0) Show that the problem "Given is a CFG $G$, is $L(G) \subseteq \{1\}^*$?" is decidable.

1) First we find all "terminable" variables, i.e. variables $A$ for which $\exists x \in \Sigma^*$ s.t. $A \Rightarrow^* x$. We had an algorithm for this in class.

2) Remove all rules that use a non-terminable variable.

3) We will recursively find the set of variables $A$ s.t. we can derive a string $x$ containing a symbol from $\Sigma - \{1\}$ from $A$, i.e. $A \Rightarrow^* x$.
   We name this set $A$.

   1. for every rule $A \rightarrow \alpha$ where $\alpha$ contains a symbol $e \in \Sigma - \{1\}$,
      let $A \in A$
   2. for every rule $A \rightarrow \alpha$ where $\alpha$ contains a variable $e \in A$,
      let $A \in A$
   3. no other variable is in $A$

4) If $S \in A$ then answer NO
   else answer YES.

EXAMPLE:

$S \rightarrow B|D|E$
$B \rightarrow 1B|\Lambda$
$C \rightarrow 1B|2D$
$D \rightarrow SB|1C$
$E \rightarrow ES|BE$

Terminable: $A, B, C, D$ (remove rules with $E$)

$A = C, D, S$

$NO_1, L(G) \notin \{1\}^*$
2) \( \text{ALL}_{\text{CFG}} : \) given a CFG \( G \), is \( L(G) = \Sigma^* \) ?

Is \( \text{ALL}_{\text{CFG}} \) is RE, coRE, neither, or both?

- It cannot be in both since if it were, it would have been recursive and thus decidable.

- It is in coRE. To see this, let’s look at the complement of \( \text{ALL}_{\text{CFG}} \). Before we do this, let’s write \( \text{ALL}_{\text{CFG}} \) as a language.

\[
\text{ALL}_{\text{CFG}} = \{ x \mid x \text{ encodes a CFG } G \text{ over } \Sigma \text{ and } L(G) = \Sigma^* \}
\]

Then,

\[
\text{ALL}_{\text{CFG}}' \text{ contains all CFGs } G \text{ s.t. there exists a string } x \in L(G) \text{ (plus, technically speaking, it contains all strings that do not encode a context-free grammar)}
\]

Here is a sketch of a TM for \( \text{ALL}_{\text{CFG}}' \):

1) verify that input encodes a CFG – if no, accept

2) if yes, then nondeterministically guess \( x \) and verify that \( x \in L(G) \) – use the algorithm from class or CYK

3) if \( x \in L(G) \), accept

Note: this sketch describes a nondet. TM but we know that an NTM can be converted to a TM.

- Intuition why not in RE: we would need to verify all \( x \) but there is an infinite \( \# \) of strings so this algorithm would not finish.
PDAeqFA: given a PDA \( M_1 \) and an FA \( M_2 \), is \( L(M_1) = L(M_2) \)?

Show that PDAeqFA is undecidable:

Suppose PDAeqFA is decidable, then we have an algorithm PeF that takes a PDA \( M_1 \), and an FA \( M_2 \) as inputs and it returns:

\[
\text{PeF}(\text{PDA } M_1, \text{FA } M_2) = \begin{cases} 
\text{YES} & \text{if } L(M_1) = L(M_2) \\
\text{NO} & \text{otherwise}
\end{cases}
\]

Then we can write the following algorithm for ALLcFG that uses PeF as a subroutine.

\[
\text{Everything (CFG } G) \{
\begin{align*}
1. & \quad \text{use the algo from class to construct a PDA } M_1 \text{ s.t. } L(M_1) = L(G) \\
2. & \quad \text{let } M_2 \text{ be an FA accepting } \Sigma^*: \quad \begin{array}{c}
\epsilon \\
\forall \sigma \in \Sigma
\end{array} \\
3. & \quad \text{return } \text{PeF}(M_1, M_2);
\end{align*}
\}
\]

Thus, if PDAeqFA is decidable, then we could decide ALLcFG. But we know that ALLcFG is undecidable. A contradiction.

Hence, PDAeqFA cannot be decidable.