

Give context-free grammars for each of the following languages. For each grammar, describe in English the language for each non-terminal, and in the examples above. As usual, we won't get to all of these in section.

1.  $\{\mathbf{0}^{2n}\mathbf{1}^n \mid n \geq 0\}$

**Solution:**  $S \rightarrow \varepsilon \mid \mathbf{00S1}$ . ■

2.  $\{\mathbf{0}^m\mathbf{1}^n \mid m \neq 2n\}$

[Hint: If  $m \neq 2n$ , then either  $m < 2n$  or  $m > 2n$ . Extend the previous grammar, but pay attention to parity. This language contains the string  $\mathbf{01}$ .]

**Solution:** To simplify notation, let  $\Delta(w) = \#(\mathbf{0}, w) - 2\#(\mathbf{1}, w)$ . Our solution follows the following logic. Let  $w$  be an arbitrary string in this language.

- Because  $\Delta(w) \neq 0$ , then either  $\Delta(w) > 0$  or  $\Delta(w) < 0$ .
- If  $\Delta(w) > 0$ , then  $w = \mathbf{0}^i z$  for some integer  $i > 0$  and some suffix  $z$  with  $\Delta(z) = 0$ .
- If  $\Delta(w) < 0$ , then  $w = x\mathbf{1}^j$  for some integer  $j > 0$  and some prefix  $x$  with either  $\Delta(x) = 0$  or  $\Delta(x) = 1$ .
- Substrings with  $\Delta = 0$  is generated by the previous grammar; we need only a small tweak to generate substrings with  $\Delta = 1$ .

Here is one way to encode this case analysis as a CFG. The nonterminals  $M$  and  $L$  generate all strings where the number of  $\mathbf{0}$ s is More or Less than twice the number of  $\mathbf{1}$ s, respectively. The last nonterminal generates strings with  $\Delta = 0$  or  $\Delta = 1$ .

$$\begin{array}{ll} S \rightarrow M \mid L & \{\mathbf{0}^m\mathbf{1}^n\} m \neq 2n \\ M \rightarrow \mathbf{0}M \mid \mathbf{0}E & \{\mathbf{0}^m\mathbf{1}^n\} m > 2n \\ L \rightarrow L\mathbf{1} \mid E\mathbf{1} & \{\mathbf{0}^m\mathbf{1}^n\} m < 2n \\ E \rightarrow \varepsilon \mid \mathbf{0} \mid \mathbf{00E1} & \{\mathbf{0}^m\mathbf{1}^n\} m = 2n \text{ or } 2n + 1 \end{array}$$

Here is a different correct solution using the same logic. We either identify a non-empty prefix of  $\mathbf{0}$ s or a non-empty prefix of  $\mathbf{1}$ s, so that the rest of the string is as “balanced” as possible. We also generate strings with  $\Delta = 1$  using a separate non-terminal.

$$\begin{array}{ll} S \rightarrow AE \mid EB \mid FB & \{\mathbf{0}^m\mathbf{1}^n\} m \neq 2n \\ A \rightarrow \mathbf{0} \mid \mathbf{0}A & \mathbf{0}^+ = \{\mathbf{0}^i\} i \geq 1 \\ B \rightarrow \mathbf{1} \mid \mathbf{1}B & \mathbf{1}^+ = \{\mathbf{1}^j\} j \geq 1 \\ E \rightarrow \varepsilon \mid \mathbf{00E1} & \{\mathbf{0}^m\mathbf{1}^n\} m = 2n \\ F \rightarrow \mathbf{0}E & \{\mathbf{0}^m\mathbf{1}^n\} m = 2n + 1 \end{array}$$

Alternatively, we can separately generate all strings of the form  $\mathbf{0}^{\text{odd}}\mathbf{1}^*$ , so that we

don't have to worry about the case  $\Delta = 1$  separately.

$$\begin{array}{ll}
 S \rightarrow D \mid M \mid L & \{0^m 1^n\} m \neq 2n \\
 D \rightarrow 0 \mid 00D \mid D1 & \{0^m 1^n\} m \text{ is odd} \\
 M \rightarrow 0M \mid 0E & \{0^m 1^n\} m > 2n \\
 L \rightarrow L1 \mid E1 & \{0^m 1^n\} m < 2n \text{ and } m \text{ is even} \\
 E \rightarrow \varepsilon \mid 00E1 & \{0^m 1^n\} m = 2n
 \end{array}$$

■

**Solution:** Intuitively, we can parse any string  $w \in L$  as follows. First, remove the first  $2k$  0s and the last  $k$  1s, for the largest possible value of  $k$ . The remaining string cannot be empty, and it must consist entirely of 0s, entirely of 1s, or a single 0 followed by 1s.

$$\begin{array}{ll}
 S \rightarrow 00S1 \mid A \mid B \mid C & \{0^m 1^n\} m \neq 2n \\
 A \rightarrow 0 \mid 0A & 0^+ \\
 B \rightarrow 1 \mid 1B & 1^+ \\
 C \rightarrow 0 \mid 0B & 01^+
 \end{array}$$

Lets elaborate on the above, since  $k$  is maximal,  $w = 0^{2k}w'1^k$ . If  $w'$  starts with 00, and ends with a 1, then we can increase  $k$  by one. As such,  $w'$  is either in  $0^+$  or  $1^+$ . If  $w'$  contains both 0s and 1s, then it can contain only a single 0, followed potentially by  $1^+$ . We conclude that  $w' \in 0^+ + 1^+ + 01^+$ .

■

3.  $\{0, 1\}^* \setminus \{0^{2n}1^n \mid n \geq 0\}$

[Hint: Extend the previous grammar. What is missing?]

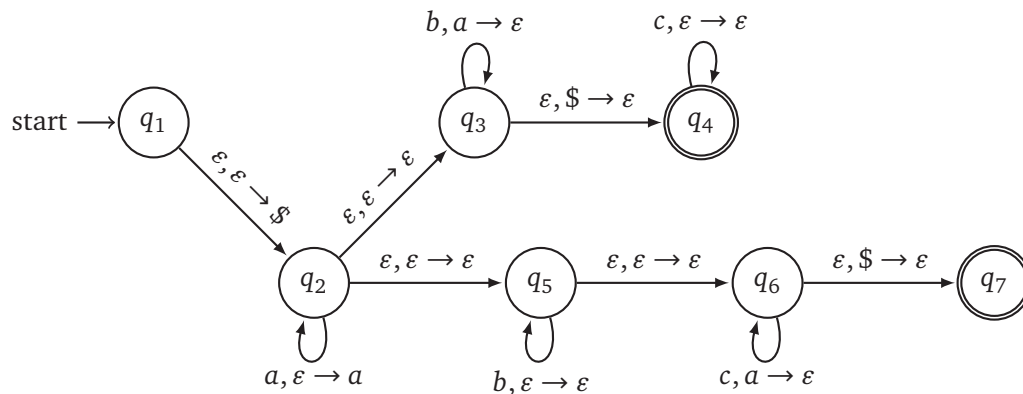
**Solution:** This language is the union of the previous language and the complement of  $0^*1^*$ , which is  $(0 + 1)^*10(0 + 1)^*$ .

$$\begin{array}{ll}
 S \rightarrow T \mid X & \{0, 1\}^* \setminus \{0^{2n}1^n\} n \geq 0 \\
 T \rightarrow 00T1 \mid A \mid B \mid C & \{0^m 1^n\} m \neq 2n \\
 A \rightarrow 0 \mid 0A & 0^+ \\
 B \rightarrow 1 \mid 1B & 1^+ \\
 C \rightarrow 0 \mid 0B & 01^+ \\
 X \rightarrow Z10Z & (0 + 1)^*10(0 + 1)^* \\
 Z \rightarrow \varepsilon \mid 0Z \mid 1Z & (0 + 1)^*
 \end{array}$$

■

The next few problems deal with push-down automata (PDA). The goal of these problems is to simply gain an understanding of PDAs which are the machines needed to recognize a context-free language:

4. What language does the following push-down automata recognize (Hint: This is a non-deterministic automata as most PDAs are)?



**Solution:** This example illustrates a pushdown automata that recognizes the language:

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\} \quad (1)$$

Informally the PDA for this language works by first reading and pushing the  $a$ 's. When the  $a$ 's are done, the machine has all of them on the stack so that it can match them with either the  $b$ 's or  $c$ 's. This maneuver is a bit tricky because the machine doesn't know in advance whether to match the  $a$ 's with the  $b$ 's or  $c$ 's. Nondeterminism comes in handy here.

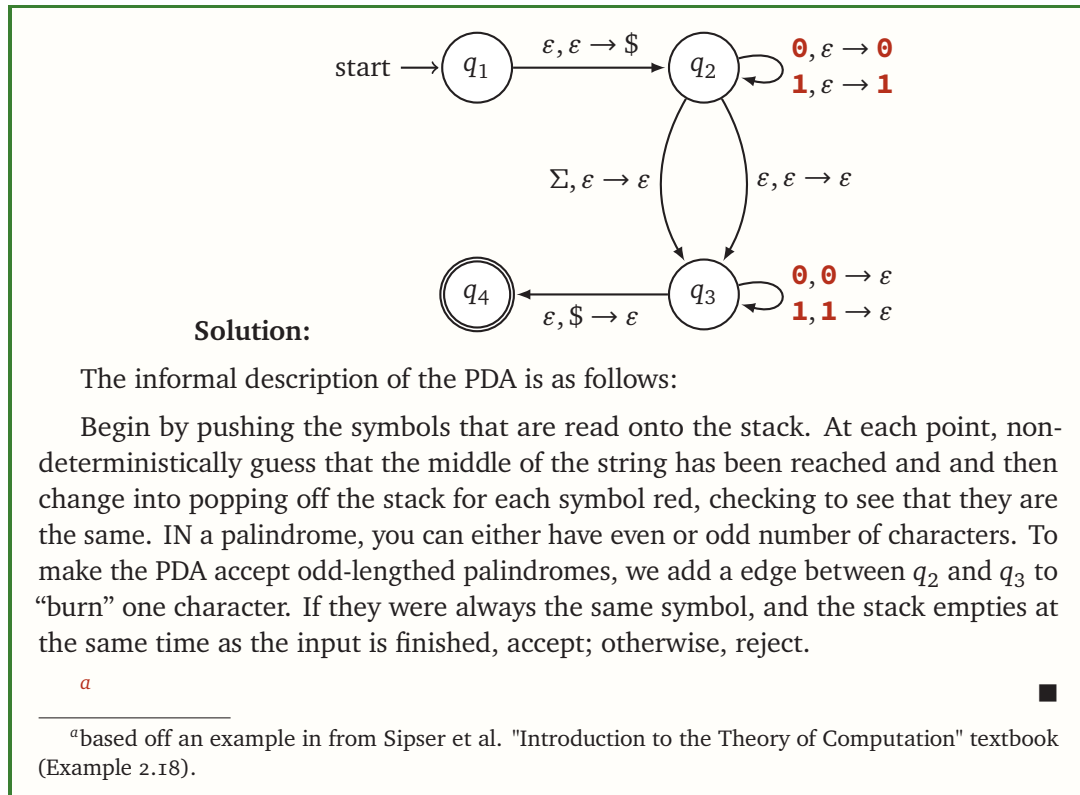
Using its nondeterminism, the PDA can guess whether to match the  $a$ 's with the  $b$ 's or with the  $c$ 's as shown above. Think of the machine as having two branches of its nondeterminism, one for each possible guess. If either of them matches, that branch accepts and the entire machine accepts.

$a$

<sup>a</sup>Solution borrowed from Sipser et al. "Introduction to the Theory of Computation" textbook (Figure 2.17). ■

5. Develop the PDA for the language:

$$L = \{w \text{ is a palindrome and } w \in \{0, 1\}^*\} \quad (2)$$



## Work on these later:

6.  $\{w \in \{0, 1\}^*\} \#(0, w) = 2 \cdot \#(1, w)$  – Binary strings where the number of 0s is exactly twice the number of 1s.

**Solution:**  $S \rightarrow \varepsilon \mid SS \mid 00S1 \mid 0S1S0 \mid 1S00$ .

For any string  $w$ , let  $\Delta(w) = \#(0, w) - 2 \cdot \#(1, w)$ . To prove the correctness of the grammar, we should prove:

- Any string  $x$  generated by the grammar satisfies  $\Delta(x) = 0$ .
- Any string  $x$  such that  $\Delta(x) = 0$  can be generated by the grammar.

Part (a) is trivial, since all the production rules add one 1 and two 0s.

Part (b) can be proved by induction.

As the base case, any string  $w$  such that  $|w| \leq 3$  and  $\Delta(w) = 0$  can trivially be generated by the grammar.

Suppose that for some constant  $k$ , any string  $w$  such that  $|w| = k - 3$  and  $\Delta(w) = 0$  can be generated by the grammar.

To prove that the grammar can generate any string  $x$  with  $|x| = k$  and  $\Delta(x) = 0$ , there are two cases to consider.

First, suppose  $x$  can be split into two non-empty substrings  $x = y_1y_2$  such that  $\Delta(y_1) = \Delta(y_2) = 0$ . Then, by the inductive hypothesis,  $y_1$  and  $y_2$  can be generated by the grammar. Therefore,  $x$  can be generated by first applying the rule  $S \rightarrow SS$  and separately generating  $y_1, y_2$ .

Then, suppose  $x$  cannot be split into two substrings like the above. In other words, for any proper prefix  $p$  of  $x$ , we have either  $\Delta(p) > 0$  or  $\Delta(p) < 0$ . This case can be further split into three sub-cases.

- Suppose for every proper prefix  $p$  of  $x$ ,  $\Delta(p) > 0$ . In this case,  $x$  must start with 00 and end with 1, because otherwise there would be some proper prefix  $q$  of  $x$  such that  $\Delta(q) < 0$ : If the string starts with 1 or 01, then we have  $q = 1$  or  $q = 01$ . If the string starts with 00 and ends with 0, then we define  $q$  in the way that  $x = q0$ . Moreover, since  $x$  starts with 00 and ends with 1, if we define  $r$  in the way that  $x = 00r1$ , then  $|r| = k - 3$  and  $\Delta(r) = 0$ . Therefore, by the inductive hypothesis and the production rule  $S \rightarrow 00S1$ , we conclude that  $x$  can be generated.
- Suppose for every proper prefix  $p$  of  $x$ ,  $\Delta(p) < 0$ . In this case,  $x$  must start with 1 and end with 00, with the similar reasoning. If  $x$  starts with 0, then we have  $q = 0$ . If it ends with 10, 01, 11, then we can let  $q$  be the string obtained by removing two symbols at the end from  $x$ . Therefore, defining  $r$  in the way that  $x = 1r00$ , it is clear that we can generate  $x$  with  $S \rightarrow 1S00$ .
- Suppose there exists a constant  $m > 0$  such that for all proper prefix  $p$  of  $x$  with  $|p| \leq m$ ,  $\Delta(p) > 0$ , and for all proper prefix  $q$  of  $x$  with  $|q| > m$ ,  $\Delta(q) < 0$ . This is possible because we are comparing  $\#(0, w)$  and  $2 \cdot \#(1, w)$  (For example, consider a string 010, and let  $p = 0$  and  $q = 01$ ). In this case,  $x$  must start with

$\mathbf{0}$ , followed by a substring  $r_1$  such that  $\Delta(r_1) = 0$ , then followed by  $\mathbf{1}$ , followed by another substring  $r_2$  with  $\Delta(r_2) = 0$ , then end with a  $\mathbf{0}$ , so that we have  $x = \mathbf{0}r_1\mathbf{1}r_2\mathbf{0}$ . If  $x$  starts with  $\mathbf{1}$ , it violates the condition  $\Delta(p) > 0$ . If it is not followed by  $r_1\mathbf{1}$ , then there would be no  $q$  such that  $\Delta(p) < 0$ . If it is not followed by  $r_2\mathbf{0}$ , then there would be  $y_1, y_2$  such that  $x = y_1y_2$ ,  $\Delta(y_1) = \Delta(y_2) = 0$ . Therefore, with the inductive hypothesis and the production rule  $S \rightarrow 0S1S0$ , we conclude that we can generate  $x$ .

Since we were able to generate any string  $x$  of length  $k$  that satisfies  $\Delta(x) = 0$ , we conclude that the the grammar is correct. ■

7.  $\{\mathbf{0}, \mathbf{1}\}^* \setminus \{ww \mid w \in \{\mathbf{0}, \mathbf{1}\}^*\}$ .

[Anti-hint: The language  $\{ww \mid w \in \mathbf{0}, \mathbf{1}^*\}$  is **not** context-free. Thus, the complement of a context-free language is not necessarily context-free!]

**Solution:** All strings of odd length are in  $L$ .

Let  $w$  be any even-length string in  $L$ , and let  $m = |w|/2$ . For some index  $i \leq m$ , we have  $w_i \neq w_{m+i}$ . Thus,  $w$  can be written as either  $x\mathbf{1}y\mathbf{0}z$  or  $x\mathbf{0}y\mathbf{1}z$  for some substrings  $x, y, z$  such that  $|x| = i - 1$ ,  $|y| = m - 1$ , and  $|z| = m - i$ . We can further decompose  $y$  into a prefix of length  $i - 1$  and a suffix of length  $m - i$ . So we can write any even-length string  $w \in L$  as either  $x\mathbf{1}x'\mathbf{0}z$  or  $x\mathbf{0}x'\mathbf{1}z$ , for some strings  $x, x', z, z'$  with  $|x| = |x'| = i - 1$  and  $|z| = |z'| = m - i$ . Said more simply, we can divide  $w$  into two odd-length strings, one with a  $\mathbf{0}$  at its center, and the other with a  $\mathbf{1}$  at its center.

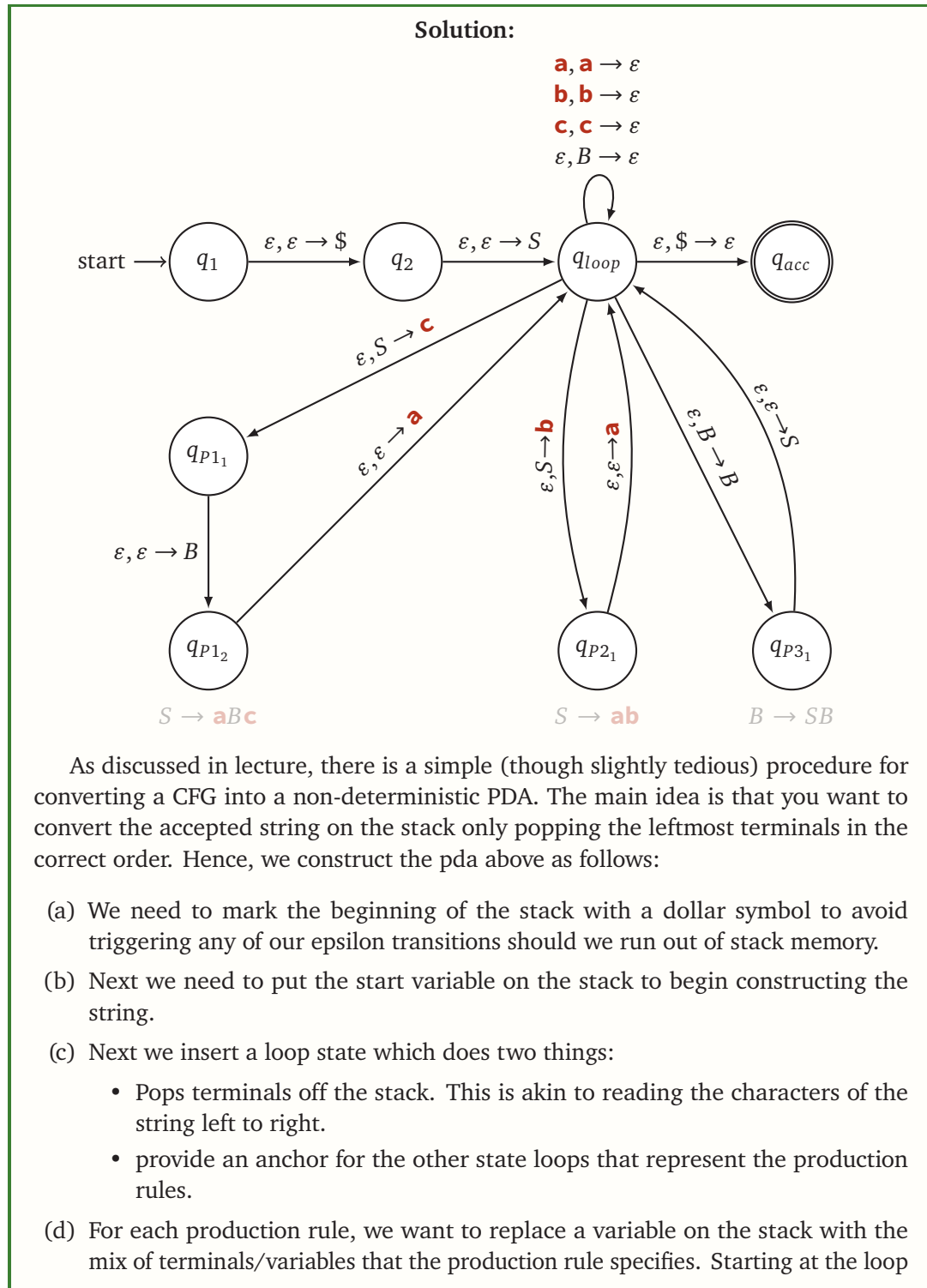
$S \rightarrow AB \mid BA \mid A \mid B$	strings not of the form $ww$
$A \rightarrow \mathbf{0} \mid \Sigma A \Sigma$	odd-length strings with $\mathbf{0}$ at center
$B \rightarrow \mathbf{1} \mid \Sigma B \Sigma$	odd-length strings with $\mathbf{1}$ at center
$\Sigma \rightarrow \mathbf{0} \mid \mathbf{1}$	single character

■

8. Convert the following CFG into a PDA:

$$S \rightarrow aBc \mid ab$$

$$B \rightarrow SB \mid \varepsilon$$



state, we create paths which describe all the production rules in the grammar.

- (e) Finally we add an accept state that is only reachable when the input is empty and the stack is clear of all terminals/variables.

*a*

---

<sup>a</sup>Inspired by the youtube video: [https://www.youtube.com/watch?v=ZImtQBMSW\\_Y](https://www.youtube.com/watch?v=ZImtQBMSW_Y)