CS 2051: Honors Discrete Mathematics
Spring 2023 `generate_cfg_tricky` Solution

Sarthak Mohanty

In this document, we cover the solution to the `generate_cfg_tricky` function. Recall that our goal in this function was to generate a context-free grammar (CFG) for the language

$$L = \{1^i 0^j : 2i \neq 3j + 1\}.$$

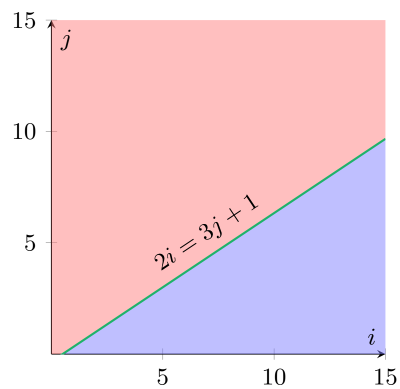
(Unless otherwise indicated, $i, j \in \mathbb{N}$ should always be assumed.)

Let's kick off our analysis by exploring the language from a geometric perspective. First, map each string of the form $1^i 0^j$ to points in a discrete 2D space, where the x-axis represents the number of 1's (i) and the y-axis represents the number of 0's (j). Next, consider the *complement* of language L:

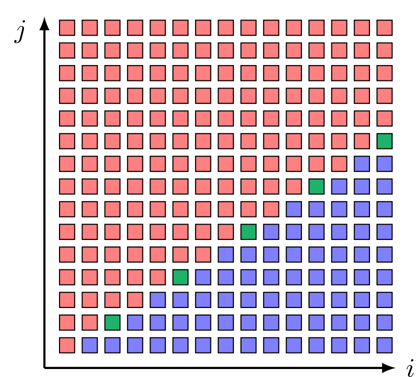
$$\bar{L} = \{1^i 0^j : 2i = 3j + 1\}.$$

If we view these two languages through the lens of a **linear classification** problem, the linear condition for $\bar{L}$ (i.e., $2i = 3j + 1$) can be seen as a decision boundary that separates points in the 2D space. With this in mind, the language L can be seen as the set of regions that result from this decision boundary, as depicted in Figure (1).

This new interpretation also gives us a strategy for generating L: construct a CFG for each of the two halfspaces, then combine them to create a CFG deciding their union. Before we start this process, though, we'll make a quick detour to create a CFG for $\bar{L}$, as the techniques used to do so are similar to those in the primary construction.



(a) Decision boundary applied to the Cartesian plane.



(b) Decision boundary applied to a 2D grid representing the strings in the language.

Figure 1: Geometric visualization of languages L and $\bar{L}$: the decision boundary $2i = 3j + 1$ divides the plane into two distinct halfspaces.

Warm-up: CFG for $\bar{L} = \{1^i 0^j : 2i = 3j + 1\}$

First, let's look at the values for i, j that satisfy our conditions. From Figure (1b), we can amass the following table of values:

i	2	5	8	11	14
j	1	3	5	7	9

A pattern begins to emerge: three right, two up, three right, two up .... Already we can guess that any satisfactory string $1^i 0^j$ must be of the form $1^{3k+2}0^{2k+1}$. We can prove it as well!

Claim: Let $k \in \mathbb{N}$. Then

$$\left\{1^i 0^j : 2i = 3j + 1\right\} = \left\{1^{3k+2}0^{2k+1}\right\}.$$

Proof. ($\subseteq$) The condition $2i = 3j + 1$ tells us that $3j + 1$ is an even number, which in turn implies j is odd. Thus by definition, $j = 2k + 1$ for some $k \in \mathbb{N}$. Plugging this back into the original condition, we obtain

$$2i = 3(2k + 1) + 1 \implies i = 3k + 2.$$

($\supseteq$) Suppose $i = 3k + 2$ and $j = 2k + 1$. Then

$$2i = 2(3k + 2) = 6k + 4 = 3(2k + 1) + 1 = 3j + 1.$$

$\square$

Now generating a CFG for $\bar{L} = \left\{1^{3k+2}0^{2k+1}\right\}$ is easy:

$$S \to 111S000 \mid 11S0.$$

Main Result: A CFG for $L = \{1^i 0^j : 2i \neq 3j + 1\}$

We generate a CFG for L using the strategy mentioned in the beginning of the document. To be specific, we split up L as

$$L = \underbrace{\{1^i 0^j : 2i > 3j + 1\}}_{L_1} \cup \underbrace{\{1^i 0^j : 2i < 3j + 1\}}_{L_2}$$

and generate CFGs for L_1 and L_2.

Part 1: CFG for $L_1 = \{1^i 0^j : 2i > 3j + 1\}$

Similar to the warm-up, we first convert our language into a more manageable form. Consider the set

$$M_1 = \{(i, j) \in L_1 : \nexists (i', j) \in L_1 \text{ such that } i' < i\}.$$

In simpler terms, M_1 consists of the "leftmost" (closest to the j-axis) cells in L_1. Writing these values down, we get the following table:

i	1	3	4	6	7	9	10	12	13
j	0	1	2	3	4	5	6	7	8

A new pattern begins to emerge (based on the parity of j): we see the same staircase pattern as in $\bar{L}$, except now it is repeated twice! In particular, we claim that any $(i, j) \in M_1$ is of the form $(3k + 1, 2k)$ or $(3k+3, 2k+1)$. In turn, this implies any $1^i 0^j \in L_1$ is of the form $1^{3k+1+c}0^{2k}$ or $1^{3k+3+c}0^{2k+1}$. An illustration of this is shown in Figure (2). [1]

[1] At this point, if I were a student, I would go straight to constructing the appropriate CFG. Woefully, I am not a student anymore!

Claim. Let $k, c \in \mathbb{N}$. Then

$$\left\{1^i 0^j : 2i < 3j + 1\right\} = \left\{1^{3k+1+c} 0^{2k}\right\} \cup \left\{1^{3k+3+c} 0^{2k+1}\right\}.$$

Proof. ($\subseteq$) Suppose we have a string that satisfies the inequality $2i > 3j + 1$. Rewriting in terms of i, we have $i > \frac{3}{2}j + \frac{1}{2}$. We now have two cases based on the parity of j.

Case 1. Suppose j is even. Then by definition, $j = 2k$ for some $k \in \mathbb{Z}$. Plugging this into the condition, we have $i > 3k + \frac{1}{2}$. Since i must be an integer, the smallest value for i that satisfies this inequality is $i = 3k + 1$. Then we have $i = 3k + 1 + c$, for some natural number c. In this case, the string is of the form $1^{3k+1+c} 0^{2k}$.

Case 2. Suppose j is odd. Then it can be written as $j = 2k + 1$, so we can express the inequality as $i > 3k + \frac{3}{2}$. The smallest value for i satisfying this inequality is $i = 3k + 2$. Therefore, any i can be expressed in the form $i = 3k + 2 + c$, for some $c \in \mathbb{N}$. In this case, the string is of the form $1^{3k+3+c} 0^{2k+1}$.

($\supseteq$) If $i = 3k + 1 + c$ and $j = 2k$, then

$$2i = 2(3k + 1 + c) = 6k + 2 + 2c > 6k + 1 = 3(2k) + 1 = 3j + 1.$$

On the other hand, if $i = 3k + 3 + c$ and $j = 2k + 1$, then

$$2i = 2(3k + 3 + c) = 6k + 6 + 2c > 6k + 4 = 3(2k + 1) + 1 = 3j + 1.$$

$\square$

We can now directly convert this alternate formulation of L_1 into a context-free grammar:

$$\begin{aligned} S &\to S_1 \mid S_2 \\ S_1 &\to 111 S_1 00 \mid 1 S_1 \mid 1 \\ S_2 &\to 111 S_2 00 \mid 1 S_2 \mid 1110 \end{aligned}$$

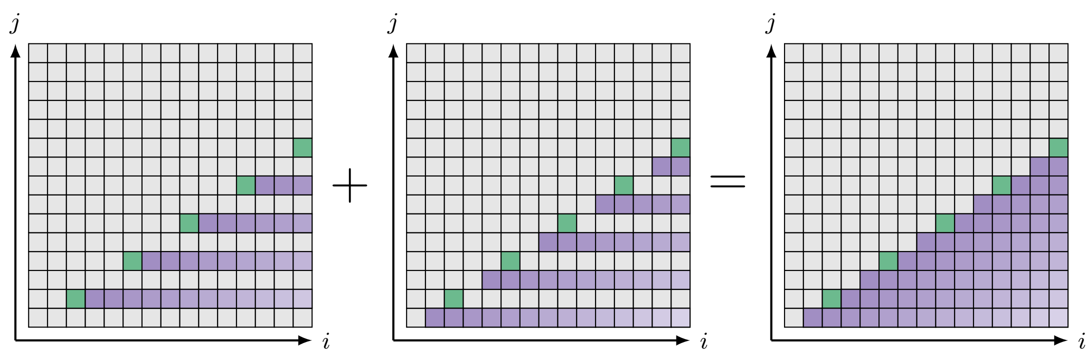


Figure 2: An alternative visualization of L_1

Part 2: CFG for $L_2 = \{1^i 0^j : 2i < 3j + 1\}$

We employ a similar strategy as in Part 1: consider the set M_2 consisting of the "bottom-most" elements in L_2. Formally,

$$M_2 = \{(i, j) \in L_2 : \nexists (i, j') \in L_2 \text{ such that } j' < j\}.$$

Again, we begin by writing values down:

i	0	1	2	3	4	5	6	7	8
j	0	1	2	2	3	4	4	5	6

Using the table, we guess that any string in L_2 must be of the form $1^{3k}0^{2k+c}$, $1^{3k+1}0^{2k+1+c}$, or $1^{3k+2}0^{2k+2+c}$ (Illustrated in Figure 3).

Claim. Let $k, c \in \mathbb{N}$. Then

$$\{1^i 0^j : 2i < 3j+1\} = \left\{1^{3k}0^{2k+c}\right\} \cup \left\{1^{3k+1}0^{2k+1+c}\right\} \cup \left\{1^{3k+2}0^{2k+2+c}\right\}.$$

Proof. ($\subseteq$) Suppose we have a string that satisfies the inequality $2i < 3j+1$. Rewriting in terms of j yields $j > \frac{2}{3}i - \frac{1}{3}$. We consider three cases, based on the value of $i \pmod 3$:

Case 1. Suppose $i \pmod 3 = 0$. Then $i = 3k$ for some $k \in \mathbb{N}$. Plugging this into the condition above, we get $j > 2k - \frac{1}{3}$. The smallest integer value for j satisfying this condition is $j = 2k$, so any satisfactory j can be expressed in the form $j = 2k + c$, for some natural number c.

In this case, the string is of the form $1^{3k}0^{2k+c}$.

Case 2. Suppose $i \pmod 3 = 1$. Then $i = 3k+1$ for some $k \in \mathbb{N}$. Plugging this into the condition above, we get $j > 2k + \frac{1}{3}$. The smallest integer value for j satisfying this condition is $2k+1$, any satisfactory j is of the form $j = 2k+1+c$.

In this case, the string is of the form $1^{3k+1}0^{2k+1+c}$.

Case 3. Suppose $i \pmod 3 = 2$. Then $i = 3k+2$ for some $k \in \mathbb{N}$. Plugging this into the condition above, we get $j > 2k+1$. The smallest integer value for j satisfying this condition is $2k+2$, so any satisfactory j is of the form $j = 2k+2+c$.

In this case, the string is of the form $1^{3k+2}0^{2k+2+c}$.

($\supseteq$) Left to the reader (similar to Part 1). $\square$

Again, it is now easy to convert this language into a context-free grammar:

$$\begin{aligned}
S &\to S_1 \mid S_2 \mid S_3 \\
S_1 &\to 111S_100 \mid S_10 \mid \epsilon \\
S_2 &\to 111S_100 \mid S_20 \mid 10 \\
S_3 &\to 111S_200 \mid S_30 \mid 1100
\end{aligned}$$

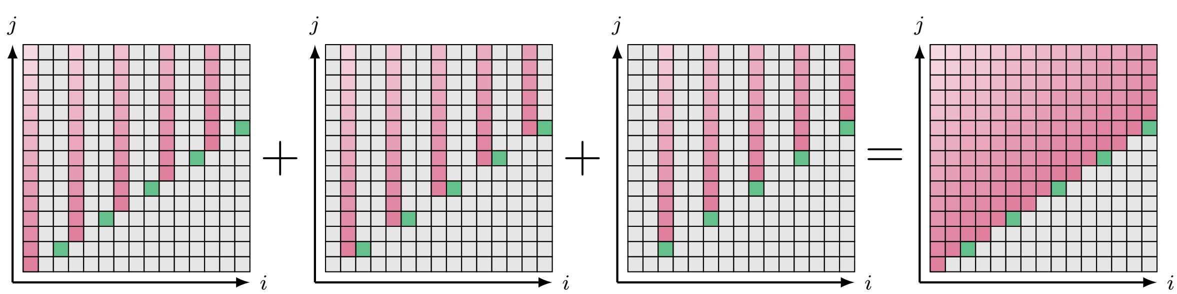


Figure 3: An alternative visualization of L_2

Part 3: Putting it all Together

Now that we have generated CFGs for L_1 and L_2, we combine them together to create our CFG for L, as shown in Figure (4). Our final CFG is

$$\begin{aligned} S &\to S_1 \mid S_2 \mid S_3 \mid S_4 \mid S_5 \\ S_1 &\to 111S_100 \mid 1S_1 \mid 1 \\ S_2 &\to 111S_200 \mid 1S_2 \mid 1110 \\ S_3 &\to 111S_300 \mid S_30 \mid \epsilon \\ S_4 &\to 111S_400 \mid S_40 \mid 10 \\ S_5 &\to 111S_500 \mid S_50 \mid 1100 \end{aligned}$$

There are ways to optimize this CFG so there are less variables, less rules, etc., but the main idea(s) should remain the same.

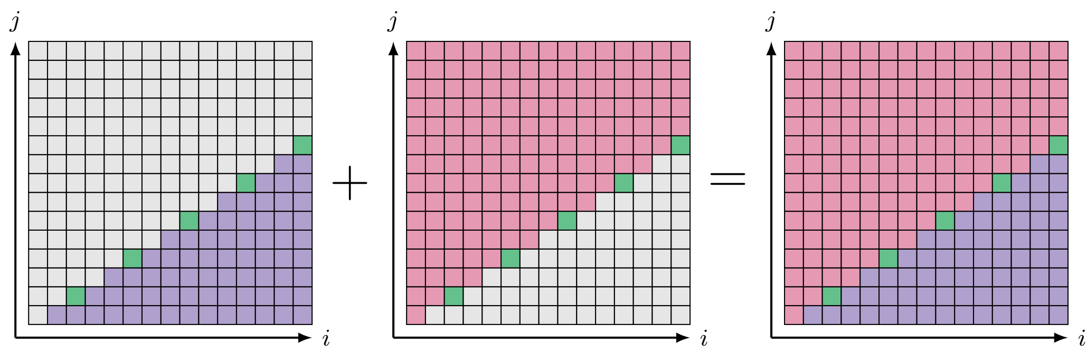


Figure 4: Unioning L_1 and L_2 results in L.