Homework 3

Antonio Zea Jr

October 15, 2022

1 Consider the following languages:

$L_1 = \{w|w \in \{a, +, -, *, /, (,)\}^*, w$ **is a legal arithmetic expression in infix form**$\}$

$L_2 = \{w|w \in \{0, 1\}^*, w$ **contains at least three 1s**$\}$

$L_3 = \{w|w \in \{0, 1\}^*,$ **the length of** w **is odd**$\}$

$L_4 = \{w|w \in \{0, 1\}^*, w$ **starts and end with the same symbol**$\}$

$L_5 = \{w|w \in \{0, 1\}^*, w$ **is a palindrome**$\}$

$L_6 = \{w|w \in \{0, 1\}^*, w$ **contains equal numbers of 0s and 1s**$\}$

2 Which languages are regular and which are not? Give the regular expressions for the regular languages.

L_1 **is nonregular**

$L_2 = 0^*10^*10^*1(0 \cup 1)^*$

$L_3 = ((0 \cup 1)(0 \cup 1))^*(0 \cup 1)$

$L_4 = (0(0 \cup 1)^*0) \cup (1(0 \cup 1)^*1)$

L_5 **is nonregular**

L_6 **is nonregular**

3 For each nonregular language prove that it is not regular by using the pumping lemma or the closure of the regular languages.

$L_1 = \{w | w \in \{a, +, -, *, /, (,)\}^*, w$ is a legal arithmetic expression in infix form$\}$

Assume L_1 is regular, then there is a number p (pumping length) where if s is any string in L_1 of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. $\forall i \geq 0$, $xy^iz \in L_1$
2. $|y| > 0$
3. $|xy| \leq p$

Let $s = (^{p+1}\{a+\}^{p+1}a)^{p+1}$

case 1: y contains the open parenthesis

In this case xy^iz will contain more open parentheses than close parentheses. Then $xy^iz \notin L_1$.

case 2: y contains the close parenthesis

In this case $|xy| > p$ which fails the pumping lemma.

case 3: y does not contain any parenthesis

Then $|x| > p$ which fails the pumping lemma.

$\therefore L_1$must be nonregular

$L_5 = \{w | w \in \{0, 1\}^*, w$ is a palindrome$\}$

Assume L_5 is regular, then there is a number p (pumping length) where if s is any string in L_5 of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. $\forall i \geq 0$, $xy^iz \in L_5$
2. $|y| > 0$
3. $|xy| \leq p$

Let $s = w(0 \cup 1 \cup \varepsilon)w^{\mathcal{R}}$, where $|w| = p + 1$, somehow $s = xyz$

case 1: y occurs in w

In this case, the left side of palindrome would get pumped and fail to land back in the language L_5.

case 2: y occurs in $w^{\mathcal{R}}$

In this case $|xy| > p$ which fails the pumping lemma.

case 3: y includes a middle character

In this case $|xy| > p$ which fails the pumping lemma.

$\therefore L_5$ must be nonregular

$L_6 = \{w | w \in \{0,1\}^*, w$ **contains equal numbers of 0s and 1s**$\}$

Assume L_6 is regular, then there is a number p (pumping length) where if s is any string in L_6 of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. $\forall i \geq 0$, $xy^iz \in L_6$
2. $|y| > 0$
3. $|xy| \leq p$

Let $s = 0^{p+1}1^{p+1}$, somehow $s = xyz$

case 1: y occurs in 0^{p+1}

In this case, the left side of s would get pumped and fail to land back in the language L_6 since there would be more 0s than 1s.

case 2: y contains characters ouside of 0^{p+1}

In this case $|xy| > p$ which fails the pumping lemma.

$\therefore L_6$ must be nonregular

4 For each language <u>give the CFG</u> that describes it.

$L_1 = \{w | w \in \{a, +, -, *, /, (,)\}^*, w$ **is a legal arithmetic expression in infix form**$\}$

CFG

$S \rightarrow a|SXS|(SXS)$
$X \rightarrow +|-|*|/$

$L_2 = \{w | w \in \{0, 1\}^*, w$ **contains at least three 1s**$\}$

CFG

$S \rightarrow X1X1X1X$
$X \rightarrow 0X|1X|\varepsilon$

$L_3 = \{w | w \in \{0, 1\}^*, w$ **the length of w is odd**$\}$

CFG

$S \rightarrow 0X|1X$
$X \rightarrow 0S|1S|\varepsilon$

$L_4 = \{w | w \in \{0, 1\}^*, w$ **starts and end with the same symbol**$\}$

CFG

$S \rightarrow 0X0|1X1|\varepsilon$
$X \rightarrow 0X|1X|\varepsilon$

$L_5 = \{w | w \in \{0, 1\}^*, w$ **is a palindrome**$\}$

CFG

$S \rightarrow 0S0|1S1|0|1|\varepsilon$

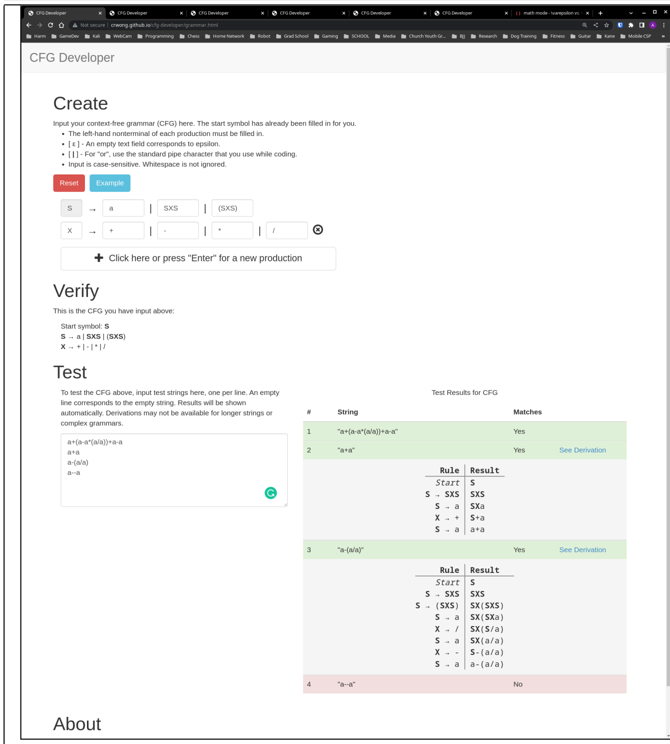
$L_6 = \{w | w \in \{0, 1\}^*, w$ **contains equal numbers of 0s and 1s**$\}$

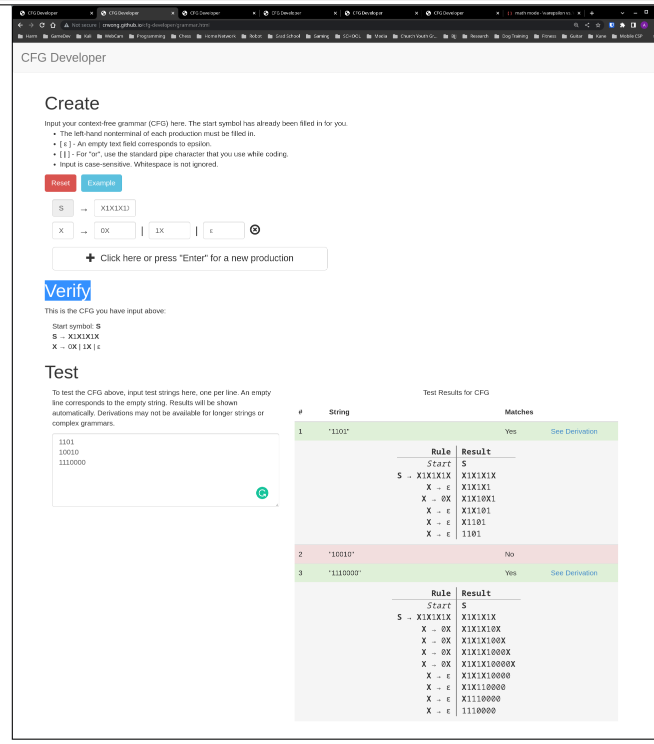
CFG

$S \rightarrow SS|0S1|1S0|\varepsilon$

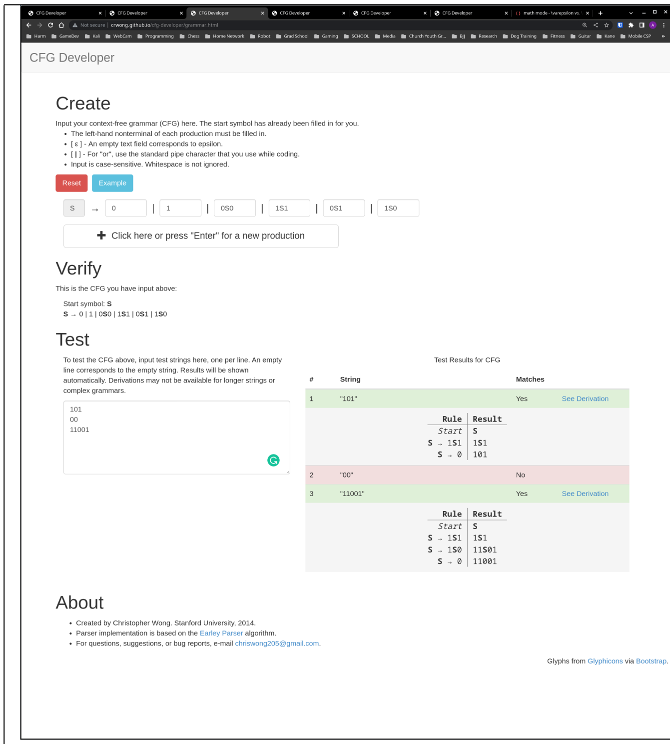
Use the CFG Developer to test the grammars for all languages. Try strings that belong to the language of the grammar and strings that do not. Show the derivations of the strings that belong to the language.

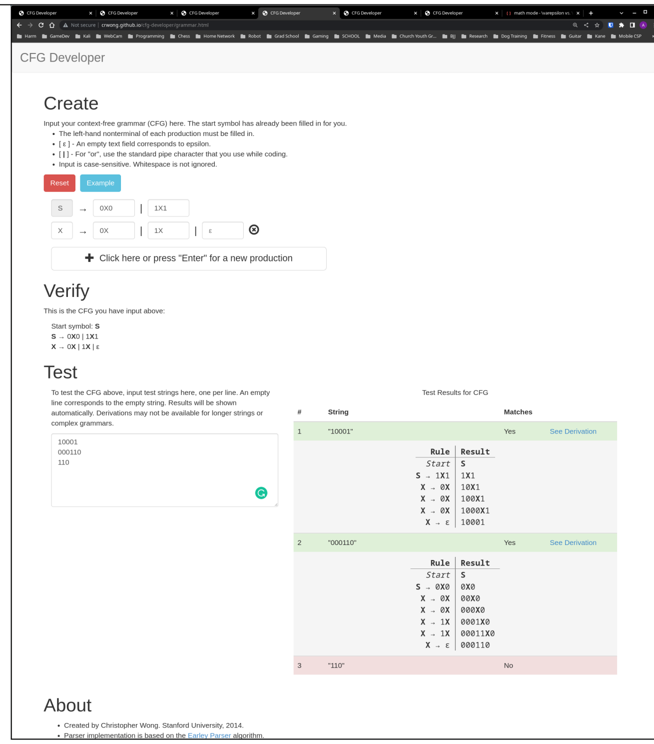
$L_1 = \{w \mid w \in \{a, +, -, *, /, (,)\},$ w is an arithmetic expression in infix form$\}$	$L_2 = \{w \mid w \in \{0,1\}^*, w$ contains at least three 1s$\}$



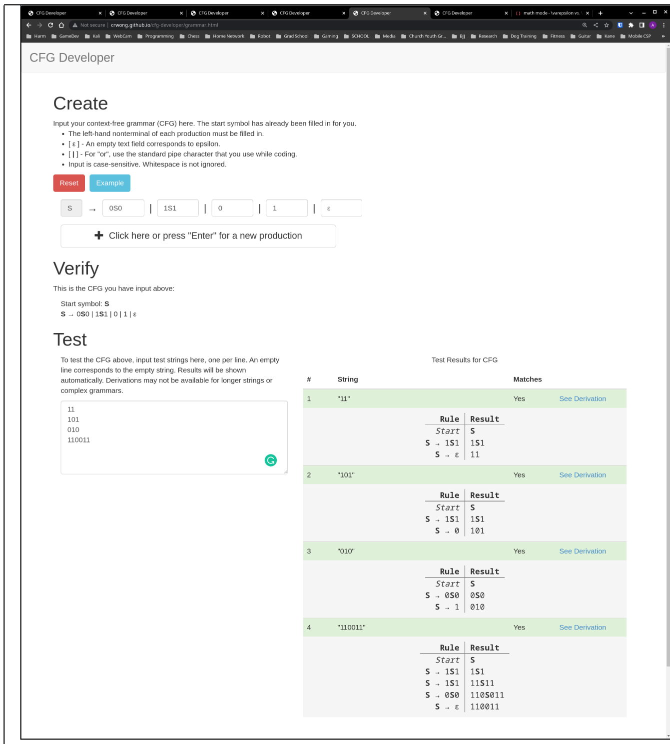


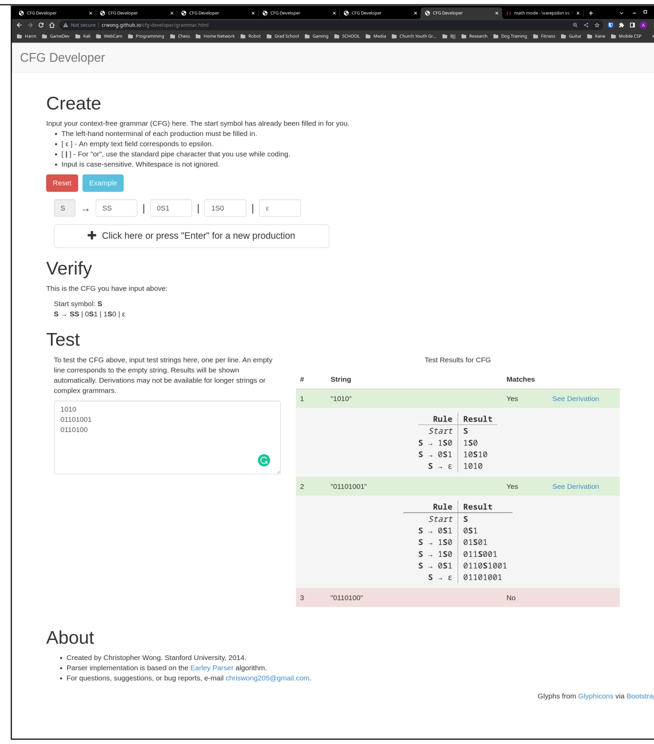
$L_3 = \{w \mid w \in \{0,1\}^*, w \text{ the length of w is odd}\}$	$L_4 = \{w \mid w \in \{0,1\}^*, w \text{ starts and end with the same symbol}\}$





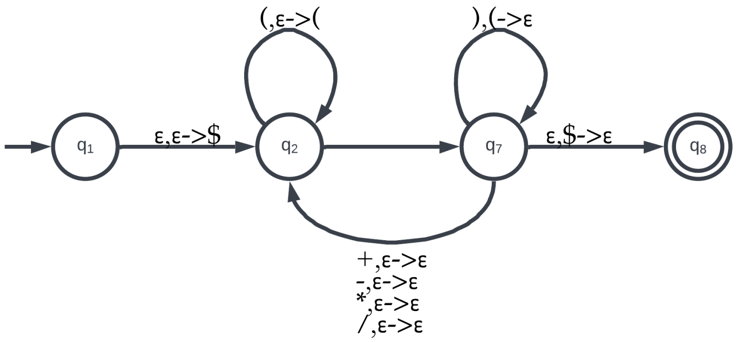
$L_5 = \{w \mid w \in \{0,1\}^*, w \text{is a palindrome}\}$	$L_6 = \{w \mid w \in \{0,1\}^*, w \text{contains equal numbers of 0s and 1s}\}$



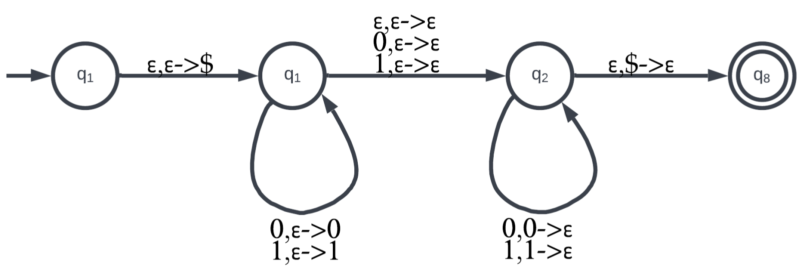


5 **Give <u>informal descriptions</u> (see the solution of Exercise 2.7) and <u>state diagrams of PDAs</u> for the <u>non-regular languages</u> above.**

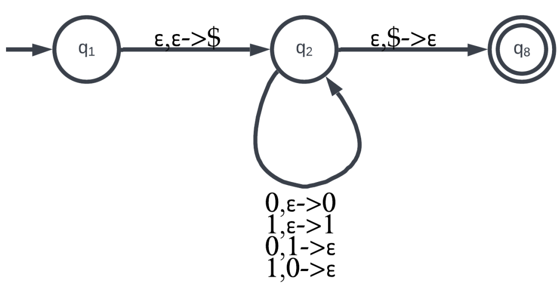
L_1 - PDA



L_5 - PDA



L_6 - PDA



Informal Description

PDA

L_1 The PDA uses its stack to count the number of open parentheses minus the number of close parentheses. It enters an accepting state whenver this count is zero. The PDA scans across the input. If it sees an open parenthesis it pushes it onto the stack. If it sees a +, - , *, or / then nothing is done with the stack. If it sees a close parenthesis then it will try to pop a open parenthesis of the stack. If it cannot then the PDA will reject. After completing the scan of the input and the stack has the starting symbol on top then the PDA accepts.

L_5 The PDA uses its stack to keep track of the order of 0s and 1s. In an accpet instance, as we process the left side of the palindrom. The 0s and 1s get pushed onto the stack. If there is a middle character(ie. the palindrom is of odd length) then $\varepsilon, \varepsilon \rightarrow \varepsilon$ will skip this middle character and then start poping of the stack the right side of the palindrome. Only if the right side, is the reverse of the left, will all characters be poped of the stack and reach our stoping character$.

L_6 The PDA uses the stack to keep track of how many 0s or 1s in the input. It pushes 0s and 1s onto the stack and will only accept if it is able to pop an equal amount of 0s and 1s. A 0 will only be poped when a 1 is read and vice versa. Due to the nondeterministic nature of PDA there will exist one computation where it successfuly reaches the accept state for string where there are equal number of 0s and 1s.