1 Regular Expressions

1) Describe in English, as briefly as possible, the language defined by each of these regular expressions:

a) (b ∪ ba) (b ∪ a)* (ab ∪ b).

The set of strings over the alphabet {a, b} that start and end with b.

b) (((a*b*)*ab) ∪ ((a*b*)*ba))(b ∪ a)*.

The set of strings over the alphabet {a, b} that contain at least one occurrence of ab or ba.

2) Write a regular expression to describe each of the following languages:

a) {*w* ∈ {a, b}* : every a in *w* is immediately preceded and followed by b}.

(b ∪ bab)*

b) {*w* ∈ {a, b}* : *w* does not end in ba}.

ε ∪ a ∪ (a ∪ b)* (b ∪ aa)

c) {*w* ∈ {0, 1}* : ∃*y* ∈ {0, 1}* (|*xy*| is even)}.

(0 ∪ 1)*

d) {*w* ∈ {0, 1}* : *w* corresponds to the binary encoding, without leading 0's, of natural numbers that are evenly divisible by 4}.

(1(0 ∪ 1)* 00) ∪ 0

e) {*w* ∈ {0, 1}* : *w* corresponds to the binary encoding, without leading 0's, of natural numbers that are powers of 4}.

1(00)*

f) {*w* ∈ {0-9}* : *w* corresponds to the decimal encoding, without leading 0's, of an odd natural number}.

(ε ∪ ((1-9)(0-9)*))(1 ∪ 3 ∪ 5 ∪ 7 ∪ 9)

g) {*w* ∈ {0, 1}* : *w* has 001 as a substring}.

(0 ∪ 1)* 001 (0 ∪ 1)*

h) {*w* ∈ {0, 1}* : *w* does not have 001 as a substring}.

(1 ∪ 01)* 0*

i) {*w* ∈ {a, b}* : *w* has bba as a substring}.

(a ∪ b)* bba (a ∪ b)*

j) {*w* ∈ {a, b}* : *w* has both aa and bb as substrings}.

(a ∪ b)* aa (a ∪ b)* bb (a ∪ b)* ∪ (a ∪ b)* bb (a ∪ b)* aa (a ∪ b)*

k) {*w* ∈ {a, b}* : *w* has both aa and aba as substrings}.

(a ∪ b)* aa (a ∪ b)* aba (a ∪ b)* ∪
(a ∪ b)* aba (a ∪ b)* aa (a ∪ b)* ∪
(a ∪ b)* aaba (a ∪ b)* ∪
(a ∪ b)* abaa (a ∪ b)*

l) {*w* ∈ {a, b}* : *w* contains at least two b's that are not followed by an a}.

(a ∪ b)* bb ∪ (a ∪ b)* bbb (a ∪ b)*

m) {*w* ∈ {0, 1}* : *w* has at most one pair of consecutive 0's and at most one pair of consecutive 1's}.

(ε ∪ 1)(01)*(ε ∪ 1) (01)* (ε ∪ (00 (ε ∪ (1 (01)* (ε ∪ 0))))) /* 11 comes first.
∪
(ε ∪ 0) (10)* (ε ∪ 0) (10)* (ε ∪ (11 (ε ∪ (0 (10)* (ε ∪ 1))))) /* 00 comes first.

n) {*w* ∈ {0, 1}* : none of the prefixes of *w* ends in 0}.

1*

o) {*w* ∈ {a, b}* : $\#_a(w) \equiv_3 0$}.

(b*ab*ab*a)*b*

p) $\{w \in \{a, b\}^* : \#_a(w) \le 3\}$.

$b^* (a \cup \varepsilon) b^* (a \cup \varepsilon) b^* (a \cup \varepsilon) b^*$

q) $\{w \in \{a, b\}^* : w$ contains exactly two occurrences of the substring aa$\}$.

(b $\cup$ ab)*aaa(b $\cup$ ba)* $\cup$ (b $\cup$ ab)*aab(b $\cup$ ab)*aa(b $\cup$ ba)* (Either the two occurrences are contiguous, producing aaa, or they're not.)

r) $\{w \in \{a, b\}^* : w$ contains no more than two occurrences of the substring aa$\}$.

(b $\cup$ ab)*(a $\cup$ ε) /* 0 occurrences of the substring aa
$\cup$
(b $\cup$ ab)*aa(b $\cup$ ba)* /* 1 occurrence of the substring aa
$\cup$
(b $\cup$ ab)*aaa(b $\cup$ ba)* $\cup$ (b $\cup$ ab)*aab(b $\cup$ ab)*aa(b $\cup$ ba)*
/* 2 occurrences of the substring aa

s) $\{w \in \{a, b\}^* - L\}$, where $L = \{w \in \{a, b\}^* : w$ contains bba as a substring$\}$.

(a $\cup$ ba)* ($\varepsilon \cup$ b $\cup$ bbb*) = (a $\cup$ ba)*b*

t) $\{w \in \{0, 1\}^* :$ every odd length string in L begins with 11$\}$.

((0 $\cup$ 1)(0 $\cup$ 1))* $\cup$ 11(0 $\cup$ 1)*

u) $\{w \in \{0\text{-}9\}^* : w$ represents the decimal encoding of an odd natural number without leading 0's$\}$.

($\varepsilon \cup$ ((1-9)(0-9)*))(1 $\cup$ 3 $\cup$ 5 $\cup$ 7 $\cup$ 9)

v) $L_1 - L_2$, where L_1 = a*b*c* and L_2 = c*b*a*.

$a^+b^+c^* \cup a^+b^*c^+ \cup a^*b^+c^+$ (The only strings that are in both L_1 and L_2 are strings composed of no more than one different letter. So those are the strings that we need to remove from L_1 to form the difference. What we're left with is strings that contain two or more distinct letters.)

w) The set of legal United States zipcodes 💻.

x) The set of strings that correspond to domestic telephone numbers in your country.

3) Simplify each of the following regular expressions:

a) (a $\cup$ b)* (a $\cup$ ε) b*.

(a $\cup$ b)*.

b) ($\varnothing^* \cup$ b) b*.

b*.

c) (a $\cup$ b)*a* $\cup$ b.

(a $\cup$ b)*.

d) ((a $\cup$ b)*)*.

(a $\cup$ b)*.

e) ((a $\cup$ b)$^+$)*.

(a $\cup$ b)*.

f) a ((a $\cup$ b)(b $\cup$ a))* $\cup$ a ((a $\cup$ b) a)* $\cup$ a ((b $\cup$ a) b)*.

a ((a $\cup$ b)(b $\cup$ a))*.

4) For each of the following expressions E, answer the following three questions and prove your answer:
(i) Is E a regular expression?
(ii) If E is a regular expression, give a simpler regular expression.
(iii) Does E describe a regular language?

a) ((a $\cup$ b) $\cup$ (ab))*.

E is a regular expression. A simpler one is (a $\cup$ b)*. The language is regular.

b) ($a^+ a^n b^n$).

E is not a regular expression. The language is not regular. It is $\{a^m b^n : m > n\}$.

c) ((ab)* $\varnothing$).

E is a regular expression. A simpler one is $\varnothing$. The language is regular.

d) (((ab) $\cup$ c)* $\cap$ (b $\cup$ c*)).

E is not a regular expression because it contains $\cap$. But it does describe a regular language (c*) because the regular languages are closed under intersection.

e) ($\varnothing^* \cup$ (bb*)).

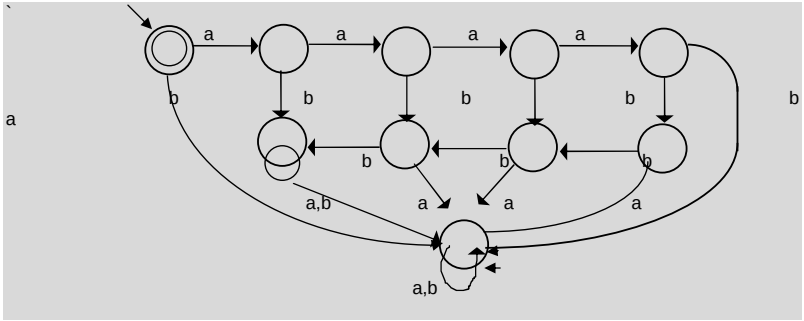
E is a regular expression. A (slightly) simpler one is ($\varepsilon \cup$ (bb*)). The language is regular.

5) Let $L = \{a^n b^n : 0 \le n \le 4\}$.

a) Show a regular expression for L.

($\varepsilon \cup$ ab $\cup$ aabb $\cup$ aaabbb $\cup$ aaaabbbb)

b) Show an FSM that accepts *L*.



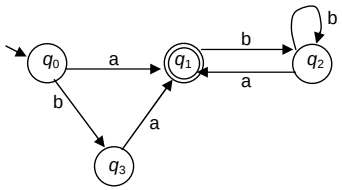
6) Let $L = \{w \in \{1, 2\}^* :$ for all prefixes p of w, if $|p| > 0$ and $|p|$ is even, then the last character of p is 1$\}$.
   a) Write a regular expression for *L*.

$((1 \cup 2)1)^* (1 \cup 2 \cup \varepsilon)$

   b) Show an FSM that accepts *L*.

7) Use the algorithm presented in the proof of Kleene's theorem to construct an FSM to accept the languages generated by the following regular expressions:
   a) (b(b ∪ ε)b)*.
   b) bab ∪ a*.

8) Let *L* be the language accepted by the following finite state machine:

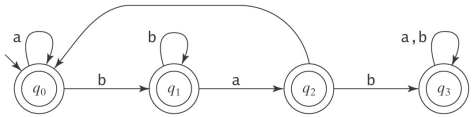


Indicate, for each of the following regular expressions, whether it correctly describes *L*:
   a) (a ∪ ba)bb*a.
   b) (ε ∪ b)a(bb*a)*.
   c) ba∪ ab*a.

d) (a ∪ ba)(bb*a)*.

a) no; b) yes; c) no; d) yes.

9) Consider the following FSM *M*:



The first printing of the book has two mistakes in this figure: The transition from q_2 back to q_0 should be labeled a, and state q_3 should not be accepting. We'll give answers for the printed version (in which we'll simply ignore the unlabeled transition from q_2 to q_0) and the correct version.

   a) Show a regular expression for *L*(*M*).

Printed version: a* ∪ a*bb* ∪ a*bb*a ∪ a*bb*ab (a ∪ b)*
Correct version: (a ∪ bb*aa)* (ε ∪ bb*(a ∪ ε)).

   b) Describe *L*(*M*) in English.

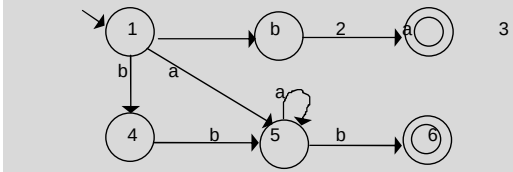
Printed version: No obvious concise description.
Correct version: All strings in {a, b}* that contain no occurrence of bab.

10) Consider the FSM *M* of Example 5.3. Use *fsmtoregexheuristic* to construct a regular expression that describes *L*(*M*).

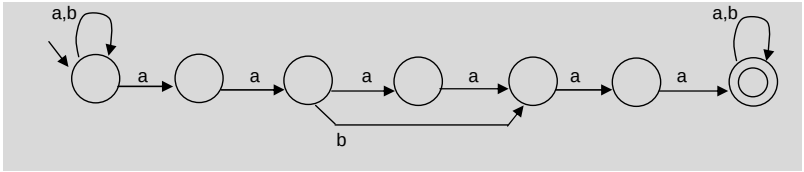
11) Consider the FSM *M* of Example 6.9. Apply *fsmtoregex* to *M* and show the regular expression that results.

12) Consider the FSM *M* of Example 6.8. Apply *fsmtoregex* to *M* and show the regular expression that results. (Hint: this one is exceedingly tedious, but it can be done.)

13) Show a possibly nondeterministic FSM to accept the language defined by each of the following regular expressions:
   a) $(((a \cup ba)\ b \cup aa)^*$.
   b) (b ∪ ε)(ab)*(a ∪ ε).
   c) (babb* ∪ a)*.
   d) (ba ∪ ((a ∪ bb) a*b)).

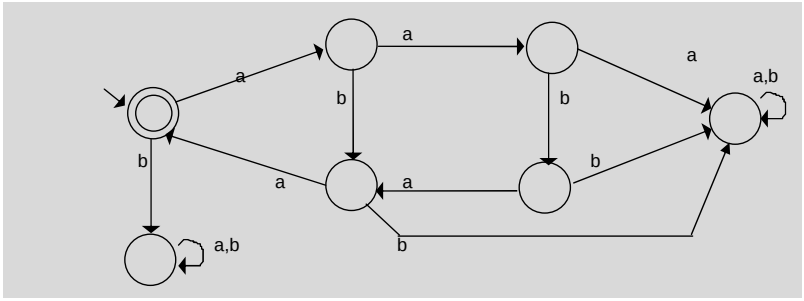


e) (a ∪ b)* aa (b ∪ aa) bb (a ∪ b)*.

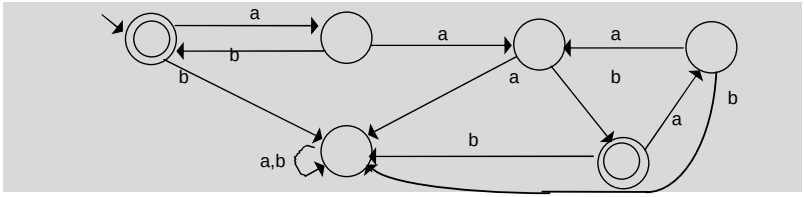


14) Show a DFSM to accept the language defined by each of the following regular expressions:

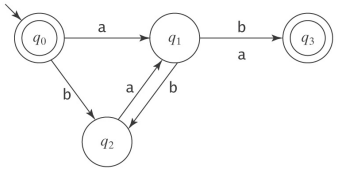
a) $(aba \cup aabaa)^*$



b) (ab)*(aab)*



15) Consider the following FSM *M*:



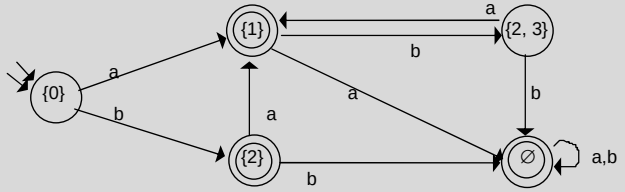
The first printing of the book has a mistake in this figure: There should not be an a labeling the transition from q_1 to q_3.

a) Write a regular expression that describes *L*(*M*).

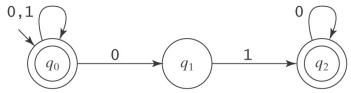
Printed version: ε ∪ ((a ∪ ba)(ba)*b ∪ a)
Correct version: ε ∪ ((a ∪ ba)(ba)*b).

b) Show a DFSM that accepts ¬*L*(*M*).

This is for the correct version (without the extra label a):



16) Given the following DFSM *M*, write a regular expression that describes ¬*L*(*M*):



Give a regular expression for ¬*L*(*M*).

We first construct a deterministic FSM *M** equivalent to *M*. *M** =
({{q_0}, {q_0, q_1}, {q_0, q_2}, {q_0, q_1, q_2}}, {0, 1}, δ*, {q_0}, {{q_0, q_2}, {q_0, q_1, q_2}}), where δ* =
{({q_0}, 0, {q_0, q_1}),
({q_0}, 1, {q_0}),
({q_0, q_1}, 0, {q_0, q_1}),
({q_0, q_1}, 1, {q_0, q_2}),
({q_0, q_2}, 0, {q_0, q_1, q_2}),
({q_0, q_2}, 1, {q_0}),
({q_0, q_1, q_2}, 0, {q_0, q_1, q_2}),
({q_0, q_1, q_2}, 1, {q_0, q_2})}

From *M**, we flip accepting and nonaccepting states to build *M***, which accepts ¬*L*(*M*). *M*** = *M** except that *A*** = {}.

So a regular expression that accepts *L*(*M***) is ∅.

17) Add the keyword able to the set in Example 6.13 and show the FSM that will be built by *buildkeywordFSM* from the expanded keyword set.

18) Let Σ = {a, b}. Let L = {ε, a, b}. Let R be a relation defined on Σ^* as follows: $\forall xy$ (xRy iff $y = xb$). Let R' be the reflexive, transitive closure of R. Let $L' = \{x : \exists y \in L\ (yR'x)\}$. Write a regular expression for L'.

R' = {(ε, ε), (ε, b), (ε, bb), (ε, bbb), … (a, a), (a, ab), (a, abb), … (b, b), (b, bb), (b, bbb), …}.

So a regular expression for L' is: $(\varepsilon \cup a \cup b)b^*$.

19) In Appendix O, we summarize the main features of the regular expression language in Perl. What feature of that regular expression language makes it possible to write regular expressions that describe languages that aren't regular?

The ability to store strings of arbitrary length in variables and then require that those variables match later in the string.

20) For each of the following statements, state whether it is *True* or *False*. Prove your answer.

a) $(ab)^*a = a(ba)^*$.

True.

b) $(a \cup b)^* b (a \cup b)^* = a^* b (a \cup b)^*$.

True.

c) $(a \cup b)^* b (a \cup b)^* \cup (a \cup b)^* a (a \cup b)^* = (a \cup b)^*$.

False.

d) $(a \cup b)^* b (a \cup b)^* \cup (a \cup b)^* a (a \cup b)^* = (a \cup b)^+$.

True.

e) $(a \cup b)^* b a (a \cup b)^* \cup a^*b^* = (a \cup b)^*$.

True.

f) $a^* b (a \cup b)^* = (a \cup b)^* b (a \cup b)^*$.

True.

g) If α and β are any two regular expressions, then $(\alpha \cup \beta)^* = \alpha(\beta\alpha \cup \alpha)$.

False.

h) If α and β are any two regular expressions, then $(\alpha\beta)^*\alpha = \alpha(\beta\alpha)^*$.

True.

2 Regular Gramamars

1) Show a regular grammar for each of the following languages:
   a) {$w \in$ {a, b}* : w contains an odd number of a's and an odd number of b's}.

G = ({EE, EO, OE, OO, a, b}, {a, b}, EE, R), where R =

$EE \rightarrow$ a OE	$EO \rightarrow \varepsilon$
$EE \rightarrow$ b EO	$EO \rightarrow$ a OO
$OE \rightarrow$ b OO	$EO \rightarrow$ b EE
$OE \rightarrow$ a EE	$OO \rightarrow$ a EO
	$OO \rightarrow$ b OE

   b) {$w \in$ {a, b}* : w does not end in aa}.

$S \rightarrow aA \mid bB \mid \varepsilon$
$A \rightarrow aC \mid bB \mid \varepsilon$
$B \rightarrow aA \mid bB \mid \varepsilon$
$C \rightarrow aC \mid bB$

   c) {$w \in$ {a, b}* : w contains the substring abb}.

   d) {$w \in$ {a, b}* : if w contains the substring aa then |w| is odd}.

It helps to begin by rewriting this as:

{$w \in$ {a, b}* : w does not contain the substring aa or |w| is odd}

The easiest way to design this grammar is to build an FSM first. The FSM has seven states:

- S, the start state, is never reentered after the first character is read.
- T_1: No aa yet; even length; last character was a.
- T_2: No aa yet; odd length; last character was a.
- T_3: No aa yet; even length; last character was b.
- T_4: No aa yet; odd length; last character was b.
- T_5: aa seen; even length.
- T_6: aa seen; odd length.

Now we build a regular grammar whose nonterminals correspond to those states. So we have:

$S \rightarrow aT_2 \mid bT_4$
$T_1 \rightarrow aT_6 \mid bT_4 \mid \varepsilon$
$T_2 \rightarrow aT_5 \mid bT_3 \mid \varepsilon$
$T_3 \rightarrow aT_2 \mid bT_4 \mid \varepsilon$
$T_4 \rightarrow aT_1 \mid bT_3 \mid \varepsilon$
$T_5 \rightarrow aT_6 \mid bT_6$
$T_6 \rightarrow aT_5 \mid bT_5 \mid \varepsilon$

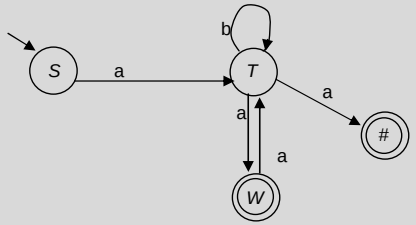
   e) {$w \in$ {a, b}* : w does not contain the substring aabb}.

$S \rightarrow aA$	$A \rightarrow aB$	$B \rightarrow aB$	$C \rightarrow aA$
$S \rightarrow bS$	$A \rightarrow bS$	$B \rightarrow bC$	$C \rightarrow \varepsilon$
$S \rightarrow \varepsilon$	$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	

2) Consider the following regular grammar G:
   $S \rightarrow aT$
   $T \rightarrow bT$
   $T \rightarrow a$
   $T \rightarrow aW$
   $W \rightarrow \varepsilon$
   $W \rightarrow aT$
   a) Write a regular expression that generates $L(G)$.

a (b $\cup$ aa) a

   b) Use *grammartofsm* to generate an FSM M that accepts $L(G)$.



3) Consider again the FSM M shown in Exercise 5.1. Show a regular grammar that generates $L(M)$.

4) Show by construction that, for every FSM M there exists a regular grammar G such that $L(G) = L(M)$.

1. Make M deterministic (to get rid of ε-transitions).
2. Create a nonterminal for each state in the new M.
3. The start state becomes the starting nonterminal
4. For each transition $\delta(T, a) = U$, make a rule of the form $T \rightarrow aU$.
5. For each accepting state T, make a rule of the form $T \rightarrow \varepsilon$.

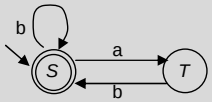
5) Let L = {$w \in$ {a, b}* : every a in w is immediately followed by at least one b}.
   a) Write a regular expression that describes L.

(ab $\cup$ b)*

   b) Write a regular grammar that generates L.

$S \rightarrow bS$
$S \rightarrow aT$
$S \rightarrow \varepsilon$
$T \rightarrow bS$

   c) Construct an FSM that accepts L.



3 Context-Free Grammars

1) Let $\Sigma = \{a, b\}$. For the languages that are defined by each of the following grammars, do each of the following:
i. List five strings that are in *L*.
ii. List five strings that are not in *L*.
iii. Describe *L* concisely. You can use regular expressions, expressions using variables (e.g., $a^n b^n$, or set theoretic expressions (e.g., {*x*: …})
iv. Indicate whether or not *L* is regular. Prove your answer.

a) $S \rightarrow aS \mid Sb \mid \varepsilon$

i. ε, $a, b, aaabbbb, ab$
ii. $ba, bbaa, bbbbba, ababab, aba$
iii. $L = a^* b^*$.
iv. *L* is regular because we can write a regular expression for it.

b) $S \rightarrow aSa \mid bSb \mid a \mid b$

i. $a, b, aaa, bbabb, aaaabaaaa$
ii. ε, $ab, bbbbbbba, bb, bbbaaa$
iii. *L* is the set of odd length palindromes, i.e., $L = \{w = x (a \cup b) x^R$, where $x \in \{a,b\}^*\}$.
iv. *L* is not regular. Easy to prove with pumping. Let $w = a^k baba^k$. *y* must be in the initial a region. Pump in and there will no longer be a palindrome.

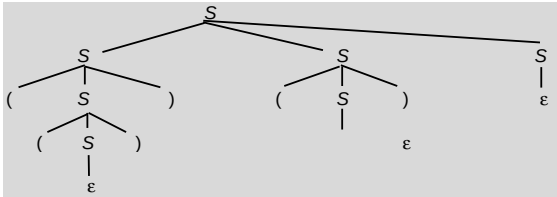
c) $S \rightarrow aS \mid bS \mid \varepsilon$

i. ε, a, aa, aaa, ba
ii. There aren't any over the alphabet $\{a, b\}$.
iii. $L = (a \cup b)^*$.
iv. *L* is regular because we can write a regular expression for it.

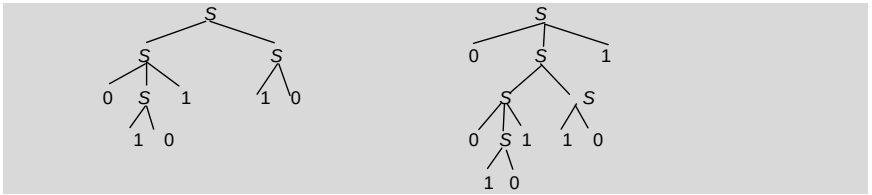
d) $S \rightarrow aS \mid aSbS \mid \varepsilon$

i. ε, $a, aaa, aaba, aaaabbbb$
ii. $b, bbaa, abba, bb$
iii. $L = \{w \in \{a, b\}^*$: in all prefixes *p* of *w*, $\#a(p) \geq \#b(p)\}$.
iv. *L* isn't regular. Easy to prove with pumping. Let $w = a^k b^k$. *y* is in the a region. Pump out and there will be fewer a's than b's.

2) Let *G* be the grammar of Example 11.12. Show a third parse tree that *G* can produce for the string (())().



3) Consider the following grammar *G*: $S \rightarrow 0S1 \mid SS \mid 10$
Show a parse tree produced by *G* for each of the following strings:
a) 010110
b) 00101101

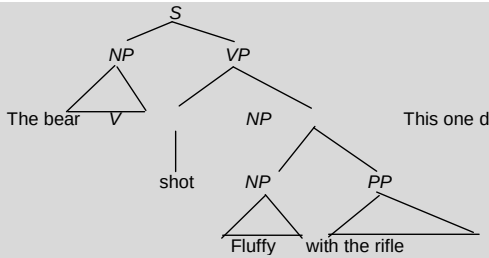


4) Consider the following context free grammar *G*:
$S \rightarrow aSa$
$S \rightarrow T$
$S \rightarrow \varepsilon$
$T \rightarrow bT$
$T \rightarrow cT$
$T \rightarrow \varepsilon$
One of these rules is redundant and could be removed without altering $L(G)$. Which one?

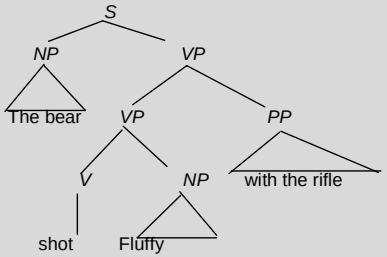
$S \rightarrow \varepsilon$

5) Using the simple English grammar that we showed in Example 11.6, show two parse trees for each of the following sentences. In each case, indicate which parse tree almost certainly corresponds to the intended meaning of the sentence:

a) The bear shot Fluffy with the rifle.

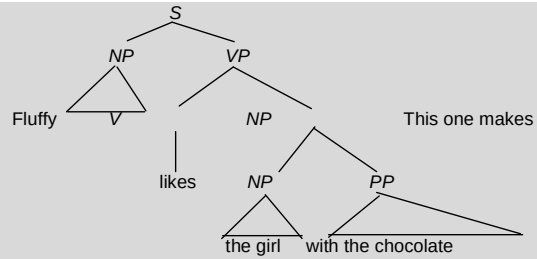


This one doesn't make much sense.

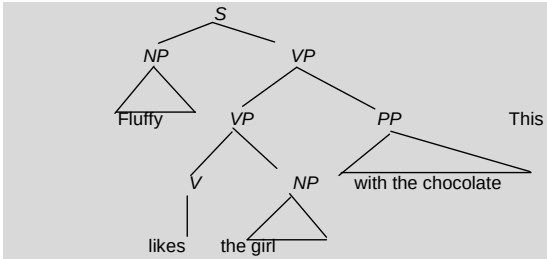


This one makes sense; the rifle was used for shooting.

b) Fluffy likes the girl with the chocolate.



This one makes sense.



This one doesn't make sense.

6) Show a context-free grammar for each of the following languages *L*:

a) BalDelim = {*w* : where *w* is a string of delimeters: (,), [,], {, }, that are properly balanced}.

$S \rightarrow (S) \mid [S] \mid \{ S \} \mid S S \mid \varepsilon$

b) $\{a^i b^j : 2i = 3j + 1\}$.

$S \rightarrow aaaSbb \mid aab$

c) $\{a^i b^j : 2i \neq 3j + 1\}$.

We can begin by analyzing L, as shown in the following table:

# of a's	Allowed # of b's
0	any
1	any
2	any except 1
3	any
4	any
5	any except 3
6	any
7	any
8	any except 5

$S \rightarrow aaaSbb$

$S \rightarrow aaaX$ /* extra a's

$S \rightarrow T$ /* terminate

$X \rightarrow A \mid A$ b /* arbitrarily more a's

$T \rightarrow A \mid B \mid$ a $B \mid$ aabb B /* note that if we add two more a's we cannot add just a single b.

$A \rightarrow$ a $A \mid \varepsilon$

$B \rightarrow$ b $B \mid \varepsilon$

d) $\{w \in \{a, b\}^* : \#_a(w) = 2\ \#_b(w)\}$.

$S \rightarrow SaSaSbS$
$S \rightarrow SaSbSaS$
$S \rightarrow SbSaSaS$
$S \rightarrow \varepsilon$

e) $\{w \in \{a, b\}^* : w = w^R\}$.

$S \rightarrow aSa$
$S \rightarrow bSb$
$S \rightarrow \varepsilon$
$S \rightarrow a$
$S \rightarrow b$ }

f) $\{a^i b^j c^k : i, j, k \geq 0$ and $(i \neq j$ or $j \neq k)\}$.

$S \rightarrow XC$ /* $i \neq j$
$S \rightarrow AY$ /* $j \neq k$
$X \rightarrow aXb$
$X \rightarrow A'$ /* at least one extra a
$X \rightarrow B'$ /* at least one extra b
$Y \rightarrow bYc \mid B' \mid C'$
$A' \rightarrow a\ A' \mid a$
$B' \rightarrow b\ B' \mid b$
$C' \rightarrow c\ C' \mid c$
$A \rightarrow a\ A \mid \varepsilon$
$C \rightarrow c\ C \mid \varepsilon$ }

g) $\{a^i b^j c^k : i, j, k \geq 0$ and $(k \leq i$ or $k \leq j)\}$.

$S \rightarrow A \mid B$
$A \rightarrow aAc \mid aA \mid M$
$B \rightarrow aB \mid F$
$F \rightarrow bFc \mid bF \mid \varepsilon$
$M \rightarrow bM \mid \varepsilon$

h) $\{w \in \{a, b\}^*$: every prefix of w has at least as many a's as b's$\}$.

$S \rightarrow aS \mid aSb \mid SS \mid \varepsilon$

i) $\{a^n b^m : m \geq n$, m-n is even$\}$.

$S \rightarrow aSb \mid S \rightarrow Sbb \mid \varepsilon$

j) $\{a^m b^n c^p d^q : m, n, p, q \geq 0$ and $m + n = p + q\}$.

For any string $a^m b^n c^p d^q \in L$, we will produce a's and d's in parallel for a while. But then one of two things will happen. Either $m \geq q$, in which case we begin producing a's and c's for a while, or $m \leq q$, in which case we begin producing b's and d's for a while. (You will see that it is fine that it is ambiguous what happens if $m = q$.) Eventually this process will stop and we will begin producing the innermost b's and c's for a while. Notice that any of those four phases could produce zero pairs. Since the four phases are distinct, we will need four nonterminals (since, for example, once we start producing c's, we do not want

ever to produce any d's again). So we have:
$S \rightarrow aSd$
$S \rightarrow T$
$S \rightarrow U$
$T \rightarrow aTc$
$T \rightarrow V$
$U \rightarrow bUd$
$U \rightarrow V$
$V \rightarrow bVc$
$V \rightarrow \varepsilon$

k) $\{x c^n : x \in \{a, b\}^*$ and $(\#_a(x) = n$ or $\#_b(x) = n)\}$.

$S \rightarrow A \mid B$
$A \rightarrow B'\ a\ B'\ A\ c \mid B'$
$B' \rightarrow b\ B' \mid \varepsilon$
$B \rightarrow A'\ b\ A'\ B\ c \mid A'$
$A' \rightarrow a\ A' \mid \varepsilon$

l) $\{b_i \# b_{i+1}{}^R : b_i$ is the binary representation of some integer i, $i \geq 0$, without leading zeros$\}$. (For example 101#011 $\in L$.)

L can be written as:
$0\#1 \cup \{1^k \#0^k 1 : k > 0\} \cup \{u01^k \#0^k 1u^R : k \geq 0$ and $u \in 1(0 \cup 1)^*\}$
So a grammar for L is:
$S \rightarrow 0\#1 \mid 1\ S_1\ 1 \mid 1\ S_2\ 1$
$S_1 \rightarrow 1\ S_1\ 0 \mid \#0$
$S_2 \rightarrow 1\ S_2\ 1 \mid 0\ S_2\ 0 \mid 0\ A\ 1$
$A \rightarrow 1\ A\ 0 \mid \#$

m) $\{x^R \# y : x, y \in \{0, 1\}^*$ and x is a substring of $y\}$.

$S \rightarrow S0 \mid S1 \mid S_1$
$S_1 \rightarrow 0S_1 0 \mid 1S_1 1 \mid T$
$T \rightarrow T1 \mid T0 \mid \#$

7) Let G be the ambiguous expression grammar of Example 11.14. Show at least three different parse trees that can be generated from G for the string id+id*id*id.

8) Consider the unambiguous expression grammar G' of Example 11.19.
a) Trace a derivation of the string id+id*id*id in G'.

$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow \text{id} + T \Rightarrow \text{id} + T * F \Rightarrow \text{id} + T * F * F \Rightarrow \text{id} + F * F * F \Rightarrow$
$\text{id} + \text{id} * F * F \Rightarrow \text{id} + \text{id} * \text{id} * F \Rightarrow \text{id} + \text{id} * \text{id} * \text{id}$

b) Add exponentiation (**) and unary minus (-) to G', assigning the highest precedence to unary minus, followed by exponentiation, multiplication, and addition, in that order.

$R = \{E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow F ** X$
$F \rightarrow X$
$X \rightarrow -X$
$X \rightarrow Y$
$Y \rightarrow (E)$
$Y \rightarrow$ id $\}$.

9) Let $L = \{w \in \{a, b, \cup, \varepsilon, (,), *, ^+\}^* : w$ is a syntactically legal regular expression$\}$.

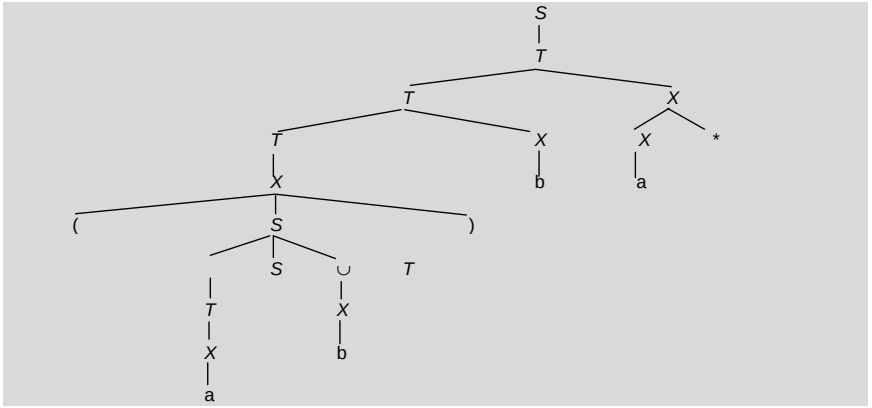
a) Write an unambiguous context-free grammar that generates L. Your grammar should have a structure similar to the arithmetic expression grammar G' that we presented in Example 11.19. It should create parse trees that:

- Associate left given operators of equal precedence, and
- Correspond to assigning the following precedence levels to the operators (from highest to lowest):
  - * and $^+$
  - concatenation
  - $\cup$

One problem here is that we want the symbol ε to be in Σ. But it is also generally a metasymbol in our rule-writing language. If we needed to say that a rule generates the empty string, we could use "" instead of ε. As it turns out, in this grammar we won't need to do that. We will, in this grammar, treat the symbol ε as a terminal symbol, just like $\cup$.

$S \rightarrow S \cup T$
$S \rightarrow T$
$T \rightarrow T X$
$T \rightarrow X$
$X \rightarrow X*$
$X \rightarrow X^+$
$X \rightarrow a$
$X \rightarrow b$
$X \rightarrow \varepsilon$
$X \rightarrow (S)$

b) Show the parse tree that your grammar will produce for the string $(a \cup b)ba^*$.



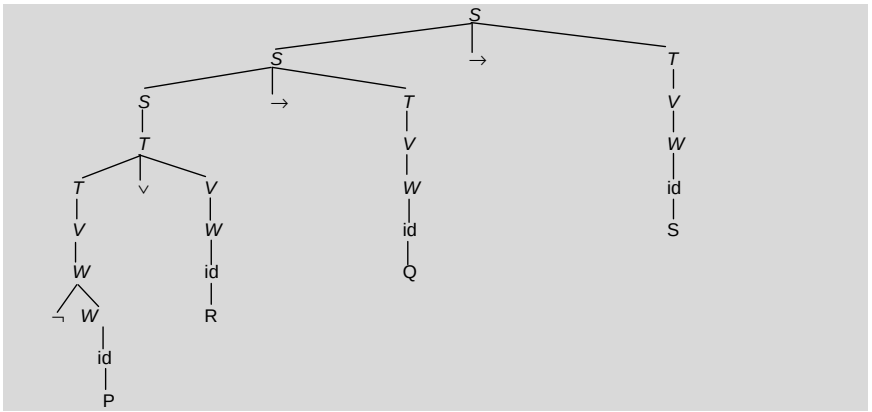
10) Let $L = \{w \in \{A - Z, \neg, \wedge, \vee, \rightarrow, (,)\}^* : w$ is a syntactically legal Boolean formula$\}$.

a) Write an unambiguous context-free grammar that generates L and that creates parse trees that:

- Associate left given operators of equal precedence, and
- Correspond to assigning the following precedence levels to the operators (from highest to lowest): $\neg$, $\wedge$, $\vee$, and $\rightarrow$.

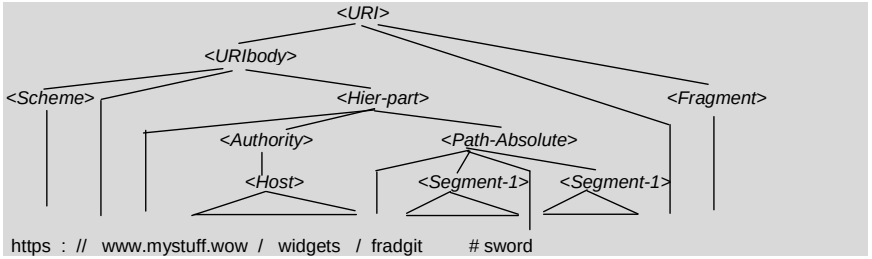
$S \rightarrow S \rightarrow T \mid T$
$T \rightarrow T \vee V \mid V$
$V \rightarrow V \wedge W \mid W$
$W \rightarrow \neg W \mid$ id $\mid (S)$

b) Show the parse tree that your grammar will produce for the string $\neg P \vee R \rightarrow Q \rightarrow S$.



11) In I.3.1, we present a simplified grammar for URIs (Uniform Resource Identifiers), the names that we use to refer to objects on the Web.
   a) Using that grammar, show a parse tree for:

      https://www.mystuff.wow/widgets/fradgit#sword



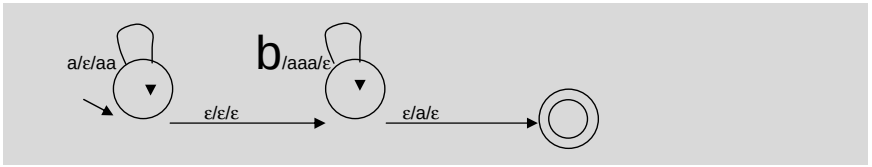
4 Pushdown Automata

1) Build a PDA to accept each of the following languages *L*:

   a) BalDelim = {*w* : where *w* is a string of delimeters: (,), [,], {, }, that are properly balanced}.

M = ({1}, {(,), [,], {, }}, {(, [, {}, Δ, 1, {1}), where Δ =
{ ((1, (, ε), (1, ()),
((1, [, ε), (1, []),
((1, {, ε), (1, {}),
((1,),)), (1, ε)),
((1,],]), (1, ε)),
((1, }, }), (1, ε)) }

   b) $\{a^i b^j : 2i = 3j + 1\}$.



   c) $\{w \in \{a, b\}^* : \#_a(w) = 2 \cdot \#_b(w)\}$.

The idea is that we only need one state. The stack will do all the work. It will count whatever it is ahead on. Since one a matches two b's, each a will push an a (if the machine is counting a's) and each b (if the machine is counting a's) will pop two of them. If, on the other hand, the machine is counting b's, each b will push two b's and each a will pop one. The only tricky case arises with inputs like aba. *M* will start out counting a's and so it will push one onto the stack. Then comes a b. It wants to pop two a's, but there's only one. So it will pop that one and then switch to counting b's by pushing a single b. The final a will then pop that b. *M* is highly nondeterministic. But there will be an accepting path iff the input string *w* is in *L*.

M = ({1}, {a, b}, {a, b}, Δ, 1, {1}), where Δ =
{ ((1, a, ε), (1, a)),
((1, a, b), (1, ε)),
((1, b, ε), (1, bb)),
((1, b, aa), (1, ε)),
((1, b, a), (1, b)) }

   d) $\{a^n b^m : m \le n \le 2m\}$.

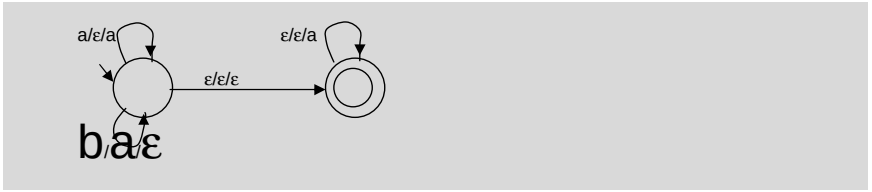
M = ({1, 2}, {a, b}, {a}, Δ, 1, {1, 2}), where Δ =
{ ((1, a, ε), (1, a)),
((1, ε, ε), (2, ε)),
((2, b, a), (2, ε)),
((2, b, aa), (2, ε)) }.

   e) $\{w \in \{a, b\}^* : w = w^R\}$.

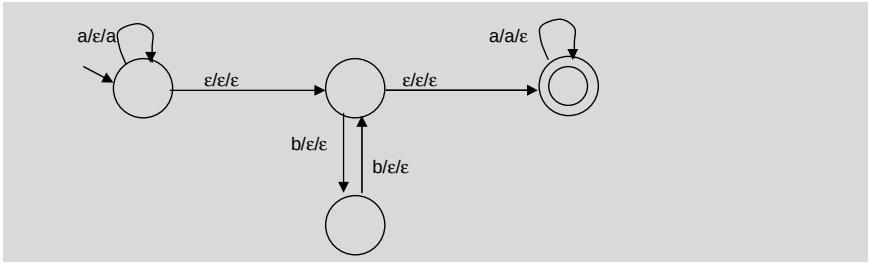
This language includes all the even-length palindromes of Example 12.5, plus the odd-length palindromes. So a PDA to accept it has a start state we'll call 1. There is a transition, from 1, labeled ε/ε/ε, to a copy of the PDA of Example 12.5. There is also a similarly labeled transition from 1 to a machine that is identical to the machine of Example 12.5 except that the transition from state *s* to state *f* has the following two labels: a/ε/ε and b/ε/ε. If an input string has a middle character, that character will drive the new machine through that transition.

   f) $\{a^i b^j c^k : i, j, k \ge 0 \text{ and } (i \ne j \text{ or } j \ne k)\}$.

   g) $\{w \in \{a, b\}^* :$ every prefix of *w* has at least as many a's as b's$\}$.



   h) $\{a^n b^m a^n : n, m \ge 0 \text{ and } m \text{ is even}\}$.



i) $\{x c^n : x \in \{a, b\}^*, \#a(x) = n \text{ or } \#b(x) = n\}$.

M will guess whether to count a's or b's. *M* = ({1, 2, 3, 4}, {a, b, c}, {c}, Δ, 1, {1, 4}), where Δ =
{ ((1, ε, ε), (2, ε)),
((1, ε, ε), (3, ε)),
((2, a, ε), (2, c)),
((2, b, ε), (2, ε)),
((2, ε, ε), (4, ε)),
((3, a, ε), (3, ε)),
((3, b, ε), (3, c)),
((3, ε, ε), (4, ε)),
((4, c, c), (4, ε)) }

j) $\{a^n b^m : m \geq n, m\text{-}n \text{ is even}\}$.

M = ({1, 2, 3, 4}, {a, b}, {a}, Δ, 1, {3}), where Δ =
{ ((1, a, ε), (1, a)),
((1, b, a), (2, ε)),
((1, ε, ε), (3, ε)),
((2, b, a), (2, ε)),
((2, ε, ε), (3, ε)),
((3, b, ε), (4, ε)),
((4, b, ε), (3, ε)) }

k) $\{a^m b^n c^p d^q : m, n, p, q \geq 0 \text{ and } m + n = p + q\}$.

l) $\{b_i \# b_{i+1}^R : b_i \text{ is the binary representation of some integer } i,\ i \geq 0, \text{ without leading zeros}\}$. (Example: 101#011 ∈ *L*.)

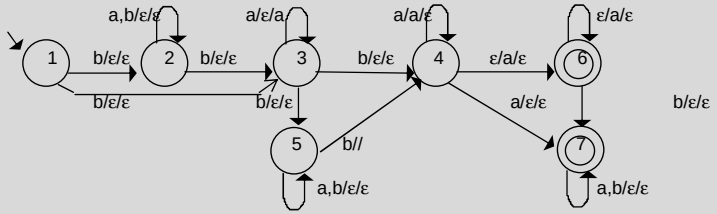
m) $\{x^R \# y : x, y \in \{0, 1\}^* \text{ and } x \text{ is a substring of } y\}$.

n) L_1^*, where $L_1 = \{xx^R : x \in \{a, b\}^*\}$.

M = ({1, 2, 3, 4}, {a, b}, {a, b, #}, Δ, 1, {4 }), where Δ =
{ ((1, ε, ε), (2, #)),
((2, a, ε), (2, a)),
((2, b, ε), (2, b)),
((2, ε, ε), (3, ε)),
((3, a, a), (3, ε)),
((3, b, b), (3, ε)),
((3, ε, #), (4, ε)),
((3, ε, #), (2, #)) }

2) Complete the PDA that we sketched, in Example 12.8, for $\neg A^n B^n C^n$, where $A^n B^n C^n = \{a^n b^n c^n : n \geq 0\}$.

3) Let $L = \{ba^{m1}ba^{m2}ba^{m1=3}\ldots ba^{mn} : n \geq 2, m_1, m_2, \ldots, m_n \geq 0, \text{ and } m_i \neq m_j \text{ for some } i, j\}$.
a) Show a PDA that accepts *L* .



We use state 2 to skip over an arbitrary number of ba^i groups that aren't involved in the required mismatch.
We use state 3 to count the first group of a's we care about.
We use state 4 to count the second group and make sure it's not equal to the first.
We use state 5 to skip over an arbitrary number of ba^i groups in between the two we care about.
We use state 6 to clear the stack in the case that the second group had fewer a's than the first group did.
We use state 7 to skip over any remaining ba^i groups that aren't involved in the required mismatch.

b) Show a context-free grammar that generates *L*.

$S \rightarrow A'bLA'$ /* L will take care of two groups where the first group has more a's
$S \rightarrow A'bRA'$ /* R will take care of two groups where the second group has more a's
$L \rightarrow aA'b \mid aL \mid aLa$
$R \rightarrow bA'a \mid Ra \mid aRa$
$A' \rightarrow bAA' \mid \varepsilon$ /* generates 0 or more ba* strings
$A \rightarrow aA \mid \varepsilon$

c) Prove that L is not regular.

Let $L_1 = ba^*ba^*$, which is regular because it can be described by a regular expression. If L is regular then $L_2 = L \cap L_1$ is regular. $L_2 = ba^nba^m$, $n \neq m$. $\neg L_2 \cap L_1$ must also be regular. But $\neg L_2 \cap L_1 = ba^nba^m$, $n = m$, which can easily be shown, using the pumping theorem, not to be regular. So we complete the proof by doing that.

4) Consider the language $L = L_1 \cap L_2$, where $L_1 = \{ww^R : w \in \{a, b\}^*\}$ and $L_2 = \{a^nb^*a^n : n \geq 0\}$.

a) List the first four strings in the lexicographic enumeration of L?

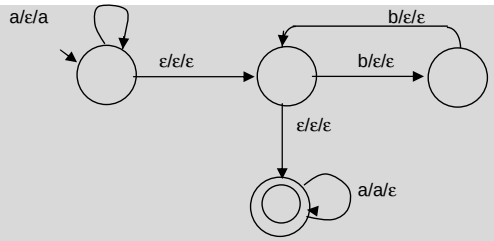
ε, aa, bb, aaaa

b) Write a context-free grammar to generate L.

Note that $L = \{a^nb^{2m}a^n : n, m \geq 0\}$.

$S \rightarrow aSa$
$S \rightarrow B$
$B \rightarrow bBb$
$B \rightarrow \varepsilon$

c) Show a natural pda for L. (In other words, don't just build it from the grammar using one of the two-state constructions presented in the book.)



d) Prove that L is not regular.

Note that $L = \{a^nb^{2m}a^n : n, m \geq 0\}$. We can prove that it is not regular using the Pumping Theorem. Let $w = a^kb^{2k}a^k$. Then y must fall in the first a region. Pump in once. The number of a's in the first a region no longer equals the number of a's in the second a region. So the resulting string is not in L. L is not regular.

5) Build a deterministic PDA to accept each of the following languages:

a) $L\$$, where $L = \{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$.

The idea is to use a bottom of stack marker so that M can tell what it should be counting. If the top of the stack is an a, it is counting a's. If the top of the stack is a b, it is counting b's. If the top of the stack is #, then it isn't counting anything yet. So if it is reading an a, it should start counting a's. If it is reading a b, it should start counting b's.

$M = (\{0, 1, 2\}, \{a, b\}, \{a, b\}, 0, \{2\}, \Delta)$, where $\Delta =$
{((0, ε, ε), (1, #)), ((1, $, #),(2, ε)), /* starting and ending.
((1, a, a), (1, aa)), ((1, b, a), (1, ε)), /* already counting a's.
((1, a, b), (1, ε)), ((1, b, b), (1, bb)), /* already counting b's.
((1, a, #), (1, a#)), ((1, b, #), (1, b#))} /* not yet counting anything. Start now.

b) $L\$$, where $L = \{a^nb^+a^m : n \geq 0$ and $\exists k \geq 0\ (m = 2k+n)\}$.

The number of a's in the second a region must equal the number in the first region plus some even number. M will work as follows: It will start by pushing # onto the stack as a bottom marker. In the first a region it will push one a for each a it reads. Then it will simply read all the b's without changing the stack. Then it will pop one a for each a it reads. When the # becomes the top of the stack again, unless $ appears at the same time, M will become a simple DFSM that has two states and checks that there is an even number of a's left to read. When it reads the $, it halts (and accepts).

$M = (\{1, 2, 3, 4, 5, 6, 7\}, \{a, b\}, \{a\}, 1, \{7\}, \Delta)$, where $\Delta =$
{((1, ε, ε), (2, #)),
((2, a, ε), (2, a)), ((2, b, ε), (3, ε)),
((3, b, ε), (3, ε)), ((3, a, a), (4, ε)),
((4, a, a), (4, ε)), ((4, $, #), (7, ε)), ((4, a, #), (6, ε)),
((5, a, ε), (6, ε)), ((5, $, ε), (7, ε)),
((6, a, ε), (5, ε)) }

6) Complete the proof that we started in Example 12.14. Specifically, show that if, M is a PDA that accepts by accepting state alone, then there exists a PDA M' that accepts by accepting state and empty stack (our definition) where $L(M') = L(M)$.

By construction: We build a new PDA P' from P as follows: Let P' initially be P. Add to P' a new accepting state F. From every original accepting state in P', add an epsilon transition to F. Make F the only accepting state in P'. For every element g of Γ, add the following transition to P': $((F, \varepsilon, g), (F, \varepsilon))$. In other words, if and only if P accepts, go to the only accepting state of P' and clear the stack. Thus P' will accept by accepting state and empty stack iff P accepts by accepting state.