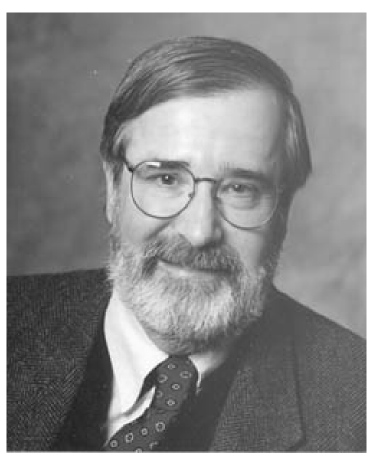
15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

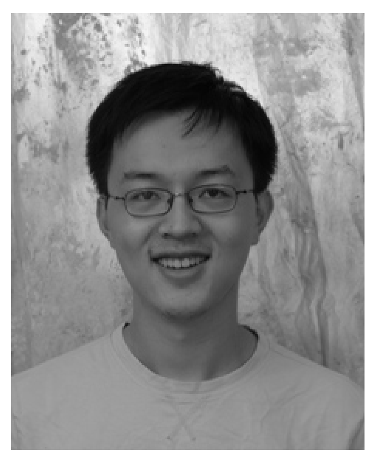
www.cs.cmu.edu/~emc/flac09

INSTRUCTORS

Will Klieber



Edmund Clarke



Yi Wu

Grading

Exams: 50%
- Final Exam: 25%
- Midterm Exam: 25%

Homework: 45%
Class Participation: 5%
Attendance is required.

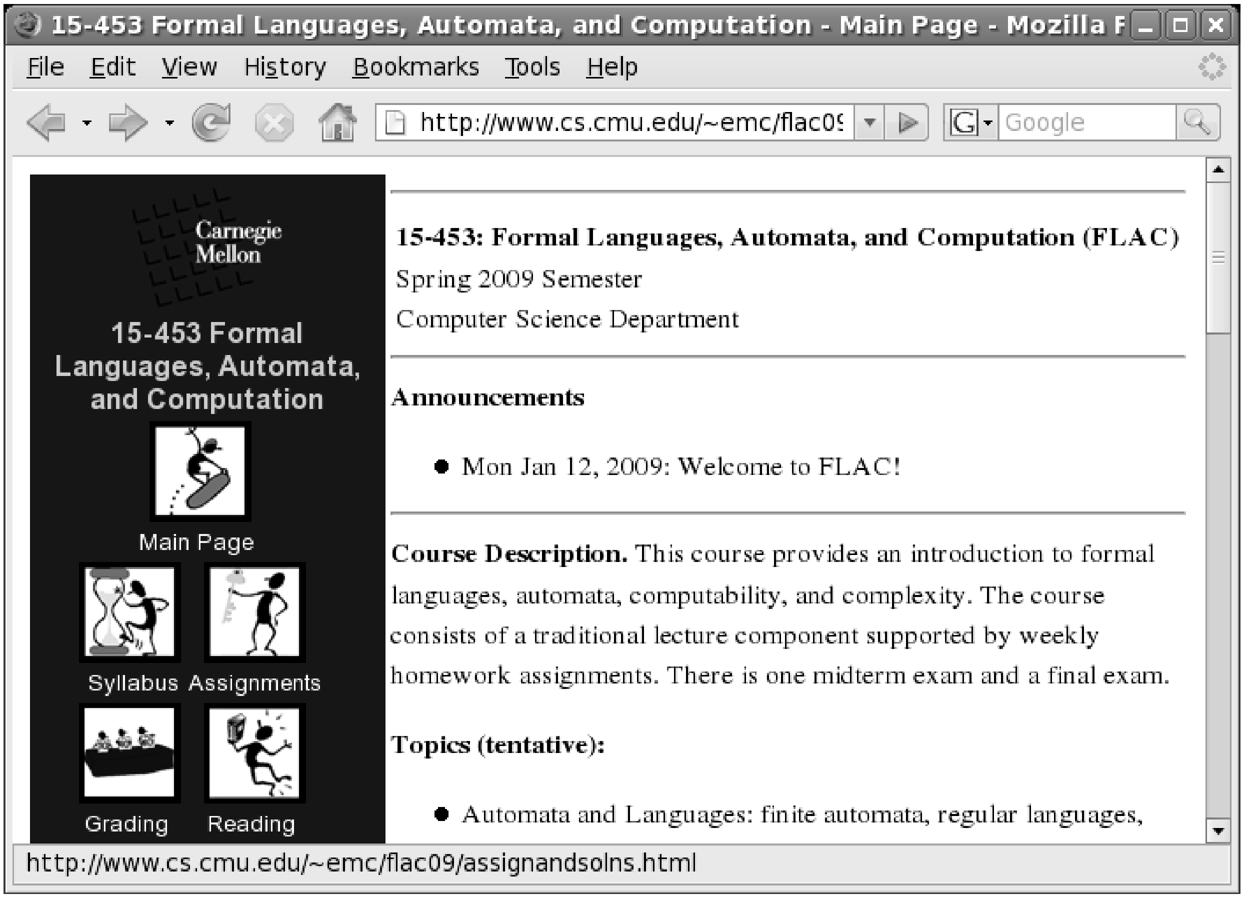
HOMEWORK

Homework will be assigned every Thursday and will be due one week later at the beginning of class

You must list your collaborators and all references in every homework assignment.

Readings will be posted on the course website.
For next class: Read Chapters 0 and 1.1

www.cs.cmu.edu/~emc/flac09



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

PART 2
Computability Theory

PART 3
Complexity and Applications

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 PPACE, 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)

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

PART 2

Computability Theory: (1930's-40's)

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

PART 3

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

time complexity, classes P and NP, NP-completeness, space complexity PPACE, 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

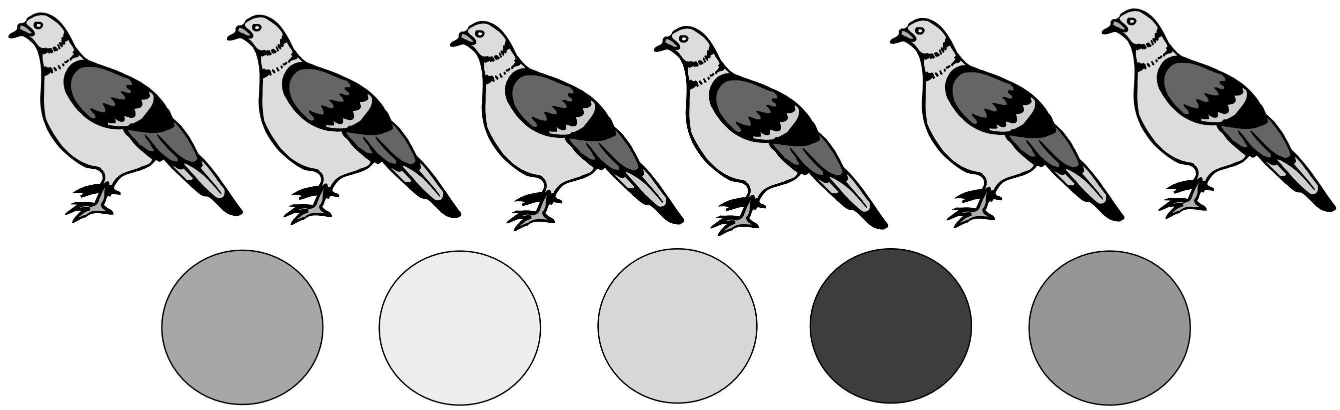
TRUE

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

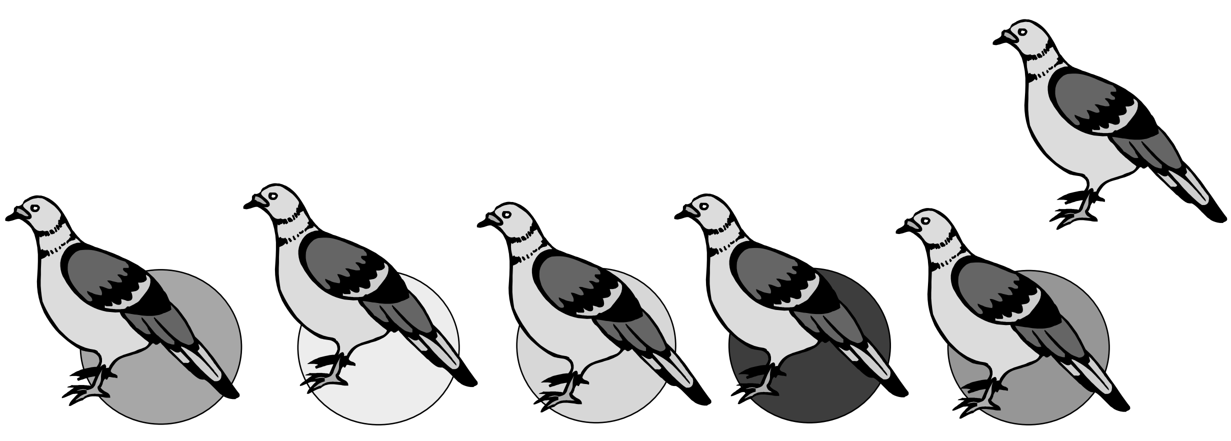


LEVEL 1

HINT 1:

THE PIGEONHOLE PRINCIPLE

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



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 2:

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

LEVEL 2

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 $\{1, \ldots, 2n-1\}$? 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")

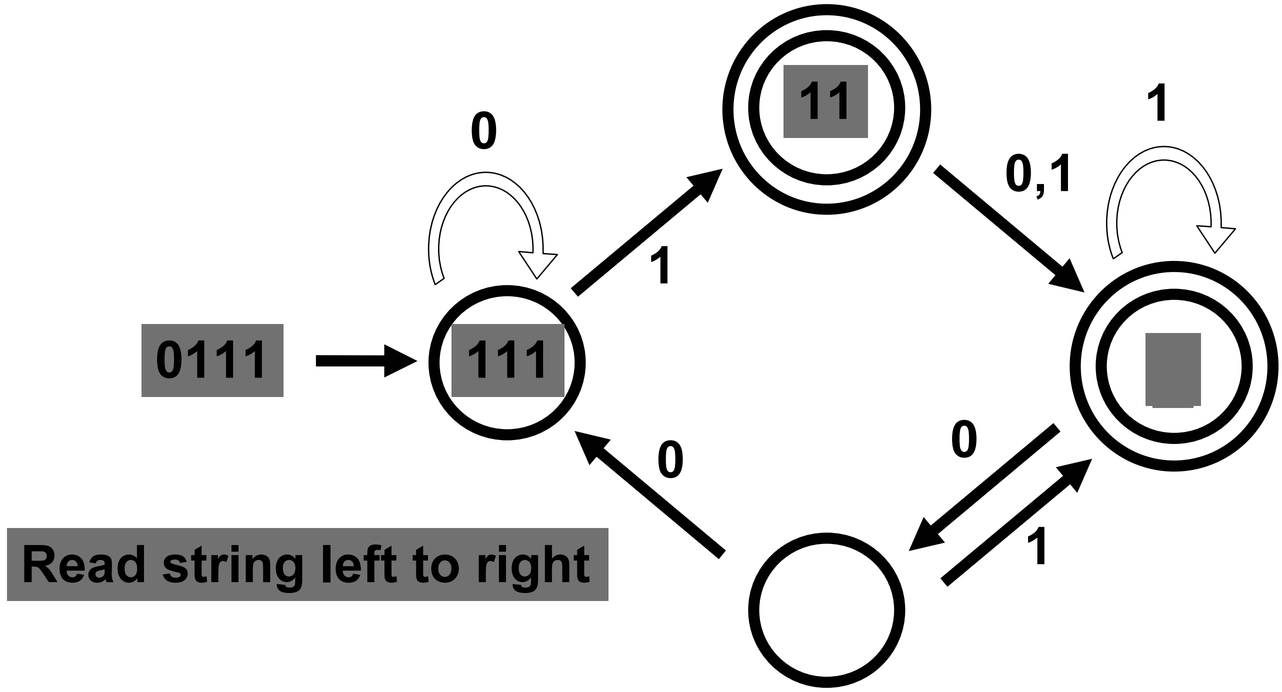
- The second level should be a short one-paragraph description or "KEY IDEA"

- The third level should be the FULL PROOF

DOUBLE STANDARDS?

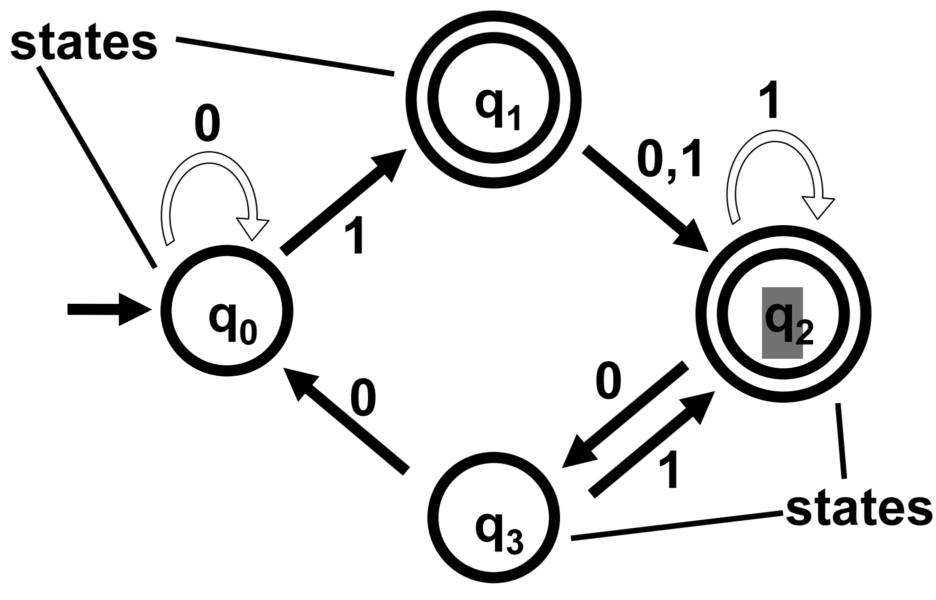
During the lectures, my proofs will usually only contain the first two levels and maybe part of the third

DETERMINISTIC FINITE AUTOMATA



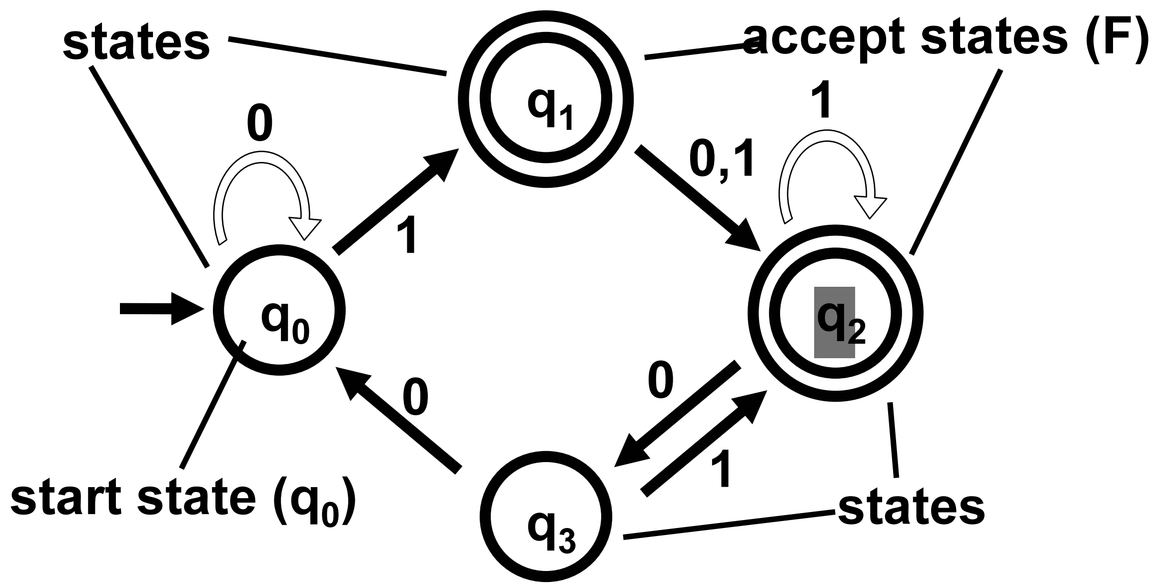
The machine accepts a string if the process ends in a double circle

A Deterministic Finite Automaton (DFA)



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

A Deterministic Finite Automaton (DFA)



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

NOTATION

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* denotes the set of finite length sequences of elements of Σ

For x a string, |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, ie, a subset of Σ*

A deterministic finite automaton (DFA) is represented by 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 states

Let $w_1, \ldots, w_n \in \Sigma$ and $\mathbf{w} = w_1 \ldots w_n \in \Sigma^*$
Then **M** accepts **w** if 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
- $r_n \in F$

A deterministic finite automaton (DFA) is represented by a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$:

Q is the set of states (finite)

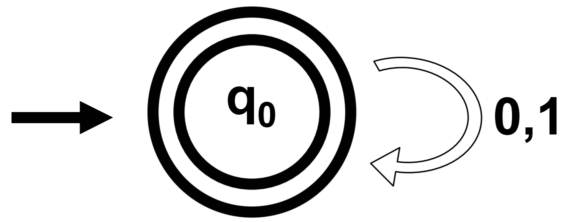
Σ is the alphabet (finite)

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

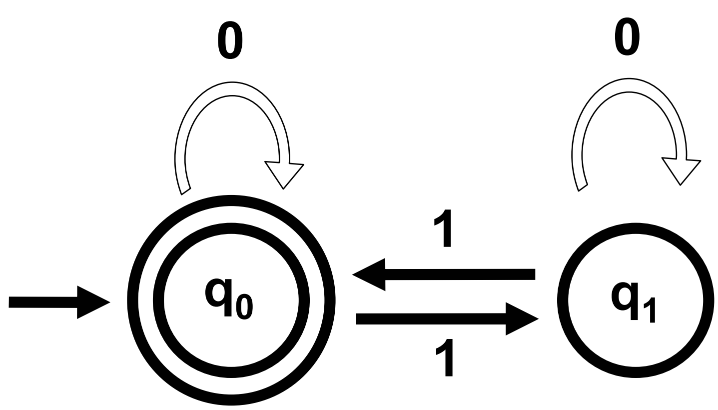
$q_0 \in Q$ is the start state

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

L(M) = the language of machine M
= set of all strings machine M accepts

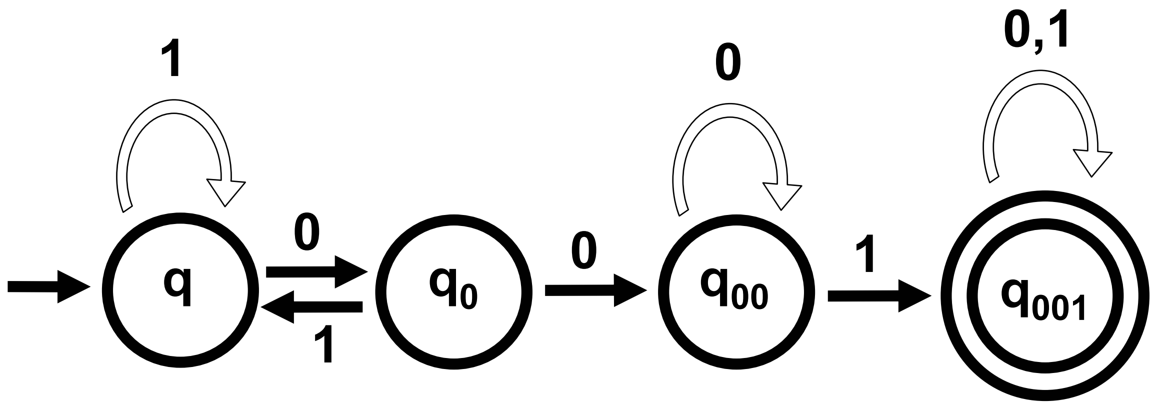


$L(M) = \{0,1\}^*$



$L(M) = \{ w \mid w \text{ has an even number of 1s}\}$

Build an automaton that accepts all and only those strings that contain 001



A language L is regular if it is recognized by a deterministic finite automaton (DFA), i.e. if there is a DFA M such that L = L (M).

L = { w | w contains 001} is regular

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

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

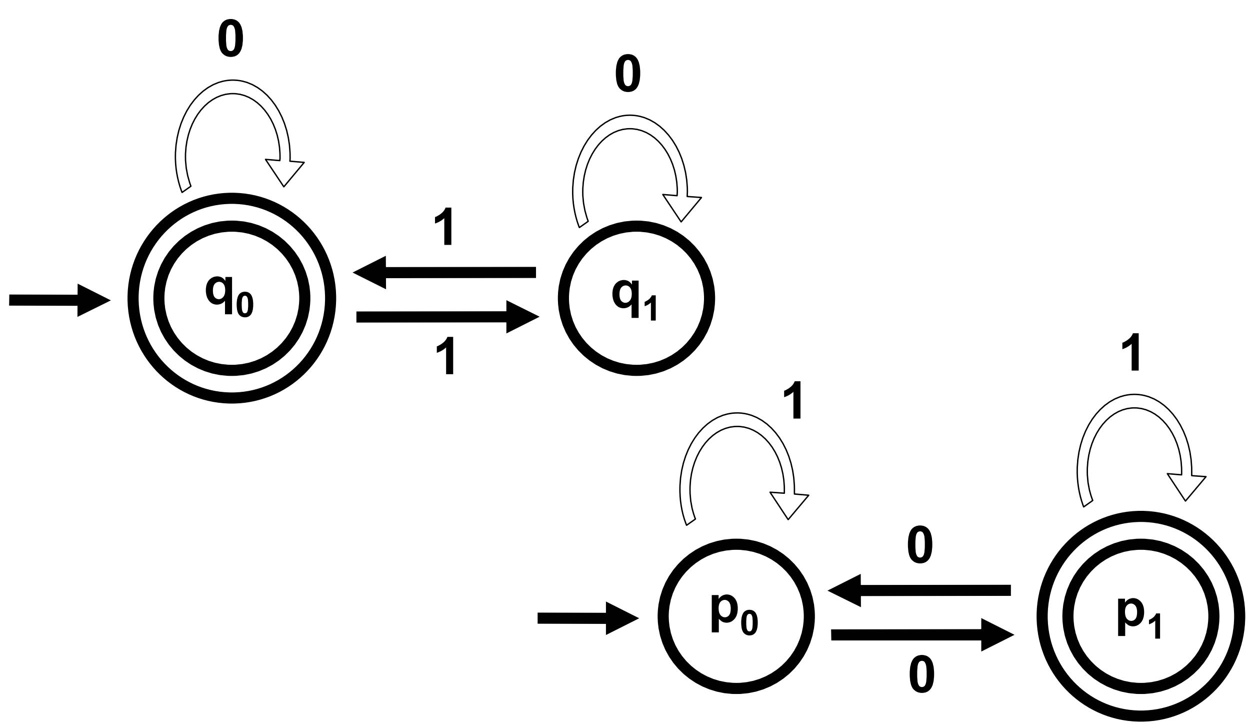
$= Q_1 \times Q_2$

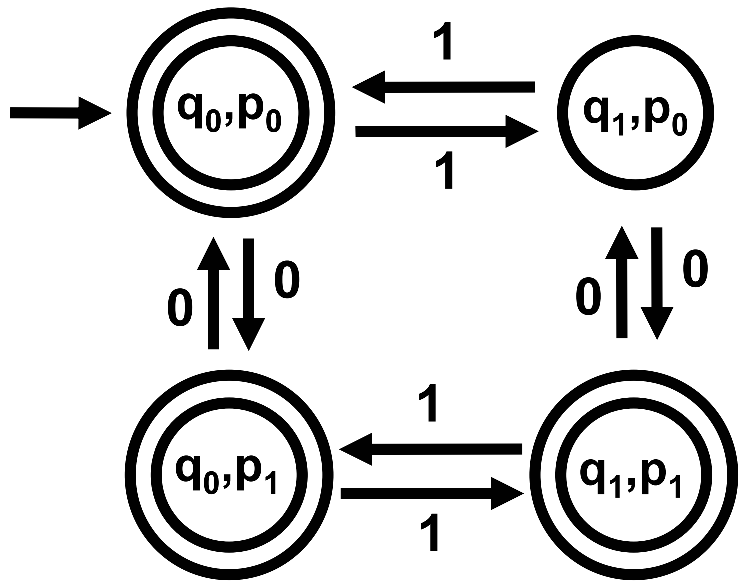
$q_0 = (q_0^1, q_0^2)$

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

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

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





Intersection **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

FLAC

Read Chapters 0 and 1.1 of the book for next time