Suppose we have a "Target Specification" $T$ of a language $T$ defined by prose, sets, or machine, etc. Suppose we have a CFG $G$ that is trying to "model" $T$.

$G$ is sound for the spec if $L(G) \subseteq T$.

$G$ is comprehensive for it if $T \subseteq L(G)$.

These concepts were first formalized in logic where $G$ is generalized to a "Formal System $F$" - think of it like a grammar where a string is generated by 2 or more not just 1. $T$ = the set of true statements. Sound means $L(F) \subseteq T$, $L(F)$ = the set of theorems of $F$. i.e. "every theorem provable".

Comprehensiveness would mean $T \subseteq L(F)$, i.e. that $F$ could prove every true statement (over a particular logical alphabet) sound and

But, Kurt Gödel proved that no executable formal system can be comprehensive for $T = \exists$ true unprovable.

I.e. for any sound and effective $F$ over the "alphabet of unprovable"

$L(F) \not\subseteq T$. Gödel's Incompleteness Theorem Uncomprehensiveness.
We will think of the concept most with CFGs and apply them even when T is given by another grammar. If we change an original grammar G into G₂, then

- the change is sound if \( L(G) \subseteq L(G₂) \);
- But of course we want \( L(G₂) = L(G) \) comprehensively too.

**Defn:** A CFG \( G₂ \) is in Chomsky Normal Form (ChNF) if every rule \( A \rightarrow X \) either has \( X \in \Sigma \) or \( X \in \mathbf{N} \).

Our text enables "ChNF" grammars to generate \( \varepsilon \) by the special exception that we can add an extra start symbol \( S₀ \) and rules \( S₀ \rightarrow \varepsilon \) and \( S₀ \rightarrow \varepsilon \) ... all right-hand side \( \varepsilon S \).

**Defn:** A variable \( A \in \mathbf{V} \) is nullable if \( A \Rightarrow^* \varepsilon \).

Note \( \varepsilon \in L(G) \iff S₀ \) is nullable.

**Theorem:** Given any CFG \( G \), we can build a CFG \( G₄ \) in ChNF s.t. \( L(G₄) = L(G) \setminus \{\varepsilon\} \) if we regard ChNF strictly \( = L(G) \) if \( \varepsilon \in L(G) \) and we allow the "\( S₀ \)" judicious above.

**Step 1:** will build \( G₄ \) s.t. \( L(G₄) = L(G) \setminus \{\varepsilon\} \) and \( G₄ \) has no nullable variables.
Algorithm and Proof of Step 1: Given $G = (V, \Sigma, R, S)$

1. First identify the subset $\text{NULLABLE} \subseteq V$ of nullable variables.

2. For every rule $A \rightarrow \bar{X}$ where $\bar{X}$ has nullable variables, add the rules $A \rightarrow \bar{X}'$ for all combinations of deleting one or more occurrences of the nullable variables in $\bar{X}$.

3. Delete all $\epsilon$-rules $B \rightarrow \epsilon$, incl. any new ones.

We will show this comprehensively except for $\epsilon$ itself.

Proof of subset (iii): Let any $y \neq \epsilon$ in $L(G)$ be given. Then we can take some parse tree $T$ for $y$ in the original $G$.

$G$ has rule $A \rightarrow CDB$ say

$G_i$ also has $A \rightarrow CD$

Pinching out any subtree of $T$ that yields $\epsilon$ inside $y$ leaves a valid parse tree in the new $G$.

That $y \neq \epsilon$ means we don't allow all of $T$.

$y = \ldots$
Algorithm for telling which vars are NULLABLE.

1. Initialize NULL = \{ A \in V : A \rightarrow \epsilon \text{ is a rule} \}.

2. \text{bool changed = true}

3. While (changed)
   \text{changed = false;}
   \text{for (each rule } A \rightarrow X \text{ in } R \text{) }
   \text{if (} X \in (\text{NULL})^* \text{ and } A \notin \text{ NULL) }
   \text{NULL = NULL } \cup \{ A \}
   \text{changed = true;}

4. Output final NULL.

Two Examples:

\[ G = S \rightarrow AB \mid CA \mid \epsilon \]
\[ A \rightarrow SS \mid Sa \]
\[ B \rightarrow AS \mid c \]

\[ S \rightarrow \epsilon \mid (S) \mid SS \text{ NULLABLE = SS} \]

\[ G_1 = S \rightarrow (\epsilon) \mid (S) \mid SS \]
\[ G_1 \text{ generates all nonempty balanced } (S) \]

Target: NULL = NULLABLE

Sound because if we add A to NULL, then A \Rightarrow X \in \text{ NULL}.

Comprehensive \rightarrow \text{ think about it}.