**CPSC 323, Handout No. 8**

**FINITE AUTOMATA WITH OUTPUT**

The finite automata we studied so far traced an input string and returned “Accept” or “Reject” whether the input string end up at a final state or not. In this handout we introduce 3 finite automat which are able to generate an output based the input string. Those machines are:

1. **Moore Machine** (Mo) Introduced by E.F. Moore in 1956
2. **Mealy Machine** (Me) Introduced by G.H. Mealy in 1955
3. **Turing Machine** (TM) introduced by Alan Turing. Turing's most notable work today is as a **computer scientist**. In 1936, he developed the idea for the Universal Turing Machine, the basis for the first computer. And he developed a test for artificial intelligence in 1950, which is still used today.
4. **Moore Machines.** In this machine while we go from state qi to another state qj, for every input letter “a”, the machine will generate an output “b” at state qj.

Input: a

A Moore machine is a collection of the following:

1. A finite set of states {q0, q1, …….., qn} where q0 is the initial state
2. Input alphabet set { a,b,c,….. }
3. Output alphabet ( x,y,……..}
4. A transition table or pictorial representation of the machine

**Example.** Suppose we want to construct a Moore machine to read a binary string and produce its 1’s complement. Recall: input:11011 should produce output: 00100 (from left-to-right, change 1 to 0 and o to 1)

***Sample I/O:*** Let’s consider a Moore machine defined by set of states={q0, q1, q2 }, input alphabet {0,1}, output alphabet {0,1}, and the following pictorial representation of the machine.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input |  | 1 | 1 | 0 | 1 | 1 |
| Output | 0 | 0 | 0 | 1 | 0 | 0 |

The shaded box is a default output when we start at q0

1

start 1 0 0

0 1

Trace binary string : 11011 as input and show its final output

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **input** |  | 1 | 1 | 0 | 1 | 1 |
| **state** | q0 | q1 | q1 | q2 | q1 | q1 |
| **output** | **0** | 0 | 0 | 1 | 0 | 0 |

The output string has one extra **0** that produced at q0 and is not effecting the value of the final string.

**Example.** Suppose we were interested in knowing exactly how many times the substring **aab** occurs in an input string of a’s and b’s. First construct a Moore machine based on substring **aab**

**a**  **a**  **b**

to make sure the substring aab ends up at state q3 to produce 1 as output. Now complete each state when the inputs are a or b.

b **a**  **a**  **b**

**b a**

**a**

**b**

Now, trace the machine for input string: aaababbaab

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** |  | a | a | a | b | a | b | b | a | a | b |
| **State** | q0 | q1 | q2 | q2 | q3 | q1 | q0 | q0 | q1 | q2 | q3 |
| **output** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

The number of 1’s in output string indicates the number of occurrence of substring aab in the input string

1. **Mealy machine.** A Mealy machine is like a Moore machine except that now we do our printing while we are traveling along an edge and not in state. Consider the following to see the difference between Moore and Mealy machine

|  |  |
| --- | --- |
| **Moore machine** | **Mealy machine** |
| **Input:a**  **Output b printed when entered qj** | **Input:a/output:b**  **Output b printed before entered qj** |

**Example.** Construct a Mealy machine to display the 1’s complement of binary strings

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| start o/1  1/0 | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **input** | 1 | 0 | 1 | 1 | 0 | 1 | 1 | | **state** | q0 | q0 | q0 | q0 | q0 | q0 | q0 | | **output** | 0 | 1 | 0 | 0 | 1 | 0 | 0 | |

**Example.** Construct a Mealy machine to display the 2’s complement of a binary string. The input string traced from right-to-left.

***Recall.*** Consider the following examples to come up with an algorithm to find the 2’s complement in one step

|  |  |
| --- | --- |
| Find the 2’c complement of  0110 1111  1001 0000 (i)find 1’s complement  + 0000 0001 (ii)add 1 to 1’s comp to  1001 0001 Find 2’s complement | Find the 2’s complement of  1100 1100  0011 0011 (i) find 1’s complement  + 0000 0001 (ii) add 1 to find 2’s  0011 0100 complement |
| ***Algorithm*** to find the 2’s complement of a binary number in one step.  Start from right leave zeros as zero until you hit the first 1. Leave 1 unchanged, continue to left and replace each binary digit with its 1’s complement | |

**Example.** This is the Mealy machine display the 2’s complement of binary strings :

0110 1111 (machine trace the input string from right-to-left

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0/0 0/1  start 1/1    1/0 | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **input** | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | | **state** | q1 | q1 | q1 | q1 | q1 | q1 | q1 | q0 | | **output** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   Trace the input from right-to-left |

**Example** . Construct a Mealy machine to count the number of substring 001 in a given binary string.

Start constructing part of the machine to produce an output 1 when the input is 001. Note: input letters {0,1}, output letters {0,1}

**0**/0 **0**/0 **1**/1

Now complete each state when the input is 0 or 1. 0/0

1/0 1/0 0/0

start 0/0 0/0 1/1

1/0

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **input** | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| **state** | q0 | q1 | q2 | q2 | q3 | q1 | q0 | q0 | q1 | q2 | Q3 |
| **output** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Trace input string: w= 0001 0110 011 from left-to-right

Hence, there are two substring 001 in the input string w.

1. **Turing Machines.**  A Turing machine consist of the following parts
2. **Input/output tape (** # indicates blank, the content of each box is a single character). For input string abaa, the tape look like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **a** | **b** | **a** | **a** | **# # …………………….** |

1. **The read/write head.** The head can move to right, left, and also able to change the content of the box pointing at
2. **Control unit.** Control unit shows the current position of read/write head and the current state
3. **Set of input alphabets**
4. **Set of states q0,q1,q2,…..** with initial state q0

When the machine is in the following state,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **a** | **b** | **a** | **a** | **b** | **# # …………………….** |

q0

q1

q2

We use (q1, #abaab#) to represent the current state. Means while the machine is at state q1, the head is pointing to the letter b.

The following are ***predefined Turing machines***

1. (q1,#abaab#)🡪**R**🡪 (q2,#abaab#), The **R**ight machine moves the head one square to right
2. (q2,#abaab#)🡪**L**🡪 (q1,#abaab#), The **L**eft machine moves the head one square to left
3. (q1,#abaab#)🡪**Rb**🡪(q2,#abaab#), move the head to the first b (b is subscript)on the right
4. (q2,#abaab#)🡪**L#**🡪(q2,#abaab#) , move the head to the first blank (# is subscript) on the left side
5. (q2,#abaab#)🡪**Rc**🡪(q2,#cbaab#), move the head one square to right and replace the current content of the box with letter c( c is not subscript )
6. Move the head to right one square, if

a

**R**  head is pointing at “a”, then follow this direction

b head is pointing at “b” follow this direction

1. #

R move the head one square to right, follow this direction if the head

is pointing at blank

# Follow this direction if the head is not pointing to blank

1. r ≠ #

R move the head to right one square, r = the content of that square if

it is not blank

# Follow this direction if the content of the new box is blank



1. (q1, #abaab#)🡪R#c🡪(q2, #abbaabc#), moves the head to the first blank on right side (R#) and replace it with letter c



**Example** . Trace the following Turing machine for input:#abc# ( the head initially points to the first blank on the right-hand-side)

r≠#

Start L# R # R#R# r L#L# r

#

R#

|  |  |  |
| --- | --- | --- |
| **action** | **description** | **Input/output string** |
| Start |  | #abc#there are more# on the right-hand-side |
| 🡪L# | move the head to the first # on left | #abc## |
| 🡪R, | move head one letter to right | #abc## |
| The head is pointing at a, so r=a. |  | #abc## |
| follow 🡪 # | Replace a with # | ##bc## |
| 🡪R# | move the head to first # on right(there are more # on right) | ##bc##, |
| 🡪R# r | and replace it with r=a (more # on right) | ##bc#a # |
| L#  L# r, | move to the first # on left,  and move to the next blank on left agian( together means move to the 2nd blank on left)and replace it with r =a. before we go back to the loop, all we did so far is to put a copy of ‘a’ on the right hand side | ##bc#a#  #abc#a# |
| R🡪# | Move head to right, r=b which is not blank. Replace it with # | #a#c#a# |
| R#R# r | Lets do all at once. Move to the second blank on right and replace # with r=b | #a#c#ab# |
| L#L# r, | Move to the second blank on Left, and replace the # with r=b | #abc#ab# |
| R🡪 # | Move the head to Right, r=c not blank. Replace c with # t | #ab##ab# |
| R#R# r | Move to the second blank on right and replace # with r=c | #ab##abc# |
| L#L# r, | Move to second blank on left and replace it with r=c | #abc#abc# |
| R | Move head to right, the box is # | #abc#abc# |
| R# | Move head to first # on right | #abc#abc# |

In summary, this machine used the input string #abc# to produce output #abc#abc#.

Thus, the input string #w# creates output string #w#w#, or word w followed by its copy. We name this machine a COPY machine or C, and from now on you can use it when you need it.

**Notation.** Since #w# was the input to the Turing machine and #w#w# was its output, we use function notation to describe its action. If the function name be f, then f(#w#) = #w#w# or #w# C #w#w#, notice that at the end we want head to point to the right most blank.

**Example**. The following Turing machine is called Shift Left (SL) which does

f(#w#) = w#.

To construct this machine, start with a simple example to come up with an algorithm. Suppose w=#abc#.

Move the head to the blank on left: #abc#

Loop: Go one square to right ( R), get the letter r=a, if r is not # go Left and replace it with r: Lr. Go to right R, and back to the loop.

Now, our algorithm look like this

Go to first blank on left (L# )

ii. loop: go to right(R), let r be the content of the box.

If r is not #, use L r R and go back to loop

If r is #, then move left and replace it with #

The TM machine will look like this. Now, lets trace this machine for input string **w=#abc#**

Start L#  R r ≠# L r R

#

L#

Trace #abc#

|  |  |  |
| --- | --- | --- |
| **Action** | **Description** | **I/O string** |
| start |  | #abc# |
| L# | Move to # on left | #abc# |
| R | Move to Right, r=a | #abc# |
| L r R | Move left replace the box with r. move to right | aabc# |
| R | Move to right, r=b | aabc# |
| L r R | Move left, replace the letter with b. move right | abbc# |
| R | Move to right, r=c | abbc# |
| L r R | Go left, replace the letter with c. move right | abcc# |
| R | r=#, | abcc# |
| L#  DONE | Go left and replace it with # | abc# |

Now, if we use the combination of these two Turing machines we obtain:

#w# -------C ------🡪 #w#w# ----- SL ----------🡪 #ww#

**Example.** Construct a Turing machine to compute f(#w#) = #wwR#. That is for input w=#abc#, the output should look like #abccba#.

Algorithm: Suppose w= #ab#

Loop:Go left

if r=b ≠ #, replace it with #( we will use the # as a flag to come back to this box

again and put b back. Go to first # on right and put r=b (R# r ). Go to first blank

on left and replace it with r=b (L# r ). Go back to the loop, until r = #, move head to the first blank on right (R#)

DONE

r ≠#

start L # R# r L#  r

#

R#



Trace #ab#

|  |  |  |
| --- | --- | --- |
| **Action** | **Description** | **I/O string** |
| start | L, move head to left, r=b ≠ # | #ab# |
| # R# r | Replace r=b with #. Move to first # on right side and replace it with r=b | #a#b# |
| L# r | Move to first # on left and replace it with r=b | #abb# |
| L | Move left, r=a ≠ # | #abb# |
| # R# r | Replace r=a with #. Move right to #, replace # with r=a | ##bba# |
| L# r | Move left to first # and replace it with r=a | #abba# |
| L | r = # | #abba# |
| R# | Move to # on right | #abba# |

CPSC 323 , Assignment No. 10 Names:

1. (10 points) Construct a Mealy machine to find the 2’s complement of a binary string. The machine reads the string from right-to-left

2’s complement is essentially 1’s complement + and additional 1.

Example: 11011011

1 1 0 1 1 0 1 1

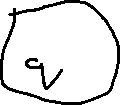


0 0 1 0 0 1 0 0



+ 0 0 0 0 0 0 0 1

0 0 1 0 0 1 0 1



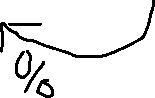
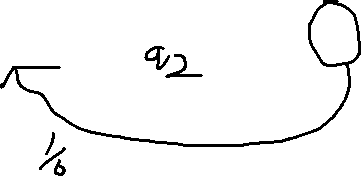
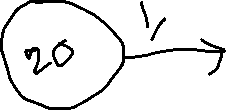
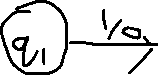
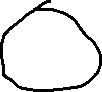
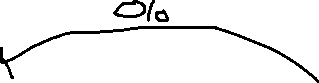
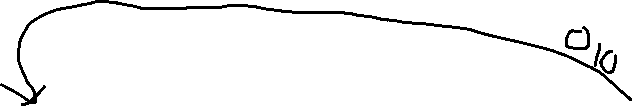
Example: 101000 input it backwards

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input | 0 | 0 | 0 | 1 | 0 | 1 |
| State | q0 | q0 | q0 | q1 | q1 | q1 |
| Output | 0 | 0 | 0 | 1 | 1 | 0 |

Final output: 011000



1. (10 points). Construct a Mealy machine to count the number of substring 111 in a given binary string. The number of 1’s in output string should indicate the number of occurrence of 111 in the input string. Suppose the input string is 110 011 111 101 111



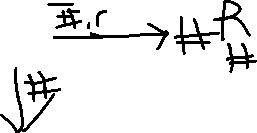
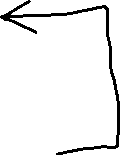
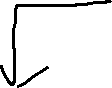
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input |  | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| State | q0 | Q1 | Q2 | Q0 | Q0 | Q1 | Q2 | Q3 | Q1 | Q2 | Q3 | Q0 | Q1 | Q2 | Q3 | Q1 |
| Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

1. (10 points) Construct a Turing machine to compute f(#w#) = #wwR#. Trace your machine

using w=#abc#



suppose w = abc



L = left, r = variable, R = Right

# a b c # | L



# a b c # | r = c

# a b c # | #

# a b # # | R#

# a b # # # | r

# a b # c # | L#

# a b # c # | r

# a b c c # | L

# a b c c # | r = b

# a b c c # | #

# a # c c # | R#

# a # c c # # | r

# a # c c b # |L#

# a # c c b # |r

# a b c c b # |L

# a b c c b # |r = a

# a b c c b # |#

# # b c c b # |R#

# # b c c b # |r

# # b c c b a # |L#

# # b c c b a # | r

# a b c c b a # | L

# a b c c b a # | R#

# a b c c b a #

Thus it returns f(#w#) = #wwR#