Recall:

\[ S \rightarrow \varepsilon | AB | BA | SS \]

\[ L = \{ x : \#a(x) = \#b(x) \} \]

\[ A \rightarrow a1as | BAA \]

\[ L_A = \{ x : \#a(x) = \#b(x) + 1 \} \]

\[ B \rightarrow b | b5 | ABB \]

\[ L_B = \{ x : \#a(x) = \#b(x) - 1 \} \]

For \( L(\varepsilon) \subseteq L \) (soundness) define \( \mathcal{P}_S, \mathcal{P}_A, \mathcal{P}_B \), where for instance:

\[ \mathcal{P}_A = \{ \text{Every string } x \text{ that I derive has } \#a(x) = \#b(x) + 1 \} \]

For \( L \subseteq L(\varepsilon) \) (comprehensiveness) we do the opposite:

\[ \mathcal{P}(n) = \mathcal{P}(n) \]

where \( \mathcal{P}(n) \equiv \{ \text{For each } x \in \Sigma^n, \text{ if } x \notin L_A \text{ then } A \not\Rightarrow^* x \} \)

which kind of says: "I derive every string \( x \) that has \( \#a(x) = \#b(x) + 1 \).

Proving \( \equiv \) Proving an algorithm for parsing. E.g. Given \( X = babaabb \).

- How does \( X \) get derived?

\[ S \rightarrow e | AB | BA | SS \leftarrow \text{unnecessary} \]

- Is \( x \in L_S, L_A, L_B, \text{ or none of the above?} \)

\[ A \rightarrow a1as | BAA \]

\[ B \rightarrow b | b5 | ABB \]

\[ x \in L_A \Rightarrow \{ \text{Proof does subcases for the first letter in } x \} \]

- \( \Rightarrow \forall x \in L_S \Rightarrow A \Rightarrow (b^n \Rightarrow aas) \Rightarrow ax = x \).

- Hence we can argue \( X \in \text{ both } \)

- Where \( |b| = 1 \), i.e. \( \text{diff}(b) = 0 \)

\[ \text{diff} \begin{bmatrix} 0 & -1 & 0 & 0 \end{bmatrix} \]

\[ X = babaabb \Rightarrow A \Rightarrow BAA \Rightarrow bAA \Rightarrow bA \Rightarrow b \]

\[ \Rightarrow \text{ba} \]

\[ \text{diff} \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \]

\[ \text{diff} \begin{bmatrix} 0 & -1 & 0 & 0 \end{bmatrix} \]

\[ \text{diff} \begin{bmatrix} 0 & -1 & 0 & 0 \end{bmatrix} \]

\[ \text{diff} \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \]
Using the "least j" rule, breaks babaaab as

\[ b \cdot a \cdot b \cdot a \cdot a \cdot a \cdot b \]

Refined grammar:

\[ S \rightarrow e \quad e \quad a \quad B \quad b \quad A \]

\[ A \rightarrow a \quad s \quad b \quad a \quad A \]

\[ A \rightarrow b \quad a \quad A \quad A \]

\[ B \quad A \quad A \quad A \]

This grammar \( G_2 \) remains comprehensive:

\[ L \subseteq L(G_2) \]

Unfortunately, \( G_2 \) is still ambiguous, as the case \( X = \text{babaaab} \) showed.

Regular Expression Parsing:

\[ b \cdot v \cdot a \left( a \cdot b \cdot a \right)^* a + a \cdot b \]

\[ b + a \cdot \left( a \cdot b \cdot a \right)^* a + a \cdot b \]

Fully parenthesized:

\[ E \rightarrow \emptyset \epsilon \left[ a \cdot b \right] \left( b + a \right) \left( b \cdot b \right) \left( b^* \right) \]

Thus, the regular-to-\( \epsilon \)-NFA proof became one by structural induction.
Chomsky Normal Form: Makes the proof of the CFL Pumping Lemma 62.3 nicer.

Def.: A CFG $G = (V, \Sigma, R, S)$ is in Chomsky Normal Form (ChNF) if every rule $A \rightarrow X$ has $X \in \Sigma$ or

Allowed: $A \rightarrow \varepsilon$, $A \rightarrow AA$, $A \rightarrow BC$  $\mid \text{LX}\mid \leq 2$

Not Allowed: $A \rightarrow \varepsilon$, $A \rightarrow B A \rightarrow BCD$ with $X \in V$.

3-rule: mixed rhs unit rule \| RHS \| \geq 3

(The text allows $S \rightarrow \varepsilon$ provided $S$ is not on the RHS of any rule.)

Can always put $\varepsilon$ in last by a new start var. So, and rules $S \rightarrow \varepsilon$.

Theorem: Given any CFG $G$, we can build $G'$ in ChNF st. $L(G') = L(G)$.

Proof: By Algorithm

1. Identify the set NULLABLE of variables $A$ st. $A \Rightarrow^* \varepsilon$.

2. For every rule $A \rightarrow X$ include every rule $A \rightarrow X'$ obtained by erasing 1 or more occurrences of vars in NULLABLE.

3. Then delete all $\varepsilon$-rules. Get $G_2$ st. $L(G_2) = L(G) \setminus \{\varepsilon\}$. Informal (or complete)

4. Determine which $A, B \in V$ allow $A \Rightarrow^* B$ (note: we may get more unit rules from step 2).

5. For each such $A, B$ (if $A \Rightarrow B$), make every RHS of $B$ a RHS of $A$.

6. Then delete all unit rules to get $G_1$ st. $L(G_1) = L(G) \setminus \{\varepsilon\}$.

Aliased every terminal $a$ to a variable $X_a$.
Example: $A \rightarrow BCdB$, then add variables $X_d, Y_1, Y_2$ and do:
$A \rightarrow BY_1, Y_1 \rightarrow CY_2, Y_2 \rightarrow X_dB, X_d \rightarrow d$.

Added - For Thursday - Full Examples:

- $S \rightarrow \varepsilon | (S)S$
- $L = \{x \epsilon \{\varepsilon, 1\}^* : x \text{ is balanced}\}$

1. Nullable $\subseteq \{S\} \text{ since } S \rightarrow \varepsilon$
2. Add rules: $S \rightarrow (\varepsilon) | (S) | (S)S$
3. Delete $S \rightarrow \varepsilon$: $G_1 = S \rightarrow (S)S | (\varepsilon) | (S) | (S)S$
4. No unit rules were introduced or were there previously. 5.6. can skip.
5. Alias $L \rightarrow (\varepsilon, R \rightarrow \varepsilon)$. $G_2 = S \rightarrow \varepsilon, RS | LR | LSR | LSR | R \rightarrow \varepsilon$
6. Add variables $Y_1, Y_2, Y_3, Y_4$.

Yes we could examine with $Y_4 \equiv Y_2$, but who cares? $G_3$ is UGLY!!!

7. New $L(G_3) = L \{\varepsilon, \varepsilon\}$. If we want to put $\varepsilon$ back in the language, do:

- $G' = S_0 \rightarrow \varepsilon | LY_1 | LR | LY_3 | L Y_4$
- $Y_1 \rightarrow SY_2, Y_2 \rightarrow RS, Y_3 \rightarrow SR, Y_4 \rightarrow RS$
- $S \rightarrow LY_1 | LR | LY_3 | LY_4 | L \rightarrow \varepsilon, R \rightarrow \varepsilon)$. OK w/ text change.

We couldn't add $S_0 \rightarrow \varepsilon$ since that would be a unit rule. Then $L(G') = L(G_3)$. \[\]