Recall algorithm for nullable vars using BFS

bool changed = true;
while (changed) {
    changed = false;
    for each B in V \ N:
        if (B has a rule B \rightarrow x with x \in N^*):
            N = N \cup {B}
    changed = true;
}

By inductive analysis, all variables added to N derive terminal strings.

Hence the EPS_{CFG} and NE_{CFG} problems are decidable (for later: indeed in \text{poly}(|G|) time).

How about "Empty Intersection" problems?

EI: Inst M_1, M_2
Ques: Is L(M_1) \cap L(M_2) = \emptyset?

INE_{CFG}: Inst G_1, G_2
Ques: Is \text{LHilb}_1 \cap \text{LHilb}_2 \neq \emptyset?
**DFA:** Given DFA, M, & M₂, is \( L(M_1) \cap L(M_2) \neq \emptyset \)

Previous lecture solved \( L(M) \neq \emptyset \) for one DFA by BFS.

Algorithm:
1. Use Cartesian product construction for \( \Delta \) to make \( M_3 \) s.t. \( L(M_3) = L(M_1) \cap L(M_2) \)
2. Run the NE-PDA decider on \( M_3 \)
3. Answer 'yes' if and only if the NE-PDA decider answers 'yes'.

Correct because \( L(M_3) \neq \emptyset \) \& \( L(M_1) \cap L(M_2) = \emptyset \).

**OPDA:** Given OPDA, M₁, & M₂, is \( L(M_1) \cap L(M_2) \neq \emptyset \)

Can we do a similar Cartesian Product idea? Issue: We can combine states but \( M_1 \) and \( M_2 \) will fight over one stack.

Cannot work because \( L(M_1) \cap L(M_2) \) might not even be a CFL.

Note: The complement of a DCFL is a DCFL because a OPDA can be made total. Hence DCFL is not closed under \( \cup \) either, though the union of two DCFLs is a CFL.

The "other shoe to drop?" is we will see that the problem INE OPDA and (hence) INE CFG are undecidable!
How about \( A_{CFG} \): \textbf{INST: A CFG \( G = (V, \Sigma, R, s) \), an \( x \in \Sigma^* \)}, \textbf{QUES: } \exists x \in L(G), \text{i.e. does } S \Rightarrow^* x? \\

Doing brute-force search on derivations may be open-ended when variables are nullable, which will

**Decider:** 1. First convert \( G \) to \( G' \) in ChNF. 
   \( n = 1 \times 1 \)
   IF \( x \notin \Sigma \), USE \( EPS \) in \( EPS \) in \text{CFG} decider.
   IF \( x \in \Sigma \), USE \( EPS \) in \( EPS \) in \text{CFG} decider.

   2. Then if \( G' \) derives \( x \), its \( S' \) can derive \( x \) in exactly \( 2^{n-1} \) steps or not at all.

   Hence brute force to depth \( 2^{n-1} \) is a decider.

**FYI (last week)**

Let \( m = 1 \times 1 \)

Instance size of \( L(G, x) \) is \( \times m \times n \).

\( A_{CFG} \) is in \( \mathcal{P} \).

**Diagonalization**

Clean, no infinities!

\textbf{INST: A CFG \( G \) encoded as } \( \langle G \rangle \)

\textbf{QUES: } \exists \langle G \rangle \in L(G), \text{i.e. does } S \Rightarrow^* x? \text{ over the } \Sigma_{CFG}

**Fact 1:** \( D_{CFG} \) flip-reduces to \( A_{CFG} \).

\[ \langle G \rangle \in D_{CFG} \iff \langle G \rangle \in L(G) \iff \langle G, G \rangle \in \mathcal{A}_{CFG} \]

Hence \( D_{CFG} \) is decidable. But it cannot be a CFL.
Theorem: There is no encoding scheme \( \langle G \rangle \) for (F6s G) that makes D\(_F\) into a CFL.

Proof: Suppose there were a "Quixotic" (F6 Q st. \( L(Q) = D\_F \langle G \rangle \)) (referring the encoding scheme \( \langle \cdots \rangle \)). Then we could take \( q = \langle Q \rangle \). Let us ask whether \( q \in D\_F \):

\[
q \in D\_F \iff q \in L(Q) \quad \text{by} \quad D\_F = \{ \langle Q \rangle : L(Q) \neq \emptyset \}
\]

\[
q \in L(Q) \quad \text{because by assumption,} \quad L(Q) \neq \emptyset
\]

A statement can never be equivalent to its negation. So this contradiction "rolls back" the assertion that Q exists. So D\(_F\) has no CF-F(A), so D\(_F\) is not a CFL.

Now define D\(_T\) = \{ \langle M \rangle : M is a DTM and \( L(M) \neq \emptyset \) \}

Theorem: D\(_T\) is undecidable, indeed not even C.E.

Proof: Suppose there were a TM Q st. \( L(Q) \neq \emptyset \).

Then we could take \( q = \langle Q \rangle \), so \( q \in \Sigma^* \).

Then \( q \in D\_T \iff q \in L(Q) \quad \text{by} \quad D\_T = \{ q : q \in \Sigma^* \}
contradiction \quad \Leftrightarrow \quad q \in L(Q) \quad \text{by} \quad L(Q) = \emptyset \).

Hence a TM Q for D\(_T\) cannot exist, so D\(_T\) is not Turing acceptible, hence not decidable either.