Show that the context-free languages are closed under operation $\text{mit}$, defined as follows:

For a language $L$ over $\Sigma$:

$$\text{mit} (L) = \{ w | w \cdot x \in L \text{ for some } x \in \Sigma^* \}$$

**Proof:**

Consider a CFL $L$ and a CFG $G = (V_N, V_T, P, S)$ in Chomsky normal form without useless symbols such that $L(G) = L$.

We want to construct a CFG $G_{\text{mit}}$ for $\text{mit} (L)$.

For a non-terminal $A$, let $G_A = (V_T, V_N, P, A)$ the grammar identical to $G$, but with $A$ as start-symbol.

Let $L_A = L(G_A)$.

Idea: for each NT $A$, we introduce a new NT $\overline{A}$ that generates $\text{mit} (L_A)$.

We construct $G_{\text{mit}} = (V_N^{\text{mit}}, V_T, P^{\text{mit}}, S^{\text{mit}})$, where

V_N^{\text{mit}} = V_N \cup \{ \overline{A} \mid A \in V_N \}

S^{\text{mit}} = S

and $P^{\text{mit}}$ defined as follows:

- for every production $A \rightarrow BC$ in $P$, we have in $P^{\text{mit}}$:
  $$A \rightarrow BC \quad \overline{A} \rightarrow BC \mid \overline{B}$$

- for every production $A \rightarrow a$ in $P$, we have in $P^{\text{mit}}$:
  $$A \rightarrow a \quad \overline{A} \rightarrow \overline{a} \mid \varepsilon$$
We prove that $L(G_{\text{in}L}) = \text{init}(L)\)  

set $w \in L(G_{\text{in}L})$. We show that $w \notin \text{init}(L_A)$, i.e. there is $x \in \Sigma^*$ s.t. $w \cdot x \notin L_A = L(G_A)$. 

We proceed by induction on the depth $m$ of the derivation tree for $w$.

**case 1** $A$ and $w \cdot x.2) \quad \overline{A} \quad \text{end}$

**case 2** $w \in L_A \leq \text{init}(L_A)$

**inductive case**: assume that if there is a derivation tree of depth $\leq m$ showing $w \in L(G_{\text{in}L})$, we have that $w \notin \text{init}(L_A)$.

Consider a derivation tree of depth $m+1$.

**case 1** $\quad A$

\begin{align*}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
B \quad \overline{C}
\end{array}
\end{array}
\end{array}
\end{align*}

$w \in L_A \leq \text{init}(L_A)$

\begin{align*}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\overline{w_1} \quad \overline{w_2}
\end{array}
\end{array}
\end{array}
\end{align*}

**case 2** $\quad \overline{A}$

\begin{align*}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
B \quad \overline{C}
\end{array}
\end{array}
\end{array}
\end{align*}

\begin{align*}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\overline{w_1} \quad \overline{w_2}
\end{array}
\end{array}
\end{array}
\end{align*}

by IH, $w_2 \in \text{init}(L_C)$, i.e.

there is $x \in \Sigma^*$ s.t. $w_2 \cdot x \notin L_C$, i.e. $C \Rightarrow^* x \cdot \overline{w_2}$. 

since $A \Rightarrow BC$ is in \text{init}$,$

we have that $A \Rightarrow BC$ is in \text{init}$.

since $B \Rightarrow^* w_1$, we have that $A \Rightarrow w_1 \cdot w_2 \cdot \overline{w_2} \cdot x$. 

Hence $w \in \text{init}(L_A)$.
(case 3) \[ \overline{A} \quad \overline{B} \quad w \quad \] 

by IH., \( w \in \text{init}(L_{\overline{B}}) \), i.e.

there is \( x \in \Sigma^* \) s.t. \( w.x \in L_{\overline{B}} \),

i.e. \( B \xrightarrow{x} w.x \).

Since \( \overline{A} \rightarrow \overline{B} \) is in \( \Pi \), there is some \( C \) s.t. \( A \rightarrow BC \) is in \( \Pi \).

Since \( G \) contains no useless symbols, \( C \) is generating, i.e. \( C \xrightarrow{*} y \) for some \( y \).

Hence \( A \rightarrow B.C \xrightarrow{*} w.x.y \), and \( w \in \text{init}(L_A) \).

"2" Let \( w \in \text{init}(L) \). Then there is \( x \in \Sigma^* \) s.t. \( w.x \in L(A) \).
Exercise (7.3.4 from textbook), 7/12/2005

Show that the context-free languages are closed under operation \( \text{init} \), defined as follows:

given a language \( L \),

\[
\text{init}(L) = \{ w | \text{ for some } x, wx \in L \}
\]

Solution

Consider a CFL \( L \), and a grammar \( G \) for \( L \).

For each nonterminal \( A \) in \( G \), we want to have an additional nonterminal \( \overline{A} \) that generates \( \text{init}(L_A) \), where \( L_A \) is the language generated by \( G \) having \( A \) as axiom (start symbol).

The goal is to construct a grammar (a CFG) that generates \( \text{init}(L) \). The start symbol of \( G \) is \( \overline{S} \).

Without loss of generality, we can assume that \( G \) is in Chomsky normal form.

For every production \( A \rightarrow BC \) in \( G \), in \( \overline{G} \) we have:

\[
\begin{align*}
\overline{A} & \rightarrow \overline{B}C \\
A & \rightarrow BC
\end{align*}
\]

end for every production \( A \rightarrow a \) in \( G \), in \( \overline{G} \) we have

\[
\begin{align*}
A & \rightarrow a \\
\overline{A} & \rightarrow \varepsilon
\end{align*}
\]

The obtained grammar (which is not in Chomsky normal form) generates \( \text{init}(L) \) — formal proof left to the reader.
Show that CNF languages are not closed under \textit{min}, defined as follows:

\[
\text{min}(L) = \{ W \mid W \in L \text{ but no proper prefix of } W \text{ is in } L \}
\]

\textbf{Solution}

The proof goes through a counterexample that shows that the closure does not hold.

Consider the CFL

\[
L = \{ a^i b^i c^k \mid k \geq i \} \cup \{ a^i b^i c^i \mid k > j \}
\]

$L$ is clearly a CFL, since it is the union of two languages for which we can straightforwardly write a CFG.

Now, notice that

\[
\text{min}(L) = \{ a^i b^i c^k \mid k = \text{min}(i,j) \}
\]

This language is not a CFL, and this can be proved by using the pumping lemma. Let $m$ be the pumping lemma constant; consider $w = a^m b^m c^m \in \text{min}(L)$. 

Give a reduction from the halting problem to the following problem:

given a program $P$ and an input $I$, does $P$
eventually halt when it is given $I$ as input?

**Solution**

We take $P$ and modify it in the following way:

(1) We make sure it never halts unless we explicitly want it. This is done by inserting some instruction like

```c
while(1) {};
```

at the end of `main()` and where there is a `return` in `main()`. 

(2) We make $P$ record the first 12 characters printed, if they are "hello, world" the program halts.

In this way, the modified program halts if and only if the original program $P$ prints "hello, world". This ends our reduction: in fact, if we are able to decide whether a program eventually halts, then we are able to decide whether it prints "hello, world".
Exercise (Example 8.2 from textbook) 04/02/005

Construct a Turing Machine accepting the language

\[ \{0^n1^n \mid n \geq 1\} \]

Solution

The idea is that the TM M that we construct needs the leftmost 0, turns it into X, and moves right until it reaches a 1, that is turned into X. Then the head moves left again to the leftmost 0 (on the right to a X), and starts again until all 0's and 1's are turned into X's and Y's respectively.

If the input is not in 0^*1^*, M will fail to find a move and it won't accept. If M changes the last 0 and the last 1 in the same round, it will go into the final state and accept.

\[ Q = \{ q_0, q_1, q_2, q_3, q_4 \} \]
\[ \Sigma = \{ 0, 1 \} \]
\[ \Gamma = \{ 0, 1, X, Y, \text{ blank} \} \]
\[ q_0 : \text{ start state} \]
\[ F = \{ q_4 \} \]

In \( q_0 \), the state in which M moves left before the head meets the leftmost 0. In state \( q_1 \), M moves right skipping 0's and Y's until it gets to a 1. In state \( q_2 \), M moves left while skipping Y's and 0's again, until it gets to a X and goes again in \( q_0 \).
Starting from $q_0$, if a $Y$ is read instead of a $0$, $M$ goes in $q_3$ and moves right: if a $1$ is found, then there are more $1's$ than $0's$; if a $0$ is read, then the initial string is accepted (transition to $q_4$).

<table>
<thead>
<tr>
<th></th>
<th>$0$</th>
<th>$1$</th>
<th>$X$</th>
<th>$Y$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$(q_1, X, R)$</td>
<td>—</td>
<td>$(q_3, Y, R)$</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$(q_2, 0, R)$</td>
<td>$(q_2, Y, L)$</td>
<td>—</td>
<td>$(q_3, Y, R)$</td>
<td>—</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$(q_2, 0, L)$</td>
<td>—</td>
<td>$(q_0, X, R)$</td>
<td>$(q_2, Y, L)$</td>
<td>—</td>
</tr>
<tr>
<td>$q_3$</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>$(q_3, Y, R)$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Exercise**
Show the computation of the TM above when the input string is:
(a) 00
(b) 000111

**Solution**
(a) $q_0 00 \rightarrow Xq_10 \rightarrow X0q_4$
and the TM halts
(b) $q_0 000111 \rightarrow Xq_40111 \rightarrow X0q_40111$
$X00q_4111 \rightarrow X0q_40111 \rightarrow Xq_200111 \rightarrow q_2X00111$
$Xq_00Y11 \rightarrow XXq_20Y11 \rightarrow XX0q_2Y11 \rightarrow XX0q_4111$
$XX0q_20Y11 \rightarrow XXq_200Y11 \rightarrow q_2X000Y11 \rightarrow q_2000Y11$
$XXXq_1YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1$
$XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1$
$XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1$
$XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1$
$XXXq_2YY1 \rightarrow XXXq_2YY1 \rightarrow XXXq_2YY1$
Exercise (8.22 from textbook)

Design Turing machines accepting the following languages:

\[ \{ w \in \{0,1\}^* \mid \text{w has an equal number of 0's and 1's} \} \]

solution

The idea is that the head of our TM M moves back and forth on the tape, "deleting" one 0 for each 1; if there are no 0's and 1's in the end, the string is accepted.

When in state \( q_2 \), M has found a 1 and looks for a 0; in state \( q_2 \) is the other way around.

Note that the head never moves left of any X, so that there are never unmatched 0's and 1's on the left of an X.

From initial state \( q_0 \), M picks up a 0 or a 1 and turns it into X. The only final state is \( q_4 \). In state \( q_3 \), M moves head left looking for the rightmost X.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>( \text{blank} )</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( q_2, X, R )</td>
<td>( q_1, X, R )</td>
<td>( q_3, 5, R )</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_3, Y, L )</td>
<td>( q_2, 1, R )</td>
<td></td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_3, 0, R )</td>
<td>( q_3, Y, L )</td>
<td></td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_3, 0, L )</td>
<td>( q_3, 0, L )</td>
<td>( q_0, X, R )</td>
<td>( q_3, Y, L )</td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exercise (8.11 from textbook)

Give a reduction from the hello-world problem to the following problem:

given a program $P$ and an input $I$, does the program ever produce any output?

solution

We modify $P$ by making it print its output on some array $A$, capable of storing 12 characters. When $A$ is full, $P$ checks whether it stores "hello world": if it does, $P$ prints (on the output, not on the array) some character (like @); if not, it does not print anything.

So the modified program prints some output if and only if $P$ prints "hello world": if we are able to determine whether a program produces any output, we can solve the hello-world problem.

This ends our reduction.