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 (Pummping-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. 8
d. Definition der Produktionsregeln ..... **Fehler! Textmarke nicht definiert.**
III. Literaturverzeichnis 10

I. <u>Aufgabe 1</u> (Pummping-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 Pummping-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 Pummping-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.

Σ = { 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:// ; "." ; "(" ; ")" ; "[" ; "]" ; "{" ; "}" ; " " ; "!" ; "/" ; "\" ; ":" ; "“" }

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

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

Abkürzungen:

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

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

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

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

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

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. Dieses Wort wird in unseren Produktionsregel durch eine einzige Variable erzeugt, daher

Die Bedingungen für das Pummping-Lemma sind folgende:

Hier wird angenommen: $n_1 = 20$

$\exists n_1, sodass\ \forall z_1 \in L, mit\ |z_1| \geq n$:

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

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

$z_1 =$ "/account/123/prompt/Njoh"

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

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

$v_1 = /prompt$, mit

$w_1 = Njoh$"

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

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

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

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

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

Dass das Pummping-Lemma für unsere Sprache gilt, dies nicht, dass es unbedingt regulär ist, weil unsere Sprache folgende Bedingung für reguläre Sprachen nicht erfüllt: w2 $\in \Sigma \cup \Sigma V$, 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:

$$A ::= eB \; oder \; A ::= Bd$$

c. *<u>Anwendung am zweiten Beispiel</u>*

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

Nehmen wir das Wort **eingetragen** für diesen Fall.

Die Bedingungen für das Pummping-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

- $|u_2| \geq 1$
- $|u_2 v_2| \leq n$
- $\forall i = 0,1,2, \ldots : \ u_2 v_2^i w_2 \in L$

$z_2 =$ eingetragen

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

$u_2 =$ ein , mit $|u_2| \geq 1$

$v_2 =$ ge, mit

$w_2 =$ tragen

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

Für $i = 0$ bekommen wir $z_2 = u_2 v_2^0 w_2 = u_2 w_2 \rightarrow z_2 =$ eintragen

Für $i = 1$ bekommen wir $z_2 = u_2 v_2^1 w_2 \rightarrow z_2 =$ eingetragen

Für $i = 2$ bekommen wir $z_2 = u_2 v_2^2 w_2 \rightarrow z_2 =$ eingegetragen

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

d. <u>Anwendung am dritten Beispiel</u>

⟶{"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 Pummping-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

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

$z_3 =$ "https://Steve. 123/Aguiwo"

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

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

$v_3 =$ Steve, mit

$w_3 = 123/$Aguiwo"

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

Für $i = 0$ bekommen wir $z_3 = u_3 v_3^0 w_3 = u_3 w_3 \rightarrow z_3 =$ "https://.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 wiederspricht. Somit bestätigt dieses Beispiel, dass die Sprache nicht regulär ist.

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

e. *Anwendung am vierten Beispiel*

⟶**{"type" : "https://lib.iso/Njoh" , "title" : "failled 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

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

$z_4 =$ "instance"

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

$u_4 =$ "inst , mit $|u_3| \geq 1$

$v_4 =$ a, mit

$w_4 =$ nce"

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

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

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

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

Das Wort z_4 wiederspricht für $i = 0 \,\&\, und\ \forall i \geq 2$ den Produktionsegeln, 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 Pummping-Lemma für das Beispiel gelten könnte. Es besteht die Möglichkeit, die Produktionsregeln so anzupassen, dass die Sprache regulär wäre und somit das Pummping-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 Pummping-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.

Σ = { 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:// ; "." ; "(" ; ")" ; "[" ; "]" ; "{" ; "}" ; " " ; "!" ; "/" ; "\"; ":" ; "‟" }

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

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

$$P = \begin{cases} < problem + json > ::= < type > "," < title > "," < detail > "," < instance > \\ < type > ::= a < uri > < string > \\ < title > ::= b < string > \\ < detail > ::= c < string > \\ < instance > ::= d < string > \\ < string > ::= < char > \mid "." \mid "\ " \mid "/" \mid "\" \mid " - " \mid ":" \mid "“" \mid < char >< string > \\ < \text{char} > ::= \text{A}|\text{B}|\text{C}|\ldots|\text{Z}|\text{a}|\text{b}|\text{c}|\ldots|\text{z}|0|1|2|3|\ldots|9| < \text{char} > \\ < uri > ::= https://\text{hp.com}/ \\ < tld > ::= < char > \mid < char >< tld > \end{cases}$$

Abkürzungen:

$a =$ "$type$" :

$b =$ "$title$" :

$c =$ "$detail$" :

$d =$ "$instance$" :

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

b. Anwendung der algorithmischen Konstruktion

Wir wissen:

Zu einem Wort w mit |w| = n < ∞, n ∈ N gibt es nur endlich viele Wörter w ∈ Σ $*$ mit $|m| <= n$

⇒ Es gibt nur endlich viele Wörter w ∈ L(G) mit |w| ≤ n.

⇒ Wir können jedes Wort w ∈ L(G) mit |w| ≤ n und mit dem zu prüfenden Wort w˜ vergleichen, in endlicher Zeit.

Angenommen $n = 10$

Dies gilt nur für |w| < ∞ .

H : {S}

W: {}

H : S →{problem+json}

R1 : <problem+json> →$< type >$ "," $< title >$ "," $< detail >$ "," $< instance >$

H : {S , <problem+json> , $< type >$ "," $< title >$ "," $< detail >$ "," $< instance >$ }

W {}

$< type >::= a < uri >< string >$

$< uri >::= https://hp.com/$

$< string >::= < char >$ |"."|" "|"/"| "\" |" – "| ":"|""| $< char >< string >$

< string >::= <char>

<char>:: = " $steve$ "

R2 : <type> → a ''https://hp.com/steve''

H : {S , <problem+json> , $< type >$ "," $< title >$ "," $< detail >$ "," $< instance >$; a<uri><string> ; https://hp.com ; <char> ; " $steve$ "}

W: { a''https//hp.com/steve''}

$< title >:: = b < string >$

<string> :: = <char>

<char> :: = ''Aguiwo II.''

R3: <title> → b ''Aguiwo II.''

H: {S , <problem+json> , $type >$ "," $< title >$ "," $< detail >$ "," $< instance >$; a<uri><string> ; https://hp.com ; <char> ; " $steve$ " ; $b < string >$; <char> ; ''Aguiwo II.'' ; b''Aguiwo II.'' }

w : {a''https//hp.com/steve'' , b ''Aguiwo II.''}

$< detail >:: = c < string >$

<string> :: = <char>

<char> :: = "Ekane Njoh ist nicht eingetragen."

R4: <detail> → c "Ekane Njoh ist nicht eingetragen."

H: {S , <problem+json> ; $type >$ "," $< title >$ "," $< detail >$ "," $< instance >$; a<uri><string> ; https://hp.com ; <char> ; " $steve$ " ; $b < string >$; <char> ; ''Aguiwo II.'' ; b ''Aguiwo II.'' ; <char>; "Ekane Njoh ist nicht eingetragen."; c "Ekane Njoh ist nicht eingetragen." }

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

<instance>::= d <string>

<string> :: = <char>

<char> ::= "/account/lesage/mgsa/Njoh "

R5: <instance> → d "/account/lesage/mgsa/Njoh "

H: {S , <problem+json>; $type >$ "," $< title >$ "," $< detail >$ "," $< instance >$; a<uri><string> ; https://hp.com ; <char> ; " $steve$ " ; $b < string >$; <char> ; ''Aguiwo II.'' ; b ''Aguiwo II.'' ; <char>; "Ekane Njoh ist nicht eingetragen."; c "Ekane Njoh ist nicht eingetragen." ; d <string>; <char> ; "/account/lesage/mgsa/Njoh " ; d "/account/lesage/mgsa/Njoh "}

W: {a''https//hp.com/steve'' , b ''Aguiwo II.'' , c"Ekane Njoh ist nicht eingetragen.", d "/account/lesage/mgsa/Njoh " }

wir können dann also mit unseren Abkürzungen (a="type": ; b = "title ": ; c="detail": ; d="instance":) in unserem beispiel dann einsetzen . und wir erhalten also folgendes

W: {"type": ''https//hp.com/steve'' , "title" : ''Aguiwo II.'' , "detail" :"Ekane Njoh ist nicht eingetragen.", "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