15-453 - Formal Languages, Automata, and Computability
Spring 2015, Tuesday/Thursday 12:00-1:20 PM, Baker Hall 136A

First class I taught at CMU.

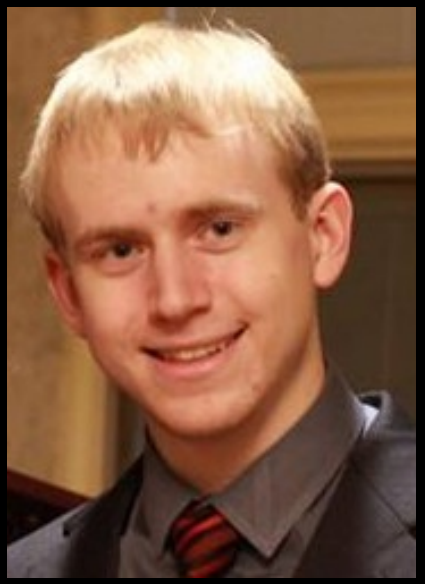
This class will be a small variation of the Of Lenore Blum's Spring 2014. (quizes)

INSTRUCTORS & TAs

Steven Rudich

Asa Frank

Owen Kahn



Andy Smith

Office Hours

Prof. Rudich: Tuesday, 1:30-2:30 PM
GHC 7219

Asa: Monday, 6:30-8:00 PM

Owen: Wednesday, 7:00-9:00 PM

Andrew: TBD

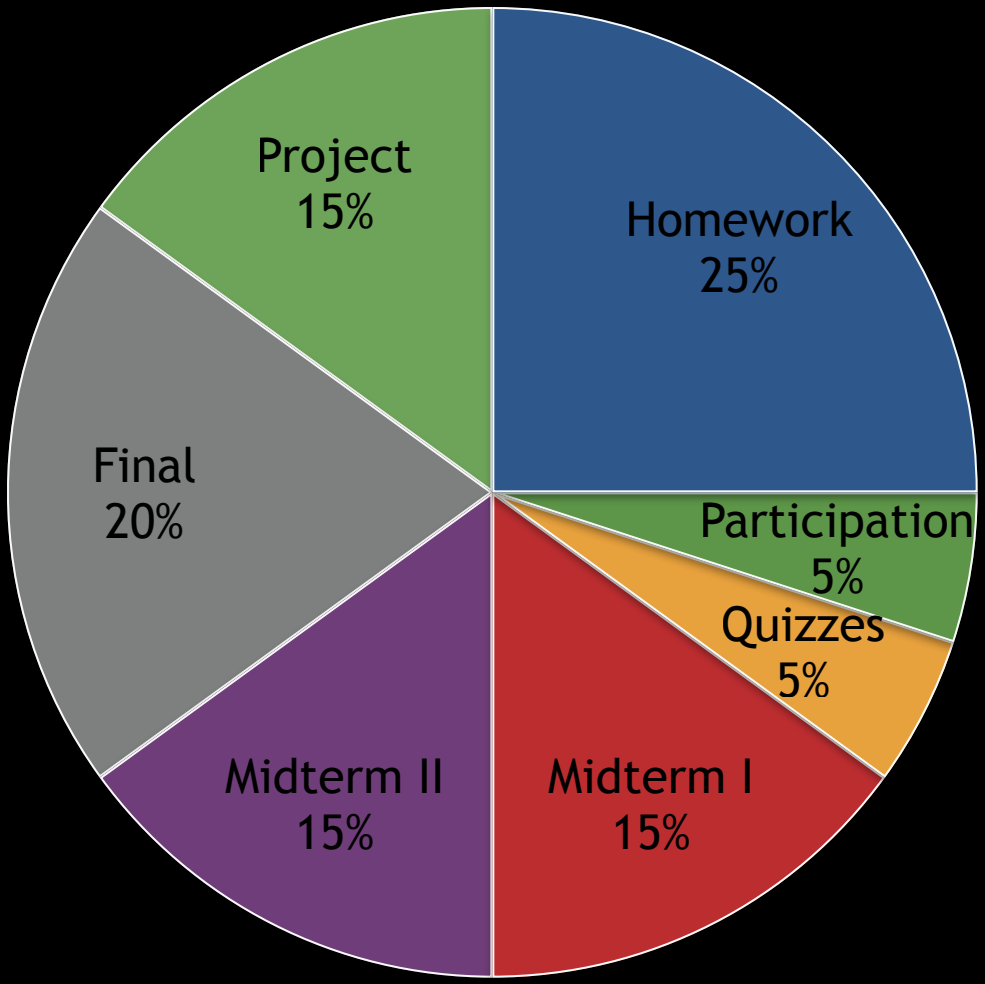
(Locations TBD)

http://www.contrib.andrew.cmu.edu/~okahn/flac-s15/index.html

15-453 - Formal Languages, Automata, and Computability

Spring 2015, Tuesday/Thursday 12:00-1:20 PM, Baker Hall 136A

Date	Lecture	Reading	Assignment
Tue Jan 13	Overview Deterministic Finite Automata and Regular Languages	Chapters 0, 1.1	
Thu Jan 15	Non-determinism and Regular Operations	Chapter 1.2	HW 1 out
Tue Jan 20	Regular Expressions and the Pumping Lemma for Regular Languages	Chapter 1.3, 1.4	
Thu Jan 22	Minimizing DFAs	Finish Chapter 1	HW 1 due HW 2 out
Tue Jan 27	Push-Down Automata and Context-Free Grammars; Pumping Lemma for CFLs	Chapter 2.1, 2.2, 2.3	
Thu Jan 29	Equivalence of PDAs and CFGs		HW 2 due HW 3 out
Tue Feb 3	Chomsky Normal Form, Turing Machines	Chapter 2, Chapter 3	
Thu Feb 5	Undecidability	Chapter 3, Chapter 4	**Project Report 1 due** HW 3 due
Tue Feb 10	Review		
Thu Feb 12	**Midterm 1**		HW 4 out


Project
15%
Homework
25%
Final
20%
Participation
5%
Quizzes
5%
Midterm II
15%
Midterm I
15%

HOMEWORK

Homework will be assigned every Thursday and will be due one week later at the beginning of class. Late homework will be accepted only under exceptional circumstances.

All assignments must be typeset (exceptions allowed for diagrams). Each problem should be done on a separate page.

HOMEWORK

Homework will be assigned every Thursday and will be due one week later at the beginning of class. Late homework will be accepted only under exceptional circumstances.

All assignments must be typeset (exceptions allowed for diagrams). Each problem should be done on a separate page.

You must list your collaborators (including yourself) and all references in every homework assignment in a References section at the end.

COURSE **PROJECT**

Choose a (unique) topic

Learn about your topic

Write progress reports
(Feb 5, March 24)

Meet with an instructor/TA once a month

Prepare an 8-minute presentation
(April 21-30)

Final Report (April 30)

COURSE **PROJECT**

Suggested places to look for project topics

Any paper that has appeared in the proceedings of FOCS or STOC in the last 5 years. FOCS (Foundations of Computer Science) and STOC (Symposium on the Theory of Computing) are the two major conferences of general computer science theory. The proceedings of both conferences are available at the E&S library or electronically.

- **Electronic version of the proceedings of STOC**
- **Electronic version of the proceedings of FOCS**

What's New

This class is about mathematical models of computation

Course Outline

PART 1

Automata and Languages:

finite automata, regular languages, pushdown automata, context-free languages, pumping lemmas.

PART 2

Computability Theory:

Turing Machines, decidability, reducibility, the arithmetic hierarchy, the recursion theorem, the Post correspondence problem.

PART 3

Complexity Theory and Applications:

time complexity, classes P and NP, NP-completeness, space complexity, PSPACE, PSPACE-completeness, the polynomial hierarchy, randomized complexity, classes RP and BPP.

Mathematical Models of Computation
(predated computers as we know them)

PART 1

1940's-50's (neurophysiology, linguistics)

Automata and Languages:
finite automata, regular languages, pushdown automata, context-free languages, pumping lemmas.

PART 2

Computability Theory: 1930's-40's (logic, decidability)
Turing Machines, decidability, reducibility, the arithmetic hierarchy, the recursion theorem, the Post correspondence problem.

PART 3

1960's-70's (computers)

Complexity Theory and Applications:
time complexity, classes P and NP, NP-completeness, space complexity, PSPACE, PSPACE-completeness, the polynomial hierarchy, randomized complexity, classes RP and BPP.

This class will emphasize **PROOFS**

A good proof should be:

Easy to understand

Correct

Suppose $A \subseteq \{1, 2, \ldots, 2n\}$ with $|A| = n+1$

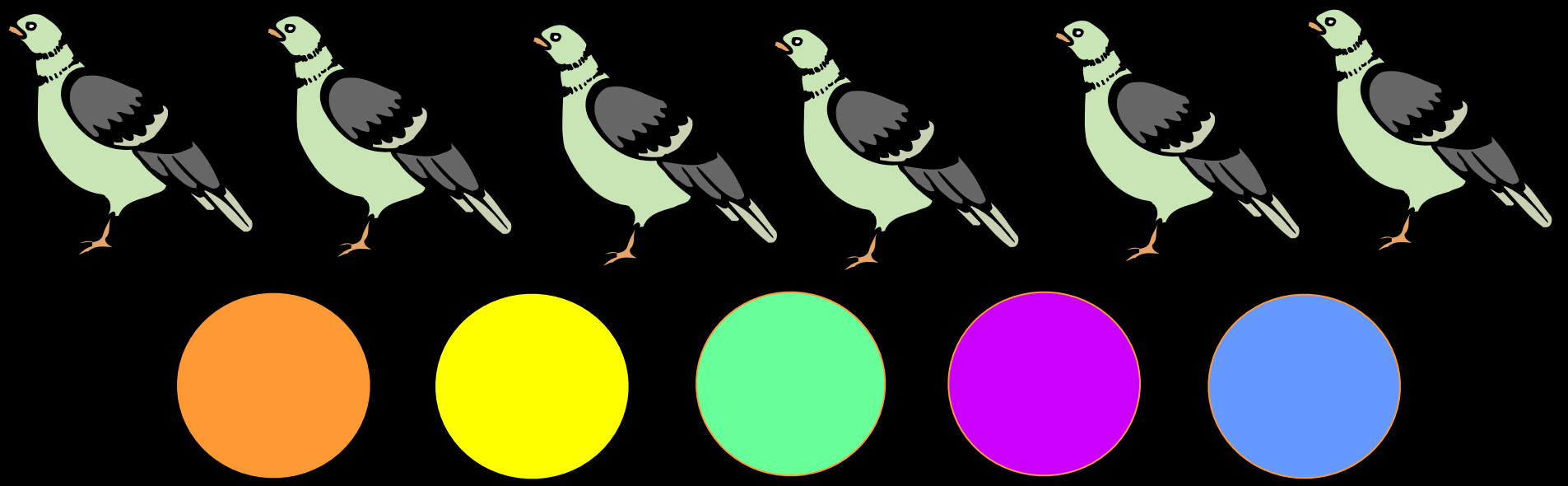
TRUE or FALSE:
There are always two numbers in A such that one divides the other

TRUE

HINT 1:

THE PIGEONHOLE PRINCIPLE

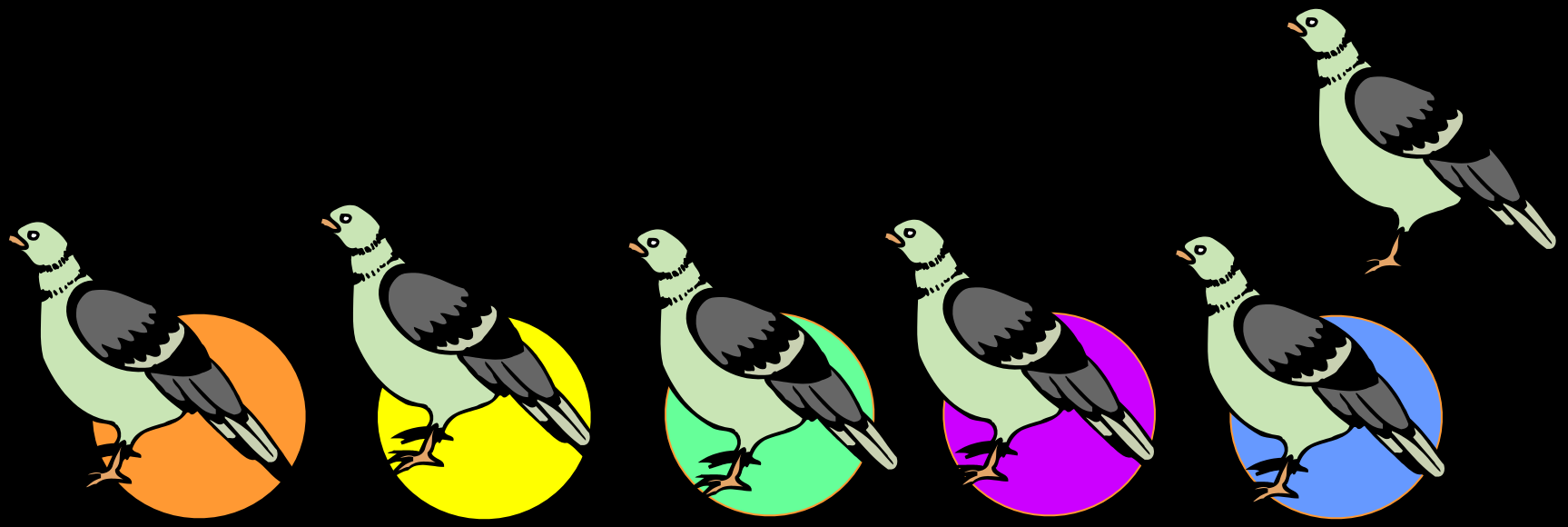
If you put 6 pigeons in 5 holes then at least one hole will have more than one pigeon



HINT 1:

THE PIGEONHOLE PRINCIPLE

If you put 6 pigeons in 5 holes then at least one hole will have more than one pigeon



HINT 1:

THE PIGEONHOLE PRINCIPLE

If you put $n+1$ pigeons in n holes
then at least one hole will have
more than one pigeon

HINT 2:

Every integer a can be written as
$a = 2^k m$, where m is an odd number

PROOF IDEA:

Given: $A \subseteq \{1, 2, \ldots, 2n\}$ and $|A| = n+1$

Show: Use PHP to prove There is an integer m and elements $a_1 = a_2$ in A

such that $a_1 = 2^i m$ and $a_2 = 2^j m$

PROOF:

Suppose $A \subseteq \{1, 2, \ldots, 2n\}$ with $|A| = n+1$

Write every number in A as $a = 2^k m$, where m is an odd number between 1 and 2n-1

How many odd numbers in $\{1, \ldots, 2n\}$? n

Since $|A| = n+1$, there must be two numbers in A with the same odd part

Say a_1 and a_2 have the same odd part m. Then $a_1 = 2^i m$ and $a_2 = 2^j m$, so one must divide the other

PROOF:

Suppose $A \subseteq \{1, 2, \ldots, 2n\}$ with $|A| = n+1$

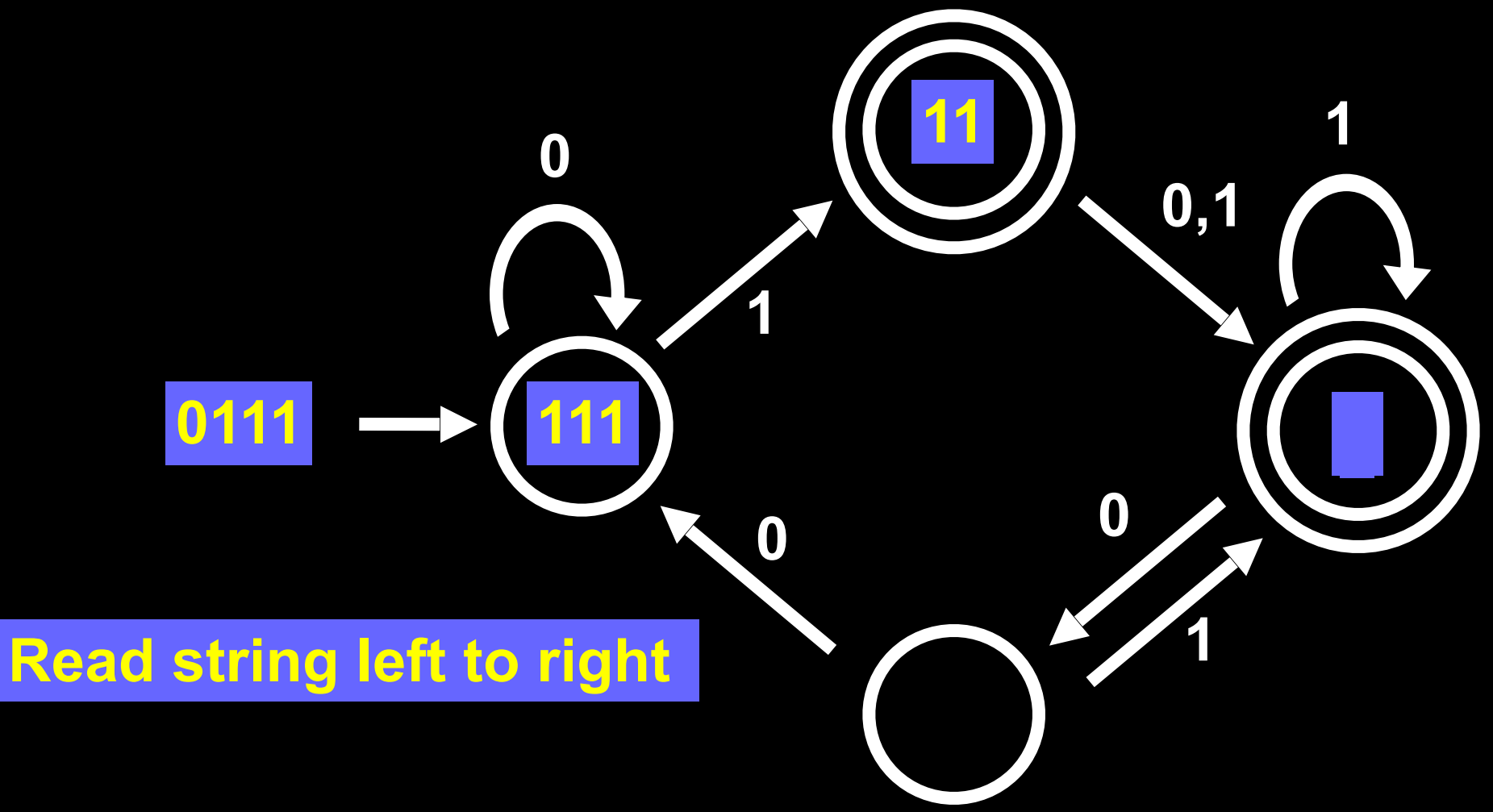
Write every number in A as $a = 2^k m$, where m is an odd number between 1 and 2n-1

Put pigeon a in hole m <= 2n-1

n holes = odd numbers in {1, 3…, 2n-1}

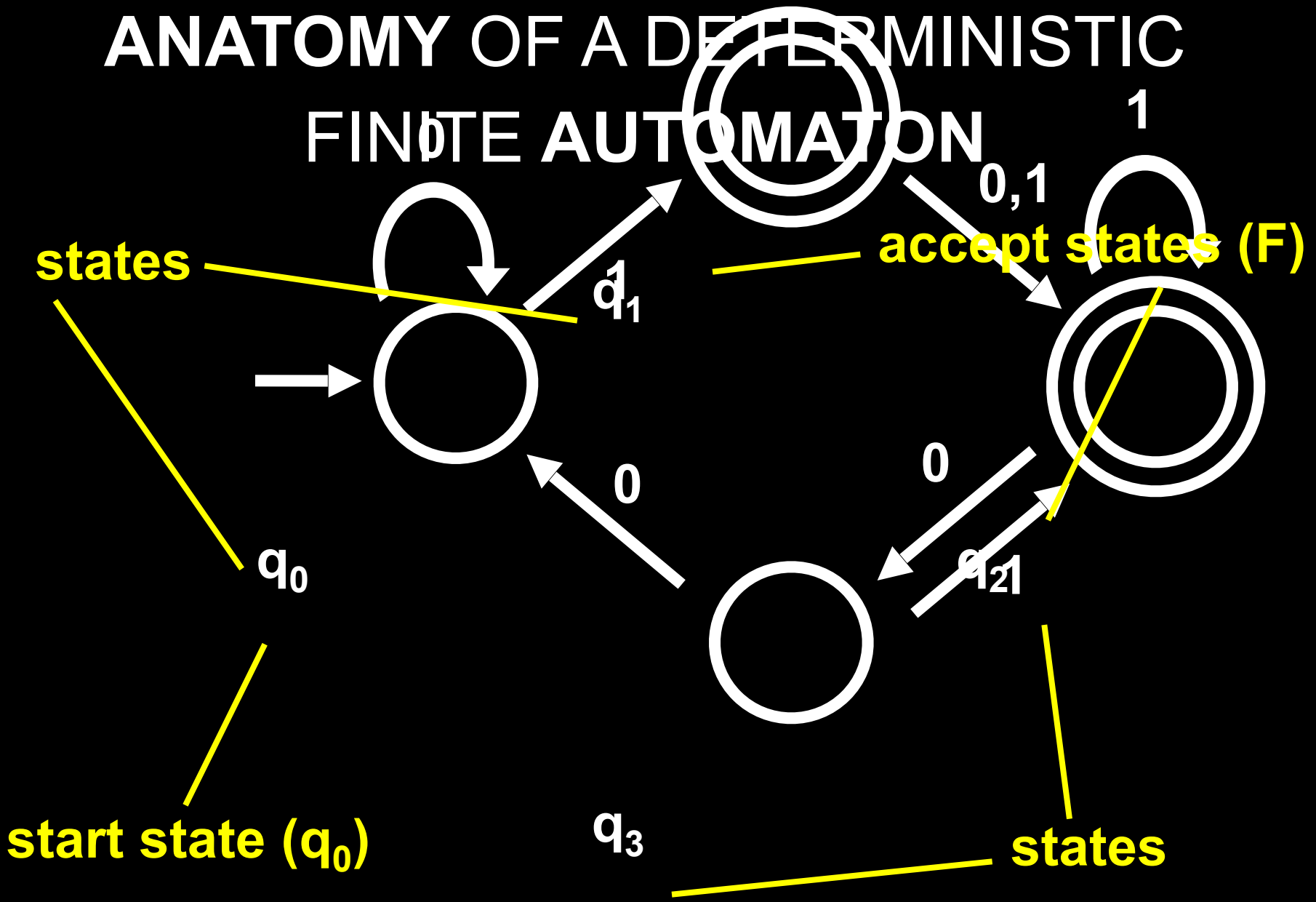
There exists a hole m with 2 pigeons $T2^i m$ and $2^j m$, so one must divide the other

DETERMINISTIC FINITE AUTOMATA
and
REGULAR LANGUAGES

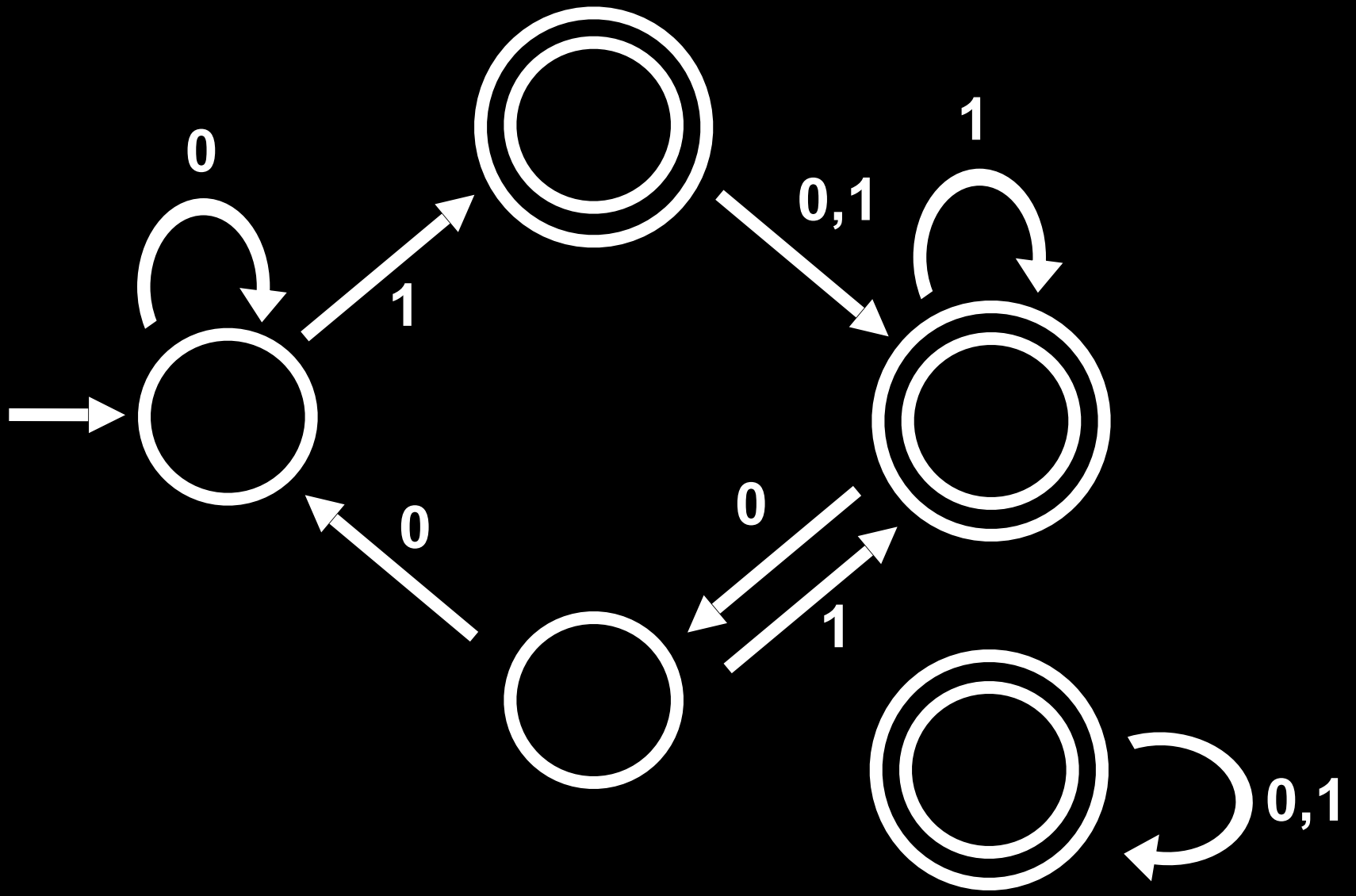


The automaton **accepts** a string if the process ends in a double circle

ANATOMY OF A DETERMINISTIC FINITE **AUTOMATON**



ANATOMY OF A DETERMINISTIC FINITE **AUTOMATON**



SOME VOCABULARY

An **alphabet Σ** is a finite set (e.g., Σ = {0,1})

A **string over Σ** is a finite-length sequence of elements of Σ

Σ^* = the set of strings over Σ

For string **x**, **|x|** is the **length** of **x**

The unique string of **length 0** will be denoted by **ε** and will be called the **empty** or **null string**

A **language over Σ** is a set of strings over Σ
In other words: a language is a subset of **Σ^***

deterministic DFA

A ^ finite automaton ^ is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

Q is the set of states (finite)

Σ is the alphabet (finite)

$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept/final states

Suppose $w_1, \ldots, w_n \in \Sigma$ and $w = w_1 \ldots w_n \in \Sigma^*$
Then **M accepts w** iff there are $r_0, r_1, \ldots, r_n \in Q$, s.t.

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$, and
- $r_n \in F$

deterministic DFA

A ^ finite automaton ^ is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

Q is the set of states (finite)

Σ is the alphabet (finite)

$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept/final states

M accepts ε iff $q_0 \in F$

deterministic DFA

A ^ finite automaton ^ is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

Q is the set of states (finite)

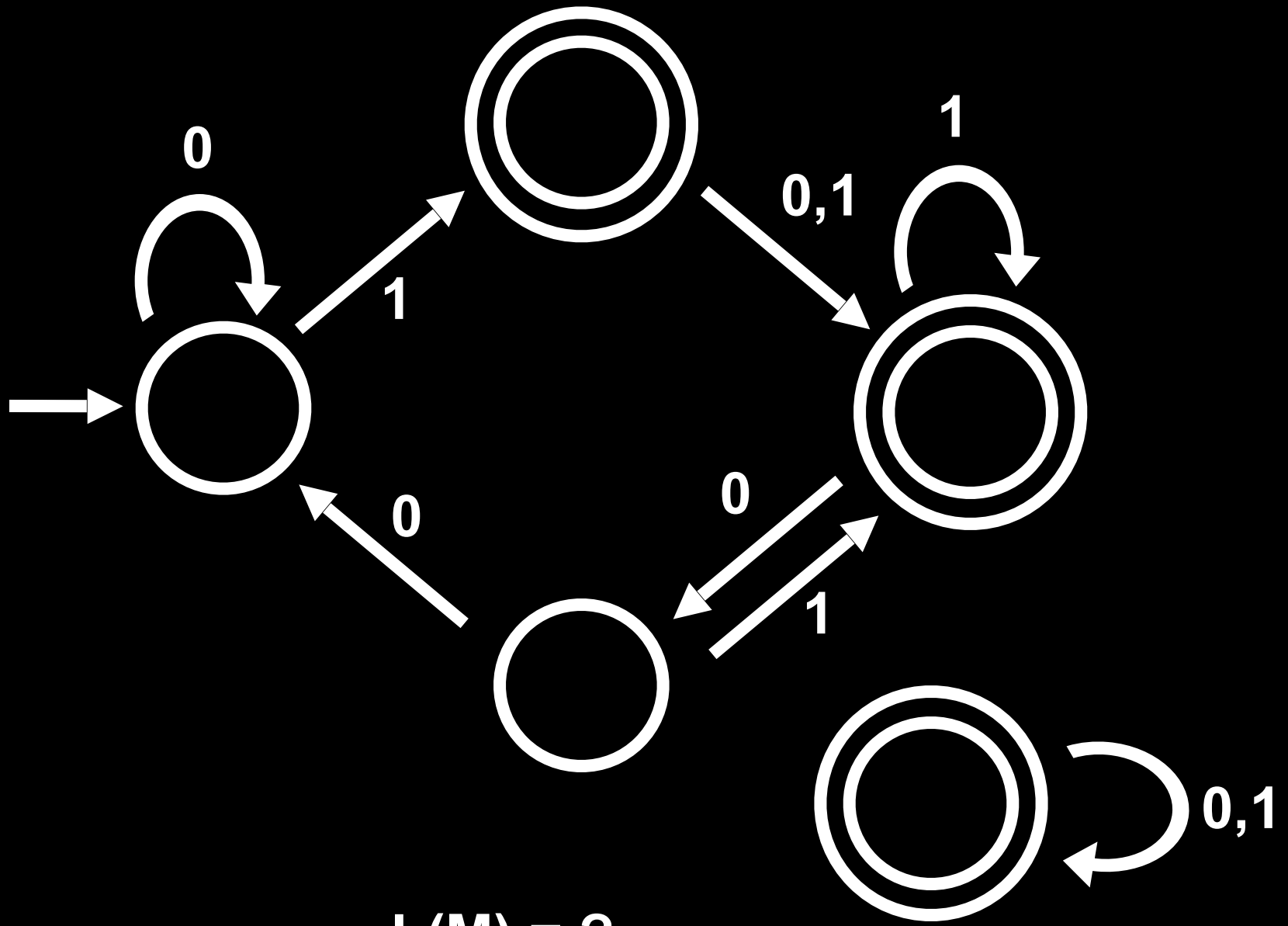
Σ is the alphabet (finite)

$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

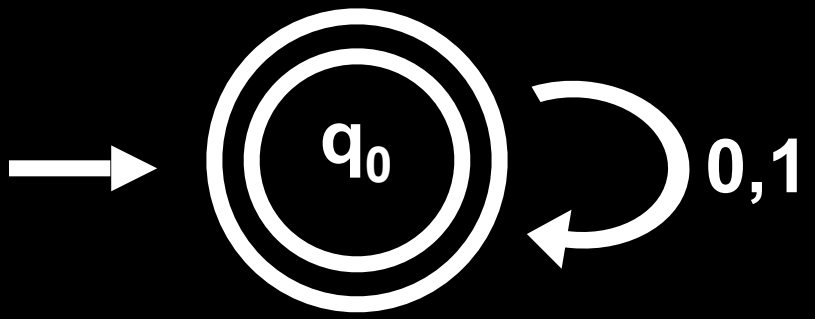
$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept/final states

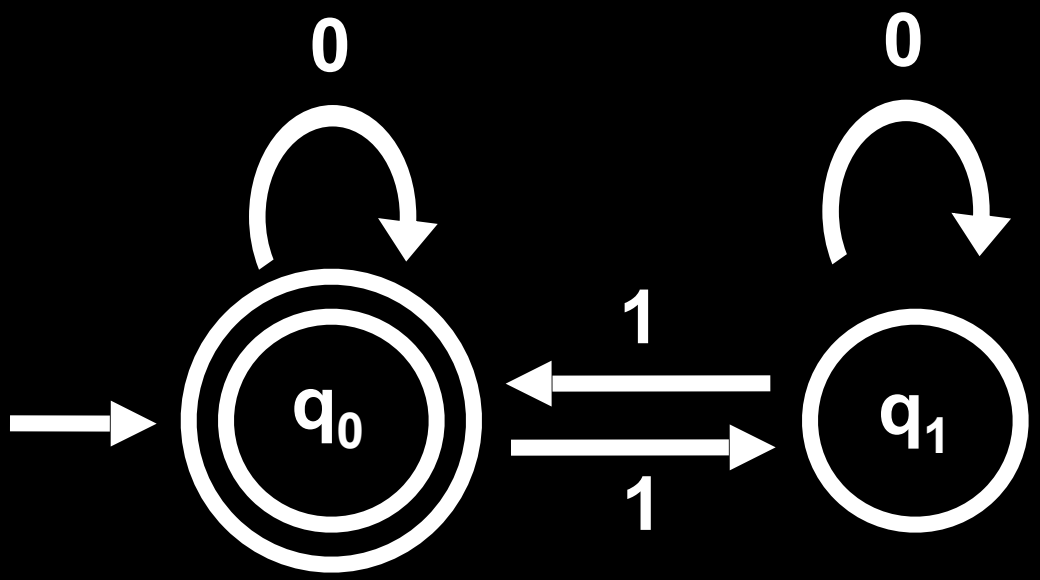
L(M) = set of all strings that M accepts
= "the language recognized by M"



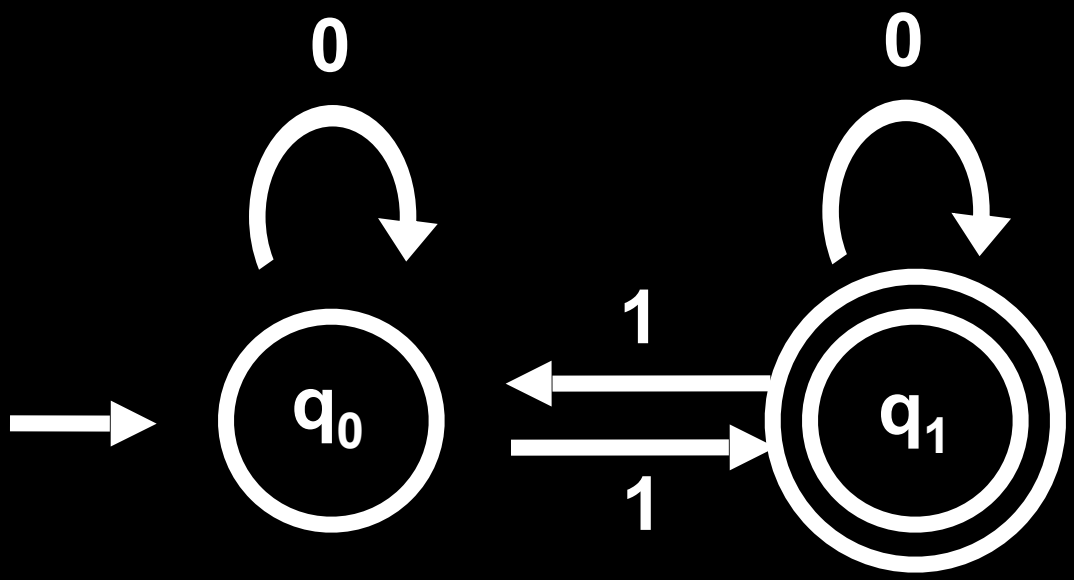
L(M) = ?



$L(M) = \{0,1\}^*$



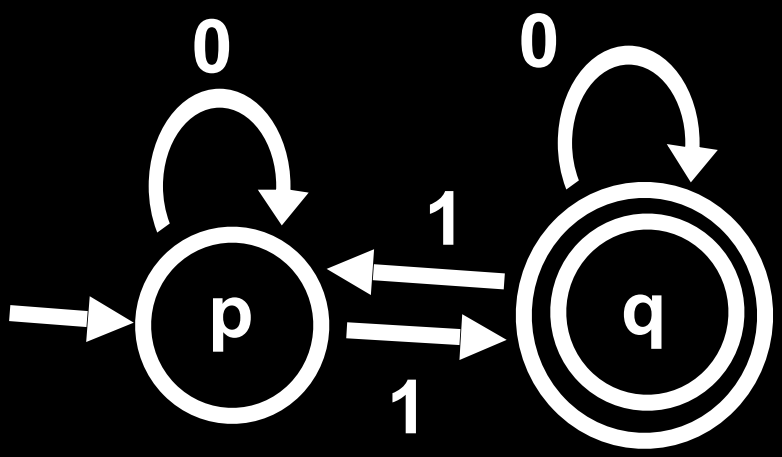
L(M) = { w | w has an even number of 1s}



L(M) = { w | w has an odd number of 1s}

Q Σ q_0 F

M = ({p,q}, {0,1}, δ, p, {q})



δ	0	1
p	p	q
q	q	p

THEOREM:

L(M) = {w | w has odd number of 1s }

Proof: By induction on **n**, the length of a string.

Base Case: n=0: ε ∉ RHS and ε ∉ L(M). Why?

Induction Hypothesis: Suppose for all **w** ∈ **Σ***, **|w| = n**, **w** ∈ L(M) iff **w** has odd number of 1s.

Induction step: Any string of length n+1 has the form w0 or w1.

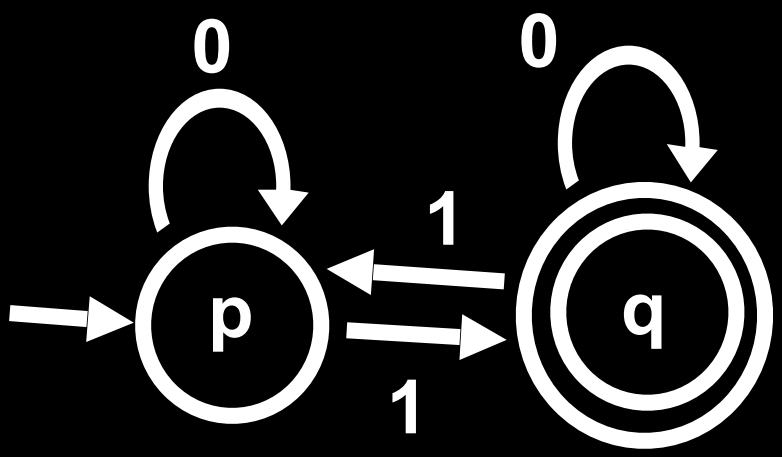
Now w0 has an odd # of 1's ⇔ w has an odd # of 1's⇔

M is in state q after reading w (why?) ⇔

M is in state q after reading w0 (why?) ⇔w0 ∈ L(M)

Q Σ q_0 F

M = ({p,q}, {0,1}, δ, p, {q})



δ	0	1
p	p	q
q	q	p

THEOREM:

L(M) = {w | w has odd number of 1s }

Proof: By induction on **n**, the length of a string.

Base Case: n=0: ε ∉ RHS and ε ∉ L(M). Why?

Induction Hypothesis: Suppose for all **w** ∈ **Σ***, **|w| = n**, **w** ∈ L(M) iff **w** has odd number of 1s.

Induction step: Any string of length n+1 has the form w0 or w1.

Now w1 has an odd # of 1's ⇔ w has an even # of 1's⇔

M is in state p after reading w (why?) ⇔

M is in state q after reading w1 (why?) ⇔w1 ∈ L(M) QED

Invariant Condition

If M in state p, M has read a W with an even number of 1s.
If M in state q, M has read a W with an odd number of 1s.

Base Case, Invariant is true at start:
Initially, M has seen 0 1s, and starts in state p.

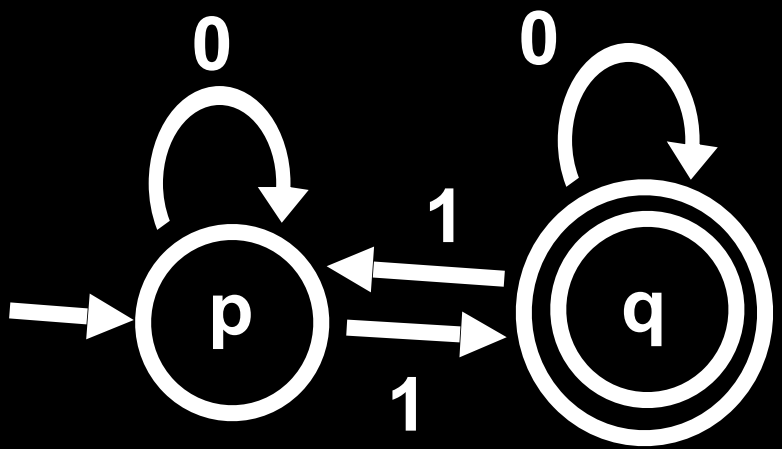
Inductive step:
M has read W with even number of 1s and is in p.
OR
M has read W with odd number of 1s and is in q.

Next M sees 0, remains in same state, maintaining parity.
OR
M sees 1, changing state, maintaining parity invariant.

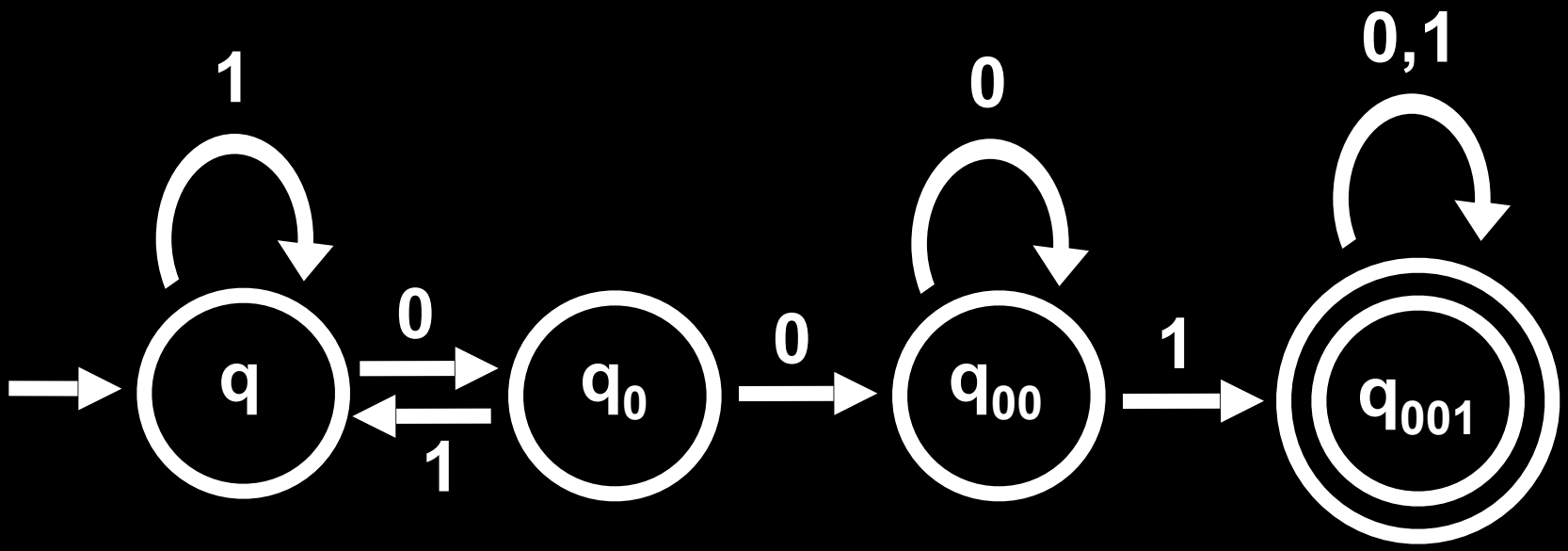
Thus, the invariant condition is always true.

What the little machine is thinking:

If I am in p, I have seen an even number of 1s
If I am in q, I have seen an odd number of 1s

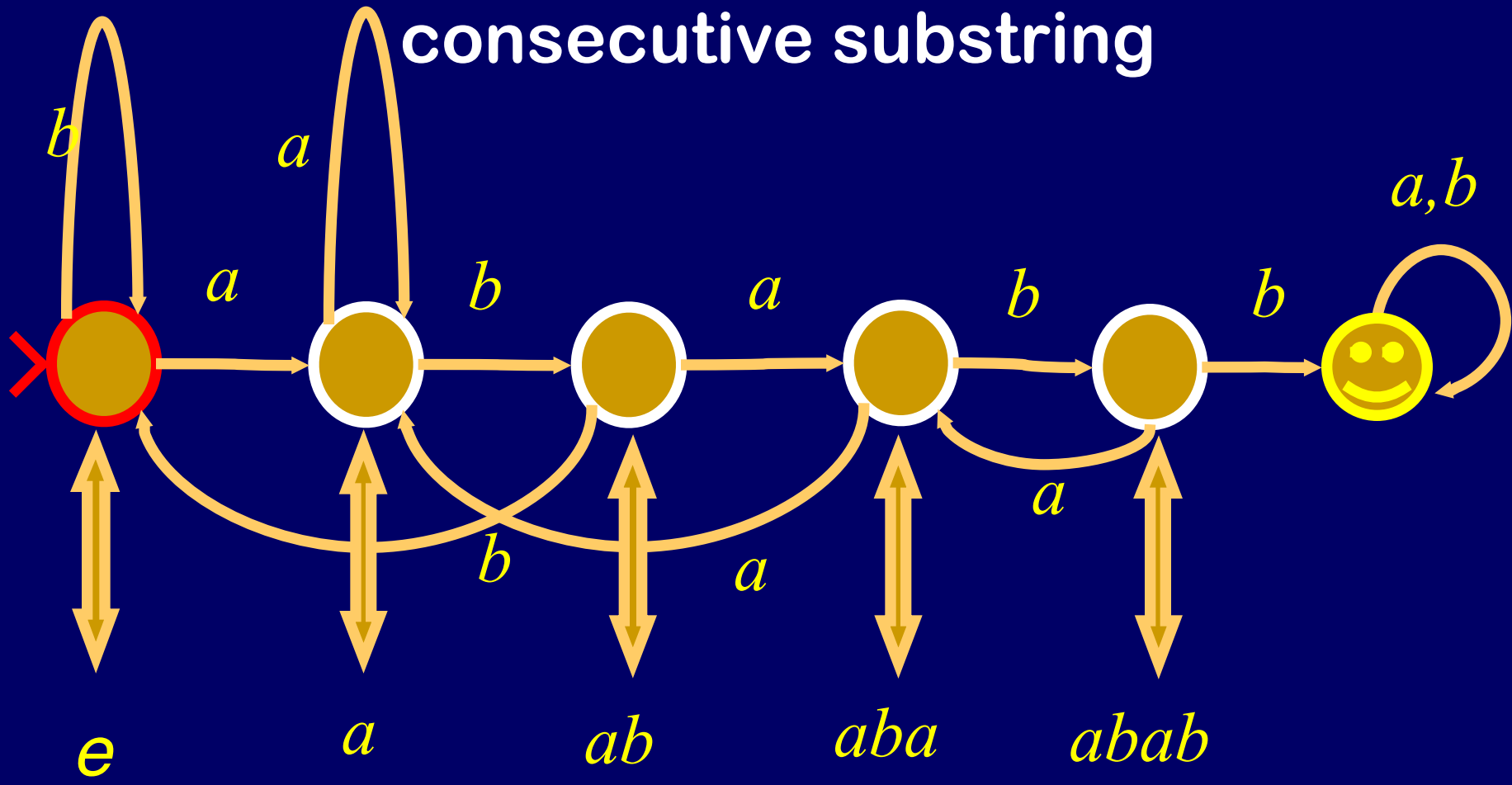


Build a DFA that accepts all and only those strings that contain 001



PROVE

$L =$ all strings containing $ababb$ as a consecutive substring



Invariant: I am state **s** exactly when **s** is the longest suffix of the input (so far) that forms a prefix of ababb

DEFINITION: A language L is regular if it is recognized by a DFA, i.e. if there is a DFA M s.t. L = L(M).

L = { w | w contains 001} is regular

L = { w | w has an even number of 1s} is regular

L = { w | w has an odd number of 1s} is regular

UNION **THEOREM**

Given two languages, L_1 and L_2, define the union of L_1 and L_2 as

$$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$$

Theorem: The union of two regular languages is also a regular language

Theorem: The union of two regular languages is also a regular language

Proof: Let

$M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ be finite automaton for L_1

and

$M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ be finite automaton for L_2

We want to construct a finite automaton

$M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $L = L_1 \cup L_2$

Idea: Run both M_1 and M_2 at the same time!

Q = pairs of states, one from M_1 and one from M_2

$= \{ (q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2 \}$

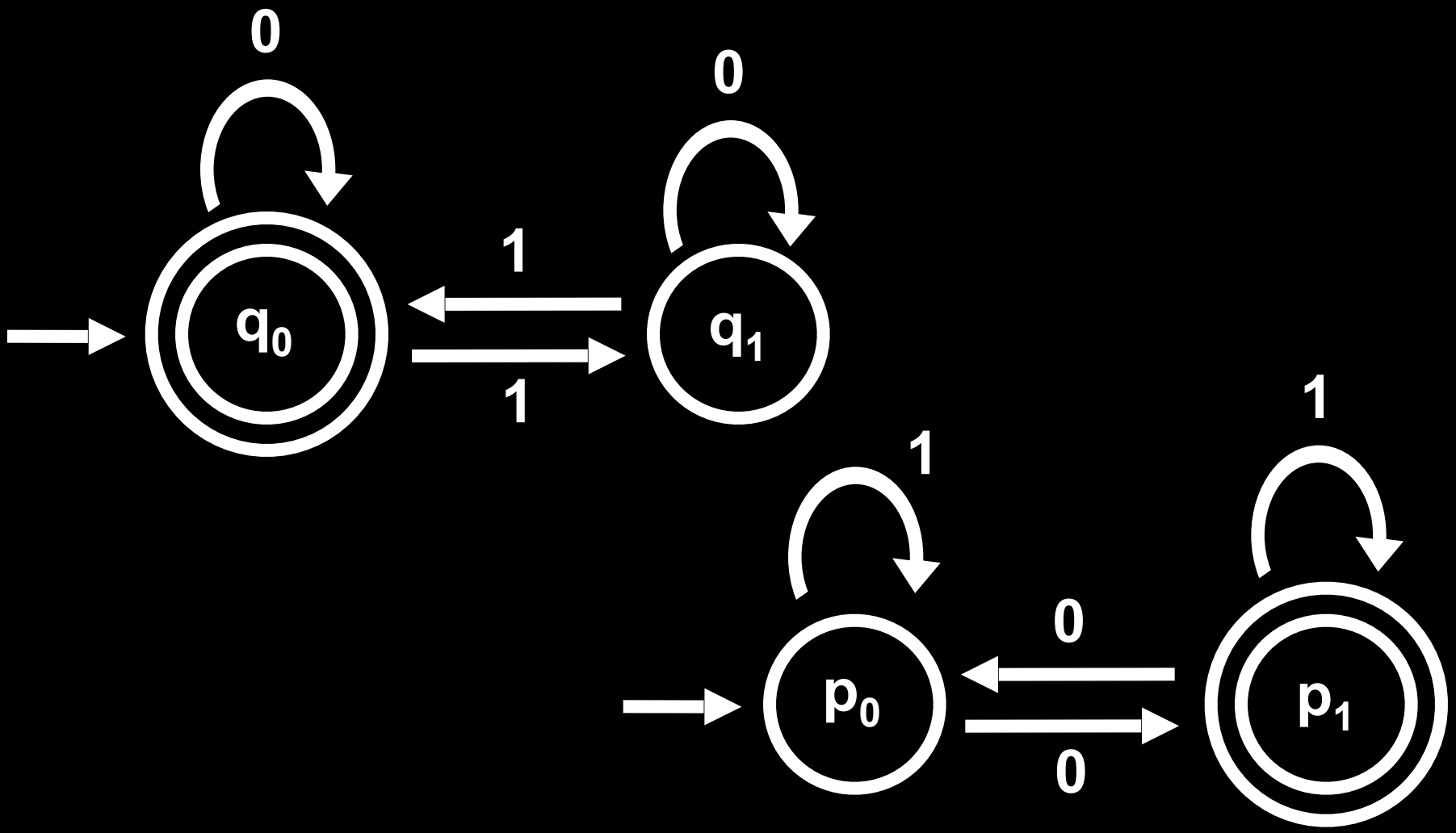
$= Q_1 \times Q_2$

$q_0 = (q_0^1, q_0^2)$

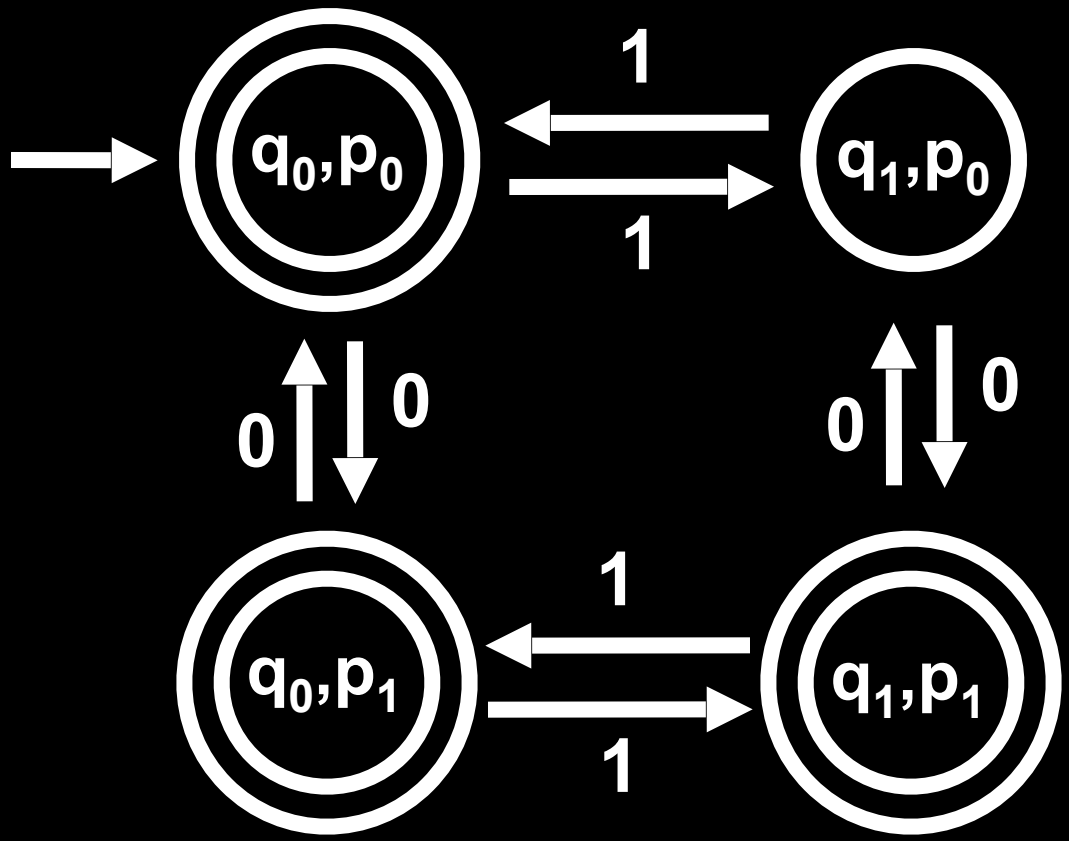
$F = \{ (q_1, q_2) \mid q_1 \in F_1 \text{ or } q_2 \in F_2 \}$

$\delta((q_1,q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$

Theorem: The union of two regular languages is also a regular language



INTERSECTION?



Intersection **THEOREM**

Given two languages, L_1 and L_2, define the intersection of L_1 and L_2 as

$$L_1 \cap L_2 = \{ w \mid w \in L_1 \text{ and } w \in L_2 \}$$

Theorem: The intersection of two regular languages is also a regular language