15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

YOU NEED TO PICK UP
THE SYLLABUS,
THE COURSE SCHEDULE,
THE PROJECT INFO SHEET,
TODAY'S CLASS NOTES

INSTRUCTORS & TAs

Lenore Blum

Aashish Jindia

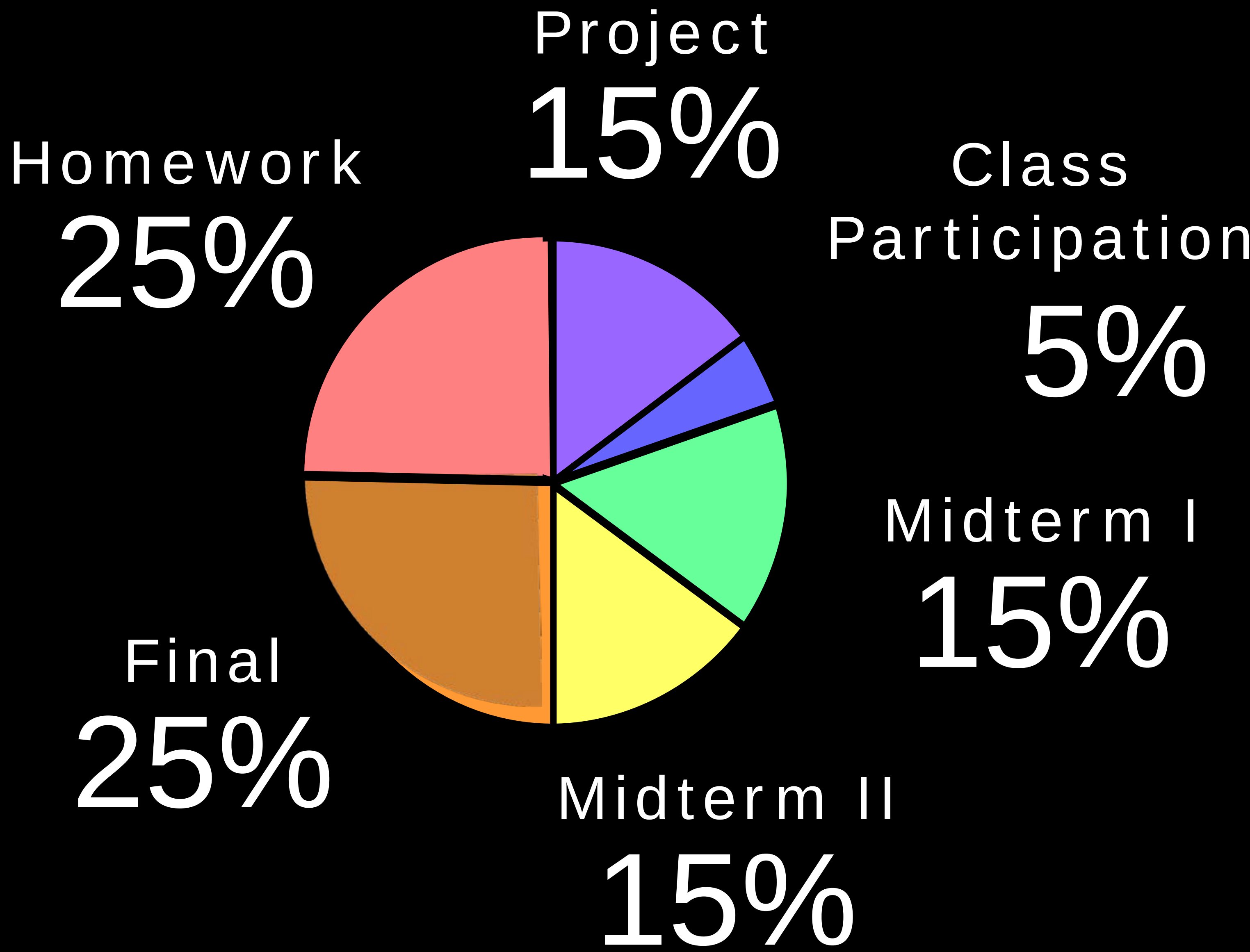
Andy Smith

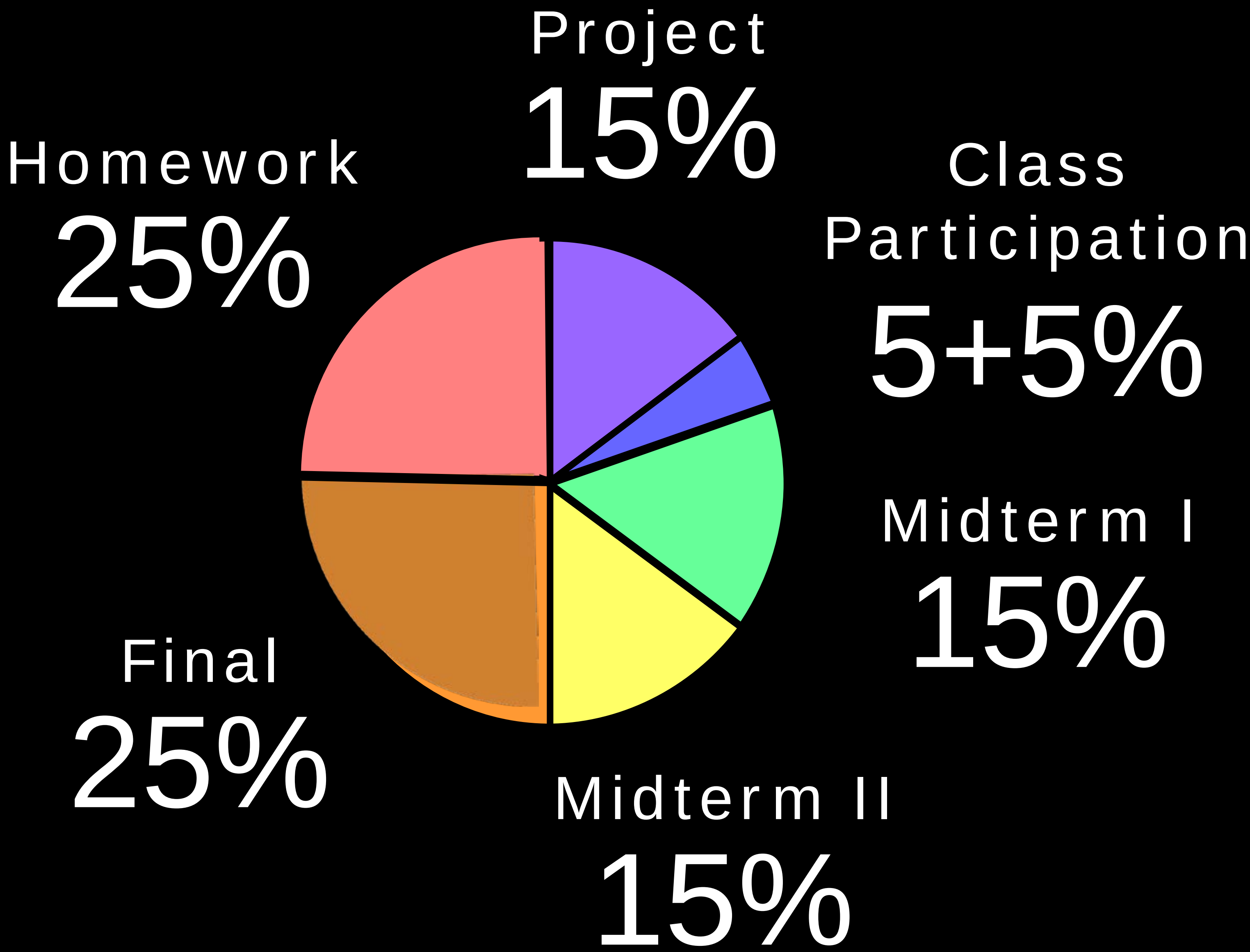
Office Hours

Lenore Blum (lblum@cs)
Office Hours: Tues 2-3pm, Gates 7105

Andy Smith (adsmith@andrew)
Office Hours: **Mon 6-8pm, GHC 7101**

Aashish Jindia (ajindia@andrew)
Office Hours: **Wed 6-8pm, GHC 7101**


Project
15%
Homework
25%
Class
Participation
5%
Midterm I
15%
Final
25%
Midterm II
15%


Project
15%
Homework
25%
Class
Participation
5+5%
Midterm I
15%
Final
25%
Midterm II
15%

HOMEWORK

Homework will be assigned every Tuesday 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 (including books, articles, websites, people) *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 6, March 22)

Meet with an instructor/TA once a month

Prepare a 10-minute presentation
(April 24, April 29, May 1?)

Final Report (May 1)

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

WHY SHOULD I CARE?

WAYS OF THINKING

THEORY CAN DRIVE PRACTICE

Mathematical models of computation predated computers as we know them

THIS STUFF IS USEFUL

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

Automata and Languages: 1940's-50's (neurophysiology, linguistics)

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

Complexity Theory and Applications: 1960's-70's (computers)

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

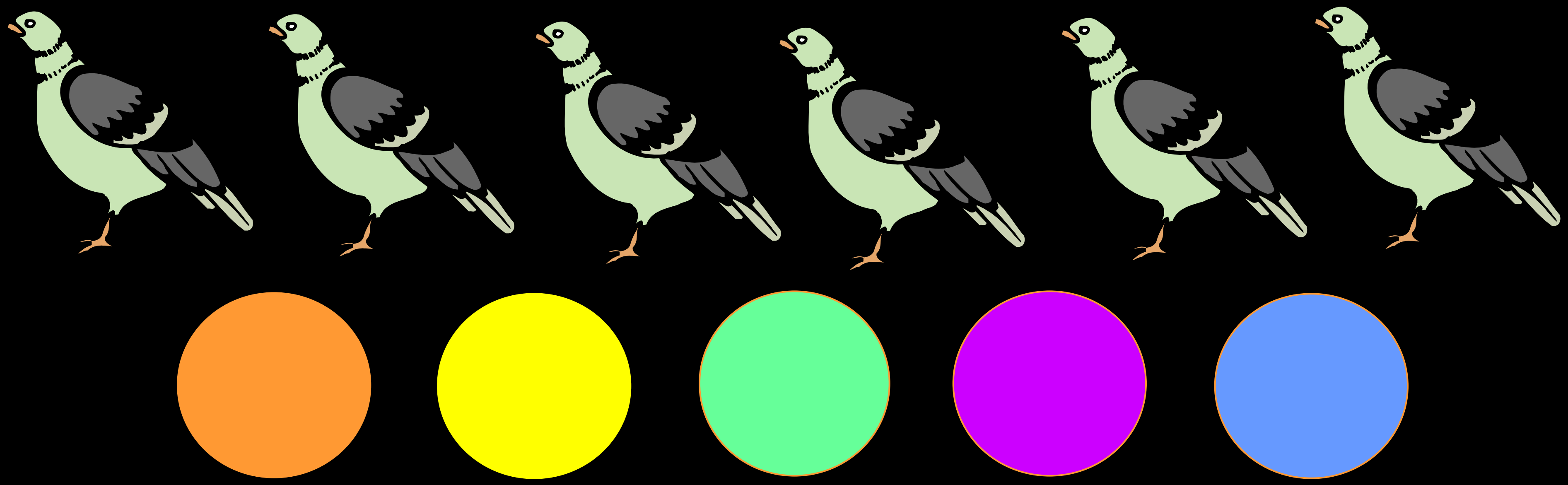
TRUE

LEVEL 1

HINT 1:

THE PIGEONHOLE PRINCIPLE

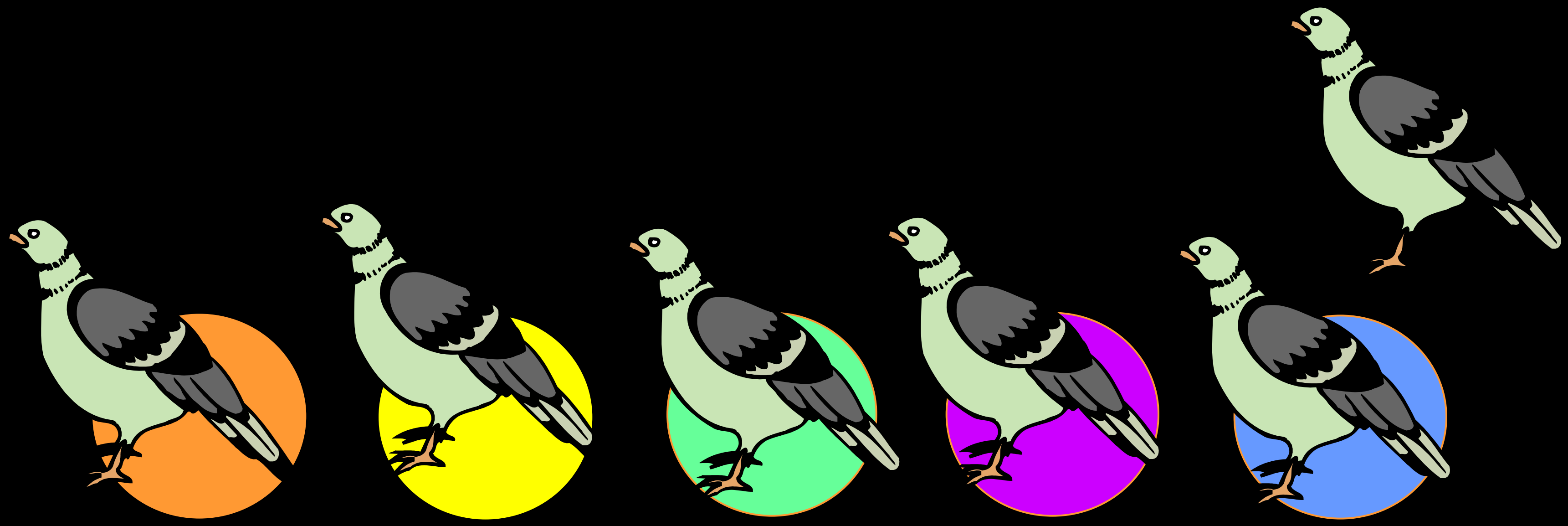
If you put 6 pigeons in 5 holes then at least one hole will have more than one pigeon



HINT 1:

THE PIGEONHOLE PRINCIPLE

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



LEVEL 1

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: There is an integer m and elements $a_1 \neq a_2$ in A

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

LEVEL 3

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

We expect your proofs to have three levels:

- **The first level should be a one-word or one-phrase "HINT" of the proof**

(e.g. "Proof by contradiction," "Proof by induction," "Follows from the pigeonhole principle")

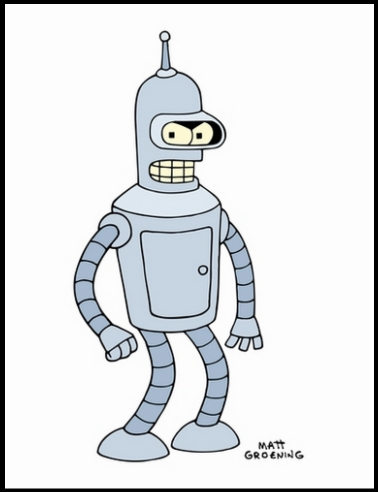
- **The second level should be a short one-paragraph description or "KEY IDEA"**

- **The third level should be the FULL PROOF**

DOUBLE STANDARDS?

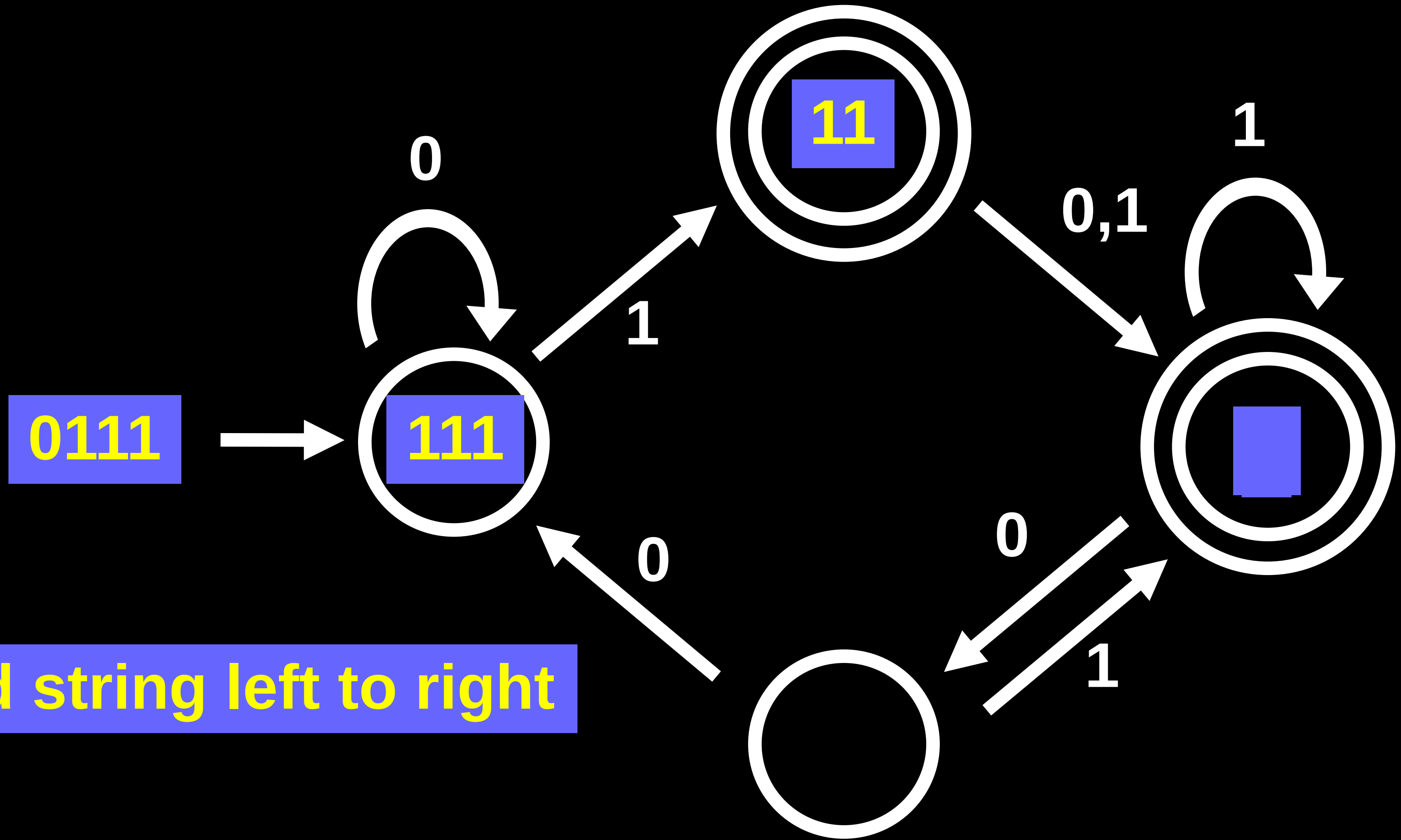
During the lectures, my proofs will usually only contain the first two levels and maybe part of the third

DETERMINISTIC FINITE AUTOMATA



DETERMINISTIC FINITE AUTOMATA
and
REGULAR LANGUAGES

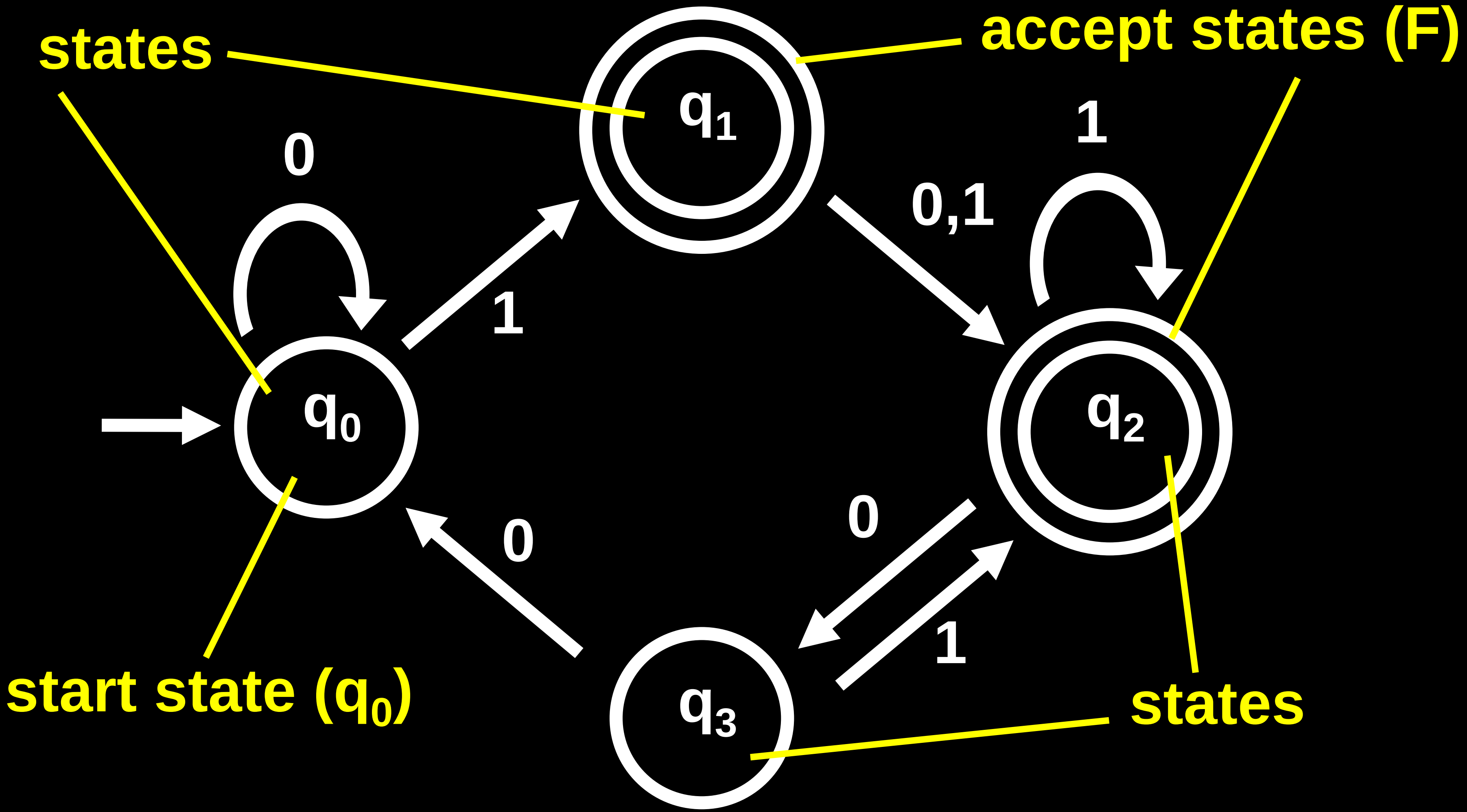
TUESDAY JAN 14



Read string left to right

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

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)

$\delta : Q \times \Sigma \to Q$ 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 $\mathbf{w} = w_1 \ldots w_n \in \Sigma^*$
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)

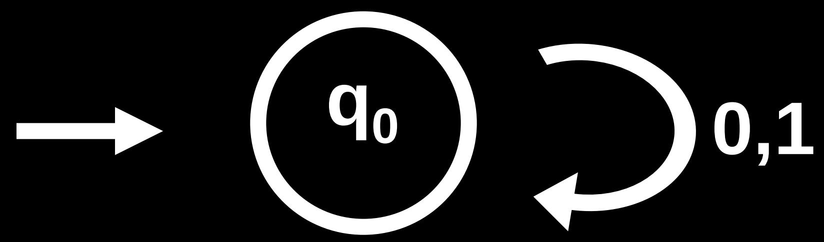
$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

$q_0 \in Q$ is the start state

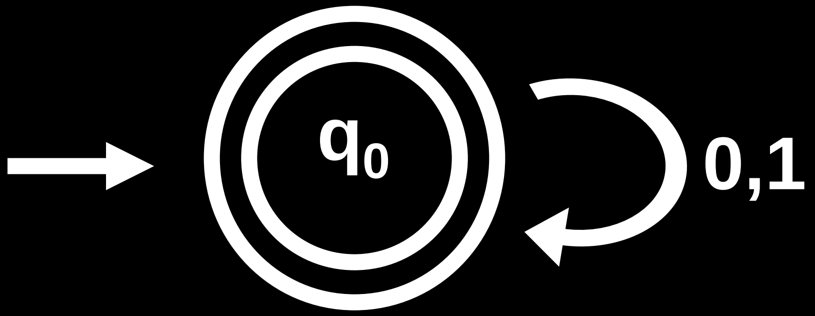
$F \subseteq Q$ is the set of accept/final states

M accepts ε iff $q_0 \in F$

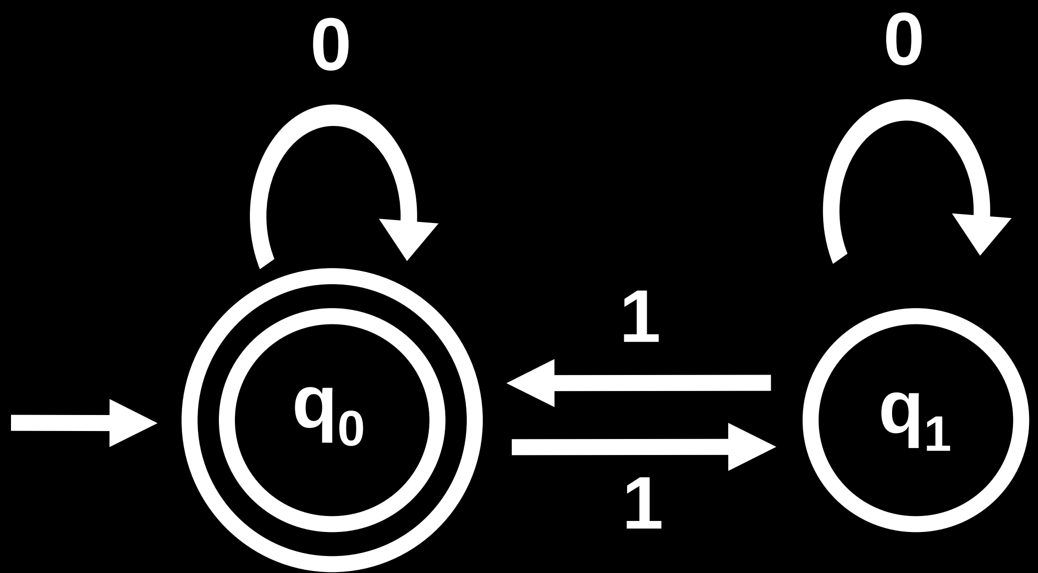
L(M) = set of all strings that M accepts
= “the language recognized by M”



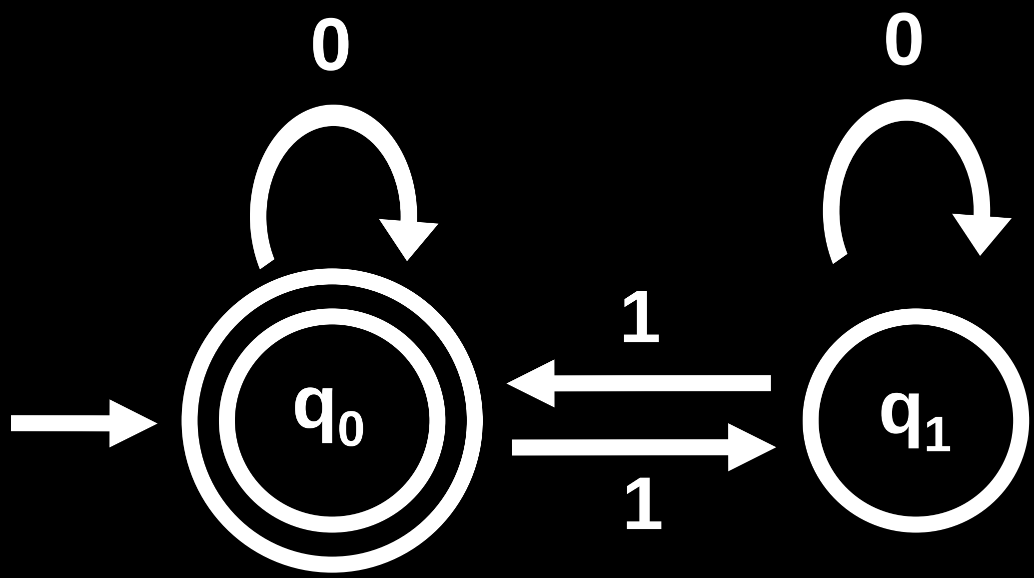
L(M) = ?



L(M) = ?



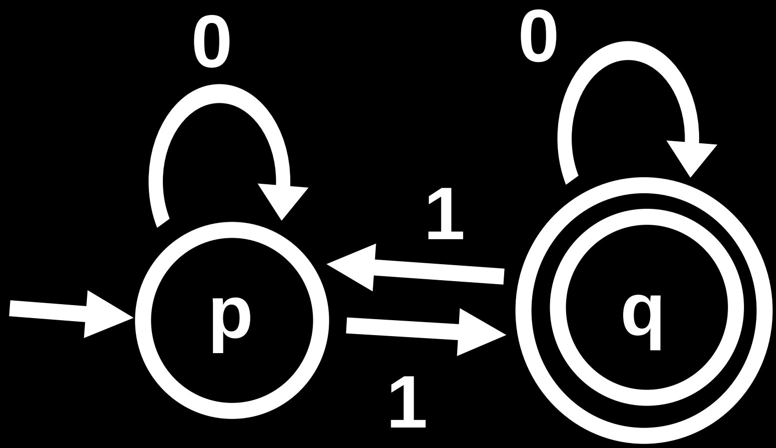
$L(M) = ?$



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

$$\overset{Q}{\{p,q\}},\ \overset{\Sigma}{\{0,1\}},\ \delta,\ \overset{q_0}{p},\ \overset{F}{\{q\}}$$

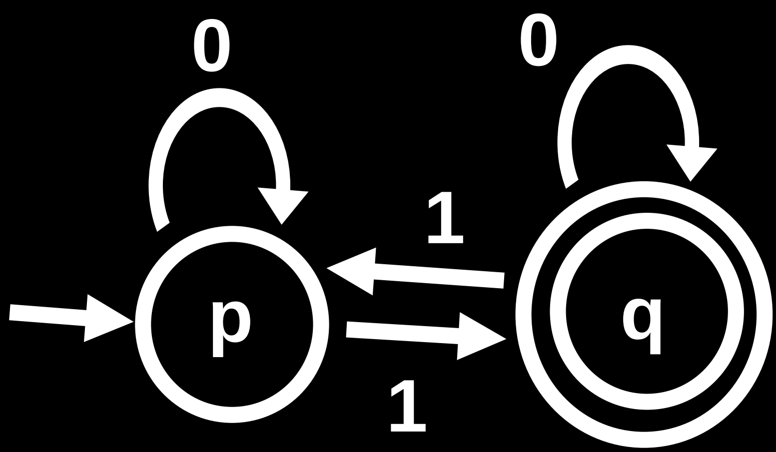
$M = (\{p,q\}, \{0,1\}, \delta, p, \{q\})$



δ	0	1
p	p	q
q	q	p

$$\overset{Q}{}\quad\overset{\Sigma}{}\quad\overset{q_0}{}\ \overset{F}{}$$

$M = (\{p,q\}, \{0,1\}, \delta, p, \{q\})$



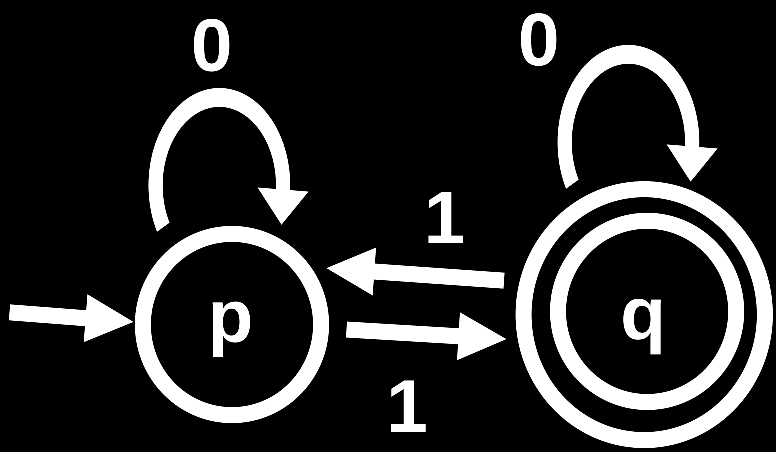
δ	0	1
p	p	q
q	q	p

THEOREM:

$L(M) = \{w \mid w$ has 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 }

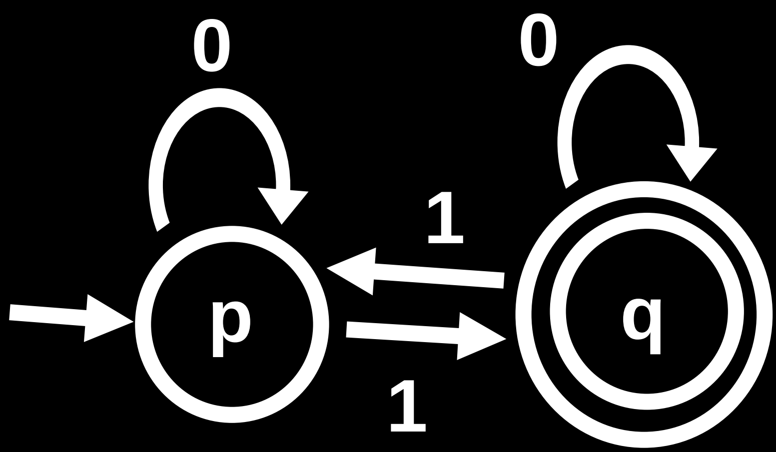
Proof: By induction on **n**, the length of a string.

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

$Q \quad \Sigma \quad q_0 \quad F$

$M = (\{p,q\}, \{0,1\}, \delta, p, \{q\})$

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: $\varepsilon \notin$ RHS and $\varepsilon \notin$ L(M). Why?

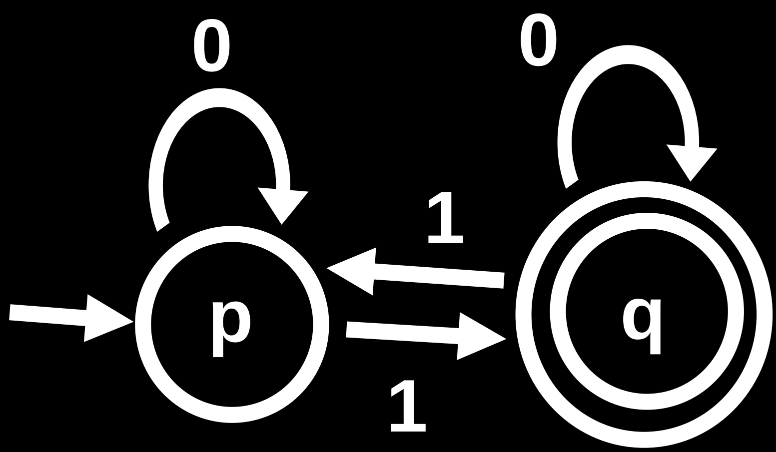
Induction Hypothesis: Suppose for all **w** $\in$ **Σ***, **|w| = n**, **w** $\in$ L(M) iff **w** has odd number of 1s.

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

Q Σ q_0 F

$M = (\{p,q\}, \{0,1\}, \delta, p, \{q\})$

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.

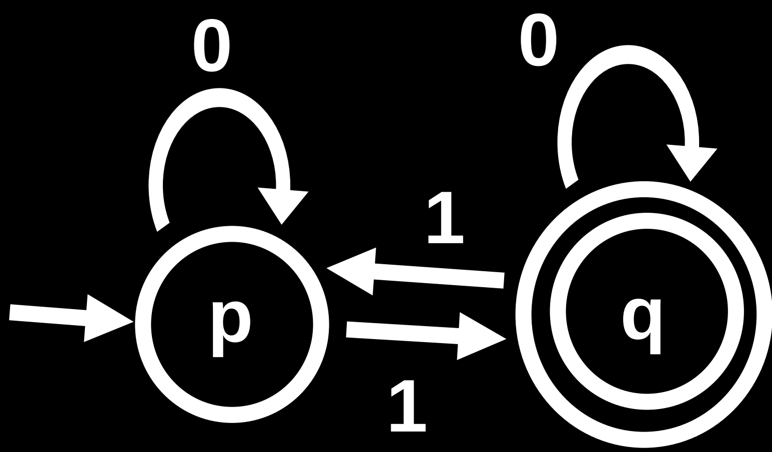
Induction step: Any string of length n+1 has the form w0 or w1.

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)

$Q \quad \Sigma \quad q_0 \quad F$

$M = (\{p,q\}, \{0,1\}, \delta, p, \{q\})$

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.

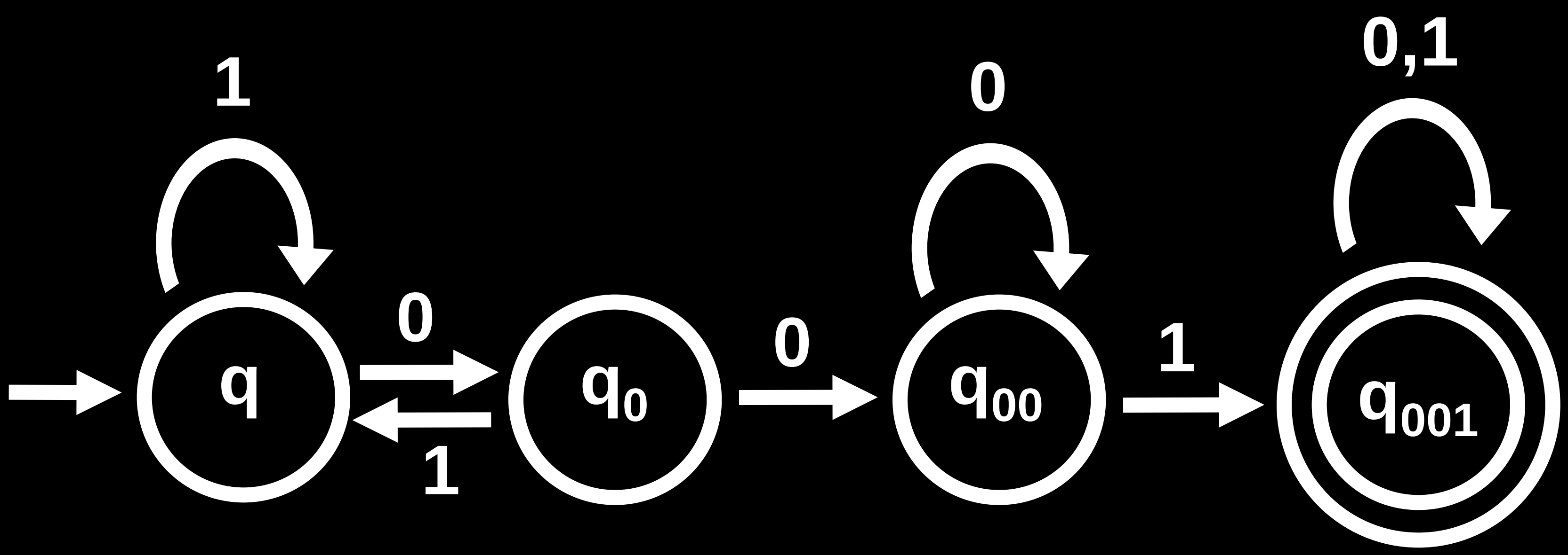
Now w1 has an odd # of 1's ⇔ w has an even # of 1's ⇔

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

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

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

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



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

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

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

$= Q_1 \times Q_2$

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

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

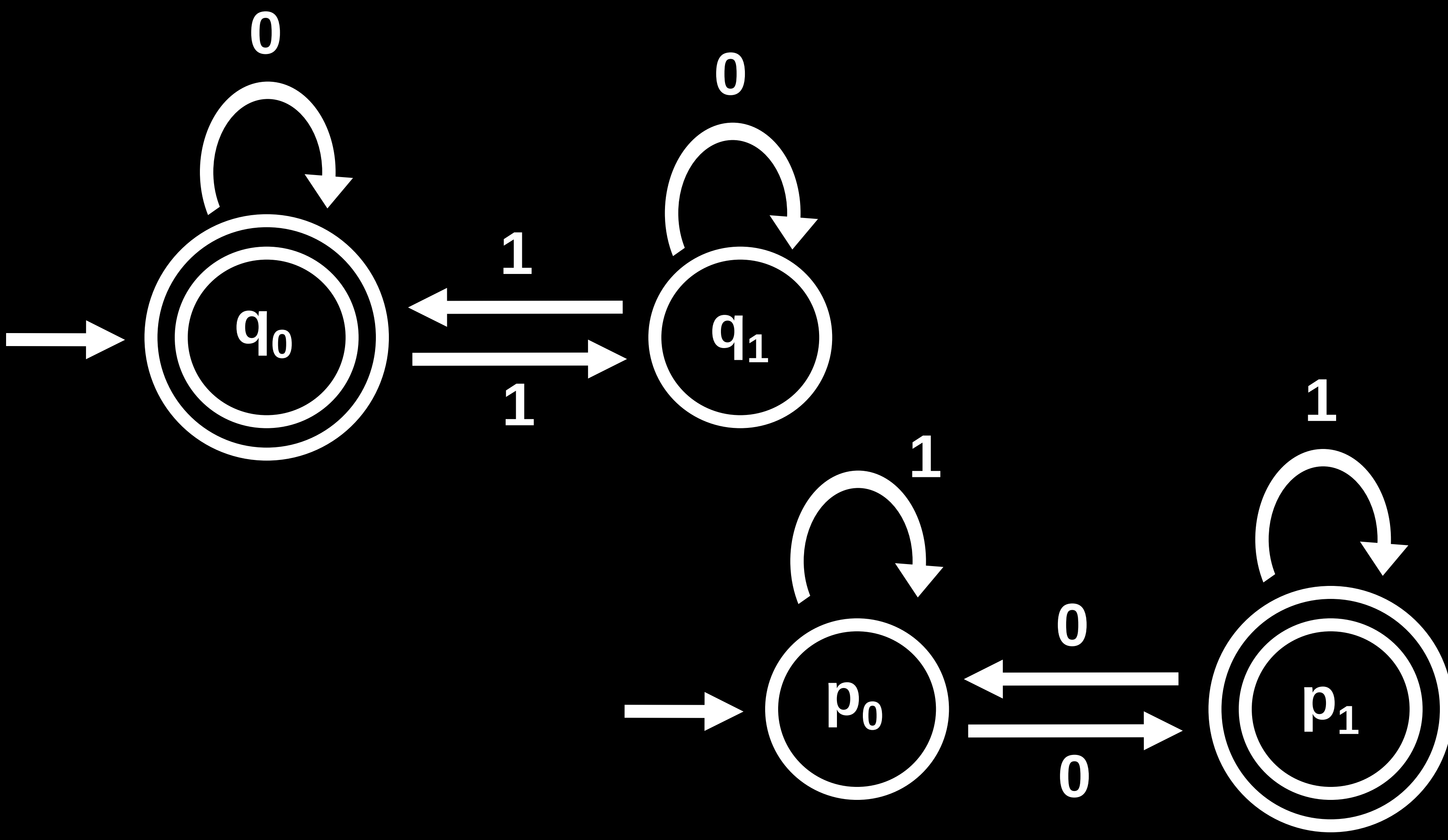
$\delta((q_1,q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$

Intersection THEOREM

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

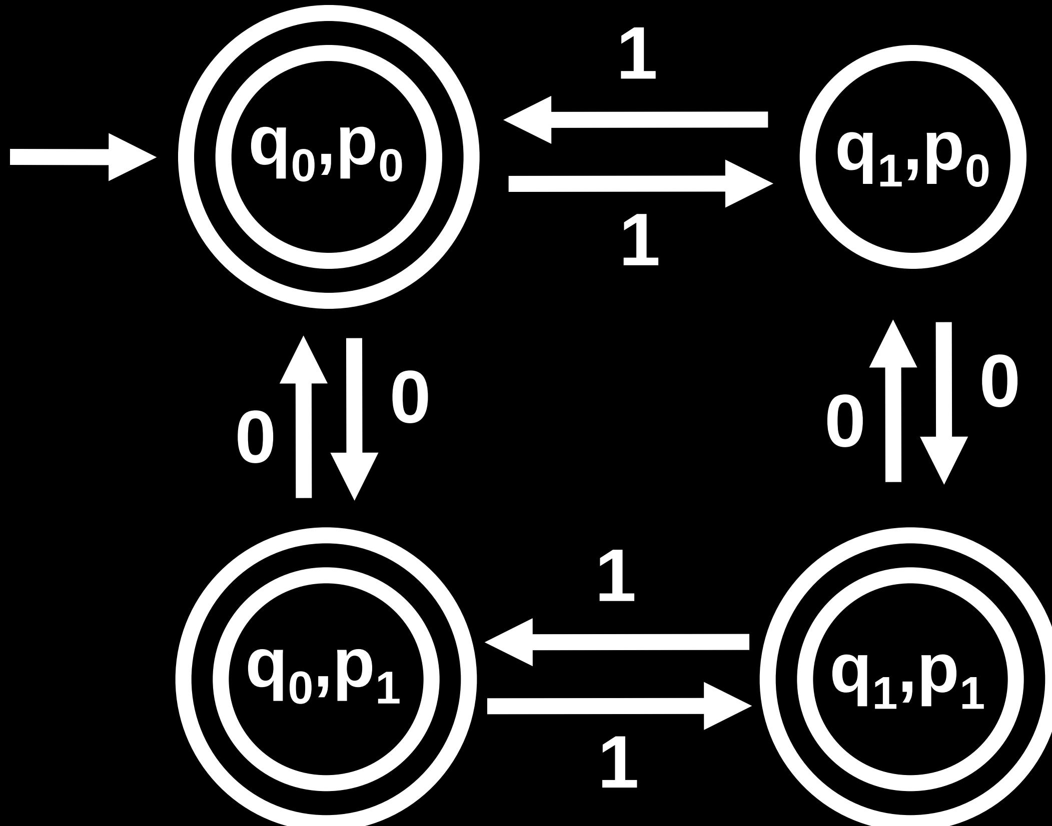
$$L_1 \cap L_2 = \{ w \mid w \in L_1 \text{ and } w \in L_2 \}$$

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

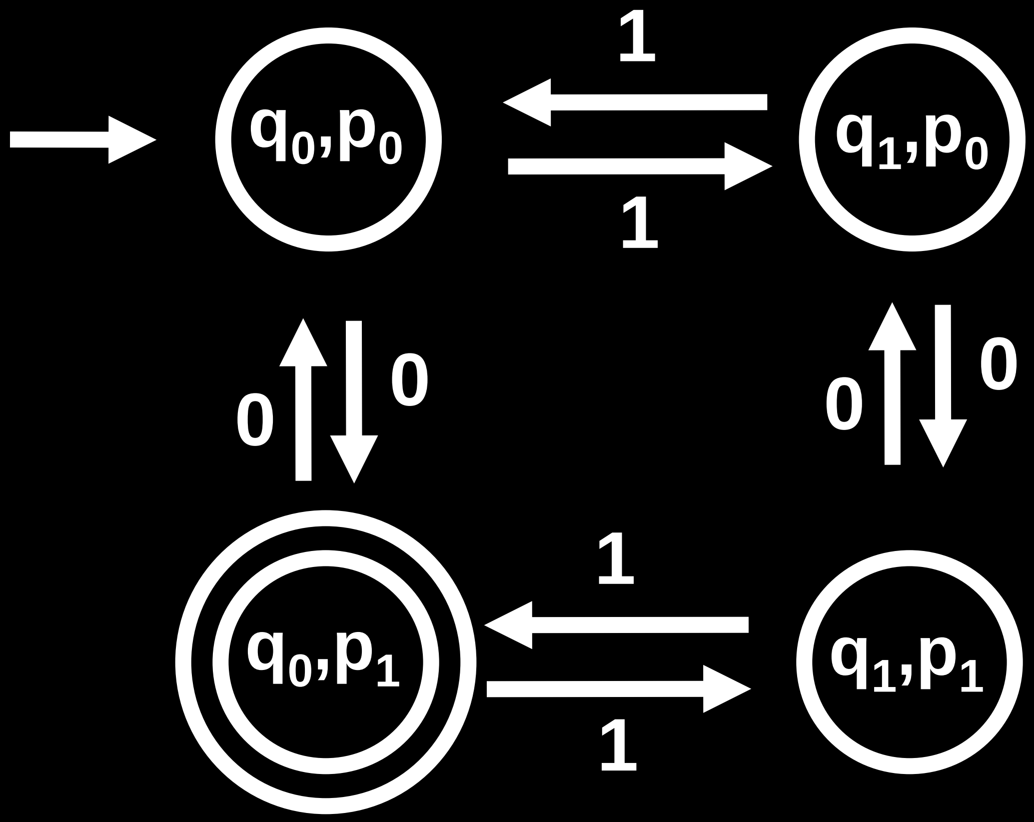


UNION? INTERSECTION?

UNION



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

Show:

$L = \{ x \in \{1, 2, 3\}^* \mid$ the digits 1, 2 and 3 appear in x in that order, but not necessarily consecutively$\}$

is regular.

Show:

$$L = \{\, x \in \{0,1\}^* \mid x \neq \varepsilon \ \text{ and } d(x) \equiv 0 \bmod 3 \,\}$$

is regular.

(d(x) is the *natural* # corresponding to x.)

WWW.FLAC.WS

YOU NEED TO PICK UP
THE SYLLABUS,
THE COURSE SCHEDULE,
THE PROJECT INFO SHEET,
TODAY'S CLASS NOTES

Read Chapters 0, 1.1 and 1.2 of Sipser for next time,
Also Rabin-Scott paper