Chapter 8

Pushdown Automata

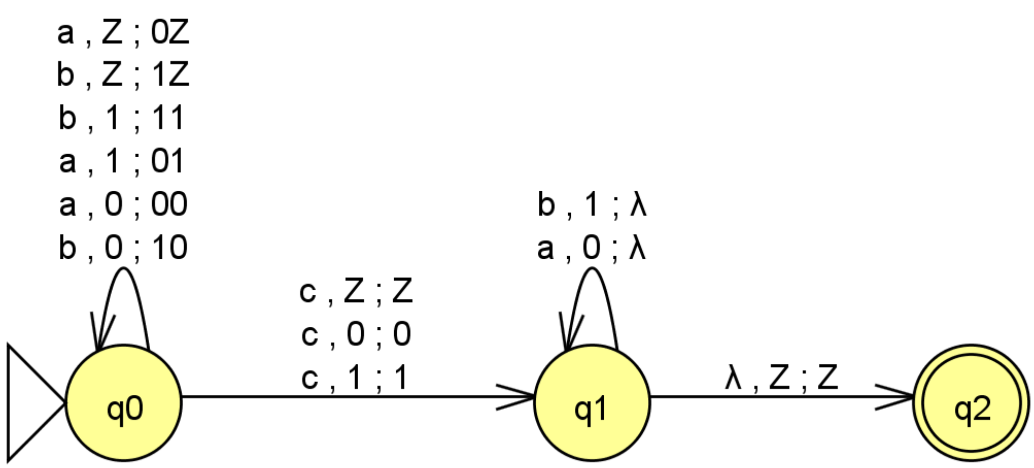
A. Construct a PDA for the following languages:

1. Odd palindromes wcw^R over {*a*, *b*, *c*} where $w = (a + b)^*$. Show an accepting sequence of configurations for the input *abbcbba*. Show how it rejects *abbcbb*. See Example 8.6.

   Solution: See Example 8.6 and Figure-8.9 for the PDA. The accepting sequence of configurations for *abbcbba* is:
   $(q_0, abbcbba, Z)$, $(q_0, bbcbba, 0Z)$, $(q_0, bcbba, 10Z)$, $(q_0, cbba, 110Z)$, $(q_1, bba, 110Z)$,
   $(q_1, ba, 10Z)$, $(q_1, a, 0Z)$, (q_1, λ, Z), (q_2, λ, Z)

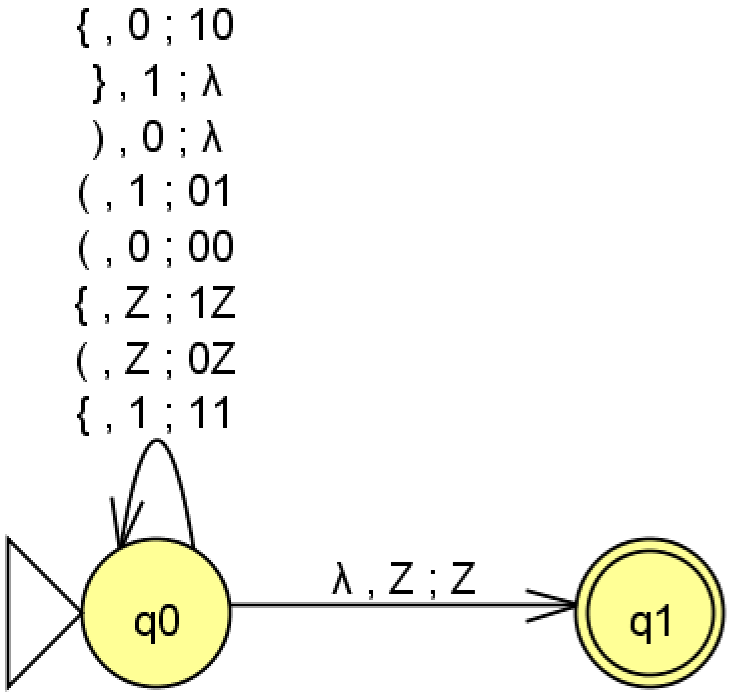
   For *abbcbb*, the rejecting sequence is:
   $(q_0, abbcbb, Z)$, $(q_0, bbcbb, 0Z)$, $(q_0, bcbb, 10Z)$, $(q_0, cbb, 110Z)$, $(q_1, bb, 110Z)$,
   $(q_1, b, 10Z)$, $(q_1, \lambda, 0Z)$

   That is, the PDA halts in a non-final state; the stack is also not empty and the string is rejected.



2. Proper nesting of parentheses and flower brackets. For example, {(())(){{()}{}}}. Show how it rejects {(){{(})}}.

   Solution: Please see also the VIDEO SOLUTION for this exercise.

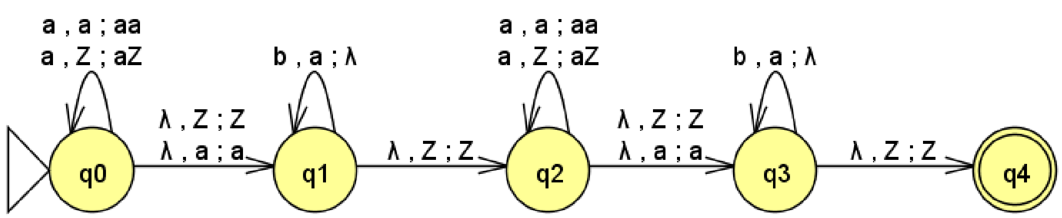


It can be seen that {(())(){{()}{}}} is accepted with the sequence (showing only successive stack contents):
Z, 1Z, 01Z, 001Z, 01Z, 1Z, 01Z, 1Z, 11Z, 111Z, 0111Z, 111Z, 11Z, 111Z, 11Z, 1Z, Z

The string {(){{(})}} is rejected with the sequence:
Z, 1Z, 01Z, 1Z, 11Z, 111Z, 0111Z,
At this point, there is a mismatch between the next input symbol } and the symbol 0 on the stack and the PDA halts right there, rejecting the input.

3. $a^n b^n a^m b^m$, $n \geq 0$, $m \geq 0$. Show, along with two different accepting sequences of configurations, how non-determinism works to accept the string *aaabbb* in two different ways.

Solution:



Accepting sequence 1 for *aaabbb*:
(q_0, *aaabbb*, Z), (q_0, *aabbb*, aZ), (q_0, *abbb*, aaZ), (q_0, *bbb*, $aaaZ$), (q_1, *bbb*, $aaaZ$),
(q_1, *bb*, aaZ), (q_1, *b*, aZ), (q_1, λ, Z), (q_2, λ, Z), (q_3, λ, Z), (q_4, λ, Z).

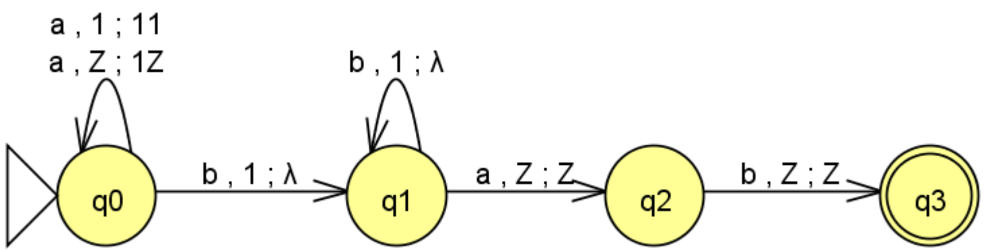
Accepting sequence 2 for *aaabbb*:

$(q_0, aaabbb, Z)$, $(q_1, aaabbb, Z)$, $(q_2, aaabbb, Z)$, $(q_2, aabbb, aZ)$, $(q_2, abbb, aaZ)$, $(q_2, bbb, aaaZ)$, $(q_3, bbb, aaaZ)$, (q_3, bb, aaZ), (q_3, b, aZ), (q_3, λ, Z), (q_4, λ, Z).

In fact, there is a third accepting sequence:
$(q_0, aaabbb, Z)$, $(q_0, aabbb, aZ)$, $(q_0, abbb, aaZ)$, $(q_0, bbb, aaaZ)$, $(q_1, bbb, aaaZ)$,
$(q_2, bbb, aaaZ)$, $(q_3, bbb, aaaZ)$, (q_3, bb, aaZ), (q_3, b, aZ), (q_3, λ, Z), (q_4, λ, Z).

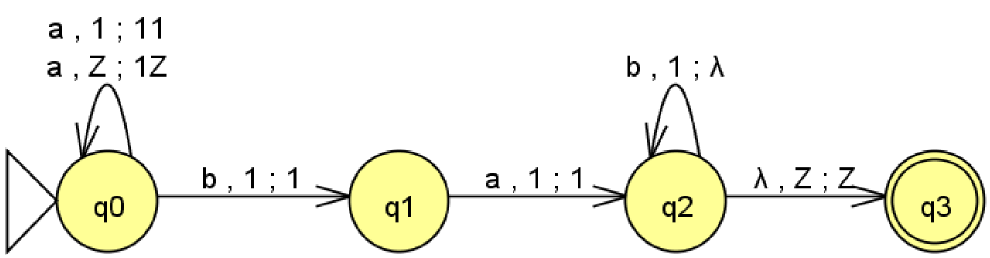
4. $a^n b^n ab$, $n > 0$. Make sure that the PDA is deterministic.

   Solution: Please see also the VIDEO SOLUTION for this exercise. Note that the video solution has a lambda transition; this is not quite deterministic. The solution below corrects this by eliminating the lambda transition.



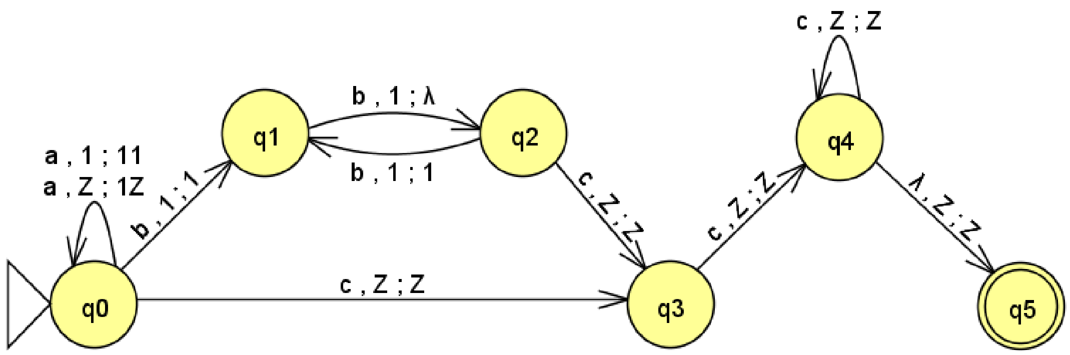
5. $a^n bab^n$, $n > 0$. Make sure that the PDA is deterministic.

   Solution:



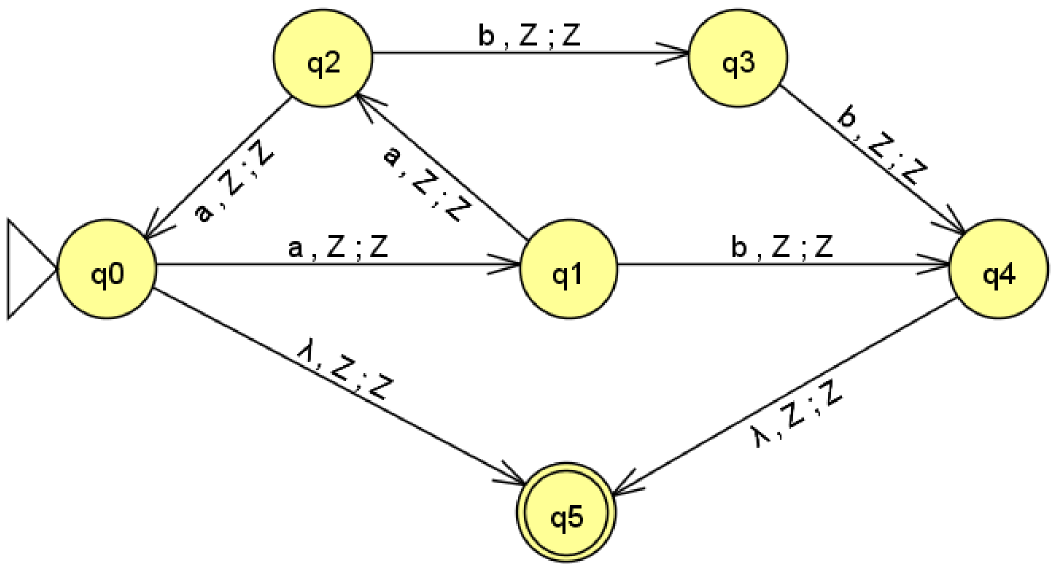
6. $a^n b^m c^k$ where $2n = m$ and $k \geq 2$. Make sure that the PDA is deterministic.

   Solution:



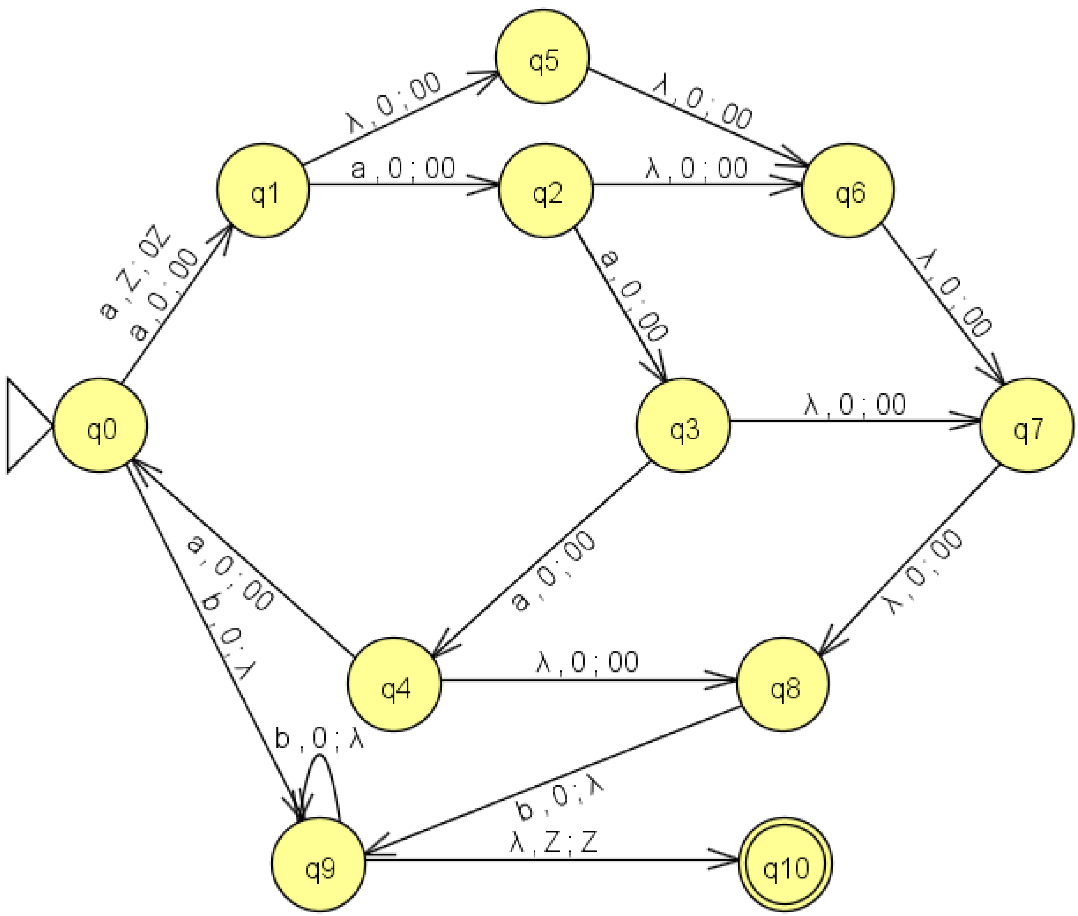
7. $a^n b^m$ where $m = n$ mod 3. How much stack memory do you need to handle this language?

   Solution: The stack is not needed at all; this is a regular language!



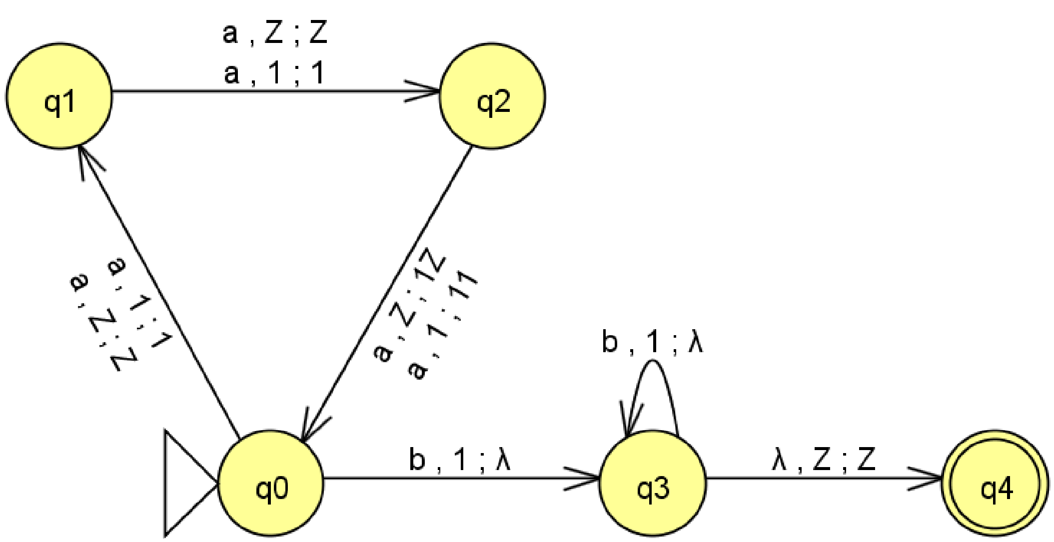
8. $a^n b^m$ where m is the nearest number equal to or higher than n that is divisible by 5. How much stack memory do you need to handle this language? How many states do you need in the PDA? Explain.

   Solution: The strategy used by the PDA shown below is to push additional symbols on to the stack when the *a* s are over to make the total number divisible by 5. As such, the amount of stack memory used is *m*. The PDA has 10 states as shown below.



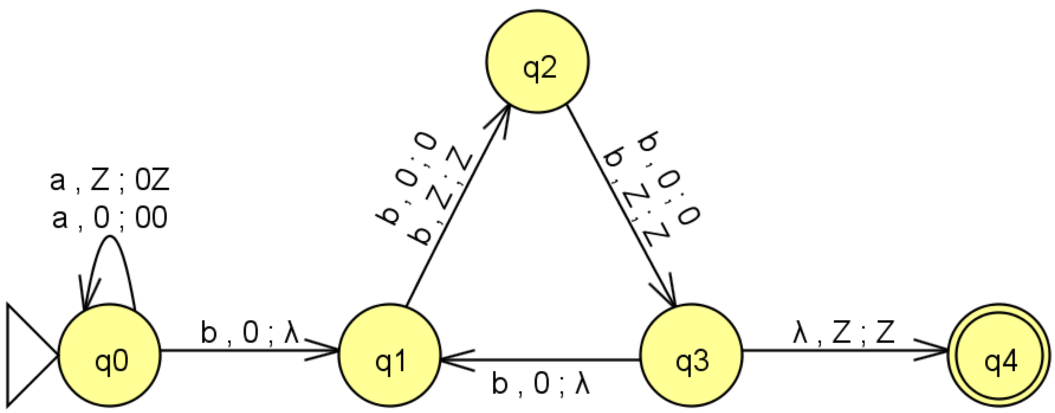
9. a^nb^m where n is a multiple of 3 and m is $n/3$. Make sure that the PDA is deterministic.

Solution:



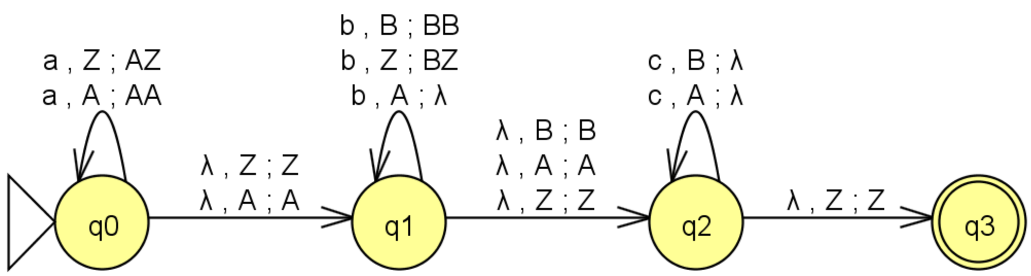
10. a^nb^m where $m = n \times 3$. Make sure that the PDA is deterministic.

Solution: Please see also the VIDEO SOLUTION for this exercise.



11. The language of subtraction, that is, strings of the form $a^n b^m c^k$ where $k = n - m$ if $n \geq m$ or else $k = m - n$. Show an accepting sequence of configurations for the input *aabbbbcc*. Show the rejecting sequence of configurations for the input *aaaabbccc*.

Solution:



The accepting sequence of configurations for *aabbbbcc* is:
$(q_0, aabbbbcc, Z)$, $(q_0, abbbbcc, AZ)$, $(q_0, bbbbcc, AAZ)$, $(q_1, bbbbcc, AAZ)$, $(q_1, bbbcc, AZ)$, $(q_1, bbcc, Z)$, (q_1, bcc, BZ), (q_1, cc, BBZ), (q_2, cc, BBZ), (q_2, c, BZ), (q_2, λ, Z), (q_3, λ, Z)

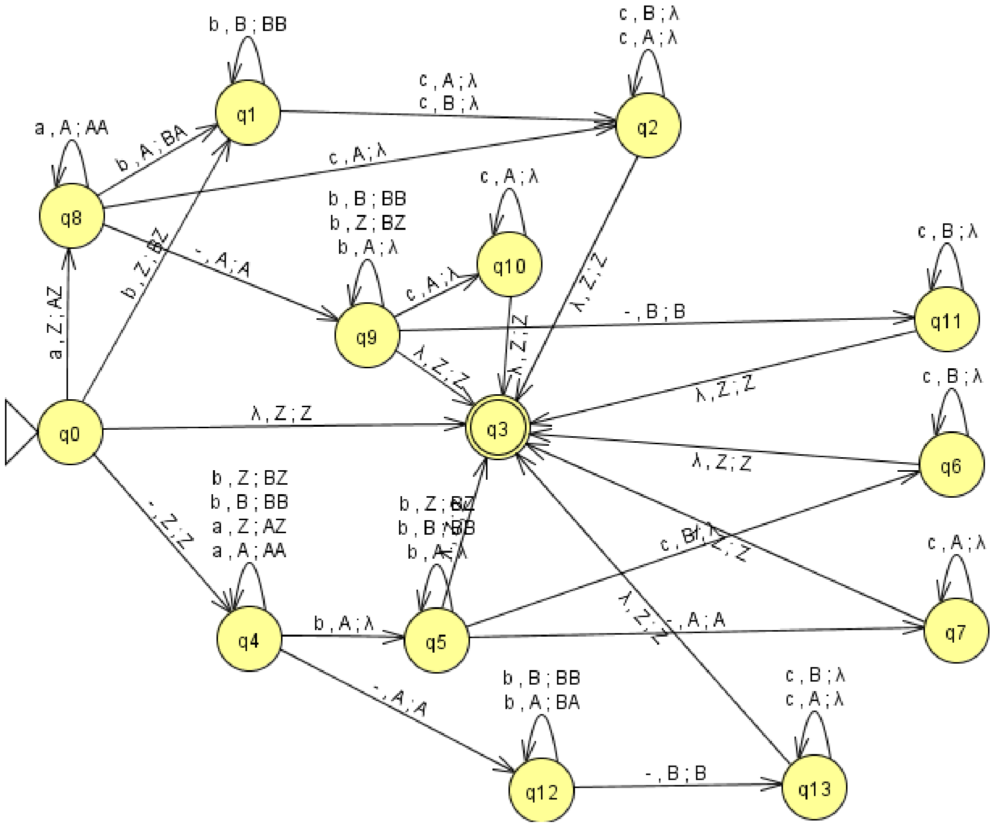
The rejecting sequence of configurations for *aaaabbccc* is:
$(q_0, aaaabbccc, Z)$, $(q_0, aaabbccc, AZ)$, $(q_0, aabbccc, AAZ)$, $(q_0, abbccc, AAAZ)$, $(q_0, bbccc, AAAAZ)$, $(q_1, bbccc, AAAAZ)$, $(q_1, bccc, AAAZ)$, (q_1, ccc, AAZ), (q_2, ccc, AAZ), (q_2, cc, AZ), (q_2, c, Z)

12. The language of addition with positive or negative numbers, that is, strings over the alphabet {*a*, *b*, *c*, –} of the form:
    (a) $a^n b^m c^k$ where $k = n + m$ and both numbers are positive; for example, *aabbcccc*.
    (b) $-a^n b^m c^k$ where $k = m - n$ if $m \geq n$ and the first number is negative; for example, *-aabbbbcc.*
    (c) $-a^n b^m\text{-}c^k$ where $k = n - m$ if $n > m$ and the first number is negative; for example, *-aaaabb-cc*.
    (d) $a^n\text{-}b^m c^k$ where $k = n - m$ if $n \geq m$ and the second number is negative; for example, *aaaa-bbcc*.
    (e) $a^n\text{-}b^m\text{-}c^k$ where $k = m - n$ if $m > n$ and the second number is negative; for example, *aa-bbbb-cc*.
    (f) $-a^n\text{-}b^m\text{-}c^k$ where $k = n + m$ and both numbers are negative; for example,

-aaa-bbb-cccccc.

Show an accepting sequence of configurations for each of the example strings shown above.

Solution: To Do : SAVE DIAGRAM



(a) Accepting sequence of configurations for *aabbcccc*:

$(q_0, aabbcccc, Z)$, $(q_8, abbcccc, AZ)$, $(q_8, bbcccc, AAZ)$, $(q_1, bcccc, BAAZ)$, $(q_1, cccc, BBAAZ)$, $(q_2, ccc, BAAZ)$, (q_2, cc, AAZ), (q_2, c, AZ), (q_2, λ, Z), (q_3, λ, Z)

(b) Accepting sequence of configurations for *-aabbbbcc*:

$(q_0, \text{-}aabbbbcc, Z)$, $(q_4, aabbbbcc, Z)$, $(q_4, abbbcc, AZ)$, $(q_4, bbbbcc, AAZ)$, $(q_5, bbbcc, AZ)$, $(q_5, bbcc, Z)$, (q_5, bcc, BZ), (q_5, cc, BBZ), (q_6, c, BZ), (q_6, λ, Z), (q_3, λ, Z)

(c) Accepting sequence of configurations for *-aaaabb-cc*:

$(q_0, \text{-}aaaabb\text{-}cc, Z)$, $(q_4, aaaabb\text{-}cc, Z)$, $(q_4, aaabb\text{-}cc, AZ)$, $(q_4, aabb\text{-}cc, AAZ)$, $(q_4, abb\text{-}cc, AAAZ)$, $(q_4, bb\text{-}cc, AAAAZ)$, $(q_5, b\text{-}cc, AAAZ)$, $(q_5, \text{-}cc, AAZ)$, (q_7, cc, AAZ), (q_7, c, AZ), (q_7, λ, Z), (q_3, λ, Z)

(d) Accepting sequence of configurations for *aaaa-bbcc*:

$(q_0, aaaa\text{-}bbcc, Z)$, $(q_8, aaa\text{-}bbcc, AZ)$, $(q_8, aa\text{-}bbcc, AAZ)$, $(q_8, a\text{-}bbcc, AAAZ)$, $(q_8, \text{-}bbcc, AAAAZ)$, $(q_9, bbcc, AAAAZ)$, $(q_9, bcc, AAAZ)$, (q_9, cc, AAZ), (q_{10}, c, AZ), (q_{10}, λ, Z), (q_3, λ, Z)

(e) Accepting sequence of configurations for *aa-bbbb-cc*:

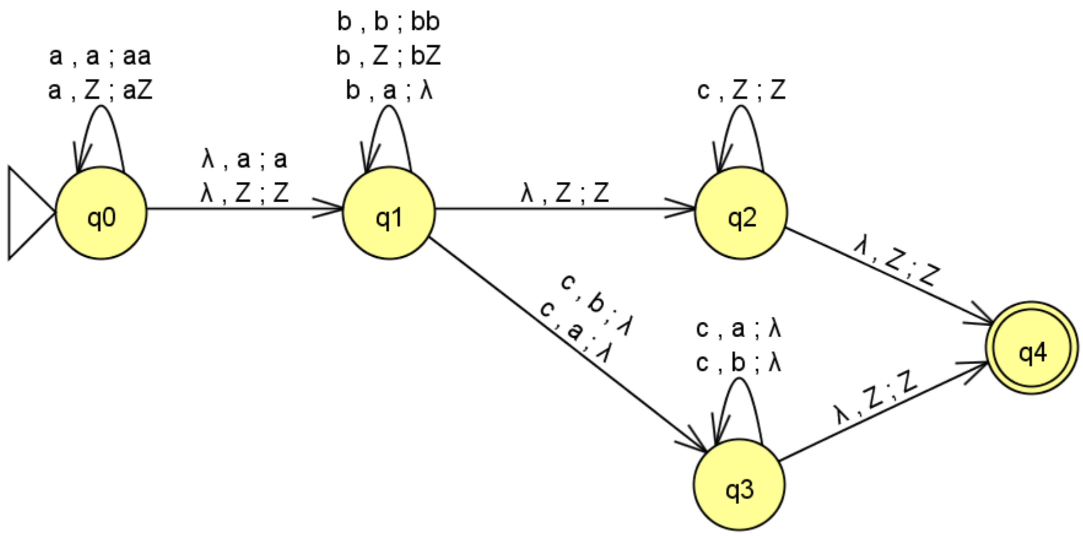
$(q_0, aa\text{-}bbbb\text{-}cc, Z)$, $(q_8, a\text{-}bbbb\text{-}cc, AZ)$, $(q_8, \text{-}bbbb\text{-}cc, AAZ)$, $(q_9, bbbb\text{-}cc, AAZ)$, $(q_9, bbb\text{-}cc, AZ)$, $(q_9, bb\text{-}cc, Z)$, $(q_9, b\text{-}cc, BZ)$, $(q_9, \text{-}cc, BBZ)$, (q_{11}, cc, BBZ), (q_{11}, c, BZ), (q_{11}, λ, Z), (q_3, λ, Z)

(f) Accepting sequence of configurations for *-aaa-bbb-cccccc*:

(q_0, *-aaa-bbb-cccccc*, *Z*), (q_4, *aaa-bbb-cccccc*, *Z*), (q_4, *aa-bbb-cccccc*, *AZ*), (q_4, *a-bbb-cccccc*, *AAZ*), (q_4, *-bbb-cccccc*, *AAAZ*), (q_{12}, *bbb-cccccc*, *AAAZ*), (q_{12}, *bb-cccccc*, *BAAAZ*), (q_{12}, *b-cccccc*, *BBAAAZ*), (q_{12}, *-cccccc*, *BBBAAAZ*), (q_{13}, *cccccc*, *BBBAAAZ*), (q_{13}, *ccccc*, *BBAAAZ*), (q_{13}, *cccc*, *BAAAZ*), (q_{13}, *ccc*, *AAAZ*), (q_{13}, *cc*, *AAZ*), (q_{13}, *c*, *AZ*), (q_{13}, *λ*, *Z*), (q_3, *λ*, *Z*)

13. $a^i b^j c^k$ where either $i = j$ (and k is any number) or k is the difference between i and j. Show an accepting sequence of configurations for the input *aabbbc*. Show how the PDA rejects both *aabbbbc* and *aaaabbbcc*.

Solution:



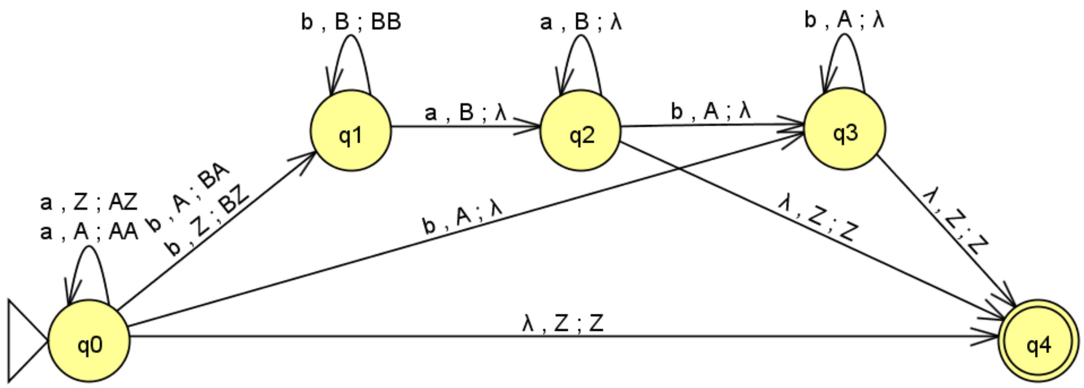
The accepting sequence of configurations for *aabbbc* is:
(q_0, *aabbbc*, *Z*), (q_0, *abbbc*, *aZ*), (q_0, *bbbc*, *aaZ*), (q_1, *bbbc*, *aaZ*), (q_1, *bbc*, *aZ*), (q_1, *bc*, *Z*), (q_1, *c*, *bZ*), (q_3, *λ*, *Z*), (q_4, *λ*, *Z*)

The rejecting sequence of configurations for *aabbbbc* is:
(q_0, *aabbbbc*, *Z*), (q_0, *abbbbc*, *aZ*), (q_0, *bbbbc*, *aaZ*), (q_1, *bbbbc*, *aaZ*), (q_1, *bbbc*, *aZ*), (q_1, *bbc*, *Z*), (q_1, *bc*, *bZ*), (q_1, *c*, *bbZ*), (q_3, *λ*, *bZ*)

The rejecting sequence of configurations for *aaaabbbcc* is:
(q_0, *aaaabbbcc*, *Z*), (q_0, *aaabbbcc*, *aZ*), (q_0, *aabbbcc*, *aaZ*), (q_0, *abbbcc*, *aaaZ*), (q_0, *bbbcc*, *aaaaZ*), (q_1, *bbbcc*, *aaaaZ*), (q_1, *bbcc*, *aaaZ*), (q_1, *bcc*, *aaZ*), (q_1, *cc*, *aZ*), (q_3, *c*, *Z*)

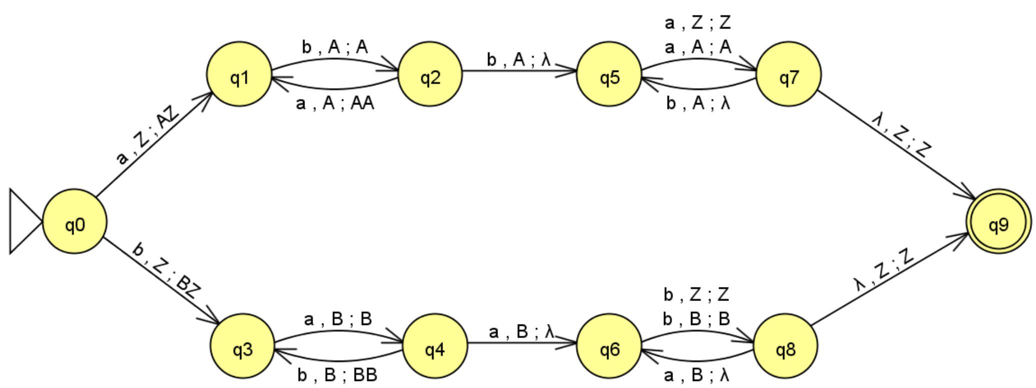
14. $a^n b^m a^m b^n$. Can this be a deterministic PDA? Explain.

Solution: No, it is nondeterministic because, after pushing the *a* s onto the stack ($n > 0$), when the first *b* is encountered, it is not known whether a symbol should be pushed ($m > 0$) or popped ($m = 0$).



15. ww^R where each $w = (ab)^* + (ba)^*$. Unlike in the generic case where w can be anything, can this be a deterministic PDA? Explain. Can you also ensure that the number of stack cells used is just one-fourth the length of the input string?

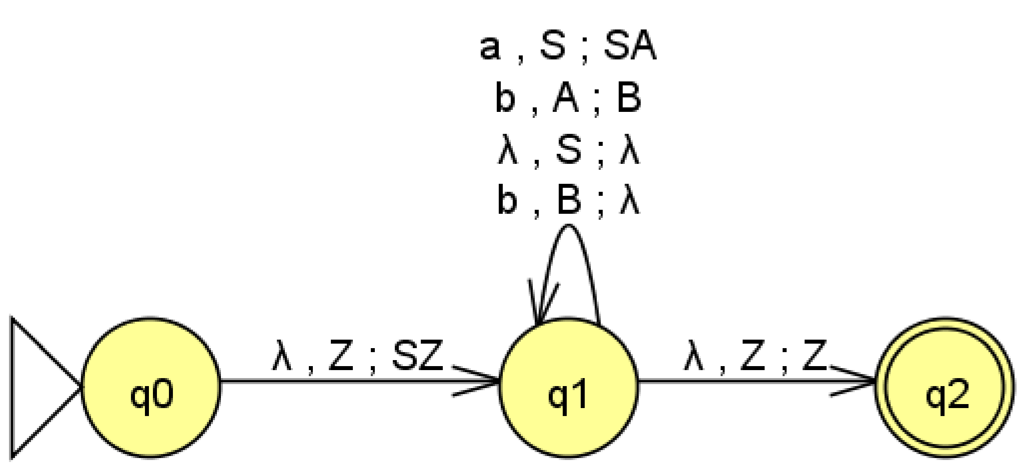
Solution: Yes, this is deterministic (the midpoint is known when the symbols change from *ab* to *ba* or vice versa) and yes, the stack size required is just one fourth since a single symbol can be used to count either *ab* or *ba*.



B. Convert each of the following context-free grammars to an equivalent PDA:

16. $S \rightarrow aSA \mid \lambda$, $A \rightarrow bB$, $B \rightarrow b$

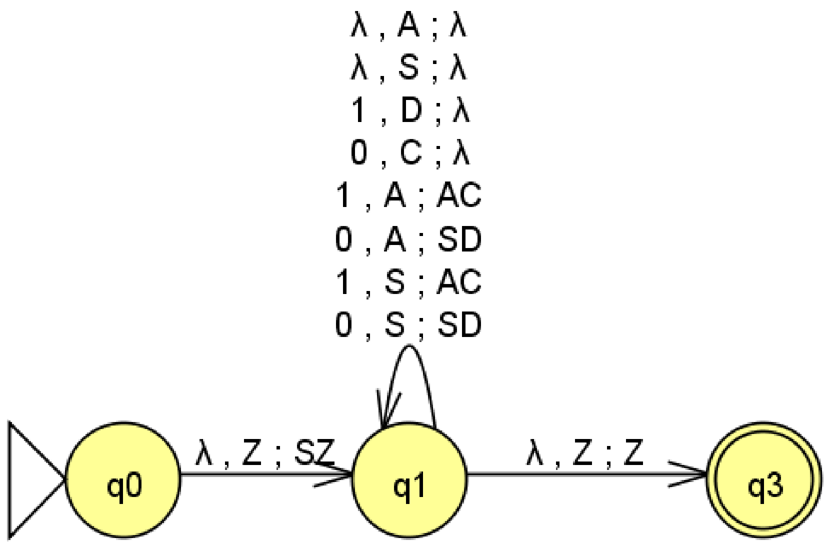
Solution:



17. $S \to 0S1 \mid A$, $A \to 1A0 \mid S \mid \lambda$

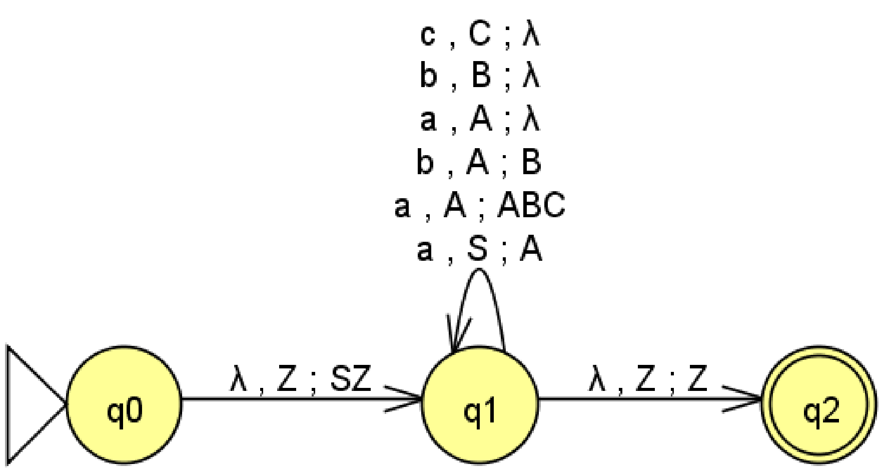
Solution: First, we need to convert the grammar to GNF:
$S \to 0SD \mid 1AC \mid \lambda$, $A \to 1AC \mid 0SD \mid \lambda$, $C \to 0$, $D \to 1$

The equivalent PDF is:



18. $S \to aA$, $A \to aABC \mid bB \mid a$, $B \to b$, $C \to c$. Show how *aaabc* is accepted.

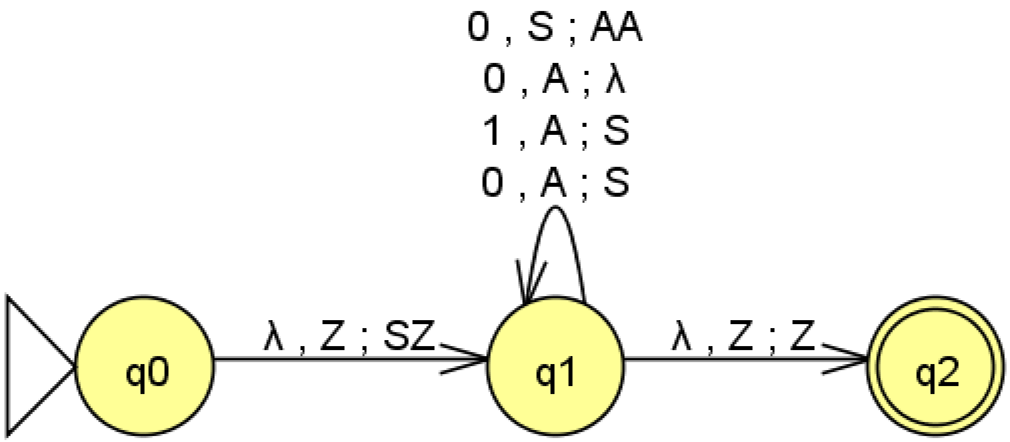
Solution:



The string *aaabc* is accepted via:
$(q_0, aaabc, Z)$, $(q_1, aaabc, SZ)$, $(q_1, aabc, AZ)$, $(q_1, abc, ABCZ)$, (q_1, bc, BCZ), (q_1, c, CZ),
(q_1, λ, Z), (q_2, λ, Z)

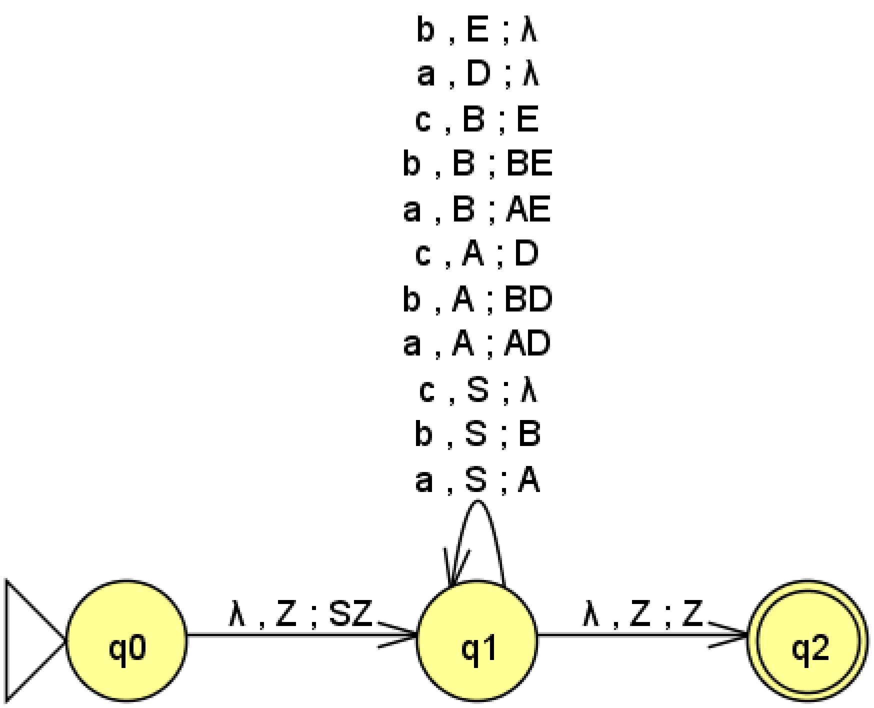
19. $S \to 0AA$, $A \to 0S \mid 1S \mid 0$

Solution: Please see also the VIDEO SOLUTION.



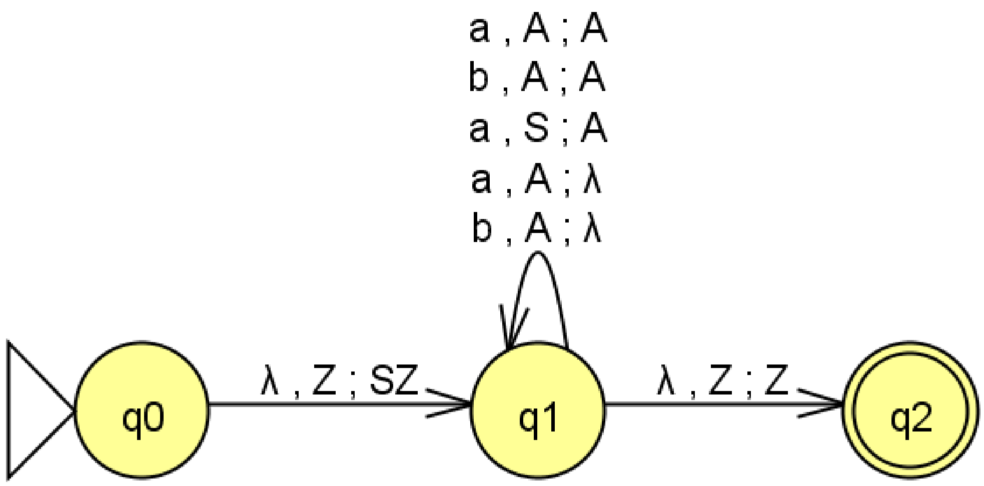
20. $S \to aA \mid bB \mid cC, A \to Sa, B \to Sb, C \to \lambda$

Solution: First we need to convert the given grammar to GNF:
$S \to aA \mid bB \mid c, A \to aAD \mid bBD \mid cD, B \to aAE \mid bBE \mid cE, D \to a, E \to b$



21. $S \to aA, A \to aA \mid bA \mid a \mid b$

Solution: The grammar is already in GNF. Hence the PDA is:



C. Convert each of the following PDAs to an equivalent context-free grammar.

22. Consider the PDA shown in Figure-8.10. What is the language of the PDA (and the resulting grammar)?

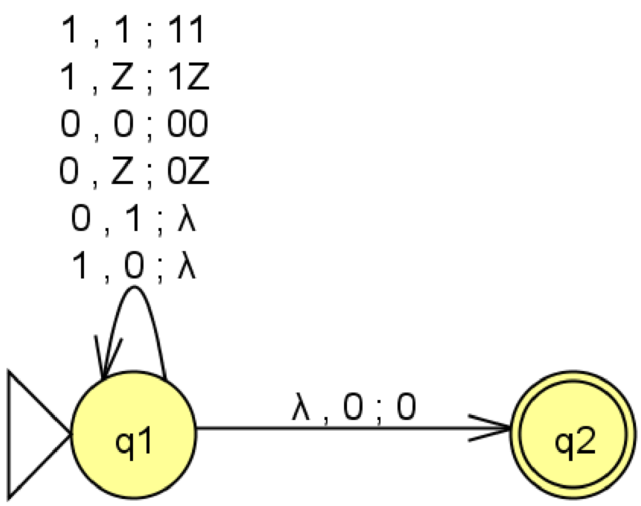
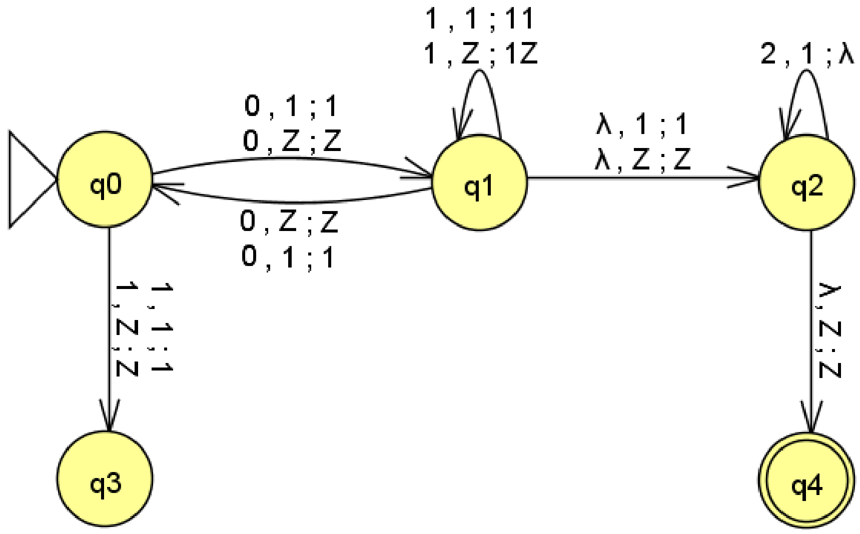


Figure 8.10

Solution: The language is the set of all binary strings containing more 0 s than 1 s. The equivalent grammar is (using the variable *A* for the stack symbol 0 and *B* for the stack symbol 1):
$S \to 0A \mid 1B$, $A \to 0AA \mid 1 \mid \lambda$, $B \to 1BB \mid 0$

23. Consider the PDA shown in Figure. 8.11. What is its language?



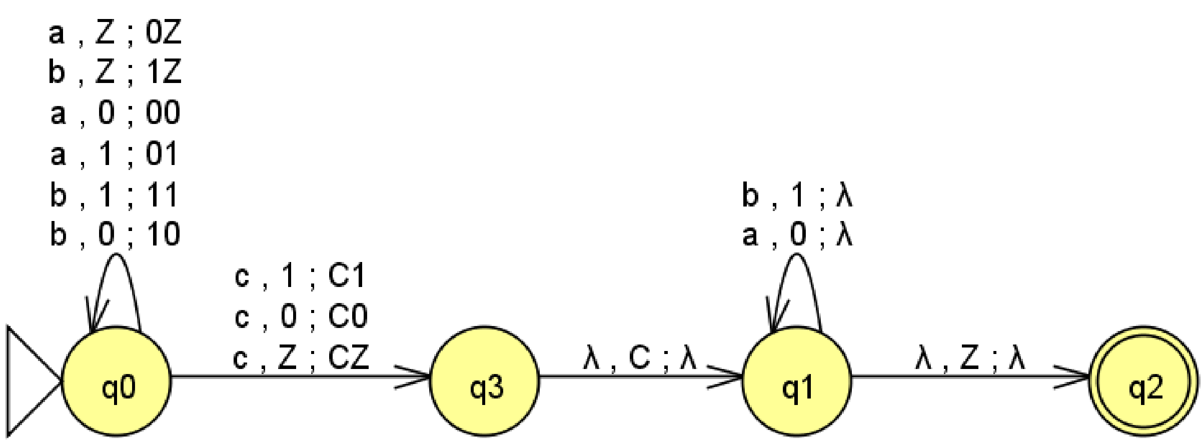
Figuer 8.11

Solution: Binary strings beginning with an odd number of 0 s and having 1 s mixed with even numbers of 0 s (i.e., occurrence of 0 except at the beginning must be a run of even length), followed by as many 2 s as there were 1 s.

An equivalent CFG is:
$S \to 0T$, $T \to 0S \mid 1TA \mid \lambda$, $A \to 1AA \mid B$, $B \to 2$

24. Consider the PDA shown in Figure-8.9 (Example 8.6). Do you get back the same grammar as in the example?

Solution: First, we need to modify the PDA slightly to enable automatic conversion to a CFG in JFLAP:



Its language is that of odd palindromes. The equivalent grammar obtained from this (after simplification) is:

LHS		RHS
S	$\rightarrow$	c
S	$\rightarrow$	bD
S	$\rightarrow$	aJ
J	$\rightarrow$	bDK
K	$\rightarrow$	a
E	$\rightarrow$	b
D	$\rightarrow$	bDE
D	$\rightarrow$	aJE
J	$\rightarrow$	aJK
D	$\rightarrow$	cE
J	$\rightarrow$	cK

A much simpler grammar that we can construct for this language is:

$S \rightarrow aSa \mid bSb \mid c$

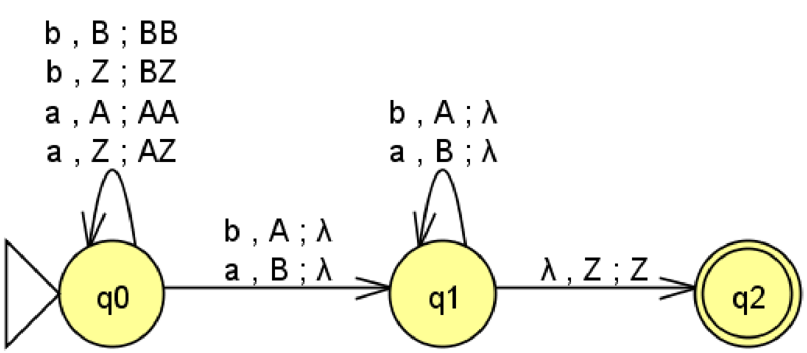
Even upon conversion to GNF, this grammar is much simpler than what we obtained from the PDA:

$S \rightarrow aSA \mid bSB \mid c, A \rightarrow a, B \rightarrow b$

25. Consider the PDA specified by the transitions shown below where q_2 is the accepting state:

$\delta (q_0, a, Z) = (q_0, AZ)$
$\delta (q_0, a, A) = (q_0, AA)$
$\delta (q_0, b, Z) = (q_0, BZ)$
$\delta (q_0, b, B) = (q_0, BB)$
$\delta (q_0, a, B) = (q_1, \lambda)$
$\delta (q_0, b, A) = (q_1, \lambda)$
$\delta (q_1, a, B) = (q_1, \lambda)$
$\delta (q_1, b, A) = (q_1, \lambda)$
$\delta (q_1, \lambda, Z) = (q_2, Z)$

Solution: The given PDA is:



The language is either $a^n b^n$ or $b^n a^n$, $n > 0$. An equivalent CFG is:
$S \rightarrow aA \mid bB$, $A \rightarrow aAA \mid b$, $B \rightarrow bBB \mid a$

D. Debug and fix the following PDAs:

26. For $a^n b^{2n}$, $n > 0$ (Figure-8.12):

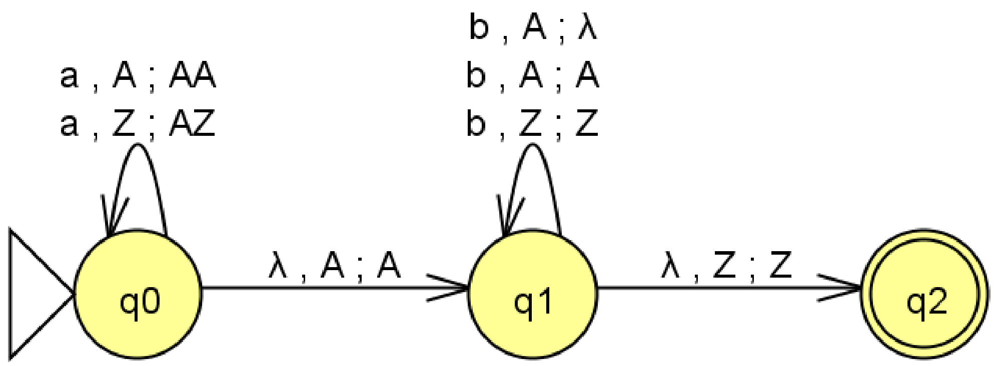
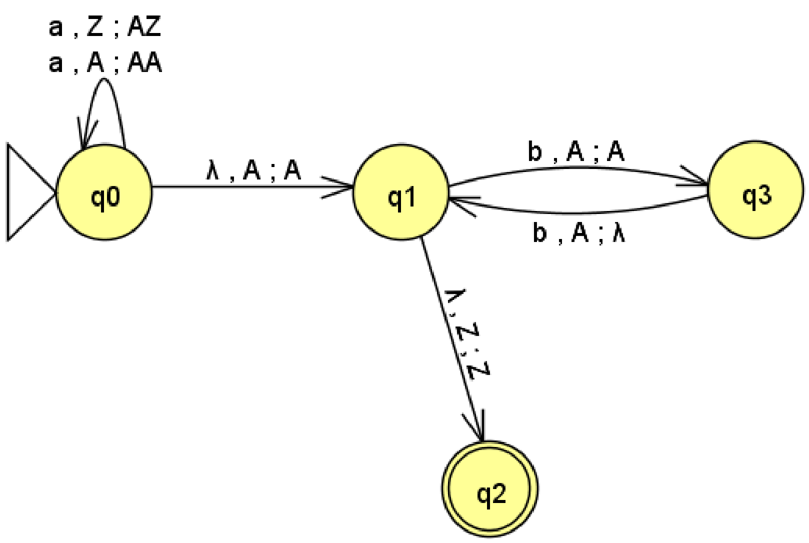


Figure 8.12

Solution: The corrected PDA is:



27. For simplified XML tag strings as used in Exercise 71 of Chapter 7, for example, *<a><a><a><c></c>*
where *a*, *b* and *c* are the only tag names (Figure-8.13):

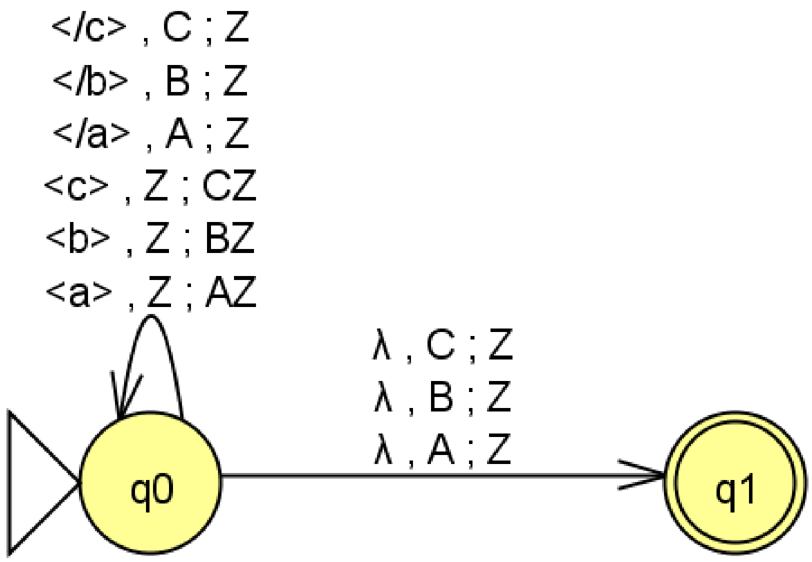
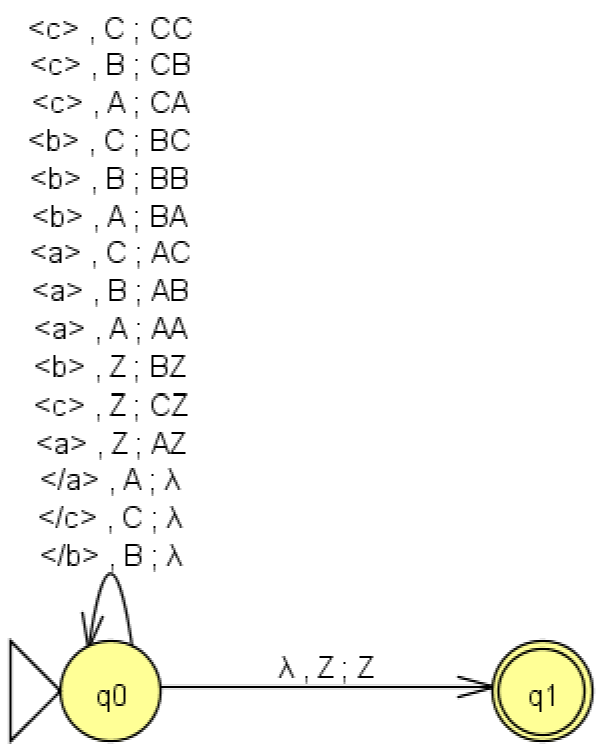


Figure 8.13

Solution: The corrected PDA is:

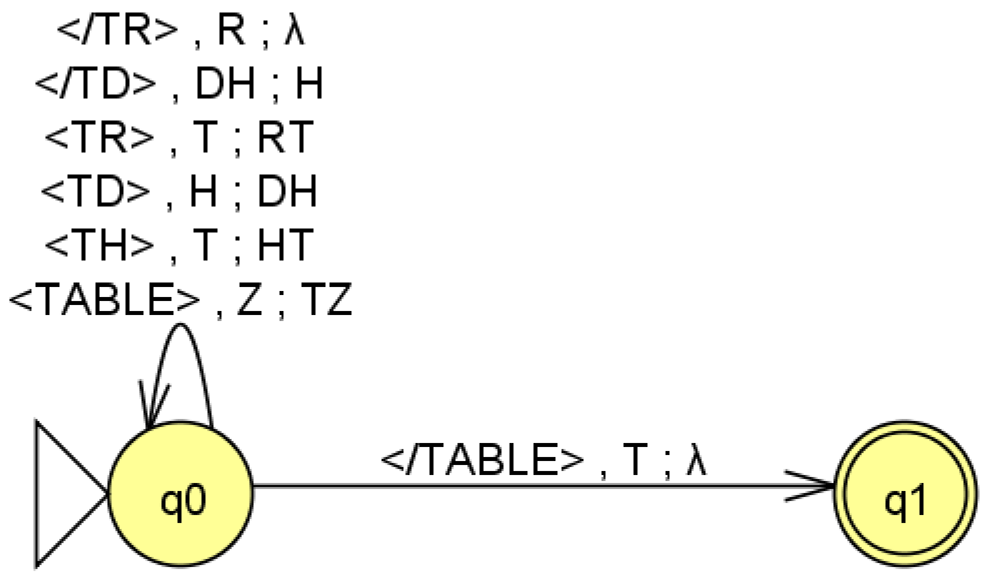


It may also be noted that it may be desirable to split the multi-character inputs such as <*a*> into separate transitions for each symbol with intermediate states.

28. For the language of table tags in HTML {*TABLE*, *TH*, *TR*, *TD*} (as specified in Exercise 69 of Chapter 7), the PDA specified by the transitions shown below where q_1 is the accepting state:
δ (q_0, <*TABLE*>, *Z*) = (q_0, *TZ*)
δ (q_0, <*TH*>, T) = (q_0, *HT*)
δ (q_0, <*TD*>, H) = (q_0, *DH*)

δ (q_0, </*TD*>, DH) = (q_0, *H*)
δ (q_0, <*TR*>, T) = (q_0, *RT*)
δ (q_0, </*TR*>, *R*) = (q_0, λ)
δ (q_0, </*TABLE*>, *T*) = (q_1, λ)

Solution: The given PDA is:



The corrected PDA is (note: tables can also be nested):