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

[I. Aufgabe 1 2](#_Toc151083640)

[II. Aufgabe 2 4](#_Toc151083641)

[a. H 4](#_Toc151083642)

[b. H 4](#_Toc151083643)

[c. H 4](#_Toc151083644)

[d. H 4](#_Toc151083645)

[e. H 4](#_Toc151083646)

[III. Literaturverzeichnis 5](#_Toc151083647)

# 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:

1. Das Extrahieren der Regeln (begrenzen Sie auf ca. 10-15) für die echt-kontextfreie Struktur
2. und Formulierung dieser als eine echt-kontextfreie und zusammenhängende Grammatik
3. und zwar mit Regeln in Chomsky-Normalform,
4. 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 echt-kontextfreie Struktur überlegt.
* Wir werden uns bei dieser Aufgabe auf die Produktion eines Pflichtfeldes in unserem RFC begrenzen. Und zwar das Feld „type“.

Aus Chomsky-Normalform wissen wir bereits, dass eine kontextfreie Grammatik mit ist in Chomsky-Normalform (CNF) genau dann, wenn alle Regeln aus P:  
entweder der Form oder der Form ,

mit und sind.

Deswegen lässt sich Folgendes ableiten:

Es ist Dabei betrachten wir erstmal die Menge V, die alle unserer Variablen enthält.

**Das Verfahren hierfür ist so gegliedert, dass wir vom BNF zu CNF gelangen**

Unsere Regelmenge

Wird zu

mit und

Es wurden die Variablen in die Form aus der Vorlesung überführt.

Wir haben wir hier leider noch keine echt-kontextfreie Grammatik in der CNF, da unsere Regeln noch nicht der Form oder der Form entsprechen, daher ist eine Erweiterung durch den nachfolgenden Schritt nötig.

Schritt 1: Lösen wir alle Zyklen

Dieser Schritt entfällt bei uns da wir keine Zyklen haben.

Schritt 2: Umnummerierung der Variablen, wobei i < j. Das heißt der Index der Variable auf der linken Seite darf nicht größer der Variable auf der rechten Seite sein.

Schritt 3: Rückwärtseinsetzen.

Es sind bei uns keine Regeln der Form A->B daher entfällt ebenfalls diesen Schritt bei uns.

Schritt 4: Fügen wir alle Buchstaben t eine neue Variable T mit T->t hinzu.

Schritt 5: Regeln aufbrechen

Wir müssen jetzt alle Regeln in der Form A -> BC oder B-> a (wobei a ein Buchstabe ist)

Es sind inzwischen neue Variablen aufgetaucht, sodass sich unsere Variable Menge geändert hat.

Allerdings bleibt die Ausgabe unverändert.

Nun entsprechen unsere neuen Produktionsregeln die CNF, denn jede Regel hat die Form oder die Form , wobei A, B und C Nichtterminale und a Terminalsymbole sind. Alle Nichtterminale in den Regeln haben korrekte Ableitungen und es gibt keine Regeln mit leeren Ableitungen.

# Aufgabe 2

Diese Aufgabe befasst sich ebenfalls mit echt-kontextfreien Strukturen, wobei wir uns eine Programmiersprache aussuchen, die das Programm kennt, d.h. eine aus der Liste, die das Programm zeigt, wenn man in der Konsole aufruft.

Zur Lösung der Aufgabe sind wir dem folgenden Weg gegangen:

## Ausgangssprache

Die von uns gewählte Programmiersprache ist Bash und seine Regeln sehen wie folgt aus:

<command> ::= <builtin-command> | <executable> | <pipeline> | <subshell> | <control-structure> | <assignment>

<builtin-command> ::= <echo> | <cd> | <ls> | ...

<executable> ::= <file>

<pipeline> ::= <command> '|' <command>

<subshell> ::= '(' <command> ')'

<control-structure> ::= <if-statement> | <while-loop> | <for-loop>

<if-statement> ::= 'if' <condition> 'then' <command-list> 'fi'

<while-loop> ::= 'while' <condition> 'do' <command-list> 'done'

<for-loop> ::= 'for' <variable> 'in' <list> 'do' <command-list> 'done'

<condition> ::= <expression> <operator> <expression>

<assignment> ::= <variable> '=' <value>

<command-list> ::= <command> ';' <command-list> | <command>

<value> ::= <string> | <number> | <variable>

<file> ::= <path>

<string> ::= "some text"

<number> ::= 123

<variable> ::= $<identifier>

<identifier> ::= [a-zA-Z\_][a-zA-Z0-9\_]\*

<path> ::= /path/to/file

## Konvertierung der Sprache in echt-kontext

BC1=pipeline; BC2= subshell , BC3= if-statement ; BC4= while-loop ; BC5= for-loop ; BC6= assignment ;

C= condition ; O=opearator ; E=value ; CL= command-list ; S= string ; N=number ; V=variable ; ID= identifier ;

## <S> ::= <BC1> | <BC2> | <BC3> | <BC4> | <BC5> | <BC6>

## <BC1> ::= <command> | <BC1> '|' <BC1>

## <BC2> ::= '(' <BC1> ')'

## <BC3> ::= 'if' <C> 'then' <CL> 'fi'

## <BC4> ::= 'while' <C> 'do' <CL> 'done'

## <BC5> ::= 'for' <V> 'in' <L> 'do' <CL> 'done'

## <BC6> ::= <V> '=' <V>

## <C> ::= <E> <O> <E>

## <E> ::= <S> | <N> | <V>

## <O> ::= '=' | '<' | '>' | ...

## <CL> ::= <C> ';' <CL> | <C>

## <S> ::= "some text"

## <N> ::= 123

## <V> ::= $<ID>

## <ID> ::= [a-zA-Z\_][a-zA-Z0-9\_]\*

## <P> ::= /path/to/file

## Literaturverzeichnis

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

[Application error: a client-side exception has occurred (codecentric.de)](https://www.codecentric.de/wissens-hub/blog/rest-standardisierte-fehlermeldungen-mittels-rfc-7807-problem-details)

[JSON - GeeksforGeeks](https://www.geeksforgeeks.org/json/?ref=gcse)