Biçimsel Diller ve Otomata Teorisi

Sunu VIII
Çıktılı Sonlu Otomata

İZZET FATİH ŞENTÜRK

Moore Machines

- The **motivation** of the finite automata is to <u>design a mathematical model</u> for a *computer*
  - The input string represents the program and input data
  - Reading the letters from the string is analogous to executing instructions which changes the state (memory) of the machine
  - How about the output?
- We could consider the output as part of the total state of the machine. This could mean two things:
  - To enter a state means change memory in a certain way and print a specific character
  - Or, a state includes both the present condition of memory plus the total output

Moore Machines

- In other words, the state could reflect (in addition to the status of the running program)
  - What we are printing
  - What we have printed in total
- A natural question to ask..
  - If we have these two different models, do tese machines have equal power or are there some tasks that one can do that the other cannot?
- The only explicit task of a machine has done so far is to recognize a language.
  - Computers often have more useful function of performing calculations and conveying results
  - This chapter expands the notion of machine task

Moore Machines

- If we assume that all the printing of output is to be done at the end of the program run
  - Then we have a maximum on the number of characters that the program can print (the size of the buffer)
  - Theoretically, we should be able to have outputs of any finite length
- We will investigate two different models for Fas with output capabilities
  - Created by G. H. Mealy (1955)
  - And independently by E. F. Moore (1956)

Moore Machines

- The **original purpose** of the inventors was to design a mathematical model for *sequential circuits*
- Sequential circuits are only one component of the architecture of a whole computer
  - It is an important component
  - It acts as a machine all by itself
- We will present these two models and prove that they are equivalent

Moore Machines - Definition

- A Moore machine is a collection of five things
  - A finite set of states q_0, q_1, q_2, …, where q_0 is designated as the start state
  - An alphabet of letters for forming the input string
    - Σ = {a b c …}
  - An alphabet of possible output character
    - Τ = {x y z …}
  - A transition table that shows for each state and each input letter what state is reached next
  - An output table that shows what character from Τ is printed by each state as it is entered

Moore Machines

- We do not assume that the input alphabet Σ and output alphabet T are same
- We refer to the input symbols as **letters**
- We call the output symbols **characters**
- The knowledge of which state is the start state is not always important in applications
    - If the machine is run several times, it may continue from where it left off rather than restart

Moore Machines

- We can define the Moore machines in two ways
  - Either the first symbol printed is the character always specified in the start state
  - Else, it is the character specified in the next state (the first state chosen by the input)
- We shall adopt the policy that a Moore machine always begins by printing the character dictated by the mandatory start state
  - This difference is not significant
  - If the input has seven letters, then the output string will have eight characters because it includes eight states in its path
- We shall say “printed” instead of “outputted”
  - Realize that the output device does not have to be a printer

Moore Machines

- A Moore machine does not define a language of accepted words
  - Every possible inout string creates an output string
  - There is no such thing as a final state
  - The processing is termInted when the last input letter is read and the last output character is printed
- There are several ways to turn Moore machines into language-definers
- The representation is very similar to FAs
  - We also specify the output character printed by that state after a slash

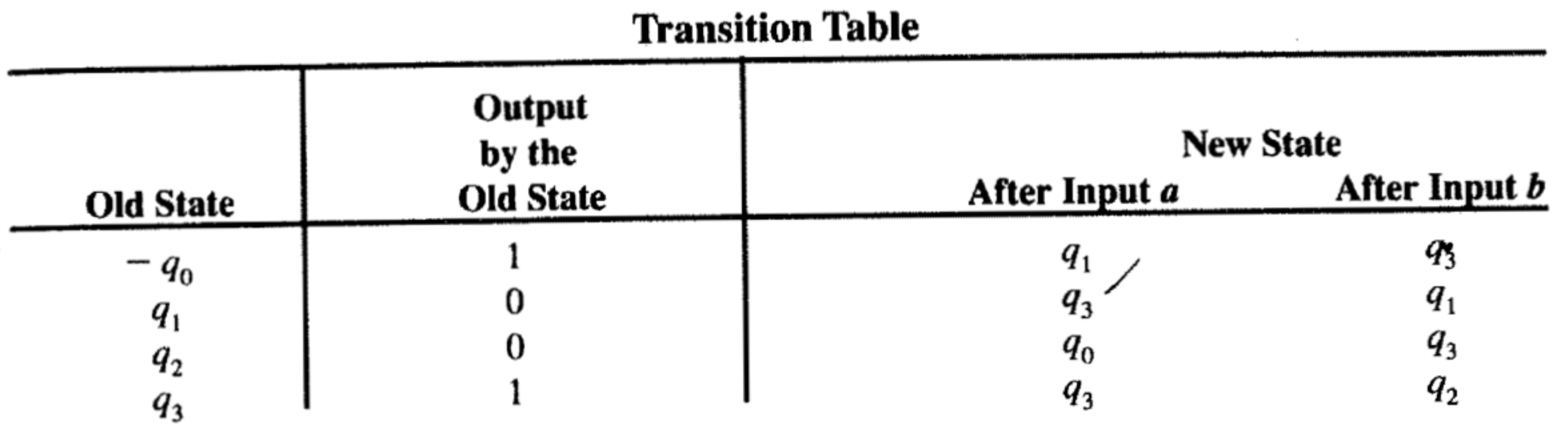
Example

Input alphabet: $\Sigma = \{a \quad b\}$

Output alphabet: $\Gamma = \{0 \quad 1\}$

Names of states: q_0, q_1, q_2, q_3 (q_0 = start state)

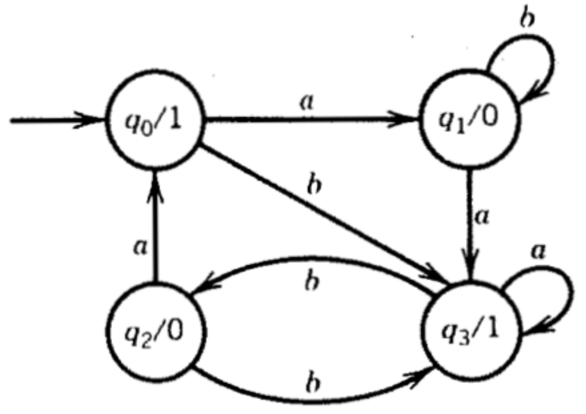
Transition Table



		New State	
Old State	**Output by the Old State**	**After Input *a***	**After Input *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

There is no room for the minus sign. Use an outside arrow instead

No need for any plus signs either



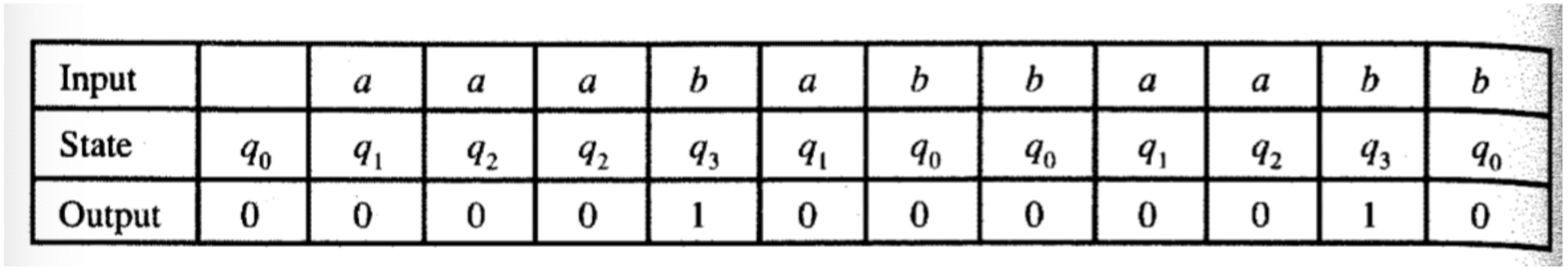
Trace the operation for: *abab*

The output is 10010

Example

- We want to know how many times the substring aab occurs in a long input string. The following machine will <u>count</u> this for us

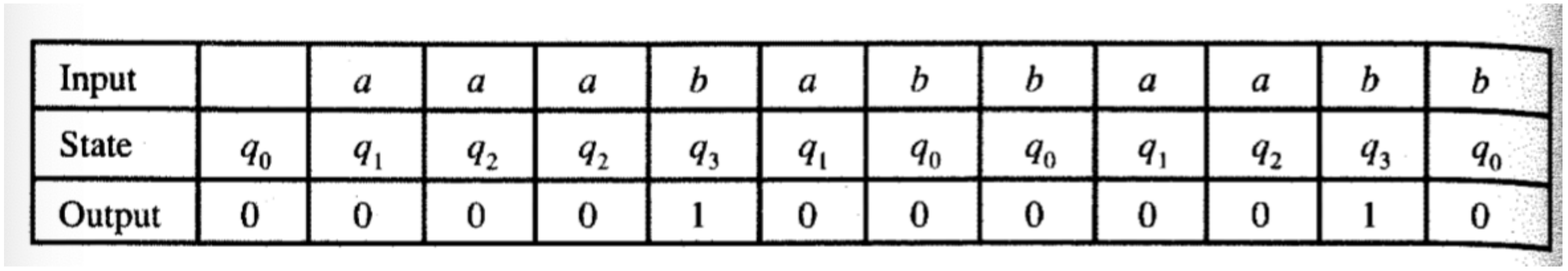
$$\begin{tikzcd}
\arrow[r] & q_0/0 \arrow[loop above, "b"] \arrow[r, bend left, "a"] & q_1/0 \arrow[l, bend left, "b"] \arrow[r, "a"] & q_2/0 \arrow[loop below, "a"] \arrow[r, "b"] & q_3/1 \arrow[ll, bend right=40, "a"'] \arrow[lll, bend left=40, "b"]
\end{tikzcd}$$



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

Moore Machines

- Given a language *L* and an FA that accepts it, if we print **0** at nonfinal states and **1** at each final state
  - The 1's in any output sequence mark the end position of all substrings of the input string starting from the first letter that are words in *L*
  - A Moore machine can *define* the language of all input strings whose output ends in a 1
  - The machine we just saw accepts all words that end in aab with q_0 = - and q_3 = +



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

Mealy Machines

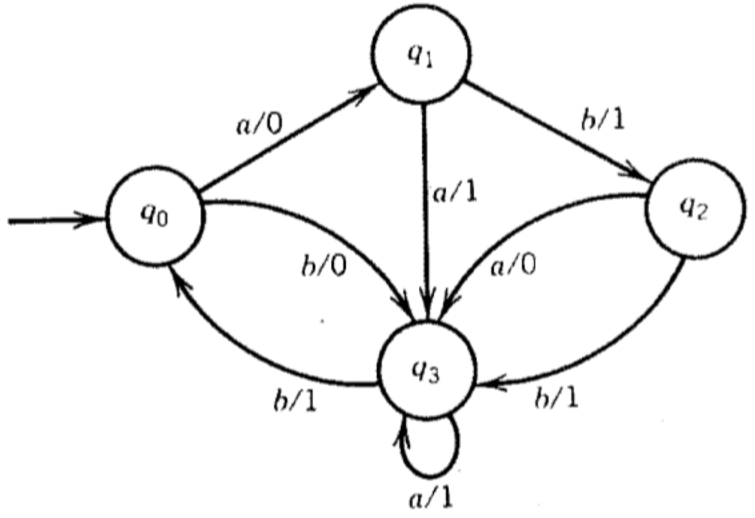
- A Mealy machine is like a Moore machine except..
  - We do printing on the edges, not in the states
- If we are in state q_4 and we are proceeding to q_7, we do not simply print what q_7 tells us
  - What we print depends on the edge we take
- If there are two different edges from q_4 to q_7, an a-edge and a b-edge
  - It is possible that they have different printing instructions

Mealy Machine - Definition

- A Mealy machine is a collection of four things
  - A finite set of states q_0, q_1, q_2, …, where q_0 is designated as the start state
  - An alphabet of letters Σ = {a b c …} for forming input strings
  - An alphabet of output characters T = {x y z …}
  - A pictorial representation with states denoted by small circles and directed edges indicating transitions between states. Each edge is labeled with a symbol *i/o*, where *i* is an input letter and *o* is an output character. Every state must have <u>exactly one outgoing edge</u> *for each possible input letter*. The edge we travel is determined by *i*. While traveling on the edge, we print *o*

Example

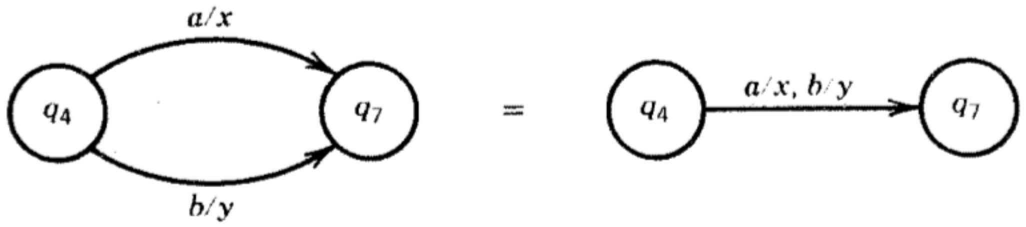
- Trace the running of this machine on the input sequence aaabb
  - The output for this input is 01110



- In a Mealy machine, the output string has the same number of characters as the input string has letters

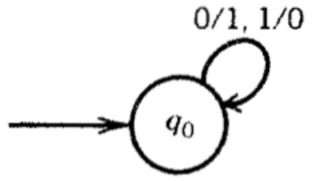
Mealy Machines

- If there are two edges going in the same direction between the same pair of states, we can draw only one arrow and represent the choice of label by a comma



Example

- A Mealy machine that prints out 1's complement of an input bit string
- If the input is 001010, the output is 110101
- This is a case where input alphabet and output alphabet are both {0 1}



Example

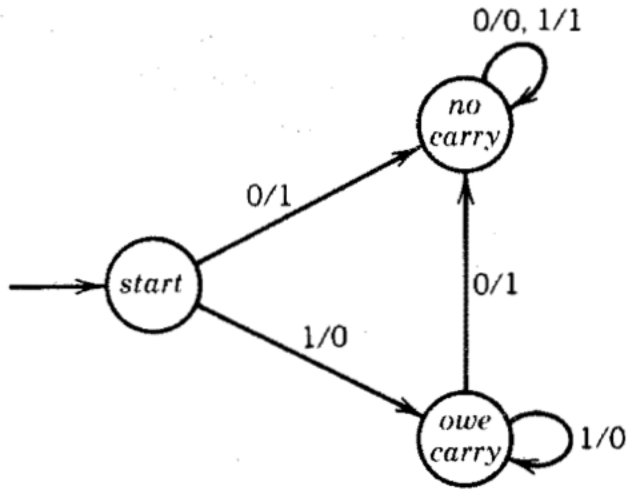
- A Mealy machine called the **increment machine**
  - Assumes that its input is a binary number
  - Prints out the binary number that is one larger
  - We assume that the input bit string is a binary number fed in backward (units digit first, then 2’s digit, 4’s digit, …)
  - The output will be the binary representation of the number one greater
  - The output will also be generated right to left

Example

- The machine will have three states: start, owe-carry, no-carry
  - The owe-carry state represents the overflow when two bits equal to 1 are added – we print a 0 and we carry a 1

Trace the execution for the number 11 (1011)

The string is fed into the machine as 1101



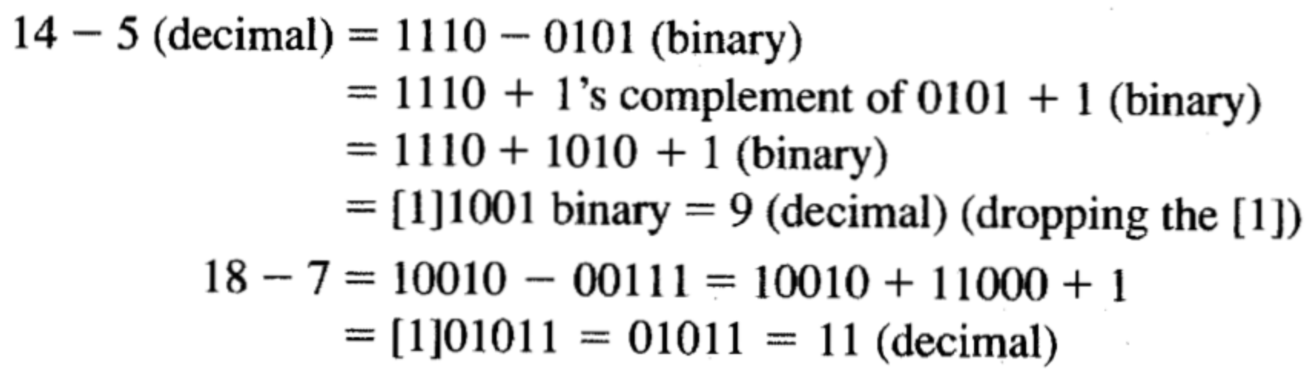
The output string is 0011 (1100 when reversed)

If we run this machine for 1111, we would get 0000

Mealy Machines and Sequential Circuits

- The connection between Mealy machines and sequential circuits makes them a *very valuable component* of **computer theory**
- We have seen an incrementer and a machine that computes 1’s complement. Now we can build a machine to perform subtraction
- If a and b are strings of bits, then a-b can be performed by
  - Adding the 1’s complement of b to a, ignoring any overflow digit
  - Incrementing the results by 1

Example



$$\begin{aligned}14 - 5 \text{ (decimal)} &= 1110 - 0101 \text{ (binary)}\\ &= 1110 + 1\text{'s complement of } 0101 + 1 \text{ (binary)}\\ &= 1110 + 1010 + 1 \text{ (binary)}\\ &= [1]1001 \text{ binary} = 9 \text{ (decimal) (dropping the [1])}\end{aligned}$$

$$\begin{aligned}18 - 7 &= 10010 - 00111 = 10010 + 11000 + 1\\ &= [1]01011 = 01011 = 11 \text{ (decimal)}\end{aligned}$$

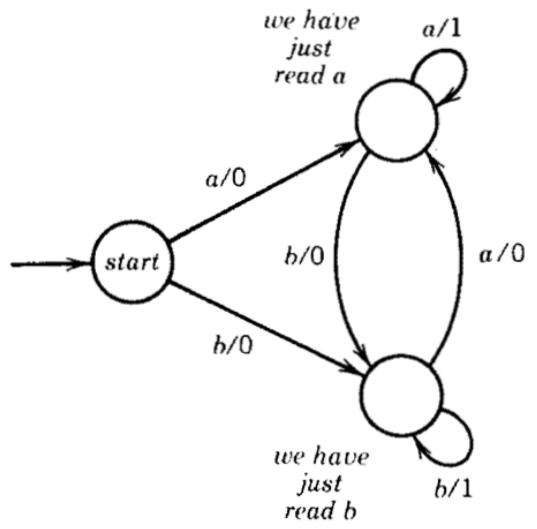
- The same trick works in decimal notation if we use 9's complements
  - Replace each digit *d* in the second number by the digit (9-d)
- For example: 46 – 17 -> 46 + 82 + 1 = [1]29 -> 29

Example

- Mealy machine does not accept or reject an input string
    - But it can recognize a language by making its output string answer some questions about the input
- We have discussed the language of all words that have a double letter in them
- The Mealy machine below will take a string of a's and b's and print out a string of 0's and 1's
    - If the *n*th output character is a 1, it means that the *n*th input letter is the second in a pair of double letters
    - For example, ababbaab becomes 00001010 with 1's in the position of the second of each pair of repeated letters

Example

- For example, *ababbaab* becomes 00001010 with 1's in the position of the second of each pair of repeated letters



- This machine recognizes the occurrences of *aa* or *bb*. The triple-letter word *aaa* produces the output 011

Moore = Mealy

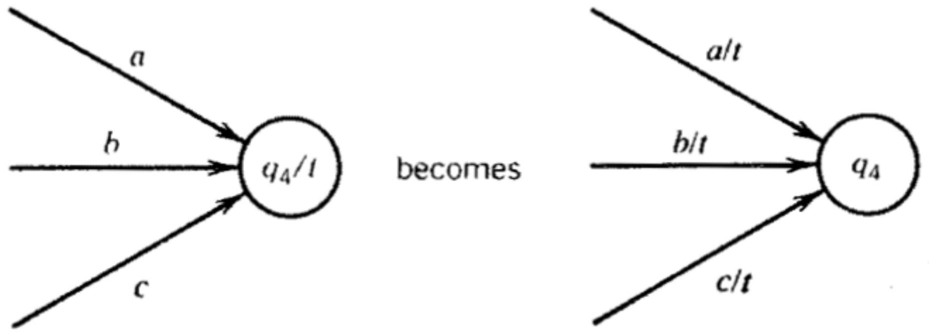
- We may say that two output automata are equivalent if they always give the same output string when presented with the same input string
  - Two Mealy machines can be equivalent
  - Two Moore machines can be equivalent
  - A Moor machine can never be <u>directly</u> equivalent to a Mealy machine – The output string from a Moore machine is one longer
- To get around this difficulty, the automatic start symbol for the Moore machine is deleted from the front of the output

Moore = Mealy

- Given the Mealy machine *Me* and Moore machine *Mo*, which prints the automatic start-state character x, these two machines are equivalent
  - If for every pair of input string the output string from *Mo* is exactly x concatenated with the output from *Me*
- We prove that for every Moore machine, there is an equivalent Mealy machine
- And for every Mealy machine, there is an equivalent Moore machine
- Then, two types of machines are functionally equivalent

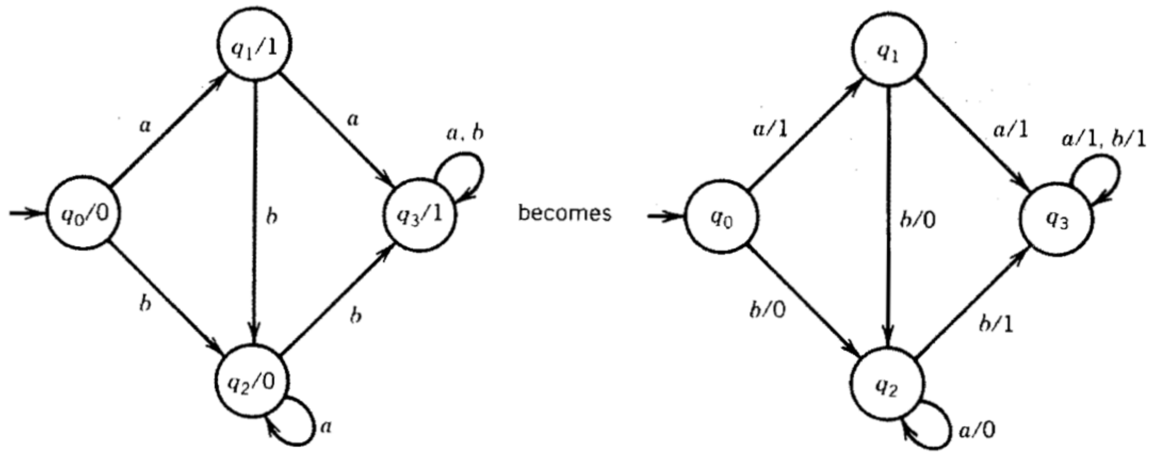
Moore = Mealy

- If *Mo* is a Moore machine, then there is a Mealy machine *Me* that is equivalent to it
- Consider a state q_4 in *Mo* which prints the character *t*
    - Consider all edges that enter this state
    - Each edge has a label (*a*, *b*, *c*, …), relabel them (*a*/*t*, *b*/*t*, *c*/*t*, …)
    - Erase *t* from inside the sate q_4
    - This means that we shall print a $_t$ on the incoming edges before they enter q_4



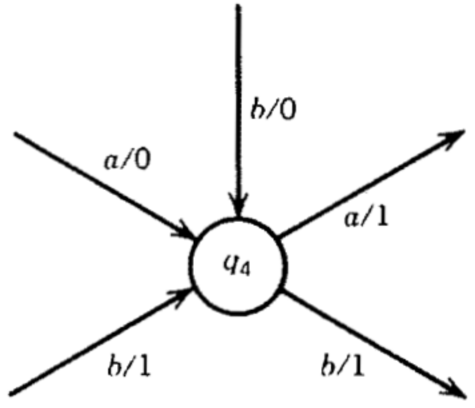
Moore = Mealy

- We leave outgoing edges from q_4 alone. They will be relabeled with the associated state they lead
- If we repeat this procedure for every state q_0, q_1 , …, we turn *Mo* into a Mealy machine *Me*
- Therefore, every *Mo* is equivalent to some *Me*
- Below is an example *Mo* is converted to a *Me*



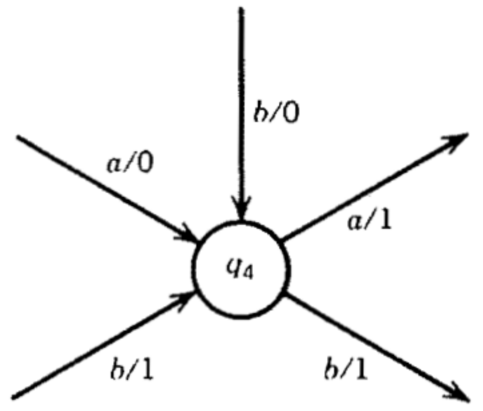
Mealy = Moore

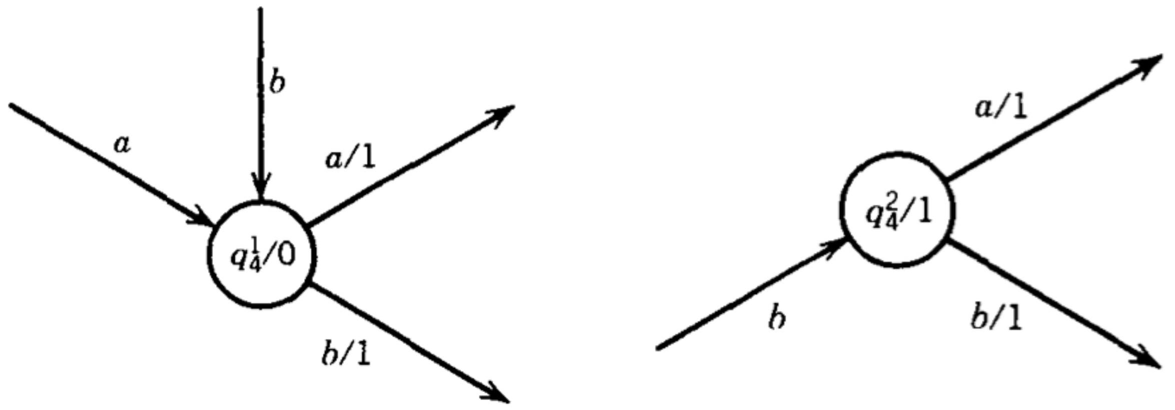
- For every Mealy machine *Me*, there is a Moore machine *Mo* that is equivalent to it
- We cannot do the reverse of the previous procedure
  - We might end up with a conflict
  - Two edges might come into the same state but have different printing instructions



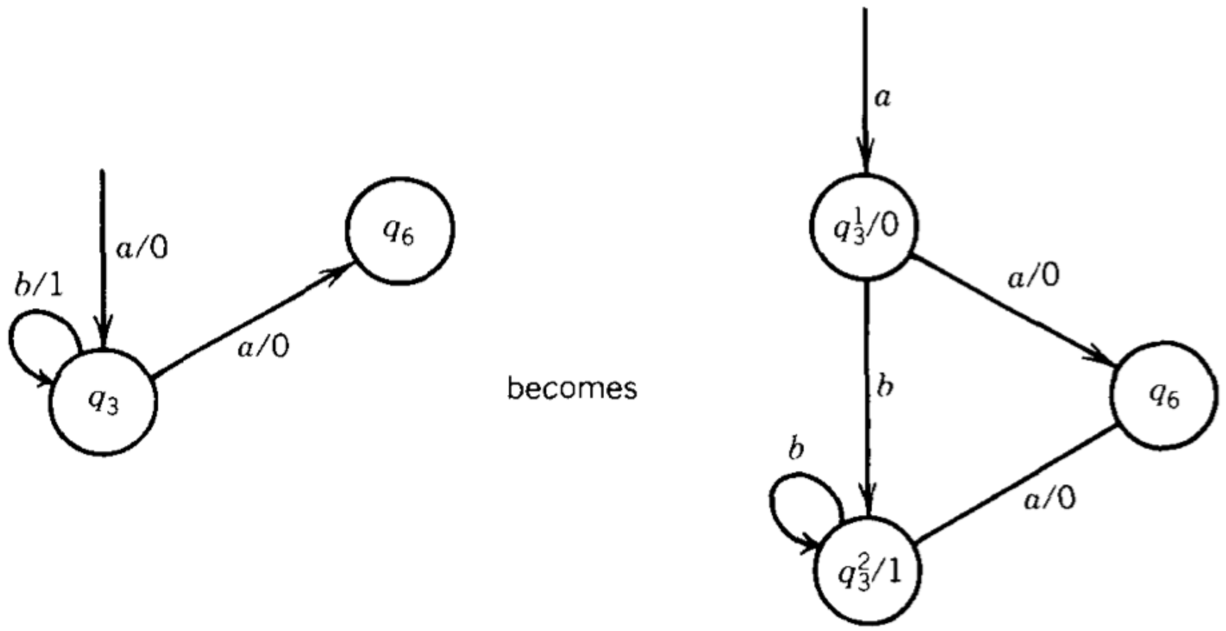
Mealy = Moore

- We need twin copies of the same state
  - The edge a/0 will go into q_4^1 (q_4 copy 1)
  - The edge b/1 will go into q_4^2 (q_4 copy 2)
  - The edge b/0 will also go into q_4^1
  - Insert printing instructions in the new states





Mealy = Moore


$a/0$
$b/1$
q_3
$a/0$
q_6
becomes
a
$q_3^1/0$
$a/0$
b
q_6
b
$a/0$
$q_3^2/1$