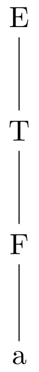
Homework #3

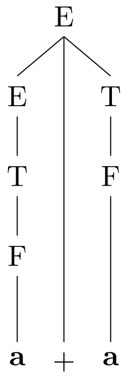
Kyle White

March 11, 2017

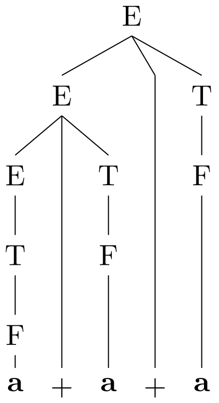
2.1a)



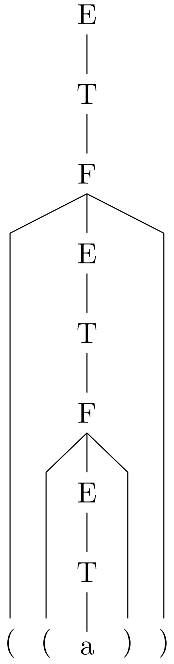
2.1b)



2.1c)



2.1d)



2.4b)
$S \rightarrow 0P0|1P1|$
$P \rightarrow 0P|1P|\epsilon$

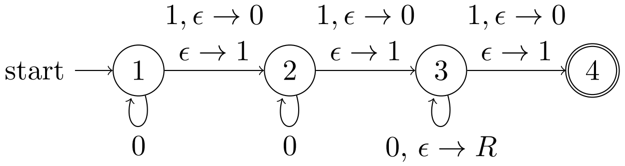
2.4c)
$S \rightarrow 0|1|0S0|0S1|1S0|1S1$

2.4e)

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

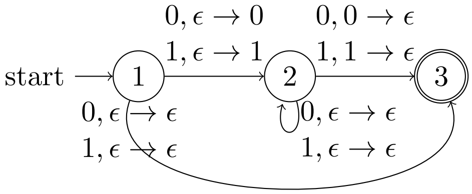
2.4f)
$S \rightarrow S$

2.5a)
Informal Description: This grammar is simple in the sense that it does not require a stack. As long as a string has at least three 1s, the push down automata will accept.



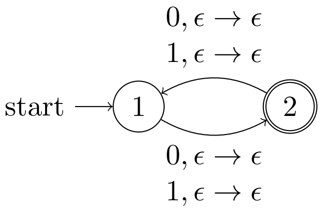
$1, \epsilon \rightarrow 0$
$1, \epsilon \rightarrow 0$
$1, \epsilon \rightarrow 0$
$\epsilon \rightarrow 1$
$\epsilon \rightarrow 1$
$\epsilon \rightarrow 1$
start
1
2
3
4
0
0
$0, \epsilon \rightarrow R$

2.5b)
Informal Description: This grammar recognizes any language that starts and ends with the same symbol. The start symbol is pushed on to the stack and nothing is popped until the same symbol has been seen again.



$0, \epsilon \rightarrow 0$
$1, \epsilon \rightarrow 1$
$0, 0 \rightarrow \epsilon$
$1, 1 \rightarrow \epsilon$
start
1
2
3
$0, \epsilon \rightarrow \epsilon$
$1, \epsilon \rightarrow \epsilon$
$0, \epsilon \rightarrow \epsilon$
$1, \epsilon \rightarrow \epsilon$

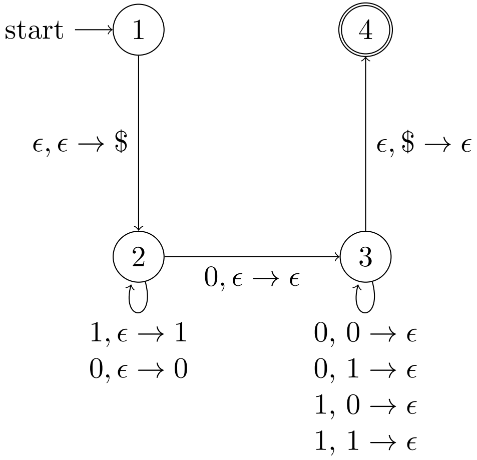
2.5c)
Informal Description: This grammar does not require a stack because odd length inputs do not require memory. In other words, it is not necessary to remember what input has already been read because as long as the length of the input is odd, it does not matter what has been read before.



$0, \epsilon \rightarrow \epsilon$
$1, \epsilon \rightarrow \epsilon$
start
1
2
$0, \epsilon \rightarrow \epsilon$
$1, \epsilon \rightarrow \epsilon$

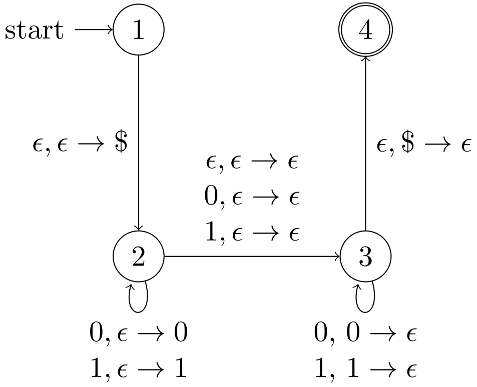
2.5d)
Informal Description: For this grammar, a symbol is pushed on at the beginning of the input. When the middle of the input is hit, a 0 will be pushed on to the stack. The stack is then popped for the same amount of

pushes on the stack before the 0. If there is no input left and 0 is the middle of the input, the symbol pushed at the start of the input is now popped and the PDA accepts. Otherwise, the PDA rejects.



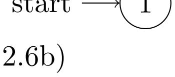
2.5e)

Informal Description: For this grammar, the input represented is a palindrome. The beginning of the palindrome is pushed on to the stack. Once the middle is reached, the remaining input must mirror the exact first half of the palindrome and pop off the stack in that order. If the stack is empty at the end of the input, the PDA accepts. Otherwise, it rejects.


start
1
4
$\epsilon, \epsilon \to \$$
$\epsilon, \epsilon \to \epsilon$
$0, \epsilon \to \epsilon$
$1, \epsilon \to \epsilon$
$\epsilon, \$ \to \epsilon$
2
3
$0, \epsilon \to 0$
$1, \epsilon \to 1$
$0,\ 0 \to \epsilon$
$1,\ 1 \to \epsilon$

2.5f)

Informal Description: This grammar recognizes the empty set so it does not accept any strings.



2.6b)

$S \to S_1|S_2$

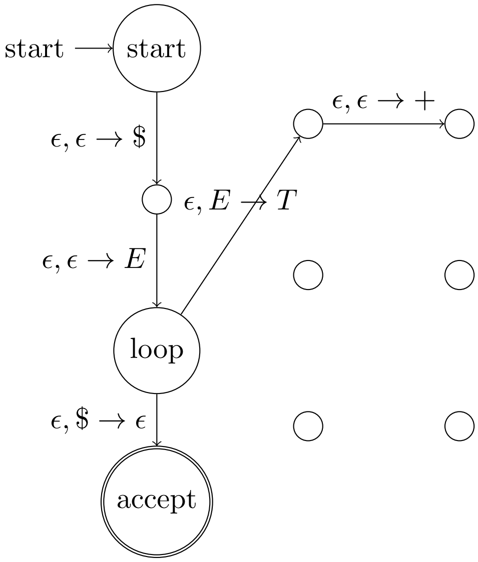
$S_1 \to aS_1b|A|B$
$S_2 \to CbaC$
$A \to aA|a$
$B \to Bb|b$
$C \to CC|a|b|\epsilon$

2.6d)
$S \to aSa|bSb|\#|\#A\#$
$A \to aA|bA|\#A|\epsilon$
$B \to S|A\#S\#A|A\#S|S\#A$

2.9)
$S \to S_1|S_2$
$S_1 \to S_1c|A|\epsilon$
$S_2 \to aS_2|B|\epsilon$
$A \to aAb|\epsilon$
$B \to bBc|\epsilon$

The CFG is ambiguous because it can be derived using S_1 or S_2.
2.10)
To design this PDA, you can split the language into two individual languages and combine them at the end. The final PDA will be described as follows: Start by either reading in a's and pushing them on the stack or by skipping them all together. If you read in a's, then read in b's and for each b, pop an a off of the stack. If the stack is empty when the the b's have all been processed, skip the remaining c's and accept the language. If you skipped reading in a's, read in b's and push them on to the stack. Then, read in c's and for each c read, pop a b off of the stack. If there are no more b's on the stack when the c's have finished being read, the language will be accepted.

2.11)


start
start
$\epsilon, \epsilon \to \$$
$\epsilon, \epsilon \to +$
$\epsilon, E \to T$
$\epsilon, \epsilon \to E$
loop
$\epsilon, \$ \to \epsilon$
accept

2.12)

2.13)
L(G) has a couple of different rules for languages that are recognized. The first is that you can have two #s in the middle of an infinite amount of 0s on either side of them. The other is that you can have 1 # and the number of 0s on the left of the # must be 1/2 the number of 0s on the right side of the #.

2.14)
Add Start Variable
$S_0 \to A$
$A \to BAB|B|\epsilon$
$B \to 00|\epsilon$

Remove all ϵ
$S_0 \to A|\epsilon$
$A \to BAB|BA|AB|A|B|BB$
$B \to 00$

Remove unit rules
$S_0 \to BAB|BA|AB|00|BB|\epsilon$
$A \to BAB|BA|AB|00|BB$

$B \rightarrow 00$

Replace terminals with new variables

$S_0 \rightarrow BAB|BA|AB|UU|BB|\epsilon$
$A \rightarrow BAB|BA|AB|UU|BB$
$B \rightarrow UU$
$U \rightarrow 0$

Shorten RHS

$S_0 \rightarrow BA_1|BA|AB|UU|BB|\epsilon$
$A \rightarrow BA_2|BA|AB|UU|BB$
$B \rightarrow UU$
$U \rightarrow 0$
$A_1 \rightarrow AB$
$A_2 \rightarrow AB$

2.26)

2.30)