CSCI 340: Computational Models

Finite Automata with Output

Goals and Limitations

Goal: designing a mathematical model for a computer where the *input string* represents the program and data

What if we want to introduce output? The *state* of the machine could be its output – a yes-no oracle of sorts

All we've done so far is recognize and accept languages! Machines should be capable of doing more.

G. H. Mealy (1955) and E. F. Moore (1956) independently invented two separate models for Finite Automata that have output possibilities

The Moore machine

Definition

A **Moore machine** is a collection of five things:

1. A finite set of states $q_0, q_1, q_2, \ldots$ where q_0 is the start state
2. An alphabet of letters for forming the input string

$$\Sigma = \{a \ b \ c \ \ldots\}$$

3. An alphabet of possible output characters

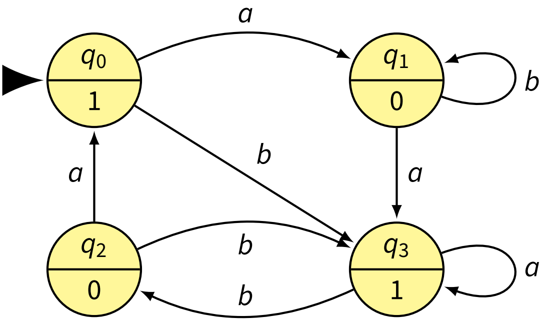
$$\Gamma = \{x \ y \ z \ \ldots\}$$

4. A transition table that shows for *each* state and *each* input letter which state is reached
5. An output table that shows which character from Γ is printed by each state as it is *entered* (this means when we “start” execution we print the character from the start state)

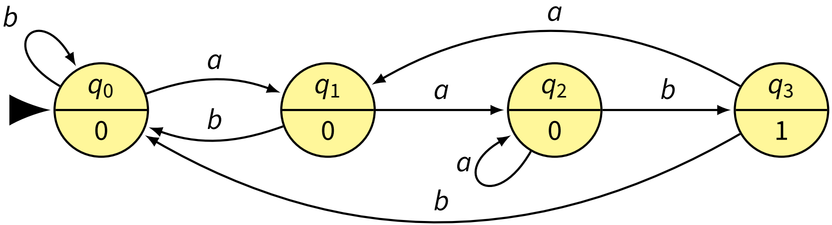
Example of a Moore machine

Input: $\Sigma = \{a\ \ b\}$ Output: $\Gamma = \{0\ \ 1\}$ States: q_0, q_1, q_2, q_3

Old	**Output**	**After** a	**After** b
q_0	1	q_1	q_3
q_1	0	q_3	q_1
q_2	0	q_0	q_3
q_3	1	q_3	q_2



Example: count how many times *aab* occurs



Tracing *aababbaabb*

Input		a	a	a	b	a	b	b	a	a	b	b
State	q_0	q_1	q_2	q_2	q_3	q_1	q_0	q_0	q_1	q_2	q_3	q_0
Output	0	0	0	0	1	0	0	0	0	0	1	0

Example: count how many times *aab* occurs

Interesting properties from the previous slide:

- The number of substrings *aab* in the input string will be exactly the number of 1's in the output string.
- Given a language L and an FA that accepts it, we can "print" 0 in any non-final state and 1 in any final state.
- The 1's in any output sequence mark the end positions of all substrings *aab*
- If we remove the output ability and mark output states with 1 as final and output states with 0 as non-final, we convert the Moore machine to a standard FA

The Mealy machine

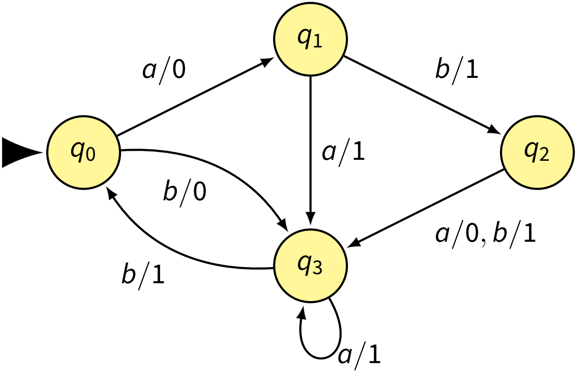
Definition

A **Mealy machine** is a collection of four things:

1. A finite set of states $q_0, q_1, q_2, \ldots$ where q_0 is the start state
2. An alphabet of letters $\Sigma = \{a \;\; b \;\; \ldots\}$ for forming input strings
3. An alphabet of output characters $\Gamma = \{x \;\; y \;\; z \;\; \ldots\}$
4. A *pictorial* representation with states represented by circles and directed edges representing transitions.
    - each edge is labeled with a compound symbol of the form i/o where i is an input letter and o is an output letter
    - *every* state must have only one outgoing edge for *each* input letter
    - the edge we travel is determined by the input letter i
    - while traveling on the edge, we must print out the output character o

Example of a Mealy machine

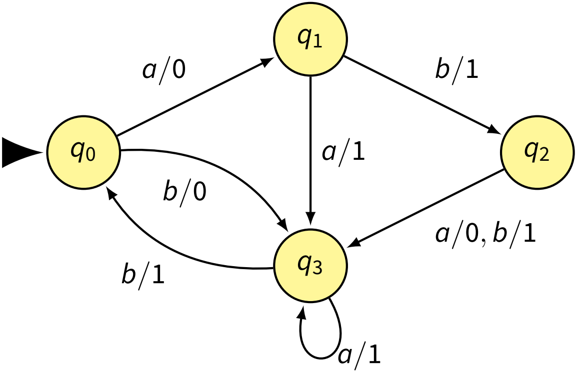
Input: $\Sigma = \{a \;\; b\}$ Output: $\Gamma = \{0 \;\; 1\}$ States: q_0, q_1, q_2, q_3



Tracing *aaabb*

Example of a Mealy machine

Input: $\Sigma = \{a \ \ b\}$ Output: $\Gamma = \{0 \ \ 1\}$ States: q_0, q_1, q_2, q_3



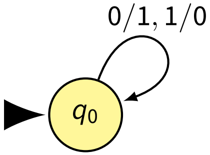
Tracing *aaabb* yields 01110

Example: One's complement of an input bit string

$$\Sigma = \Gamma = \{\mathtt{0}\ \ \mathtt{1}\}$$

Example: One's complement of an input bit string

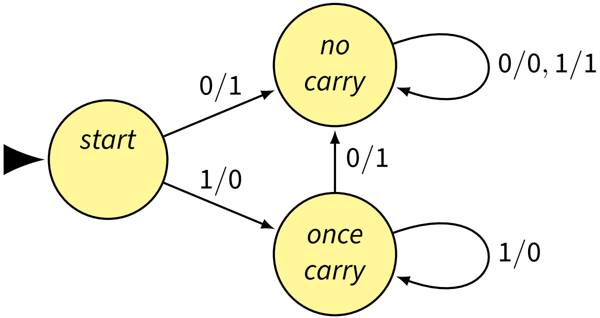
$$\Sigma = \Gamma = \{0\ 1\}$$



Yes, really – that's it

Example: The increment machine

- assume input is a binary number which is reversed
- output should be “one larger” than the input
- Approach: Three states representing
  1. *start* – have yet to read anything
  2. *oncecarry* – “right-most” bits were set – carry until 0
  3. *nocarry* – we have already added the one



Connection between Mealy machines and circuits?

- Once we have an incrementer, we can build another machine to perform binary addition
- Once we have binary addition and 1's complement, we can implement subtraction!

Theorem

If a and b are strings of bits, then $a - b$ can be performed by

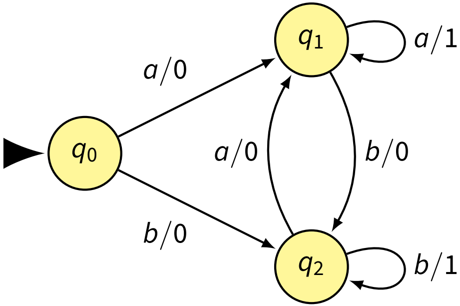
1. *adding the 1's complement of b to a ignoring overflow*
2. *incrementing the results by 1*

$$\begin{aligned}
14 - 5 &= 1110 - 0101 \\
&= 1110 + \textit{ones-complement}(0101) + 1 \\
&= 1110 + 1010 + 1 \\
&= [1]1001 \\
&= 9
\end{aligned}$$

Mealy machines recognizing languages

- Mealy machines cannot “accept” or “reject” an input string
- We can recognize a language by having the output string answer some questions about the input

Recognizing words that contain a double letter



Moore = Mealy

Equivalence has been measured by *accepting* the same language. Moore and Mealy machines cannot be compared in this way.

Problem – why not just compare output?

Moore machines output one extra character!

Solution

Given the Mealy machine *Me* and the Moore machine *Mo* which prints the automatic start character x, we will say these two machines are **equivalent** if for every input string, the output string from *Mo* is exactly x concatenated with the output from *Me*.

Given *Mo*, there is an equivalent *Me*

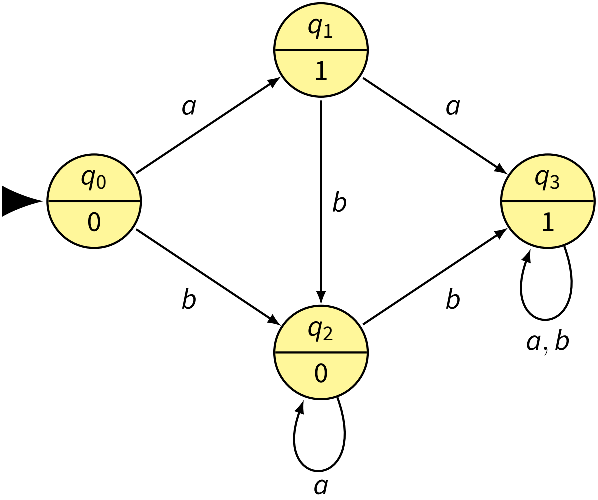
Theorem

If Mo is a Moore machine, then there is a Mealy machine, Me, that is equivalent to it

Proof (by construction).

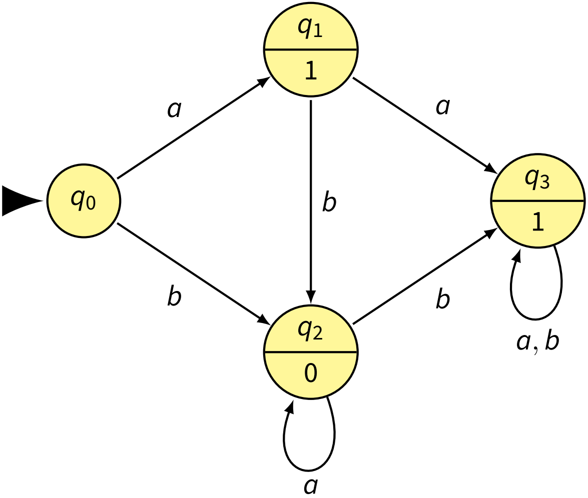
- Consider any state in *Mo*, *q*.
- The output character for *q* shall be *t*
- All incoming edges for *q* are labeled with an input letter.
- Relabel the edges coming into *q* (e.g. $a, b, c, \ldots$) with the output from the state *q* (e.g. $a/t, b/t, c/t, \ldots$)
- remove the output of character *t* from *q*
- repeat for each state – this results in a *Me* where the characters that get printed are exactly what *Mo* would have printed $\square$

Example of *Mo* to *Me* (Moore to Mealy)



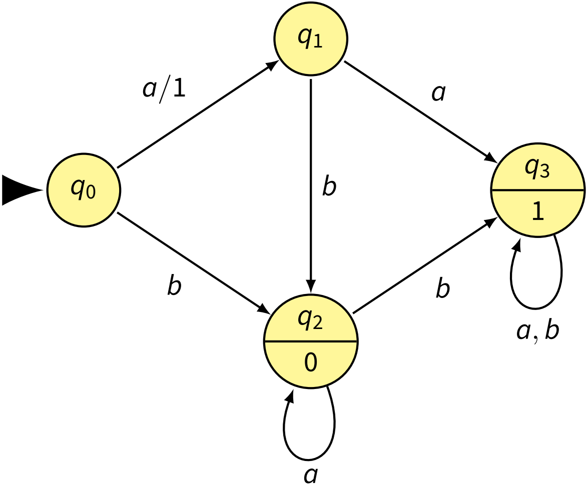
Example of *Mo* to *Me* (Moore to Mealy)

Handling q_0



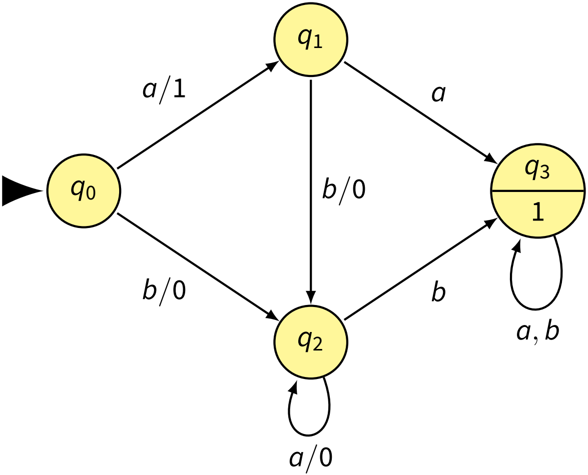
Example of *Mo* to *Me* (Moore to Mealy)

Handling q_1



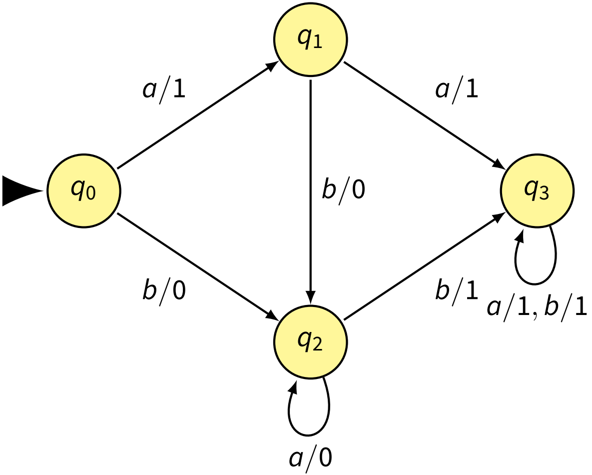
Example of *Mo* to *Me* (Moore to Mealy)

Handling q_2



Example of *Mo* to *Me* (Moore to Mealy)

Handling q_3



Given *Me*, there is an equivalent *Mo*

Theorem

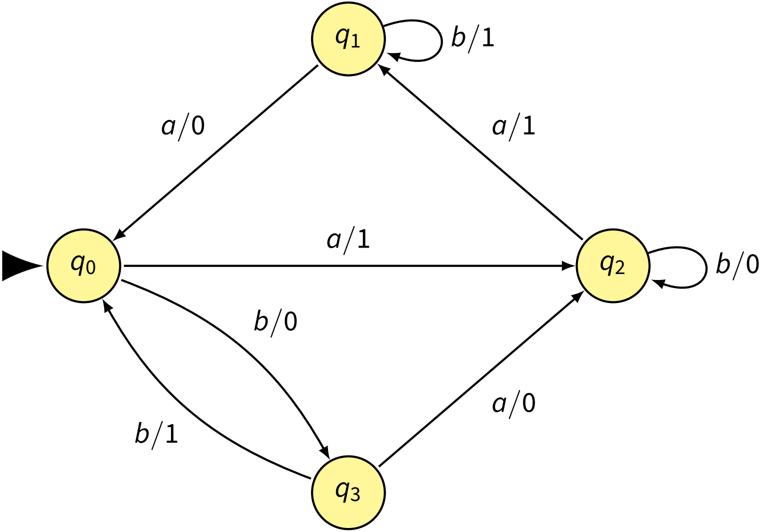
If Me is a Mealy machine, then there is a Moore machine, Mo, that is equivalent to it

Proof (by construction).

- Consider any state in *Me*, *q*.
- All incoming edges for *q* are labeled with an input letter and character *t*.
- If all incoming edges have the same *t*, set the output character of *q* to be *t* and remove *t* from all incoming edges
- Else, replicate *q* (including outgoing edges) for each unique *t* and assign incoming edges to the right copy of *q*. Remove *t* from all incoming edges.
- repeat for each state
- if there is a state with no incoming edges, assign from Γ

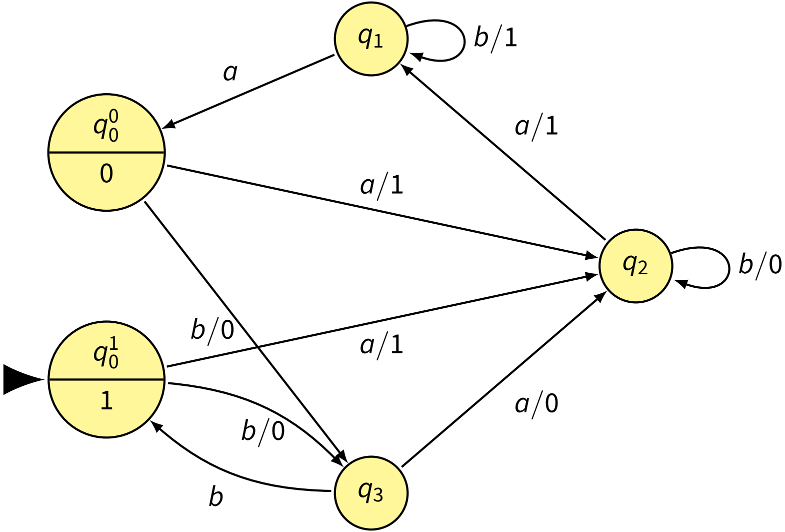
□

Example of *Me* to *Mo* (Mealy to Moore)



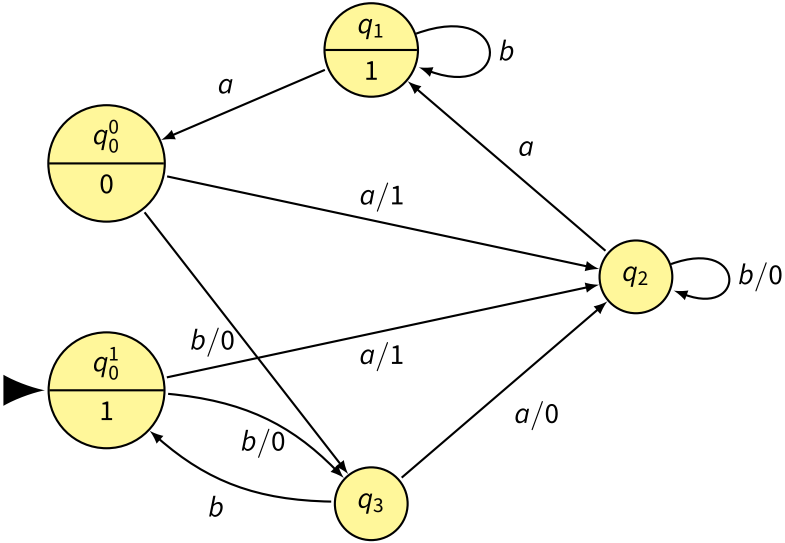
Example of *Me* to *Mo* (Mealy to Moore)

Handling q_0 (incoming edges of 0 and 1)



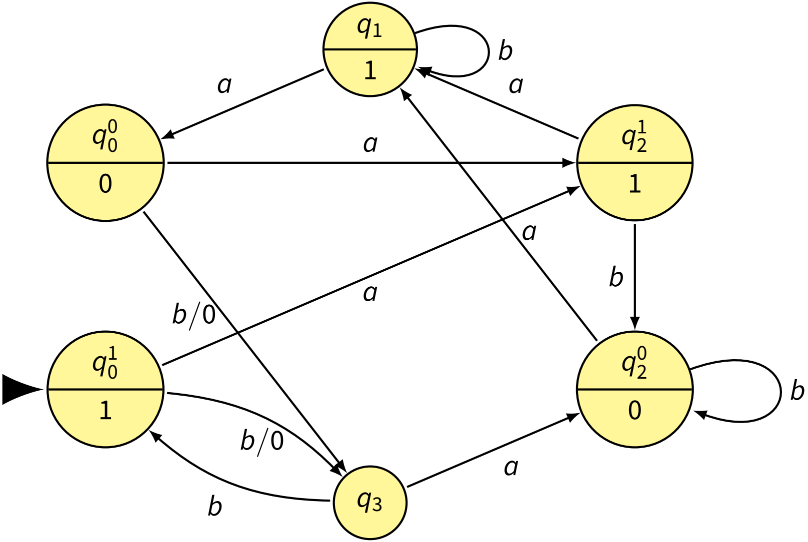
Example of *Me* to *Mo* (Mealy to Moore)

Handling q_1 (incoming edges of only 1)



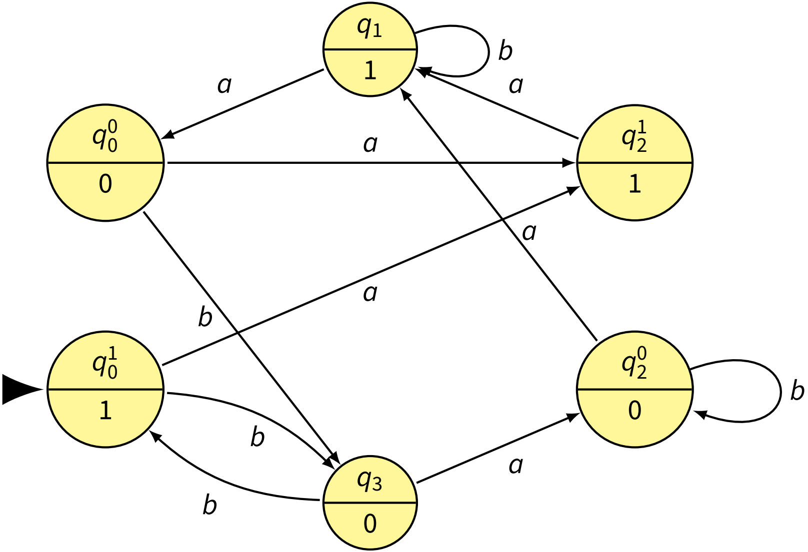
Example of *Me* to *Mo* (Mealy to Moore)

Handling q_2 (incoming edges of 0 and 1)



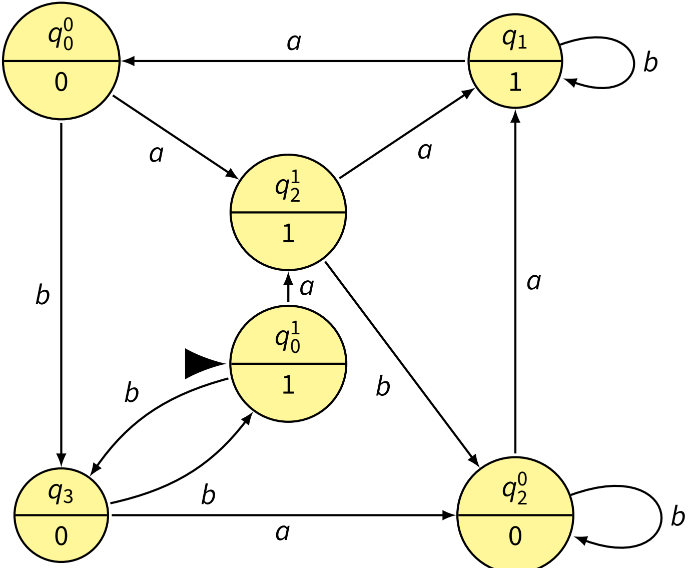
Example of *Me* to *Mo* (Mealy to Moore)

Handling q_3 (incoming edges of only 0)



Example of *Me* to *Mo* (Mealy to Moore)

Redrawn to eliminate overlapping edges



Comparison Table for Automata

	FA	**TG**	**NFA**	**NFA-λ**	**Moore**	**Mealy**
Start States	1	1+	1	1	1	1
Final States	0+	0+	0+	0+	0	0
Edge Labels	$e \in \Sigma$	$w \in \Sigma^*$	$e \in \Sigma$	$e \in \Sigma$ and λ	$e \in \Sigma$	i/o $i \in \Sigma$ $o \in \Gamma$
Number of edges from each state	1 per $e \in \Sigma$	Arbitrary	Arbitrary	Arbitrary	1 per $e \in \Sigma$	1 per $e \in \Sigma$
Deterministic	Yes	No	No	No	Yes	Yes
Output	No	No	No	No	Yes	Yes
Page in Book	53	79	135	146	150	152
Slide	Ch 5 #2	Ch 6 #5	Ch 7 #17	N/A	Ch 8 #2	Ch 8 #6