BUILDING FINITE-STATE MODELS/AUTOMATA

Issue I.

- There is *no algorithm* if the problem is stated in English or some other informal language because FSA is a formal structure.
  - Human understanding/interpretation is involved.
  - The FSA can also depend on the approach taken or make a different interpretation of the problem.
- There is *an algorithm* if you give a formal description of the problem, say, as a *regular expression*.
  - The FSA may depend on the regular expression (different regular expressions may describe the same language).

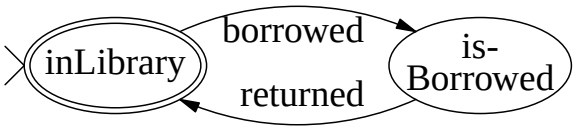
Issue II.

- Combine FSAs for simpler problems to build FSA for more complex problems.

FINITE-STATE MODEL OF A LIBRARY BOOK

A Simple Case: Two operations "borrowed" and "returned".

- The FSM shows that the two operations must alternate.
- The start-operation is borrow (why?) and we assume that a book once borrowed will be returned at some point.



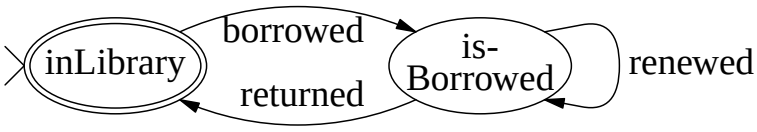
- Shows the applicable operations in each state.
- We need two states (why?).

Question:

•? Can you think of two operations *b* and *r* in some situation where they also alternate but the starting operation is *r*?

More Complex FSM: Adding renew-operation.

- A book can be renewed only if it is borrowed but not yet returned.
- There is no limit on the number of renewals.



Questions

•? What is the similarity between returned and renewed operations? What distinguishes them and how is it reflected in the FSM? Do you see any shortcoming in this model?

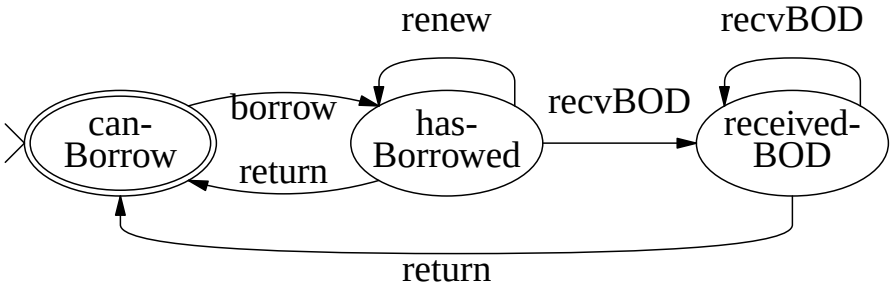
•? Show the new FSM if we assume that a book can be renewed at most 2 times. (Is there a need for such a restriction?)

•? How to model the fact that the person borrowing the book is the person renewing it? (Is this restriction necessary?)

FINITE-STATE MODEL OF A PERSON FOR LIBRARY EXAMPLE

Assumptions:

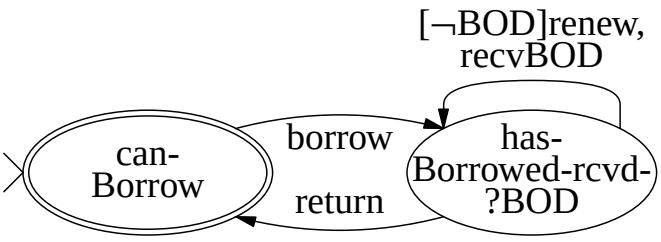
- At most one book may be borrowed at any time.
- If the book is not returned by the due date, then a book-overdue notice (BOD) is sent to the person periodically till it is returned.
- If the book is borrowed, it can be returned; it can also be renewed provided no BOD-notice has been sent/received.



This is also a model for a book (why?) with 5 operations borrow(ed), return(ed), etc

A Simplified Form Using Guards:

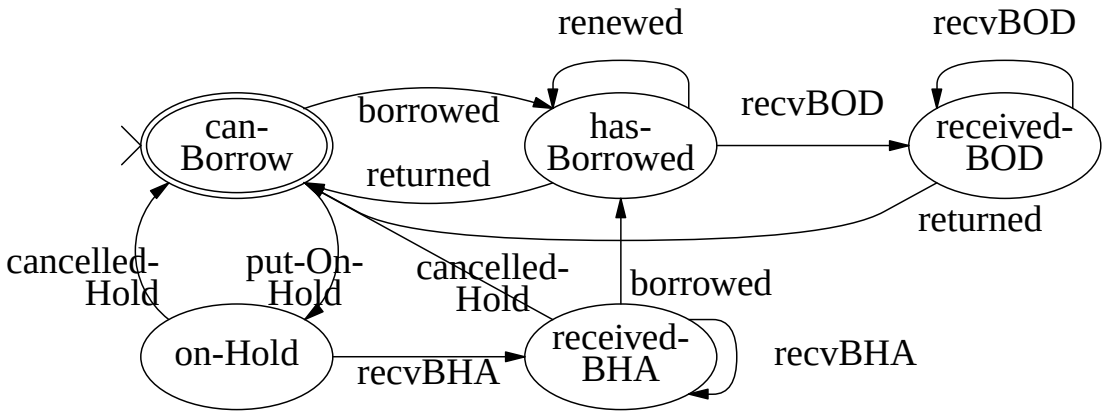
- Initially, variable BOD = "no".
- The borrow-operation does not change BOD.
- The recvBOD-operation sets it to ’yes’ and the return-operation resets it ’no’.



EXERCISE

1. Modify the model of a person for the following additional assumptions:
   (a) The person can put a hold on a book (borrowed by someone else) if he does not have another book borrowed.
   (b) He can put at most one book on hold at any time (just as he can borrow at one book at a time).
   (c) He may borrow the book on which he has put a hold after he gets a Book-on-Hold-Available (BHA) notice. He may cancel the hold after or before getting BHA-notice.
   (d) The BHA-notice is sent periodically until the person borrows the book or cancels the "hold". (Assume no limit on the number of BHA-notice.)

2. Shown below is a finite-state model of a book which allows hold-operation and BHA-notice-operation. State all restrictions implied by this model. Should we merge the states "on-Hold' and "received-BHA"; what is the new model after merging?

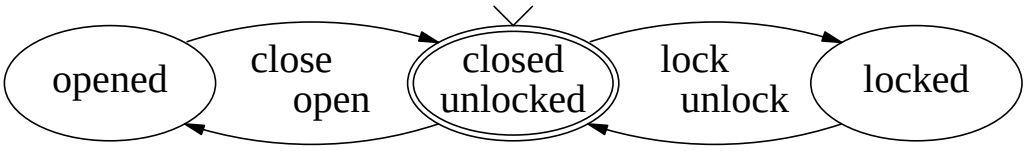


Generalize this model when a book may also be put on hold if it is borrowed and may or maynot have received BOD. (The person putting the hold must be different from the one having the book.)

FINITE-STATE MODEL FOR A DOOR WITH LOCK

Assumptions: Four operations: open, close, lock, unlock.

- Door can be opened if it is closed and unlocked, and it can be closed if it is opened.
- Door can be locked only if it is closed and unlocked, and it can be unlocked only if it is locked.
- Initially, the door is closed and unlocked.



The meaning of states determines the transitions among them.

EXERCISE

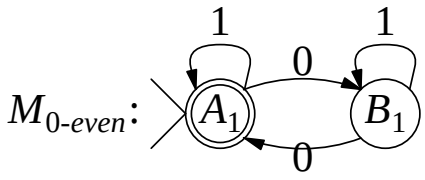
1. Show the transitions among the states A = "the last symbol seen is 0" and B = "the last symbol symbol seen is 1" for $\Sigma = \{0, 1\}$. Add a suitable start-state and the transitions from it.
2. Repeat Problem 1 with the state B replaced by B_0 = "the last two symbols seen are 01" and B_1 = "the last two symbols seen are 11". If you need to add more states to complete all transitions, then do so and describe them.
3. Break up the state A in Problem 2 into two or more states (and describe them) if we are looking for strings which contain "11", and then show the transitions among the states. What problems do you see, what's causing them, and how do you propose to resolve them?

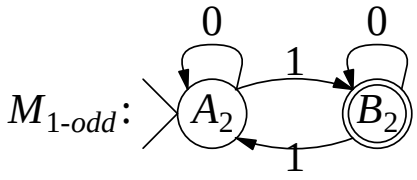
AND-COMBINATION OF TWO FSAs

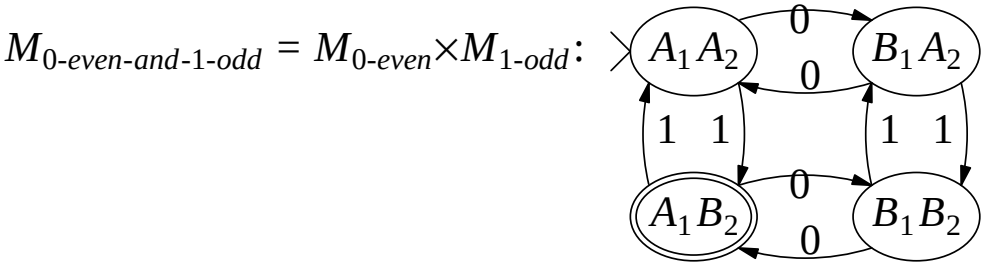
$L = \{x \in (0 + 1)^*: \#(0, x) = \text{even and } \#(1, x) = \text{odd}\}.$

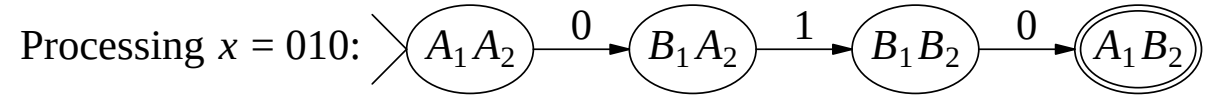
- $L = L_{0\text{-}even} \cap L_{1\text{-}odd}$
  $= \{1, 001, 010, 100, 00001, 00100, 10000, 00111, \cdots\}.$

AND-Combination $M_1 \times M_2$**:** $L(M_1 \times M_2) = L(M_1) \cap L(M_2)$

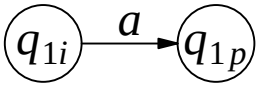




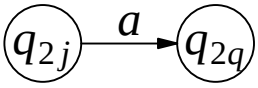




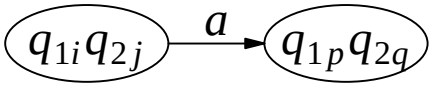
General-case: $Q = Q_1 \times Q_2$, $q_0 = (q_{10}, q_{20})$, $F = F_1 \times F_2$, $\Sigma = \Sigma_1 \cap \Sigma_2$, and the transitions as shown, where we write (q_{1i}, q_{2j}) in short as $q_{i1}q_{2j}$:



in M_1



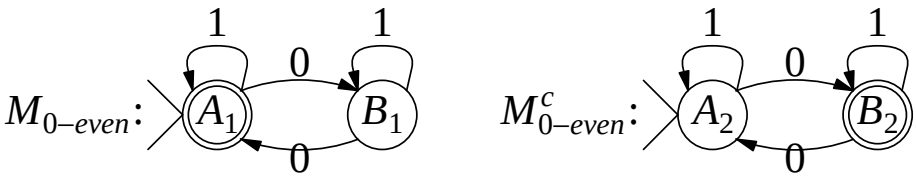
in M_2

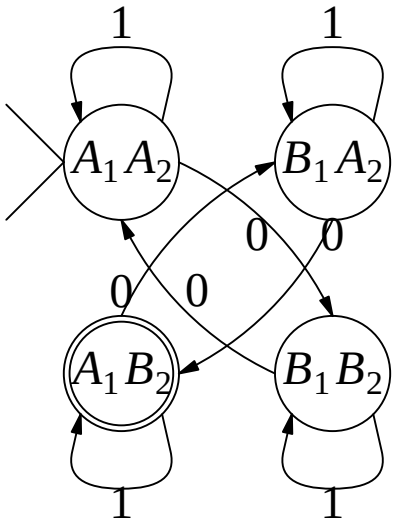


in $M_1 \times M_2$

AN EXTREME CASE: $M \times M^c$

- $L(M \times M^c) = L(M) \cap L(M^c) = L(M) \cap L^c(M) = \varnothing$





$M_{0\text{-}even} \times M^c_{0\text{-}even}$: The final-state is unreachable!

Avoid Constructing Unreachable States in $M_1 \times M_2$:

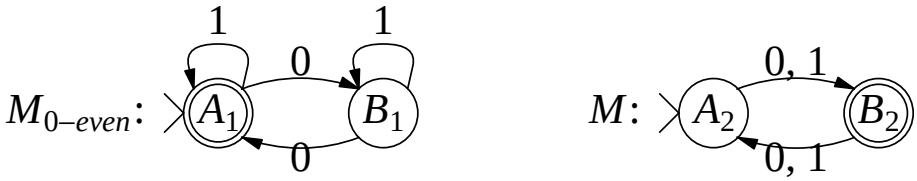
(1) First create the start-state.

(2) Construct transitions only from the states that have been created and adding new states when necessary. (States A_1B_2 and A_2B_1

$$L(M_1) \cup L(M_2) = [L^c(M_1) \cap L^c(M_2)]^c$$
$$= [L(M_1^c) \cap L(M_2^c)]^c = L^c(M_1^c \times M_2^c).$$

EXERCISE

1. Give an English description of $L(M)$ for M shown below and also of $L(M_{0\text{–}even}) \cap L(M)$. Explain why the FSA $M_{0\text{–}even} \times M$ is identical to $M_{0\text{-}even\text{-}and\text{-}1\text{-}odd}$ except for the state-names.



2. Construct an FSA for $L = \{x: x$ is a binary string which contains "11", but not "01"$\}$ using the AND-construction. What is the relationship between the error states of the component FSAs and those of the product?
3. Show the first few strings in L given in Problem 2, and construct an FSA for L directly (without using the AND-construction). Compare this FSA with the one obtained in your solution to Problem 2.
4. Show the state-diagram for the FSA for $L(M_1) \cup L(M_2)$ for the case $M_1 = M_{0\text{–}even}$ and $M_2 = M_{1\text{–}odd}$ using the concept of product.
5. Describe the states, start-state, final-states of the FSA for $L(M_1) \cup L(M_2)$ obtained from M_1 and M_2 using the product $M_1^c \times M_2^c$ for the general case.
6. Describe the steps to construct an FSA for $L = \{x \in (0+1)^*: \text{if } x$ contains "11", then x contains "01"$\}$, by starting from $M_{has\text{-}11}$ and $M_{has\text{-}01}$. No need to construct the FSA itself.
7. Find a completely defined FSA with at most 4 states which accepts as much of the language $L_{sym} = \{x \in (a+b)^*: x$ is a palindrome$\}$ as possible, but does not accept any string not in L.
8. Suppose $L_1 \subset L_2$ are two regular languages. How do you prove that there is an FSA for L_1 from which we can obtain an FSA for L_2 by merging some of its states?

HOW TO TEST $L(M) = \emptyset$

Testing $L(M) = \emptyset$:

- There is no path from start-state to any of the final-states, i.e.,
  - either $F = \emptyset$,
  - or none of the final-states are reachable from the start-state.

Question: Suppose M_1 and M_2 are two arbitrary FSAs.

(T1) How to test "$L(M_1) \subseteq L(M_2)$"?

$L(M_1) \subseteq L(M_2)$ if and only if $L(M_1) \cap L^c(M_2) = \emptyset$, i.e., $L(M_1 \times M_2^c) = L(M_1) \cap L(M_2^c) = L(M_1) \cap L^c(M_2) = \emptyset$.

(T2) How to test "$L(M_1) = L(M_2)$"?

Apply (T1) twice: $L(M_1) \subseteq L(M_2)$ and $L(M_2) \subseteq L(M_1)$.

EXERCISE

- Find an FSA for binary strings which are of length $k \geq 3$ and the last $k = 3$ symbols are the same. How many states will your FSA have for a general $k \geq 1$.

FSA FOR BINARY NUMBERS DIVISIBLE BY 3

Ideas:

- If n_j = integer($a_1 a_2 \cdots a_j$), then $n_j = 2n_{j-1} + a_j$ (where $n_0 = 0$).
- If r_j = rem(n_j, 3), then we can compute r_j from r_{j-1} and a_j; $r_0 = 0$.
- Since r_j has a finite number of values {0, 1, 2}, we can take r_j as the state; $r_j = 0$ the final-state (and also the start-state).

$$n_j = 2n_{j-1} + a_j \quad (\text{for } j = 1, \text{ take } n_0 = 0)$$

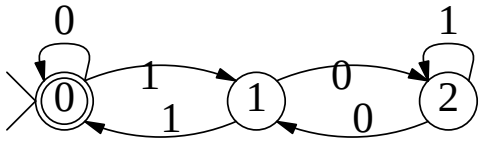
If $n_{j-1} = 3m + r_{j-1}$ for some $m \geq 0$ $(r_0 = 0)$

then $n_j = 6m + 2r_{j-1} + a_j$ and $r_j = \text{rem}(2r_{j-1} + a_j, 3)$.

x=110101	a_1=1	a_2=1	a_3=0	a_4=1	a_5=0	a_6=1
$a_1 a_2 \cdots a_j$	1	11	110	1101	11010	110101
n_j	1	3	6	13	26	53
r_j	1	0	0	1	2	2
a_j	1	1	0	1	0	1
$k_j = 2r_{j-1}+a_j$	1	3	0	1	2	5
rem(k_j, 3)	1	0	0	1	2	2

	r_j	$a_j = 0$	$a_j = 1$
r_{j-1}:	0	0	1
	1	2	0
	2	1	2

M_{3n}:



State = "current remainder"; it is based on the "past", i.e., the processed part of input.

Question: Give a future-based description of the above states.

REVERSE BINARY NUMBERS OF THE FORM 3n

Example.

- x = 11001 gives the reverse string 10011, which corresponds to $n(x) = 19$, and thus x is not to be accepted.
- But x = 1100 gives the reverse string 0011, which corresponds to $n(x) = 3$, and is to be accepted.

Question: Given $x = a_1 a_2 \cdots a_k$,

•? How to compute r_j = remainder(n_j, 3), where $n_j = n(a_1 a_2 \cdots a_j)$ = the binary number corresponding to $a_j a_{j-1} \cdots a_1$, without computing n_j directly?

Hint: $n_j = a_j \cdot 2^{j-1} + n_{j-1}$. What does say about the relationship among r_j, r_{j-1}, and a_j? (It depends on whether j is even or odd for $a_j = 1$.)

•? Show the table of the relationship among r_j, r_{j-1}, and a_j for each of j = even and j = odd. Then, give the state diagram of your FSA.

EXERCISE

1. Consider the following six "future"-based states

   (even, 0), (even, 1), (even, 2), (odd, 0), (odd, 1), and (odd, 2).

   for the problem of "divisibility of binary strings by 3". Each state has two parts: (Remaining number of bits, Remainder from that part). Show the transitions, the start-states, and the final-states.

2. Since 101 = 5 and 010 = 2 both have the same remainder when divided by 3, it follows that $x101y$ and $x010y$ are both divisible by 3 or none is divisible by 3. Thus, at any state of a minimum FSA for binary strings divisible by 3, the input strings 101 and 010 will go to the same state. Verify this property for the FSA on the previous page. Why do we need the FSA to be minimum?

FSA FOR VERIFYING BINARY SUM RIGHT TO LEFT

Example.

35	100011
17	010001
(decimal) 52	(binary) 110100

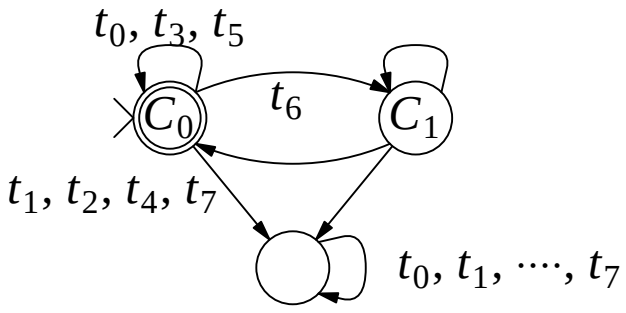
Corresponding Input Encoding:

$$\begin{matrix} t_6 & t_4 & t_1 & t_0 & t_3 & t_5 \end{matrix}$$
$$\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}$$

Input Symbols: 3rd item of each triplet gives a possible sum-bit.

$$\begin{matrix} t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \end{matrix}$$
$$\begin{pmatrix}0\\0\\0\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\1\\0\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix}$$

Transitions at C_0: C_0 (carry = 0) and C_1 (carry = 1).



EXERCISE

1. Complete the transitions from C_1.
2. Obtain the FSA for verifying binary sum from left to right, and explain the meaning of the states to justify the transitions, transitions to the dead-state (if any).

EXERCISE

1. Consider all properly formed propositional formulas $\phi(p, q)$ involving the propositions $\{p, q\}$ and the operators $\{\neg, \wedge, \vee\}$. Thus, the alphabet is $\Sigma = \{p, q, \neg, \wedge, \vee\}$; in particular, we do not allow the use of parentheses '(' and ')' in the input formula. Example of some improperly formed formula are $p \wedge \vee p$ and $\wedge p \vee q$; the formula $p \wedge \neg\neg\neg p \vee q \vee q$ is properly formed. A precise definition of properly formed formulas is:

   1. p and q are properly formed.
   2. If ϕ is properly formed, then $\neg\phi$ is properly formed.
   3. If ϕ_1 and ϕ_2 are properly formed, then $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$ are also properly formed.

   Give the (min-state) FSA for all properly formed propositional formula $\phi(p, q)$.

2. Now consider properly formed formulas $\phi(p)$ involving just one proposition p. Give the min-state FSA for all unsatisfiable $\phi(p)$ and an FSA for all satisfiable $\phi(p)$. A formula $\phi(p)$ is *satisfiable* if $\phi(p)$ has the value true (T) for at least one of the cases p = T and p = F. For example, $p \wedge \neg p$ and $p \wedge p \wedge \neg p$ are unsatisfiable and $p \vee \neg p$ and $p \wedge p \vee \neg p$ are satisfiable. Assume that $\neg$ and $\vee$ have respectively the highest and lowest priority in evaluating $\phi(p)$. (This result generalizes to formulas with ≥ 2 propositions.)

4. Is it true that the min-state FSA for properly formed formulas on the propositions $\{p_1, p_2, \cdots, p_k\}$ which are tautologies has exponential number (in k) of states? How about the FSA for strings in $\{a_1, a_2, \cdots, a_k\}$ which contain each symbol a_i at least once? How about the FSA for binary strings of length $\geq 2n$ ($n \geq 1$ is a given constant) which are of the form xxy, where $|x| = n$?