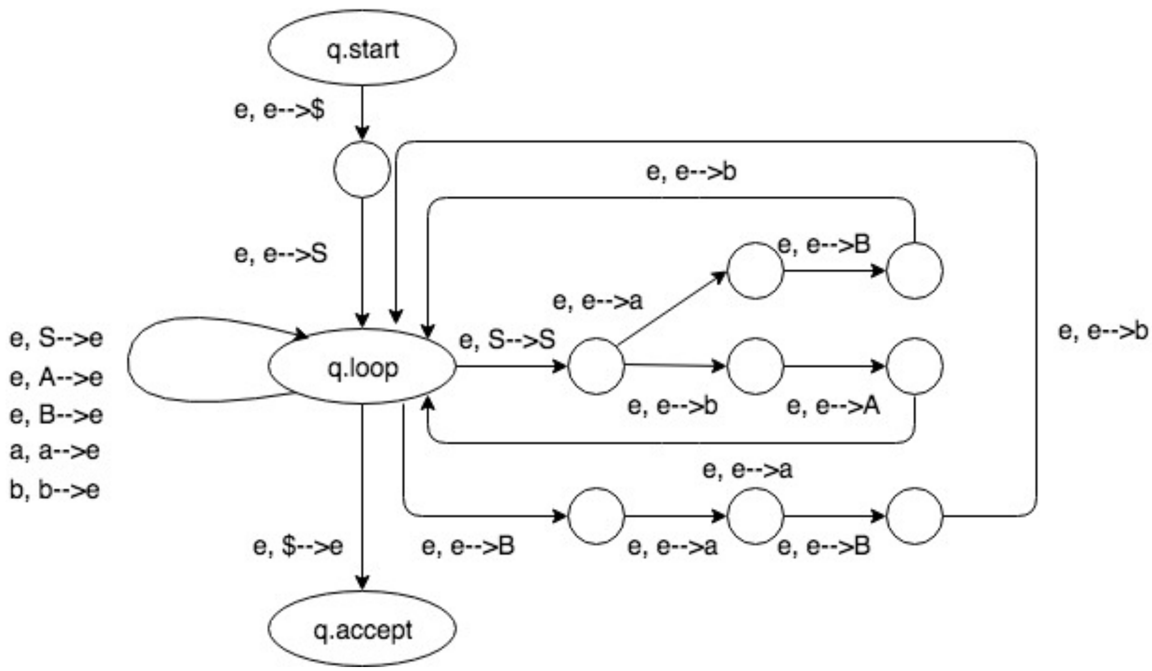
Solutions of Assignment #3 --- CSC320, Summer, 2018

Zhaocheng Li

V00832770

Q1-Answer:

Converting CFG to PDA by diagram:



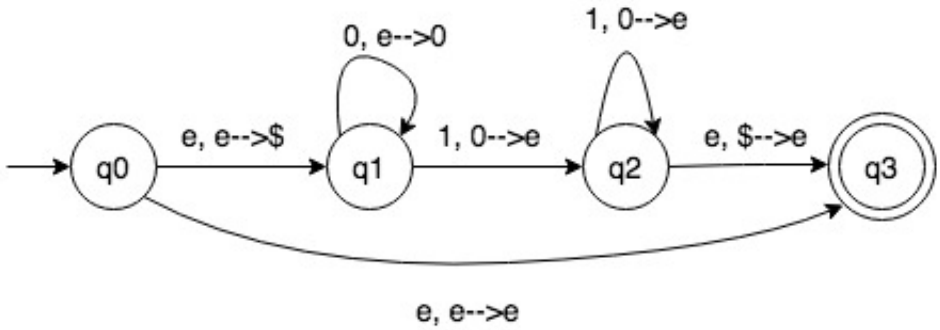
(here e stands for ε)

Q2-Answer:

Converting PDA to CFG using the standard procedure.

1. Preprocessing the PDA diagram:

    - Single accept state? No. we convert it into:



- Empty stack before accepting? Check.
- Pop/push each transition but not both? Check.

2. CFG construction.

   Construct a CFG G = (V, Σ, R, S), where

   - The start variable is S = A_{q0q3};
   - Σ = {a, b}
   - V = {A_{q0q3}, A_{q0q1}, A_{q0q2}, A_{q1q2}, A_{q1q3}, A_{q2q3}}
   - And mostly importantly, the R is (by the construction rule): (note, the variables like A_{12} and A_{21} are identical)

$A_{q0q3} \rightarrow A_{q1q2}$

$A_{q0q3} \rightarrow A_{q0q1}\ A_{q1q3}$, $A_{q0q3} \rightarrow A_{q0q2}\ A_{q2q3}$

$A_{q0q1} \rightarrow A_{q0q2}\ A_{q2q1}$, $A_{q0q1} \rightarrow A_{q0q3}\ A_{q3q1}$

$A_{q0q2} \rightarrow A_{q0q1}\ A_{q1q2}$, $A_{q0q2} \rightarrow A_{q0q3}\ A_{q3q2}$

$A_{q1q2} \rightarrow A_{q1q0}\ A_{q0q2}$, $A_{q1q2} \rightarrow A_{q1q3}\ A_{q3q2}$

$A_{q1q3} \rightarrow A_{q1q0}\ A_{q0q3}$, $A_{q1q3} \rightarrow A_{q1q2}\ A_{q2q3}$

$A_{q2q3} \rightarrow A_{q2q0}\ A_{q0q3}$, $A_{q2q3} \rightarrow A_{q2q1}\ A_{q1q3}$

$A_{q0q0} \rightarrow \varepsilon$, $A_{q1q1} \rightarrow \varepsilon$, $A_{q2q2} \rightarrow \varepsilon$

$A_{q3q3} \rightarrow \varepsilon$

Q3-Answer:

First of all, the class of languages this model recognize is regular languages only.

Since we can simulate any DFA on a Turing Machine with stay put instead of left,

The only non-trivial modification is to add transitions from state in *F (DFA)* to q_{accept} when reading a blank, and from states not in F to q_{reject} when reading a blank.

Suppose the Turing machine M = $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ with stay put instead of left.

Then we can **prove by constructing a DFA** $(Q', \Sigma', \delta', q_0', F)$ that recognizes the same language as M does.

Given the fact that M cannot move to the right and cannot read anything it has written on the tape as soon as it moves right. Hence it is actually a one-way access to the input, similar with a DFA.

Modification as follow:

- Add a new symbol so that M never writes blanks on the tape; M writes new symbol when it is going to write blanks

- The reading head moves to the right and never stays put when M transitions into q_{accept} or q_{reject};
- Set $Q' = Q$, $\Sigma' = \Sigma$, $q_0' = q_0$, and set the transition function δ' as follow: ($q \in Q$, and $a \in \Sigma$)
  - $\delta'(q, a) = q$ if $q \in \{q_{accept}, q_{reject}\}$
  - $\delta'(q, a) = q_{reject}$ if M starting at q and reading a keeps staying put, or
  - $\delta'(q, a) = q'$ where q' is the state that M enters, when it first moves right when starting at q and reading a.

Observing that with such construction, we make F the set containing q_{accept} and all states $q \in Q$, $q \neq q_{accept}, q_{reject}$, such that M, starting at q and reading blanks, would enters q_{accept} in the end.

Q4-Answer: (prove by construction)

(a): suppose that L1 and L2 are two decidable languages and

M1 and M2 be to deciders (Turing machines halting) for L1 and L2 respectively. Then there is a decider M for L1L2 where:

Given the input w, M non-deterministically partitions w=w1w2; M calls M1 to run on w1 and calls M2 to run on w2; M accepts w if and only if M1 accepts w1 and M2 accepts w2.

Since M1 and M2 halt, then so does M.

Prove done.

(b): Suppose L1 and L2 are two decidable languages and M1 and M2 be two deciders for L1 and L2 respectively (same as above). Then there is a decider M for L1$\cap$L2 where:

Given the input w, M calls M1 to run on w, and calls M2 to run on w. Then M accepts w if and only if both M1 accepts w and M2 accepts w.

Since M1 and M2 halt, then so does M.

Prove done.

(c): Similarly, suppose L1 is a decidable language and M1 is a decider for M1. Then there is a decider M for L_{hat} (complement of L) where:

Given the input w; M calls M1 to run on w; M accepts w if and only if M1 rejects w.

Since M1 halts, then so does M.

Prove done.