Hochschule Bremerhaven University of Applied Sciences

Fakultät II – Management und Informationssysteme Informatik

Modul Theoretische Informatik

Prof. Dr.-Ing Henrik Lipskoch

Protokoll zu Aufgabenblatt 05: Team: ti2023_22

Von

Ekane Njoh Junior Lesage Matrikelnmr: 40128

Aguiwo II Steve Matrikelnmer: 40088

Inhalt

l.	,	Aufgabe 1	2
II.		Aufgabe 2	5
		H	
		H	
(Э.	H	5
(d.	H	5
6	€.	H	5
ı	II.	Literaturverzeichnis	5

I. Aufgabe 1

Es handelt es sich bei dieser Aufgabe um die Formulierung unseres RFCs als echt-kontextfreie Sprache. Dazu sollte wir in unseren RFC nach einer echt-kontextfreie Struktur zu suchen und dabei folgende Punkte beachten:

- a) Das Extrahieren der Regeln (begrenzen Sie auf ca. 10-15) für die echt-kontextfreie Struktur
- b) und Formulierung dieser als eine echt-kontextfreie und zusammenhängende Grammatik
- c) und zwar mit Regeln in Chomsky-Normalform,
- d) mit den vereinbarten Symbolen und der Schreibweise aus der Vorlesung auf.

Anmerkungen:

- Da unser RFC keine echt-kontextfreie Struktur hat, haben wir uns eine dazu passende echtkontextfreie Struktur überlegt.
- Wir werden uns bei dieser Aufgabe auf die Produktion eines Pflichtfeldes in unserem RFC begrenzen. Und zwar das Feld "type".

Aus [Folie] 5 – 8 Chomsky-Normalform wissen wir bereits, dass eine kontextfreie Grammatik $G = (\Sigma, V, P, S)$ mit $\in \not\in L(G)$ ist in Chomsky-Normalform (CNF) genau dann, wenn alle Regeln aus P: entweder der Form $X \to YZ$ oder der Form $X \to a$,

```
mit X, Y, Z \in V \text{ und } a \in \Sigma \text{ sind.}
```

Deswegen lässt sich Folgendes ableiten:

Unsere Regelmenge P:

```
< problem + json > ::= "{" < type > "}" < type > ::= A < 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 > < tld > ::= A|B|C| ... |Z|a|b|c| ... |z|0|1|2|3| ... |9| < tld >
```

Wird zu

```
    < problem + json >::= < type > < problem + json - erweitert >
    < problem + json - erweitert >::= < erstes - zeichen >< type >
    < erstes - zeichen >::= "{"
    < type >::= < type - erweitert > < type - erweitert2 >
    < type >::= A
    < type - erweitert >::= < protokoll >< uri >
    < protokoll >::= "https://"
    < uri >::= < uri0 > < uri3 >
    < uri0 >::= < uri1 >< uri2 >
    < char >::= A|B|C| ... |Z|a|b|c| ... |z|0|1|2|3|... |9|"."|"{"|"}"""""
    < uri2 >::= < tld - char >< uri2 >
    < tld - char >::= a|b|c| ... |z
    < uri3 >::= "}"
```

Es gibt allerdings Regeln, die auf sich selbst verweisen. Daher entspricht das noch nicht wirklich die CNF. Um dieses Problem zu lösen haben wir anhand [Folie] 5 – 12 Algorithmus zur Umformung versucht die Zyklen aufzulösen. Daraus ergibt sich dann Folgendes:

Auflösen der Zyklen und Nummerierung der Variablen

Es gibt als Zyklen <uri1> und <uri2>

```
< uri1 > ::= < char > < uri1 >  wird zu < uri1 > ::= < char > < uri4 >  < uri4 > ::= < char > < uri4 > | <math>\varepsilon Und < uri2 > ::= < tld - char > < uri2 >  wird zu < uri2 > ::= < tld - char > < uri5 >  < uri5 > ::= <math>\varepsilon | < tld - char > < uri5 >
```

<u>Anmerkung:</u> der Algorithmus erlaubt nämlich die Benutzung der leeren Menge mit: Auf dieser und weiteren Folien bedeuten

- Ai jeweils eine Variable
- α, β Wortteile aus Variablen und Buchstaben, regulärer Ausdruck $\alpha \equiv (A|t)*$, d.h. es ist möglich, dass der Wortteile nur aus Buchstaben, nur aus Variablen, aus einem Gemisch von Variablen und Buchstaben besteht oder auch leer ist.

Rückwärtseinsetzen

Es gibt keine Regeln der Form $A \rightarrow B$, daher entfällt dieser Schritt.

Hinzufügen von Buchstabenvariablen

Dieser Schrittentfällt ebenfalls, denn In Regeln der Form $F \rightarrow f$ wird f nicht ersetzt.

> Aufbrechen von langen Regeln

Keine Regel muss in diesem Fall aufgebrochen werden, da keine Regel der Form $A \rightarrow A_1 A_2 A_3 \dots A_k$ existiert.

Schließlich erhalten wir:

```
1. < problem + json > := < type > < problem + json - erweitert >
2. < problem + json - erweitert > := < erstes - zeichen > < type >
3. < erstes - zeichen > := "{}"
4. < type > := < type - erweitert > < type - erweitert2 >
5. < type > := \alpha
6. < type - erweitert > := < protokoll > < uri >
7. < protokoll > := "https://"
8. \langle uri \rangle := \langle uri0 \rangle \langle uri3 \rangle
9. < uri0 > := < uri1 > < uri2 >
10. < uri1 > := < char > < uri4 >
11. < uri4 > : : = < char > < uri4 > | \varepsilon|
12. < \text{char} > ::= A|B|C| ... |Z|a|b|c| ... |z|0|1|2|3|... |9|". "|"{"|"}"|" "|": "|"""
13. < uri2 > := < tld - char > < uri5 >
14. < uri5 > := \varepsilon \mid < tld - char > < uri5 >
15. < tld - char > : : = a|b|c|...|z
16. < uri3 >::= "}"
```

Letztendlich ist unsere Grammatik mit neuen Variablen und Buchstaben befüllt, die auch in einer Menge angegeben werden müssen.

```
V = \{< problem + json >; < type >; < problem + json - erweitert >; < erstes - zeichen >; < type - erweitert >; < type - erweitert2 >; < protokoll >; < uri >; < uri 0 >; < uri 1 >; < uri 4 >; < char >; < uri 2 >; < tld - char >; < uri 3 >\}
```

II. Aufgabe 2

\sim	ᆫ
a.	

e. H

III. Literaturverzeichnis

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

Application error: a client-side exception has occurred (codecentric.de)

JSON - GeeksforGeeks