

### Algorithm to Minimize DFAs

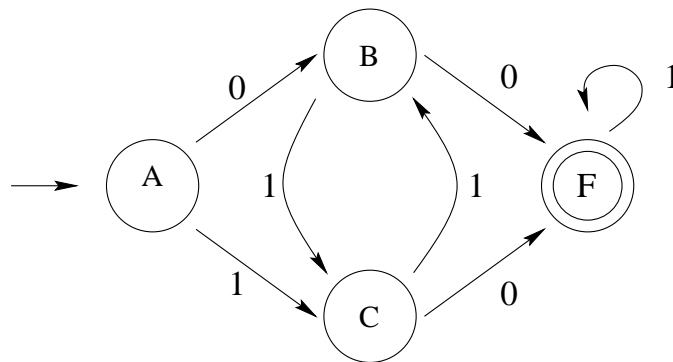
1. Mark final states and non-final states as distinguished.
2. Recursively iterating over all pairs of states.  
For any transition of the form

$$\delta(q_i, x) = q_j \quad \delta(q_k, x) = q_l$$

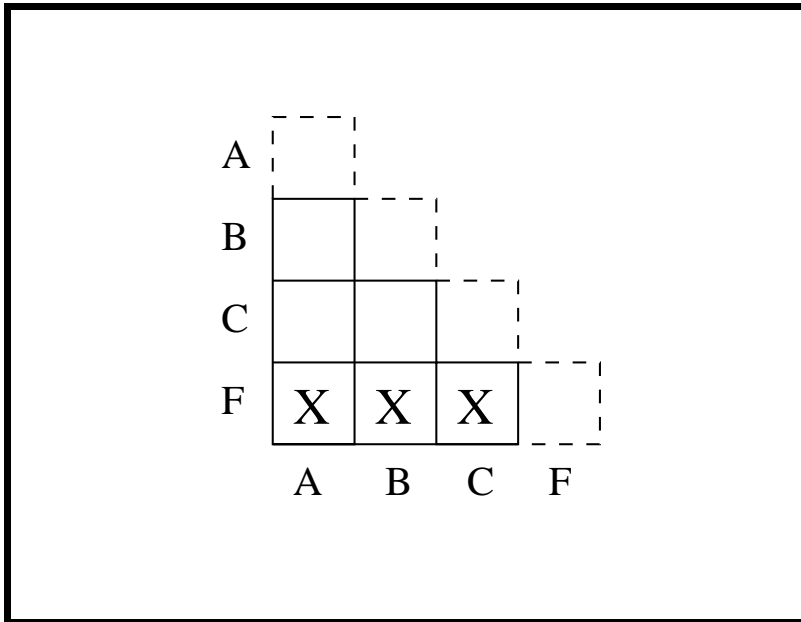
and for some  $x \in \Sigma$ , if  $q_j$  and  $q_l$  are distinguishable, mark  $q_i$  and  $q_k$  as distinguish.

3. If on any iteration over all possible state pairs one fails to find a new pair of states that are distinguishable, terminate.
4. All states that are not distinguishable are equivalent.

Slide Lecture 7 –185



Slide Lecture 7 –186



Slide Lecture 7 -187

Consider pair (A,B)

$$\delta(A, 0) = B \quad \delta(B, 0) = F$$

$$\delta(A, 1) = C \quad \delta(B, 1) = C$$

$A \equiv B$  IFF  $B \equiv F$  and  $C \equiv C$ .

Since B is distinguishable from F, A is distinguishable from B.

Slide Lecture 7 -188

Consider pair (A,C)

$$\delta(A, 0) = B \quad \delta(C, 0) = F$$

$$\delta(A, 1) = C \quad \delta(C, 1) = B$$

Since B is distinguishable from F, A is distinguishable from C.

**Slide Lecture 7 -189**

Consider pair (B,C)

$$\delta(B, 0) = F \quad \delta(C, 0) = F$$

$$\delta(B, 1) = C \quad \delta(C, 1) = B$$

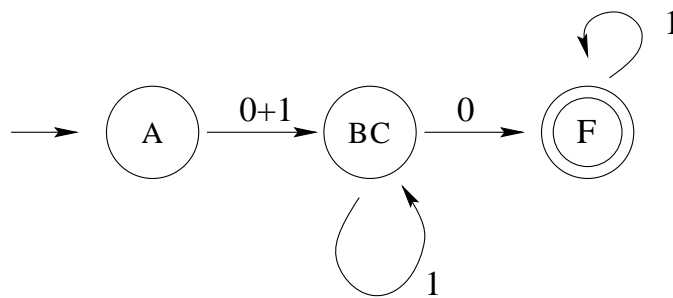
$$B \equiv C \quad \text{IFF} \quad C \equiv B.$$

**Slide Lecture 7 -190**

A				
B	X			
C	X			
F	X	X	X	
	A	B	C	F

A circular dependency exists. Since this circular dependency can't be broken given our algorithm, these should be labeled as equivalent.

Slide Lecture 7 -191



The minimal DFA. Note that state F implicitly goes to a trap state on 0.

Slide Lecture 7 -192

**REGULAR LANGUAGES  
AND NONREGULAR LANGUAGES**

Consider again the relation  $I_L$  defined over  $\Sigma^*$ :

For  $x, y \in \Sigma^*$ ,  $xI_Ly$  means that  $x$  and  $y$  are indistinguishable with respect to  $L$ ; for every  $z \in \Sigma^*$ , either  $xz$  and  $yz$  are both in  $L$ , or  $xz$  and  $yz$  are both in the complement of  $L$ .

A language  $L$  in  $\Sigma^*$  is regular if and only if the set of equivalence classes of the relation  $I_L$  is finite.

This should be obvious, since the equivalence classes induced by  $I_L$  correspond to states in a minimal DFA, and the number of states in a DFA is finite.

**Slide Lecture 7 –193**

A language  $L$  in  $\Sigma^*$  is nonregular if and only if there is an infinite subset of  $\Sigma^*$ , any two elements of which are distinguishable with respect to  $L$ .

There is just another way of saying the language cannot be recognized with a Finite State Machine. The number of distinct states is infinitely large: hence to process these languages we need infinite memory.

**Slide Lecture 7 –194**

**THE PUMPING LEMMA FOR NONREGULAR  
LANGUAGES**

Since every regular language is processed by a finite state machine, what happens if we process strings greater than length  $N$ , where  $N$  is the number of states in the machine.

We assume all transitions are fully defined  
(e.g., trap states are explicit).

Answer: we must visit some state twice. In other words, we loop in the finite state machine.

**Slide Lecture 7 –195**

Assume we process the following string

1001011010100...001111101100101101101...0

which is greater than length  $N$ ,  
the number of states in our machine,  $M$ .

Assume this string is accepted by  $M$ .

We must have visited at least one state twice by the time that  
we had processed  $N$  characters.

Mark the points with at which we visited this state the first  
and second time using #.

1001011010#100...0011111#01100101101101...0

**Slide Lecture 7 –196**

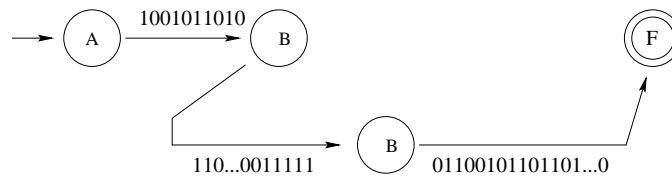
Now consider the regular expression

$1001011010(100\dots0011111)^*01100101101101\dots0$

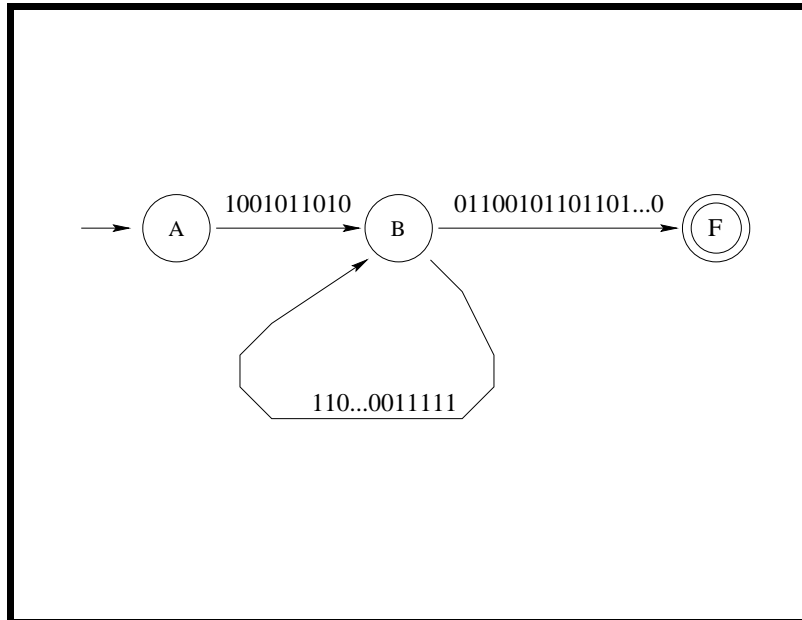
What must true about the strings generated by this regular expression?

All of these strings must be recognized by M.

Slide Lecture 7 -197



Slide Lecture 7 -198



Slide Lecture 7 –199

The Pumping Lemma: Let  $L$  be a regular language recognized by some DFA with  $N$  states. FORALL  $x \in L$  with  $|x| \geq N$ , there EXISTS strings  $u, v$  and  $w$  such that

$$x = uvw$$

$$|v| > 0$$

$$|uv| \leq N$$

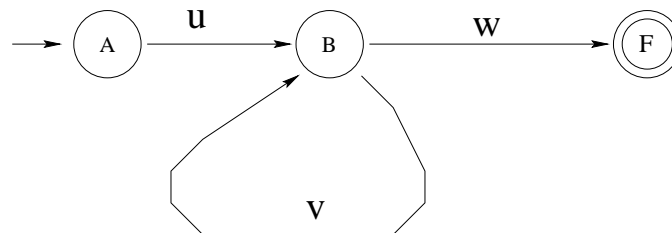
and FORALL  $i \geq 0$ ,  $uv^i w \in L$ .

Slide Lecture 7 –200

The Pumping Lemma sometimes just states that there exists some integer  $N$ ; the Pumping Lemma isn't true for all integers, but it is true for all integers greater than the number of states in the machine.

Also note that the pumping lemma states  
"FORALL  $x \in L$  with  $|x| \geq N$ ,  
there EXISTS strings  $u, v$  and  $w$ "

Slide Lecture 7 -201



Slide Lecture 7 -202

### SO WHAT?

Assume we have some language  $L$  that includes  $x$ .

“FORALL  $x \in L$ ”

We can pick any string as long as we pick  $x$  to be sufficiently long such that if there is DFA that recognizes  $x$ ,  $|x| > N$ .

“There EXISTS strings  $u, v$  and  $w$ ”

If we show that there does not exist  $u, v$  and  $w$  that satisfy the following conditions

$$x = uvw \quad |v| > 0 \quad |uv| \leq N$$

and FORALL  $i \geq 0$ ,  $uv^i w \in L$

then language  $L$  is not regular.

**Slide Lecture 7 –203**

### THE CLASSIC EXAMPLE

Consider  $L = \{0^n 1^n | n \geq 0\}$ .

Assume  $L$  is regular.

Pick  $x = 0^N 1^N$  where  $N$  is the number of states in the corresponding minimal DFA.

Since  $|v| > 0$  and  $|uv| \leq N$

the loop corresponding to  $v$  processes only 0 symbols (i.e.  $v \in 00^*$ ).

Thus, for any  $i$  other than 1,

$$uv^i w \notin 0^n 1^n$$

and this language cannot be regular.

**Slide Lecture 7 –204**

ANOTHER EXAMPLE

$L = \{x \in \{0,1\}^* \mid x \text{ has an equal number of 0's and 1's}\}$ .

Since this language contains  $L = \{0^n 1^n \mid n \geq 0\}$  as a special subset, just pick  $x = 0^N 1^N$  again.

Again, for any  $i$  other than 1,

$$uv^i w \notin L$$

and this language cannot be regular.

In this case, you cannot pick just any string,

Slide Lecture 7 –205

If you pick some member of  $(01)^*$ , which is in  $L$ , then you can have  $v = 01$  or  $0101$ , etc., and

$$uv^i w \in L$$

**But the Pumping Lemma holds FORALL strings.**

So we must be careful when picking string  $x$ .

**We cannot use the Pumping Lemma to show that languages are regular.**

Slide Lecture 7 –206

OTHER EXAMPLES

Consider  $L = \{0^k 1^j \mid k < j\}$ .

Consider string  $x = 0^N 1^{N+1}$ .

Since  $|v| > 0$  and  $|uv| \leq N$  we know  $v \in 00^*$ .

Therefore FORALL  $i > 2, uv^i w \notin L$ .

Slide Lecture 7 -207

Consider  $L = \{0^k 1^j \mid j < k\}$ .

We have to be careful here.

Not just any  $x$  will do.

Consider string  $x = 0^{N+1} 1^N \in L$ .

Now consider the case where  $i = 0$ .

Slide Lecture 7 -208

Consider  $L = \{1^p \mid p \text{ is prime}\}$ .

Let  $x = uvw$  be any prime number where  $|x| > N$ .

Then  $|x| = |u| + |v| + |w| = |v| + |uw|$ .

Let  $k = |v|$  and  $q = |uw|$ ; then  $|uv^i w| = q + ik$ .

Consider  $uv^i w$  when  $i = q + 2k + 2$

$$|uv^i w| = q + ik \quad (1)$$

$$= q + (q + 2k + 2)k \quad (2)$$

$$= q + kq + 2k^2 + 2k \quad (3)$$

$$= q(1 + k) + 2k(k + 1) \quad (4)$$

$$= (q + 2k)(1 + k) \quad (5)$$

which is not prime.

**Slide Lecture 7 –209**

Show that  $L = 0^{n^2}$  is not regular.

Let  $x = 0^{N^2}$

Consider  $uv^2w$

We know that  $|uv^2w| = N^2 + |v|$

Since  $|v| > 0$  and  $|uv| \leq N$

we know  $N^2 + 1 \leq |uv^2w| \leq N^2 + N$ .

Since  $|uv^2w| < (N + 1)^2$

the string  $uv^2w \notin L$  and

$L = 0^{n^2}$  is not regular.

**Slide Lecture 7 –210**

## **DECISION PROBLEMS**

“Decision problems” have “yes” and “no” answers.  
We have often looked at FSAs as decision machines.  
Other theory (e.g., NP-Completeness)  
is based on decision problems.

**Slide Lecture 7 –211**

A decision problem related to FSAs.

Given a FSA  $M$ , is  $L(M)$  finite? yes or no.

The pumping lemma is useful here.

If  $N$  is the number of states in the machine, and  $M$  accepts some string of length  $N$  or greater, then  $L(M)$  is infinite. If  $M$  contains a loop,  $L(M)$  is infinite.

**Slide Lecture 7 –212**

If M does not accept any string  $x$  such that  $N \leq |x| \leq 2N$  then  
L is finite and strictly contains strings of length  $< N$   
(i.e., M does not loop).

**Slide Lecture 7 –213**

Proof by contradiction:

Assume there are no strings where  $N \leq |x| \leq 2N$ .

Pick string  $z$  to be the shortest string  
such that  $|z| \geq 2N$ .

By the pumping lemma,  $z = uvw$ ,  
where  $|v| > 0$  and  $|uv| \leq N$ .

Now when  $i = 0$ ,  $uv^i w = uw \in L$ .

But since  $|uvw| \geq 2N$  and  $|v| \leq N$ ,  
we know  $|uw| \geq 2N - N$ ,

so that string  $uw$  also contains a loop.

So either  $N \leq |uw| < 2N$  (which is ruled out by our test)

or  $|uw| \geq 2N$  which means  $z$  isn't minimal.

A Contradiction.

**Slide Lecture 7 –214**

SOME TEASERS:

Programming Languages are not regular:  
Consider  $((((x)))) = ({}^n x)^n$  in some language.

Or consider

```
BEGIN
  BEGIN
    END
  END
END
```

This is similar to  $L = 0^n 1^n$ .

The pumping lemma holds here.

Slide Lecture 7 -215

Finite State Machines cannot “parse” programming languages.

Finite State Machines are good for recognizing words or tokens:

Lexical Analysis

```
begin
  if
  then
  else
chocolate
```

Slide Lecture 7 -216

All computers are just BIG Finite State Machines:  
Consider all the possible BITS in our machine.  
Memory + ALU + Buses + EVERYTHING.  
Our computer has a (REALLY LARGE) finite number of states  
 $2^{BITS}$   
For sufficiently large  $n$ , we can't recognize  $L = 0^n 1^n$  without  
running out of memory.

Slide Lecture 7 -217

Also note our computer is nondeterministic: we can change  
states without scanning input.  
We can only really have infinite  $\lambda$  loops  
(otherwise we run out of memory again).  
Thus, a deterministic model of our computer  
might have  $2^{2^{BITS}}$

Slide Lecture 7 -218

Now wait, assume we wanted to “enumerate”,  
say for example,  $2^{400}$  states.

If all the atom in the universe were computers working in  
parallel at picosecond rates since the big bang, we still would  
have enumerated only a fraction of the  $2^{400}$  states.

So while all computers are just big finite state machines,  
this isn't a very useful model.

So we turn to models with infinite memory.