Question 1.
We consider the expression tree for a boolean expression with operators \( \lor \) (or), \( \land \) (and) and \( \neg \) (not). An example is given below:

An evaluation of the expression gives the value 1. A preorder traversal of the tree returns the string \( \lor \land \neg \lor \land \). Let \( L \) be the set of strings returned by the preorder traversals of all (boolean) expression trees that are evaluated to 1.

(a) (25%) Show that \( L \) is not regular.

(b) (25%) Give a context-free grammar for \( L \).

(c) (25%) Give a deterministic pushdown automaton for \( L \). Besides giving a formal presentation of a deterministic PDA, you are encouraged (though not required) to provide an informal discussion of the design of the automaton.

Question 2. (25%)
Let \( \Sigma = \{0, 1, +, =\} \) and
\[ ADD = \{x = y + z \mid x, y, z \text{ are binary integers, and } x \text{ is the sum of } y \text{ and } z\}. \]
For example, \( 10100 = 1001 + 1011 \in ADD \) and \( 110 = 10 + 1 \notin ADD \).
Show that \( ADD \) is not context-free. (Hint: consider strings \( x = 0 + x \in ADD \).)
Answers:

1. (a)
Suppose the contrary that \( L \) is regular and there exists a DFA \( M = (Q, \Sigma, \delta, q_0, F) \) for \( L \). Consider the strings \( w_k = \lambda^k \), for \( k \geq 1 \). We claim for \( 1 \leq i < j \), \( \delta(q_0, w_i) \neq \delta(q_0, w_j) \). If \( \delta(q_0, w_i) = \delta(q_0, w_j) \), then \( \delta(q_0, w_i \lambda^{i+1}) = \delta(q_0, w_i, 1^{i+1}) = \delta\left(\delta(q_0, w_i), 1^{i+1}\right) = \delta(q_0, \lambda^i 1^{i+1}) = \delta(q_0, \lambda^i \lambda^{i+1}) \in F \) which contradicts with the fact that \( \lambda^i \lambda^{i+1} \notin L \) when \( i \neq j \). Thus, \( M \) has infinite states, a contradiction.

1. (b)
\[
\begin{align*}
S_1 & \rightarrow \land S_1 S_1 | \lor S_0 S_1 | \lor S_1 S_0 | \lor S_1 S_1 | \neg S_0 | 1 \\
S_0 & \rightarrow \lor S_0 S_0 | \land S_0 S_0 | \land S_0 S_1 | \land S_1 S_0 | \neg S_1 | 0
\end{align*}
\]

1. (c)
Let \( q \) be the starting state and \( q_a \) be the accepting state. The transitions are:
- if (state \( q \), empty stack, input 1) then (state = \( q_1 \))
- if (state \( q \), input \( x \) is an operator) then (push \( x \))
- if (state \( q \), stack top = operator, input 0) then (state = \( q_0 \))
- if (state \( q \), stack top = operator, input 1) then (state = \( q_1 \))
- if (state \( q_0 \), stack top = \( \neg \)) then (state = \( q_1 \), pop stack)
- if (state \( q_1 \), stack top = \( \neg \)) then (state = \( q_0 \), pop stack)
- if (state \( q_0 \), stack top = 0) then (state = \( q_{0,0} \), pop stack)
- if (state \( q_0 \), stack top = 1) then (state = \( q_{0,1} \), pop stack)
- if (state \( q_1 \), stack top = 0) then (state = \( q_{0,1} \), pop stack)
- if (state \( q_1 \), stack top = 1) then (state = \( q_{1,0} \), pop stack)
- if (state \( q_0 \), stack top = operator, input \( x \) is an operator) then (state = \( q \), push 0, push \( x \))
- if (state \( q_1 \), stack top = operator, input \( x \) is an operator) then (state = \( q \), push 1, push \( x \))
- if (state \( q_{0,0} \), top stack = \( \land \) or \( \lor \)) then (state = \( q_0 \), pop stack)
- if (state \( q_{0,1} \), top stack = \( \land \)) then (state = \( q_0 \), pop stack)
- if (state \( q_{0,1} \), top stack = \( \lor \)) then (state = \( q_1 \), pop stack)
- if (state \( q_{1,1} \), top stack = \( \land \) or \( \lor \)) then (state = \( q_1 \), pop stack)
- if (state \( q_{1,1} \), stack empty, end-of-input) then (state = \( q_a \))

2. (The solution is similar to that of Example 2.22 in Sipser’s book.) (Sketch)
Choose \( s \) to be \( 1^p 0^p = 0 + 1^p 0^p \) where \( p \) is the pumping constant. Note that we assume that \( x, y \) and \( z \) cannot be empty strings. That is, empty string is not a binary integer. When pumping, always pump down by taking \( i = 0 \). One case of the pumping is when \( u = 0^p 1^p =, v = 0, x = \epsilon, y = \epsilon \) and \( z = +1^p 0^p \). Another case of the pumping is when \( u = 0^p 1^p =, v = \epsilon, x = \epsilon, y = 0 \) and \( z = +1^p 0^p \). Of course, there are still many other cases.