1. Prove whether the following languages are regular or not.

(a) Strings over the alphabet $\Sigma = \{0, \dots, 9, \#\}$ that contain a substring $c\#^c c$, where $c \in \{0, \dots, 9\}$. E.g., 382103###38592, 7892##234 and 00 are in the language.

**Solution:** $\Sigma^*(00 + 1\#1 + 2\#\#2 + 3\#^3 3 + 4\#^4 4 + 5\#^5 5 + 6\#^6 6 + 7\#^7 7 + 8\#^8 8 + 9\#^9 9)\Sigma^*$

Since we can make a regular expression for this language, the language is regular. ■

(b) Strings over the alphabet $\Sigma = \{0, \dots, 9, \#\}$ of the form $\langle n \rangle \#^n$, where $n$ is a sequence of digits interpreted as a decimal number. E.g., 0, 3###, 11########### are in the language.

**Solution:** Let $F = \{1^n \mid n > 0\}$.
Let $x$ and $y$ be arbitrary distinct elements of $F$.
Then $x = 1^i$ and $y = 1^j$ for some positive integers $i$ and $j$, where $i \neq j$.
Let $z = \#^{<x>}$.
Then $xz = 1^i \#^{<1^i>} \in L$
And $yz = 1^j \#^{<1^i>} \notin L$
Thus, $F$ is a fooling set for $L$.
Since there are infinitely many elements in $F$, $L$ cannot be regular.

■

(c) Strings over the alphabet $\Sigma = \{a, b, \dots, z\}$ that have the same 3 characters repeated in two places. E.g., ur**bane****ban**ana, trampo**lin**ejugg**lin**g, acclim**ati**z**ati**on.

**Solution:**

$$\Sigma^*((aaa)\Sigma^*(aaa) + aab\Sigma^*(aab) + \dots + (zzz)\Sigma^*(zzz))\Sigma^*$$

Since we can make a regular expression for the language (albeit a long and unwieldy one), the language is regular.

Alternately, for any *specific* three character sequence xyz, the language of strings where that sequence repeats twice is clearly regular since it has the regex $\Sigma^*$xyz$\Sigma^*$xyz$\Sigma^*$. Our desired language is the union of all of the $26^3$ such languages. Since $26^3$ is finite, the union is also regular. ■

2. Let $f : \Sigma_1 \to \Sigma_2^*$ be a function from symbols in one alphabet to strings in another. We can extend $f$ to apply to strings in $\Sigma_1^*$ by the following recursive definition:

$$\begin{aligned} f(\epsilon) &= \epsilon \\ f(ax) &= f(a) \cdot f(x) && \text{for } a \in \Sigma_1, x \in \Sigma_1^* \end{aligned}$$

Likewise, we can apply $f$ to languages by defining $f(L) = \{f(w) | w \in L\}$.

$f$ is known as a *language homomorphism*. For example, we can define $f$ to map `0` to `batman` and `1` to `robin`, then $f(110) =$ `robinrobinbatman`. As another example, we can define $f_{\text{ASCII}}$ that maps each character to its 8-bit ASCII binary representation, in which case $f_{\text{ASCII}}($`374`$) =$ `001100110011011100110100`.

Given a DFA $M$ that accepts $L$, show how to construct an NFA $N$ that accepts $f(L)$. Formally prove the correctness of your construction.

Note that we are looking for an explicit construction of an NFA here, rather than simply a proof that $f(L)$ is regular, which implies the existence of such an NFA $N$.

**Solution:** We are given $f : \Sigma_1 \to \Sigma_2^*$ and a DFA $M = (\Sigma_1, Q_1, s_1, A_1, \delta_1)$. We will construct an NFA $N = (\Sigma_2, Q_2, s_2, A_2, \delta_2)$

If we want $N$ to accept the language $f(L)$, we want to convert every transition $\delta(q, a) = p$ in M into a series of transitions and new states so that $p \in \delta^*(q, f(a))$ in N. In order to achieve the goal, we first define the set of states as:

$$\begin{aligned} Q_2 &= Q_1 \cup Q_1 \times \Sigma_1 \times \{1, 2, ..., n\}, \\ n &= \max_{a \in \Sigma_1} |f(a)| \end{aligned}$$

Intuitively, the states $q$ for $q \in Q_1$ correspond to being in the same state in the original DFA. The state $(q, c, i)$ corresponds to having seen $i$ characters of $f(c)$ after arriving from state $q$.

Then, we define the transition function $\delta_2$ for N as following

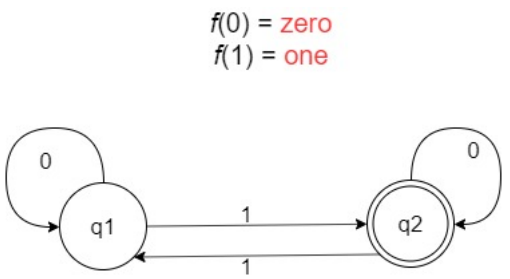
$$\begin{aligned} \delta_2(q, \epsilon) &= \{(q, a, 0) | a \in \Sigma_1\}, && \text{for } q \in Q_1 \\ \delta_2((q, a, i), f(a)[i+1]) &= \{(q, a, i+1)\}, && \text{for } q \in Q_1,\ a \in \Sigma_1,\ 0 \le i \le |f(a)| - 1 \\ \delta_2((q, a, |(f(a)|), \epsilon) &= \{\delta_1(q, a)\}, && \text{for } q \in Q_1,\ a \in \Sigma_1 \end{aligned}$$

Note that $f(a)[i]$ denotes the the $i^{th}$ character in string $f(a)$ and $|f(a)|$ denotes the length of $f(a)$. The idea here is that we build a simple NFA $N_a$ that matches $f(a)$ using $|f(a)| + 1$ states and add a copy of it for every DFA state $q$, using states $(q, a, 0), \ldots, (q, a, |f(a|)$. In each state $q$ we guess which DFA character would be used next by making an $\epsilon$-transition to $(q, a, 0)$, and once all of $f(a)$ is matched, we take another epsilon transition from $(q, a, |f(a)|)$ to $\delta_1(q, a)$

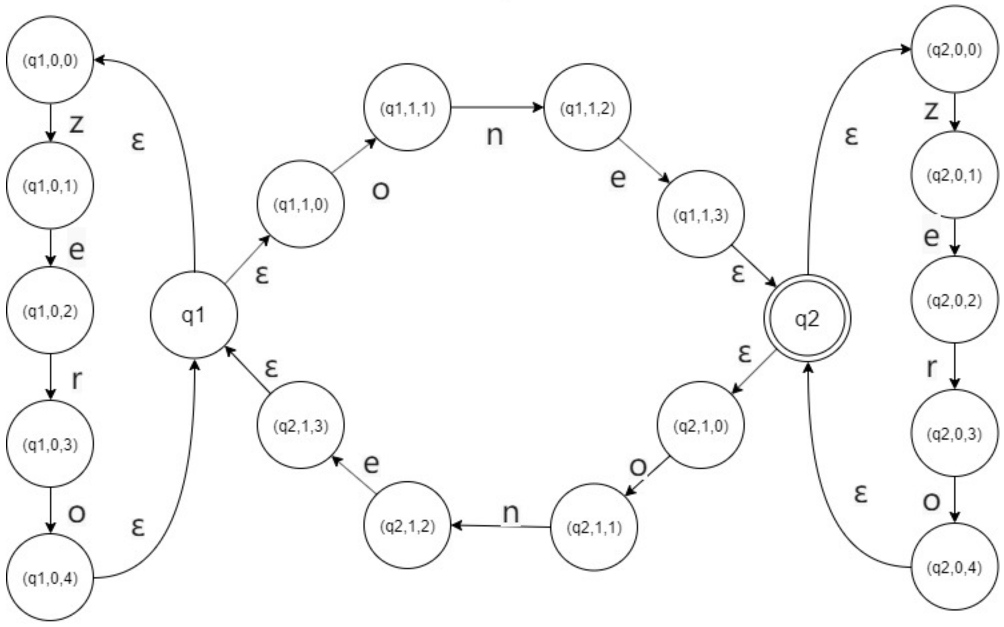
The new starting state will still be $s_1$ and the new accepting states will be also remain unchanged:

$$\begin{aligned} s_2 &= s_1 \\ A_2 &= A_1 \end{aligned}$$

An example for such construction is provided in the following figure.



Example DFA M



Corresponding NFA N

Below we provide a formal proof of correctness. Note that this proof is long and tedious; it's shown here to give you some perspective on the things you have to think about when writing proofs. Some of the lemmas could probably be stated directly in Before proving the correctness of the construction, we first prove a couple of lemmas about NFAs that will be useful.

**Lemma 1.** *For any state $q \in Q$, $\epsilon\text{—reach}(\epsilon\text{—reach}(q)) = \epsilon\text{—reach}(q)$*

(As in the notes, when $S \subset Q$ we use $\epsilon\text{—reach}(S)$ to mean $\bigcup_{q\in S} \epsilon\text{—reach}(q)$. Likewise, $\delta(S,x) = \bigcup_{q\in S} \delta(q,x)$ and $\delta^*(S,x) = \bigcup_{q\in S} \delta^*(q,x)$.)

**Proof:** For any $r \in \epsilon\text{—reach}(q)$, we have $r \in \epsilon\text{—reach}(r)$. Therefore:

$$\epsilon\text{—reach}(q) = \bigcup_{r\in\epsilon\text{—reach}(q)} r \subset \bigcup_{r\in\epsilon\text{—reach}(q)} \epsilon\text{—reach}(r) = \epsilon\text{—reach}(\epsilon\text{—reach}(q))$$

We also have that $\epsilon\text{—reach}(r) \subset \epsilon\text{—reach}(q)$, so:

$$\epsilon\text{—reach}(\epsilon\text{—reach}(r)) = \bigcup_{r \in \epsilon\text{—reach}(q)} \epsilon\text{—reach}(r) \subset \bigcup_{r \in \epsilon\text{—reach}(q)} \epsilon\text{—reach}(q) = \epsilon\text{—reach}(q)$$

□

**Lemma 2.** *For any $w, y \in \Sigma^*, q \in Q$:*

$$\delta^*(q, wy) = \delta^*(\delta^*(q, w), y)$$

**Proof:** Let $w$ and $y$ be arbitrary strings from $\Sigma^*$. Assume $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$ for every possible $x$ that is shorter than $w$, i.e., $|x| < |w|$.

- Suppose $|w| = 0$, $w = \epsilon$. Either $y = \epsilon$, in which case:

$$\begin{aligned}
\delta^*(\delta^*(q, w), y) &= \delta^*(\delta^*(q, \epsilon), \epsilon) & \\
&= \delta^*(\epsilon\text{—reach}(q), \epsilon) & \text{by definition of } \delta^* \\
&= \bigcup_{r \in \epsilon\text{—reach}(q)} \epsilon\text{—reach}(r) & \text{by definition of } \delta^* \\
&= \epsilon\text{—reach}(q) & \text{by Lemma 1} \\
&= \delta^*(q, \epsilon) = \delta^*(q, wy) & \text{by definition of } \delta^*
\end{aligned}$$

  Otherwise, $y = az$ for $a \in \Sigma, z \in \Sigma^*$

$$\begin{aligned}
\delta^*(q, wy) = \delta^*(q, \epsilon y) = \delta^*(q, az) & & w = \epsilon, y = az \\
&= \delta^*(\delta(\epsilon\text{—reach}(q), a), z) & \text{by definition of } \delta^* \\
&= \delta^*(\delta(\epsilon\text{—reach}(\epsilon\text{—reach}(q)), a), z) & \text{by Lemma 1} \\
&= \delta^*(\delta(\epsilon\text{—reach}(\delta^*(q, \epsilon)), a), z) & \text{by definition of } \delta^* \\
&= \delta^*(\delta^*(q, \epsilon), az) & \text{by definition of } \delta^* \\
&= \delta^*(\delta^*(q, w), y) & w = \epsilon, y = az
\end{aligned}$$

- Suppose $w = ax$, for some $a \in \Sigma$ and $x \in \Sigma^*$. Then $|x| < |w|$

$$\begin{aligned}
\delta^*(q, wy) &= \delta^*(q, axy) & w = ax \\
&= \delta^*(\delta(\epsilon\text{—reach}(q), a), xy) & \text{by definition of } \delta^* \\
&= \delta^*(\delta^*(\delta(\epsilon\text{—reach}(q), a), x), y) & \text{by inductive hypothesis} \\
&= \delta^*(\delta^*(q, ax), y) & \text{by definition of } \delta^* \\
&= \delta^*(\delta^*(q, w), y) & w = ax
\end{aligned}$$

Therefore, we conclude that $\delta^*(q, wy) = \delta^*(\delta^*(q, w), y)$ is true for any possible $wy \in \Sigma^*$ and $q \in Q$.

□

Next we will prove a key property of the constructed NFA. Let $\text{substring}(x, i, k)$ be the substring of $x$ of length $k$ starting at position $i$.

**Lemma 3.** *For any $q \in Q_1$, $a \in \Sigma_1$ and $x \in \Sigma_2^*$, $i$ a non-negative integer with $|x| + i \leq |f(a)|$, we have:*

$$\begin{aligned}
\delta_2^*((q, a, i), x) &= \epsilon\text{—reach}((q, a, i + |x|)) & \textit{if } x = \text{substring}(f(a), i, |x|) \\
\delta_2^*((q, a, i), x) &= \emptyset & \textit{otherwise}
\end{aligned}$$

This result is a straightforward consequence of the way the NFA is constructed, which we can prove by induction on $|x|$

**Proof:** Suppose the lemma holds true for all $y \in \Sigma_2^*$ with $|y| < |x|$.

First consider $|x| = 0$. Then $x = \epsilon$ and $x$ equal to substring$(f(a), i, 0)$). By definition of $\delta_2^*$,

$$\delta_2^*((q,a,i),x) = \delta_2^*((q,a,i),\epsilon) = \epsilon\text{—reach}((q,a,i)) = \epsilon\text{—reach}((q,a,|x|))$$

Suppose now that $|x| > 0$. Then $x = yc$ for $y \in \Sigma_2^*, c \in \Sigma_2$. If $x =$ substring$(f(a), i, |x|)$ then $y =$ substring$(f(a), i, |x| - 1)$ and $c = f(a)[|x| + i]$. Therefore:

$$\begin{aligned}\delta_2^*((q,a,i),y) &= \epsilon\text{—reach}((q,a,i+|x|-1)) && \text{by the inductive hypothesis}\\ &= \{(q,a,i+|x|-1)\} && \text{since } (q,a,i+|x|-1) \text{ has no } \epsilon\text{-transitions for } i+|x|-1 < |f(a)|\end{aligned} \tag{1}$$

Then:

$$\begin{aligned}\delta_2^*((q,a,i),x) &= \delta_2^*((q,a,i),yc)\\ &= \delta_2^*(\delta_2^*((q,a,i),y),c) && \text{by Lemma 2}\\ &= \delta_2^*((q,a,i+|x|-1),c) && \text{per (1)}\\ &= \delta_2^*(\delta_2((q,a,i+|x|-1,f(a)[i+|x|]),\epsilon) && \text{by definition of } \delta_2^*\\ &= \delta_2^*((q,a,|x|+i),\epsilon) && \text{by definition of } \delta_2\\ &= \epsilon\text{—reach}((q,a,|x|+i) && \text{by definition of } \delta_2^*\end{aligned}$$

If $x \neq$ substring$(f(a), i, |x|)$ but $y =$ substring$(f(a), i, |x| - 1)$, we must have $c \neq f(a)[|x| + i]$. In this case:

$$\begin{aligned}\delta_2^*((q,a,i),x) &= \delta_2^*((q,a,i+|x|-1),c) && \text{as above}\\ &= \delta_2^*(\delta_2((q,a,i+|x|-1,c),\epsilon) && \text{by definition of } \delta_2^*\\ &= \delta_2^*(\emptyset,\epsilon) && \text{by definition of } \delta_2\text{, since } c \neq f(a)[|x|+i]\\ &= \emptyset\end{aligned}$$

On the other hand, if $y \neq$ substring$(f(a), i, |x| - 1)$ then:

$$\begin{aligned}\delta_2^*((q,a,i),x) &= \delta_2^*((q,a,i),yc)\\ &= \delta_2^*(\delta_2^*((q,a,i),y),c) && \text{by Lemma 2}\\ &= \delta_2^*(\emptyset,c) && \text{by the inductive hypothesis}\\ &= \emptyset\end{aligned}$$

This completes the case analysis and proves the lemma. □

Our next lemma ties $\delta_1^*$ and $\delta_2^*$.

**Lemma 4.** *For $x \in \Sigma_1^*$, $q \in Q_1$, we have that:*

$$\delta_1^*(q,x) \in \delta_2^*(q,f(x))$$

**Proof:** Again, we can prove this by induction on $|x|$. Suppose that the lemma holds for all $y$ with $|y| < |x|$. If $|x| = 0$ then

$$\delta_1^*(q,x) = \delta_1^*(q,\epsilon) = q$$

$$\delta_2^*(q,f(x)) = \delta_2^*(q,f(\epsilon)) = \delta_2^*(q,\epsilon) = \epsilon\text{—reach}(q)$$

Since $q \in \epsilon\text{—reach}(q)$ the result holds. If $|x| > 0$ then $x = cy$ for some $y \in \Sigma_1^*, c \in \Sigma_1$. First we have:

$$\delta_1^*(q,x) = \delta_1^*(q,cy) = \delta_1^*(\delta_1(q,c),y)$$

Next, since $(q,c,0) \in \epsilon\text{—reach}(q)$, $\delta_2^*((q,c,0),f(x)) \subset \delta_2^*(q,f(x))$. We now have:

$$\begin{aligned}
\delta_2^*((q,c,0),f(x)) &= \delta_2^*(q,f(cy)) \\
&= \delta_2^*((q,c,0),f(c)f(y)) && \text{by homomorphism of } f \\
&= \delta_2^*(\delta_2^*((q,c,0),f(c)),f(y)) && \text{by Lemma 2} \\
&= \delta_2^*(\epsilon\text{—reach}((q,c,|f(c)|)),f(y)) && \text{by Lemma 3} \\
&\supset \delta_2^*(\delta_1(q,c),f(y)) && \text{since } \delta_2((q,c,|f(c)|),\epsilon) = \{\delta_1(q,c)\}
\end{aligned}$$

By the inductive hypothesis, $\delta_1^*(\delta_1(q,c),y) \in \delta_2^*(\delta_1(q,c),f(y))$. Since $\delta_2^*(\delta_1(q,c),f(y)) \subset \delta_2^*((q,c,0),f(x)) \subset \delta_2^*(q,f(x))$ we have $\delta_1^*(q,x) = \delta_1^*(\delta_1(q,c),y) \in \delta_2^*(q,f(x))$, proving the lemma. □

Given $y \in f(L)$, there must exist $x \in L$ such that $f(x) = y$. Since $x \in L$, $\delta_1^*(s,x) \in A$. By the above lemma, $\delta_1^*(s,x) \in \delta_2^*(s,f(x)) = \delta_2^*(s,y)$. Therefore $\delta_2^*(s,y) \cap A \neq \emptyset$ and therefore $y \in L(N)$.

To conclude the proof, we need to show that if $y \in L(N)$, then there must be some $x \in L$ with $f(x) = y$. First we need to deal with the fact that $f(c)$ could be $\epsilon$ for some $c \in \Sigma_1$.

**Lemma 5.** *For $q,p \in Q_1$, if $q \in \epsilon\text{—reach}(p)$ then $\exists x \in \Sigma_1^*$ such that $q = \delta_1^*(p,x)$ and $f(x) = \epsilon$.*

**Proof:** If $q \in \epsilon\text{—reach}(p)$ then there must exist a sequence of states $r_1, r_2, \ldots, r_n$ with $r_1 = p, r_n = q$ and $r_{i+1} \in \delta_2(r_i,\epsilon)$. We will prove the result by induction of the length of this sequence. Note first that our NFA has two types of $\epsilon$-transitions: from a state $q \in Q_1$ to $(q,c,0)$ for some $c \in \Sigma_1$ and from $(q,c,|f(c)|)$ to $\delta_1(q,c)$. It is therefore easy to see that in the sequence $r_1, r_2, \ldots, r_n$, $n$ must be odd; every odd $r_i$ is in $Q_1$ and every even $r_i$ is of the form $(r_{i-1},c,0)$ for some $c \in \Sigma_1$.

Suppose that the lemma is true for every sequence $t_1, \ldots, t_m$ for $m < n$. If $n = 1$, then $q = p$, and setting $x = \epsilon$ gets us the desired result that $f(x) = \epsilon$ and $\delta_1^*(p,x) = \delta_1^*(p,\epsilon) = p = q$.

Now suppose that $n > 1$. Then $r_{n-1} = (r_{i-2},c,0)$ for some $c \in \Sigma_1$. The existence of an $\epsilon$-transition from $r_{n-1}$ to $r_n \in Q_1$ implies that $|f(c)| = 0$ (so $f(c) = \epsilon$)) and $\delta_1(r_{i-2},c) = r_n$. By the inductive hypothesis, there exists $y \in \Sigma_1^*$ such that $\delta_1^*(p,y) = r_{n-2}$ and $f(y) = \epsilon$. Then $\delta_1^*(p,yc) = \delta_1(\delta_1^*(p,y),c) = \delta_1(r_{n-2},c) = r_n = q$ and $f(yc) = f(y)f(c) = \epsilon$, proving the lemma. □

We're finally ready for the lemma that will imply the main result.

**Lemma 6.** *If $p,q \in Q_1$ and $x \in \Sigma_2^*$ with $q \in \delta_2^*(p,x)$, then there exists $y \in \Sigma_1^*$ with $f(y) = x$ and $\delta_1^*(p,x) = q$*

**Proof:** We prove this by induction on $|x|$. If $|x| = 0$ then $\delta_2^*(p,x) = \delta_2^*(p,\epsilon) = \epsilon\text{—reach}(p)$. By Lemma 5, $q \in \epsilon\text{—reach}(p)$ implies that there is a $y \in \Sigma_1^*$ with $f(y) = \epsilon = x$ and $\delta_1^*(p,y) = q$.

Suppose $|x| > 0$ and assume that the lemma holds for all $y \in \Sigma_2^*$ with $|y| < |x|$. Let $x = aw$ for $a \in \Sigma_2$ and $w \in \Sigma_2^*$.

$$\delta_2^*(p,x) = \delta_2^*(p,aw) = \delta_2^*(\delta_2(\epsilon\text{—reach}(p),a),w)$$

Since $q \in \delta_2^*(p,x)$, there must be some $r \in \epsilon\text{—reach}(p)$ with $q \in \delta_2^*(\delta_2(r,a),w)$. Note that for $r \in Q_1$, $\delta_2(r,a) = \emptyset$, so we must have that $r = (r',c,0)$ for some $r' \in Q_1, c \in \Sigma_1$. Since the only $\epsilon$-transition to $(r',c,0)$ is from $r'$, it must be that $r' \in \epsilon\text{—reach}(p)$ and thus by Lemma 5 there is a $u \in \Sigma_1$ such that $f(z) = \epsilon$ and $\delta_1^*(p,u) = r'$.

Since $\delta_2(r,a) \neq \emptyset$, by Lemma 3, we must have $a = f(c)[1]$. Furthermore, for $1 \leq i \leq |f(c)|-1$, it must be the case that substring$(w,1,i) =$ substring$(f(c),2,i)$, since otherwise

$$\begin{aligned}
\delta_2^*(\delta_2(r,a),w) &= \delta_2^*((r',c,1),w) && \text{since } a = f(c)[1]\\
&= \delta_2^*(\delta_2^*((r',c,1),\text{substring}(w,1,i)),\text{substring}(w,1+i,|w|-i)) && \text{by Lemma 2}\\
&= \delta_2^*(\emptyset,\text{substring}(w,1+i,|w|-i) && \text{by Lemma 3}\\
&= \emptyset
\end{aligned}$$

Let $z = \text{substring}(w, 1+|f(c)|-1, |w|-|f(c)|+1)$. Then

$$\begin{aligned}
x = aw &= a \cdot \text{substring}(w,1,|f(c)|-1) \cdot \text{substring}(w,1+|f(c)|-1,|w|-|f(c)|+1)\\
&= f(c) \cdot z
\end{aligned}$$

Since $q \in \delta_2^*(\delta_2(r,a),w)$ we must have:

$$\begin{aligned}
q \in \delta_2^*(\delta_2(\epsilon\text{—reach}(r),a),w) &= \delta_2^*(r,aw)\\
&= \delta_2^*(r,f(c)\cdot z)\\
&= \delta_2^*(\delta_2^*(r,f(c)),z) && \text{by Lemma 2}\\
&= \delta_2^*(\epsilon\text{—reach}((r',c,|f(c)|)),z) && \text{by Lemma 3}\\
&= \delta_2^*(\epsilon\text{—reach}(\delta_1(r',c)),z) && \text{since } \delta_2((r',c,|f(c)|),\epsilon) = \{\delta_1(r',c)\}\\
&= \delta_2^*(\delta_1(r',c),z)
\end{aligned}$$

Since $|z| < |x|$ by the inductive hypothesis there exists $z' \in \Sigma_1^*$ with $f(z') = z$ and $\delta_1^*(\delta_1(r',c),z') = q$. Putting everything together, we have:

$$\begin{aligned}
f(u\cdot c\cdot z') &= \epsilon \cdot f(c) \cdot z = x\\
\delta_1^*(p,u\cdot c\cdot z') &= \delta_1^*(\delta_1(p,u),c\cdot z')\\
&= \delta_1^*(r',c\cdot z')\\
&= \delta_1^*(\delta_1(r',c),z') = q
\end{aligned}$$

□

Therefore, if $x \in L(N)$ then for some $q \in A$, $q \in \delta_2^*(s,x)$. By the above Lemma, there exists $y \in \Sigma_1^*$ with $f(y) = x$ and $\delta_1^*(s,y) = q$. Since $q \in A$, $y \in L$, and therefore $x \in f(L)$

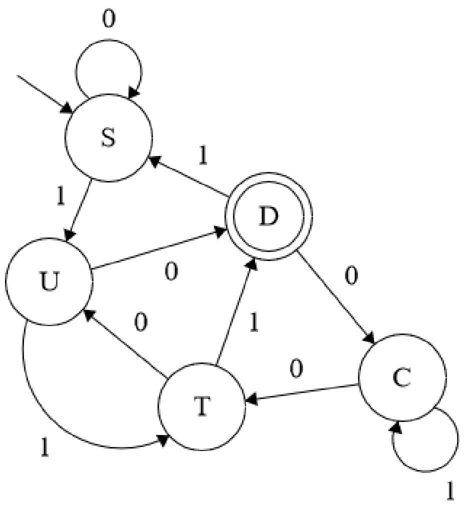
3. Give a context-free grammar for the following languages. You must specify what language is generated by each non-terminal and briefly explain why.

   (a) Binary strings that have remainder of 2 when divided by 5 (e.g., 111, 10, 10001).

   **Solution:**

$$\begin{aligned}
S &\rightarrow 0S \mid 1U\\
U &\rightarrow 0 \mid 0D \mid 1T\\
D &\rightarrow 0C \mid 1S\\
T &\rightarrow 0U \mid 1 \mid 1D\\
C &\rightarrow 0T \mid 1C
\end{aligned}$$

   This language is in fact regular; the DFA from which the production rules above are derived is here shown.



   Every transition from a state $a$ on symbol $x$ to state $b$ (where states $a$ and $b$ each represent different remainders when the string read so far is divided by 5) is paralleled by expanding a non-terminal $A$ into the symbol $x$ followed by a non-terminal $B$. Therefore each of $S$ (the starting non-terminal), $U$ ('uno'), $D$ ('dos'), $T$ ('tres'), and $C$ ('cuatro') generates some arbitrary binary string, this string being suffixed to an already generated binary prefix whose value has a remainder of 0, 1, 2, 3, or 4 respectively when divided by 5. The expansion of $U$ to 0 *or* 0D, and that of $T$ to 1 *or* 1D, represents the option of terminating the resulting 'acceptable' string, since its value will have remainder 2 when divided by 5, or the option of continuing to expand the string beyond the current 'acceptable' result. ■

   (b) Strings over the alphabet {0, 1} that have two blocks of 0's of equal length. E.g., 0011$\underline{000}$1$\underline{000}$1110 or 1$\underline{0}$11001110001$\underline{0}$ but not 0 or 0100.

   **Solution:**

$$\begin{aligned}
S &\rightarrow CVD\\
C &\rightarrow F1 \mid \varepsilon\\
D &\rightarrow 1F \mid \varepsilon\\
V &\rightarrow 0W0\\
W &\rightarrow 0W0 \mid 1 \mid 1F1\\
F &\rightarrow 0F \mid 1F \mid \varepsilon
\end{aligned}$$

- $F$ generates any arbitrary string of `0`s and `1`s, as these are what the portions of the string not corresponding to the two equal-length blocks of `0`s consist of.
- $W$ generates blocks of zeros of length $\geq 0$ which are separated by either a single `1` or an arbitrary string beginning and ending with `1` (so that neither block of `0`s runs together with the substring separating the blocks).
- $V$ generates a language similar to that generated by $W$, except that the blocks of `0`s are of length $\geq 1$ (since blocks of length 0 amount to blocks that do not exist).
- $C$ generates an arbitrary string which ends with a `1` (to prevent the arbitrary string and the first block of `0`s from possibly running together), or alternatively the empty string (to allow the first block of `0`s to start the string).
- $D$ generates an arbitrary string which starts with a `1` (to prevent the second block of `0`s and the arbitrary string from possibly running together), or alternatively the empty string (to allow the second block of `0`s to end the string).
- $S$, as the starting non-terminal, generates two separated blocks of `0`s optionally padded with arbitrary strings on either side (the one on the left ending with `1` and the one on the right beginning with `1`, as applicable), and thus generates the language described in the subproblem.

■

(c) Arithmetic expressions over decimal numbers using addition (`+`), multiplication (`*`), and exponentiation (`^`) with minimal parentheses. Here are the rules:

- The usual precedence rules apply, so `1+2*3^4` is equivalent to `1+(2*(3^4))`
- Any parentheses that could be removed without changing the meaning of the expression are not allowed. E.g., `1+(2*(3^4))` is an invalid expression, as are `(2*3)+5`, `3+(4+8)`, `(4+6)`, `3^((4+5))`. `2*(3+5)`, however, is valid.
- Since exponentiation is not associative, any double (or more) exponentiation must be parenthesized to remove ambiguity. I.e., `2^3^4` is invalid, instead you have to write `(2^3)^4` or `2^(3^4)`. Likewise `(1+2)^(3*4)^5` is invalid.

**Solution:**

$$
\begin{aligned}
S &\to A \mid M \mid E \mid D \\
A &\to B\texttt{+}B \\
B &\to A \mid M \mid E \mid D \\
M &\to N\texttt{*}N \\
N &\to \texttt{(}A\texttt{)} \mid M \mid E \mid D \\
E &\to F\texttt{\^{}}F \\
F &\to \texttt{(}A\texttt{)} \mid \texttt{(}M\texttt{)} \mid \texttt{(}E\texttt{)} \mid D \\
D &\to \texttt{0} | \texttt{1}G | \texttt{2}G | \texttt{3}G | \texttt{4}G | \texttt{5}G | \texttt{6}G | \texttt{7}G | \texttt{8}G | \texttt{9}G \\
G &\to \texttt{0}G | \texttt{1}G | \texttt{2}G | \texttt{3}G | \texttt{4}G | \texttt{5}G | \texttt{6}G | \texttt{7}G | \texttt{8}G | \texttt{9}G | \varepsilon
\end{aligned}
$$

- $G$ generates arbitrary decimal numbers of length $\geq 0$, as these are what the operations in the arithmetic expression fundamentally deal with.
- $D$ generates any arbitrary decimal number of length $\geq 1$ (the omission of $\varepsilon$ and the substitution of `0` for `0`G, when comparing this production rule to that for $G$, is to prevent including the empty string or decimal numbers beginning with `0`–other than `0` itself–as arithmetic expressions).
- $A$ generates arithmetic expressions in which two operands are added. $M$ does similarly for when two operands are multiplied and $E$ does similarly for when an operand is raised to the power of another.

- $B$ generates arithmetic expressions denoting an operand in an addition expression. $N$ does similarly for a multiplication expression and $F$ does similarly for an exponentiation expression. While all may be expanded to decimal numbers $D$ without the need for parentheses in any case, there are some differences in parenthesis inclusion when handling subexpressions:
  - Because addition occurs last in the order of precedence, no parentheses are necessary when $B$ expands to $M$ or $E$. Because addition is associative, expansion from $B$ to $A$ does not require parentheses either.
  - Because multiplication occurs before addition and after exponentiation in the order of precedence, parentheses are needed when $N$ expands to $A$ but not when $N$ expands to $E$. Because multiplication is associative, no parentheses are necessary when $N$ expands to $M$.
  - Because exponentiation occurs first in the order of precedence, parentheses are necessary when $F$ expands to $A$ or $M$. Because exponentiation is not associative, expansion from $F$ to $E$ requires parentheses as well.
- $S$, as the starting non-terminal, generates a top-level expression (be it a decimal number or a subexpression involving addition, multiplication, or exponentiation), and thus generates the language described in the subproblem.

■

## Solved problem

4. Let $L$ be the set of all strings over $\{\texttt{0},\texttt{1}\}^*$ with exactly twice as many 0s as 1s.

   (a) Describe a CFG for the language $L$.
   *[Hint: For any string $u$ define $\Delta(u) = \#(\texttt{0},u) - 2\#(\texttt{1},u)$. Introduce intermediate variables that derive strings with $\Delta(u) = 1$ and $\Delta(u) = -1$ and use them to define a non-terminal that generates $L$.]*

   **Solution:** $S \to \varepsilon \mid SS \mid \texttt{00}S\texttt{1} \mid \texttt{0}S\texttt{1}S\texttt{0} \mid \texttt{1}S\texttt{00}$ ■

   (b) Prove that your grammar $G$ is correct. As usual, you need to prove both $L \subseteq L(G)$ and $L(G) \subseteq L$.
   *[Hint: Let $u_{\le i}$ denote the prefix of $u$ of length $i$. If $\Delta(u) = 1$, what can you say about the smallest $i$ for which $\Delta(u_{\le i}) = 1$? How does $u$ split up at that position? If $\Delta(u) = -1$, what can you say about the smallest $i$ such that $\Delta(u_{\le i}) = -1$?]*

   **Solution:** We separately prove $L \subseteq L(G)$ and $L(G) \subseteq L$ as follows:

   **Claim 1.** *$L(G) \subseteq L$, that is, every string in $L(G)$ has exactly twice as many 0s as 1s.*

   **Proof:** As suggested by the hint, for any string $u$, let $\Delta(u) = \#(\texttt{0},u) - 2\#(\texttt{1},u)$. We need to prove that $\Delta(w) = 0$ for every string $w \in L(G)$.

   Let $w$ be an arbitrary string in $L(G)$, and consider an arbitrary derivation of $w$ of length $k$. Assume that $\Delta(x) = 0$ for every string $x \in L(G)$ that can be derived with fewer than $k$ productions.[1] There are five cases to consider, depending on the first production in the derivation of $w$.

   - If $w = \varepsilon$, then $\#(\texttt{0},w) = \#(\texttt{1},w) = 0$ by definition, so $\Delta(w) = 0$.
   - Suppose the derivation begins $S \rightsquigarrow SS \rightsquigarrow^* w$. Then $w = xy$ for some strings $x, y \in L(G)$, each of which can be derived with fewer than $k$ productions. The inductive hypothesis implies $\Delta(x) = \Delta(y) = 0$. It immediately follows that $\Delta(w) = 0$.[2]
   - Suppose the derivation begins $S \rightsquigarrow \texttt{00}S\texttt{1} \rightsquigarrow^* w$. Then $w = \texttt{00}x\texttt{1}$ for some string $x \in L(G)$. The inductive hypothesis implies $\Delta(x) = 0$. It immediately follows that $\Delta(w) = 0$.
   - Suppose the derivation begins $S \rightsquigarrow \texttt{1}S\texttt{00} \rightsquigarrow^* w$. Then $w = \texttt{1}x\texttt{00}$ for some string $x \in L(G)$. The inductive hypothesis implies $\Delta(x) = 0$. It immediately follows that $\Delta(w) = 0$.
   - Suppose the derivation begins $S \rightsquigarrow \texttt{0}S\texttt{1}S\texttt{1} \rightsquigarrow^* w$. Then $w = \texttt{0}x\texttt{1}y\texttt{0}$ for some strings $x, y \in L(G)$. The inductive hypothesis implies $\Delta(x) = \Delta(y) = 0$. It immediately follows that $\Delta(w) = 0$.

   In all cases, we conclude that $\Delta(w) = 0$, as required. □

   **Claim 2.** *$L \subseteq L(G)$; that is, $G$ generates every binary string with exactly twice as many 0s as 1s.*

   **Proof:** As suggested by the hint, for any string $u$, let $\Delta(u) = \#(\texttt{0},u) - 2\#(\texttt{1},u)$. For any string $u$ and any integer $0 \le i \le |u|$, let $\boldsymbol{u_i}$ denote the $i$th symbol in $u$, and let $\boldsymbol{u_{\le i}}$ denote the prefix of $u$ of length $i$.

   Let $w$ be an arbitrary binary string with twice as many 0s as 1s. Assume that $G$ generates every binary string $x$ that is shorter than $w$ and has twice as many 0s as 1s. There are two cases to consider:

   - If $w = \varepsilon$, then $\varepsilon \in L(G)$ because of the production $S \to \varepsilon$.

[1] Alternatively: Consider the *shortest* derivation of $w$, and assume $\Delta(x) = 0$ for every string $x \in L(G)$ such that $|x| < |w|$.

[2] Alternatively: Suppose the *shortest* derivation of $w$ begins $S \rightsquigarrow SS \rightsquigarrow^* w$. Then $w = xy$ for some strings $x, y \in L(G)$. Neither $x$ or $y$ can be empty, because otherwise we could shorten the derivation of $w$. Thus, $x$ and $y$ are both shorter than $w$, so the induction hypothesis implies. . . . We need some way to deal with the decompositions $w = \varepsilon \bullet w$ and $w = w \bullet \varepsilon$, which are both consistent with the production $S \to SS$, without falling into an infinite loop.

- Suppose $w$ is non-empty. To simplify notation, let $\Delta_i = \Delta(w_{\le i})$ for every index $i$, and observe that $\Delta_0 = \Delta_{|w|} = 0$. There are several subcases to consider:
  - Suppose $\Delta_i = 0$ for some index $0 < i < |w|$. Then we can write $w = xy$, where $x$ and $y$ are non-empty strings with $\Delta(x) = \Delta(y) = 0$. The induction hypothesis implies that $x, y \in L(G)$, and thus the production rule $S \to SS$ implies that $w \in L(G)$.
  - Suppose $\Delta_i > 0$ for all $0 < i < |w|$. Then $w$ must begin with $\mathtt{00}$, since otherwise $\Delta_1 = -2$ or $\Delta_2 = -1$, and the last symbol in $w$ must be $\mathtt{1}$, since otherwise $\Delta_{|w|-1} = -1$. Thus, we can write $w = \mathtt{00}x\mathtt{1}$ for some binary string $x$. We easily observe that $\Delta(x) = 0$, so the induction hypothesis implies $x \in L(G)$, and thus the production rule $S \to \mathtt{00}S\mathtt{1}$ implies $w \in L(G)$.
  - Suppose $\Delta_i < 0$ for all $0 < i < |w|$. A symmetric argument to the previous case implies $w = \mathtt{1}x\mathtt{00}$ for some binary string $x$ with $\Delta(x) = 0$. The induction hypothesis implies $x \in L(G)$, and thus the production rule $S \to \mathtt{1}S\mathtt{00}$ implies $w \in L(G)$.
  - Finally, suppose none of the previous cases applies: $\Delta_i < 0$ and $\Delta_j > 0$ for some indices $i$ and $j$, but $\Delta_i \ne 0$ for all $0 < i < |w|$.

    Let $i$ be the smallest index such that $\Delta_i < 0$. Because $\Delta_j$ either increases by 1 or decreases by 2 when we increment $j$, for all indices $0 < j < |w|$, we must have $\Delta_j > 0$ if $j < i$ and $\Delta_j < 0$ if $j \ge i$.

    In other words, there is a *unique* index $i$ such that $\Delta_{i-1} > 0$ and $\Delta_i < 0$. In particular, we have $\Delta_1 > 0$ and $\Delta_{|w|-1} < 0$. Thus, we can write $w = \mathtt{0}x\mathtt{1}y\mathtt{0}$ for some binary strings $x$ and $y$, where $|\mathtt{0}x\mathtt{1}| = i$.

    We easily observe that $\Delta(x) = \Delta(y) = 0$, so the inductive hypothesis implies $x, y \in L(G)$, and thus the production rule $S \to \mathtt{0}S\mathtt{1}S\mathtt{0}$ implies $w \in L(G)$.

In all cases, we conclude that $G$ generates $w$. □

Together, Claim 1 and Claim 2 imply $L = L(G)$. ■

**Rubric:** 10 points:
- part (a) = 4 points. As usual, this is not the only correct grammar.
- part (b) = 6 points = 3 points for $\subseteq$ + 3 points for $\supseteq$, each using the standard induction template (scaled).