1. Describe context-free grammars for the following languages over the alphabet $\Sigma = \{0, 1\}$. For each non-terminal in your grammars, describe in English the language generated by that non-terminal.

   (a) $\{0^a 1 0^b 1 0^c \mid a + b = c\}$

Solution: Any string $w = 0^a 1 0^b 1 0^{a+b} \in L$ can be decomposed into substrings as $w = 0^a 1 \bullet (0^b 1 0^b) \bullet 0^a$. The non-terminal B generates the parenthesized substring $0^b 1 0^b$; the starting non-terminal then generates the outer 0s.

$$S \rightarrow 0S0 \mid 1B$$
$$B \rightarrow 0B0 \mid 1$$

■

Rubric: 2 points = 1 for grammar + 1 for justification.

(b) $\left\{w \in (0+1)^* \;\middle|\; \#(0,w) \le 2 \cdot \#(1,w)\right\}$

Solution (dropping 0s): Let $L^{\le}$ be the target language. We modify the grammar for the language $L^{=} := \{w \in (0+1)^* \mid \#(0,w) = 2\cdot\#(1,w)\}$ described in the lab solutions:

$$S \to \varepsilon \mid SS \mid 00S1 \mid 1S00 \mid 0S1S0$$

Dropping any number of 0s from any string in $L^{=}$ leaves a string in $L^{\le}$; moreover, *every* string in $L^{\le}$ can be obtained from some string in $L^{=}$ by dropping 0s.

Thus, we can transform the grammar for $L^{=}$ into a grammar for $L^{\le}$ by dropping 0s from the right side of every production in all possible ways:

$$S \to \varepsilon \mid SS \mid \underbrace{00S1 \mid 0S1 \mid S1}_{00S1} \mid \underbrace{1S00 \mid 1S0 \mid 1S}_{1S00} \mid \underbrace{0S1S0 \mid S1S0 \mid 0S1S \mid S1S}_{0S1S0}$$

The last three productions in this grammar are redundant, so we can remove them. For example, we don't need the production $S \to S1S$, because other productions give us the derivation $S \rightsquigarrow SS \rightsquigarrow 1S$.

$$S \to \varepsilon \mid SS \mid 00S1 \mid 0S1 \mid S1 \mid 1S00 \mid 1S0 \mid 1S \mid 0S1S0$$

If we add the trivial production $S \to 1$ (replacing the derivation $S \rightsquigarrow 1S \rightsquigarrow 1$), we can remove two more redundant productions.

$$S \to \varepsilon \mid 1 \mid SS \mid 00S1 \mid 0S1 \mid 1S00 \mid 1S0 \mid 0S1S0$$

■

Solution (0, −1, or less): For any string w, let $\Delta(w) = \#(0,w) - 2\cdot\#(1,w)$.

$$\begin{array}{ll} L \to \varepsilon \mid ZL \mid ML \mid 1L & \Delta \le 0 \\ Z \to \varepsilon \mid ZZ \mid 00Z1 \mid 1Z00 \mid 0Z1Z0 & \Delta = 0 \\ M \to ZM \mid MZ \mid 0Z1 \mid 1Z0 & \Delta = -1 \end{array}$$

The non-terminals in this grammar respectively generate the following languages.

$$\begin{array}{ll} L = \left\{w \in (0+1)^* \;\middle|\; \Delta(w) \le 0\right\} & \text{“Less or equal”} \\ Z = \left\{w \in (0+1)^* \;\middle|\; \Delta(w) = 0\right\} & \text{“Zero”} \\ M = \left\{w \in (0+1)^* \;\middle|\; \Delta(w) = -1\right\} & \text{“Minus one”} \end{array}$$

Our target language is L. A grammar for Z appears in the Lab 4½ solutions. For any string w, we have $\Delta(w) = -1$ (that is, $w \in M$) if and only if at least one of the following conditions holds:

- w has a non-empty proper prefix in Z, and the rest of w is in M.
- w has a non-empty proper prefix in M, and the rest of w is in Z.
- $\Delta(x) > 0$ for *every* non-empty proper prefix x of w. Then $w = 0z1$ for some string $z \in Z$

- $\Delta(x) < -1$ for *every* non-empty proper prefix x of w. Then $w = \texttt{1}z\texttt{0}$ for some string $z \in Z$.

Finally, $\Delta(w) \leq 0$ if and only if at least one of the following conditions holds:

- w is empty.
- w has a non-empty prefix in Z, and the rest of w is in L. (This includes the case $w \in Z$.)
- w has a non-empty prefix in M, and the rest of w is in L. (This includes the case $w \in M$.)
- $\Delta(x) \leq -2$ for *every* non-empty prefix x of w. (In particular, $\Delta(w) \leq -2$.) Then $w = \texttt{1}y$ for some string y with $\Delta(y) \leq 0$.

■

Rubric: 4 points = 2 for grammar + 2 for justification. These solutions have more detail than necessary for full credit.

(c) Strings in which the substrings 00 and 11 appear the same number of times. For example, $\mathtt{1100011} \in L$ because both substrings appear once, but $\mathtt{01000011} \notin L$. *[Hint: This is the complement of the language you considered in HW2.]*

Solution (counting): Let $\#(\mathtt{11}, w)$ denote the number of times 11 appears as a substring of w, and let $\#(\mathtt{1}^*, w)$ denote the number of runs of 1s in w. For example:

$$\#(\mathtt{11}, \mathtt{1111001110 1}) = 5 \qquad \#(\mathtt{1}^*, \mathtt{11110011101}) = 3$$

Our grammar is based on the observation that each 1 in a binary string is the start of a 11 substring, except for the last 1 in every run; thus, symbolically, we have

$$\#(\mathtt{11}, w) = \#(\mathtt{1}, w) - \#(\mathtt{1}^*, w).$$

Symmetrically, $\#(\mathtt{00}, w) = \#(\mathtt{0}, w) - \#(\mathtt{0}^*, w)$. But because runs of 0s and 1s in any binary string w alternate, $\#(\mathtt{0}^*, w)$ and $\#(\mathtt{1}^*, w)$ always differ by at most 1.

Further case analysis implies that L contains a binary string w if and only if one of the following conditions holds:

- $w = \varepsilon$
- w starts with 0 and ends with 1, and $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$
- w starts with 1 and ends with 0, and $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$
- w starts with 0 and ends with 0, and $\#(\mathtt{0}, w) = \#(\mathtt{1}, w) + 1$. (In this case, dropping the final 0 leaves a string with equal 0s and 1s.)
- w starts with 1 and ends with 1, and $\#(\mathtt{0}, w) = \#(\mathtt{1}, w) - 1$. (In this case, dropping the final 1 leaves a string with equal 0s and 1s.)

In the following grammar, each nonterminal ${}_aE_z$ generates strings that start with symbol a, end with symbol z, and have Equal numbers of 0s and 1s. Missing subscripts indicate that we don't care about the first and/or last symbol. The nonterminals ${}_0E_0$ and ${}_0E_1$ never appear on the right side of a production, so we can ignore them. In each case, we derive the production rules from the grammar for $L(E) = \{w \mid \#(\mathtt{0}, w) = \#(\mathtt{1}, w)\}$ described in the lecture notes.

$$\begin{aligned}
S &\to \varepsilon \mid \mathtt{0} \mid \mathtt{1} \mid {}_0E\,\mathtt{0} \mid {}_0E_1 \mid {}_1E_0 \mid {}_1E\,\mathtt{1} \\
E &\to \varepsilon \mid EE \mid \mathtt{0}E\mathtt{1} \mid \mathtt{1}E\mathtt{0} \\
{}_0E &\to {}_0E\,E \mid \mathtt{0}E\mathtt{1} \\
{}_1E &\to {}_1E\,E \mid \mathtt{1}E\mathtt{0} \\
E_0 &\to E\,E_0 \mid \mathtt{1}E\mathtt{0} \\
E_1 &\to E\,E_1 \mid \mathtt{0}E\mathtt{1} \\
{}_0E_1 &\to {}_0E\,E_1 \mid \mathtt{0}E\mathtt{1} \\
{}_1E_0 &\to {}_1E\,E_0 \mid \mathtt{1}E\mathtt{0}
\end{aligned}$$

■

Solution (simpler counting): For any string w, let $\Delta(w) = \#(0, w) - \#(1, w)$. The case analysis in the previous solution implies that our target language L contains a binary string w if and only if one of the following conditions holds:

- $w = \varepsilon$
- w starts with 0 and ends with 1, and $\Delta(w) = 0$
- w starts with 1 and ends with 0, and $\Delta(w) = 0$
- w starts with 0 and ends with 0, and $\Delta(w) = 1$
- w starts with 1 and ends with 1, and $\Delta(w) = -1$

In the third case, either $w = 0$ or $w = 0x0$ for some string x with $\Delta(x) = -1$. If $w = 0x0$, then $\Delta(0) = +1$ and $\Delta(0x) = 0$, so w has a prefix with $\Delta = 0$, and the shortest such prefix must end with 1; it follows that $w = 0y1z0$ where $\Delta(y) = 0$ and $\Delta(z) = 0$. Similar analysis applies to the fourth case above.

We conclude with the following grammar for L, where E generates all strings with $\Delta = 0$.

$$S \to \varepsilon \mid 0 \mid 1 \mid 0E1 \mid 1E0 \mid 0E1E0 \mid 1E0E1$$
$$E \to \varepsilon \mid EE \mid 0E1 \mid 1E0$$

■

Solution (brute-force terminators): Sorry, this is really ugly, but it was the first solution I found.

Our grammar treats the last symbol in each run of 0s or 1s as a distinct symbol; we indicate this distinction by writing these symbols in blue with hats ($\hat{0}$ or $\hat{1}$). For example, we would write the string 110000111110111 as $1\hat{1}000\hat{0}1111\hat{1}\hat{0}11\hat{1}$. With this marking, every binary string matches the regular expression

$$(0^*\hat{0} + \varepsilon)(1^*\hat{1}0^*\hat{0})^*(1^*\hat{1} + \varepsilon).$$

Let L' be the set of all strings in $\{0, 1, \hat{0}, \hat{1}\}^*$ that match this regular expression and have equal numbers of red 0s and red 1s. We construct a grammar for L by first constructing a grammar for L', and then ignoring the colors in the terminal alphabet (but *not* in the non-terminal names).

The following grammar generates L', using a modification of the grammar $S \to \varepsilon \mid SS \mid 0S1 \mid 1S0$ for all binary strings with equal 0s and 1s. Specifically, each of the 16 non-terminals S_{ab} generates the strings in L' that start with a and end with b.

$$S \to \varepsilon \mid S_{0\hat{0}} \mid S_{0\hat{1}} \mid S_{1\hat{0}} \mid S_{1\hat{1}} \mid S_{\hat{0}\hat{0}} \mid S_{\hat{0}\hat{1}} \mid S_{\hat{1}\hat{0}} \mid S_{\hat{1}\hat{1}}$$

$$S_{\hat{0}\hat{0}} \to S_{\hat{0}0}\hat{0} \mid S_{\hat{0}\hat{1}}\hat{0} \mid \hat{0}$$
$$S_{\hat{0}\hat{1}} \to S_{\hat{0}1}\hat{1} \mid S_{\hat{0}\hat{0}}\hat{1}$$
$$S_{\hat{1}\hat{0}} \to S_{\hat{1}0}\hat{0} \mid S_{\hat{1}\hat{1}}\hat{0}$$
$$S_{\hat{1}\hat{1}} \to S_{\hat{1}1}\hat{1} \mid S_{\hat{1}\hat{0}}\hat{1} \mid \hat{1}$$

$$
\begin{aligned}
S_{0\hat{0}} &\to S_{00}\hat{0} \mid S_{0\hat{1}}\hat{0}\\
S_{0\hat{1}} &\to S_{01}\hat{1} \mid S_{0\hat{0}}\hat{1}\\
S_{1\hat{0}} &\to S_{10}\hat{0} \mid S_{1\hat{1}}\hat{0}\\
S_{1\hat{1}} &\to S_{11}\hat{1} \mid S_{1\hat{0}}\hat{1}
\end{aligned}
$$

$$
\begin{aligned}
S_{\hat{0}0} &\to \hat{0}S_{10} \mid \hat{0}S_{\hat{1}0}\\
S_{\hat{0}1} &\to \hat{0}S_{11} \mid \hat{0}S_{\hat{1}1}\\
S_{\hat{1}0} &\to \hat{1}S_{00} \mid \hat{1}S_{\hat{0}0}\\
S_{\hat{1}1} &\to \hat{1}S_{01} \mid \hat{1}S_{\hat{0}1}
\end{aligned}
$$

$$
\begin{aligned}
S_{00} &\to S_{00}S_{00} \mid S_{00}S_{\hat{0}0} \mid S_{01}S_{10} \mid S_{01}S_{\hat{1}0}\\
S_{01} &\to 0\,S_{01}\,1 \mid 0\,S_{\hat{0}1}\,1 \mid 0\,S_{0\hat{0}}\,1 \mid 0\,S_{\hat{0}\hat{0}}\,1\\
S_{10} &\to 1\,S_{10}\,0 \mid 1\,S_{\hat{1}0}\,0 \mid 1\,S_{1\hat{1}}\,0 \mid 1\,S_{\hat{1}\hat{1}}\,0\\
S_{11} &\to S_{10}S_{01} \mid S_{10}S_{\hat{0}1} \mid S_{11}S_{11} \mid S_{11}S_{\hat{1}1}
\end{aligned}
$$

In the first three groups of productions, we peel off as many blue bits as possible from either end of the string; most the real work is done in the last group. In the productions for S_{00} and S_{11}, we reduce the number of cases by splitting off the shortest non-empty prefix with equal 0s and 1s. ■

Rubric: 4 points = 2 for grammar + 2 for justification. These solutions have more detail than necessary for full credit.

2. Let $inc\colon \{\texttt{0},\texttt{1}\}^* \to \{\texttt{0},\texttt{1}\}^*$ denote the *increment* function, which transforms the binary representation of an arbitrary integer n into the binary representation of $n+1$, truncated to the same number of bits.

   Let $L \subseteq \{\texttt{0},\texttt{1}\}^*$ be an arbitrary regular language. Prove that $inc(L) = \{inc(w) \mid w \in L\}$ is also regular.

Solution (forward NFA): Let $M = (Q, s, A, \delta)$ be a DFA that accepts L. We construct an NFA $M' = (Q', s', A', \delta')$ that accepts $inc(L)$ as follows:

$$\begin{aligned}
Q' &= Q \times \{0,1\} \cup \{s'\}\\
s' &\text{ is an explicit state in } Q'\\
A' &= \{(q,1) \mid q \in Q\}
\end{aligned}$$

$$\begin{aligned}
\delta(s',\varepsilon) &= \big\{(s,0),\ (s,1)\big\}\\
\delta(s',a) &= \varnothing && \text{for all } a \in \Sigma
\end{aligned}$$

$$\begin{aligned}
\delta'((q,0),\texttt{0}) &= \big\{(\delta(q,\texttt{0}),0)\big\} && \text{for all } q \in Q\\
\delta'((q,0),\texttt{1}) &= \big\{(\delta(q,\texttt{1}),0),\ (\delta(q,\texttt{0}),1)\big\} && \text{for all } q \in Q\\
\delta'((q,1),\texttt{0}) &= \big\{(\delta(q,\texttt{1}),1)\big\} && \text{for all } q \in Q\\
\delta'((q,1),\texttt{1}) &= \varnothing && \text{for all } q \in Q
\end{aligned}$$

$$\delta'((q,b),\varepsilon) = \varnothing \qquad \text{for all } q \in Q \text{ and } b \in \{0,1\}$$

Our machine M' reads a string of the form $x\texttt{1}\texttt{0}^n$ or $\texttt{0}^n$ and passes the *decremented* string $x\texttt{0}\texttt{1}^n$ or $\texttt{1}^n$ to M. State $(q,0)$ indicates that M is is in state q and M' is reading the initial prefix x; state $(q,1)$ indicates that M is in state q and M' is flipping bits before passing them to M. When M' begins or reads a $\texttt{1}$, it guesses whether to switch from passing bits directly to M to inverting them. ■

Rubric: 10 points: standard language transformation rubric.

Solution (reverse DFA): Recall from class that the reversal $rev(L) = \{rev(w) \mid w \in L\}$ of any regular language L is regular.

For any string w, define $linc(w) := rev(inc(rev(w)))$, and for any language L, define

$$linc(L) \;:=\; \{linc(w) \mid \in L\} \;=\; rev(inc(rev(L))).$$

The name *linc* is short for "left increment"; this is the increment function for binary strings whose *least* significant bits are on the *left*. For any integer $n \geq 0$ and any string x, we have $linc(\texttt{1}^n\texttt{0}x) = \texttt{0}^n\texttt{1}x$ and $linc(\texttt{1}^n) = \texttt{0}^n$.

I claim that for any regular language L, the language $linc(L)$ is also regular. This claim immediately implies that $inc(L) = rev(linc(rev(L)))$ is regular, solving the homework problem.

To prove my claim, let $M = (Q, s, A, \delta)$ be an arbitrary DFA that accepts L. We construct a ***DFA*** $M' = (Q', s', A', \delta')$ that accepts $linc(L)$ as follows:

$$
\begin{aligned}
Q' &= Q \times \{0, 1\} \\
s' &= (s, 1) \\
A' &= A \times \{0, 1\}
\end{aligned}
$$

$$
\begin{aligned}
\delta'((q,1), \texttt{0}) &= (\delta(q, \texttt{1}), 1) && \text{for all } q \in Q \\
\delta'((q,1), \texttt{1}) &= (\delta(q, \texttt{0}), 0) && \text{for all } q \in Q
\end{aligned}
$$

$$
\begin{aligned}
\delta'((q,0), \texttt{0}) &= (\delta(q, \texttt{0}), 0) && \text{for all } q \in Q \\
\delta'((q,0), \texttt{1}) &= (\delta(q, \texttt{1}), 0) && \text{for all } q \in Q
\end{aligned}
$$

The transition function can be more concisely written as

$$\delta'((q, b), a) = \big(\delta(q, a \oplus b),\ a \wedge b\big)$$

for all $q \in Q$ and all $a, b \in \{0, 1\}$. Intuitively, our new machine M' reads a string of the form $\texttt{0}^n\texttt{1}x$ or $\texttt{0}^n$ and passes the *decremented* string $\texttt{1}^n\texttt{0}x$ or $\texttt{1}^n$ to M. Each state (q, b) indicates that M is in state q and the current “borrow” bit is b. Equivalently, $b = 1$ if and only if M' has not yet read a $\texttt{0}$. ■

Rubric: 10 points = 2 for defining *linc* + 5 points for *linc* transformation (standard language transformation rubric, scaled) + 1 for reversal transformation (from lecture/notes) + 2 for remaining details. No credit for assuming/building a DFA that “reads its input from right to left”; that’s not how DFAs are defined.

3. Prove that if L and L' are regular languages, then *shuffles*(L, L') is also a regular language.

Solution: Let L_1 and L_2 be arbitrary regular languages. Let $M_1 = (Q_1, s_1, A_1, \delta_1)$ be an arbitrary DFA that accepts L_1, and let $M_2 = (Q_2, s_2, A_2, \delta_2)$ be an arbitrary DFA that accepts L_2. We build a new NFA $M = (Q, s, A, \delta)$ that accepts *shuffles*(L_1, L_2) using a modified product construction:

$$
\begin{aligned}
Q &= Q_1 \times Q_2 \\
s &= (s_1, s_2) \\
A &= A_1 \times A_2 \\
\delta((q_1, q_2), a) &= \big\{(\delta_1(q_1, a), q_2),\ (q_1, \delta_2(q_2, a))\big\}
\end{aligned}
$$

Intuitively, M runs the given machines M_1 and M_2 simultaneously. At each step, M nondeterministically chooses whether to pass the next input symbol to M_1 or to M_2. Finally, when the input is consumed, M accepts if and only of both M_1 and M_2 are in accepting states. ■

Rubric: 10 points: standard language transformation rubric