INT201 Assignment 1

Xinrong Li (ID: 2363123)

1 DFA for Strings Starting with `b` and Ending with `a`

We construct a deterministic finite automaton (DFA) for the language

$$L_1 = \{\, x \in \{a, b\}^* \mid x \text{ starts with "b" and ends with "a"} \,\}.$$

Recall that a DFA is formally defined as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states.

DFA Construction: Let $M_1 = (Q, \Sigma, \delta, q_0, F)$ with:

- $Q = \{q_0, q_1, q_2, q_d\}$, where q_0 is the start state and q_d is a dead (sink) state.
- $\Sigma = \{a, b\}$.
- Transition function δ defined as:
$$\begin{aligned} \delta(q_0, a) &= q_d, & \delta(q_0, b) &= q_1, \\ \delta(q_1, a) &= q_2, & \delta(q_1, b) &= q_1, \\ \delta(q_2, a) &= q_2, & \delta(q_2, b) &= q_1, \\ \delta(q_d, a) &= q_d, & \delta(q_d, b) &= q_d\ , \end{aligned}$$
  where q_d is a non-accepting sink that traps any string that does not meet the criteria.
- q_0 is the initial state.
- $F = \{q_2\}$, since q_2 represents having read a string that ends in `a` (and by construction also started with `b`).

Intuitively, q_1 signifies that the string read so far starts with `b` and the last symbol read was `b`, whereas q_2 signifies the string starts with `b` and the last symbol read was `a`. The state q_2 is accepting, ensuring the input ends in `a`. Any input that either fails to start with `b` or cannot end in `a` (e.g. it ended in `b`) will eventually reach the sink q_d and be rejected.

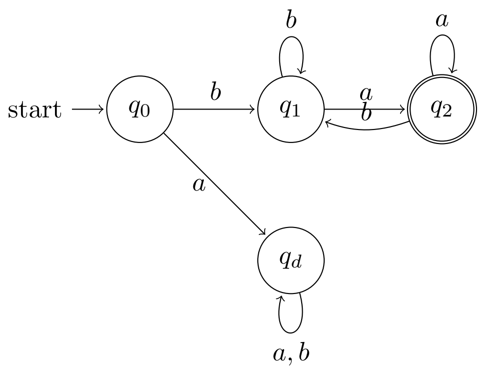

Figure 1: DFA M_1 for all strings over $\{a, b\}$ that start with `b` and end with `a`.

2 Converting an NFA to an Equivalent DFA

We convert the following NFA over $\Sigma = \{a, b, c\}$ to an equivalent DFA via the subset construction. The NFA has states $Q = \{q_0, q_1, q_2\}$, start state q_0, and accepting state q_2, with transitions:

$$q_0 \xrightarrow{\varepsilon} q_1, \quad q_1 \xrightarrow{\varepsilon} q_2, \quad q_0 \xrightarrow{a} q_0, \quad q_1 \xrightarrow{b} q_1, \quad q_2 \xrightarrow{c} q_2.$$

Its diagram is shown in Figure 2.

Subset construction (concise). The ε-closures are

$$\varepsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}, \quad \varepsilon\text{-closure}(q_1) = \{q_1, q_2\}, \quad \varepsilon\text{-closure}(q_2) = \{q_2\}.$$

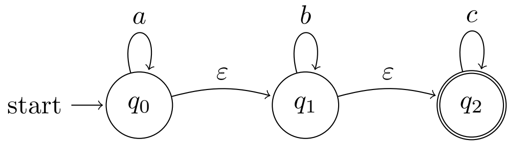

Figure 2: NFA M_2 used in Question 2.

Thus the reachable DFA states (as subsets of Q) are:

$$A = \{q_0, q_1, q_2\} \text{ (start)}, \quad B = \{q_1, q_2\}, \quad C = \{q_2\}, \quad D = \emptyset \text{ (dead)}.$$

Accepting DFA states are those containing q_2: namely A, B, C.

The transition function δ' of the DFA is:

	a	b	c
A	A	B	C
B	D	B	C
C	D	D	C
D	D	D	D

The resulting DFA is shown in Figure 3.

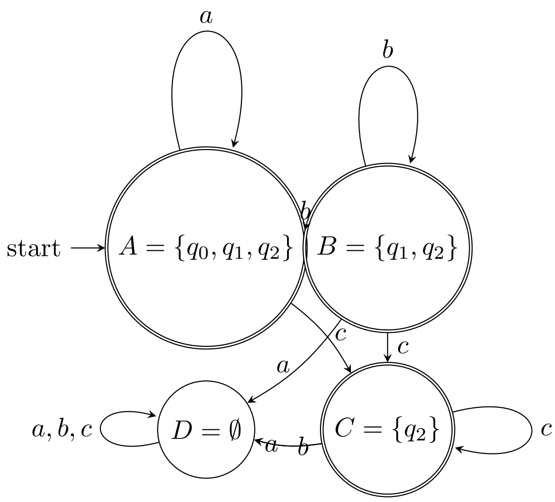

Figure 3: Equivalent DFA M_2' obtained by subset construction. States A, B, C are accepting.

3 NFA Construction from Regular Expression $(00)^*(11)$

We now convert the regular expression $r = (00)^*(11)$ into an NFA M_3 using Thompson's construction. The regex r denotes the language

$$L_3 = \{\, (00)^n\, 11 \mid n \geq 0 \,\}\,,$$

i.e. any string consisting of some even number of `0`'s (including zero `0`'s) followed by `11`.

Thompson's Construction: We build M_3 step by step:

1. For the sub-expression 00: create an NFA with states A (start) $\xrightarrow{0} B \xrightarrow{0} C$ (final for this sub-NFA).
2. Apply Kleene-* to 00: introduce a new start state I and new final state F. Add ε-transitions $I \xrightarrow{\varepsilon} A$ (enter the 00 sub-NFA) and $I \xrightarrow{\varepsilon} F$ (to allow zero occurrences). Also add ε-transitions $C \xrightarrow{\varepsilon} A$ (to loop back after one occurrence) and $C \xrightarrow{\varepsilon} F$ (to exit after one or more occurrences). Now I is the start and F is the (temporary) final for $(00)^*$.
3. For the sub-expression 11: create an NFA with states X (start) $\xrightarrow{1} Y \xrightarrow{1} Z$ (final).
4. Concatenate $(00)^*$ and (11): connect the former's final to the latter's start by an ε-transition $F \xrightarrow{\varepsilon} X$. In the combined NFA, I remains the initial state and Z is the sole accepting state. (Note: F is no longer an accepting state once concatenated, as a valid string must continue through the 11 part.)

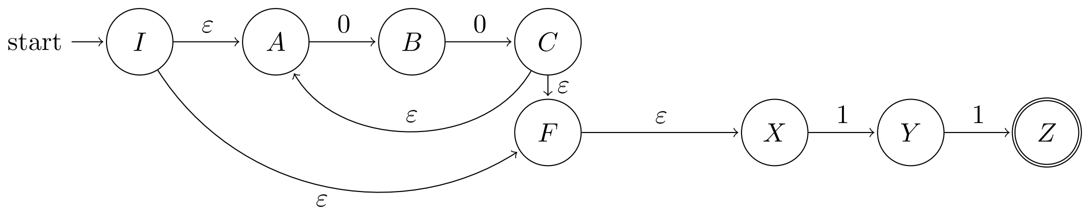

Figure 4: NFA M_3 constructed for the regular expression $(00)^*(11)$.

Figure 4 shows the resulting NFA. State I is the start and state Z (double circle) is the accepting state.

To verify correctness: from I to F, the machine can loop through $A \to B \to C$ any number of times, consuming pairs of `0`'s (each loop adds "00" to the string). The ε-transition $I \to F$ allows zero iterations of "00". After that, the $F \to X$ ε-move and the $X \to Y \to Z$ transitions consume `11`. Thus, the NFA accepts exactly those strings of the form $(00)^n 11$ for $n \geq 0$, as required.

4 Non-Regularity of $A_1 = \{www \mid w \in \{a,b\}^*\}$

Finally, we prove that the language

$$A_1 = \{\, www \mid w \in \{a,b\}^* \,\}$$

is *not* a regular language. Intuitively, A_1 consists of strings that can be divided into three equal parts, all identical. This is a classic example of a language that fails to meet the *pumping lemma* criteria for regular languages. We provide a formal proof using the Pumping Lemma for regular languages.

Theorem 1. *$A_1 = \{www \mid w \in \{a,b\}^*\}$ is not regular.*

Proof. Suppose, for sake of contradiction, that A_1 is regular. Then by the Pumping Lemma, there exists a pumping length $p \geq 1$ such that any string $s \in A_1$ with $|s| \geq p$ can be decomposed as $s = x\,y\,z$, with $|xy| \leq p$ and $|y| \geq 1$, and for all $i \geq 0$ the string $x\,y^i\,z$ is also in A_1.

Consider the specific string

$$s = w\,w\,w \in A_1, \qquad \text{where } w = a^p b^p.$$

Notice that $|w| = 2p$, so $|s| = 6p \geq p$. By construction, $s = a^p b^p\, a^p b^p\, a^p b^p$ (three copies of w). According to the lemma, s can be written as xyz with the stated properties. Since $|xy| \leq p$, the substring y lies entirely within the first p characters of s. The first p characters of s are a^p (all `a`). Thus, y consists only of the letter `a`; say $y = a^k$ for some $k \geq 1$.

Now, consider pumping y by taking $i = 2$. The pumped string is:

$$s' \;=\; x\,y^2\,z \;=\; a^{p+k} b^p\, a^p b^p\, a^p b^p\ ,$$

which has length $|s'| = 6p + k$.

If A_1 were regular, s' must also belong to A_1. This means s' should be expressible as $s' = t\,t\,t$ for some substring t. However, we will show this is impossible, leading to a contradiction:

- *Length mismatch:* If k is not a multiple of 3, then $|s'| = 6p + k$ is not divisible by 3. In that case, s' cannot be split into three equal-length parts at all (let alone three identical parts), so $s' \notin A_1$.
- *Content mismatch:* Suppose k *is* a multiple of 3 (say $k = 3m$) so that $|s'|$ is divisible by 3. Then we can partition s' into three substrings of equal length, each of length $2p + m$. However, these three substrings cannot all be equal. In s', the first $2p + m$ characters (the first third of s') include a longer prefix of `a`'s than the next $2p + m$ characters do. In fact, the first third of s' starts with a^{p+3m}, whereas by the time we reach the beginning of the second third of s', some of those `a`'s have been exhausted and replaced by `b`'s. Consequently, the second third of s' begins with a different symbol than the first third does (it begins partway through the block of `b`'s), so the first and second segments of s' are not identical. This means s' is not of the form $t\,t\,t$.

In both cases, we reach a contradiction: $xy^2z = s' \notin A_1$, despite $s \in A_1$. Therefore, our original assumption that A_1 is regular must be false. We conclude that A_1 is **not** a regular language. $\square$