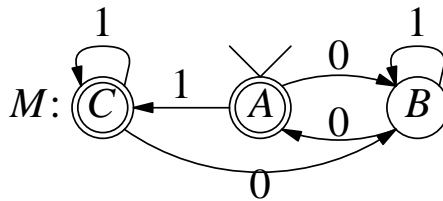


EQUIVALENCE OF STATES AND STATE-MINIMIZATION OF AN FSA

The "future" Language of a State:

- $L_{q_i}(M) = \{x: x \text{ takes } M \text{ from } q_i \text{ to a final-state}\}$; in short, L_{q_i} .
- $L_{q_i}(M)$ captures the notion "future" of state q_i .
- $L_{\text{start-state}}(M) = L(M)$.

Example. $L_B(M) = \{0, 01, 10, 000, 011, 101, 110, \dots\} = L_{0\text{-odd}}$



Equivalence of Two States q_i and q_j of an FSA:

- q_i is equivalent to q_j (in short, $q_i \approx q_j$) if $L_{q_i} = L_{q_j}$.

Question:

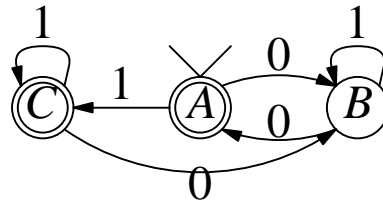
- ? How to determine the equivalence of q_i and q_j in an FSA M ?
(Here, the problem is that $L_{q_i}(M)$ and $L_{q_j}(M)$ can be infinite!)

m -equivalence: A finite form of " \approx "

- $L_{q_i}^{(m)} = \{x: x \in L_{q_i} \text{ and } |x| = m\}$, a finite subset of $\subseteq L_{q_i}$.
 - $L_{q_i}^{(0)} = \{\lambda\}$, if q_i is a final-state; otherwise, $= \emptyset$.
 - $L_{q_i}^{(k)} \cap L_{q_i}^{(m)} = \emptyset$ for $k \neq m$ and $\bigcup_{m \geq 0} L_{q_i}^{(m)} = L_{q_i}$.
- q_i is m -equivalent to q_j (in short, $q_i \approx_m q_j$) if $L_{q_i}^{(m)} = L_{q_j}^{(m)}$.

Question: What is $L_B^{(3)}$ for the FSA above?

m -EQUIVALENCE



$L_{q_i}^{(m)}$	$m = 0$	$m = 1$	$m = 2$	$m = 3$...
$q_i = A$	$\{\lambda\}$	$\{1\}$	$\{00, 11\}$	$001, 010, 100, 111$...
$q_i = B$	\emptyset	$\{0\}$	$\{01, 10\}$	$000, 011, 101, 110$...
$q_i = C$	$\{\lambda\}$	$\{1\}$	$\{00, 11\}$	$001, 010, 100, 111$...

- (1) The states A and C are m -equivalent for each $m \geq 0$.
- (2) The states A and B are not m -equivalent for any $m \geq 0$.

0-equivalence: For any FSA,

- All final states are 0-equivalent to each other, and all non-final states are 0-equivalent to each other.

STATE-MINIMIZATION ALGORITHM

Algorithm-1 (using $L_{q_i}^{(m)}$):

1. Begin with two groups of states: F (the final-states) and $Q - F$ (the non-final states), corresponding to the 0-equivalence classes.
2. For each $m = 1, 2, \dots$ do the following until you reach a stage where no state-group is split up.

Decompose each state-group into two or more groups, if necessary, based on $L_{q_i}^{(m)}$, i.e., if q_i and q_j are presently in the same group and $L_{q_i}^{(m)} \neq L_{q_j}^{(m)}$, then separate them into different groups. (At this point, q_i and q_j are in the same group if and only if $q_i \approx_k$ for all $k \leq m$.)
3. The minimum-state FSA is obtained by merging the states in each group into a single state.

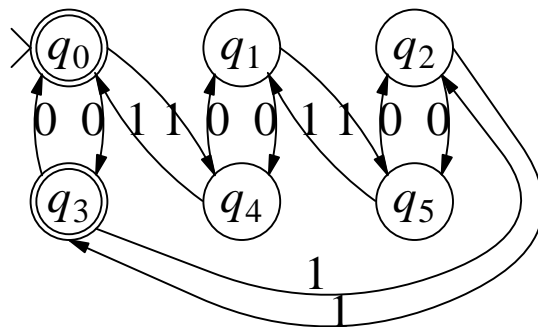
Algorithm-2 (*without* using $L_{q_i}^{(m)}$):

1. Begin with two groups of states: F (the final-states) and $Q - F$ (the non-final states).
2. For each $m = 1, 2, \dots$ do step (3) until you reach a stage where no group is split up.
3. Repeat the following for each input symbol $a_k \in Z$.
 - (i) Compute $\delta(q_i, a_k)$ for all states.
 - (ii) Decompose each group into two or more groups, if necessary, based on $\delta(q_i, a_k)$ as follows:

If q_i and q_j are presently in the same group and $\delta(q_i, a_k)$ and $\delta(q_j, a_k)$ are not in the same group, then separate them into two different groups.
4. The minimum-state FSA is obtained by merging the states in each group into a single state.

ILLUSTRATION OF ALGORITHM-1

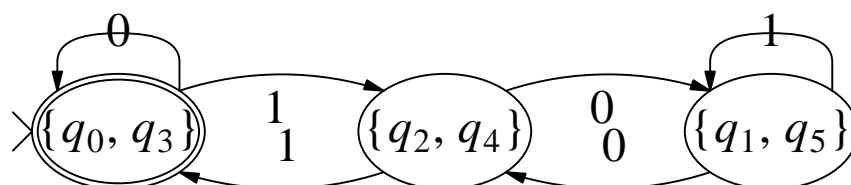
State (p, r) : $p = \text{odd/even parity of number of binary bits read}$ and $r = \text{remainder}(n_j, 3)$, where $n_j = \text{binary number read}$.



$q_0 = (\text{even}, 0)$
 $q_1 = (\text{even}, 1)$
 $q_2 = (\text{even}, 2)$
 and so on $q_5 = (\text{odd}, 2)$

State	$Lq_i^{(m)}$			
	$m = 0$	$m = 1$	$m = 2$	$m = 3$
q_0	$\{\lambda\}$	$\{0\}$	$\{00, 11\}$	$\{000, 011, 110\}$
q_3	$\{\lambda\}$	$\{0\}$	$\{00, 11\}$	$\{000, 011, 110\}$
q_1	\emptyset	\emptyset	$\{01\}$	$\{010, 101\}$
q_2	\emptyset	$\{1\}$	$\{10\}$	$\{001, 100, 111\}$
q_4	\emptyset	$\{1\}$	$\{10\}$	$\{001, 100, 111\}$
q_5	\emptyset	\emptyset	$\{01\}$	$\{010, 101\}$
	$\{q_0, q_3\}$ $\{q_1, q_2, q_3, q_4\}$	$\{q_0, q_3\}$ $\{q_1, q_5\}$ $\{q_2, q_4\}$	No change in any group.	No need to consider this.

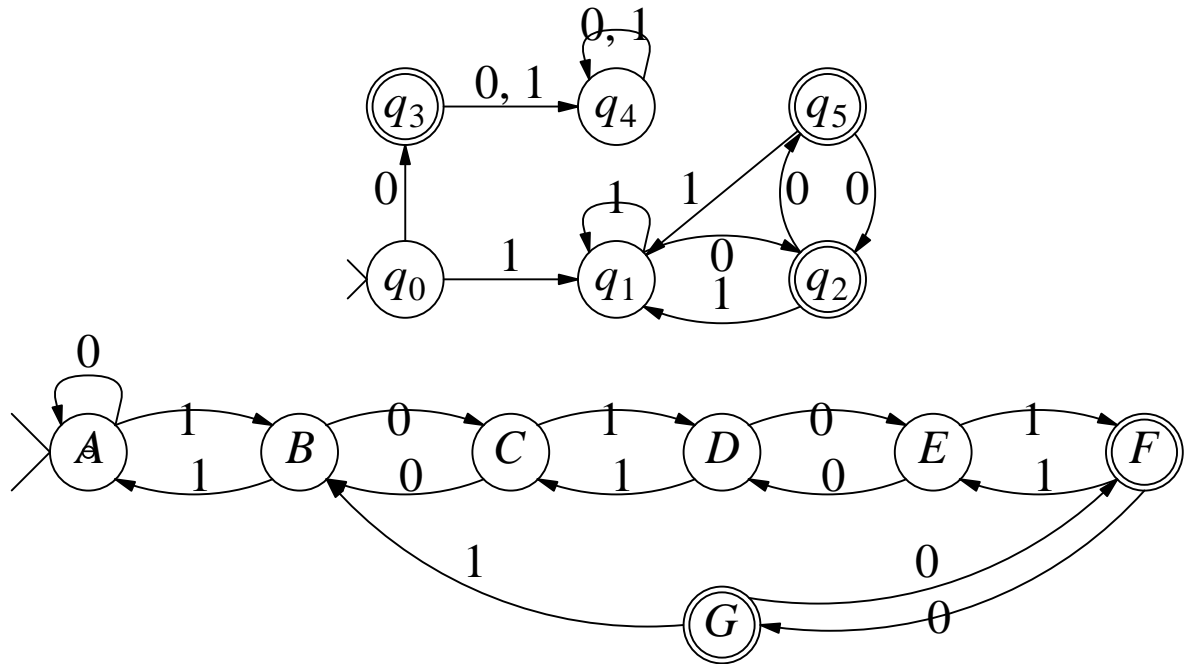
After Merging Equivalent Classes of States:



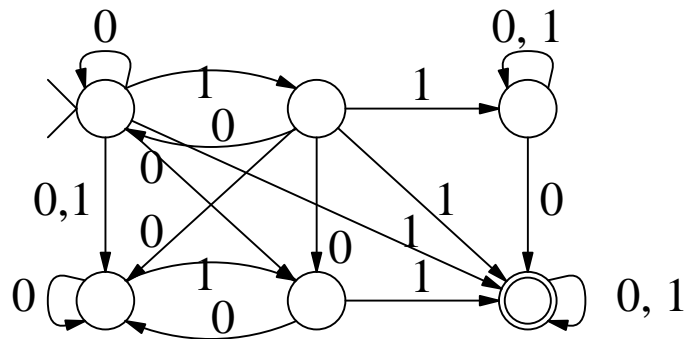
The minimum-state form of the above FSA.

EXERCISE

1. Apply Algorithm-1 to each FSA below and show all details.



2. Apply the minimization algorithm to the NFSA below (which has some important connection with M_{has-11}).



3. Is it true that all error states in an arbitrary FSA will be merged into a single state in the minimization process (why)?
4. Give an example FSA to show that all unreachable states in an arbitrary FSA may not get merged into a single state.
5. Argue that $q \approx q'$ implies $\delta(q, a) \approx \delta(q', a)$ for all $a \in \Sigma$.

WHAT IS A STATE OF AN FSA

Importance of Answering This Question:

- It explains why certain languages like L_{sym} and $L_{a^n b^n}$ are not acceptable by any FSA,
 - There is an FSA for a language L , i.e., L is regular if and only if it has only a finitely many quotient languages.

States of the optimal FSA for L are the quotient languages of L .
--

- The optimal FSA (which has the smallest number of states) for a language is unique; its states correspond to the quotient languages of L .

QUOTIENT LANGUAGES

Quotient of L :

- For any string x , $L/x = \{y: xy \in L\}$. Here, y are the "futures" (to do) after x .
- $L/\lambda = L$, for any L and $(L/x)/y = L/(xy)$.

Example.

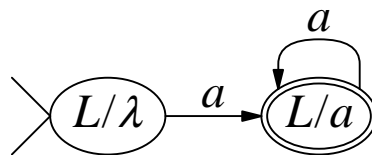
- For $L = \{a, ab, aab, b\}$,

$$\begin{array}{ll} \overline{L/a = \{\lambda, b, ab\}} & \overline{L/ab = \{\lambda\}} \\ \overline{L/b = \{\lambda\}} & \overline{L/ba = \emptyset} \end{array}$$

- For $L = \{a, aa, aaa, \dots\} = a^+$,

$$\begin{array}{ll} \overline{L/a = \{\lambda, a, aa, \dots\} = a^* \supset L} & \overline{L/aa = L/a} \\ \overline{L/b = \emptyset} & \overline{L/ab = \emptyset} \end{array}$$

States of $M(a^+)$ vs. Quotients of a^+ :



Question:

- ? For $L = \{a, ab, aab, b\}$, show the states $M(L)$ in terms of the quotients of L .

REGULAR LANGUAGES AND QUOTIENT LANGUAGES

L is regular $\Leftrightarrow L$ has finitely many quotient languages.

Example. Let $L = \{a^n b^n : n \geq 1\} = \{ab, aabb, aaabbb, \dots\}$.

$$\begin{aligned} L/\lambda &= L \\ L/a &= \{b, abb, aabbb, \dots\} = \{a^n b^{n+1} : n \geq 0\} \\ L/aa &= (L/a)/a = \{bb, abbb, \dots\} = \{a^n b^{n+2} : n \geq 0\} \\ L/aaa &= (L/aa)/a = \{bbb, abbbb, \dots\} = \{a^n b^{n+3} : n \geq 0\} \\ &\dots \end{aligned}$$

- These quotient-languages are distinct (they differ in the smallest string in them).
- The quotients $L/b = L/bb = \dots = \emptyset$, and they are different from the ones above. (Are there still other quotients of L ?)

EXERCISE

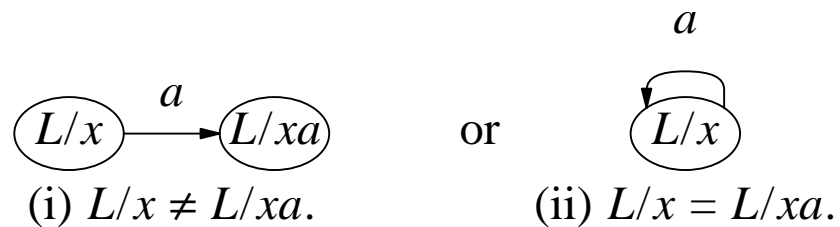
1. Find the quotient languages of L_{has-11} .
2. Let $L = \{a^m b^n : m, n \geq 1\} \subseteq (a+b)^*$. Give a regular expression for each of the distinct quotient languages of L .
3. Give a clear argument to show that $L_{sym} = \{x \in (a+b)^+ : x \text{ is symmetric, i.e., } x = x^r\} = \{\lambda, a, b, aa, bb, aaa, aba, bab, bbb, \dots\}$ has infinitely many quotient languages.
4. Let $L_{left-shift} = \{x \in \{b_0, b_1, b_2, b_3\}^* : \text{the lower part of } x \text{ is the left-shift of the top part}\}$. Explain whether $L_{left-shift}$ is regular or not.

$$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = b_2 b_1 b_3 b_2 \in L_{left-shift}.$$

5. If L_1 and L_2 are two languages over the alphabet Σ and $L_1/x = L_2/x$ for all $x \in \Sigma^+$, then prove that $L_1 = L_2$.

UNIQUENESS OF MINIMUM FSA FOR A REGULAR LANGUAGE L

- States have the form L/x .
- Start-state = $L/\lambda = L$.
- The state L/x is final if $x \in L$, i.e., L/x contains λ .
- The transitions are of the form shown below for each $a \in \Sigma$:



- The state L/x is an error-state (dead-state) if $L/x = \emptyset$.