COMP 335: Introduction to Theoretical Computer Science

Assignment 6

Nathan Grenier

December 2, 2024
Fall 2024

1. [30 Points] Classify the following languages into one of the three categories:

   (a) regular

   (b) context-free but not regular

   (c) not context-free

   Prove your answer. Note that in order to show that a language is context-free but not regular, you need to prove both that it is context-free and also that it is not regular.

   (a) $L_1 = \{a^i b^j c^k \mid k = i \times j \text{ and } 0 < i < 10 < j\}$

   Showing that L_1 is context-free: Since i is bounded by a finite amount, we can construct individual grammars that accept the language for every specific i. Then we can combine all of these grammars into a single grammar that accepts the language L_1.

   When $i = 1$: $G_1 =$

$$S \to aB$$
$$B \to b^{11}Cc^{11}$$
$$C \to bCc \mid \lambda$$

   When $i = 2$: $G_2 =$

$$S \to aaB$$
$$B \to b^{11}Cc^{22}$$
$$C \to bCcc \mid \lambda$$

   ...

When $i = 9$: $G_9 =$

$$S \to a^9 B$$
$$B \to b^{11} C c^{99}$$
$$C \to bCc^9 \mid \lambda$$

Finally, $G_1 \cup G_2 \cup \cdots \cup G_9 = G_f$:

$$S \to G_1 \mid G_2 \mid \ldots \mid G_9$$
$$G_1 \to \ldots$$
$$G_2 \to \ldots$$
$$\ldots$$
$$G_9 \to \ldots$$

Showing that L_1 is not regular: We can use the pumping lemma for regular languages to prove that L_1 is not regular.

Let m be the integer if the PL.

$$w = a^i b^{11+m} c^{i(11+m)}$$

- $w \in L_1$
- $\mid w \mid = i(11+m) + m + 11 + i. \therefore \mid w \mid \geq m$

By PL $\exists\, x, y, z : w = xyz, \mid xy \mid \leq m, \mid y \mid \geq 1$

$$\mid --i--- \mid - \mid -m-- \mid i(11+m) \mid$$
$$w_j = (aa \ldots aa) b^{11} (bb \ldots bb)(ccc \ldots ccc)$$
$$\mid ---xy------ \mid -- z----- \mid$$

Let $j = 2$: $w_2 = xyyz = a^i b^{m-(11+i)} b^k b^{11+i} c^{i(11+m)} = a^i b^{m+k} c^{i(11+m)}$

- $w_2 \in L_1$ by PL
- $w_2 \notin L_1$ by the definition of L_1. $n_a(w) \times n_b(w) = n_c(w)$ but $i(m + k) \neq i(11 + m)$

(b) $L_2 = \{xyz \mid x, y, z \in \{a, b\}^* \text{ and } n_a(x) = n_b(z)\}$

Answer: L_2 is a regular language.

Proof: There is a redundancy in the definition of language L_2. Since $x, y, z \in \{a, b\}^*$, all strings in the alphabet Σ can be generated by only one of the 3 variables. Therefore, if we make $x, z = \lambda$, the language can still generate all possible strings in Σ^* through x and the condition of $n_a(x) = n_b(z)$ is true because $0 = 0$.

We can now construct a regular grammar G that accepts the language $L_2 = \{x \mid x \in \{a, b\}^* \text{ where } n_a(x) = n_b(z)\}$.

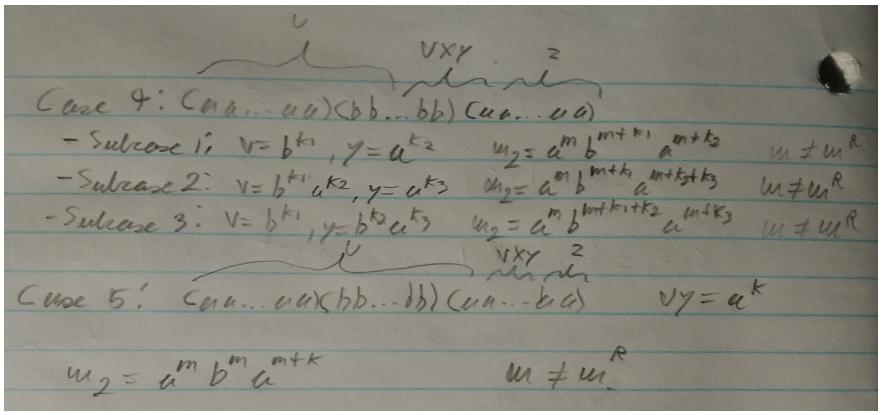
$G =$

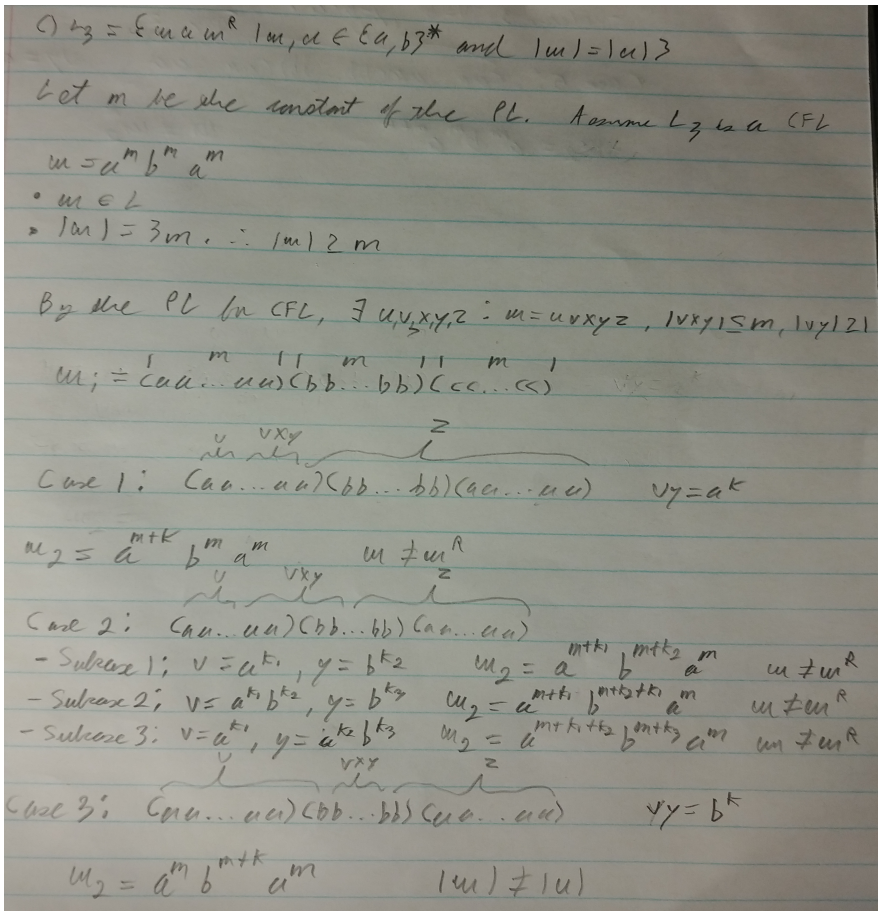
$S \rightarrow aS \mid bS \mid \lambda$

(c) $L_3 = \{wuw^R \mid w, u \in \{a, b\}^* \text{ and } \mid w \mid = \mid u \mid\}$

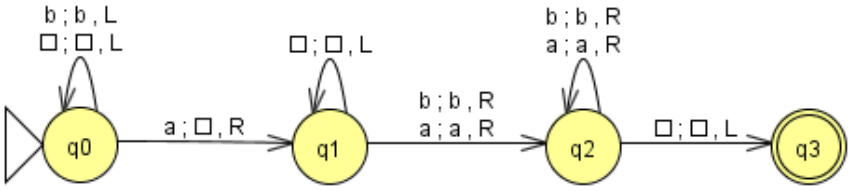
Answer: L_3 is not a context-free language.

Proof:



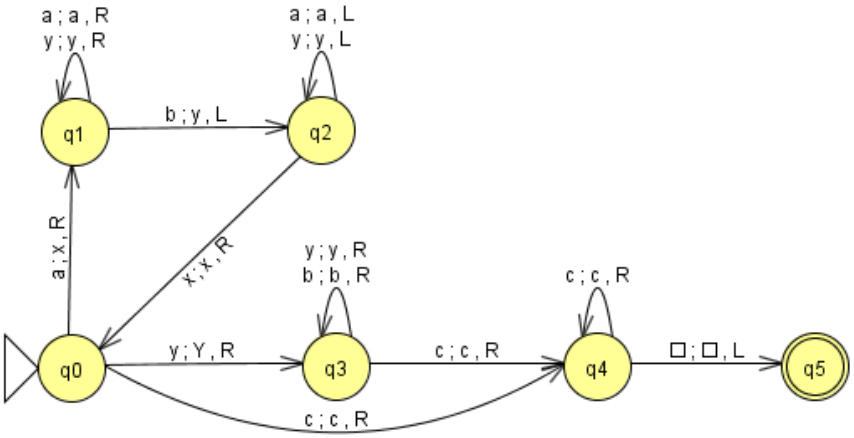


2. [10 Points] Give a Turing machine for $L = \{a\} \cdot \{a, b\}^+$ that does not halt on rejection.

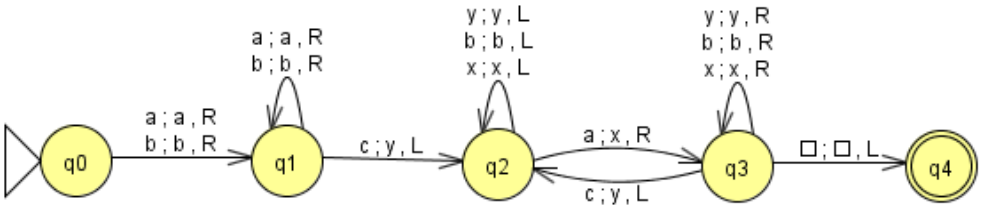


3. [20 Points] Give a Turing machine for each of the following languages.

(a) $L_1 = \{a^n b^m c^k \mid m \geq n, k \geq 1\}$



(b) $L_2 = \{xy \mid x \in \{a, b\}^+, y \in \{c\}^+ \text{ and } n_a(x) = n_c(y)\}$



4. [20 Points] Draw transition diagrams for Turing machines that compute the following functions. In each case, give a brief description in English of your solution strategy.

   (a) $f(1^n) = 1^{2n}$

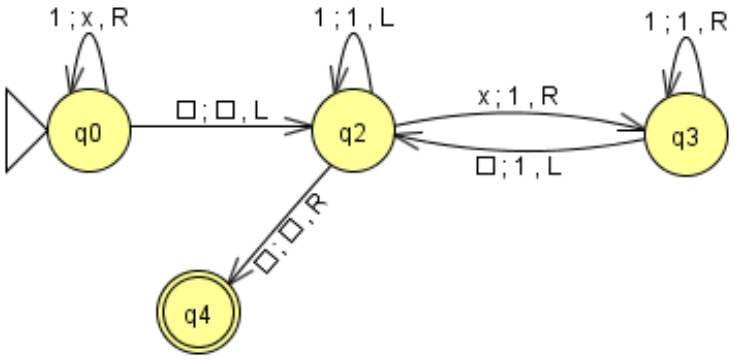
   Algorithm Description: The algorithm is as follows:

   i. Iterate over the initial number, replacing every 1 with an x.

   ii. Once we reach the end of the string, use the zig zag pattern to replace every x with a 1 and simultaneously add a 1 at the end of the string.

   iii. When there are no more xs to replace we've successfully duplicated the entire input. Therefore we move the head to the start of the string and halt.

   Note that this algorithm can also handle the case where $n = 0$. Since there are no 1s to copy, the algorithm will terminate early in state q_2.



(b) $f(1^n) = 1^{n^2}$

Algorithm Description: A number to the power of 2 is the number multiplied by itself. Multiplication is a chain of additions. Example: $3^3 = 3 \times 3 = 3 + 3 + 3$.

The algorithm is as follows:

i. Copy the initial number to the right of the input string delimited by a 0. This copy will be used for future additions / copies.

ii. Iterate over the initial number, replacing every x with a \$.Each \$ represents the number of times the initial number needs to be copied (added).

iii. Now, iterate over the \$s and copy the copy of the initial number for as many times as there are \$ characters (each copy is delimited by a 0). We replace \$ that have already been iterated on with Z. Note that we mark off an extra \$ at the start since we've already copied the initial number once.

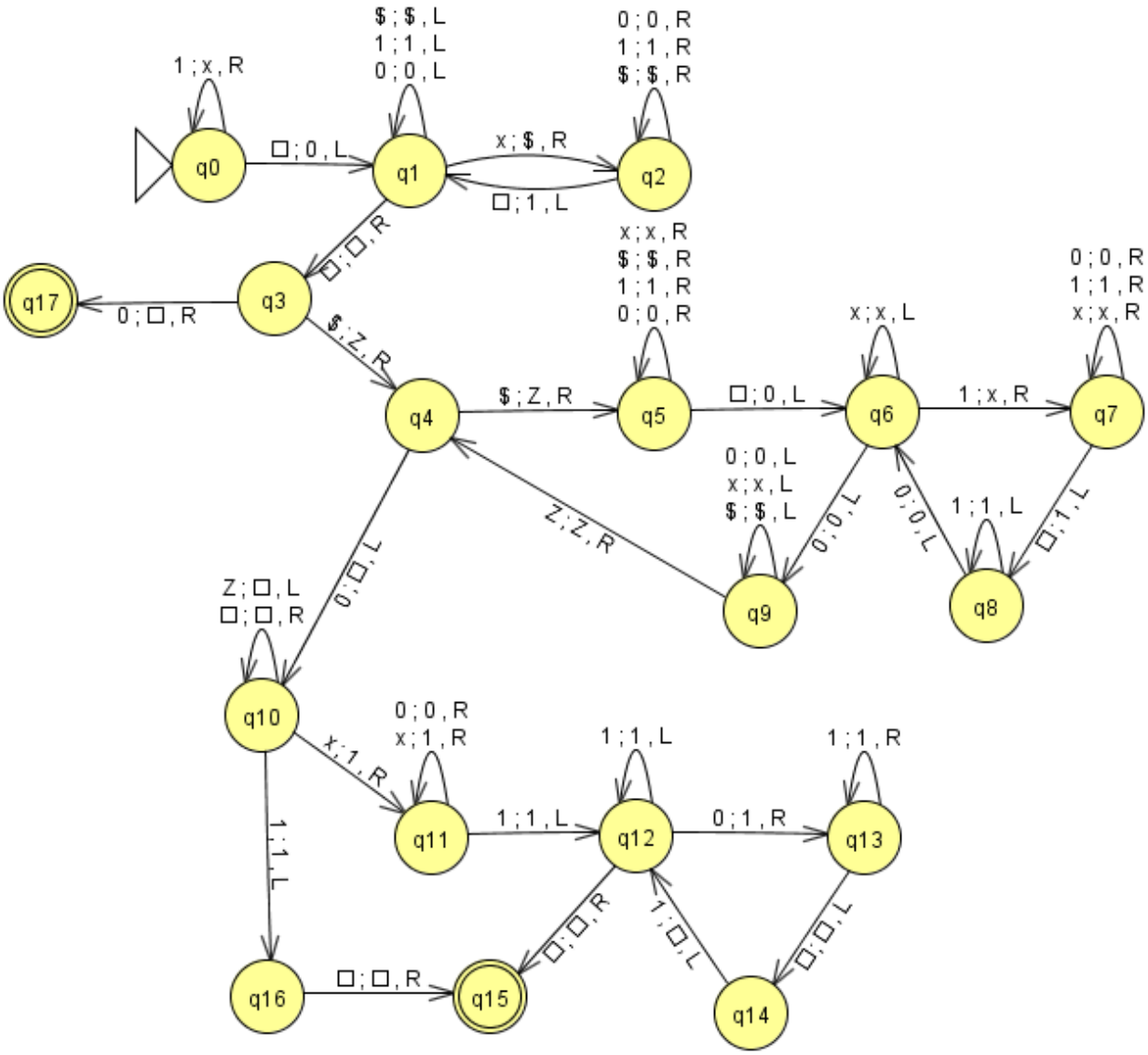
iv. Once there are no more \$s, we erase the iterators and it's delimiter.

v. Finally, we clean up the computed string. First we replace all xs with 1s. Then we remove all delimiters by replacing them with 1 and removing a 1 at the end of the string.

vi. We then move the head to the start of the string and halt.

Note that there are certain edge cases that must be handled:

- $1^2 = 1$: Therefore we must terminate early in state q_{10}.
- $0^2 = 0$: Therefore we must terminate early in state q_3.


q0
q1
q2
q3
q4
q5
q6
q7
q8
q9
q10
q11
q12
q13
q14
q15
q16
q17
1 ; x , R
□ ; 0 , L
$; $, L
1 ; 1 , L
0 ; 0 , L
x ; $, R
□ ; 1 , L
0 ; 0 , R
1 ; 1 , R
$; $, R
□ ; □ , R
0 ; □ , R
$; Z , R
x ; x , R
$; $, R
1 ; 1 , R
0 ; 0 , R
$; Z , R
□ ; 0 , L
x ; x , L
1 ; x , R
0 ; 0 , R
1 ; 1 , R
x ; x , R
0 ; 0 , L
x ; x , L
$; $, L
Z ; Z , R
0 ; □ , L
0 ; 0 , L
1 ; 1 , L
□ ; 1 , L
Z ; □ , L
□ ; □ , R
0 ; 0 , R
x ; 1 , R
1 ; 1 , L
1 ; 1 , L
0 ; 1 , R
1 ; 1 , R
1 ; 1 , L
□ ; □ , R
□ ; □ , R
1 ; □ , L
□ ; □ , L