Assignment 4

COMPSCI 3331

Due: December 7, 2022 at 11:59 PM

General notes:

- Assignments **must** be submitted on gradescope. You must indicate the locations of all answers for questions using gradescope. A video demonstrating how to do this can be found here.
- Assignments can be hand-written or typeset, as long as they are submitted to gradescope as an electronic file (pdf, png or other accepted format). It is your responsibility to submit a file that can be marked (i.e., images of pages are clear and handwriting, if any, can be read).
- Assignments can be submitted up to 48 hours late. A deduction of 1 % (of the total assignment value) will be applied per hour (rounded up) that the assignment is submitted past the deadline.
- You may also use your **once-per-course** 3-day extension on this assignment. Please submit the form on owl to declare that you want to use this extension. (choose 'Individual Extension' from the tool menu on owl.) Recall that extensions do not stack – you may either choose the late submission penalty or the individual extension for an assignment, but not both.

(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, 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

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

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.

(b) Modify the construction from part (a) to get a grammar for the language 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.

(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.

(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.

(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.

(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, 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

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

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.

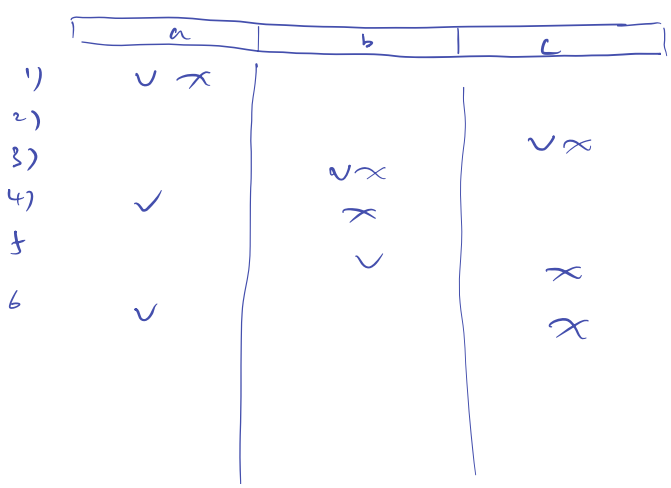
(b) Modify the construction from part (a) to get a grammar for the language pref$(L(G))$. In particular, you will need to:

- 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.


a
b
c
1)
2)
3)
4)
5
6

(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.

$a^i b^j c^k = uvwxyz$ $\quad |vwx| \leq n$ ← pumping length.
$|vx| > 0$.

$z = a^i b^j c^k$ $j = \min(i,k)$. $\quad i, j, k \geq 0$ $\quad i + k \neq j \leq n$

0) v, x contains more than one character:
in this situation, the number contains is out of order.

1) both v, x in a:
$u = a^a$ $v = a^b$ $w = a^c$ $x = a^d$ $y = a^{i-a-b-c-d} b^j c^k$.
$uv^2wx^2y = a^{i+b+d} b^j c^k \notin L$ since $|vx| > 0$.

2) both v, x in c:
Similar with case 1)

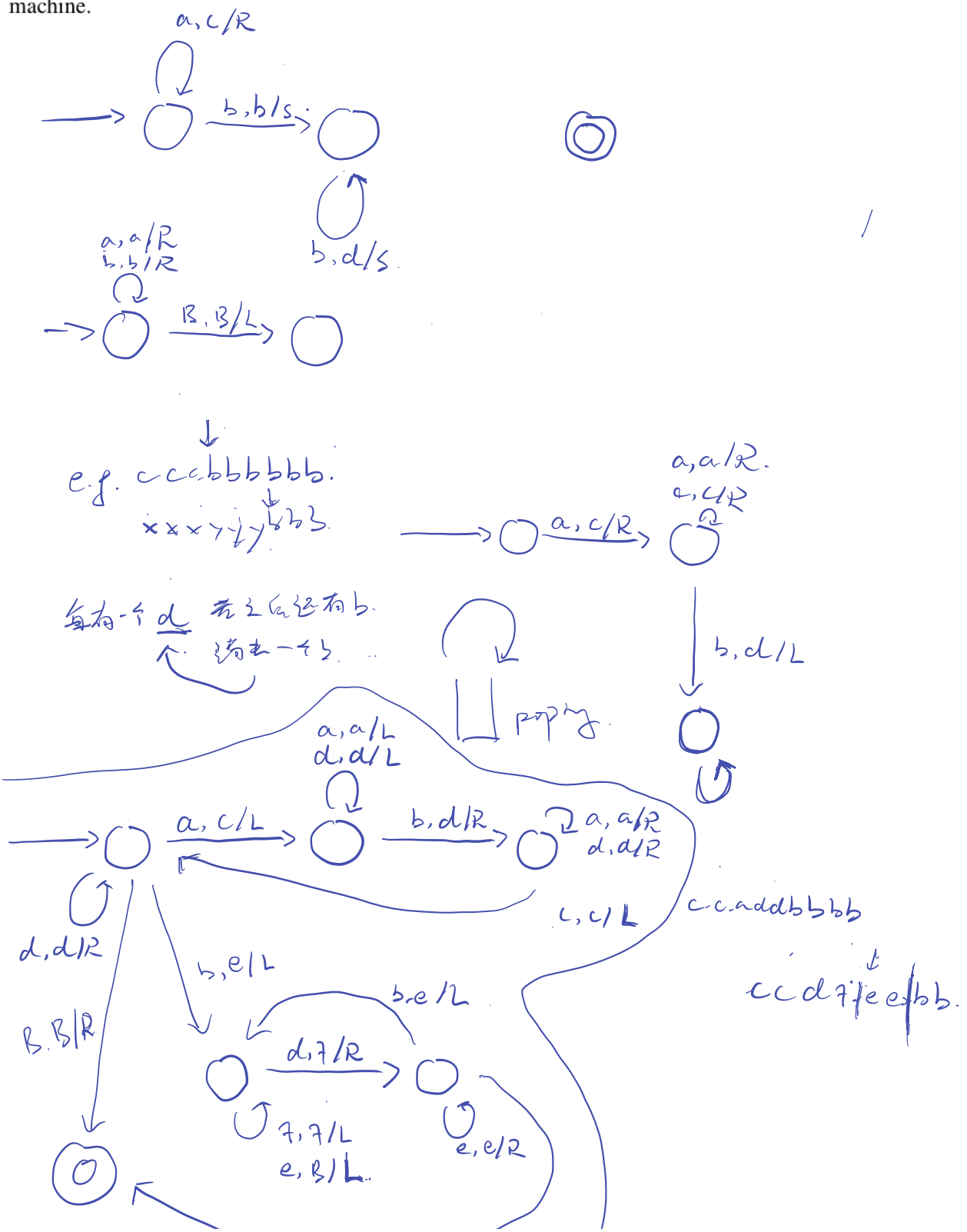
3) both v, x in b:
$u = a^i b^a$ $v = b^b$ $w = b^c$ $x = b^d$ $y = b^{j-a-b-c-d} c^k$.
$uv^2wx^2y = a^i b^{j+b+d} c^k \notin L$ since $j+b+d \neq i$ and $j+b+d \neq j$.

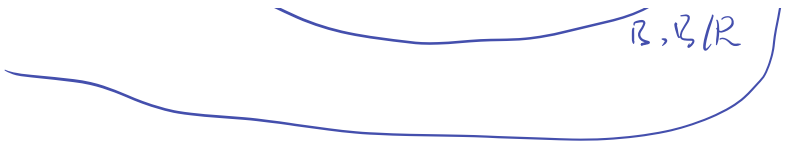
4) v in a, x in b.
$u = a^a$ $v = a^b$ $w = a^{i-a-b}$ $x = b^c$ $y = b^{j-c} c^k$
$uv^2wx^2y = a^{i+b} b^{j+c} c^k$ this is not in L.

5) v in b x in c:
Similar with 4).

6) v in a, x in c:
this case does not matter b.

(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.





(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.

tape 1: make up words

tape 2: counting the number of a's to coordinate in this loop.

tape 3: counting the number to go.

on start up, the machine creating three tapes.

it accepts words that could finish looping, and reject words that could not finish any one of the loops.