SWP Assignment 1

Alexander Frewein (01430019)
Klaus Fabian Frühwirt (01131523)
Stephany Amizic (01331786)

Institute of Software Technology
alexander.frewein@student.tugraz.at
fabian.fruehwirth@student.tugraz.at
stephany.amizic@student.tugraz.at

Beispiel 1

a.)

$$
\begin{aligned}
&L = \{\underline{a}(\underline{aa}|\underline{b})^*\underline{c}\}\\
&S \to \underline{a}A\\
&S \to \underline{a}B\\
&S \to \underline{a}C\\
&A \to \underline{a}E\\
&A \to \underline{c}D\\
&B \to \underline{b}B\\
&B \to \underline{c}D\\
&C \to \underline{c}D\\
&D \to \epsilon\\
&E \to \underline{a}A
\end{aligned}
$$

b.)

$$
\begin{aligned}
&L = \{\underline{a}^{(2n)}\underline{b}\ \underline{c}^*\ (\underline{bb}|\underline{d})\ |n > 0\}\\
&S \to \underline{a}A\\
&A \to \underline{a}A\\
&A \to \underline{b}C\\
&A \to \underline{b}B\\
&A \to \underline{b}D\\
&B \to \underline{b}E\\
&B \to \epsilon\\
&C \to \underline{c}C\\
&C \to \underline{c}B\\
&C \to \underline{c}D\\
&D \to \underline{d}D\\
&D \to \epsilon\\
&E \to \underline{b}B
\end{aligned}
$$

Beispiel 2

a.)

Dies ist eine **allgemeine** Grammatik da $\mid \alpha \mid \leq \mid \beta \mid$ **nicht** gilt und somit keine Restriktion $\alpha \rightarrow \beta$ gilt

b.)

Dies ist eine **reguläre** Grammatik da $\mid \alpha \mid \leq \mid \beta \mid, \alpha \in \mathrm{V_N}$ β hat form aA oder a, mit $\mathrm{a} \in \mathrm{V_T} \bigcup \{\epsilon\}$, $\mathrm{A} \in \mathrm{V_N}$

c.)

Dies ist eine **kontextfreie** Grammatik da $\mid \alpha \mid \leq \mid \beta \mid, \alpha \in \mathrm{V_N}$

d.)

Dies ist keine **gültige** Grammatik da $R \rightarrow Q\underline{y}$ nicht laut Definition $\alpha\,\beta \in (V_\mathrm{N} \cup V_\mathrm{T})$ diese Form nicht in der Grammatik definiert ist.

e.)

Dies ist eine **allgemeine** Grammatik da $\mid \alpha \mid \leq \mid \beta \mid$ **nicht** gilt und somit keine Restriktion $\alpha \rightarrow \beta$ gilt

f.)

Dies ist keine **gültige** Grammatik da $\underline{num} \rightarrow \underline{var}$ nicht in den Grammatik definiert ist

Beispiel 3

First und Follow Mengen:

	FIRST	FOLLOW
S	a b c d e	$
A	a b c	d e
B	d e	c d e
C	b	c d e
D	c	d e
E	d e	c d e

LL(1) Tabelle

	a	b	c	d	e	$
S	S → AB	S → AB	S → AB	S → AB	S → AB	S → AB
A	A → aA	A → CD	A → CD			
B				B → E	B → E	
C		C → b B	C → ϵ	C → ϵ	C → ϵ	
D			D → cC	D → ϵ	D → ϵ	
E				E → d	E → e	

Beispiel 4

Gegeben ist die folgende LL(1) Tabelle, welche eine grobe Abstraktion der Variablendeklaration in Scala beschreibt. Die unterstrichenen Zeichenketten in den Spalten der ersten Zeile stellen jeweils ein Terminalsymbol dar.

	var	val	one	two	String	Int	0	1	"	=	$
S	AB	AB									
A	CN:	CN:									
B					String="*V*"	Int=U					
C	var	val									
N			one	two							
U							0V	1V			
V							0V	1V	ε		ε

Überprüfen Sie mittels der gegebenen LL(1) Tabelle ob folgende Ausdrücke gültige Sätze der definierten Grammatik sind:a)var one : String = 1
b)val two : int = 10
Die Lösung für die Unterpunkte a und b soll im folgenden Format erarbeitet und abgegeben werden:

Stack	Input	Produktion/Kommentar
$S	val one: Int= "11"$	...

a.)

Stack	Input	Produktion/Kommentar
$S	var one : String = 1 $	S:= AB
$BA	var one : String = 1$	A:= CN:
$B:NC	var one : String = 1 $	C:= var
$B:N~~var~~	~~var~~ one : String = 1$	
$B:N	one : String= 1$	N:= one
$B:~~one~~	~~one:~~ String= 1$	
$B	String = 1$	B:= String="V"
$"V"~~=String~~	~~String=~~1$	
$"V"	1$	Keine Regel

Diese Satz ist nicht Valid!

b.)

Stack	Input	Produktion/Kommentar
$S	val two : int = 10$	S:= AB
$BA	val two : int= 10$	A:= CN:
$B:NC	val two : int= 10$	C:= ~~val~~
$B:N~~val~~	~~val~~ two : int = 10$	
$B:N	two : int = 10$	N:=two
$B:~~two~~	~~two:~~ int = 10$	
$B	int= 10$	B:= int=U
$U=~~int~~	~~int=~~ 10$	
$U	10$	U:=1V
$V~~1~~	~~1~~0$	
$V	0$	V:= 0V
$V~~0~~	~~0~~$	
$V	$	V:= ϵ
$	$	Valid

Diese Satz ist Valid!

Beispiel 5

Die gegebene Gramatik ist keine LL(1) Gramatik weil:

- X ist Mehrdeutig (Linksfaktorisierung)
- Y ist indirekt Links recrusiv (über Z)

Äquivalente LL(1) Grammatik:

$$
\begin{aligned}
S &\to XY \\
V &\to \underline{x} \\
V &\to \underline{-} \\
W &\to \underline{0}W \\
W &\to \underline{1}W \\
W &\to \underline{\epsilon} \\
X &\to \underline{x}X_2 \\
X_2 &\to V \\
X_2 &\to W \\
Y &\to \underline{+}Y_2 \\
Y_2 &\to \underline{y}W\underline{-}Y_2 \\
Y_2 &\to \underline{\epsilon}
\end{aligned}
$$

LL(1) Tabelle

	$\underline{\text{x}}$	$\underline{\text{-}}$	$\underline{0}$	$\underline{1}$	$\underline{+}$	$\underline{\text{y}}$	\$
S	$S \to XY$						
V	$V \to X$	$V \to \underline{-}$					
W			$W \to \underline{0}W$	$\text{W} \to \underline{1}W$	$W \to \epsilon$	$\text{W} \to \epsilon$	$W \to \epsilon$
X	$X \to \underline{x}X_2$						
X_2	$X_2 \to V$	$X_2 \to V$	$X_2 \to W$	$X2 \to W$	$X_2 \to W$		
Y					$Y \to \underline{+}Y_2$		
Y_2						$Y_2 \to \underline{y}W\underline{-}Y_2$	$Y_2 \to \epsilon$

Beispiel 6

a.)

```
object SimpleParserA extends JavaTokenParsers {
  def A: Parser[Any] = "aa" ~ A | "c" ~ ""
  def B: Parser[Any] = "b" ~ B | "c" ~ ""
  def AB: Parser[Any] = A | B

  def S: Parser[Any] = "a" ~ AB | "ac"
  def apply(s: String) = parseAll(S, s)
}
```

b.)

```
object SimpleParserB extends JavaTokenParsers {
  def N: Parser[Double] = "[1-9][0-9]*|0|-[1-9][0-9]*".r
                          ^^ {i => i.toDouble}
  def S: Parser[Double] = N~"*10^"~N ^^ {case a~_~b => a*math.pow(10,b)}
  def apply(s: String) = parseAll(S, s)
}
```