# CS 3313
# Foundations of Computing:

# Properties of RLs and non-RL
# Exam 1 Review

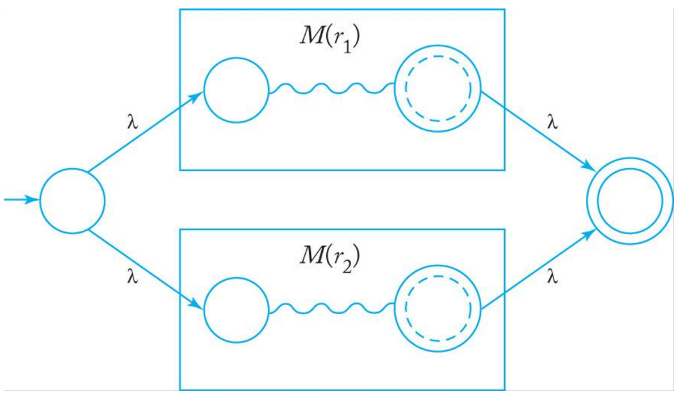
http://gw-cs3313-2021.github.io

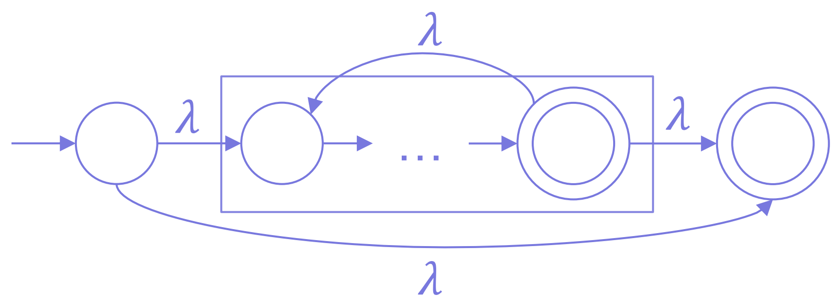
# Properties of Regular Languages

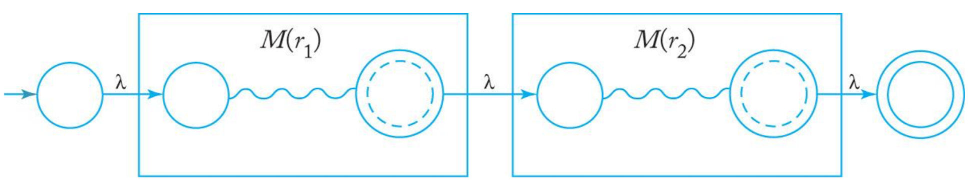
- **Closure Properties**: what happens when we “combine” two regular languages or perform set operations on them ?
  - Ex: Is Intersection of two regular languages still a regular language ?
  - Why is this important ?
    - Construct a larger set from smaller sets
    - Problem decomposition
- **Decision Problems**: can we provide procedures to determine properties of a language ?
  - Ex: are two machines equivalent? Does a DFA accept an infinite set ?
- How do we determine if a language does not belong to that class of languages ?
  - Ex: **Non-Regular Languages**: How do we show that a language (problem?) cannot be accepted by a DFA ?

# Summary of Closure Properties

- Regular **languages** are closed under Union, Concatenation, star closure, complementation, reversal, intersection, homomorphism (and reverse homomorphisms).

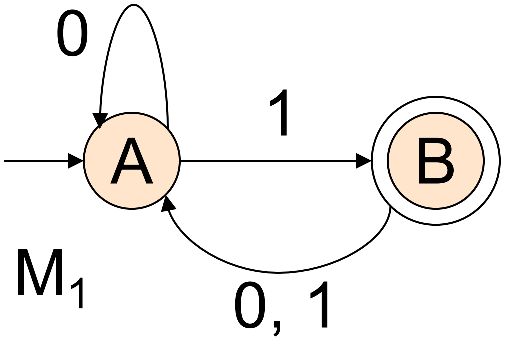


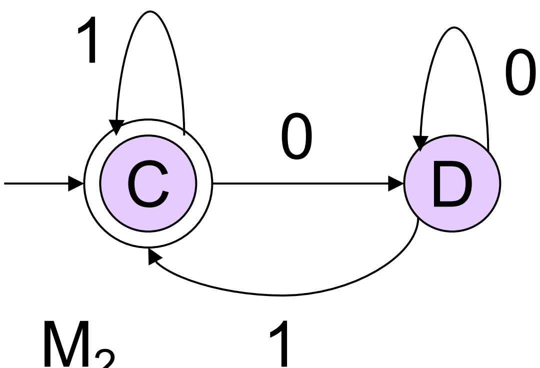




- Complementation
  - Swap final and non-final states.
- Reversal (HW1)
  - Reverse the arrows.
- Homomorphism (Piazza @31)
  - Substitutions.

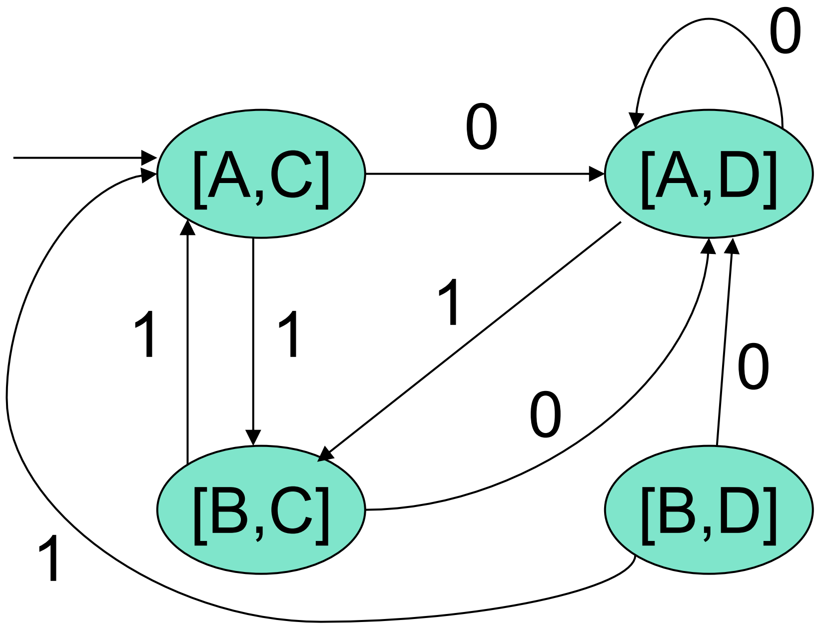
# Product DFA: Example




$M_1$

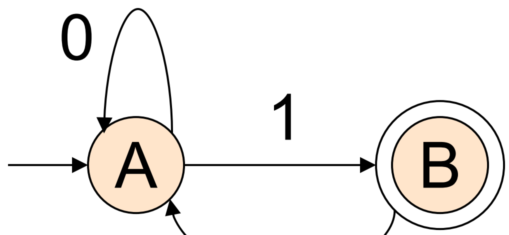



$M_2$

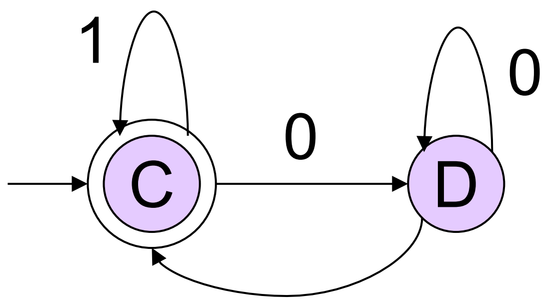



- Product DFA has set of states $Q \times R$
  - Not all are reachable.
- Start state = $[q_0, r_0]$
  - Start state in each machine.
- Transitions: $\delta([q,r], a) = [\delta_1(q,a), \delta_2(r,a)]$
  - $\delta([A,C], 1) = [\delta_1(A,1), \delta_2(C,1)] = [B,C]$
  - $\delta([A,C], 0) = [\delta_1(A,0), \delta_2(C,0)] = [A,D]$
  - $\delta([A,D], 0) = [\delta_1(A,0), \delta_2(D,0)] = [A,D]$

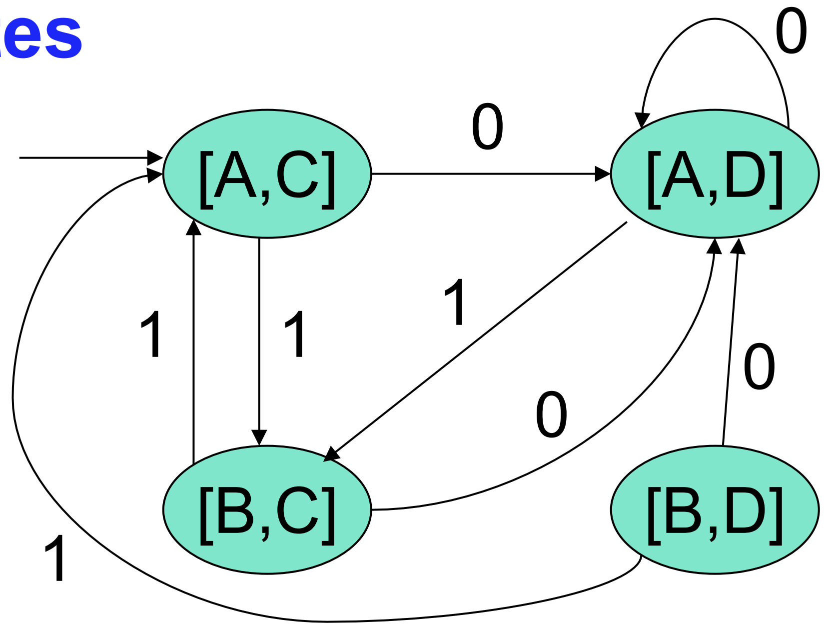
# Product DFA: Final States




$M_1$




$M_2$




- Final States Set F depends on the operator(s)
  - $\overline{L_2}$: $F_1 = \{D\}$
  - $L_1 \cup L_2$: $F = \{[q,r] \mid q \in F_1 \text{ or } r \in F_2\}$
  - $L_1 \cap L_2$: $F = \{[q,r] \mid q \in F_1 \text{ and } r \in F_2\}$
  - $L_1 - L_2$: $F = \{[q,r] \mid q \in F_1 \; and \; r \notin F_2\}$
  - $= L_1 \cap \overline{L_2}$
  - $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$: $F = \{[q,r] \mid \ldots\}$
  - $= (L_1 - L_2) \cup (L_2 - L_1) = L_1 \text{ XOR } L_2$

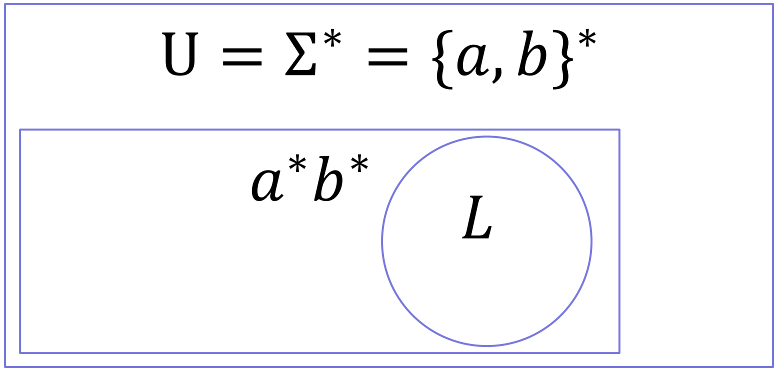
# Examples: Applying closure properties

Skip: Please make sure you understand.

- L1={ w | w has a's followed by b's}
- L2={ w | w has even length}
- L3 = { w | w has odd number of a's and even number of b's}
- **If L1,L2, L3 are regular then:**
- L1 ∪ L2 = {w | w has a's followed by b's or w has even length} is regular
- L1 ∩ L3 = { w | w has odd number of a's followed by even number of b's} is regular
- $\overline{L1}$ = {w| w does not have a's followed by b's } is regular
- L = L1 ∩ $\overline{L3}$ = {w| w has a's followed by b's and not (a is odd and b is even) } is regular

# From HW2

- $L_4$ is the complement of $L = \{a^{2n}b^{2m+1} \mid n, m \geq 0\}$.




1. In the $a^*b^*$ form:

2. Not in the $a^*b^*$ form:

- $L_4$ is $\overline{L} \cap a^*b^*$

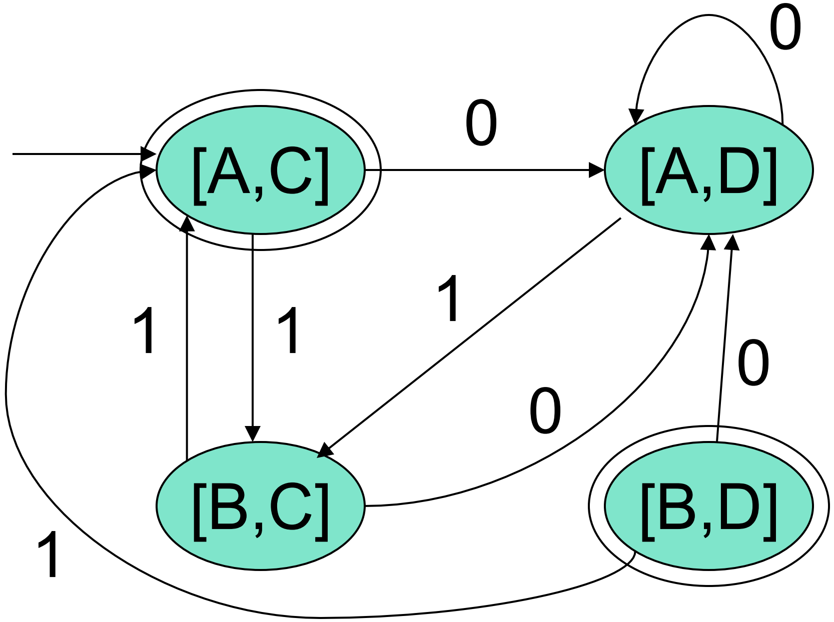
# Decision Properties for Regular Languages

- Membership: Is w in L(M) ?
- Emptiness: Is L(M) empty ?
- **Equivalence**: Is L(M1) = L(M2) ?
- Subset: Is L(M1) a subset of L(M2) ?
- Infiniteness: Is L(M) infinite ?

- **Equivalence**: Is $L_1 = L_2$ for the two <u>regular languages</u>.
  - $L_2 = \left(L_3 \cup \overline{(L_4 - L_5)}\right)^* \cap \ldots \ldots$
  - As long as RLs are (proven to be) closed under those operators.

# Example: Equivalence

- If $L_1 = L_2$, then what is
  - $L_1 - L_2$?
  - $L_2 - L_1$?
  - Their union?
  - If and only if:




- B is final state of $M_1$ and C is final state in $M_2$
  - Therefore [A,C] and [B,D] are final states in product automaton

# The Pumping Lemma for Regular Languages

*Number of states of DFA for L*

For every regular language L

There is an integer n, such that

For every string w in L of length $\geq$ n

We can write w = xyz such that:

1. $|xy| \leq n$.
2. $|y| > 0$.
3. For all $i \geq 0$, $xy^iz$ is in L.

*Labels along first cycle on path labeled w*

# Pumping Lemma as Adversarial Game

- 1: Player 1 (me) picks the language to be proved nonregular
  - ❖ Prove $L = \{ww^R \mid w \in \{a,b\}^*\}$ is not regular. (See Examples 4.7-4.13 in Linz.)
- 2. Player 2 picks *n*, but doesn't reveal to player 1 what n is; player 1 must devise a play for all possible *n's*
  - ❖ We don't need to/can't do anything.
- 3. Player 1 picks *s*, which may depend on n and which must be of <u>length at least *n*</u>
  - ➢ Assume $L$ is regular. Let $s = a^n b^1 b^1 a^n \in L$, i.e., $s = a^n b^1$; as well as $|s| \geq n$.

<u>Note</u>: Words in purple are the example wordings we use in this type of proofs.

# Pumping Lemma as Adversarial Game

- 4: Player 2 divides s into x,y,z obeying the constraints that are stipulated in the lemma: y is not empty and |xy| <= n.
  - Again, Player 2 does not tell Player 1 what x,y,z are; just that they obey the constraints. Meaning, we (P1) cannot choose *y=a^5*, *etc*.

➢ Then by the Pumping Lemma, $s$ can be divided into three parts $s = xyz$, such that $x = a^{\alpha}, y = a^{\beta}, z = a^{n-\alpha-\beta}b^1b^1a^n$, where $\beta \geq 1, (\alpha + \beta) \leq n$.

- 5. Player 1 "wins" by picking $k \in \mathbb{Z}^*$, which may be a number or function of *n,x,y*, and *z*, such that $xy^kz$ is not in L.

➢ Now, consider $k = 0$. Then the string after the pumping becomes $s' = xy^0z = xz = a^{n-\beta}b^1b^1a^n$. Note that since $\beta \geq 1$, there's no way for $s'$ to be in the form of a string followed by its reverse; hence $s' \notin L$. *Contradiction*. $\Longrightarrow L$ not regular.

<u>Note</u>: See Piazza @29 on $xy^kz$.

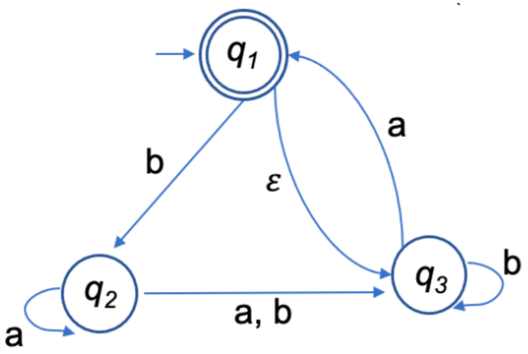
# Pumping Lemma Remarks

- Proving non-regular using Closure properties
  - In-Class last Thursday: Homomorphism
  - Other operators and their combinations
  - Known or easier to prove/disprove.
- Piazza @27: how do we know we need to choose …
  - **Trial and Error** and some eureka
  - $L = \{ww^R \mid w \in \{a,b\}^*\}$, if we'd chose $s = a^n a^n$, then for $s' = a^{n-\beta} a^n$, then adversary can just choose $\beta \geq 1$ to be of even length, such that $s' = w' w'^R$. So, choosing such an $s$ has no use for us.
    - Similar with constrains of $s$ and the value of $k$.
  - $L = \{a^n b^m \mid m \neq n,\ n, m \geq 1\}$, by choose $s = a^p b^{p+1}$ or $s = a^p b^{2p}$, can we find some integer $k$ such for $s' = xy^k z$, number of a's equals to number of b's. [Example 4.13 from Linz; Last problem in HW3]
  - $L = \{a^m b^n \mid m < n\}$, do we choose $s = a^{p-1} b^p$ or $s = a^p b^{p+1}$?

# Exam 1 Review

- Be careful: Precedence of operators and parenthesis (HW2)
  - $x + y^*$ vs $(x + y)^*$
  - $x^* + y^*$ vs $(x + y)^*$
  - Parenthesis > Kleene Star > Concatenation > Union
  - Parenthesis > Exponent > Multiplication > Addition

- NFA to DFA
  - $\hat{\delta}(q, x) = \hat{\delta}(q, \epsilon x \epsilon) = CL\Big(\delta\big(\hat{\delta}(q, \epsilon),\ x\big)\Big) = CL\big(\delta\big(CL(q),\ x\big)\big)$
    - $\hat{\delta}(q_1, 0) =$
    - $\hat{\delta}(q_1, 1) =$



# Q&A

- HW/Quiz/In-Class solutions all available.
- HW3 Solution available tonight after midnight.

- Q&A
- HW3 Problem 1: Given $N_1$, construct two machines
  - One from the method we discussed during lecture.
  - The other from the problem description.
  - Are they representing the same language?

- $L = \{a^n b^m \mid m \neq n,\ n, m \geq 1\}$