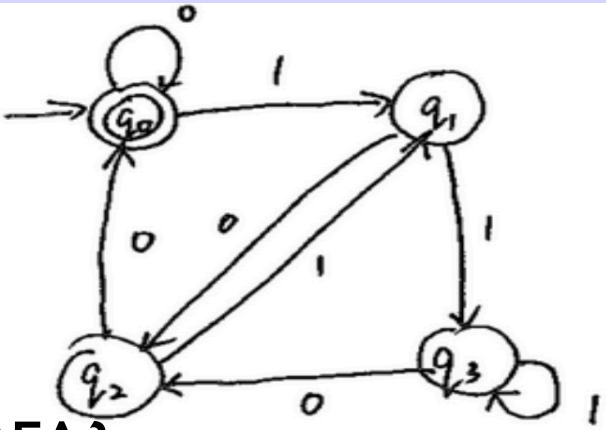
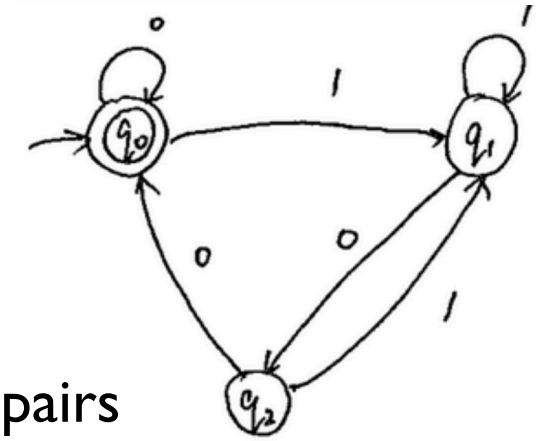
Binary Number Divisible by 4

- 4-state solution (trivial)
- 3-state solution (merge q1 w/ q3)
- in general, how do you:
  - reduce a DFA to a smaller but equivalent DFA?
    - see Linz 2.4 or Sipser problem 7.42 (p. 327)
    - will discuss later after NFA
  - test if two DFAs are equivalent?
    - follow pairs of states, check if all visited state-pairs agree on finality (both accept or both reject)
    - $O(n^2 \Sigma)$ time and space

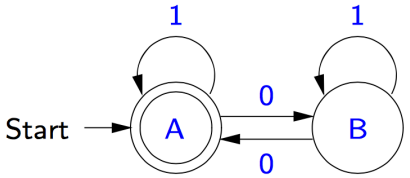


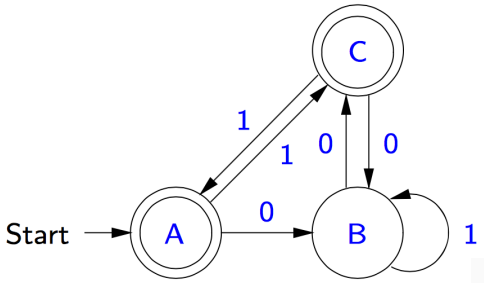


https://www.cse.iitb.ac.in/~trivedi/courses/cs208-spring14/lec05.pdf

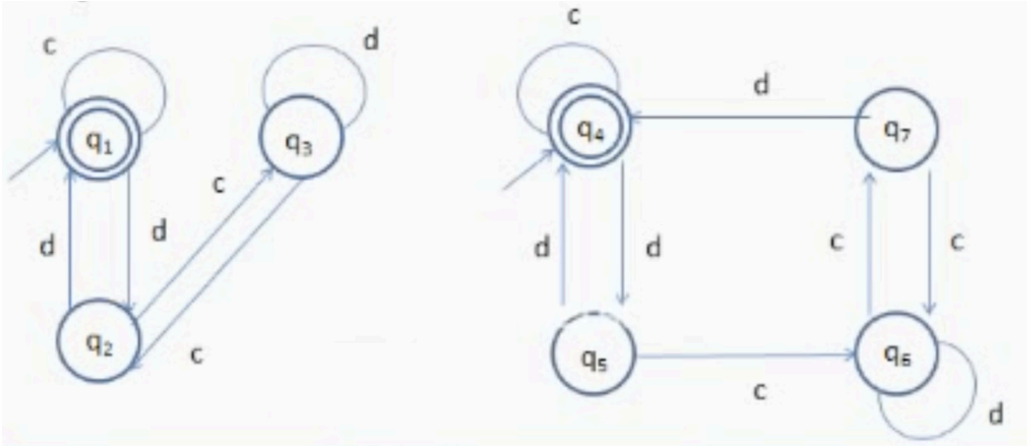
Test if two DFAs are equivalent

- traverse all state-pairs and make sure each pair agrees on "finality" (both accept or both reject)





pair	final?	on 0	on 1
(A, A)	(y, y)	(B, B)	(A, C)
(B, B)	(n, n)	(A, C)	(B, B)
(A, C)	(y, y)	(B, B)	(A, A)
no new pairs found			



Q Q'	Q_c Q_c'	Q_d Q_d'
q_1 , q_4	q_1 , q_4	q_2 , q_5
q_2 , q_5	q_3 , q_6	q_1 , q_4
q_3 , q_6	q_2 , q_7	q_3 , q_6
q_2 , q_7	q_3 , q_6	q_1 , q_4

Proof by Induction (Linz 1.1-1.2, Sipser 0.4)

- Theorem to prove: $|uv| = |u| + |v|$
- first define string length rigorously and inductively:
  - ~~$|a| = 1$~~, $|\varepsilon| = 0$, $|wa| = |w| + 1$
- now prove the Theorem by induction on $|v|$
  - base case: $|v| = 0$, then $v = \varepsilon$, so $|uv|=|u|=|u|+0=|u|+|v|$
  - inductive case: assume Theorem holds for any $|v|$ of length $0..n$ (IH)
    Now take any v of length $n+1$. Let $v = wa$, then $|v| = |w|+1$ (by def.)
    - then $|uv| = |uwa| = |uw|+1$ (by definition)
    - by induction hypothesis (applicable to any w of length n)
      - $|uw| = |u|+|w|$, so that $|uv| = |u|+|w|+1 = |u|+|v|$

HW: prove by induction on $|u|$

What's wrong with this proof?

Theorem(?!): All horses are the same color.

Proof: Let $P(n)$ be the predicate "in all non-empty collections of n horses, all the horses are the same color." We show that $P(n)$ holds for all n by induction on n (using 1 as the base case).

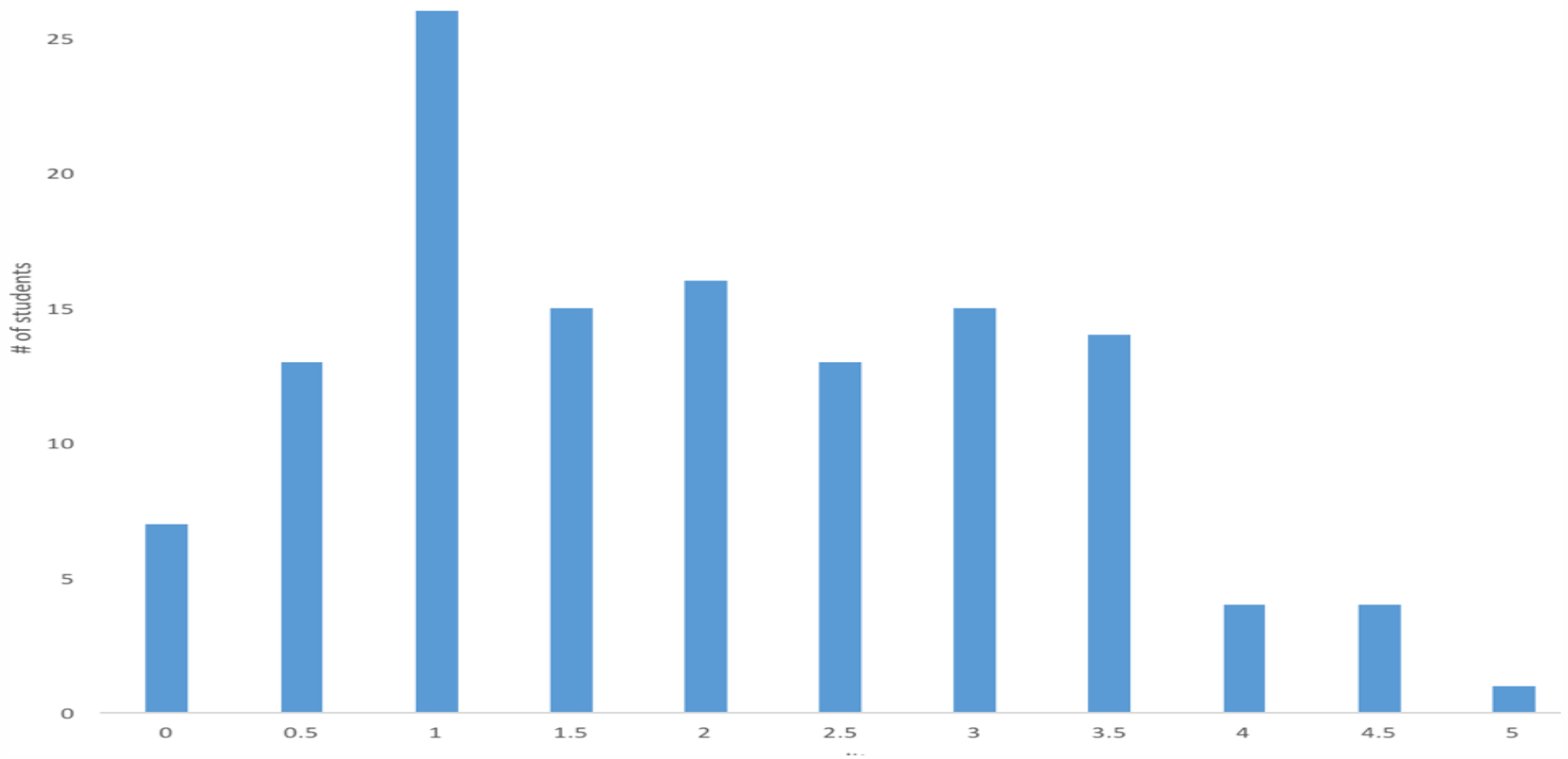
Base case: Clearly, $P(1)$ holds.

Induction case: Given $P(n)$, we must show $P(n+1)$.

Consider an arbitrary collection of $n+1$ horses. Remove one horse temporarily. Now we have n horses and hence, by the induction hypothesis, these n horses are all the same color. Now call the exiled horse back and send a different horse away. Again, we have a collection of n horses, which, by the induction hypothesis, are all the same color. Moreover, these n horses are the same color as the first collection. Thus, the horse we brought back was the same color as the second horse we sent away, and all the $n+1$ horses are the same color.

Quiz 1 scores and Projected Final Grade

- mean: 2.0, median: 2



projected final grade

0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
F		C/C-	C+	B-	B	B+	A-	A		

Regular Language

DEFINITION

A language is a **REGULAR LANGUAGE** iff some Finite State Machine recognizes it.

What languages are NOT regular?

Anything that requires memory.

The F.S.M. memory is very limited

Cannot store the string.

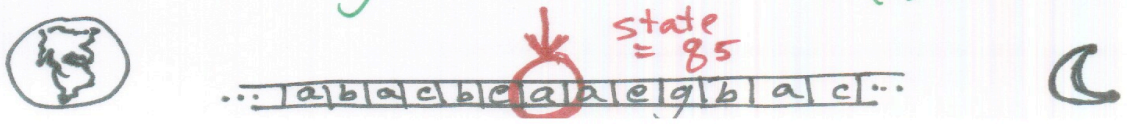
Cannot "count."

Not Regular:

ww 01101, 01101 (w, w)

$0^N 1^N$ 000000, 111111 (6, 6)

Imagine a String from 6 here to 6 the Moon. You are trying to recognize it. Your only memory: A single small number (i, # of states)



Regular Operations

UNION

$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

CONCATENATION

$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

STAR "closure"

$A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

EXAMPLE

$\Sigma = \{a, b, c, \ldots z\}$

$A = \{aa, b\}$

$B = \{x, yy\}$

$A \cup B = \{aa, b, x, yy\}$

$A \circ B = \{aax, aayy, bx, byy\}$

$A^* = \{\epsilon, aa, b, aaaa, aab, baa, bb, aaaaaa, aaaab, aabaa, aabb, \ldots\}$

- other operations:
  - intersection
  - complement
  - difference
- regular languages are <u>closed</u> under all these operations

Union

THEOREM

The class of Regular Languages is CLOSED under UNION.

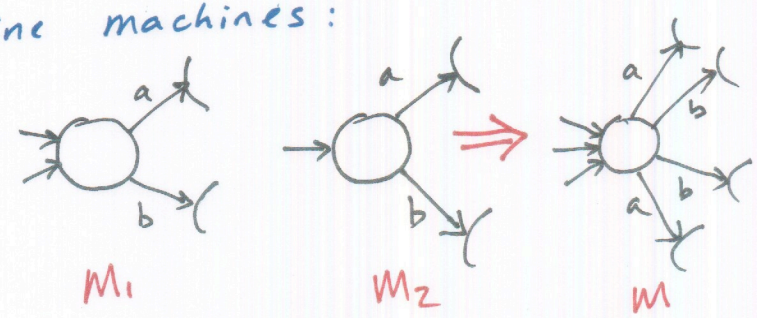
If L_1 and L_2 are regular languages, then so is $L_1 \cup L_2$.

PROOF (BY CONSTRUCTION)

Assume $L_1 = L(M_1)$
$L_2 = L(M_2)$

Build M to recognize $L_1 \cup L_2$.

Combine machines:



M_1 M_2 M

WHOOPS!
Not a F.S.M!

- the union of regular languages is still regular

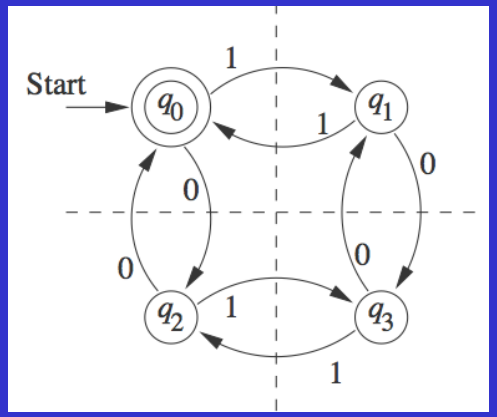
Union by Cross-Product Construction

Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ be two FA's.

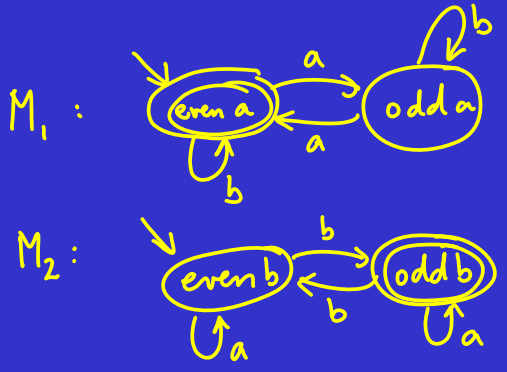
Union:

We know that $L(M_1) \cup L(M_2)$ is regular.
Construct an FA M such that $L(M) = L(M_1) \cup L(M_2)$.

Construction:
see next slide



example:



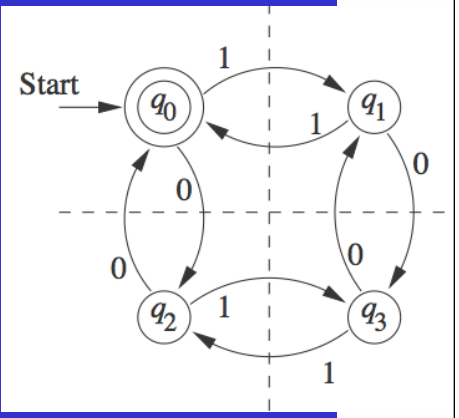
$L(M_1) \cup L(M_2)$ contains the strings over $\{a,b\}$ s.t. the string has an even #a's or an odd #b's

$aabbb \in L(M_1) \cup L(M_2)$

see Sipser 1.1, Linz

Intersection by Cross-Product Construct.

Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ be two FA's.



Intersection:

Is $L(M_1) \cap L(M_2)$ is regular ?
Construct an FA M such that $L(M) = L(M_1) \cap L(M_2)$.

$M = (Q, \Sigma, q_0, A, \delta)$ where

$Q = Q_1 \times Q_2$

$q_0 = (q_1, q_2)$

$A = A_1 \times A_2$

$\delta((p,q), \sigma) = (\delta_1(p,\sigma), \delta_2(q,\sigma))$

$\forall p \in Q_1 \quad \forall q \in Q_2 \quad \forall \sigma \in \Sigma$

IDEA:
automata "walk" together,
new states = pair of states,
one from M_1, one from M_2

see the "even a, odd b" example
a few slides back

for union: $A = A_1 \times Q_2 \cup Q_1 \times A_2$

Complement and Difference

Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ be an FA.

Complement:

Is $L(M_1)'$ regular ?
Construct an FA M such that $L(M) = L(M_1)'$.

Idea: switch accepting for non-accepting states in M_1

$M = (Q, \Sigma, q_0, A, \delta)$ where $Q = Q_1$

$q_0 = q_1$

$\delta = \delta_1$

$A = A_1' = Q_1 - A_1$ ✓

Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ be two FA's.

Difference:

Construct an FA M such that $L(M) = L(M_1) - L(M_2)$.

Summary: Union, Intersection, Difference

- "cross-product" construction (also used in table-filling algorithm)

$$M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$$
$$M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$$
$$M = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$$
$$\delta : (Q_1 \times Q_2) \times \Sigma \mapsto Q_1 \times Q_2,$$
$$\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a))$$
$$F_\cap = F_1 \times F_2$$
$$F_\cup = F_1 \times Q_2 \cup Q_1 \times F_2$$
$$F_- = F_1 \times \bar{F}_2$$

LaTeX source:

```
M_1 & = (Q_1, \Sigma, \delta_1, q_{01}, F_1)\\
M_2 & = (Q_2, \Sigma, \delta_2, q_{02}, F_2)\\
M & = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)\\
\delta & : (Q_1 \times Q_2) \times \Sigma \mapsto Q_1 \times Q_2, \\
& \delta  ((p,q), a) = (\delta_1(p,a), \delta_2(q,a))\\
F_\cap & = F_1 \times F_2   \\
F_\cup & = F_1 \times Q_2 \cup Q_1 \times F_2 \\
F_- & = F_1 \times \bar{F_2} \\
\\
\delta^* & : (Q_1 \times Q_2) \times \Sigma^* \mapsto Q_1 \times Q_2, \\
\delta^* & ((p,q), w) = \begin{cases}
                          (p,q) & w = \epsilon \\
                          \delta(\delta^*((p,q), x), a)  & w = xa
                        \end{cases}\\
&\text{alternatively, }\\
\delta^* & ((p,q), w) = (\delta_1^*(p, w), \delta_2^*(q, w))
```

$$\delta^* : (Q_1 \times Q_2) \times \Sigma^* \mapsto Q_1 \times Q_2,$$

$$\delta^*((p,q),w) = \begin{cases} (p,q) & w = \epsilon \\ \delta(\delta^*((p,q),x),a) & w = xa \end{cases}$$

alternatively,

$$\delta^*((p,q),w) = (\delta_1^*(p,w), \delta_2^*(q,w))$$

HW: prove equivalence