Dustin Newman

CS 181, Discussion 1A

Campbell, Mathur

Assignment Three

1

$$R := \Sigma^* \circ ((a \circ (\Sigma\Sigma\Sigma)^* \circ a) \cup ((b \circ (\Sigma\Sigma\Sigma)^* \circ b) \cup ((c \circ (\Sigma\Sigma\Sigma)^* \circ c)) \circ \Sigma^*$$

To recognize identical characters separated by a multiple of three other characters, we specify three different regular expressions of the form $x(\Sigma\Sigma\Sigma)^*x$ where $x \in \Sigma$ e.g. $x := a$. This captures the first occurrence of the character, followed by any other characters (including that first character itself) in multiples of three (including zero), followed by the same first character. We then union together all three regular expressions and make sure to concatenate Σ^* to both ends of the overall regular expression, as any other characters can precede and follow our string.

2

a

Yes, the GNFA does recognize the string "ababba." It first matches "ab" in the transition $q_0 \rightarrow q_2$, then matches 0 occurrences of "a" in the transition $q_2 \rightarrow q_1$, then recognizes "ab" in the loop $q_1 \rightarrow q_1$. It then matches 0 occurrences of "aa" in the transition $q_1 \rightarrow q_2$ and ends in q_a by matching "ba" in the transition $q_2 \rightarrow q_a$. Since we end on an accepting state, we accept.

No, the GNFA does not recognize the string "aabbba." Taking the first transition $q_0 \rightarrow q_2$ by matching "a" (with zero occurrences of "b"), it arrives at q_2. If it matches "ab" with $q_2 \rightarrow q_a$, it blocks and fails to accept on the next input symbol. If it matches the empty string with $q_2 \rightarrow q_1$, it can match "ab" with the q_1 loop, but cannot match the remaining occurrences of "b" followed by a single "a."

b

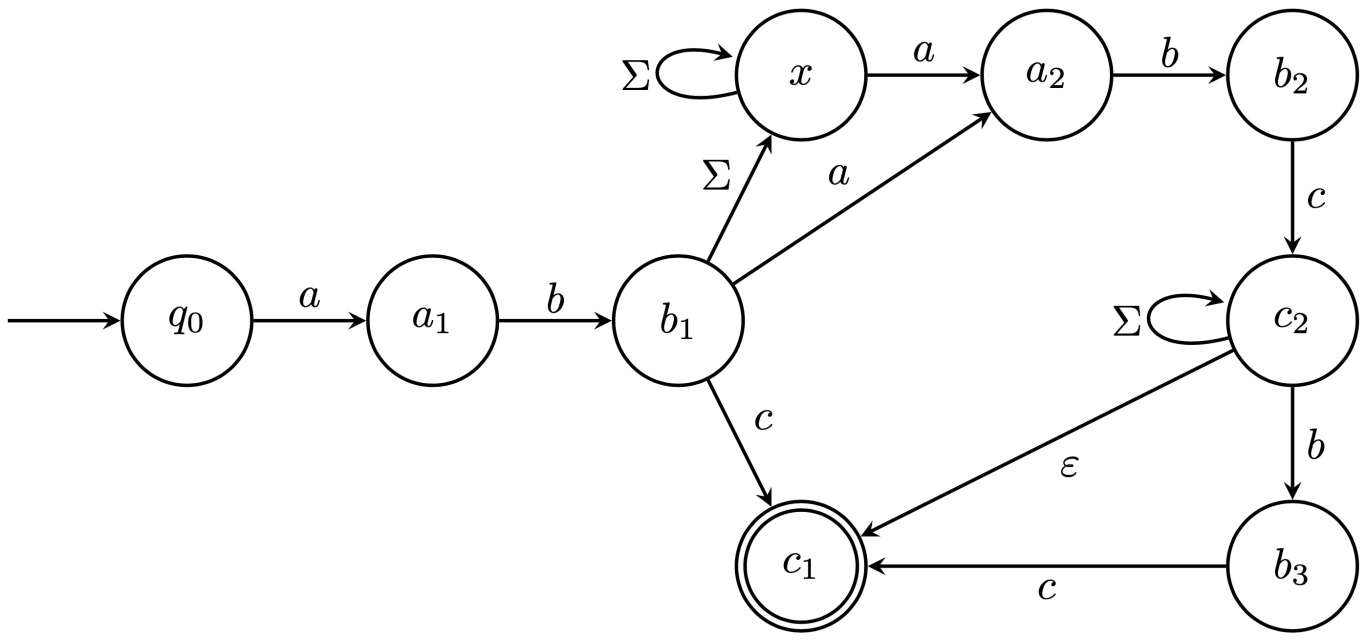
$$w_1, w_2 := \text{``abab''}$$

"abab" can be accepted by either going from q_2 directly to q_a or by going from q_2 to q_1 and then to q_a.

$$w_1 := q_0 \xrightarrow{ab} q_2 \xrightarrow{ab} q_a$$

$$w_2 := q_0 \xrightarrow{ab} q_2 \xrightarrow{a} q_1 \xrightarrow{b} q_a$$

3



The NFA above accepts the strings which begin with "ab." At that point, there are four possibilities: (1) a "c" is next and ends the string; (2) the beginning of the "abc" substring is next; (3) any other character is next; or (4) it is the end of the string. In the case of (4), we do not accept as the string does not end in "bc." In case of (1), we accept by moving to state c_1, which has no outgoing edges. Per the rules of NFAs, therefore c_1 must end the string or else it will fail to accept the string. In the case of (2) or (3), we move to the substring handling states. We use the "guess and check" view of NFAs to "wait" at state x until the "abc" substring starts. When we guess correctly, we move to a_2, b_2, c_2. Note that we now have two options: either the substring ends the string (which is still legal as "abc" ends in "bc") or the substring does not end the string. If the substring does end the string, we use the ε-transition to move to accepting state c_1. If we use the ε-transition with some other character following "c," thus forming an illegal string, we will block and fail to accept, as c_1 has no outgoing edges. If the substring does not end the string, we may still have a correct string if it ends in "bc" still. We "wait" on c_2 with a self-loop on any character, until we "guess" that we have seen the "bc" ending substring. Once we do so, we move to b_3 and then c_1, where we accept.

4

This language is all strings that either (1) are the empty string, (2) start with "0," or (3) start with "1" but contain no more than two consecutive zeroes. If the string starts with "0," there are no restrictions on it. e.g. "" (the empty string), "0", "0100010", "1," "100," "10010" are all accepted but "1000" and "10010001" are not.

5

$$
\begin{gathered}
G_5 := (V, \Sigma, R, S) \\
V := \{S, Z, Y\} \\
\Sigma := \{0, 1\} \\
R := \{S \to ZZSY, \\
S \to \varepsilon, \\
Z \to 0, \\
Y \to 1\}
\end{gathered}
$$

L_5 describes a set of strings with an even number of consecutive 0's followed by half as many consecutive 1's. So, our start variable S has just three variables: Z, representing the terminal symbol "0," Y, representing the terminal "1," and itself. We have two copies of Z for every copy of Y, ensuring that there are exactly twice as many 0's as 1's. We also have a recursive definition for S, allowing further terminals to be "pumped" into the middle, ensuring that the 0's are consecutive and immediately followed by the 1's.

6

We will prove that L_6 is not regular with a proof by contradiction.

Assume that L_6 is regular. Then, by the pumping lemma, there is some natural number p for all strings $w \in L_6$ of at least length p such that $w := xy^i z$ and $|xy| \leq p$ and $|y| > 0$.

Since we may choose any w parametrized by p, let $w := (111)^p 0^p$.

$w \in L_6$ because it contains three times as many 1's as it does 0's. $|w| \geq p$ because it contains 4 times as many symbols for every $p > 0$ and contains 0 symbols when $p := 0$ (and thus trivially $|w| = 0 = p$).

By the pumping lemma, there exists some x, y, z such that $w := xy^i z$ for all $i \geq 0$.

Because the length of xy must be no greater than p, we know that x and y must consist of some portion of the $(111)^p$ substring within w. Since there are three "1" symbols per p and $|xy| \leq p$, xy constitutes only some fraction of the "111" substring within w.

Therefore, we can more accurately describe $x := 1^m$ and $y := 1^n$ and $z := 1^{(p-n-m)}0^p$ where $(m+n) \leq p$ and $n > 0$.

No matter the partition of w into $xy^i z$, consider the case when $i := 2$. Then we have $w := xyyz := 1^m 1^n 1^n 1^{(p-n-m)} 0^p$. With simple arithmetic, we expect $3p = m + n + n + p - n - m$ per the definition of L_6. Reducing this, we see:

$$\begin{gathered} 3p = m + n + n + p - n - m \\ 2p = n \end{gathered}$$

However, this is impossible from our former definition of $(m+n) \leq p$. Even when $m := 0$, thus making n its maximum value, we get $n \leq p$. Since $n \leq p$, it is not possible for $2p = n$.

Since $3p \neq m + n + n + p - n - m$, our string $w \notin L_6$, since there are not three times as many 1's as there are 0's. However, this is a **contradiction** as we stipulated, per the pumping lemma, that there exists $w := xy^i z$ for *all* $i \geq 0$. Since $w \notin L_6$ for $i := 2$, we have a contradiction. Thus, L_6 is not regular.

$\therefore$ QED.