Hochschule Bremerhaven

University of Applied Sciences

Fakultät II – Management und Informationssysteme

Informatik

Modul Theoretische Informatik

Prof. Dr.-Ing Henrik Lipskoch

**Protokoll zu Aufgabenblatt 03: Team: ti2023\_22**

**Von**

**Ekane Njoh Junior Lesage**  Matrikelnmr: 40128

**Aguiwo II Steve** Matrikelnmer: 40088

Inhaltsverzeichnis

Inhalt

[I. Aufgabe 1 (Pummping-Lemma für reguläre Sprachen) 2](#_Toc150476498)

[a. Definierte Grammatik für RFC 7807 2](#_Toc150476499)

[b. Anwendung am ersten Beispiel 3](#_Toc150476500)

[c. Anwendung am zweiten Beispiel 4](#_Toc150476501)

[d. Anwendung am dritten Beispiel 5](#_Toc150476502)

[e. Anwendung am vierten Beispiel 5](#_Toc150476503)

[II. Literaturverzeichnis 6](#_Toc150476504)

# Aufgabe 1 (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 :

* Dass es eine Zahl gibt,
* eine Zerlegung existiert
* sodass alle drei Bedingungen des Pummping-Lemmas gelten

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

## Definierte Grammatik für RFC 7807

Es ist G = (Σ, 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 }**

**<problem+json> ::= "{" <type> "," <title> "," <detail> "," <instance> "}"**

**<type> ::= "type" ":" <uri> <string>**

**<string>::= <char>|"."|"("|")"|"["|"]"|"{"|"}"|" "|"!"|"/"| "\"| "?" | "@" | ":"|"“"|<char><string>**

**<char>::=A|B|C|…|Z|a|b|c|…|z|0|1|2|3|...|9|"-"|"\_"|<char>**

**<uri> ::= "https://" <char> "." <tld> <string> <char>**

**<tld>::= A|B|C|…|Z|a|b|c|…|z|0|1|2|3|...|9|<tld>**

**<title>::= "title" ":" <string>**

**<detail>::= "detail" ":" <string>**

**<instance>::= "instance" ":" <string>**

Sei

## 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.

Die Bedingungen für das Pummping-Lemma sind folgende:

und

* |
* |

Zerlegen wir in , so erhalten wir folgendes:

, mit |

, mit

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

Für bekommen wir

Für bekommen wir

Für bekommen wir

Daraus können wir schließen, dass bleibt 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: , 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.

## 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 **eingetragen** für diesen Fall.

Die Bedingungen für das Pummping-Lemma sind folgende:

Angenommen

und

* |
* |

Zerlegen wir in , so erhalten wir folgendes:

, mit |

, mit

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

Für bekommen wir

Für bekommen wir

Für bekommen wir

Daraus können wir schließen, dass bleibt in unserer Sprache enthalten, weil es in jeden geprüften Fällen Produktionsregeln entspricht. Daher gilt auch das Pummping-Lemma für dieses Beispiel.

## 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 Pummping-Lemma legt folgende Bedingungen fest:

Angenommen

und

* |
* |

Zerlegen wir in , so erhalten wir folgendes:

, mit |

, mit

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

Für bekommen wir

Es wird sofort auffällig, dass das Wort 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.

## 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

und

* |
* |

Zerlegen wir in , so erhalten wir folgendes:

, mit |

, mit

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

Für bekommen wir

Für bekommen wir

Für bekommen wir

Das Wort wiederspricht für den Produktionsegeln, weil nicht mehr in der Sprache enthalten ist. Die Produktionsregeln legen fest, dass in der Sprache fest definiert ist. Es werden daher keine Änderungen an diesem Wort zugelassen.

## Aufgabe 2

## Literaturverzeichnis

[**https://www.rfc-editor.org/rfc/rfc7807**](https://www.rfc-editor.org/rfc/rfc7807)

[Pumping Lemma: Kontextfreie und Reguläre Sprache · [mit Video] (studyflix.de)](https://studyflix.de/informatik/pumping-lemma-1445)

[Produktionsregel – Wikipedia](https://de.wikipedia.org/wiki/Produktionsregel)