Problem 1

(a) We proceed by defining separate CFGs for the two parts of the language and then combining them.

Let the CFG $G = (V, \Sigma, R, S)$ be defined as follows:

Variables: $V = \{S, S_1, S_2, L_3, L_4, C, D\}$

Terminals: $\Sigma = \{0, 1, \#\}$

Start Variable: S

Production Rules R:

Start Rule:

$$S \to S_1 \mid S_2$$

Productions for S_1 (strings where $|x| \neq |y|$):

$$\begin{aligned} S_1 &\to L_3 \mid L_4 \\ L_3 &\to C\ D\ \#\ D \\ L_4 &\to D\ \#\ C\ D \end{aligned}$$

Here, C generates non-empty strings, ensuring $|x| > |y|$ in L_3 and $|x| < |y|$ in L_4.

Productions for C and D:

$$\begin{aligned} C &\to 0 \mid 1 \mid 0C \mid 1C \\ D &\to \varepsilon \mid 0D \mid 1D \end{aligned}$$

Productions for S_2 (strings where $x = y^R$):

$$S_2 \to 0\ S_2\ 0 \mid 1\ S_2\ 1 \mid \#$$

This generates reverse strings with '#' in the middle.

S: Generates all strings in L.

S_1: Generates strings $x\#y$ where $|x| \neq |y|$.

S_2: Generates strings $x\#y$ where $x = y^R$.

L_3: Generates strings where $x = CD$, $y = D$, ensuring $|x| > |y|$.

L_4: Generates strings where $x = D$, $y = CD$, ensuring $|x| < |y|$.

C: Generates all non-empty strings over $\{0, 1\}$.

D: Generates all strings (including empty string) over $\{0, 1\}$.

The grammar is unambiguous. This is because: Strings where $x = y^R$ are generated exclusively by S_2. Strings where $|x| \neq |y|$ are generated exclusively by S_1. There is no overlap between S_1 and S_2 since $x = y^R$ implies $|x| = |y|$, and strings with $|x| \neq |y|$ cannot be palindromes centered at '#'. Therefore, every string in L has exactly one derivation in the grammar, we can draw the conclusion that the grammar is unambiguous.

Jingheng Huan

(b) Let the CFG $G = (V, \Sigma, R, S)$ be defined as follows:
Variables: $V = \{S\}$
Terminals: $\Sigma = \{a, b, c\}$
Start Variable: S
Production Rules R:

$$S \to a\ S\ c\ \mid\ b\ S\ c\ \mid\ \varepsilon$$

S: Generates all strings in L where the number of c's equals the total number of a's and b's.

This structure guarantees that each 'a' or 'b' before the c's is matched with a 'c' after the S in the production. Therefore, the total number of c's (p) will always equal the number of a's (m) plus the number of b's (n).

The grammar is unambiguous. Each production choice corresponds directly to the next symbol in the string. There is no alternative way to derive a given string because the positions of 'a' and 'b' before the 'c's dictate the derivation path. The recursive nature of the grammar ensures a one-to-one mapping between strings and their derivations.

Collaborators: None