionsregeln so anzupassen, dass die Sprache regulär wäre und somit das Pumping-Lemma gelten würde.

## II. Aufgabe 2

Bei dieser Aufgabe handelt es sich um das Nachvollziehen vom konstruktiven Algorithmus zur Lösung des Wortproblems. Hierbei müssen wir weiterhin die von uns gewählte Sprache aus dem ersten Übungsblatt verwenden und folgende Schritte und Hinweise durchgehen und beachten:

- Folien 3-11 bis 3-13 für unsere Sprache nachvollziehen
- beim Startsymbol beginnen und dann
- die Menge H und die Menge W durch sukzessives Regelanwenden füllen, bis wir alle Wörter der Länge n erreicht haben;
- die Zahl n entspricht dabei dem kleinsten n für das, das Pumping-Lemma auf unsere Sprache zutrifft.

Zur Lösung dieser Aufgabe ist es notwendig erstmal unsere Produktionsregeln und dadurch auch unsere gesamte Grammatik zu ermitteln, damit die restlichen Schritte nachvollziehbar bleiben.

Bei dieser Aufgabe brauchen wir nicht wie im vorherigen anzunehmen, dass wir mit einer regulären Sprache zu tun haben, weil die Entscheidbarkeit vom Typ 1 bis 3 definiert ist.

### a. Definition der Produktionsregeln

Es ist  $G = (\Sigma, V, P, problem + json)$  Dabei betrachten wir erstmal die Menge V, die alle unserer Variablen enthält.

$\Sigma = \{A; B; C; D; E; F; G; H; I; J; K; L; M; N; O; P; Q; R; S; T; U; V; W; X; Y; Z; a; b; c; d; e; f; g; h; i; j; k; l; m; n; o; p; q; r; s; t; u; v; w; x; y; z; 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; ", " ; "-" ; "_" ; https:// ; "." ; "(" ; ")" ; "[" ; "]" ; "{" ; "}" ; " " ; "!" ; "/" ; "\" ; ":" ; "" \}$

$V = \{problem + json; type; title; detail; instance; string; uri; char; tld\}$

$$P = \left\{ \begin{array}{l} \langle problem + json \rangle ::= \{ \langle type \rangle , \langle title \rangle , \langle detail \rangle , \langle instance \rangle \} \\ \langle type \rangle ::= a \langle uri \rangle \langle string \rangle \\ \langle string \rangle ::= \langle char \rangle | "." | "(" | ")" | "[" | "]" | "{" | "}" | " " | "!" | "/" | "\" | "?" | "@" | ":" | "|" | "< char >" | "< string >" \\ \langle char \rangle ::= A|B|C|...|Z|a|b|c|...|z|0|1|2|3|...|9| "-" | "_" | "< char >" \\ \langle uri \rangle ::= "https://" \langle char \rangle "." \langle tld \rangle \langle string \rangle \langle char \rangle \\ \langle tld \rangle ::= A|B|C|...|Z|a|b|c|...|z|0|1|2|3|...|9| \langle tld \rangle \\ \langle title \rangle ::= b \langle string \rangle \\ \langle detail \rangle ::= c \langle string \rangle \\ \langle instance \rangle ::= d \langle string \rangle \end{array} \right.$$

Abkürzungen:

$a = \text{"type"}$  :

$b = \text{"title"}$  :

$d = \text{"detail"}$  :

$d = \text{"instance"}$  :

Sei  $L(G) = \{ w \mid w \text{ ist eine Zeichenfolge, die den definierten } P \text{ entspricht} \}$

## b. Anwendung der algorithmischen Konstruktion

Wir wissen:

Zu einem Wort  $w$  mit  $|w| = n < \infty, n \in \mathbb{N}$  gibt es nur endlich viele Wörter  $w \in \Sigma^*$  mit  $|w| \leq n$

$\Rightarrow$  Es gibt nur endlich viele Wörter  $w \in L(G)$  mit  $|w| \leq n$ .

$\Rightarrow$  Wir können jedes Wort  $w \in L(G)$  mit  $|w| \leq n$  und mit dem zu prüfenden Wort  $\tilde{w}$  vergleichen, in endlicher Zeit.

Dies gilt nur für  $|w| < \infty$ .

$H : \{S\}$

$W : \{\}$

$H : S \rightarrow \{\text{problem} + \text{json}\}$

$R1 : \langle \text{problem} + \text{json} \rangle \rightarrow \langle \text{type} \rangle ", \langle \text{title} \rangle ", \langle \text{detail} \rangle ", \langle \text{instance} \rangle "$

$H : \{S, \langle \text{problem} + \text{json} \rangle, \langle \text{type} \rangle ", \langle \text{title} \rangle ", \langle \text{detail} \rangle ", \langle \text{instance} \rangle "\}$

$W : \{\}$

$\langle \text{type} \rangle ::= a \langle \text{uri} \rangle \langle \text{string} \rangle$

$\langle \text{uri} \rangle ::= \text{https://hp.com/}$

$\langle \text{string} \rangle ::= \langle \text{char} \rangle | "." | "/" | "\" | "-" | ":" | "\"" | \langle \text{char} \rangle \langle \text{string} \rangle$

$\langle \text{string} \rangle ::= \langle \text{char} \rangle$

$\langle \text{char} \rangle ::= \text{"steve"}$

$R2 : \langle \text{type} \rangle \rightarrow a \text{"https://hp.com/steve"}$

$H : \{S, \langle \text{problem} + \text{json} \rangle, \langle \text{type} \rangle ", \langle \text{title} \rangle ", \langle \text{detail} \rangle ", \langle \text{instance} \rangle "; a \langle \text{uri} \rangle$

$\langle \text{string} \rangle ; \text{https://hp.com} ; \langle \text{char} \rangle ; \text{"steve"} \}$

$W : \{ a \text{"https://hp.com/steve"} \}$



$\langle \text{title} \rangle ::= b \langle \text{string} \rangle$

$\langle \text{string} \rangle ::= \langle \text{char} \rangle$

$\langle \text{char} \rangle ::= \text{"Aguiwo II."}$

R3:  $\langle \text{title} \rangle \rightarrow b \text{"Aguiwo II."}$

$H: \{S, \langle \text{problem} + \text{json} \rangle, \text{type} \rangle, \langle \text{title} \rangle, \langle \text{detail} \rangle, \langle \text{instance} \rangle; a \langle \text{uri} \rangle \langle \text{string} \rangle;$

$\text{https://hp.com}; \langle \text{char} \rangle; \text{"steve"}; b \langle \text{string} \rangle; \langle \text{char} \rangle; \text{"Aguiwo II."}; b \text{"Aguiwo II."}\}$

$w: \{a \text{"https://hp.com/steve"}, b \text{"Aguiwo II."}\}$

$\langle \text{detail} \rangle ::= c \langle \text{string} \rangle$

$\langle \text{string} \rangle ::= \langle \text{char} \rangle$

$\langle \text{char} \rangle ::= \text{"Ekane Njoh ist nicht eingetragen."}$

R4:  $\langle \text{detail} \rangle \rightarrow c \text{"Ekane Njoh ist nicht eingetragen."}$

$H: \{S, \langle \text{problem} + \text{json} \rangle; \text{type} \rangle, \langle \text{title} \rangle, \langle \text{detail} \rangle, \langle \text{instance} \rangle; a \langle \text{uri} \rangle$

$\langle \text{string} \rangle; \text{https://hp.com}; \langle \text{char} \rangle; \text{steve}; b \langle \text{string} \rangle; \langle \text{char} \rangle;$

$\text{"Aguiwo II."}; b \text{"Aguiwo II."}; \langle \text{char} \rangle; \text{Ekane Njoh ist nicht eingetragen.};$

$c \text{"Ekane Njoh ist nicht eingetragen."}\}$

$W: \{a \text{"https://hp.com/steve"}, b \text{"Aguiwo II."}, c \text{"Ekane Njoh ist nicht eingetragen."}\}$

$\langle \text{instance} \rangle ::= d \langle \text{string} \rangle$

$\langle \text{string} \rangle ::= \langle \text{char} \rangle$

$\langle \text{char} \rangle ::= \text{"/account/lesage/mgsa/Njoh"}$

R5:  $\langle \text{instance} \rangle \rightarrow d \text{"/account/lesage/mgsa/Njoh"}$

$H: \{S, \langle \text{problem} + \text{json} \rangle; \langle \text{type} \rangle, \langle \text{title} \rangle, \langle \text{detail} \rangle, \langle \text{instance} \rangle; a \langle \text{uri} \rangle$

$\langle \text{string} \rangle; \text{https://hp.com}; \langle \text{char} \rangle; \text{steve}; b \langle \text{string} \rangle; \langle \text{char} \rangle;$

$\text{"Aguiwo II."}; b \text{"Aguiwo II."}; \langle \text{char} \rangle; \text{Ekane Njoh ist nicht eingetragen.};$

$c \text{Ekane Njoh ist nicht eingetragen.}; d \langle \text{string} \rangle; \langle \text{char} \rangle;$

$\text{"/account/lesage/mgsa/Njoh"}; d \text{"/account/lesage/mgsa/Njoh"}\}$

$W: \{a \text{"https://hp.com/steve"}, b \text{"Aguiwo II."}, c \text{"Ekane Njoh ist nicht eingetragen."},$

$d \text{ "/account/lesage/mgsa/Njoh " }$

wir können dann also mit unseren Abkürzungen ( $a = \text{"type":}$ ;  $b = \text{"title":}$ ;  $c = \text{"detail":}$ ;

$d = \text{"instance":}$ ;) in unserem Beispiel dann einsetzen und wir erhalten also folgendes

$W: \{ \text{"type": "https://hp.com/steve"}, \quad \text{"title": "Aguiwo II."},$

$\text{"detail": "Ekane Njoh ist nicht eingetragen."}, \text{"instance": "/account/lesage/mgsa/Njoh"} \}$

*wir haben den Begriffe wie "steve", "Aguiwo", "https://hp.com/", "Ekane Njoh ist nicht eingetragen" usw... als eine Art von Buchstaben in unserem Beispiel verwendet haben, um die Länge der Aufgabe zu reduzieren.*

### III. Literaturverzeichnis

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

[Pumping Lemma: Kontextfreie und Reguläre Sprache · \[mit Video\] \(studyflix.de\)](#)

[Produktionsregel – Wikipedia](#)