# Homework 4

## CSC 445-01: Theory of Computation

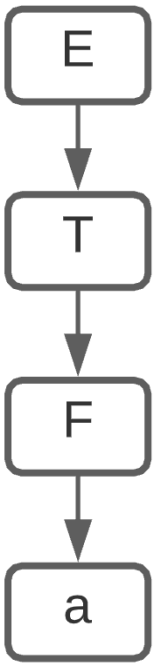
Matthew Mabrey, Luke Kurlandski

March 22, 2021

## 2.1

### a

The parse tree for $a$ is

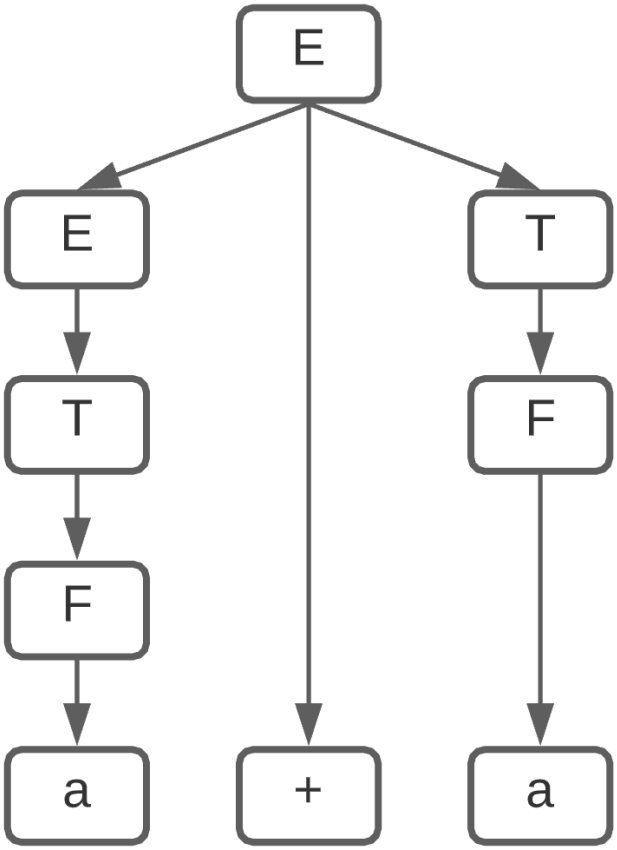



The derivation for $a$ is

$$E \Rightarrow T \Rightarrow F \Rightarrow a$$

**b**

The parse tree for $a + a$ is

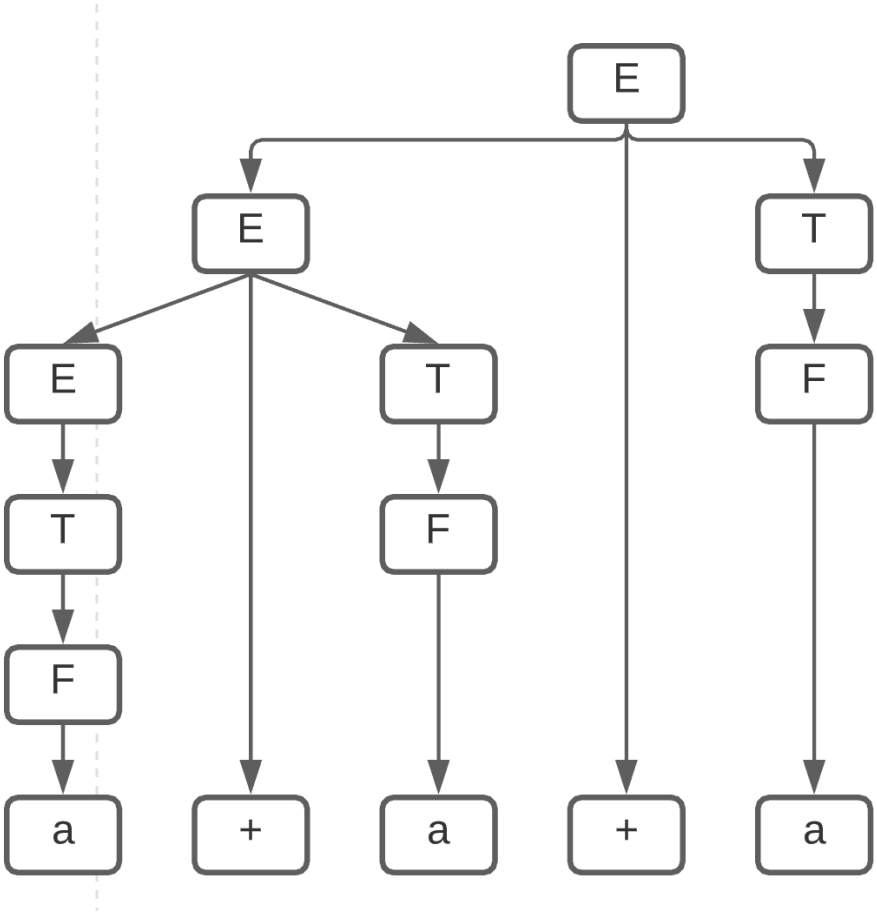



The derivation for $a + a$ is

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + F \Rightarrow a + a$$

**c**

The parse tree for $a + a + a$ is

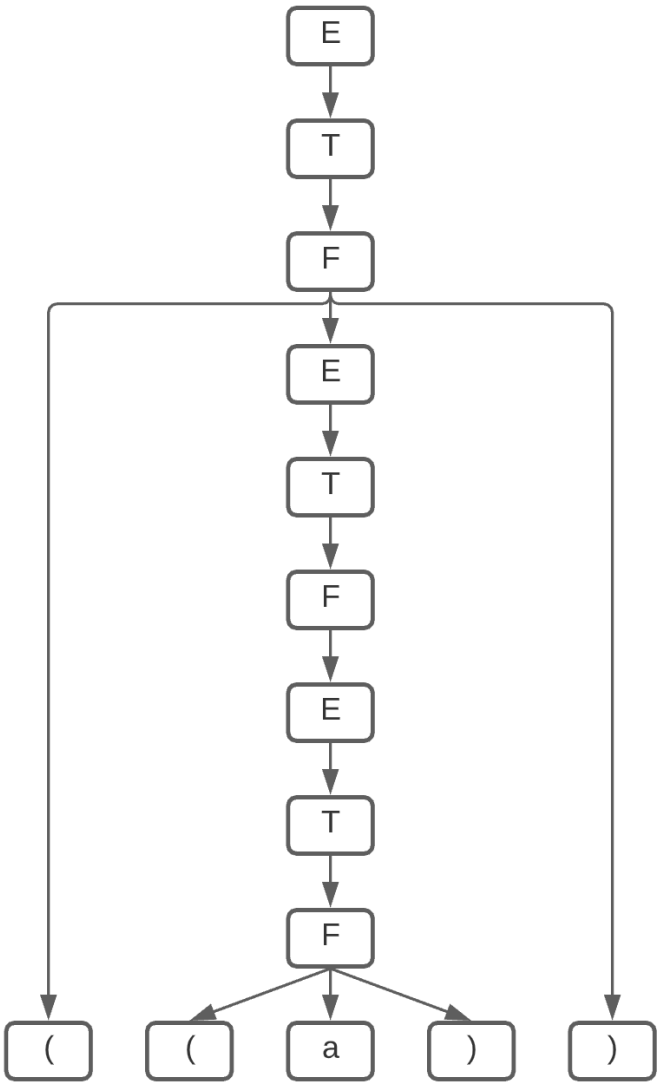



The derivation for $a + a + a$ is

$$E \Rightarrow E + T \Rightarrow E + T + T \Rightarrow T + T + T \Rightarrow F + T + T \Rightarrow a + T + T$$
$$\Rightarrow a + F + T \Rightarrow a + a + T \Rightarrow a + a + F \Rightarrow a + a + a$$

## d

The parse tree for $((a))$ is




The derivation for $((a))$ is

$$E \Rightarrow T \Rightarrow F \Rightarrow (E) \Rightarrow (T) \Rightarrow (F) \Rightarrow ((E)) \Rightarrow ((T)) \Rightarrow ((F)) \Rightarrow ((a))$$

## 2.4 (b)

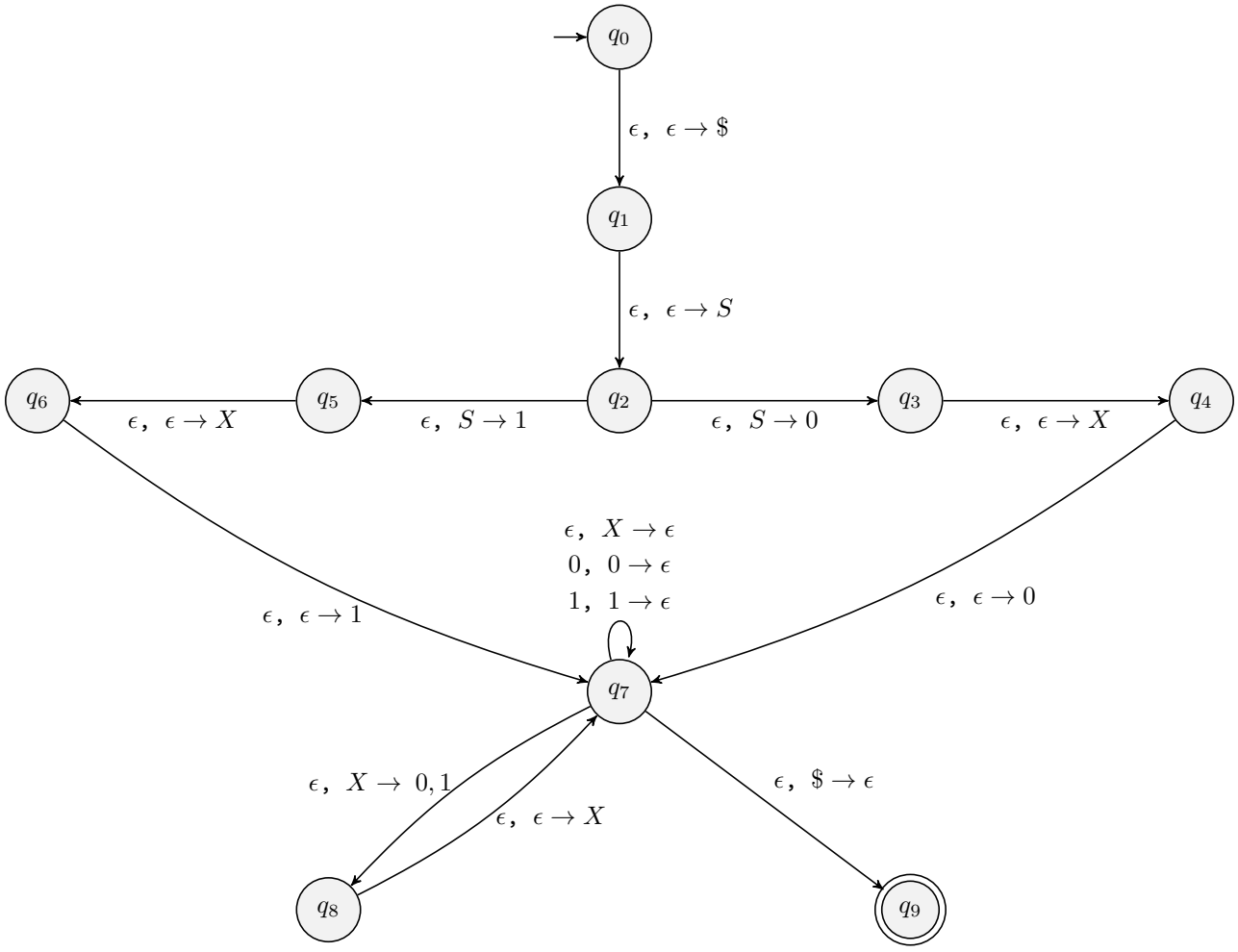
To generate the language $A = \{w \mid w \text{ starts and ends with the same symbol}\}$ given $\Sigma = \{0, 1\}$, we create the context-free grammar:

$$\begin{aligned} S &\rightarrow 0X0 \mid 1X1 \\ X &\rightarrow 0X \mid 1X \mid \epsilon \end{aligned}$$

## 2.5

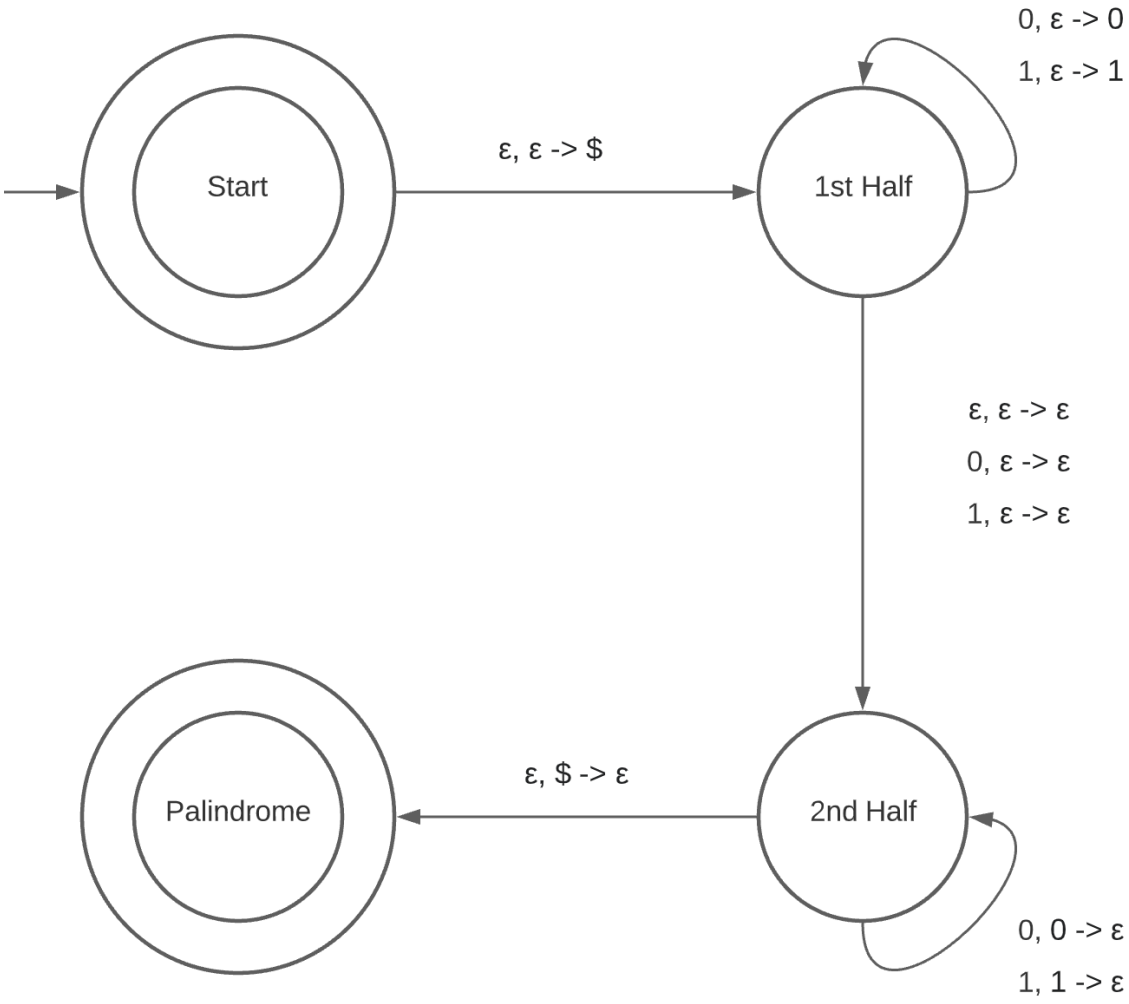
### b

The following state diagram represents the pushdown automata that accepts the context-free language $\{w \mid w$ starts and ends with the same symbol$\}$ given $\Sigma = \{0, 1\}$. We begin by pushing the special \$ symbol onto the stack to mark the bottom of the stack and then our start symbol $S$. From there we pop $S$ and either push $0X0$ or $1X1$ onto the stack to ensure our string begins and ends with the same symbol. The PDA can then continue adding 0's or 1's to the stack through the variable $X$ until the special \$ is reached. At this point the PDA pops the \$ symbol and enters the accept state as our stack is empty and it has successfully created a string that begins and ends with the same symbol.



**e**

Begin by placing $ on the stack. Read input and push the input symbols on the stack. This first phase is symbolic of reading the first half of the string. Nondeterministically, at some point of reading input, start comparing input symbols with symbols popped from the stack. If input symbol and stack symbol do not match, halt without accepting, else, continue. This second phase is symbolic of reading the second half of the string and checking that it matches the first half. Our PDA carefully handles this transition to account for even/odd numbered strings. Finally, if input symbol is $\epsilon$ and stack symbol is $, halt and accept.



## 2.6 (b)

The CFG that generates the complement for the language $A = \{a^n b^n \mid n \geq 0\}$ is such:

$$S \rightarrow A \mid bX \mid aXbXaX$$

$$X \rightarrow aX \mid bX \mid \epsilon$$

$$A \rightarrow aA \mid \epsilon$$

## 2.13 (a)

This grammar may produce a string that belongs to one of two different languages. The first language is that of zero or more 0s, followed by a #, followed by two times as many 0s that came before. The second language is that of zero or more 0s, followed by a #, followed by zero or more 0s, followed by a #, followed by zero or more 0s.

More formally,

$$L(G) = \{0^n \# 0^{2n} \mid n \geq 0\} \cup \{0^a \# 0^b \# 0^c \mid a, b, c \geq 0\}$$