Assignment 4

Naman Mishra

14 March, 2024

Problem 4.1. Construct a context-free grammar for the language

$$\{xy \in \{0,1\}^* \mid |x| = |y| \text{ and } y \text{ contains a } 1\}.$$

Solution.

$$\begin{aligned} S &\to 0S0 \mid 1S0 \\ S &\to 0R1 \mid 1R1 \\ R &\to 0R0 \mid 0R1 \mid 1R0 \mid 1R1 \mid \epsilon \end{aligned}$$

■

Problem 4.2. Convert the following context-free grammar to Chomsky Normal Form (CNF):

$$\begin{aligned} S &\to AAA \mid B \\ A &\to aA \mid B \\ B &\to \epsilon \end{aligned}$$

Solution. We construct the new set of productions using these.

We add the transition $A \to \epsilon$, since $A \to B$ and $B \to \epsilon$. This allows us to add the transition $S \to AA$ and $A \to a$.

But $S \to AA$ and $A \to \epsilon$ means we can add $S \to A$. Now that $S \to A$ is a production, we can add $S \to a$.

We then drop unit- and ϵ-productions to get the rules

$$\begin{aligned} S &\to a \mid AA \mid AAA \\ A &\to a \mid aA \end{aligned}$$

The problematic productions are $S \to AAA$ and $A \to aA$. We can fix these by introducing non-terminals X and Y with rules

$$\begin{aligned} X &\to AA \\ Y &\to a \end{aligned}$$

This gives the CNF

$$\begin{aligned} S &\to a \mid AA \mid XA \\ X &\to AA \\ A &\to a \mid YA \\ Y &\to a \end{aligned}$$

■

Problem 4.3. Consider the language $L = \{a^n b^{n^2} \mid n \geq 0\}$. Use the Pumping Lemma for CFLs to show that L is not a CFL.

Solution. Suppose L is a CFL. Let k be the pumping constant. Consider the string $z = a^k b^{k^2}$. Then $z = uvwxy$ such that

- $|vwx| \leq k$,
- $vx \neq \epsilon$, and
- $uv^iwx^iy \in L$ for all $i \geq 0$.

Suppose $v = a^\alpha b^\beta$ with $\alpha\beta \neq 0$. Then $uv^2wx^2y = ua^\alpha b^\beta a^\alpha b^\beta wx^2y$ contains at least two runs of a's and hence cannot be in L. Thus v is not of this form. Similarly, neither is x.

Thus v and x can only contain either all a's or all b's. If they were both all a's, then uv^2wx^2y would contain more than k a's but only k^2 b's. Similarly, if they were both all b's, then uv^2wx^2y would contain more than k^2 b's but only k a's.

Thus $v = a^\alpha$ and $x = b^\beta$. Then $uv^2wx^2y = a^{k+\alpha}b^{k^2+\beta} \in L$. This gives $\beta = 2k\alpha + \alpha^2$. Also, $uwy = a^{k-\alpha}b^{k^2-\beta} \in L$. This gives $\beta = 2k\alpha - \alpha^2$. Thus $\alpha^2 = 0 \implies \alpha = 0$ and $\beta = 0$. But then $vx = \epsilon$, a contradiction.

Thus L is not a CFL. ■

Problem 4.4. Consider the CFG G below:

$$\begin{aligned} S &\to XC \mid AY \\ X &\to aXb \mid ab \\ Y &\to bYc \mid bc \\ A &\to aA \mid a \\ C &\to cC \mid c \end{aligned}$$

(a) Describe the language accepted by G.

(b) Use the construction in Parikh's theorem to construct a semi-linear expression for $\psi(L(G))$. That is

  i. Identify the basic pumps for G.

  ii. Identify the $\leq$-minimal trees.

  iii. Use those to obtain an expression for $\psi(L(G))$.

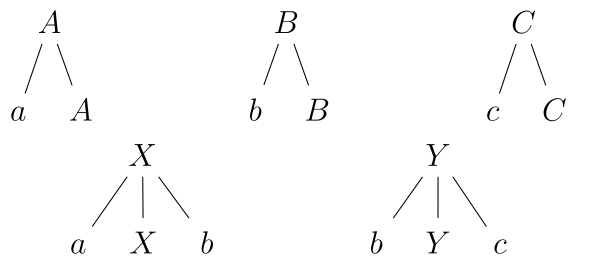
(c) Use the semi-linear expression obtained above to give a regular expression that is letter-equivalent to $L(G)$.

Solution.

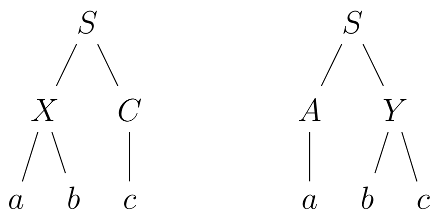
(a)

$$L(G) = \left\{a^l b^m c^n \mid l, m, n \geq 0 \text{ and } m = l \text{ or } m = n\right\}.$$

(b) i. The basic pumps are $A \to aA$, $B \to bB$, $C \to cC$, $X \to aXb$, and $Y \to bYc$.



ii. The $\leq$-minimal trees are $S \to XC \Rightarrow abc$ and $S \to AY \Rightarrow abc$.



iii. The basic pumps, in the order they are listed, are of lengths 1, 1, 1, 2 and 2.

For the first tree, we can apply the pumps $X \to aXb$, $B \to bB$ or $C \to cC$ Thus that "bucket" gives Parikh map $\{3 + 2m + n\}$.

For the second tree, we can apply the pumps $A \to aA$, $B \to bB$ or $Y \to bYc$. Thus that bucket gives the same Parikh map.

Thus the Parikh map of the language is

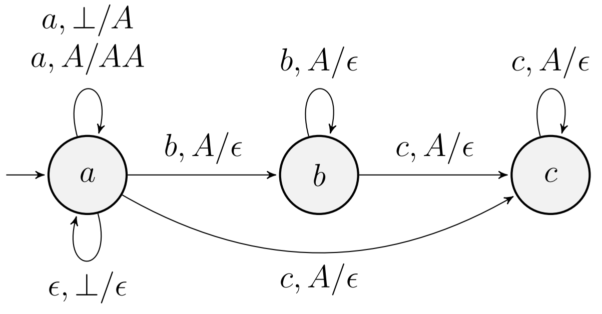
$$\psi(L(G)) = 3 + \langle\!\langle 2, 1 \rangle\!\rangle = 3 + \langle\!\langle 1 \rangle\!\rangle$$

or simply $\{3, 4, \ldots\}$. ■

(c) $aaaa^*$ also has Parikh map $\{3, 4, \ldots\}$.

Problem 4.5. Give the transition diagram of a PDA for the language $L = \{a^l b^m c^n \mid l = m + n\}$.

Solution.



accepts L via empty stack. ■

Problem 4.6. Is the class of context-free languages closed under the prefix operation? Justify your answer.

Solution. Yes.

Let pref(L) denote the set of prefixes of strings in L.

$$\text{pref}(L) = \{u \mid \exists v \in A^*, uv \in L\}.$$

We first do away with the case of the empty language. pref$(\varnothing) = \varnothing$ is a CFL iff $\varnothing$ is a CFL. Whether $\varnothing$ is a CFL or not (it is) is not even relevant.

Next, notice that pref(L) contains the empty string for any non-empty L. Thus pref$(L) =$ pref$(L \setminus \{\epsilon\})$. So we can assume that L does not contain the empty string.

Now let $G = (N, A, S, P)$ be a CFG for L in CNF. Assume that there are no non-terminals from which *no* terminal string can be derived.[1]

For each non-terminal $X \in N$, introduce a new non-terminal $X_\perp \notin N$. Denote the set of these new non-terminals $N_\perp = \{X_\perp \mid X \in N\}$. We will call these "vanishing" non-terminals, or simply "vanisher"s.

For each $X_\perp$, we add the following productions to P:

- $X_\perp \to \epsilon$.
- $X_\perp \to a$, for each production $X \to a$.
- $X_\perp \to YZ_\perp$ and $X_\perp \to Y_\perp$, for each production $X \to YZ$.

Call this new set of productions P'

Intuitively, vanishing non-terminals are allowed to go to the empty string directly. They represent truncation, and as such can only be allowed to appear at the end of any sentential form.

We claim that $G' = (N \cup N_\perp, A, S_\perp, P')$ is a CFG for the prefix of L, *i.e.*,

$$L(G') = \text{pref}(L).$$

We will use "non-terminal sentential form" to mean a sentential form which contains at least one non-terminal. We use $\Rightarrow$ to denote derivation steps in both G and G'. The context will make it clear which grammar we are referring to (if any vanisher is present, we are referring to G').

Lemma 4.1. *Let $S_\perp \overset{n}{\Rightarrow} \omega \overset{1}{\Rightarrow} w$ be a leftmost derivation of $w \in L(G')$. Then $\omega = zX_\perp$ for some $z \in A^*$ and $X \in N$.*

Moreover, each intermediate sentential form is of the form $\gamma X_\perp$ for some $\gamma \in (A \cup N)^*$ and $X \in N$.

[1]If there are, we can remove them, and all productions containing them, from G without affecting the language generated. We handled $L = \varnothing$ separately because the starting symbol cannot be removed.

Proof. The last symbol in $S_\perp$ is vanishing. From the definition of P' we see that for any production $X_\perp \to \gamma$, γ is either a terminal string or ends with a vanisher.

If a production from the rightmost non-terminal contains no non-terminals, then it must be the last step in the derivation (since it is leftmost).

Combining these two, we can induct on the length n of the derivation.

For $n = 0$, the only sentential form is $S_\perp$, and the claim holds. In each step before the last, it is not possible for the last non-terminal to go to a terminal string. But then each step must maintain that the last symbol is a vanisher.

By induction, each intermediate sentential form (including ω) is of the form $\gamma X_\perp$ for some $\gamma \in (A \cup N)^*$ and $X \in N$.

But ω contains exactly one non-terminal, since it derives a terminal string in a single step. Thus $\omega = zX_\perp$ for some $z \in A^*$ and $X \in N$. □

Lemma 4.2. *Suppose $zX_\perp$ is a sentential form in G'. Then there is a sentential form $zX\gamma$ in G.*

Proof. Consider a leftmost derivation $S_\perp \overset{*}{\Rightarrow} zX_\perp$. By the previous lemma, each intermediate sentential form is of the form $\gamma Y_\perp$. For any production $Z \to \zeta$, keep it as is. For any production $Z_\perp \to \zeta$, replace it with the corresponding production in P. That is,

$$\begin{array}{lcl} Z_\perp \to a & \Longrightarrow & Z \to a \\ Z_\perp \to Z_1(Z_2)_\perp & \Longrightarrow & Z \to Z_1Z_2 \\ Z_\perp \to (Z_1)_\perp & \Longrightarrow & Z \to Z_1Z_2 \end{array}$$

where in the last case, $Z \to Z_1Z_2$ must exist in P for some Z_2, in order for $Z_\perp \to (Z_1)_\perp$ to be in P'.

Then for any $\gamma Y_\perp \overset{1}{\Rightarrow} \gamma'$, this process preserves the first $|\gamma'|$ symbols, but with each vanishing non-terminal replaced by its non-vanishing counterpart.

Thus, this yields a derivation of some sentential form γ'', whose $|zX_\perp|$-length prefix is zX. □

Proposition 4.3. $L(G') \subseteq \mathrm{pref}(L(G))$.

Proof. Let $S_\perp \overset{n}{\Rightarrow} \omega \overset{1}{\Rightarrow} w$ be a leftmost derivation of $w \in L(G')$.

By lemma 4.1, $\omega = zX_\perp$ for some $z \in A^*$ and $X \in N$. By lemma 4.2, there is a sentential form $zX\gamma$ of G.

We assumed that each non-terminal derives some terminal string. By extension, each sentential form of G derives a terminal string. Let γ derive a terminal string y, and X derive a terminal string x.

The last step in the derivation of w must involve $X_\perp$. If it is $X_\perp \to a$, then $X \to a$ is a production in P and so $zay \in L(G)$. Then $w = za \in \mathrm{pref}(L(G))$.

If it is $X_\perp \to \epsilon$, then $w = x$. But $zxy \in L(G)$, so $w \in \mathrm{pref}(L(G))$.

Thus $L(G') \subseteq \mathrm{pref}(L(G))$. □

Proposition 4.4. $\mathrm{pref}(L(G)) \subseteq L(G')$.

Proof. Let $w \in \mathrm{pref}(L(G))$.

Look at the parse tree of a derivation of w in G. Since $w \in \mathrm{pref}(L(G))$, the leftmost $|w|$ leaves of the parse tree give w.

We will give a scheme to derive w in G'.

If $w = \epsilon$, then apply the production $S_\perp \to \epsilon$. Otherwise, take the subtree rooted at the root which contains only the leaves for w. Look at the first level.

If the root production is $S \to a$, then $w = \epsilon$ or $w = a$, which can both be derived from $S_\perp$ in a single step.

Otherwise, the root production is $S \to XY$, and the w-subtree must contain at least one child of the root.

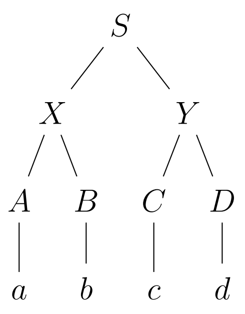
- If the subtree for w only contains left child, then apply the production $S_\perp \to X_\perp$. This reduces the problem to deriving w from $X_\perp$, which can be done by repeating this process on the subtree rooted at X.
- If it contains both, then $w = xy$, where x is derived from X, and y is the prefix of a terminal string derived from Y. Keep the productions from X as is. Apply this same process to the subtree rooted at Y to derive y.

For example, consider the grammar G in CNF with productions

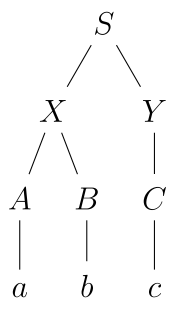
$$S \to XY$$
$$X \to AB$$
$$Y \to CD$$

$$A \to a \qquad B \to b \qquad C \to c \qquad D \to d$$

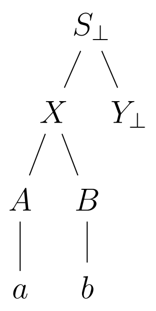
The parse tree for $abcd$ is



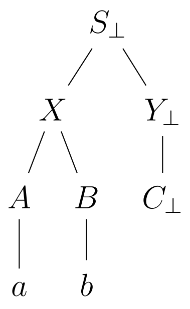
For the prefix abc, the subtree is



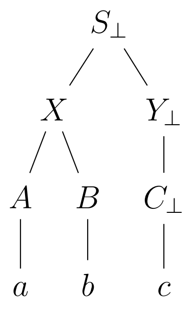
Then the above algorithm gives the first step as $S_\perp \to XY_\perp$ and asks to keep the tree rooted at X fixed.



Then applying this process to derive c from $Y_\perp$ gives



and then



which is a derivation of abc in G'. $\square$

Combining the two propositions, we have $L(G') = \text{pref}(L(G))$. $\blacksquare$