STALGCM NOTES FOR LE2 

By: Roemer Caliboso

Special Thanks to Anthony Baybayon and Kenneth Go and PTS

IMPORTANT REMINDERS:

Reminders for STALGCM Exams:

- TLDR: FOLLOW INSTRUCTIONS
- **DONT** FORGET TO PUT START STATE
- **DONT** FORGET TO PUT START STATE
- **DONT** FORGET TO PUT FINAL STATE
- **DONT** FORGET TO PUT FINAL STATE
- **DONT** EXCEED STATE LIMIT OR PRODUCTION RULES OR NONTERMINAL
- **DONT** FORGET TO CONVERT NFA TO DFA WHEN NECESSARY IN CLOSURES
- **ONLY 1** FINAL STATE AND 1 START STATE IN PDA
- **INFORMAL PROOF** CAN BE USED FOR PUMPING LEMMA
- DON'T POP AND PUSH IN THE SAME TRANSITION IN PDA
- NO λ ALLOWED OTHER THAN THE START SYMBOL (Σ)
- ALWAYS START WITH (for all $p \in \mathbb{Z}^+$) and END WITH (therefore L is not regular) for PUMPING LEMMA PROOF
- PDA STRING CAN ONLY BE ACCEPTED IF EMPTY STACK (LAST READ IS Z) AND IF INPUT IS EXHAUSTED
- Σ CANNOT BE USED IN PRODUCTION
- MARCH 12, 2025 | 1:00PM - 3:00PM
- BRING LARGE TEST BOOKLET
- 5 ITEMS, 70 POINTS TOTAL

NOTE THAT SOME OF THESE REMINDERS ARE SPECIFIC TO SR AUSTIN :)

if were wrong,,,,,, sorry

"$\mathcal{H}$*ello*...? *Delta*? ...Where are you?  "
- A. Fernandez **(2025)**

[1] Closure Properties of Regular Languages

Recap on Regular Languages

- A regular language is a set of strings that can be defined by a regular expression.
- A regular language is "regular" if it can be modeled by a DFA.or its equivalents.

Closure Operations

- A set S is closed under a binary operation $\oplus$ if and only if applying it on two elements in S produces a result that is still an element of S. That is $(A, B \in S),\ (A \oplus B) \in S$.
  - Extension to unary: A set S is closed under the unary operator f if $A \in S$, $f(A) \in S$.
- Operations on Regular Languages that are Closed:
  - Complement of a Language $\overline{L}$
  - Union $L_1 \cup L_2$
  - Intersection $L_1 \cap L_2$
  - Difference $L_1 - L_2$
  - Concatenation $L_1 \cdot L_2$
  - Kleene Closure L^*
  - Reversal of a Language L^R

NOTE: The procedures here take into account that you have a DFA that describes your language.

1. Complement $\overline{L}$

- If L is a language $\overline{L}$ is the language accepting all strings rejected by L and rejecting all strings accepted by L.

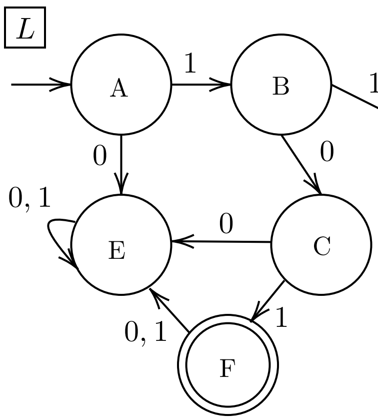
Formal Procedure:

Given a machine $M = (Q, \Sigma, \delta, q_I, F)$ be the DFA that recognizes a language L.
$\overline{M}$ is the machine that recognizes $\overline{L}$ where:
$M = (Q, \Sigma, \delta, q_I, F')$ where $F' = Q - F$.

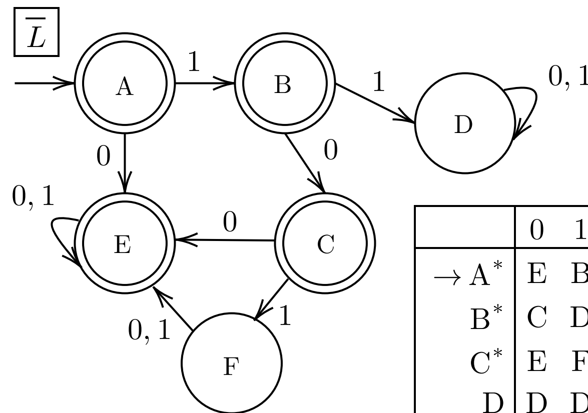
Informal Procedure:
Turn all final states into non-final states, and all non-final states to be final.

Given the language $L = \Big\{\omega \in 0,1 \mid \omega = 1\big(01 \cup 1(0 \cup 1)^*\big)\Big\}$

Example:



	0	1
→ A	E	B
B	C	D
C	E	F
D*	D	D
E	E	E
F*	E	E



	0	1
→ A*	E	B
B*	C	D
C*	E	F
D	D	D
E*	E	E
F	E	E

2. Union $\cup$, Intersection $\cap$, Difference $-$

Computation of $L_1 \cup L_2$, $L_1 \cap L_2$, or $L_1 - L_2$ have

Product Machine:

M_1 and M_2

$M_1 = (Q_1, \Sigma, \delta_1, q_{I1}, F_1)$

$M_2 = (Q_2, \Sigma, \delta_2, q_{I2}, F_2)$

$M_P = M_1 \times M_2 = (Q, \Sigma, \delta, q_I, F)$

$Q = Q_1 \times Q_2$

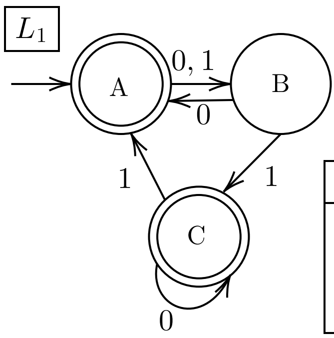
$\delta : Q \times \Sigma \mapsto Q$

$\forall q_1 \forall q_2 \forall s (\delta(q_1, q_2), s) = (\delta_1(q_1, s), \delta_2(q_2, s)), q_1 \in Q_1, q_2 \in Q_2, s \in \Sigma$

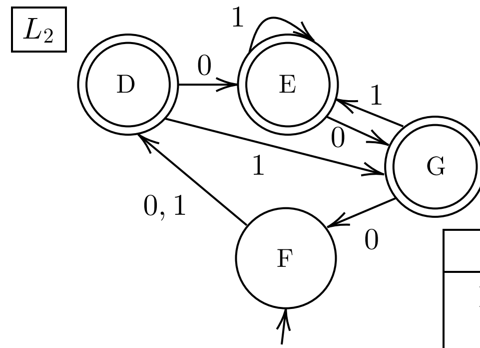
$q_I = (q_{I1}, q_{I2})$

F varies

Note that this definition may not necessarily provide a connected or reduced machine.



	0	1
$\to$ A*	B	B
B	A	C
C*	C	A



	0	1
D*	E	G
E*	G	E
$\to$ F	D	D
G*	F	E

To form the product machine between L_1 and L_2 we start with a machine containing the Cartesian product of both machines:
The start state of the machine will be the union of the start state of both machines.

For each state and for each transition, the state travelled to is the union of the states travelled to in the original machine.
The final state/s of each machine varies based on the operation.

	0	1
AD		
AE		
$\to$ AF		
AG		
BD		
BE		
BF		
BG		
CD		
CE		
CF		
CG		

	0	1
AD	BE	BG
AE		
$\to$ AF		
AG		
BD		
BE	AG	CE
BF		
BG		
CD		
CE		
CF		
CG		

	0	1
$\to$ A*	B	B
B	A	C
C*	C	A

L_1

	0	1
D*	E	G
E*	G	E
$\to$ F	D	D
G*	F	E

L_2

	0	1
AD	B	B
AE	B	B
$\to$ AF	B	B
AG	B	B
BD	A	C
BE	A	C
BF	A	C
BG	A	C
CD	C	A
CE	C	A
CF	C	A
CG	C	A

	0	1
AD	BE	BG
AE	BG	BE
$\to$ AF	BD	BD
AG	BF	BE
BD	AE	CG
BE	AG	CE
BF	AD	CD
BG	AF	CE
CD	CE	AG
CE	CG	AE
CF	CD	AD
CG	CF	AE

For difference ($L_1 - L_2$),
Final states:
All states containing a final state in L_1 but not containing a final state L_2

	0	1
AD	BE	BG
AE	BG	BE
$\to$ AF*	BD	BD
AG	BF	BE
BD	AE	CG
BE	AG	CE
BF	AD	CD
BG	AF	CE
CD	CE	AG
CE	CG	AE
CF*	CD	AD
CG	CF	AE

For union ($L_1 \cup L_2$),
Final states:
Final states in EITHER L_1 OR L_2

	0	1
AD*	BE	BG
AE*	BG	BE
$\to$ AF*	BD	BD
AG*	BF	BE
BD*	AE	CG
BE*	AG	CE
BF	AD	CD
BG*	AF	CE
CD*	CE	AG
CE*	CG	AE
CF*	CD	AD
CG*	CF	AE

For intersection ($L_1 \cap L_2$),
Final states:
Final states in BOTH L_1 AND L_2

	0	1
AD*	BE	BG
AE*	BG	BE
$\to$ AF	BD	BD
AG*	BF	BE
BD	AE	CG
BE	AG	CE
BF	AD	CD
BG	AF	CE
CD*	CE	AG
CE*	CG	AE
CF	CD	AD
CG*	CF	AE

L_1

	0	1
$\to$ A*	B	B
B	A	C
C*	C	A

L_2

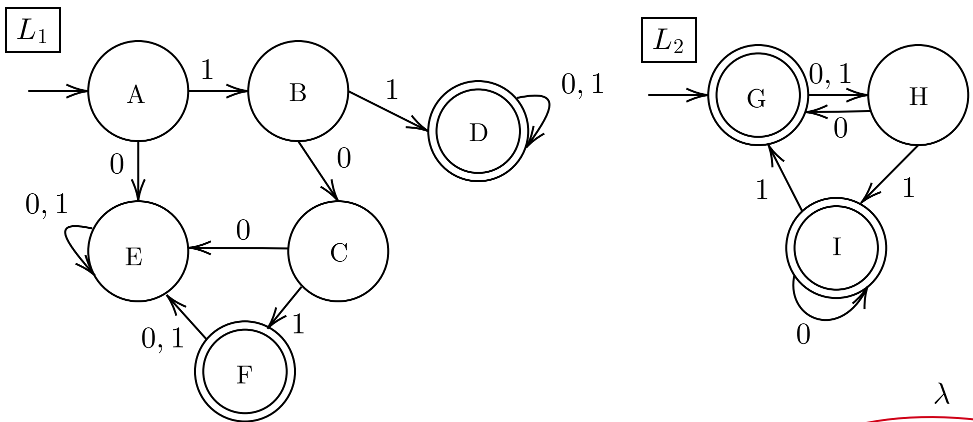
	0	1
D*	E	G
E*	G	E
$\to$ F	D	D
G*	F	E

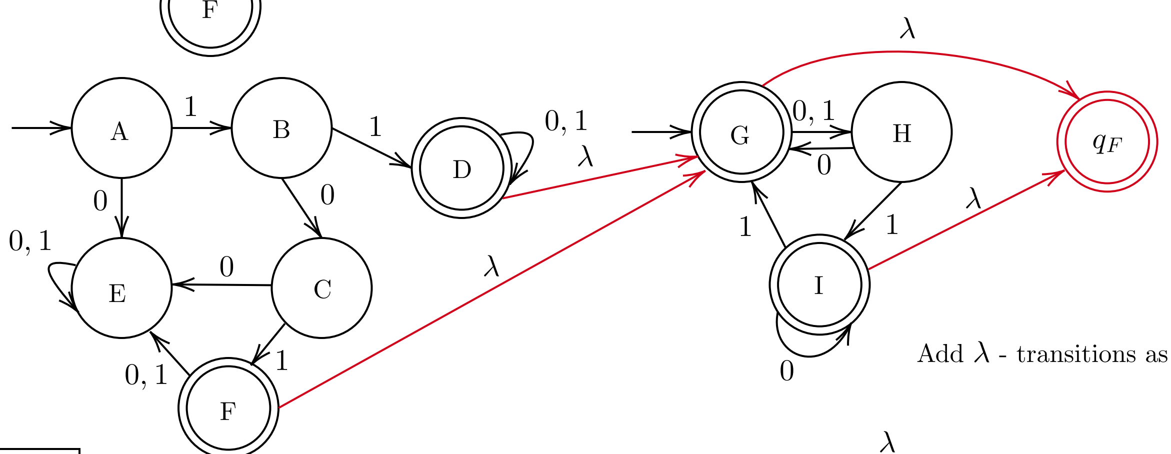
3. Concatenation $L_1 \cdot L_2$

The concatenation $L_1 \cdot L_2$ is regular.

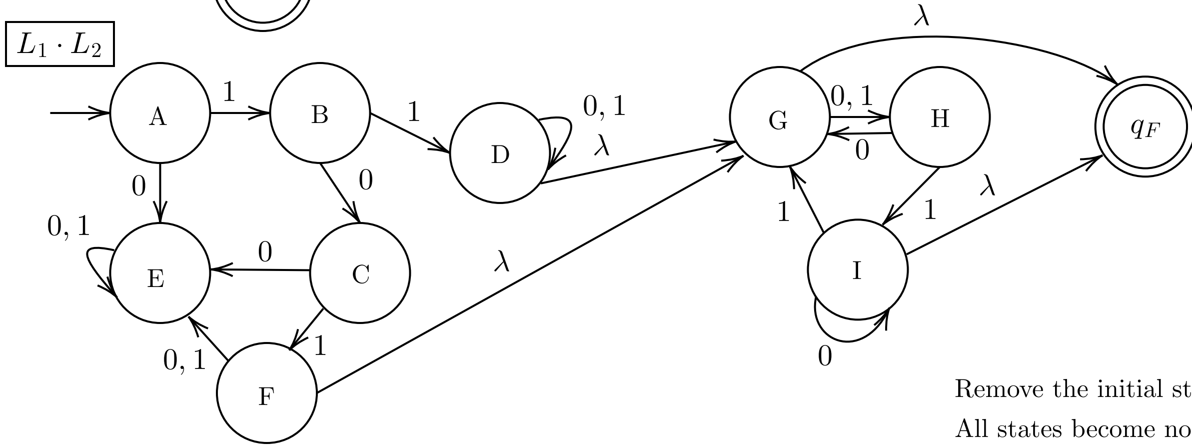
Informal Procedure:

1. Assume all states are disjoint.
2. All final states in L_1 and L_2 become rejecting.
3. Add a new final state q_F.
4. Add a λ - transition from each final state of M_1 to the start state of M_2.
5. Add a λ - transition from each final state of M_2 to the start state of q_F.
6. [OPTIONAL] Convert to DFA (this is optional because λ - NFAs are equivalent to some DFAs)





Add λ - transitions as follows.



Remove the initial state of M_2

All states become non-accepting except q_F.

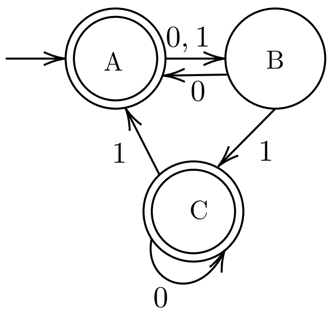
4. Kleene Closure L^*

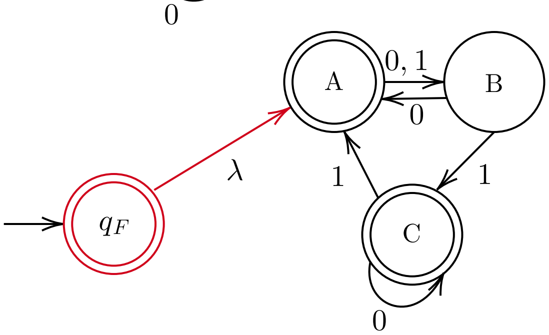
Kleene Closure $L^\star$ is regular.

Informal Procedure:

1. Add a new final state q_F. This becomes the an initial and final state.
2. Add a λ - transition from q_F to the initial state of L. The original initial state does not become initial anymore.
3. Add a λ - transition from each final state of L to q_F. The original accepting states become non-accepting.
4. [OPTIONAL] Convert to DFA (this is optional because λ - NFAs are equivalent to some DFAs)

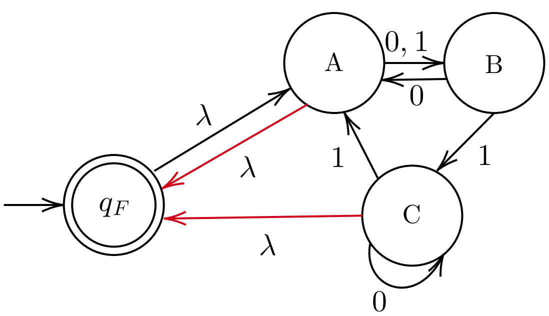
$\boxed{L}$





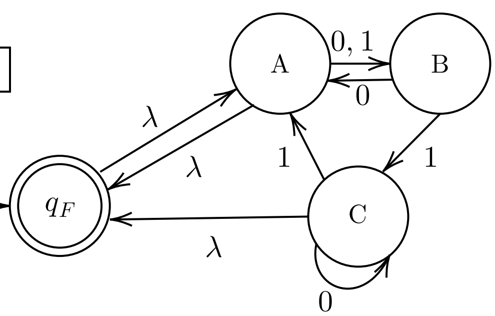
Add a new final state q_F. This becomes the an initial and final state.

Add a λ - transition from q_F to the initial state of L.
The original initial state does not become initial anymore.



Add a λ - transition from each final state of L to q_F.
The original accepting states become non-accepting.

$\boxed{L^*}$

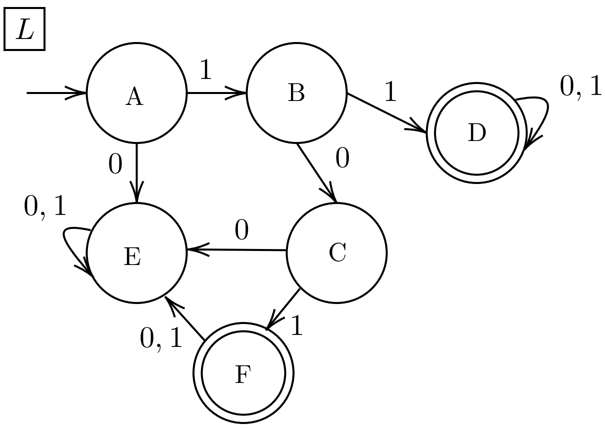


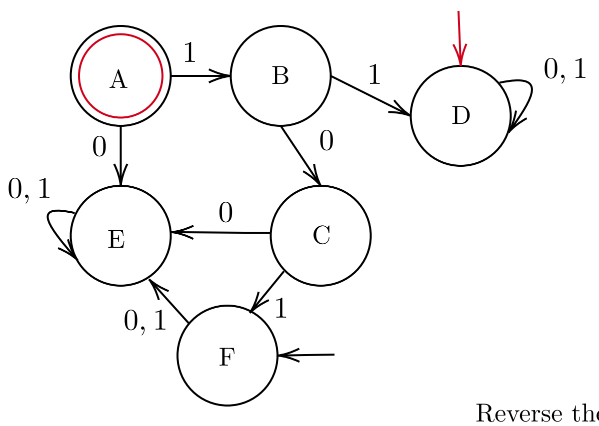
5. Reverse L^R

The reverse of a machine L^R is regular.

Informal Procedure:

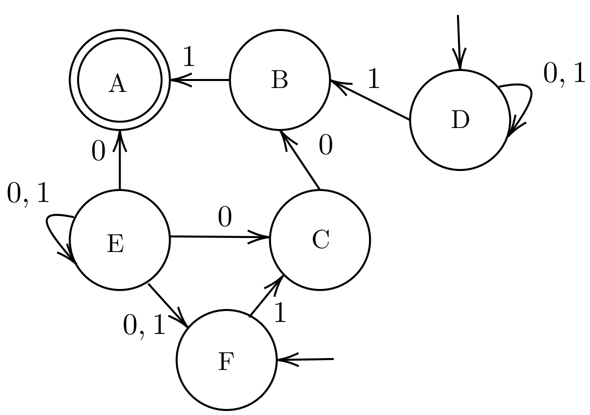
1. All initial states become final and all final states become initial states. (A state that is both final and initial remains final and initial)
2. Reverse the direction of all transitions. (Given L has $\delta(a, s) = b$, L^R will have $\delta(b, s) = a$).

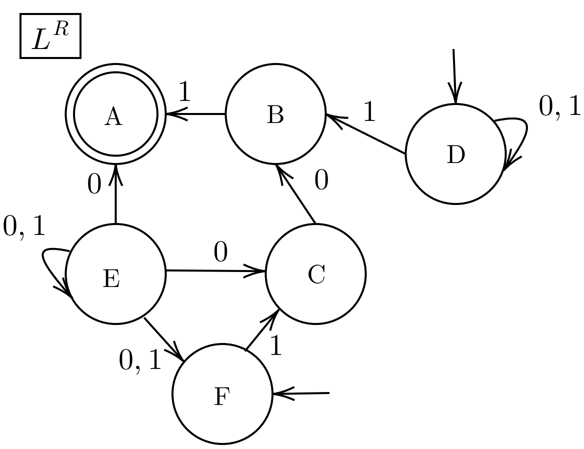




All initial states become final and all final states become initial states.

Reverse the direction of all transitions.





I. Consider the following DFAs:

L_1

	0	1
$\rightarrow A$	A	B
B^*	B	A

L_2

	0	1
$\rightarrow C$	E	C
D^*	D	E
E^*	D	C

L_3

	0	1
$\rightarrow F^*$	G	F
G	F	G

Find the following:

$$\overline{\left[(L_1 \cup L_3) - L_2^R\right]}^*$$

$$\overline{\left[(L_1 \cup L_3) - L_2^R\right]}^*$$

$(L_1 \cup L_3)$

	0	1
$\rightarrow AF$	AG	BF
AG	AF	BG
BF^*	BG	AF
BG^*	BF	AG

$\left(L_2^R\right)$

	0	1
C^*		C, E
$\rightarrow D$	D, E	
$\rightarrow E$	C	D

$\left(L_2^R\right)$

	0	1
$\rightarrow DE$	CDE	D
CDE^*	CDE	CDE
D	DE	X
X	X	X

$\left(L_2^R\right)$

	0	1
$\rightarrow H$	I	D
I^*	I	I
D	H	X
X	X	X

$(L_1 \cup L_3) - L_2^R$

	0	1
$\rightarrow AFH$	AGI	BFD
AFI	AGI	BFI
AFD	AGH	BFX
AFX	AGX	BFX
AGH	AFI	BGD
AGI	AFI	BGI
AGD	AFH	BGX
AGX	AFX	BGX
BFH^*	BGI	AFD
BFI	BGI	AFI
BFD^*	BGH	AFX
BFX^*	BGX	AFX
BGH^*	BFI	AGD
BGI	BFI	AGI
BGD^*	BFH	AGX
BGX^*	BFX	AGX

$\overline{(L_1 \cup L_3) - L_2^R}$

	0	1
$\rightarrow AFH^*$	AGI	BFD
AFI^*	AGI	BFI
AFD^*	AGH	BFX
AFX^*	AGX	BFX
AGH^*	AFI	BGD
AGI^*	AFI	BGI
AGD^*	AFH	BGX
AGX^*	AFX	BGX
BFH	BGI	AFD
BFI^*	BGI	AFI
BFD	BGH	AFX
BFX	BGX	AFX
BGH	BFI	AGD
BGI^*	BFI	AGI
BGD	BFH	AGX
BGX	BFX	AGX

$\overline{(L_1 \cup L_3) - L_2^R}^*$

	0	1	λ
$\rightarrow {q_F}^*$			AFH
AFH	AGI	BFD	q_F
AFI	AGI	BFI	q_F
AFD	AGH	BFX	q_F
AFX	AGX	BFX	q_F
AGH	AFI	BGD	q_F
AGI	AFI	BGI	q_F
AGD	AFH	BGX	q_F
AGX	AFX	BGX	q_F
BFH	BGI	AFD	
BFI	BGI	AFI	q_F
BFD	BGH	AFX	
BFX	BGX	AFX	
BGH	BFI	AGD	
BGI	BFI	AGI	q_F
BGD	BFH	AGX	
BGX	BFX	AGX	

[2] Pumping Lemma

The pumping lemma for regular languages states the following:
If L is a language.
$\big(L \text{ is regular language }\big) \rightarrow \exists p \forall \omega \big(|\omega| \geq p \rightarrow \exists x \exists y \exists z \big(\omega = xyz \wedge |xy| \leq p \wedge |y| > 0 \wedge xy^{*}z \in L\big)\big), p \in Z^{+}, \omega \in L$
If L is a regular language.

Translation:
There exists a positive integer p, where for ANY string ω, if ω is AT LEAST p symbols long, there exists a splitting of ω into xyz WHERE:

x is the string that leads to the pumping string y
y is the pumping string. (MUST BE NONEMPTY)
z is the set of symbols that lead it to an accepting state

$\lvert xy \rvert \leq p$	The concatenation of x and y does not exceed p in length)
$\lvert y \rvert > 0$	The pumping string is not empty.
$xy^{*}z \in L$	Pumping the string y (or repeating it 0 to many times) still leads it to be a member of L.

The contrapositive of the pumping lemma can be used to show that a language is NOT regular. Note that we cannot use the contrapositive to prove that a language IS regular just that its not.

$$\forall p \exists \omega (|\omega| \geq p \wedge \forall x \forall y \forall z \big((\omega = xyz \wedge |xy| \leq p \wedge |y| > 0) \rightarrow xy^{*}z \notin L \ \big) \rightarrow \big(L \text{ is NOT a regular language}\big), p \in Z^{+}, \omega \in L$$

Translation:
For all positive integers p, there will exist a counterexample ω where REGARDLESS of how you divide ω into $\omega = xyz$:

$\lvert xy \rvert \leq p$	The concatenation of x and y does not exceed p in length)
$\lvert y \rvert > 0$	The pumping string is not empty.
$xy^{*}z \in L$	Pumping the string y (or repeating it 0 to many times) may lead to a string L

Prove $L = \left\{\omega \in \{0,1\}^* \mid \omega = 0^n 1^n, \text{where } n \geq 1\right\}$ is non regular.

Our strategy is MAINLY to get counterexample where we can isolate the pumping string into $|xy| \leq p$.

INFORMAL:
We first need to find a ω that will always break if we pump the symbol:

For all positive integers p, we generate $0^p 1^p \in L$ (THIS LINE IS IMPORTANT)
Because $|xy| \leq p$, we will always pump 1 to p 0s.

Doing this may lead to strings in the form of $0^{kp}1^p$ where $k \in \mathbb{Z}$ (formality)
This will lead to pumping strings of 0 that will cause there to be more 0s than 1s leading to a string not in L.

Therefore L is not a regular language. (THIS LINE IS IMPORTANT)

FORMAL:
For all $p \in Z^+$ we can generate a string $0^p 1^p \in L$.
Let:
$x = 0^\alpha$
$y = 0^{(p-\alpha)}$
$z = 1^p$ where $\alpha < p$
In this case xy^*z will generate $L' = \left\{\omega \in \{0,1\}^* \mid \omega = 0^{kp-(k-1)\alpha}1^p, k \geq 0\right\}$[1] .
Since it is not true $kp - (k-1)\alpha = p$ for all choices of p, L is not a regular language.

Note the pattern[1]: (no need to put this in the exam)
Given xyz: $x = 0^\alpha$, $y = 0^{p-\alpha}$, z
$n = 0,\ 0^\alpha z$
$n = 1,\ 0^\alpha 0^{(p-\alpha)} z = 0^p z$
$n = 2,\ 0^\alpha 0^{2(p-\alpha)} z = 0^{2p-\alpha} z$
$n = 3,\ 0^\alpha 0^{3(p-\alpha)} z = 0^{3p-2\alpha} z$
$n = k,\ = 0^{kp-(k-1)\alpha} z$

Example: Prove that the following languages are not regular.

1. $L = \left\{\omega \in \{0,1\}^* \mid \omega = 0^{2n}1^n, \text{ where } n \geq 1\right\}$

INFORMAL:

For all positive integers p, we generate $0^p0^p1^p \in L$
Because $|xy| \leq p$, we will always pump 1 to p 0s. ()

Doing this may lead to strings in the form of $0^{kp}0^p1^p$ where $k \in \mathbb{Z}$
This will lead to pumping strings of 0 that will cause that will violate $0^{2n}1^n, n \geq 1$ leading to a string not in L.

Therefore L is not a regular language.

FORMAL:

For any positive integer p we can generate the string $0^p0^p1^p \in L$:
Let:
$x = 0^\alpha$
$y = 0^{p-\alpha}$
$z = 0^p1^p$

Where $p > \alpha$:
In this case, xy^*z will generate $L' = \left\{\omega \in \{0,1\}^* \mid \omega = 0^{kp-(k-1)\alpha}0^p1^p, k \geq 0\right\}$.
It is not true that $kp - (k-1)\alpha = p$ for all $k \geq 0$.

So the language generated by the expression is non-regular.

2. $L = \left\{\omega \in \{0,1\}^* \mid \omega = 0^{2n}1^n, \text{where } n \geq 1\right\}$ $0^m1^{m+n}0^n$ $m, n > 0$

INFORMAL:

For all positive integers p, we generate $0^p1^{p+1}0 \in L$

Because $|xy| \leq p$, we will always pump 1 to p 0s.

Doing this may lead to strings in the form of $0^{kp}1^{p+1}0$ where $k \in \mathbb{Z}$

This will lead to pumping strings of 0 that will cause that will violate $0^m1^{m+n}0^n$ $m, n \geq 1$ where the number of 0s are not equal to the number of 1s. This leads to strings not in L.

Therefore L is not a regular language.

FORMAL

For any positive integer p we can generate the string $0^p1^{p+1}0$:

Let:

$\left(0^p1^{p+1}0\right)$

$x = 0^\alpha$

$y = 0^{p-\alpha}$

$z = 1^{p+1}0$

Where $p > \alpha$:

In this case, xy^*z will generate $L' = \left\{\omega \in \{0,1\}^* \mid \omega = 0^{kp-(k-1)\alpha}1^{p+1}0, k \geq 0\right\}$.

It is not true that $(kp - (k-1)\alpha) + 1 = p + 1$ for all $k \geq 0$.

So the language generated by the expression is non-regular.

3. $L = \left\{\omega \in \{0,1\}^* \mid \omega = 0^n 1^m, \text{ where } n > m\right\}$

INFORMAL:

For all positive integers p, we generate $0^p 1^{p-1} \in L$

Because $|xy| \leq p$, we will always pump 1 to p 0s.

Doing this may lead to strings in the form of $0^{kp} 1^{p+1} 0$ where $k \in \mathbb{Z}$

This will lead to pumping strings of 0 0 times that will cause that will violate $0^n 1^m$, where $n > m$ where the number of 0s are less than or equal to the number of 1s. This leads to strings not in L.

Therefore L is not a regular language.

FORMAL

For any positive integer p we can generate the string $0^p 1^{p-1} \in L$:

Let:

$\left(0^p 1^{p-1}\right)$,

$x = 0^\alpha$

$y = 0^{p-\alpha}$

$z = 1^{p-1}$

Where $p > \alpha$ and $\alpha \geq 0$:

In this case, xy^*z will generate $L' = \left\{\omega \in \{0,1\}^* \mid \omega = 0^{kp-(k-1)\alpha} 1^{p-1}, k \geq 0\right\}$.

It is not true that $kp - (k-1)\alpha > p - 1$ for all $k \geq 0$, because when $k = 0$, $\alpha > p - 1$ but $p > a$.

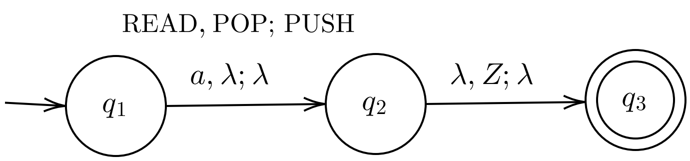
So the language generated by the expression is non-regular.

[3] Pushdown Automata

A pushdown accepter (PDA) is defined a 7 - tuple:
$M = (Q, \Sigma, \Gamma, \delta, q_I, z_I, q_F)$

- Q -- finite set of states
- $\Sigma -$ finite input alphabet
- $\Gamma -$ finite stack alphabet
- $\delta : Q \times \Sigma_\lambda \times \Gamma \mapsto \mathcal{P}(Q \times \Gamma_\lambda)$ is the transition function
- $q_I -$ initial state, $q_I \in Q$
- $z_I -$ initial stack symbol, $z_I \in \Gamma$
- $q_F -$ final state, $q_F \in Q$

The OVERALL state of a machine is $Q \times \Gamma^*$ where we a state and the contents of stack ϕ. PDAs are non-deterministic.



Note that for a machine to successfully read, ALL input must have been read as well symbol at the bottom of the stack Z.

Examples:

$\left(\omega\omega^R\right)$

$M = (Q, \Sigma, \Gamma, \delta, q_I, z_I, q_F)$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{X, Y, Z\}$ $\boxed{\delta(q, s, (pop)) = (q, (push))}$

$\delta(q_0, 0, \lambda) = \{(q_0, X)\}$

$\delta(q_0, 1, \lambda) = \{(q_0, Y)\}$

$\delta(q_0, \lambda, \lambda) = \{(q_1, \lambda)\}$

$\delta(q_1, 0, X) = \{(q_1, \lambda)\}$

$\delta(q_1, 1, Y) = \{(q_1, \lambda)\}$

$\delta(q_1, \lambda, Z) = \{(q_2, \lambda)\}$

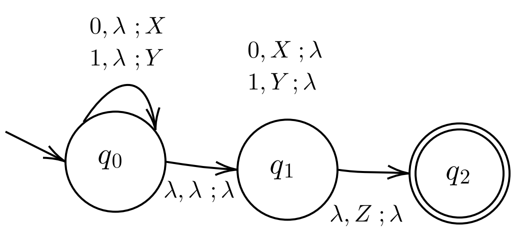
$\left(\delta(q, s, \gamma) \ = \{\},\ q \in Q\ ,\ \gamma \in \Gamma,\ s \in \Sigma \text{ otherwise}\right)*$

$q_I = q_0$

$z_I = Z$

$q_F = q_2$

*Might not be necessary



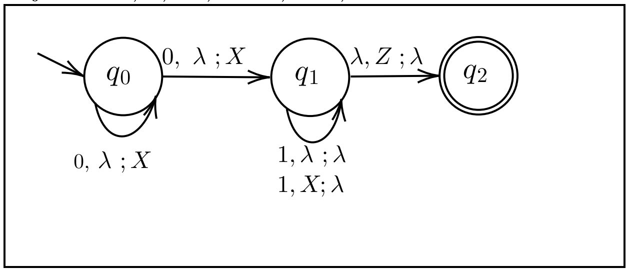
Examples:

a.

$L = \left\{ \omega \in \{0,1\}^* \mid \omega = 0^n 1^m,\ 1 \le n \le m \right\}$

Accepted: $0011, 011, 00011111, 011$

Rejected: $0, 1, 01, 0011, 001, \lambda$

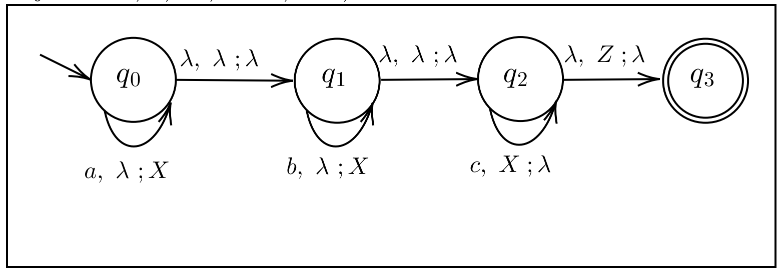


b.

$L = \left\{ \omega \in \{a,b,c\}^* \mid \omega = a^i b^j c^k, k = i + j,\ (i, j \ge 0) \right\}$

Accepted: $0011, 011, 00011111, 011$

Rejected: $0, 1, 01, 0011, 001, \lambda$

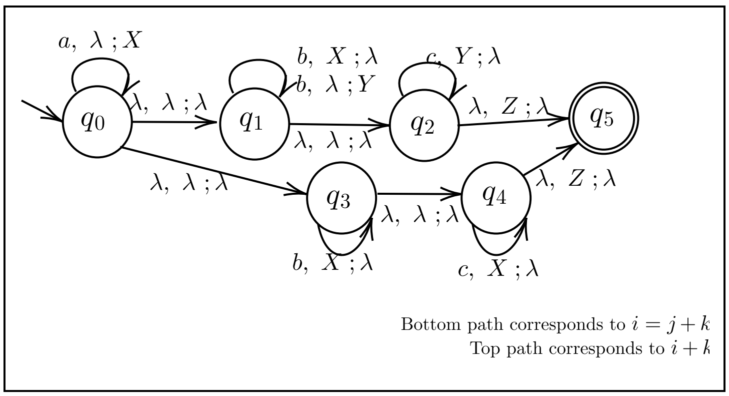


c.

$L = \left\{ \omega = \{a,b,c\}^* \mid \omega = a^i b^j c^k, i + k = j \vee i = j + k \right\}$

Accepted: *aaabbbbc*, *aabb*, *aaaaabcccc*

Rejected: *aaabbbbbc*, *aaaaabbcc*

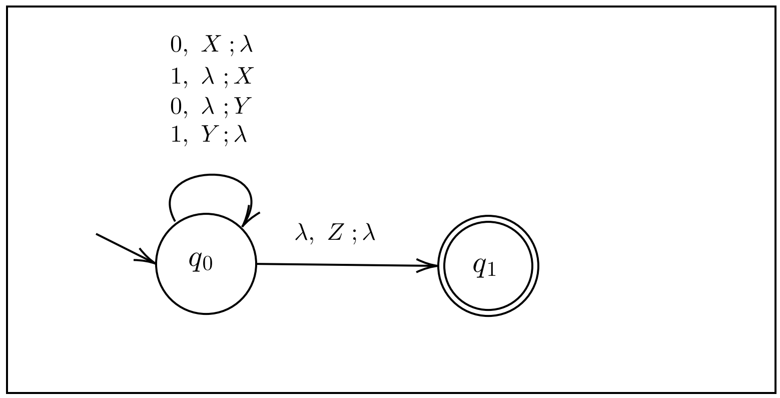


d. $L = \left\{ \omega \in \{0,1\}^* \mid N_0(\omega) = N_1(\omega) \right\}$

(Same no. of 0s and 1s)

Accepted: 010101, 0000111011, 10, λ

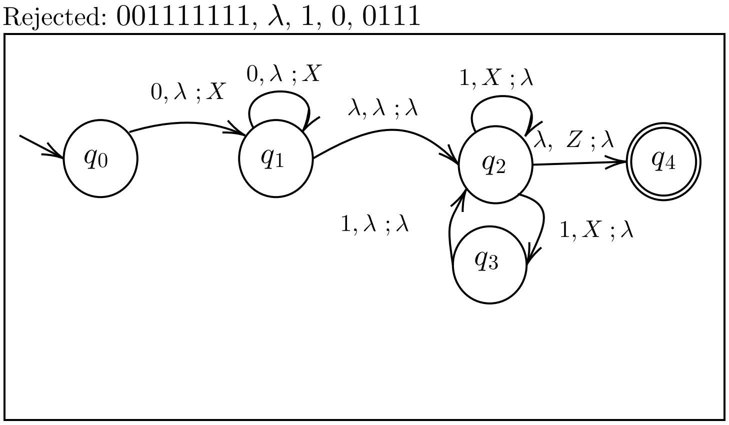
Rejected: 1110, 00, 10011, 00011



e. $L = \left\{ \omega \in \{0,1\}^* \mid \omega = 0^n 1^m, 1 \le n \le m \le 2n \right\}$

Accepted: 01, 0011, 00011111, 000011111

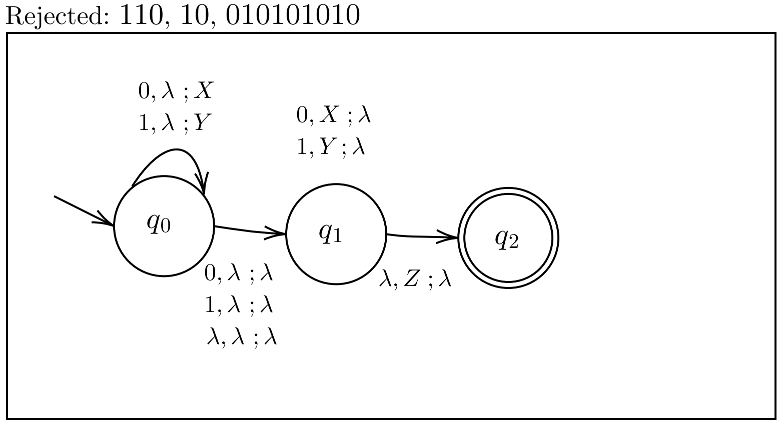
Rejected: 001111111, λ, 1, 0, 0111



f. $L = \left\{ \omega \mid \omega = (0 \cup 1)^*,\ \omega \text{ is a palindrome} \right\}$

Accepted: λ, 101, 10001, 010010, 00000000, 1111

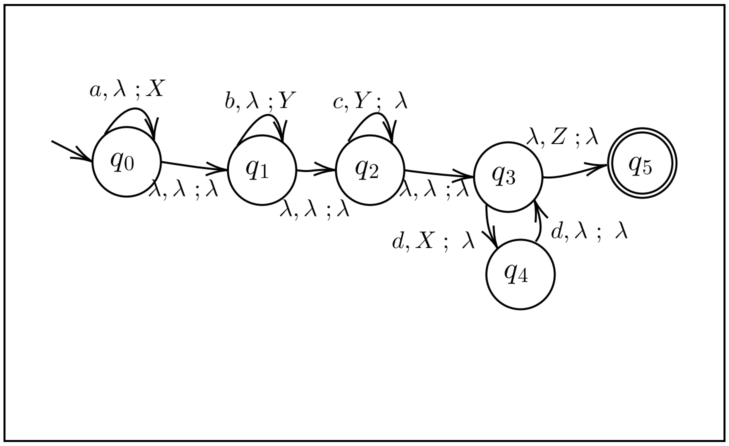
Rejected: 110, 10, 010101010



h. $L = \Big\{\omega \in \{a,b,c,d\}^\star \mid \omega = a^n b^m c^m d^{2n}\Big\}$

Accepted: *aadddd*, *aabcdddd*, λ, *abbccdd*

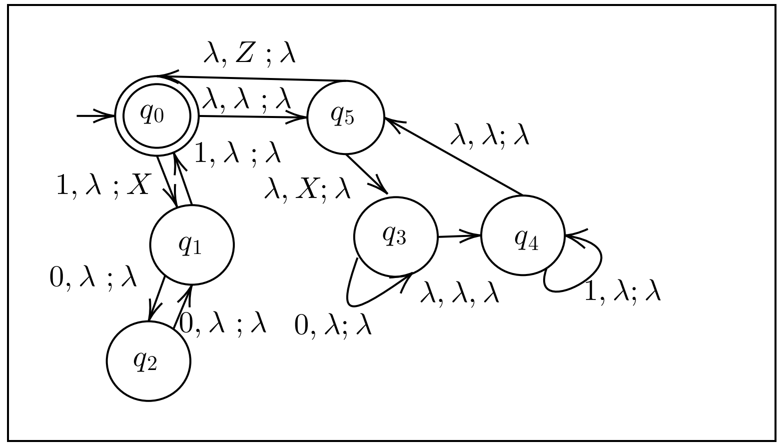
Rejected: *addd*, *bcddd*, *abccdd*



i. $\Big\{\omega \in \{0,1\}^* \mid \omega = \big[1(00)^*1\big]^n \big(0^*1^*\big)^n,\ n \geq 0\Big\}$

Accepted: 111111010, 10000110000110010, 1001, 11

Rejected: 1111010101, 101, 0101, 100101



[4] Context-Free Grammar

A context free grammar is a 4-tuple:
$G = (N, T, P, \Sigma)$ where:

- $N -$ finite set of nonterminals
- $T -$ finite set of terminals
- $P -$ finite set of productions
  $P \subset (N \cup \{\Sigma\}) \times (N \cup T)^*$
  Each production can be written as $\alpha \to \beta$, $\alpha \in N \cup \{\Sigma\}, \beta \in (N \cup P)^*$
  $\alpha \to \beta, \alpha \in N \cup \{\Sigma\}, \beta \in (N \cup T)^*$
  Note that only $\Sigma \to \lambda$ is allowed to have λ on the right hand side.
- $\Sigma -$ start symbol

Production: Rule that can be used to expand a sentence
Nonterminal Symbol: Is something that can be expanded using a production rule (CAPITAL)
Terminal Symbol: Is something that cannot be expanded using a production rule (lowercase)
A string of nonterminals is called a sentential form.

A sentence that consists entirely of terminal symbols is called a sentence.

Example:

a. $L = \left\{\omega \in \{0,1\}^* \mid \omega = 0^n 1^m,\ 1 \le n \le m\right\}$
$\Sigma \to A$
$A \to 0A1 \mid 01 \mid A1$

b.
$L = \left\{\omega \in \{a,b,c\}^* \mid \omega = a^i b^j c^k, k = i + j,\ (i, j \ge 0)\ \right\}$
$\Sigma \to A \mid \lambda$
$A \to aAc \mid B \mid ac$
$B \to bBc \mid bc$

c.
$L = \left\{\omega = \{a,b,c\}^* \mid \omega = a^i b^j c^k, i + k = j \vee i = j + k\right\}$
$\Sigma \to A \mid G \mid \lambda$
$(i + k = j)$
$A \to BC \mid B \mid C$
$B \to\ aBb \mid ab$
$C \to\ bCc \mid bc$

$(i = j + k)$
$G \to aGc \mid ac \mid H$
$H \to aHb \mid ab$

d. $L = \left\{\omega \in \{0,1\}^* \mid N_0(\omega) = N_1(\omega)\right\}$ (Same no. of 0s and 1s)
$\Sigma \to A \mid \lambda$
$A \to 0A1 \mid 1A0 \mid 01 \mid 10$

e. $L = \left\{\omega \in \{0,1\}^* \mid \omega = 0^n 1^m, 1 \le n \le m \le 2n\right\}$

$\Sigma \to A$
$A \to 0A1 \mid 0A11 \mid 01 \mid 011$

f. $L = \left\{\omega \mid \omega = (0,1)^*,\ \omega \text{ is a palindrome}\right\}$
$\Sigma \to A \mid \lambda$
$A \to 0A0 \mid 1A1 \mid 1 \mid 0 \mid 11 \mid 00$

g. $L = \left\{\omega \text{ is balanced parenthesis}\right\}$ (λ is accepted)
$\Sigma \to A \mid \lambda$
$A \to AA \mid (A) \mid ()$

h. $L = \left\{\omega \in \{a,b,c,d\}^\star \mid \omega = a^n b^m c^m d^{2n}\right\}$
$\Sigma \to A \mid \lambda$
$A \to aAdd \mid add \mid B$
$B \to bBc \mid bc$

i. $L = \left\{\omega \in \{0,1\}^* \mid \omega = 0^n 1^m, n \ne m\right\}$
$\Sigma \to A \mid B$
$A \to 0 \mid 0A1 \mid 0A$
$B \to 1 \mid 0B1 \mid B1$

A , 0A1, 001
B, 0B1, 00B11

j. $L = \left\{\omega \in \{a,b,c\}^* \mid \omega = a^n b^n c b^m a^m \wedge n, m \ge 0\right\}$
$\Sigma \to A$
$A \to BcC \mid Bc \mid c \mid cC$
$B \to aBb \mid ab$
$C \to bCa \mid ba$

k. $L = \left\{\omega \in \{a,b,c\}^* \mid \omega = a^i b^j c^k, 2i = j + k \vee i = 3j\right\}$
$\Sigma \to A \mid G \mid \lambda$
$(2i = j + k)$
$A \to aAcc \mid acc \mid B \mid aBbc \mid abc$
$B \to aBbb \mid abb$

$(i = 3j)$
$G \to HI \mid H \mid I$
$H \to aaaHb \mid aaab$
$I \to Ic \mid c$

l. $L = \left\{\omega \in \{0,1\}^* \mid \omega = \left[1(00)^* 1\right]^n \left(0^* 1^*\right)^n,\ n \ge 0\right\}$
$\left(\text{14 productions},\ \text{5 non-terminals}\right)$
$\Sigma \to A \mid \lambda$
$A \to BAD \mid BD \mid BA \mid B$

$B \to 1C1 \mid 11$
$C \to 00 \mid 00C$

$D \to 0D \mid 0 \mid D1 \mid 1$