1. Eliminate ε-rules, as before. The \( G' \) may have more "unit rules" \( A \rightarrow B \).
2. Form \( R^*(A, B) = \{ A \rightarrow B \mid A, B \in V, \text{ which is the transitive closure of the unit-rule relation } R(A, B) \} \). "A \rightarrow B is a rule in \( G' \)."
3. For all cases of \( A \rightarrow B \), add the non-unit \( RHS \) sides of \( B \) as rules for \( A \).
4. Then it is safe to delete all unit rules.

For "long rules" with \( |RHS| \geq 3 \), use new "Y" variables to break them up two-at-a-time.

\( G: \)
\[
S \rightarrow SS \mid ASX_a \mid X_a A \quad L(G') = L(G).
\]
\[
A \rightarrow \times_A X_a A, \quad X_a \rightarrow a, \quad X_a \rightarrow b.
\]

\( G'': \)
\[
S \rightarrow SS \mid AY \mid X_a A, \quad Y \rightarrow SX_a \quad \text{in Chomsky NF}: \quad L(G'') = L(G).
\]
\[
A \rightarrow \times_X X_a A, \quad X_a \rightarrow a, \quad X_a \rightarrow b \quad \text{(no change with } \varepsilon \text{ since } \varepsilon \notin L(G)).
\]

Example of Unit Rules:

\( S \rightarrow AS \mid SC \mid a \)
\( A \rightarrow AB \mid BC \mid a \)
\( B \rightarrow C \mid AS \mid b \)
\( C \rightarrow SS \mid \varepsilon \)

S & NULLABLE, so \( \varepsilon \in L(G) \).

\( G: \)
\[
S \rightarrow AS \mid SC \mid a \quad \text{0. } C \text{ is nullable. } C \rightarrow \varepsilon
\]
\[
B \rightarrow C, \text{ so } B \rightarrow \varepsilon
\]
\[
A \rightarrow BC \text{ with } B, C \rightarrow \varepsilon
\]
\[
\Rightarrow A \rightarrow \varepsilon \text{, so } A \text{ is nullable too.}
\]
\[
\text{NULLABLE} = \{ A \in V : A \rightarrow \varepsilon \} = \{ A, B, C \}
\]

No point in adding \( S \rightarrow S \), few we get:

\( G': \)
\[
A \rightarrow AB \mid BC \mid a \mid BC \mid R(A, B), R(A, C), R(B, C), R(B, S)
\]
\[
B \rightarrow C \mid S \mid b \mid AS
\]
\[
C \rightarrow SS \quad \text{delete } C \rightarrow \varepsilon. \text{ } B \rightarrow AS \mid SC \mid a \mid b \mid SS
\]

\( G'': \) same; \( A \rightarrow AS \mid SC \mid a \mid AB \mid BC \mid b \mid SS \) \( G'' \) is in \( \text{ChNF} \), \( L(G'') = L(G) \).
Features of ChNF:

- Each non-terminal rule has \(|RHS| = 2\) linking.
- All leftmost derivations have steps belonging to \(\Sigma^* \cdot V^*\).

**Step:** \(X_1 \cdot X_2 \cdot X_3 \cdots \cdot X_{i-1} \cdot \overbrace{\bigwedge\cdots\bigwedge}^{c \in \Sigma} \cdot V_2 \cdots \cdot V_k \in V\)

(SENTENTIAL FORM)

Suppose \(V_1 \rightarrow AC \cdot b\)

\(\equiv\) Tape plus stack:

\[X: \quad \overbrace{X_1 \cdot Y_2 \cdot X_3 \cdots \cdot X_{i-1} \cdot X_j}^c \cdot \overbrace{V_2 \cdot V_3 \cdots \cdot V_k}^{all} \]

In the last step we got \(C = X_2\) by applying the terminal rule \(V_0 \rightarrow C\).

Then if \(X_{i+1} = a\) and \(A \rightarrow a\) is a rule, then we could pop \(A\) and process \(X_{i+1} = a\).

If \(X_{i+1} = b\), then we can pop \(V_1\) and process the \(b\).

If \(X_{i+1} = a\), or even if it is equal \(b\), we also have the option of: push \(C\), push \(A\), don't process \(X_{i+1}\).

Text in §2.2 expands this into a formal definition of a (nondeterministic!) Pushdown Automaton (PDA) \(P\), and proves Theorem: A language \(L\) is context-free iff there is a PDA \(P\) such that \(L = L(P)\), such that for any \(p \geq L(P)\).

ChNF (used C, S, \(\delta\), \(\tau\), \(\gamma\))

**CFL Pumping Lemma:** For any CFL \(L\), there is a number \(p \geq 0\) such that every \(X \in L\) with \(|X| \geq p\) can be broken as \(X =: YUVWZ\) s.t.:

1. \(|UVW| \leq p\)
2. The "pumpable parts" \(u\) and \(w\) are not both \(\epsilon\), i.e. \(|u \cdot w| > 0\).
3. For all \(i \geq 0\), \(x(i) = Y \cdot u^i \cdot v \cdot W^i \cdot Z\) belongs to \(L\).
4. In particular, \(x(0) = Y \cdot \overbrace{\varepsilon}^0 = \varepsilon \in L\). (["pumping down"])
Proof: For the given CFL L, it doesn't matter whether \( E \in L \) or not. So let us take a CFG \( G \) in ChNF such that \( L(G) = L \setminus \{ \varepsilon \} \).

Take \( N \) to be the number of variables in \( V \). Take \( p = 2^N + 1 \) (can be \( \leq p \)).

Take \( x \in L(G) \), \( 1 \leq p \), and build the binary parse tree. Every binary tree with \( 2^N + 1 \) or more nodes has a path including \( N+1 \) or more nodes.

\[ N = 3 \]
\[ 2^N + 1 = 9 \]

\( 2N \) is enough.

For any path of \( N+1 \) or more nodes, e.g. \( V = \{ S, A, B, C \} \), \( N = 4 \) some variable A must appear twice along the path. (Could have \( A \rightarrow \varepsilon \)). Pigeonhole Principle again!

In particular, some repeated occurrence must happen among the bottom \( N+1 \) nodes of that path.

Divide the parse tree of the string \( x \) of length \( \geq p \) according to the deeper node that has lower occurrence of that variable.

\[ x = u v w z \] is the yield of the upper one.

Then:

- \[ |u v w| \leq p \] because the upper occurrence is in the bottom \( N+1 \) nodes of the path.
- Here we have a case of \( w = \varepsilon \) but \( u \neq \varepsilon \). We can't have both \( w = \varepsilon \) because the other branch of the upper variable has to derive something nonempty.
- We can edit the parse tree legally either:
  - Deriving the upper variable the same way the other one was. Then the upper yields \( v \) not \( u v w \), yielding \( Y \).
  - Iterate deriving the lower \( A \) the same as the upper \( A \).

For all \( i \geq 0 \), \( u^i v w^i z \) has a parse tree, so \( E \in L(G) \).
Contrapositive of CFL Pumping Lemma

If for all \( p > 0 \)

there exists a string \( x \in L \) with \( |x| \geq p \) such that

for all breakdowns \( x = \gamma \cdot u \cdot v \cdot w \cdot z \) with \( |uvw| \leq p \) and \( u \neq \epsilon \)

there exists \( i \geq 0 \) such that

\[ x(i) = u \cdot \gamma \cdot (v \cdot w)^i \cdot z \] does not belong to \( L \).

THEN \( L \) is not a CFL!