# Introduction to Automata Theory, Languages, and Computation

## Solutions for Chapter 6

# Solutions for Section 6.1

### Exercise 6.1.1(a)

In what follows, *e* stands for epsilon, the empty string, and *Z* stands for the initial symbol, *Z_0*.

```
(q,01,Z) |- (q,1,XZ) |- (q,e,XZ) |- (p,e,Z)
                     |- (p,1,Z)  |- (p,e,e)
```

# Solutions for Section 6.2

### Exercise 6.2.1(a)

We shall accept by empty stack. Symbol *X* will be used to count the 0's on the input. In state *q*, the start state, where we have seen no 1's, we add an *X* to the stack for each 0 seen. The first *X* replaces *Z_0*, the start symbol. When we see a 1, we go to state *p*, and then only pop the stack, one *X* for each input 1. Formally, the PDA is *({q,p},{0,1},{X,Z_0},delta,q,Z_0)*. The rules:

1. *delta(q,0,Z_0) = {(q,X)}*
2. *delta(q,0,X) = {(q,XX)}*
3. *delta(q,1,X) = {(p,epsilon)}*
4. *delta(p,1,X) = {(p,epsilon)}*

### Exercise 6.2.2(a)

Revised 6/20/02.

Begin in start state *q0*, with start symbol *Z_0*, and immediately guess whether to check for:

1. *i=j=0* (state *q1*).
2. *i=j>0* (state *q2*).
3. *j=k* (state *q3*).

We shall accept by final state; as seen below, the accepting states are $q1$ and $q3$. The rules, and their explanations (again, $e$ stands for epsilon):

- $delta(q0,e,Z_0) = \{(q1,Z_0), (q2,Z_0), (q3,Z_0)\}$, the initial guess.
- $delta(q1,c,Z_0) = \{(q1,Z_0)\}$. In case (1), we assume there are no $a$'s or $b$'s, and we consume all $c$'s. State $q1$ will be one of our accepting states.
- $delta(q2,a,Z_0) = \{(q2,XZ_0)\}$, and $delta(q2,a,X) = \{(q2,XX)\}$. These rules begin case (2). We use $X$ to count the number of $a$'s read from the input, staying in state $q2$.
- $delta(q2,b,X) = delta(q4,b,X) = \{(q4,e)\}$. When $b$'s are seen, we go to state $q4$ and pop $X$'s against the $b$'s.
- $delta(q4,e,Z_0) = \{(q1,Z_0)\}$. If we reach the bottom-of-stack marker in state $q4$, we have seen an equal number of $a$'s and $b$'s. We go spontaneously to state $q1$, which will accept and consume all $c$'s, while continuing to accept.
- $delta(q3,a,Z_0) = \{(q3,Z_0)\}$. This rule begins case (3). We consume all $a$'s from the input. Since $j=k=0$ is possible, state $q3$ must be an accepting state.
- $delta(q3,b,Z_0) = \{(q5,XZ_0)\}$. When $b$'s arrive, we start counting them and go to state $q5$, which is not an accepting state.
- $delta(q5,b,X) = \{(q5,XX)\}$. We continue counting $b$'s.
- $delta(q5,c,X) = delta(q6,c,X) = \{(q6,e)\}$. When $c$'s arrive, we go to state $q6$ and match the $c$'s against the $b$'s.
- $delta(q6,e,Z_0) = \{(q3,e)\}$. When the bottom-of-stack marker is exposed in state $q6$, we have seen an equal number of $b$'s and $c$'s. We spontaneously accept in state $q3$, but we pop the stack so we cannot accept after reading more $a$'s.

## Exercise 6.2.4

Introduce a new state $q$, which becomes the initial state. On input epsilon and the start symbol of $P$, the new PDA has a choice of popping the stack (thus accepting epsilon), or going to the start state of $P$.

## Exercise 6.2.5(a)

We again use $e$ to represent epsilon.

$(q0,bab,Z_0)$ |- $(q2,ab,BZ_0)$ |- $(q3,b,Z_0)$ |- $(q1,b,AZ_0)$ |- $(q1,e,Z_0)$ |- $(q0,e,Z_0)$ |- $(f,e,e)$

## Exercise 6.2.8

Suppose that there is a rule that $(p,X1X2...Xk)$ is a choice in $delta(q,a,Z)$. We create $k-2$ new states $r1,r2,...,r\{k-2\}$ that simulate this rule but do so by adding one symbol at a time to the stack. That is, replace $(p,X1X2...Xk)$ in the rule by $(r\{k-2\},X\{k-1\}Xk$. Then create new rules $delta(r\{k-2\},e,X\{k-1\}) = \{(r\{k-1\},X\{k-2\}X\{k-1\})\}$, and so on, down to $delta(r2,e,X3) = \{(r1,X2X3)\}$ and $delta(r1,X2) = \{(p,X1X2)\}$.

# Solutions for Section 6.3

## Exercise 6.3.1

*({q},{0,1),{0,1,A,S},delta,q,S)* where *delta* is defined by:

1. *delta(q,e,S) = {(q,0S1), (q,A)}*
2. *delta(q,e,A) = {(q,1A0), (q,S), (q,e)}*
3. *delta(q,0,0) = {(q,e)}*
4. *delta(q,1,1) = {(q,e)}*

In the above, *e* represents the empty string.

## Exercise 6.3.3

In the following, *S* is the start symbol, *e* stands for the empty string, and *Z* is used in place of *Z_0*.

1. *S -> [qZq] | [qZp]*

   The following four productions come from rule (1).

2. *[qZq] -> 1[qXq][qZq]*
3. *[qZq] -> 1[qXp][pZq]*
4. *[qZp] -> 1[qXq][qZp]*
5. *[qZp] -> 1[qXp][pZp]*

   The following four productions come from rule (2).

6. *[qXq] -> 1[qXq][qXq]*
7. *[qXq] -> 1[qXp][pXq]*
8. *[qXp] -> 1[qXq][qXp]*
9. *[qXp] -> 1[qXp][pXp]*

   The following two productions come from rule (3).

10. *[qXq] -> 0[pXq]*
11. *[qXp] -> 0[pXp]*

    The following production comes from rule (4).

12. *[qXq] -> e*

    The following production comes from rule (5).

13. $[pXp] \rightarrow 1$

The following two productions come from rule (6).

14. $[pZq] \rightarrow 0[qZq]$
15. $[pZp] \rightarrow 0[qZp]$

### Exercise 6.3.6

Convert $P$ to a CFG, and then convert the CFG to a PDA, using the two constructions given in Section 6.3. The result is a one-state PDA equivalent to $P$.

# Solutions for Section 6.4

### Exercise 6.4.1(b)

Not a DPDA. For example, rules (3) and (4) give a choice, when in state $q$, with 1 as the next input symbol, and with $X$ on top of the stack, of either using the 1 (making no other change) or making a move on epsilon input that pops the stack and going to state $p$.

### Exercise 6.4.3(a)

Suppose a DPDA $P$ accepts both $w$ and $wx$ by empty stack, where $x$ is not epsilon (i.e., $N(P)$ does not have the prefix property). Then $(q_0,wxZ_0) \vdash^* (q,x,\epsilon)$ for some state $q$, where $q_0$ and $Z_0$ are the start state and symbol of $P$. It is not possible that $(q,x,\epsilon) \vdash^* (p,\epsilon,\epsilon)$ for some state $p$, because we know $x$ is not epsilon, and a PDA cannot have a move with an empty stack. This observation contradicts the assumption that $wx$ is in $N(P)$.

### Exercise 6.4.3(c)

Modify $P'$ in the following ways to create DPDA $P$:

1. Add a new start state and a new start symbol. $P$, with this state and symbol, pushes the start symbol of $P'$ on top of the stack and goes to the start state of $P'$. The purpose of the new start symbol is to make sure $P$ doesn't accidentally accept by empty stack.
2. Add a new ``popping state'' to $P$. In this state, $P$ pops every symbol it sees on the stack, using epsilon input.
3. If $P'$ enters an accepting state, $P$ enters the popping state instead.

As long as $L(P')$ has the prefix property, then any string that $P'$ accepts by final state, $P$ will accept by empty stack.