- Start as early as possible, and contact the instructor if you get stuck.
- See the course outline for details about the course's marking policy and rules on collaboration.
- Submit your completed solutions to **Crowdmark**.

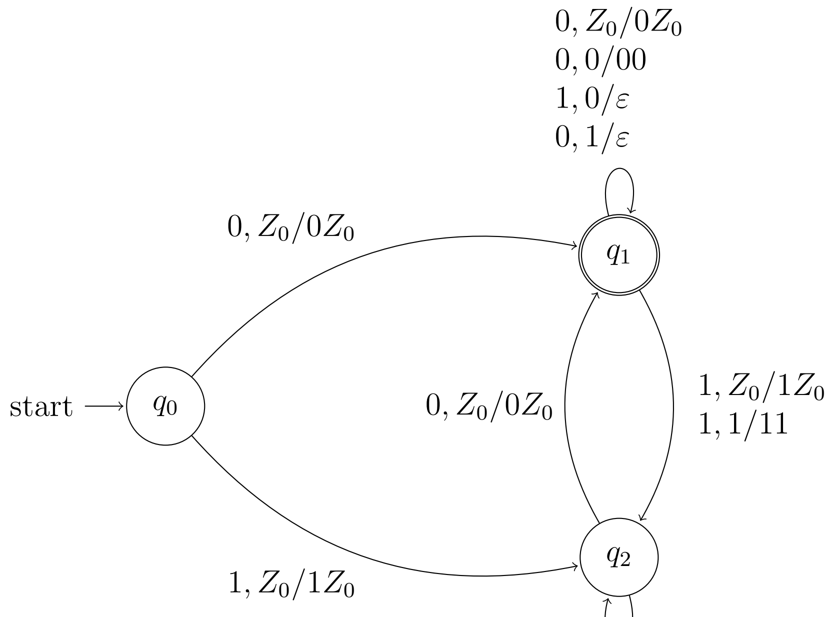
1. A pushdown automaton

[6] Let $\Sigma = \{0, 1\}$. Construct a pushdown automaton, P, which accepts exactly the language

$$L = \{w \in \Sigma^* \mid \text{every prefix of } w \text{ has at least as many 0s as 1s}\}.$$

State explicitly whether P accepts by final state or by empty stack. Argue informally why your P accepts exactly L.

- Let P be a PDA **accepting by final state** defined by the following diagram:



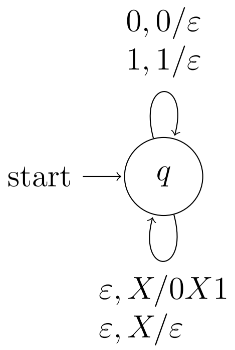
- I claim P accepts exactly the language defined by L.
- To argue this, we will transition between states $Q = \{q_0, q_1, q_2, q_3\}$ and show that only when the number of 0 symbols read is at least equal to the number of 1s read will P accept.
- State q_0:
  - In q_0 we are in our initial state where the stack contains only the empty stack symbol Z_0.
  - Reading a 0 will move us to q_1 and the 0 will be pushed on to the stack.

  - Reading a 1 will move us to q_2 and the 1 will be pushed on to the stack.
  - As there are no arrows pointing to q_0 and thus no way to return to this state after we have left we have considered all possibilities for our initial state and can move on.
- State q_1:
  - I claim in state q_1 there will always have been at least as many 0s read as 1s and therefore q_1 is an accepting state.
  - We note any transition to this state will come when a 0 is read and the stack is empty.
  - This means that when we arrive in this state there will have always been one more 0 symbols read than 1 symbols.
  - While in this state any 0s read will be added to the stack.
  - When a 1 is read, we pop a 0 from the stack if the stack contains a 0 as its top element.
  - If a 1 is read and the empty stack symbol or another 1 is at the top of the stack we can no longer garuntee there are at least as many 0s as 1s so we move to state q_2
  - In all other cases we know we have read at least as many 0s as 1s and can accept.
- State q_2:
  - I claim in state q_2 there will always have been at least as many 1s read as 0s which includes the case where there have been more 1s read than 0s so q_2 is not an accepting state.
  - We note any transition to this state will come when a 1 is read and the stack is empty or contains a 1 as its top symbol.
  - This means there will be have been more 1s read than 0s.
  - From this state any 1s read will be placed on the stack and any 0s read will pop a 1 from the stack and be discarded.
  - We will stay in q_2 as long as there have been a greater number of 1s read than 0s.
- We have shown that in state q_1 there will always be more or equal 0 symbols read compared to 1s.
- Furthermore, in state q_2 the converse is true where at any time in state q_2 there have been more 1s read than 0s.
- Since q_1 is the only accepting state and P accepts by final state we conclude that P accepts the language exactly defined by L.

2. Building a context-free grammar from a PDA
   Define the pushdown automaton, $P = (Q, T, \Gamma, \delta, q, X)$ (which accepts by empty stack), with
   - $Q = \{q\}$
   - $T = \{0, 1\}$
   - $\Gamma = \{0, 1, X\}$
   - q = start state for machine
   - X = stack start letter

   and transition function
   1. $\delta(q, \varepsilon, X) = \{(q, \varepsilon), (q, 0X1)\}$
   2. $\delta(q, 0, 0) = \{(q, \varepsilon)\}$
   3. $\delta(q, 1, 1) = \{(q, \varepsilon)\}$

[2] (a) Draw a diagram for P.



[4] (b) Use the technique described on slides 51-54 of Module 6 to construct a context-free grammar, G, such that $L(G) = N(P)$. In your final grammar, replace the non-terminals from the construction with single capital letters $A, B, C, \ldots$. Simplify your grammar as much as possible after you have completed the construction. You do **not** have to prove that $L(G) = N(P)$. But you should convince yourself that the equality holds once you have completed the construction and simplification of G.

- We start by noting that P only contains one state and the alphabet Γ has exactly three symbols.
- From this we know we must create $1 * 3$ non-terminals in the form $[qXp]$.
- Our non-terminals are:
  - $[qXq]$ for the state q and stack symbol X.
  - $[q0q]$ for the state q and stack symbol 0.
  - $[q1q]$ for the state q and stack symbol 1.
- For each instance of the transition function from P we create our productions in G:
  1. $\delta(q, \varepsilon, X) = \{(q, \varepsilon), (q, 0X1)\}$
     - $\delta(q, \varepsilon, X) = (q, \varepsilon)$ Gives us the production $[qXq] \to \varepsilon$
     - $\delta(q, \varepsilon, X) = (q, 0X1)$ Gives us the production $[qXq] \to [q0q][qXq][q1q]$
  2. $\delta(q, 0, 0) = \{(q, \varepsilon)\}$ Gives us the production $[q0q] \to 0$
  3. $\delta(q, 1, 1) = \{(q, \varepsilon)\}$ Gives us the production $[q1q] \to 1$
- Putting this all together we have the productions in G as

$$\begin{aligned}
&[qXq] \to \varepsilon \mid [q0q][qXq][q1q] \\
&[q0q] \to 0 \\
&[q1q] \to 1
\end{aligned}$$

- After changing the variable names to simplify we are left with,

$$\begin{aligned}
&A \to \varepsilon \mid BAC \\
&B \to 0 \\
&C \to 1
\end{aligned}$$

3. The analog of Kleene's Theorem for CFLs and PDAs

[8] Let Σ be an alphabet. For any language L over Σ, Define

$$SUFFIX(L) = \{y \mid xy \in L, \text{ for some string } x \in \Sigma^*\}.$$

Prove that the class of context-free languages over Σ is closed under the $SUFFIX$ operation.

Proof. We define a CFG $G = (V, T, P, S)$ which accepts the language L such that $L(G) = L$

- We will construct a new CFG $G' = (V', T', P', S)$ which accepts the language defined by $SUFFIX(L)$ such that $G' = L(SUFFIX(L))$
  - We define $V' = V \cup X$ where $X \notin V$. We will use X to generate our suffixes in our new language. We keep our start symbol as S.
  - For each production in G we add a production in G' with the following rules:
    * Let $A \to \alpha$ be an arbitrary production in G such that $A \in V$ and $\alpha \in \{V \cup \Sigma\}$
    * If α is of the form βB where $\beta \in (V \cup \Sigma)^*$ and $B \in V$: add the rule $B \to X$
    * If α is of the form βa where $\beta \in (V \cup \Sigma)^*$ and $a \in \Sigma$: add the rule $X \to aX$
  - We add a final production $X \to \varepsilon$ for the case when the suffix is the empty string.
- To prove the language generated by G' is exactly the language defined by $SUFFIX(L)$ we consider an arbitrary string $y \in SUFFIX(L)$.
  - For any $y \in SUFFIX(L)$ there exists a string $xy \in L$
  - We know G' can generate x using the original productions in G letting the new variable X be the empty string.
  - When $X \neq \varepsilon$ we can generate the suffix y using the modified rules in G' which we defined above.
  - Therefore G' accepts xy and $xy \in L$ where y is a suffix of xy
- Since G' generates all the suffixes of string in L, the language generated by G' is exactly $SDUFFIX(L)$
- Since G and G' are arbitrary we have shown that the class of context-free langauges isd closed under the $SUFFFIX$ operation.

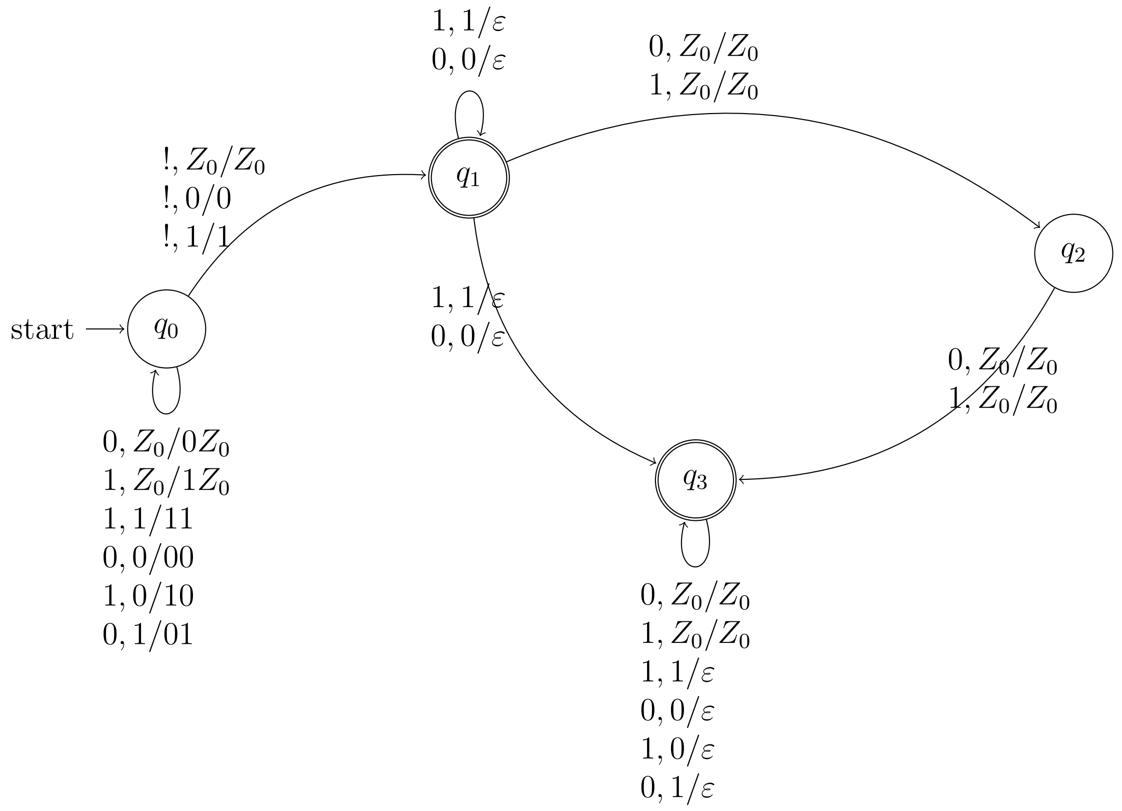
□

4. Deterministic PDAs and DCFLs

[6] Prove that the language

$$L = \left\{ x!y \mid x \neq y^R \text{ and } x, y \in \{0,1\}^* \right\}$$

is a DCFL, by giving a natural DPDA for it and explaining how that DPDA works. Your machine should accept by final state.



- We let $w \in L$ be arbitrarry then write $w = x!y$ for some $x, y \in \Sigma*$
- Starting at q_0, any symbol read is placed on the stack until a '!' is read. The '!' is discarded and our stack now contains the string x^R and we move to q_1
- In q_1 we continue to read symbols while they match the top of our stack.
- We stay in q_1 as long as the string read exactly matches our stack symbols. Since our stack symbols are read in the reverse order from which they were pushed, the string we are reading in q_1 will partially match the reverse of whatever we have pushed on the stack.
- As our stack contains x^R and we are building y in q_1 we will continue in this way until the empty stack symbol is reached.
- If at any time a symbol is read that does not match the top of our stack we know that $y \neq x^R$ and move to q_3 where we will accept.
- Similarly, if the input ends before our stack empties we know $y \neq x^R$ and we can accept in q_1
- Only when we are in q_1 and the empty stack symbol is seen do we have the case where $y = x^R$ and we must move to q_2

- In state q_2 we know from the above that $y = x^R$ and by the predicate defining L we cannot accept.
- If any symbol is read in q_2 we immediately move to q_3.
- In state q_3 we know that $y \neq x^R$ and never will. Thus we remain in this state on any symbol read and accept any time we run out of input.

5. Recall from Slide 18 of Module 5, a grammar for words having a balanced number of 0s and 1s:

$$G : S \to \varepsilon \mid 0S1 \mid 1S0 \mid SS.$$

[8] (a) Use the technique on slides 6-17 of Module 7 to produce a grammar, G', in Chomsky Normal Form, such that $L(G) = L(G') \cup \{\varepsilon\}$. You do **not** need to prove this equality of languages: following the algorithm correctly will guarantee this fact.

1. Nullable variables S is nullable. Hence to construct G' from G we
   A. add $S \to 01$ and $S \to 10$ then
   B. remove $S \to \varepsilon$

$$G_1 : S \to 01 \mid 10 \mid 0S1 \mid 1S0 \mid SS.$$

2. Unit pairs No changes since there are no unit pairs in G

$$G_1 = G_2$$

3. Long productions $1S0$ and $0S1$ are long productions thus,
   - For $S \to 1S0$ we,
     A. add productions
        - $S \to 1C$
        - $C \to S0$
     B. then remove the production $S \to 1S0$
   - For $S \to 0S1$ we,
     A. add productions
        - $S \to 0D$
        - $D \to S1$
     B. then remove the production $S \to 0S1$

$$\begin{aligned} G_3 : \; & S \to 01 \mid 10 \mid 1C \mid 0D \mid SS \\ & C \to S0 \\ & D \to S1 \end{aligned}$$

4. Terminals in two-letter productions We have the terminals 0 and 1 which show up in multiple two-letter productions.
   A. add the productions $A \to 0$ and $B \to 1$ to define each terminal.
   B. add the productions
      - $S \to AB$, $S \to BA$, $S \to BC$, $S \to AD$
      - $C \to SB$
      - $D \to SA$
   C. remove the productions
      - $S \to 01$, $S \to 10$, $S \to 1C$, $S \to 0D$
      - $C \to S1$
      - $D \to S0$

$$\begin{aligned} G_4 :\ & S \to AB \mid BA \mid BC \mid AD \mid SS \\ & A \to 0 \\ & B \to 1 \\ & C \to SA \\ & D \to SB \end{aligned}$$

We let $G' = G_4$ and as we have followed the algorithm to completion it is now clear $L(G') = L(G)$

[4] (b) Give an explicit derivation, in G', for the word: 0101.

Let $w = 0101$ Then $S \underset{G'}{\overset{*}{\Rightarrow}} w$

$$S \underset{G'}{\Rightarrow} SS \underset{G'}{\Rightarrow} ABS \underset{G'}{\Rightarrow} 0BS \underset{G'}{\Rightarrow} 01S \underset{G'}{\Rightarrow} 01AB \underset{G'}{\Rightarrow} 010B \underset{G'}{\Rightarrow} 0101$$