Lecture 2: Welcome to the Machine

Michael Engling

Department of Computer Science
Stevens Institute of Technology

CS 334 Automata and Computation
Fall 2015

OUTLINE: Five Machines

Our First Machine Redux: Machine M_1

Switcheroo: $\hat{M}_1$

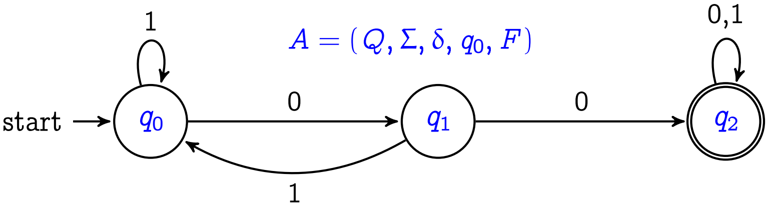
Machine M_2

Machine M_3

Machine XY

Our First Automaton Redux

$A = (Q, \Sigma, \delta, q_0, F)$



$Q = \{q_0, q_1, q_2\}$

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

$\delta : Q \times \Sigma \to Q$

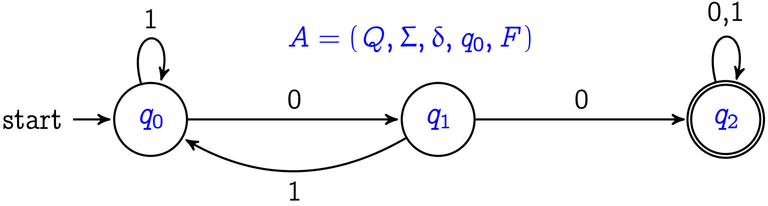
$q_0 = q_0$

$F = \{q_2\}$

$\delta((q_0, 0)) \to q_1$	$\delta((q_0, 1)) \to q_0$
$\delta((q_1, 0)) \to q_2$	$\delta((q_1, 1)) \to q_0$
$\delta((q_2, 0)) \to q_2$	$\delta((q_0, 1)) \to q_2$

See why q_2 is special?

Our First Automaton Redux



$$A = (Q, \Sigma, \delta, q_0, F)$$

$Q = \{q_0, q_1, q_2\}$

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

δ :

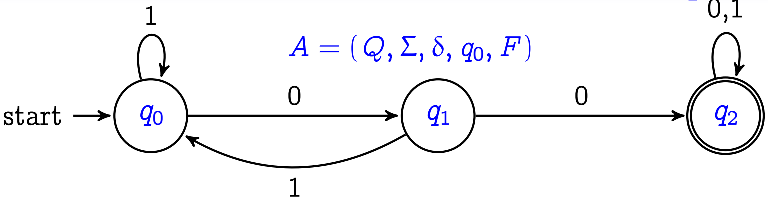
	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_2

$q_0 = q_0$

$F = \{q_2\}$

Our First Automaton Redux: M_1

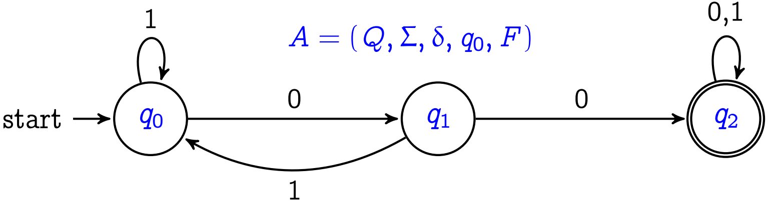
$$A = (Q, \Sigma, \delta, q_0, F)$$



This is the **State Diagram** for our automaton; let us call it M_1 The (singleton) start state is clearly labeled. The set of final states, F, is in this case also a singleton, q_2. It is identified by the double circle.

We "compute" with such an automaton by feeding it strings: String 1010 begins at q_0 and transitions to q_0 with a '1'. It then continues to q_1 with '0', returns to q_0 with the second '1' and again to q_1 with the last '0'. It ends in q_1 which is not a final state what we will call an **accepting state**, and thus M_1 **rejects** 1010.

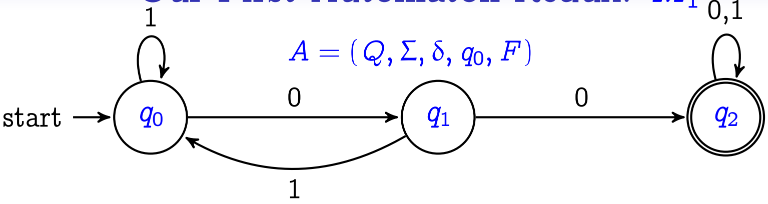
Our First Automaton Redux: M_1



This is the **State Diagram** for our automaton; let us call it M_1
The (singleton) start state is clearly labeled. The set of final states, F, is in this case also a singleton, q_2. It is identified by the double circle.

We "compute" with such an automaton by feeding it strings:
String 1001 begins at q_0 and transitions to q_0 with a '1'. It then continues to q_1 with '0', along to q_2 with the second '0' and remains in q_2 with the last '1'. It ends in q_2 which *is* an **accepting state**, and thus M_1 **accepts** 1001.

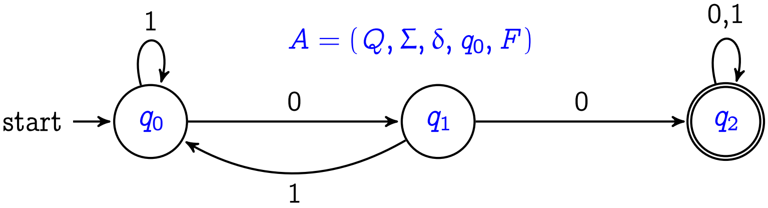
Our First Automaton Redux: M_1



The question before us now is, precisely what strings does M_1 accept?

ε	$\times$	011	$\times$
0	$\times$	100	$\surd$
1	$\times$	101	$\times$
00	$\surd$	110	$\times$
01	$\times$	111	$\times$
000	$\surd$	0000	$\surd$
001	$\surd$	0001	$\surd$
010	$\times$	0010	$\surd$

Our First Automaton Redux: M_1



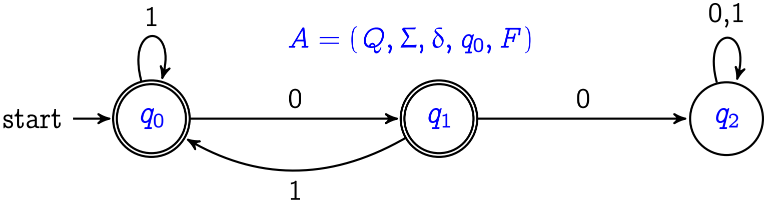
M_1 accepts all strings over $\Sigma = \{0, 1\}$ (our alphabet) which contain '00' as a substring, and no others.

These strings are a set; they are a set of strings. That is the definition of a *language*. Let L_1 be the set of bitstrings which contain '00' as a substring.
We say M_1 *recognizes* L_1.

M_1 accepts every element of L_1 and no other strings.

L_1 is the language of M_1.

Switcheroo: $\hat{M}_1$

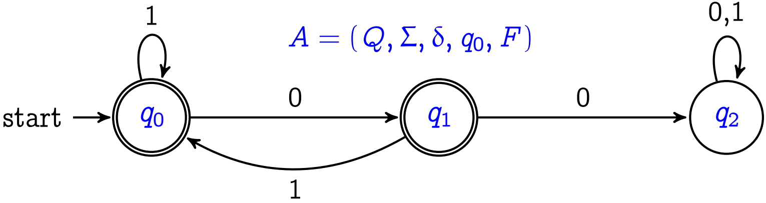


$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$

$$\delta : \begin{array}{c|cc} & 0 & 1 \\ \hline q_0 & q_1 & q_0 \\ q_1 & q_2 & q_0 \\ q_2 & q_2 & q_2 \end{array}$$

$q_0 = q_0$
$F = \{q_0, q_1\}$

Switcheroo: $\hat{M}_1$



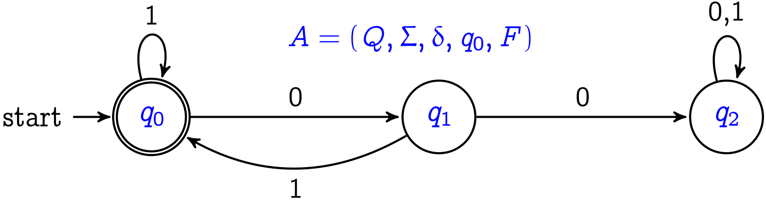
What is the language of $\hat{M}_1$?

It is easy to see, after a little "computation", that $\hat{M}_1$ accepts all strings which do *not* have '00' as a substring. So $\hat{L}_1$ is the set of all strings that do not have '00' as a substring.

Then in some real sense $L_1 \cup \hat{L}_1$ is the set of all strings we are considering. That should be our universal set U:

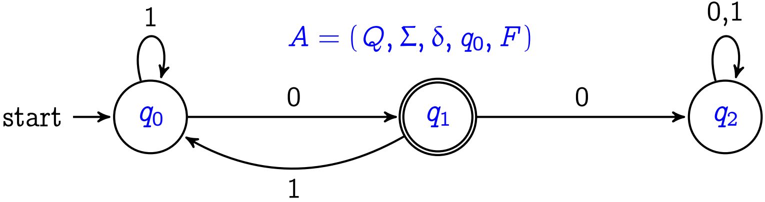
$$L_1 \cup \hat{L}_1 = U = \Sigma^*$$

M_2



What is L_2, the language of M_2?

M_3



What is L_3, the language of M_3?

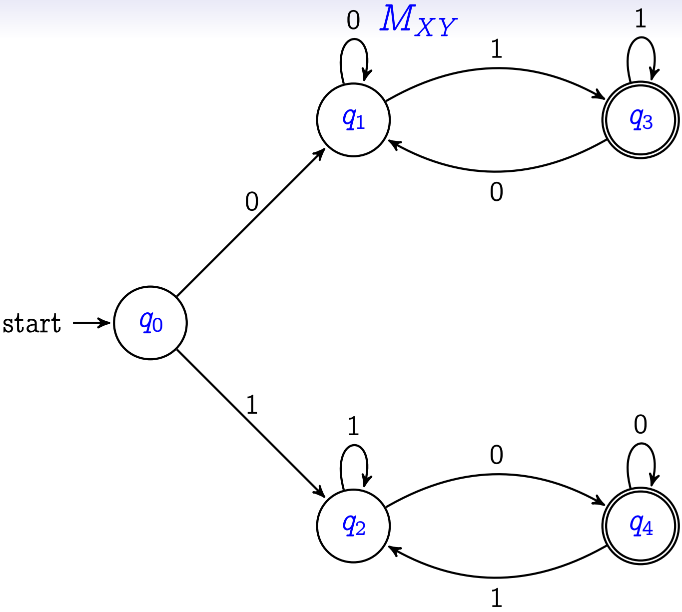
Partitioning of Σ^*

We did something cheeky with M_1 and M_2 and M_3.

Because each state was an accepting state for one of the automata, every string was accepted by one of them.

So we obviously get $L_1 \cup L_2 \cup L_3 = U = \Sigma^*$.

But we get more than that. Each state was an accepting state *exactly once*. Thus each string is in exactly one of those languages. Our automata *partition* Σ^*. Every string appears in exactly one of those languages. (And none of those languages is empty.)



What is L_{XY}, the language of M_{XY}?

The language of M_{XY}

The language of M_{XY} is all bitstrings (strings over $\Sigma = \{0, 1\}$) which begin and end with different symbols.

We can write that as $L_{XY} = 0\Sigma^*1 \cup 1\Sigma^*0$

So mathematically transliterated:

A bitsring in L_{XY}

begins with 0 and ends in 1

OR

begins in 1 and ends in 0

(and anything might come between the first and last symbols, even ε the empty string.)

What CAN We Compute?

So now the question is...what can we *compute* with these simple machines, these Deterministic Finite Automata?

Can we recognize the language of all strings of even length?
How about prime length?
Can we recognize the language of all palindromes?
How about all strings with six specified substrings?

Is there anything we *can't* compute?