Assignment 4

COMPSCI 3331

Due: December 7, 2022 at 11:59 PM

(5 marks) 1. Recall that for a language $L \subseteq \Sigma^*$,

$$\text{pref}(L) = \{u \in \Sigma^* \ : \ \exists w \in \Sigma^*, v \in L \text{ such that } v = uw\}.$$

That is, $\text{pref}(L)$ is the language of all prefixes of all words in L.

(a) Let $G = (V, \Sigma, P, S)$ be a context free grammar in CNF for a language L. Consider the following modified grammar $G' = (V, \Sigma, P', S)$ where

$$P' = \{A \to BC, A \to B, A \to \varepsilon \ : \ A \to BC \text{ is a production in P }\}. \cup \{A \to a : A \to aa \in P\}.$$

That is, for every rule in P, we add three rules in P'. The idea is that by adding the prefixes of the rules of the form $A \to BC$, that will allow A to now generate any prefix of the words that it used to be able to generate. (Note that G' is not in CNF, but that's alright for this construction. We are not insisting that the grammar is in CNF.)

Show, by giving a counter-example, that $L(G') = \text{pref}(L(G))$ does not hold for all grammars G.

Let $G = (V, \Sigma, P, S)$ where $V = \{S, A, B, C\}$, $\Sigma = \{a, b\}$ and the productions

$$\begin{aligned} S &\to AA \\ A &\to BC \\ B &\to a \\ C &\to b \end{aligned}$$

Note that $L(G) = \{abab\}$ and $\text{pref}(L(G)) = \{abab, aba, ab, a, \varepsilon\}$.

The grammar G' has the productions

$$\begin{aligned} S &\to AA \mid A \mid \varepsilon \\ A &\to BC \mid B \mid \varepsilon \\ B &\to a \\ C &\to b \end{aligned}$$

Then note that $S \Rightarrow AA \Rightarrow BB \Rightarrow aa$ is a derivation in G and thus $L(G') \neq \text{pref}(L(G))$ as $aa \notin \text{pref}(L(G))$.

(b) Modify the construction from part (a) to get a grammar for the language $\text{pref}(L(G))$. In particular, you will need to:

- Add new nonterminals to the grammar. These nonterminals will mirror the nonterminals in G, but allow a prefix to be generated. That is, if X is a nonterminal in G and $X \Rightarrow^* x$ then the new nonterminal Q will be able to generate $Q \Rightarrow^* q$ for any prefix q of x. (The names of the nonterminals are made up here – they don't mean anything.) The existing nonterminals should also be kept.
- Add new rules to the grammar for the new variables. These new rules will use the new nonterminals and allow them to produce the prefixes that they are supposed to predict. You will probably use the idea in part (a) as a starting point for doing this, but consider the cases carefully.
- Incorporate the new nonterminals by adding new rules that use them and link them to the existing nonterminals.

This is not an exhaustive list and you will likely need to consider other matters. Give a justification for your construction.

Our grammar is $G = (V, \Sigma, P, S)$ and our modified grammar is $G' = (V', \Sigma, P', S')$ where

- $V' = \{A, \overline{A} \ : \ A \in V\}$, i.e., there is a copy of every nonterminal in V with a bar.
- The set of productions are $P' = P \cup \overline{P}$ where if $A \to BC$ is a production in P, then the following productions are added to $\overline{P}$:
$$\overline{A} \to B\overline{C} \mid \overline{B} \mid \varepsilon$$
And if $A \to a$ is a rule in P, then the following productions are added to $\overline{P}$:
$$\overline{A} \to a \mid \varepsilon$$
- The new start symbol is $\overline{S}$.

(5 marks) 2. Let $L = \{a^i b^j c^k \ : \ i, j, k \geq 0 \text{ and } j = \min(i,k)\}$. Prove that L is not a CFL.

Assume that L is context-free and let n be the constant from the pumping lemma. Let $z = a^n b^n c^n$. Then $z \in L$ and $|z| = 3n$. Therefore, there exist $u, v, w, x, y \in \{a, b, c\}^*$ such that $z = uvwxy$ and

pick a specific case that fits the requirement.

(i) $|vwx| \leq n$;

(ii) $vx \neq \varepsilon$;

Consider a factorization $z = uvwxy$. If either one of v or x contain more than one letter from $\{a,b,c\}$ (in other words, $v,x \notin a^* + b^* + c^*$), then $uv^2wx^2y \notin a^*b^*c^*$ (it would, e.g., contain b followed by an a). In this case, $uv^2wx^2y \notin L$. Thus, we must have $v,x \in a^* + b^* + c^*$, that is, each of v and x consist of repetitions of a single letter (possibly different letters for v and x, however).

As $|vwx| \leq n$, we cannot have that $v \in a^*$ and $x \in c^*$ (in this case, w would contain at least each of the n occurrences of the letter b and thus $|vwx| > n$.

There are five remaining cases:

(a) v,x **are contained in the first block of** a**s.** Thus, there exist j,k,ℓ,m such that

$$\begin{aligned} u &= a^j \\ v &= a^k \\ w &= a^\ell \\ x &= a^m \\ y &= a^{n-j-k-\ell-m}b^nc^n. \end{aligned}$$

Further, we must have that $k+m > 0$, by (ii). Consider $i = 0$. Then $uv^0wx^0y = a^{n-(k+m)}b^nc^n$. As $n-(k+m) < n$, $uv^0wx^0y \notin L$.

pick a specific number for i.

(b) $v \in a^*$ **and** $x \in b^*$**.** Thus, there exist j,k,ℓ,m such that

$$\begin{aligned} u &= a^j \\ v &= a^k \\ w &= a^{n-(j-k)}b^\ell \\ x &= b^m \\ y &= b^{n-\ell-m}c^n. \end{aligned}$$

Further, we must have that $k+m > 0$, by (ii). Therefore either $k > 0$ or $m > 0$. If $m = 0$, then we must have that $k > 0$. Consider $i = 0$. Then $uv^0wx^0y = a^{n-k}b^{n-m}c^n = a^{n-k}b^nc^n$. As $n-k < n$, $uv^0wx^0y \notin L$. If $m \neq 0$, then consider $i = 2$. Then $uv^2wx^2y = a^{n+k}b^{n+m}c^n$. As $n+m > n$, $uv^2wx^2y \notin L$ (the number of b's is greater than the number of c's).

(c) v,x **are contained in the block of** b**s.** Thus, there exist j,k,ℓ,m such that

$$\begin{aligned} u &= a^nb^j \\ v &= b^k \\ w &= b^\ell \\ x &= b^m \\ y &= b^{n-j-k-\ell-m}c^n. \end{aligned}$$

Further, we must have that $k+m > 0$, by (ii). Consider $i = 2$. Then $uv^2wx^2y = a^nb^{n+(m+k)}c^n$. As $n+(m+k) > n$, $uv^2wx^2y \notin L$ (the number of c's/a's is less than the number of b's).

(d) $v \in b^*$ **and** $x \in c^*$**.** This case is similar to case (b). There exist j,k,ℓ,m such that

$$\begin{aligned} u &= a^n b^j \\ v &= b^k \\ w &= b^{n-(j-k)} c^\ell \\ x &= c^m \\ y &= c^{n-\ell-m}. \end{aligned}$$

Further, we must have that $k+m>0$, by (ii). Therefore either $k>0$ or $m>0$. If $k=0$, then $m>0$, and consider $i=0$. Then $uv^0wx^0y = a^n b^{n-k} c^{n-m} = a^n b^n c^{n-m}$. As $n-m<n$, $uv^0wx^0y \notin L$. If $k \neq 0$, then consider $i=2$. Then $uv^2wx^2y = a^n b^{n+k} c^{n+m}$. As $n+k>n$, $uv^2wx^2y \notin L$ (the number of as is less than the number of bs).

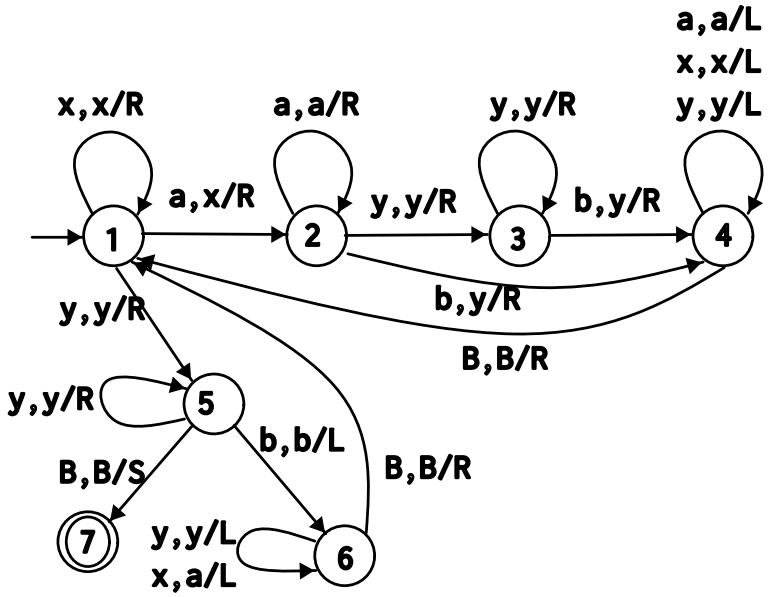
(e) $v,x \in c^*$. This case is similar to case (a). There exist j,k,ℓ,m such that

$$\begin{aligned} u &= a^n b^n c^j \\ v &= c^k \\ w &= c^\ell \\ x &= c^m \\ y &= c^{n-j-k-\ell-m}. \end{aligned}$$

Further, we must have that $k+m>0$, by (ii). Consider $i=0$. Then $uv^2wx^2y = a^n b^n c^{n-(m+k)}$. As $n-(m+k)<n$, $uv^2wx^2y \notin L$ (the number of cs is less than the number of bs).

Therefore, L is not a context-free language.

(5 marks) 3. Give the transition diagram for a one-tape Turing machine that recognizes the language $L = \{a^n b^{kn} \ : \ n \geq 1, k \geq 1\}$. That is, L is the language of words of the form $a^i b^j$ where j is a multiple of i, so $aabbbb, aaabbb, abbbbbbb \in L$ but $aabbb \notin L$. Your TM must be a one-tape machine.



The idea of this TM:

- In the course of a successful computation, all a's are converted to x's and all b's are converted to y's, by marking them off.
- States 1–4 mark one a and one b. When this is done, the tape is rewound to the beginning and we return to state 1.
- Once we run out of a's to mark, in state 1, we encounter a y without finding an a to mark. At this stage, the first time through the loop, we have an input that looks like $x^i y^i b^j$ for some i, j. In later loops, we have an input that will look like $x^i y^{ki} b^j$ for some i, k, j (we will argue this below). Then we go to state 5.
- State 5 verifies that there are no more b's to mark. If that's the case, then we accept in state 7: the input has the form $x^i y^{ki}$.
- Otherwise, we go to state 6 when we find there is a b. This means we are possibly not done.
- We rewind the tape, **BUT** we unmark all the a's in the process. Then we return to state 1.
- At this point, the tape looks like $a^i y^{ri} b^j$ for some i, r, j. That is, we've marked a multiple of i copies of b's.
- We now restart the process of marking a's and b's using the state 1–4 loop. After this, mark i more b's in the input.

(5 marks) 4. Give an informal description of a Turing machine that recognizes the language $L = \{a^{n!} : n \geq 1\}$. Your Turing machine can be a multi-tape TM. Your description should not be a transition diagram. You should describe the function of the TM exactly: what are the tapes (if more

than one)? what does the machine do on start up? when does it accept? under what conditions does it not accept? etc.

Our TM has 4 tapes:

- Tape 1 is the input tape. We treat it as read-only.
- Tapes 2,3,4 are work tapes.
- Our TM will work in loops (step 3 below).
- After each loop, tape 4 will contain values of $a^{i!}$ for successive values of i.
- During each loop, if tape 4 contains $a^{i!}$, then tape 2 will contain $a^{(i-1)!}$, and tape 3 will contain a^i.

The basic idea is to calculate $a^{2!}, a^{3!}, \ldots$ on tape 4 until we determine if the input is exactly equal to one of these values. If not, then we reject.

We perform the following algorithm:

1. We determine if there is $1 = 1!$ occurrence of the letter a on the input tape. If so, then we accept.
2. Otherwise, we write one a on tape 2, two a's on tape 3.
3. We repeat the following:
   - 3.-a. If there are i a's on tape 2 and j a's on tape 3, we write ij a's on tape 4 (by repeatedly writing j a's to tape 4 for each occurrence of an a on tape 2, marking occurrences of a on tape 2 as we go.)
   - 3.-b. We compare the number of a's on tape 4 and tape 1 (we can walk along both tape 1 and 4 simultaneously and check which encounters a blank symbol first).
   - 3.-c. If the number of a's on tapes 1 and 4 are the same, we accept.
   - 3.-d. If the number of as on tape 4 is strictly greater than the number of as on tape 1, we reject the input.
   - 3.-e. Otherwise, we copy the contents of tape 4 to tape 2, and write one additional a on tape 3, and go to step 3.-a.

The machine works as follows: on tape 4, we first calculate successive values of the factorials: $2!, 3!, 4!$ and so on. If the input matches on of these words, we accept. If the input is between $n!$ and $(n+1)!$, we will eventually write $a^{(n+1)!}$ on tape 4 and in step 3-.d., we will reject.