1 Finite State Automata and Regular Language

The NFA A:

(i) The DFA B:

(ii) Our initial split differentiates non-final and final states:

\[ \left[ \left[ S_0, S_{01} \right], \left[ S_{02}, S_{012} \right] \right] \]

Testing the non-final group with \( b \) produces a split, as for \( S_0 \) we stay in the non-final group, while for \( S_{01} \) we go to the final group:

\[ \left[ \left[ S_0 \right], \left[ S_{01} \right], \left[ S_{02}, S_{012} \right] \right] \]

Testing the final group with \( a, b, c \) produces no split, as each leaves us still in the final group. Every other group is a singleton. Hence the algorithm terminates with the groups as above, and \( S_{02} \) and \( S_{012} \) are the only equivalent states.
(iii) We convert $B$ above into the minimal $C$ by deleting the state $S_{012}$ and re-assigning every arc into $S_{012}$ to go to $S_{02}$:

(iv) **Subgoal one:** $C$ accepts all strings $\gamma ab\delta$, for any $\gamma, \delta \in \{a, b, c\}^*$, i.e.,

\[ N^*(S_0, \gamma ab \delta) = S_{02} \]

**Subgoal two:** if $C$ accepts a string $w$, i.e., $N^*(S_0, w) = S_{02}$, then there exists $\gamma, \delta \in \{a, b, c\}^*$ such that $w = \gamma ab\delta$.

**Proof of subgoal one:** we first prove the following lemma:

**Lemma 1**

$N^*(S_0, \gamma ab) = S_{02}$

Proof: the left hand side equals $N^*(N^*(S_0, \gamma), ab)$ by the append theorem. But we have no way of knowing what $N^*(S_0, \gamma)$ is, so we consider every possibility:

- $N^*(S_0, ab) = S_{02}$
- $N^*(S_{01}, ab) = S_{02}$
- $N^*(S_{02}, ab) = S_{02}$

We next prove

**Lemma 2**

$N^*(S_{02}, \delta) = S_{02}$

Proof: this follows by an easy induction on the length of $\delta$:

(Base case) $N^*(S_{02}, \epsilon) = S_{02}$ \hspace{1cm} (Def. of $N^*$)

(Inductive case)

\[
N^*(S_{02}, x\delta) = N^*(N(S_{02}, x), \delta) = N^*(S_{02}, \delta) = S_{02}
\]

(Holds for all $x \in \{a, b, c\}$) \hspace{1cm} (IH)

We may now complete our proof of subgoal one:

\[
N^*(S_0, \gamma ab \delta) = N^*(N^*(S_0, \gamma ab), \delta)
\]

(Append Theorem)
\[ N^*(S_0, \delta) = S_{02} \]  \hspace{1cm} \text{(Lemma 1)}

\[ = S_{02} \]  \hspace{1cm} \text{(Lemma 2)}

**Proof of subgoal two:** We first prove that

**Lemma 3**

\[ N^*(S_0, w) = S_{02} \Rightarrow \exists \gamma', \delta. (w = \gamma'b\delta \land N^*(S_0, \gamma') = S_{01}) \]

Proof: by induction on the length of \( w \). Base case, \( w = \epsilon \), which follows because \( N^*(S_0, \epsilon) \neq S_{02} \), so the LHS of the implication is false, and the implication is vacuously true. Inductive case, suppose \( N^*(S_0, wx) = S_{02} \). Now \( N^*(S_0, wx) = N(N^*(S_0, w), x) \) by the corollary to the append theorem. We cannot know what \( N^*(S_0, w) \) is, but can eliminate one possibility: there is no \( x \in \{a, b, c\} \) such that \( N(S_0, x) = S_{02} \), so \( N^*(S_0, w) \neq S_{02} \). If \( N^*(S_0, w) \) were \( S_{01} \) then \( x \) can only be \( b \), and so Lemma 3 holds by setting \( \gamma' = w \) and \( \delta = \epsilon \). If \( N^*(S_0, w) \) were \( S_{02} \), then by the IH \( w = \gamma'b\delta \) and \( N^*(S_0, \gamma') = S_{01} \). Therefore \( wx = \gamma'(b\delta x) \), whatever \( x \) is. So Lemma 3 holds in this case too.

We next observe that

**Lemma 4**

\[ N^*(S_0, w) = S_{01} \Rightarrow \exists \gamma. w = \gamma a \]

Proof: simply because all arcs into \( S_{01} \) are labelled by \( a \).

Hence there exist strings \( \gamma, \gamma', \delta \) such that

\[ N^*(S_0, w) = S_{01} \Rightarrow w = \gamma'b\delta \land N^*(S_0, \gamma') = S_{01} \]  \hspace{1cm} \text{(Lemma 3)}

\[ \Rightarrow w = \gamma'b\delta \land \gamma' = \gamma a \]  \hspace{1cm} \text{(Lemma 4)}

\[ \Rightarrow w = \gamma ab\delta \]

(v)

\[
S_0 \rightarrow aS_0 \mid bS_0 \mid cS_0 \mid aS_1 \\
S_1 \rightarrow bS_2 \\
S_2 \rightarrow aS_2 \mid bS_2 \mid cS_2 \mid \epsilon
\]

where \( S_0 \) is the start symbol

(vi) Suppose for contradiction that there exists a DFA \( D \) that recognises \( M \).

There are infinitely many strings of the form

\[ a^0, a^1, a^2, \ldots \]

but only finitely many states, so by the pigeonhole principle there must be some state \( S \) such that \( S = N^*(S_0, a^i) \) for infinitely many \( i \) (where \( S_0 \) is the start state). In particular, pick some \( j \geq 2 \) and \( k \geq j + 1 \) with \( S = N^*(S_0, a^j) = N^*(S_0, a^k) \).
Now $2j > j + 1 > j$ (because $j \geq 2$), so $a^j b^{j+1} \in M$ and will be accepted by $D$, i.e. $N^*(S_0, a^j b^{j+1}) \in F$, where $F$ is $D$’s set of final states.

By the Append Theorem $N^*(S_0, a^j b^{j+1}) = N^*(N^*(S_0, a^j), b^{j+1})$, so because $N^*(S_0, a^j) = N^*(S_0, a^k)$ we have $N^*(N^*(S_0, a^k), b^{j+1}) \in F$. By the Append Theorem again we have $N^*(S_0, a^k b^{j+1}) \in F$. But $k \geq j + 1$, so it is not the case that $2k > j + 1 > k$ (as $k$ is too big!), so $a^k b^{j+1} \notin M$ and should not have been accepted, contradicting the existence of the DFA $D$ accepting $M$.

2 Pushdown Automata and Context-Free Language

(i) It is impossible for the number of $a$s to be less than two, so the grammar below starts off by adding two $a$s and three $b$s, then proceeds by adding either one or two $b$s for each $a$ added.

$$S \rightarrow aaTbbb$$
$$T \rightarrow \epsilon \mid aTb \mid aTbb$$

where $S$ is the start symbol.

This is not the only correct answer – most context-free languages are definable via many different context-free grammars, and the language $M$ is no exception to this. Of note, the grammar above is ambiguous (see below); designing an unambiguous grammar for $M$ is an worthwhile exercise.

(ii) For the grammar above either of these trees are correct:

(iii) The constructed (non-deterministic PDA) has an initial state $q_0$, with the following transitions:
Initialisation: \( \delta(q_0, \epsilon, Z) \rightarrow q_1/SZ \)
Non-terminals:
\( \delta(q_1, \epsilon, S) \rightarrow q_1/T \)
\( \delta(q_1, \epsilon, S) \rightarrow q_1/W \)
\( \delta(q_1, \epsilon, T) \rightarrow q_1/UV \)
\( \delta(q_1, \epsilon, U) \rightarrow q_1/aUb \)
\( \delta(q_1, \epsilon, U) \rightarrow q_1/\epsilon \)
\( \delta(q_1, \epsilon, V) \rightarrow q_1/eV \)
\( \delta(q_1, \epsilon, V) \rightarrow q_1/\epsilon \)
\( \delta(q_1, \epsilon, W) \rightarrow q_1/XY \)
\( \delta(q_1, \epsilon, X) \rightarrow q_1/aX \)
\( \delta(q_1, \epsilon, X) \rightarrow q_1/\epsilon \)
\( \delta(q_1, \epsilon, Y) \rightarrow q_1/bYc \)
\( \delta(q_1, \epsilon, Y) \rightarrow q_1/\epsilon \)
Terminals:
\( \delta(q_1, x, x) \rightarrow q_1/\epsilon \)
Termination:
\( \delta(q_1, \epsilon, Z) \rightarrow q_2/\epsilon \)

where \( x \in \{a, b, c\} \).

(iv) The PDA trace for processing the string \( abbc \) is as below.

\[
\begin{align*}
(q_0, abbc, Z) & \Rightarrow (q_1, abbc, SZ) \\
& \Rightarrow (q_1, abbc, WZ) \\
& \Rightarrow (q_1, abbc, XYZ) \\
& \Rightarrow (q_1, abbc, aXYZ) \\
& \Rightarrow (q_1, bbc, XYZ) \\
& \Rightarrow (q_1, bbc, bYcZ) \\
& \Rightarrow (q_1, bbc, bYcZ) \\
& \Rightarrow (q_1, cc, YcZ) \\
& \Rightarrow (q_1, cc, ccZ) \\
& \Rightarrow (q_1, c, cZ) \\
& \Rightarrow (q_1, \epsilon, Z) \\
& \Rightarrow (q_2, \epsilon, \epsilon) \\
& \text{Accept.}
\end{align*}
\]
3 Turing Machine and Computability

(i) An example Turing machine looks like follows:

(ii) Call this language $P$ and suppose it is accepted by some $TM M_j$. Since all $TM$ codes end in a 0, $j$ must be even, and hence equal some $2i$. We now ask whether $w_i$ is accepted by $M_{2i}$. If it’s accepted, then $w_i$ is not in $P$, thus not accepted by $M_j$, but $M_j$ is the same as $M_{2i}$. If it’s rejected, then $w_i$ is in $P$, hence accepted by $M_j$, which is the same as $M_{2i}$. In both cases we have a contradiction. Therefore $M_j$ cannot exist.