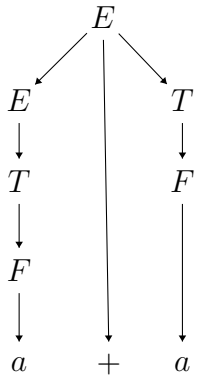
Jason Downing
Email: jason_downing@student.uml.edu
Foundations of Computer Science
Homework #3 - Chapter 2
10/31/2016

2.1 Recall the CFC G4 that we gave in Example 2.4.

a. Parse tree for a:

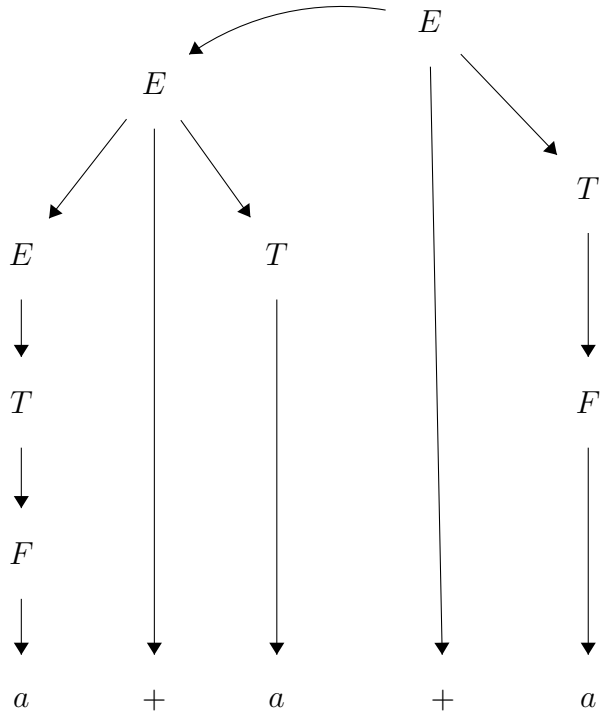
Derivation for a: $E \to T \to F \to a$

b. Parse tree for $a + a$:



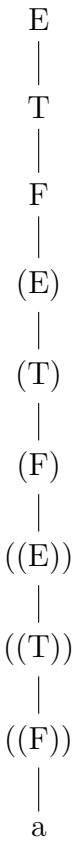
Derivation for $a + a$: $E \to E + T \to T + T \to T + F \to F + a \to a + a$

c. Parse tree for $a + a + a$:



Derivation for $a + a + a$: $E \to E + T \to T + T + T \to F + T + F \to F + F + T \to F + F + F \to a + F + F \to a + a + F \to a + a + a$

d. Parse tree for $((a))$:



Derivation for $((a))$: $E \to T \to F \to (E) \to (T) \to (F) \to ((E))$
$\to ((T)) \to ((F)) \to ((a))$

2.4
Give context-free grammars that generate the following languages.
In all parts, the alphabet $\sum$ is $\{0, 1\}$.

b. {w— w starts and ends with the same symbol}
$S \to 0P0|1P1|\epsilon$
$P \to 0P|1P|\epsilon$

c. {w— the length of w is odd}
$S \to 0|1|00S|01S|10S|11S$

e. {w— w = w^{R}, that is, w is a palindrome}
$S \to 0|1|0S0|1S1|\epsilon$

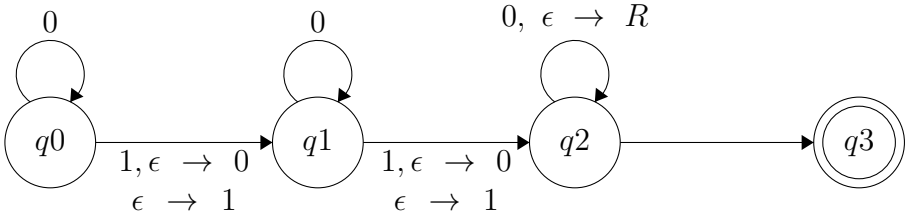
f. The empty set
$S \to \epsilon$

2.5
Give informal descriptions and state diagrams of pushdown automata for the languages in Exercise 2.4.

a. {w— w contains at least three 1s}

Informal: In this grammar, there is no need of a stack (as seen in the diagram). In the grammar only the input is read, and the input is only accepted when there is at least three 1's. The only time the machine will accept a string in this case is when at least three 1's are present; otherwise it is rejected.

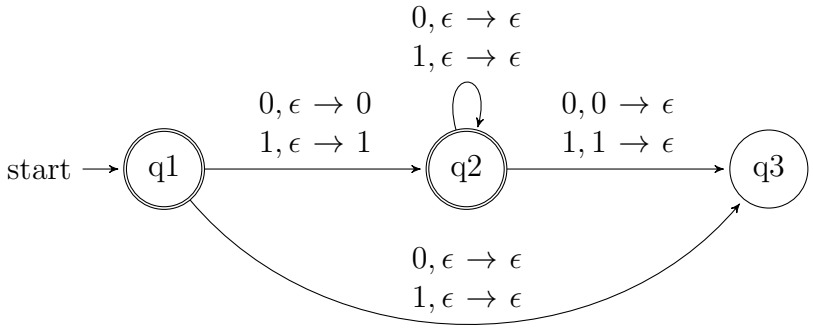
Pushdown Automata:



b. {w— w starts and ends with the same symbol}

Informal: In this grammar, there is an assumption that there is only one symbol in the string that can be accepted without the use of a stack. If there is more than one symbol in the string, then all of the symbols are put into the stack in order. We also check to see whether the last symbol is the one being read. If it is, and this symbol matches with the symbol on the stack, and if all the input has been checked with no input left, then we accept that state. Otherwise we reject the input.

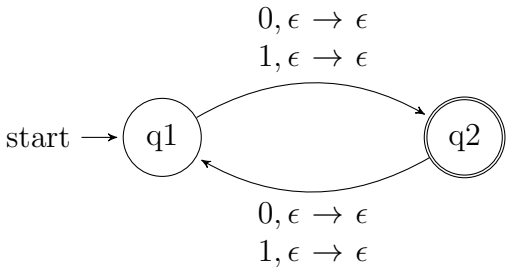
Pushdown Automata:



c. {w— the length of w is odd}

Informal: In this grammar, there is no need of a stack. In this grammar, we only read the input and this input is also only accepted when there is an odd length.

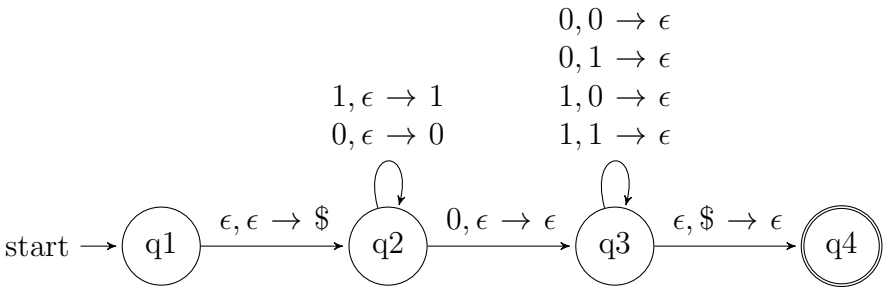
Pushdown Automata:



d. {w— the length of w is odd and its middle symbol is a 0}

Informal: In this grammar, the input is read by the pushdown automata and each symbol is placed onto the stack. At each point, the pushdown automata guesses if the input in the middle or if the next symbol read is in the middle of the string. If the next symbol read is the middle of the string then it does not get read onto the stack. After this, we pop symbols off the stack if they match the input symbols that were read. If the symbols that are popped match the symbols that were pushed earlier, and the stack is empty as the input is finished, we accept the state. Otherwise we reject the state.

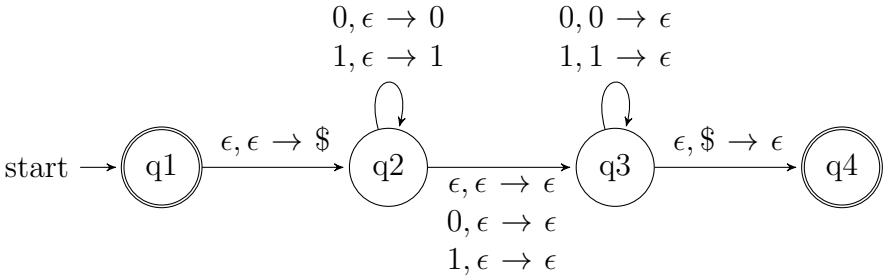
Pushdown Automata:



e. {w— w = w^{R}, that is, w is a palindrome}

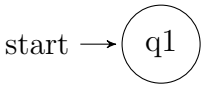
Informal: In this grammar, the input is read by the pushdown automata and each symbol is placed onto the stack. At each point, the pushdown automata guesses if the input in the middle or if the next symbol read is in the middle of the string. If the next symbol read is the middle of the string then it does not get read onto the stack. After this, we pop symbols off the stack if they match the input symbols that were read. If the symbols that are popped match the symbols that were pushed earlier, and the stack is empty as the input is finished, we accept the state. Otherwise we reject the state.

Pushdown Automata:



f. The empty set
Informal: In this grammar, the language only accepts an empty set so it implies that it does not terminate any derivation. And if none of the derivations are terminated, that means that no string is accepted by the CFG, including the empty string.

Pushdown Automata:



2.6
b. The complement of the language $\{a^n b^n \mid n >= 0\}$ The language is:
$S \rightarrow S_1 \quad | \quad S_2$
$S_1 \rightarrow aS_1b \quad | \quad T \quad | \quad U$
$S_2 \rightarrow RbaR$
$T \rightarrow aT \quad | \quad a$
$U \rightarrow Ub \quad | \quad a$
$R \rightarrow RR \quad | \quad a \quad | \quad b \quad | \quad \epsilon$

d. $\{x_1\#x_2\# \ldots \#x_k \mid k \geq 1, each\ x_i \epsilon \{a,b\}^*, and\ for\ some\ i\ and\ j,\ x_i = x_j^R\}$
The language is:
$R \rightarrow S \quad | \quad J\#S\#J \quad | \quad J\#S \quad | \quad S\#J$
$S \rightarrow aSa \quad | \quad bSb \quad | \quad \# \quad | \quad \#J\#$
$J \rightarrow aJ \quad | \quad bJ \quad | \quad \#J \quad | \quad \epsilon$

2.9 $A = \{a^i b^j c^k \mid i = j \text{ or } j = k, \text{ where } i, j, k \geq 0\}$
The context free grammar that generates the language is:

First, split the languages into two parts:
$A_1 = \{a^i b^j c^k \mid i, j, k \geq 0,\ i = j\}$
$A_2 = \{a^i b^j c^k \mid i, j, k \geq 0,\ j = k\}$

Now we can define the grammar for the language A:
$S \to S_1 \mid S_2$

In A_1, the values of i and j are equal. So, there must be an equal number of A's and B's in this language.

CGF for language A_1 is:
$S_1 \to S_1 c \mid E \mid \epsilon$
$E \to aEb \mid \epsilon$

In A_2, j and k are equal, so there are an equal number of B'c and C's.

CGF for language A_2 is:
$S_2 \to aS_2 \mid F \mid \epsilon$
$F \to bFc \mid \epsilon$

Since we can use either A_1 or A_2 to generate the string, either S_1 or S_2 can be used. Thus, the context free grammar for the language A is ambiguous.

2.10 Give an informal description of a pushdown automaton that recognizes the language A in Exercise 2.9.

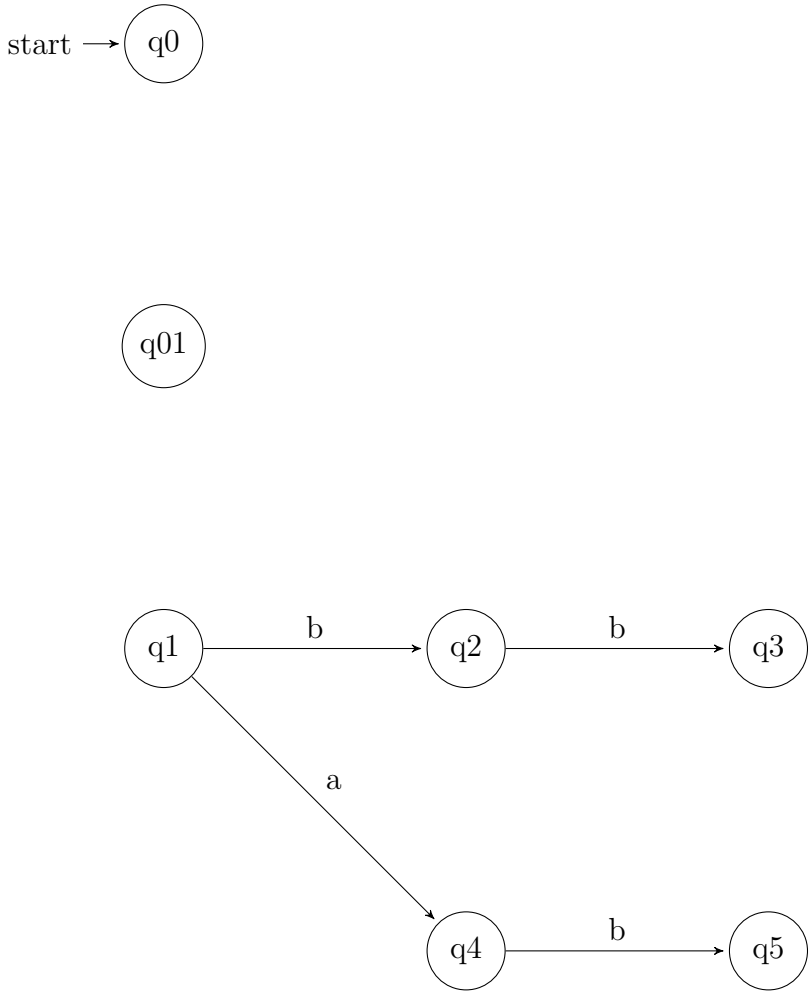
At the beginning, we break into two branches. In the first branch, for each a that is read, we push an a onto the stack. Then, we guess when the first b is read, and begin popping symbols from the stack for every b that we read. When the stack is empty, the only input that is remaining should be C, and then we read the input without adjusting the stack. If we read the symbols in the proper order, that is a followed by b followed by c, and the stack is empty by the time the B's are done reading, then we accept the state. Otherwise, we reject the state.

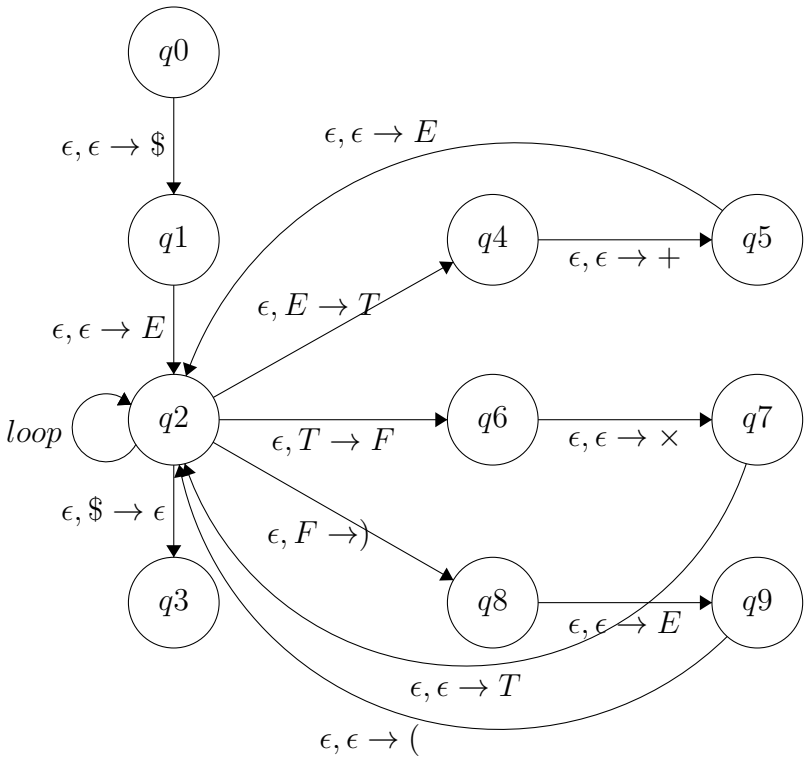
In the second branch, for a that we read, we do not adjust the stack. We then guess when the first b is read and we begin pushing symbols onto the stack for every b that we read. We then guess when the first c is read - and we begin popping symbols from the stack for every c that is read. If the stack becomes empty right when the input string is finished, and the symbols were read in the right order, we accept. Otherwise, we reject the state.

2.11 Convert the CFG G_4 given in Exercise 2.1 to an equivalent PDA, using the procedure given in Theorem 2.20.

Given CFG:
$E \to E + T \mid T$
$E \to T \times F \mid F$
$F \to (E) \mid a$

PDA for G_4:




$q0$
$\epsilon, \epsilon \to \$$
$q1$
$\epsilon, \epsilon \to E$
$loop$
$q2$
$\epsilon, \$ \to \epsilon$
$q3$
$\epsilon, \epsilon \to E$
$\epsilon, E \to T$
$q4$
$\epsilon, \epsilon \to +$
$q5$
$\epsilon, T \to F$
$q6$
$\epsilon, \epsilon \to \times$
$q7$
$\epsilon, F \to)$
$q8$
$\epsilon, \epsilon \to E$
$q9$
$\epsilon, \epsilon \to T$
$\epsilon, \epsilon \to ($

2.12 Convert the CFG G 4 given in Exercise 2.1 to an equivalent PDA, using the procedure given in Theorem 2.20.

PDA for G:

2.13 Let G = (V, $\sum$, R, S) be the following grammar. V = {S, T, U}; = {0, #}; and R is the set of rules:

$S \to TT \mid U$
$T \to 0T \mid T0 \mid \#$
$U \to 0U00 \mid \#$

a. Describe L(G) in English.
b. Prove that L(G) is not regular.

2.14 Convert the following CFG into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.

$A \to BAB \mid B \mid \epsilon$
$B \to 00 \mid \epsilon$

2.26
Show that if G is a CFG in Chomsky normal form, then for any string $w \epsilon L(G)$
of length $n \geq 1$, exactly $2n - 1$ steps are required for any derivation of w.

2.30

I hope this one isn't annoying...

2.31

I hope this one isn't annoying...

2.32

I hope this one isn't annoying...

2.47

I hope this one isn't annoying...