CS 3313
Foundations of Computing:

Exam 1 Review and Q&A

http://gw-cs3313.github.io

Definitions

- A language is regular iff it is accepted by a DFA/NFA or Reg. Expr.
  - To prove a language is regular, you have to provide a DFA/NFA or Reg.Expr.
- Closure Properties: applying set operations to reg. languages will result in a regular language
  - Reg Languages closed under: union, intersect, complement, etc.
- Decision Algorithm: a property is decidable iff there is an algorithm that can check if the property holds for the regular language
  - Ex: are two DFAs equivalent, is L infinite, etc.
- Non-regularity of languages – prove using pumping lemma

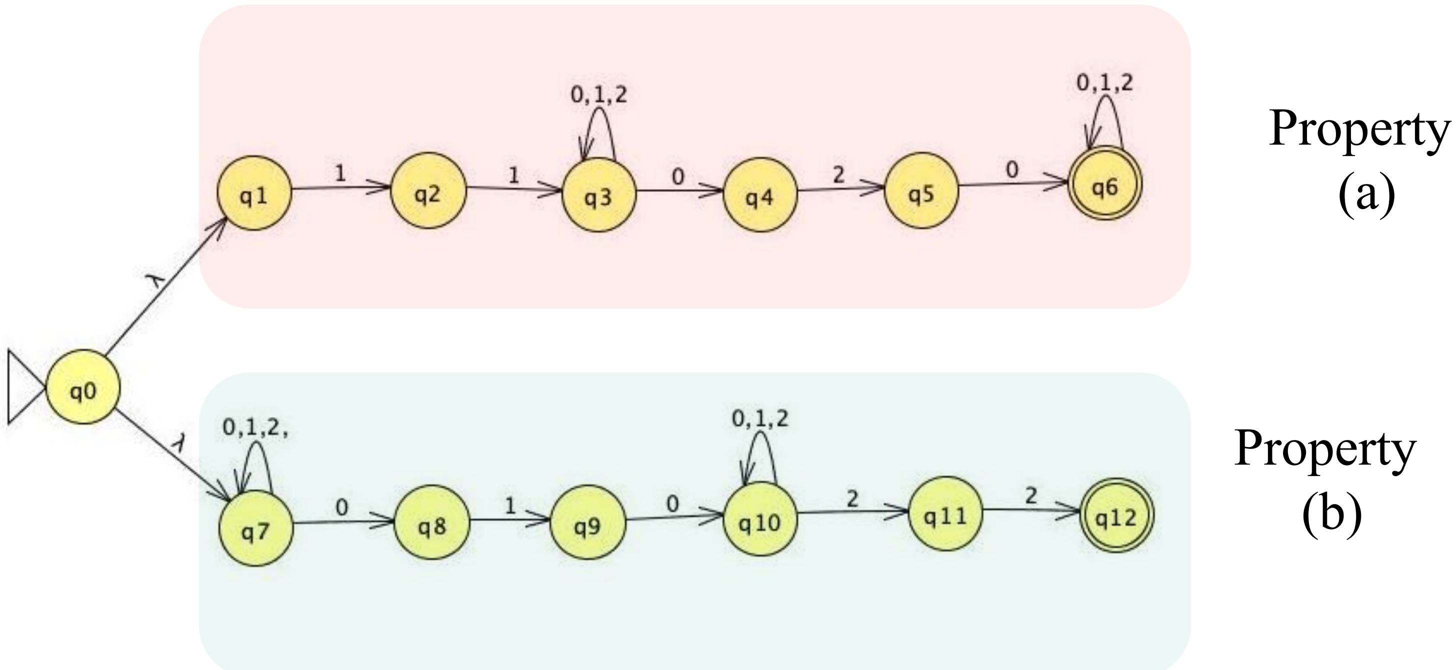
Review via Exercises…Question 1

- Give a finite state automata (DFA or NFA) and a regular expression for:

  $L = \{ w \mid w \ \{0,1,2\}^*$ *and (a) w starts with 11 and contains substring 020 or (b) w contains substring 010 and ends with 22* $\}$

Answer Exercise 1

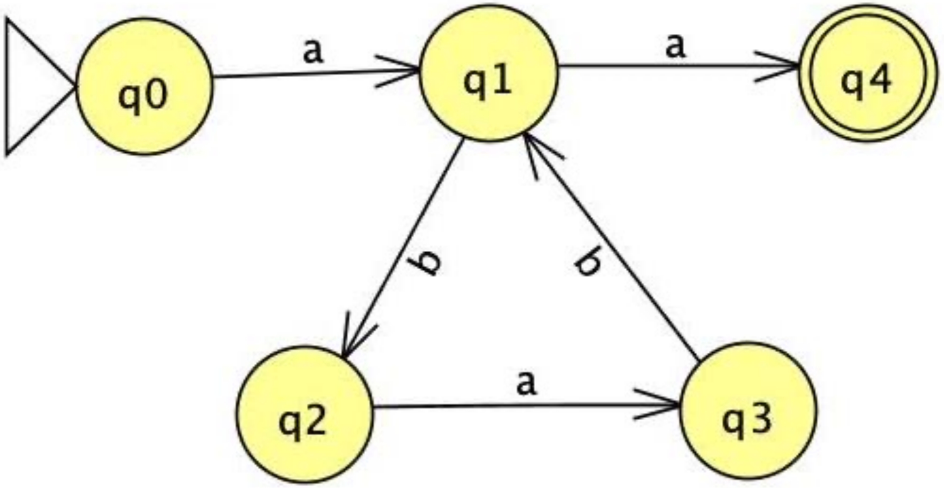
- Easier to design an NFA – break up the design into two NFAs for property (a) and property (b). Each has two NFAs for the concatenation property.



- Reg. Expr. : substring x in a regular expression is $(0+1+2)^* \, x \, (0+1+2)^*$
  - property (a) says string 11 followed by $(0+1+2)^* \, 020 \, (0+1+2)^*$ and property (b) says $(0+1+2)^* \, 010 \, (0+1+2)^* \, 22$
- *Ans.:* $(11 \, (0+1+2)^* \, 020 \, (0+1+2)^*) + ((0+1+2)^* \, 010 \, (0+1+2)^* \, 22)$

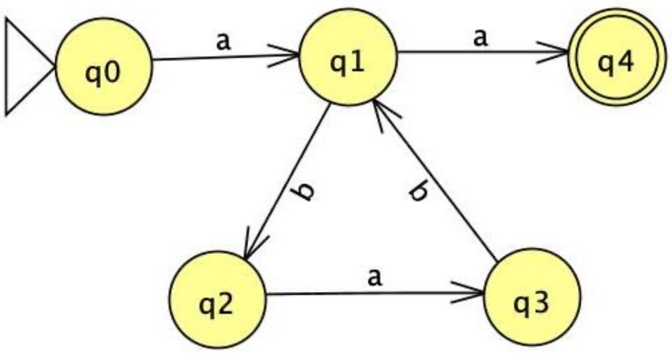
Review Exercise 2: DFA to Regular Expression

- Give a regular expression for the language accepted by this DFA:

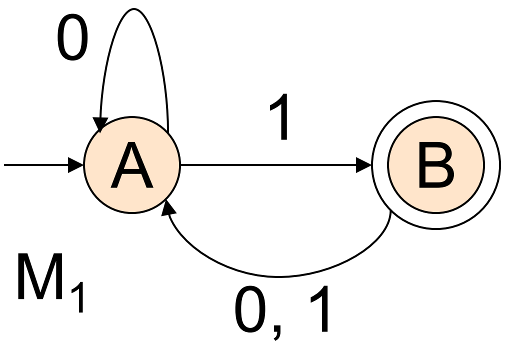


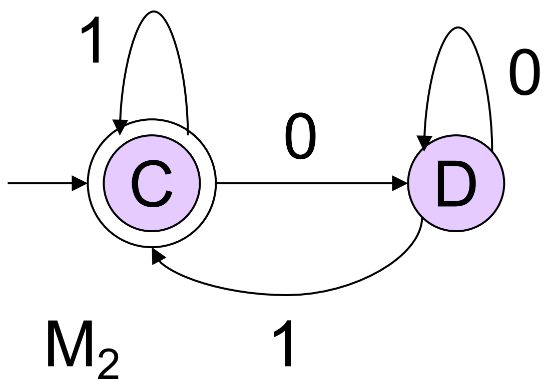
Answer Exercise 2: DFA to Regular Expression

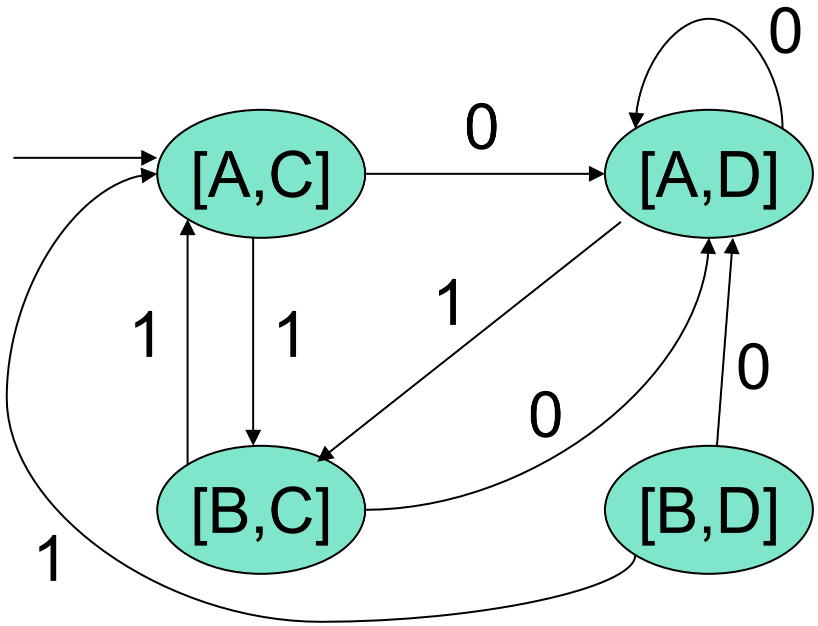
- Give a regular for the language accepted by this DFA:
- Path without a cycle from start to final state q_4 is aa
  - Vertices visited q_0 q_1 q_2
- Cycle from q_1 to q_1 labelled bab
- Therefore we can have paths q_0 q_1(q_1 q_1)* q_1 q_2
- Reg. Expr. = a (bab)* a



Product DFA: Example





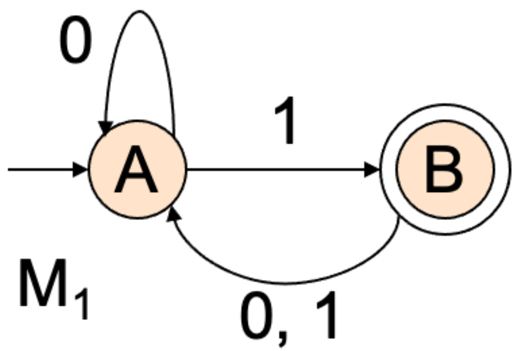


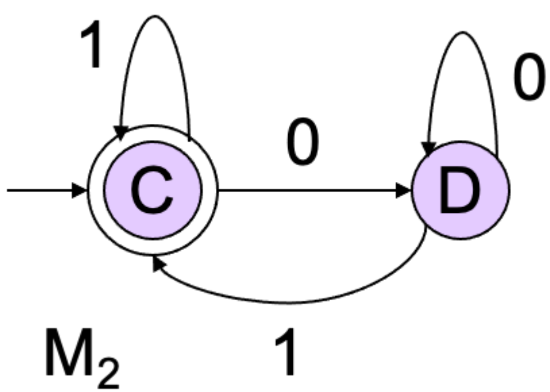
- Product DFA has set of states $Q \times R$
  - Not all are reachable.
- Start state = $[q_0, r_0]$
  - Start state in each machine.
- Transitions: $\delta([q,r], a) = [\delta_1(q,a), \delta_2(r,a)]$
  - $\delta([A,C], 1) = [\delta_1(A,1), \delta_2(C,1)] = [B,C]$
  - $\delta([A,C], 0) = [\delta_1(A,0),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

Review Exercise 3

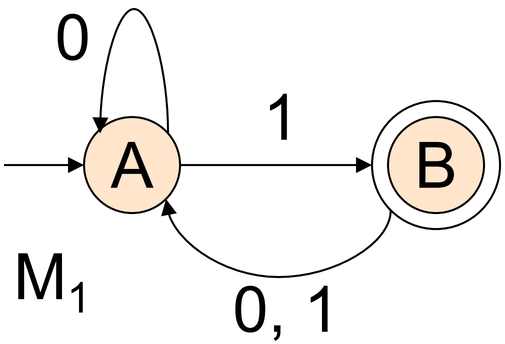
- For the two DFAs shown below, provide a DFA that accepts

$$L = L(M_2) - L(M_1)$$

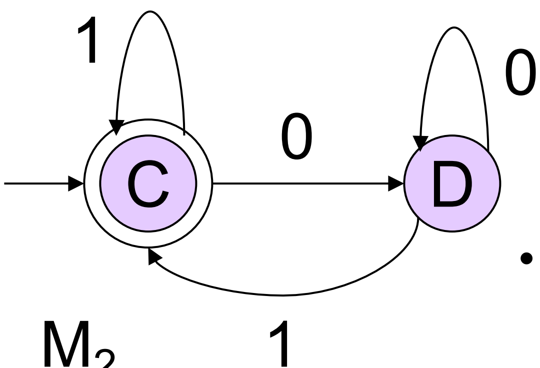




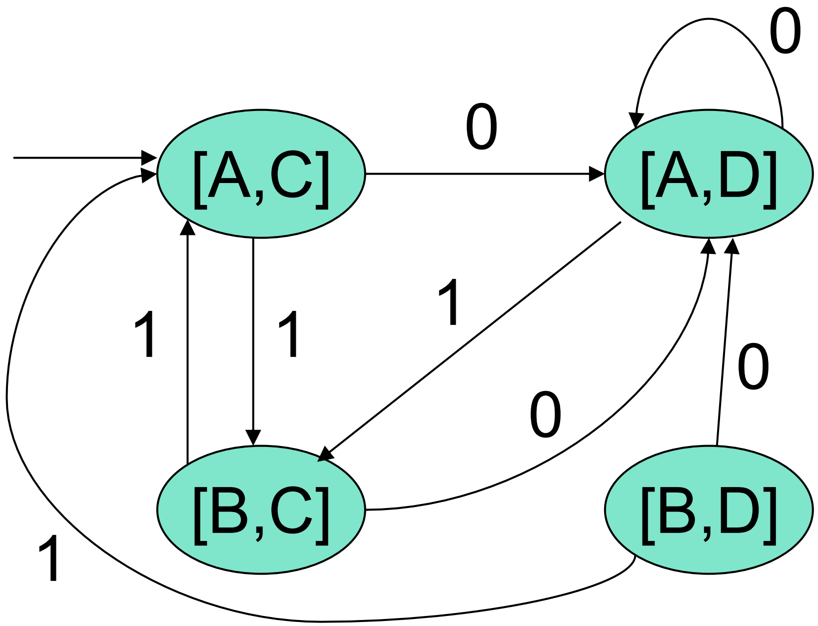
Answer – Exercise 3
Product DFA: Final States



M_1



M_2



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

Examples: Applying closure properties

- 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 $\cup$ L2 = {w | w has a's followed by b's or w has even length} is regular
- L1 $\cap$ L3 = { w | w has odd number of a's followed by even number of b's} is regular
- $\overline{L1}$ = {w| w does not strictly have a's followed by b's } is regular
- L = L1 $\cap$ $\overline{L3}$ = {w| w has a's followed by b's and not (a is odd and b is even) } is regular

Applying Closure Properties

- Closure property theorems state: If "regular" then "operations" result in regular
  - Logic: If P then Q (P implies Q) whose contrapositive is NOT Q implies NOT P
- Can use closure properties to simplify proofs/logic
- Prove $L=\{ a^j\ b^k \mid j \text{ is not equal to } k \}$ is not regular
  - From Theorem on closure on complement of reg. lang

    If L is regular then L' is regular.

  <u>Note</u>: What is L' ?

  - From theorem on closure on intersection with reg. lang.

    If L' is regular then $L' \cap\ a^*b^* = \{ a^j\ b^k \mid j=k\}$ is regular

    contradiction. Therefore L is not regular.

Review Exercise 4

- Prove or disprove: If L is not regular then $h(L)$ is not regular for a homomorphism h.

Answer Exercise 4

- Prove or disprove: If *L* is not regular then *h(L)* is not regular for a homomorphism *h*.
- Note that P implies Q is not equivalent to NOT P implies NOT Q
- Theorem says "If L is regular then h(L) is regular, for a homomorphism h"
- Proof by counterexample:
  - Let $L = \{ a^k b^k \mid k >= 0\}$ we have proved L is not regular.
  - Let h: {a,b} => {0,1}* and h(a)=0 and h(b)=0
  - $h(L) = \{0^k 0^k \} = \{ 0^{2k} \}$ which is given by the reg expr (00)*
  - Therefore h(L) is regular.
  - Therefore the statement is false.

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 ?

Review Exercise 5

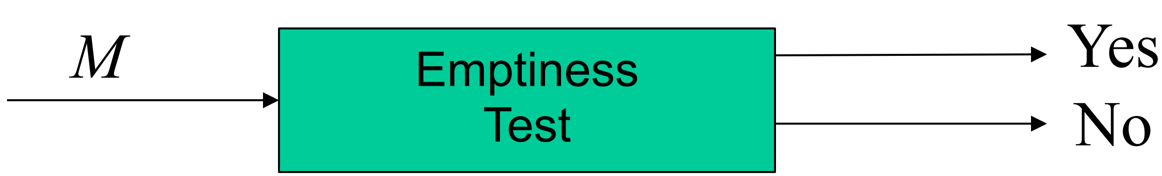
- For any two regular languages $L_1 = L(M_1)$ and $L_2 = L(M_2)$, prove that checking property $(L_1 - L_2) = L_1$ is decidable.

Answer Review Exercise 5

- For any two regular languages $L_1 = L(M_1)$ and $L_2 = L(M_2)$, prove that checking property $(L_1 - L_2) = L_1$ is decidable.
- Proof:
- Observation: $(L_1 - L_2) = L_1$ is equivalent to saying $L_1 \cap L_2 = \emptyset$.
- We can construct a product DFA M that accepts the intersection of the two languages
- Next, we have an algorithm that tests for emptiness of a regular language

Send M as input to emptiness testing algorithm:

$(L_1 - L_2) = L_1$ iff Algorithm answers Yes

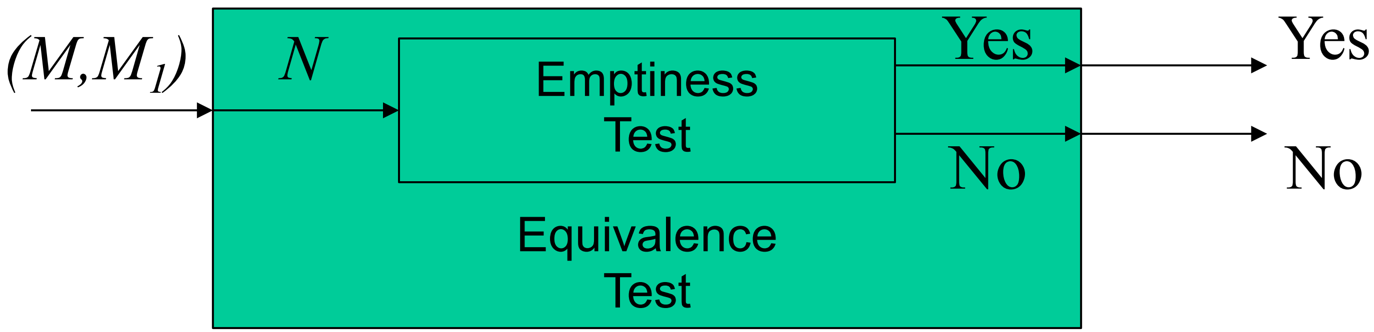


Answer Review Exercise 5

- For any two regular languages $L_1 = L(M_1)$ and $L_2 = L(M_2)$, prove that checking property $(L_1 - L_2) = L_1$ is decidable.
- Proof (w/o the observation):
- We can construct a product DFA M that accepts $L_1 - L_2$.
- Next, we have an algorithm that tests for equivalence of two regular languages (that construct prod. DFA N for symm. sum)

Send M and M_1 as input to equivalence testing algorithm:

$(L_1 - L_2) = L_1$ iff Algorithm answers Yes



How to prove a language is not regular…
The Pumping Lemma for Regular Languages

For every regular language L

There is an integer n, such that (note; you cannot fix n)

For every string w in L of length $\geq n$ *(you can choose w)*

We can write $w = xyz$ such that:

1. $|xy| \leq n$ *(this lets you focus on pumping within first n symbols)*
2. $|y| > 0$ *(y cannot be empty)*
3. *For all* $i \geq 0$, xy^iz *is in* L. *(to get contradiction find one value of i where pumped string is not in L)*

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, w 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*, which may be an integer 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.

Pumping Lemma Remarks

- Proving non-regular using closure properties
  - In-Class example using: intersection, Homomorphism
    - Other operators and their combinations
  - Known or easier to prove/disprove.
- how do we know what string 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.
  - $L = \{a^m b^n \mid m < n\}$, do we choose $s = a^{p-1} b^p$ or $s = a^p b^{p+1}$?
  - $L = \{a^n b^m \mid m \neq n,\ \ n, m \geq 1\}$, 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]

Pumping Lemma Remarks

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

- Need to choose $s = a^{p!} b^{(p+1)!}$ and choose a positive integer i such that

  $$p! - \beta + i\beta = (p+1)! = p!\,(p+1) = pp! + p!.$$

Questions ?