«نظریه زبان ها و ماشین ها»

Automata Theory

① deals with the defintions and properties of mathematical models of computation

با مدل های ریاضی برای محاسبات / معرفی مدل ها و قابلیت ها و محدودیت های آنها

② these models play a role in several applied areas of computer science.

یکی از اولین مدل ها — کاربردی

- finite automation — کاربرد → text processing, compilers, hardware design

یکی دیگر از مدل ها

- context-free grammer — کاربرد → programming language, AI (artificial intelligence)

این مدل ها به خوبی بیان کننده ی کامپیوتر هستند
یک تقریب از واقعیت

Computability Theory

تئوری محاسبه پذیری

هدف : تقسیم بندی مسائل به مسائل حل پذیر به شکل الگوریتمی و مسائل حل ناپذیر به شکل الگوریتمی

in this theory, the classification of problem is by those that are solvable and those that are not. * computability theory introduces several of the concepts used in complexity theory.

Complexity Theory

طبقه‌بندی مسائل با توجه به میزان سختی آنها و پیچیدگی محاسباتی آن

alphabet: any nonempty finite set

Σ and Γ

① الفبا: یک مجموعه‌ی متناهی و ناتهی از سمبل‌ها است.

② سمبل: اعضای تشکیل دهنده‌ی یک الفبا

③ برای نشان دادن

الفبا $\Sigma_1 = \{0,1\}$ $\Sigma_2 = \{a,b,c,d\}$

$\Gamma = \{0,1,x,y,z\}$

④ String over an alphabet: finite sequence of symbols from that alphabet, usually written next to one another and not separated by commas.

رشته: رشته‌ای متناهی از سمبل‌ها که روی یک الفبا هستند که در کنار هم نوشته می‌شوند (آخری)

if $\Sigma_1 = \{0,1\}$ then 0100 is a string over Σ_1

$\Sigma_2 = \{a,b,c,\dots,z\}$ → abracadabra is a string over Σ_2

⑤ length of w = $|w|$ = number of symbols that it contains.

طول رشته = تعداد سمبل‌هایی که در بردارد

⑥ The string of length zero is called the empty string

رشته‌ی تهی یا به طول صفر: ε یا λ

empty string plays the role of 0 in a number system.

⑥ if w has length n, we can write $w = w_1 w_2 \ldots w_n$ where each $w_i \in \Sigma$ معکوس w

⑦ The reverse of w, is w^R → is the string obtained by writing w in the opposite order (i.e., $w_n w_{n-1} \ldots w_1$) کاراکترها را برعکس بنویسیم.

⑧ string z is a substring of w if z appears consecutively within w.

abra[cad]abr مثلاً cad یک زیر رشته است از abracadabr

پشت سر هم شده باشد

⑨ string x → length = m concatenation of x and y = xy

string y → " = n

با اضافه کردن y به آخر x به دست میاد.

$x_1 \ldots x_m y_1 \ldots y_n$ ← رشته

⑩ to concatenate a string with itself many times → x^k $\underbrace{xx\ldots x}_{k}$

k بار x را به خودش می‌چسبانیم.

⑪ lexicographic قانونمند →

ترتیب رشته‌ها مثل ترتیب الفبا. dictionary

ما از یک قانون نامه دیگر به نام (shortlex) استفاده می‌کنیم

ترتیب قانونمند مطابق الفبا اما کلمات کوتاه‌تر اولویت بیشتری دارند → The string ordering of all strings over the alphabet {0,1} is (ε, 0, 1, 00, 01, 10, 11, 000, …)

رشته تهی · طول ۱ · رشته‌ها به طول ۲

پیشوند

(۱۲) x is a prefix of y if → حداول y باشد / یک z وجود داشته باشد که xz = y

suffix : پسوند → اگر رشته‌ی x آخر رشته‌ی y ظاهر شود

(۱۳) A language is a set of strings.

زبان = مجموعه‌ای از رشته‌ها

prefix-free if no member is a propper prefix of another member.

الفبا ← متناهی   رشته ← متناهی   زبان‌ها ← لزوما متناهی نیستند

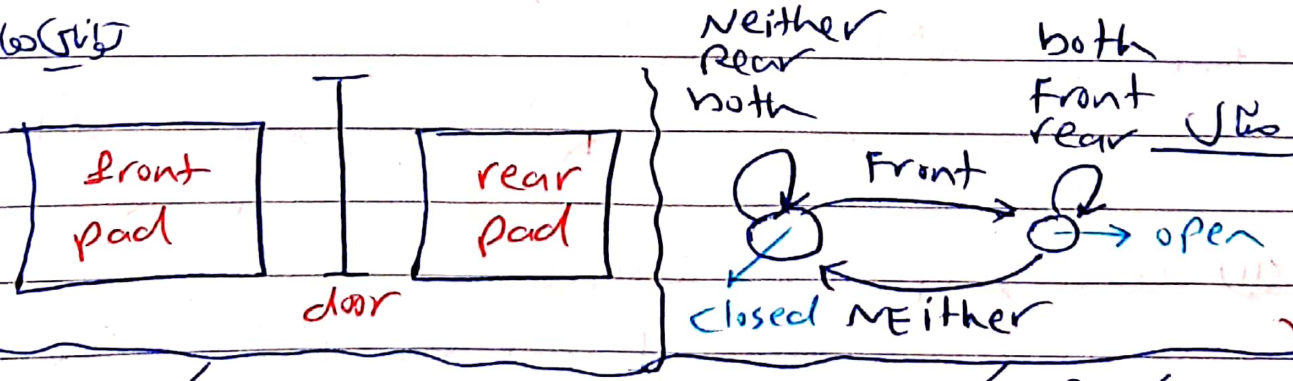
مدل‌هایی که معرفی می‌شوند abstrac هستند.

we begin with the simplest model called the finite state machine or finite automation

مدل‌های خوبی برای کامپیوترها هستند

مقدار خیلی کم و محدودی memory دارند ← finite automata

چه کار می‌توانند بکنند و چه کاری نمی‌توانند؟ محدودیت‌ها و توانایی‌هایشان؟



مثال

برای نگه داشتن اینکه در توی چه حالتی (state) ای هست به یک بیت نیاز داریم

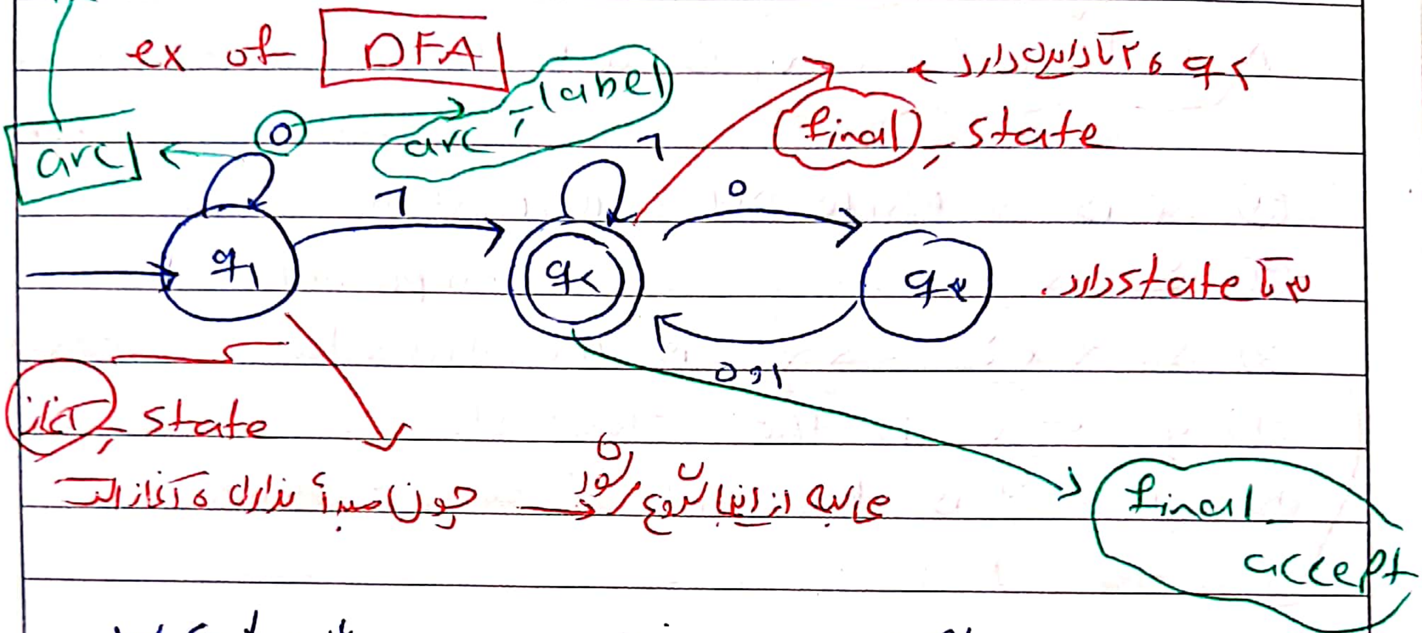
۲ تا pad داریم ← ۴ تا سیگنال می‌توانند ارسال کنند

state diagram

state	Neither	front	Rear	Both
closed	closed	open	closed	closed
open	closed	open	open	open

transition

ex of DFA



چون مبدأ نداره و آغاز است → یعنی از اینجا شروع شود

$w \in \{0,1\}$ ورودی

خروجی → accept
→ reject

1101 → ✓ q1 →1 q2 →1 q2 →0 q3 →1 (q2)

تک تک حروف ها

حروف اسم

چون کار موقع q2 ختم شد

و چون q2 یک state accept هست و

این رشته قبول میشه

کار دستگاه با را چیه؟ e تعدادی رشته را قبول میکند و تعدادی را رد میکند

DFA → Deterministic Finite automata

قطعی

صفرها زوج باشد

دراین ماشین ها و رفتارها قطعی است.

زبان این ماشین رشته‌های توالی DFA را قبول میکند باید ① نباید تمام صفرها ② تعداد آخرش ۱

یا زوج تا صفر داریم یا صفری نداریم

زبان ماشین M1 ← مجموعه رشته هایی که توسط DFA پذیرفته می شوند

(زبان) → L(M1)

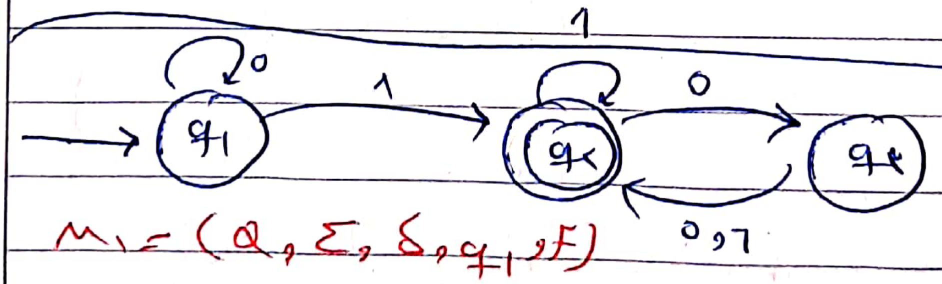
«formal definition of finite automata»

A finite automaton (→ DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where
(چندتایی مرتب)

① Q is a finite set called states
② Σ is a finite set called alphabet
③ $\delta: Q \times \Sigma \to Q$ is the transition function (دامنه: $Q \times \Sigma$ ، برد: Q ، ضرب دکارتی)
④ $q_0 \in Q$ is the start state
⑤ $F \subseteq Q$ is the set of accept states (زیرمجموعه F)

$\delta(q_0, a) = q_3$ اگر در q_0 باشیم و a را خواندیم به q_3 می رویم.
(a: کاراکتری که ... ، q_0: که توش هستیم الان ، q_3: مرحله‌ی بعد)

δ (دلتا) باید مشخص کند به ازای هر عضو از الفبا اگر توی یک state ای، اون کاراکتر را خواندیم به کجا باید بریم.



← M1

$M_1 = (Q, \Sigma, \delta, q_1, F)$

$Q = \{q_1, q_2, q_3\}$

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

q_1 is the start

$F = \{q_2\} \subseteq Q$

δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

$\delta(q_1, 1) = q_2$

تغییر هر کدوم الفبا دقیقا یک خروجی

arc

نکته: در دیاگرام DFA لازم هست تغییر هر کدام از اعضای الفبا ۰ و ۱ یک arc خروجی

از هر state داشته باشیم ←

در q_1 ← تغییر صفر یک خروجی   در q_2 ← تغییر صفر یک خروجی

۱ و ۰ هر یک   هر یک

if A is the set of all strings that machine M accepts → we say that A is the language of machine M → $L(M) = A$

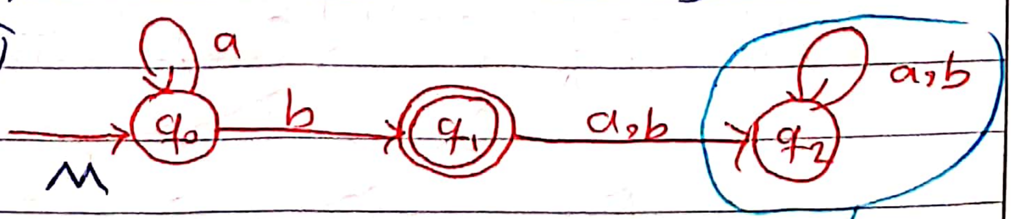
M recognizes A or M accepts A

A machine may accept several strings, but it always recognizes only one language. if the machine accepts no string, it still recognize one language - namely the empty language $\emptyset$.

$A = \{w \mid w$ contains at least one 1 and an even number of 0s follow the last 1$\}$.

then → $L(M_1) = A$

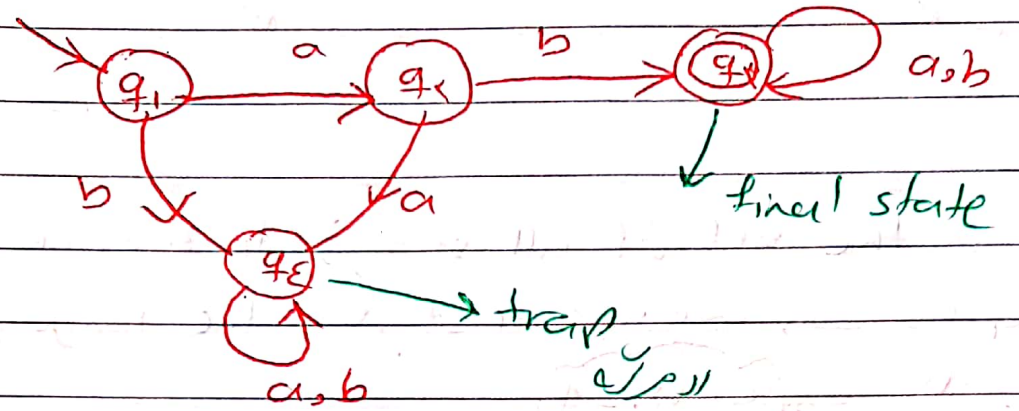
example



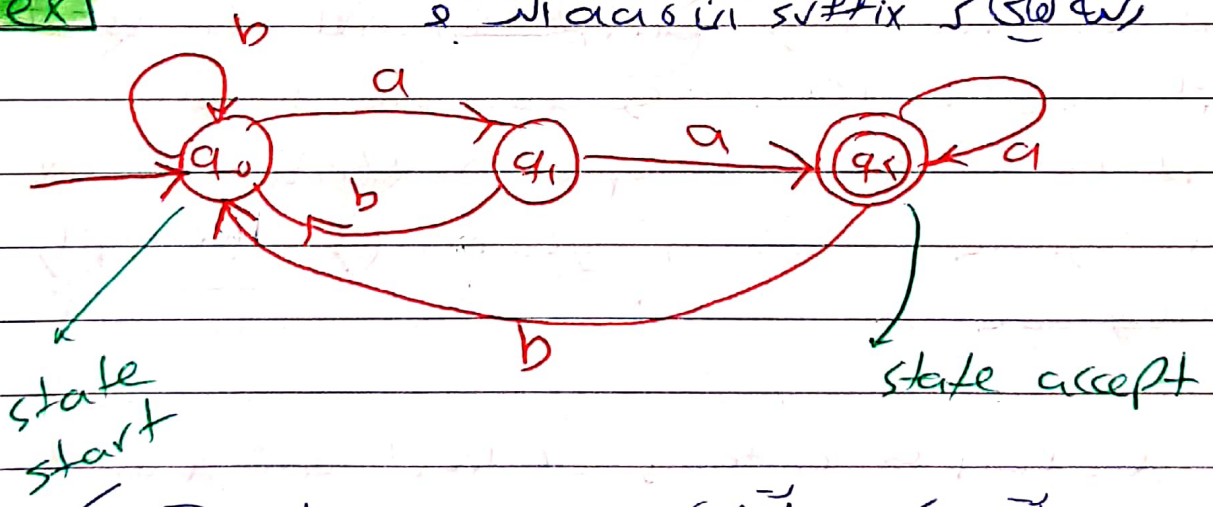
$\Sigma = \{a, b\}$ $L(M) = \{a^i b \mid i \in \mathbb{Z}^{\geq 0}\}$

ex DFA $\Sigma = \{a, b\}$ رشته‌ای که اگر اولش ab باشد قبول و اگر نباشد رد

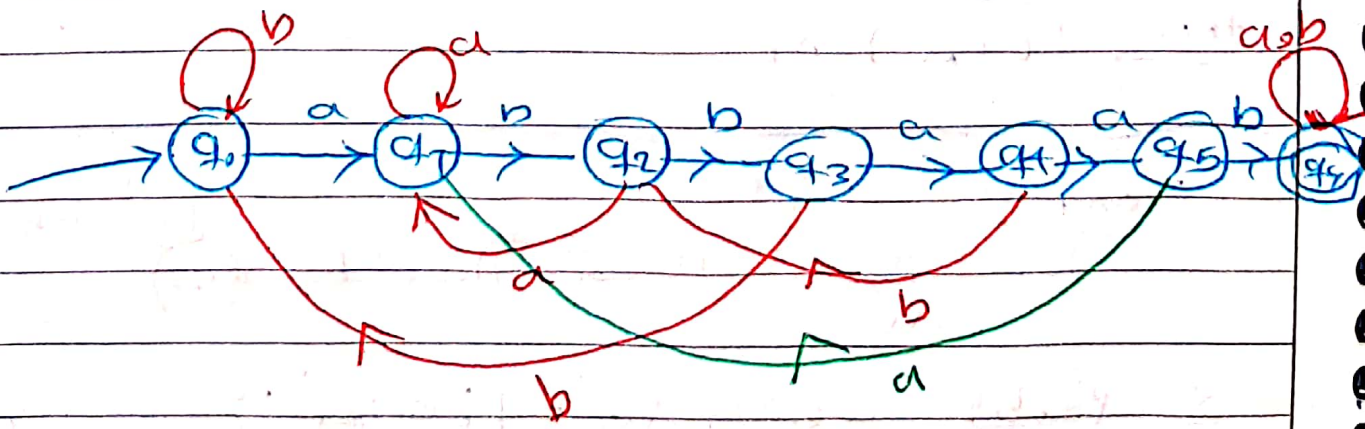
یعنی ab یک prefix باشد.



ex رشته‌هایی که suffix آن aaa باشد.



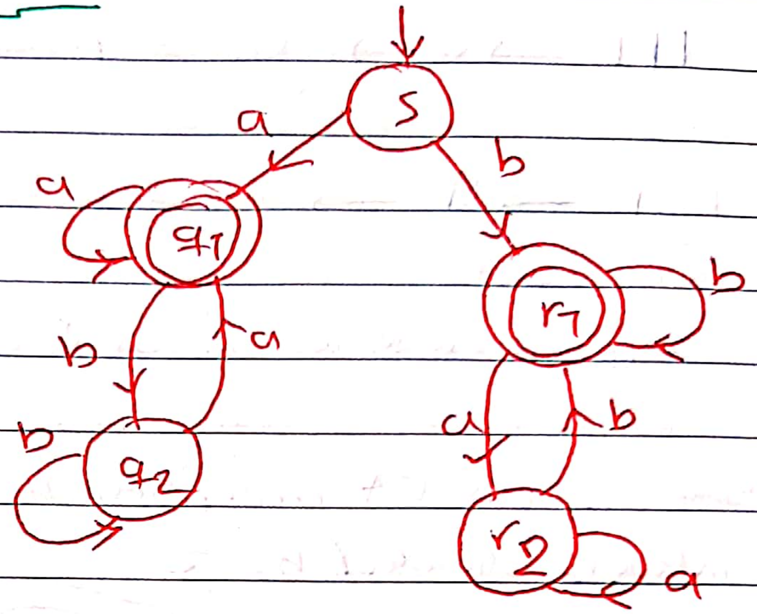
ex چون رشته‌هایی که زیررشته‌ی substring ababaab را دارند قبول کند



(reject ← یعنی trap)

ex strings that start and end with the same symbol.

$\Sigma = \{a, b\}$

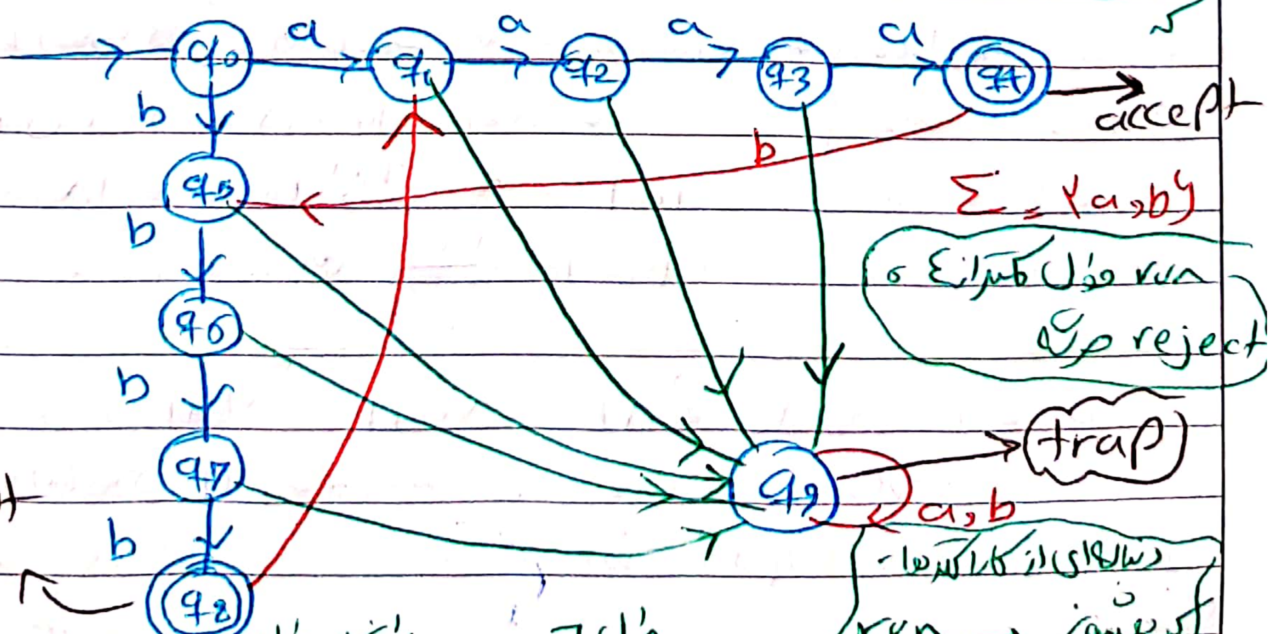


اگر اول b دیدیم تا زمانی که
b ببینیم در فاینال state
هستیم و به محض دیدن a
از final state خارج می‌شویم
و تا وقتی که دوباره b
نبینیم به r1 برنمی‌گردیم

زمانی قبول می‌شود که
کاراکتر اول با آخری یکی باشد

Example (no runs of length less than four)

رشته‌هایی (run های به طول کمتر از ۴ ندارند)



$\Sigma = \{a, b\}$

run به طول کمتر از ۴
reject می‌شه

دنباله‌ای از کاراکترها
run ← که همه‌شون
کاراکتر یکسان دارند.

رشته به طول ۷ — طول 7

(aaa) bb aaaa ba

run به طول ۳ — run به طول ۲

مثال از کتاب martin

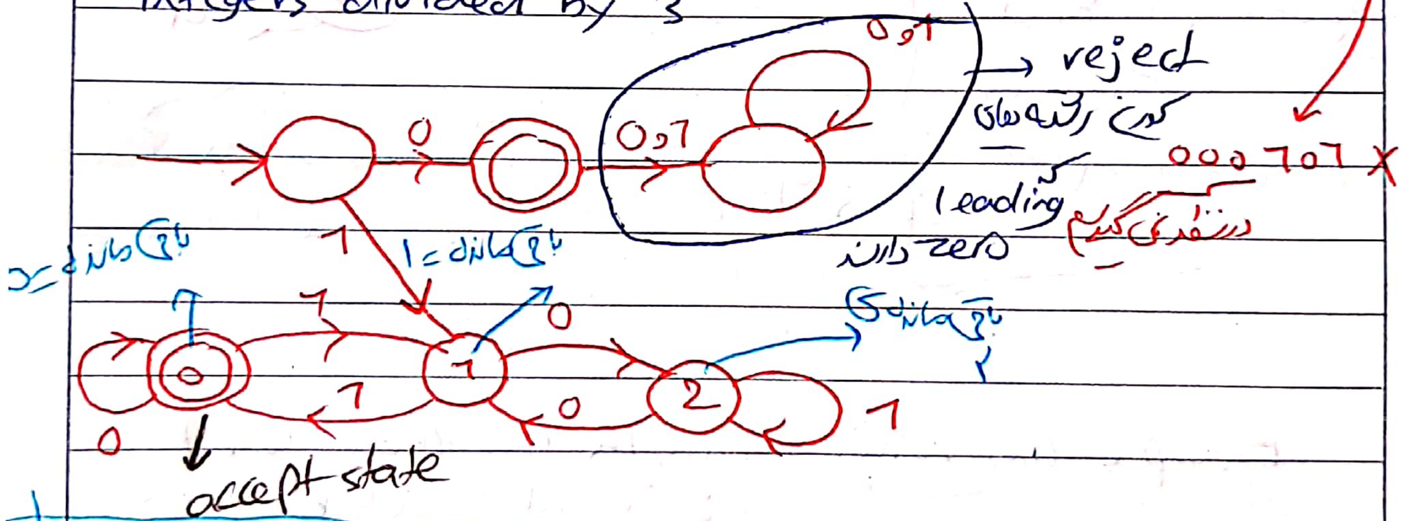
ex رشته‌هایی که فرم دسیمال آنها بر ۳ بخش‌پذیر باشد قبول در غیر این صورت رد

111 → ۷ → باقی‌مانده بر ۳ = ۱ → reject

1001 → ۹ → ۳ = ۰ → accept

leading zero ← صفرهایی که سمت چپ عدد هستند و بدون خواندن را رد می‌کنیم

مسئله → an FA accepting binary representations of integers divided by 3



به ازای هر صفر به سمت راست اضافه می‌شود عدد ۲ برابر می‌شود

* با اضافه کردن یک صفر در سمت راست یک عدد باینری آن عدد در مبنای دسیمال ۲ برابر می‌شود

1001 = ۹ $\xrightarrow{9\times 2}$ 10010 = ۱۸ →

100100 $\xrightarrow{9\times 4}$ ۳۶ و ...

$a \overset{3}{\equiv} b \rightarrow 2a \overset{3}{\equiv} 2b$, $2a+1 \overset{3}{\equiv} 2b+1$

با اضافه کردن **1** به سمت راست ← عدد ۲ برابر می‌شود و به علاوه‌ی یک

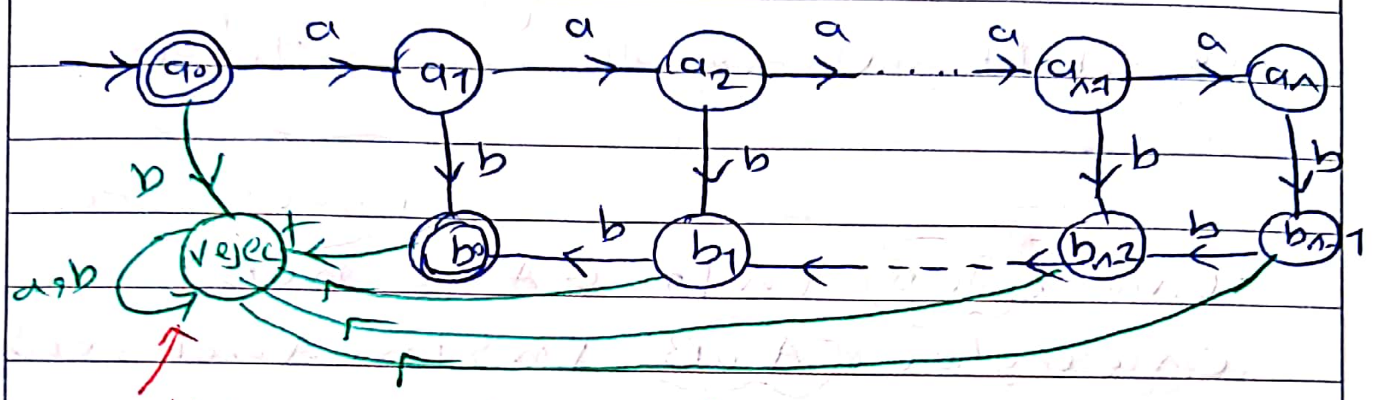
* رشته را از چپ به راست می‌خوانیم و چک هر کدام از کاراکترها را تعیین می‌کنیم هر کدام جدا کنند.

ex

قبول کردن همه ی رشته های به فرم $\{a^i b^i \mid i \le n\}$

$\Sigma = \{a, b\}$

n یک ثابت داده شده الف



the incomplete specified DFA

states a_k count the number of a's

〃 b_k 〃 〃 b's

DFA

تعریف رسمی محاسبات DFA ها

Formal Definition of Computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \dots w_n$ (کاراکتر اول ... کاراکتر آخر) be a string where each w_i is a member of the alphabet Σ.

then M accepts w if a sequence of states $r_0, r_1, \dots, r_n$ in Q exists with three condition

1. $r_0 = q_0$
2. $r_n \in F$
3. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, \dots, n-1$

شرط ها

مجموعه ی state ها

روندمان باید از q_0 ـ state آغاز شود

state start

$\delta(r_0, w_1) = r_1$

اگر در r_0 باشیم و w_1 را بخوانیم میریم توی r_1

regular زبان منظم

We say that M recognizes language A if A = {w | M accepts w}.

A language is called a regular if some finite automaton recognizes it. ← شرط منظم بودن

زبان منظم است ← اگر تشخیص یک زبان با یک DFA در آن قابل باشد

عملگرهای regular

Union: (A ∪ B) = {x | x ∈ A or x ∈ B} (اجتماع)

Concatenation: (A ∘ B) = {xy | x ∈ A and y ∈ B}

Star: A^* = {$x_1 x_2 \ldots x_k$ | k ≥ 0 and each x_i ∈ A} (unary عملگر)

تعداد اعضای مجموعه |A ∘ B| = |A| × |B| تعداد اعضای تعداد اعضای

* the empty string ε is always a member of A^*, no matter what A is.

کار A^* ← کارش چسباندن رشته های داخل زبان A به اندازه ی K بار به یکدیگر
صفر بار چسباندن ← اپسیلون ← عضو همیشگی از A^*

زبان ← مجموعه ای از string ها

ex: alphabet (Σ) → be the standard 26 letters {a, b, ..., z}

if A = {good, bad} and B = {boy, girl},

A ∪ B = {good, bad, boy, girl}

A ∘ B = {goodboy, goodgirl, badboy, badgirl}

A^* = {ε, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, ...}

یک بار تکرار ← {۰, ۱, ۲, ۳, ...} ← دو بار تکرار

مجموعه‌ی زبان‌های منظم، بسته است.

Theorem: The class of regular language is closed under the union operation. In other words if A_1, A_2 are regular languages, so is $A_1 \cup A_2$

① یعنی اجتماع بسته‌اند ، زبان‌های منظم تحت عمل اجتماع بسته‌اند

یعنی اگر A_1 , A_2 زبان‌های منظم باشند ← $A_1 \cup A_2$ هم زبان منظم است.

$A_1, A_2 \rightarrow$ regular languages

M_1 recognizes $A_1 \rightarrow L(M_1) = A_1$

M_2 〃 $A_2 \rightarrow L(M_2) = A_2$

حالا می‌خواهیم ثابت کنیم اجتماع هم منظم است که یک DFA طراحی کنیم یا نشان بدهیم DFA ای وجود دارد که زبان تشخیص آن $A_1 \cup A_2$ است. ↙

$L(M) = A_1 \cup A_2$

* در DFA ها فقط می‌توانیم رشته را یک بار از چپ به راست بخوانیم * و بعد از خواندن آخرین کاراکتر تصمیم گرفته می‌شود که آیا رشته پذیرفته می‌شود یا نه!!

پس باید یک کاری کنیم که M همزمان A_1 , A_2 را بخواند

if M_1 has K_1 states and M_2 has K_2 states → the number of states one from M_1 and the other from M_2 → $(K_1 \cdot K_2)$ $(\bigcirc , \bigcirc)$

متعلق به A_1 ، متعلق به A_2

* به شکل همزمان simulate می‌کنیم M_1 , M_2 را *

$\Sigma_1 \cup \Sigma_2$

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

$\Sigma_1 \cup \Sigma_2$

(q_1, q_2)

$M = (Q, \Sigma, \delta, q_0, F)$

① $Q_1 \times Q_2$ ضرب دکارتی دو مجموعه

$L(M) = A_1 \cup A_2$

اون حالت های state را قبول می کنیم که یا M_1 یا M_2 قبول کردن

مولفه ی اول از Q_1

مولفه ی دوم از Q_2

یا مولفه ی اول توی F_1 هست

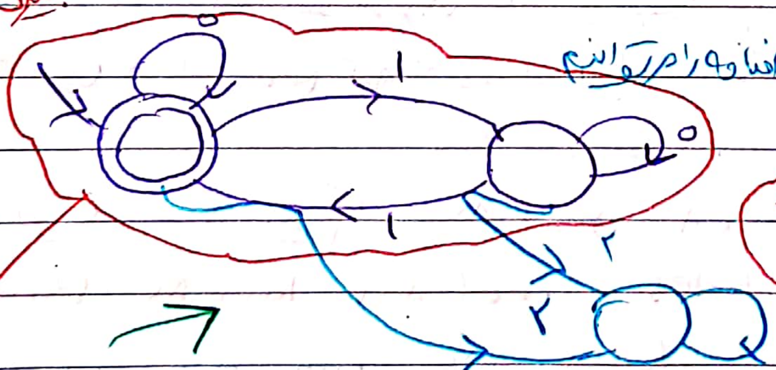
یا مولفه ی دوم در F_2 هست

② $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

③ الفبای تغییر → $\Sigma_1 \cup \Sigma_2$

الفبای تغییر پذیری

اگر الفبای M_1 و M_2 تفاوت داشت چه کار می کنیم؟ الفباها شان را یکی می کنیم و مثل مثال زیر حرف های اضافه را trap کنیم



reject می کنیم | ۲ را لازم کنیم

$\Sigma = \{0, 1, 2\}$

حوزه ی الفبای M_1 را بزرگ کنیم در حالی که زبان تغییر M_1 مثل قبل باشد

اگر الفبا واحد نبود کارهای بالا را انجام می دهیم

رشته های a_1 و a_2 زوج تایی دارند

Q_1 توی Q_2 توی

تابع transition

④ $\delta: (Q_1 \times Q_2) \times \Sigma \rightarrow Q_1 \times Q_2$

$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

عضو Q الف $a \in \Sigma$

که هستیم، چه سمبلی را از ورودی می خوانیم و کجا می خواهیم بریم

$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

وقتی که در r_1 هستی و a مرفوتی ببین چه اتفاقی می‌افتد ← در M_2 چه اتفاقی می‌افتد که تنها در r_2 هستی و a را مرفوتی

(c) $q_0 \rightarrow (q_1, q_2)$

(A) برای اشتراک ← $F = F_1 \times F_2$

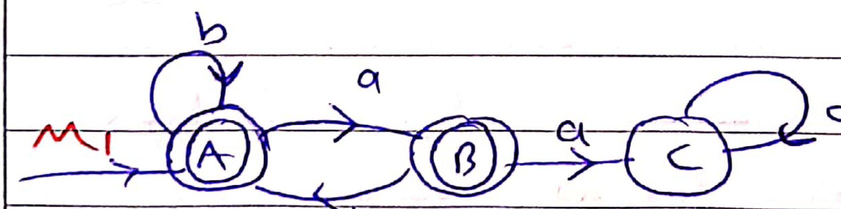
حالت‌های اول در M_1 و حالت‌های دوم در M_2

رشته‌هایی که هم M_1 آنها را بپذیرد هم M_2 → $L(M) = A_1 \cap A_2$

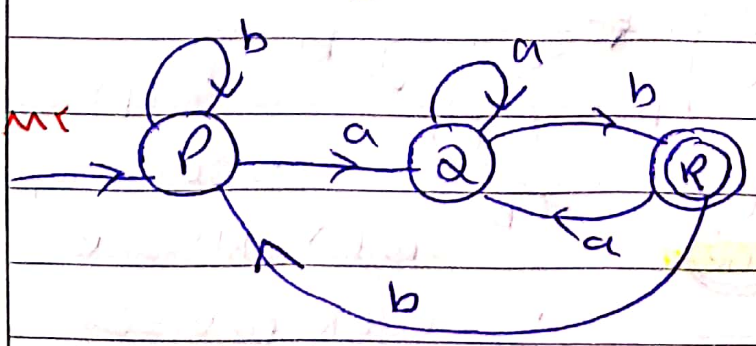
سوال ۵ رشته‌های باینری را مرفوتیم accept کنیم که یا لزوما L_1 را داشته باشند یا لزوما L_2

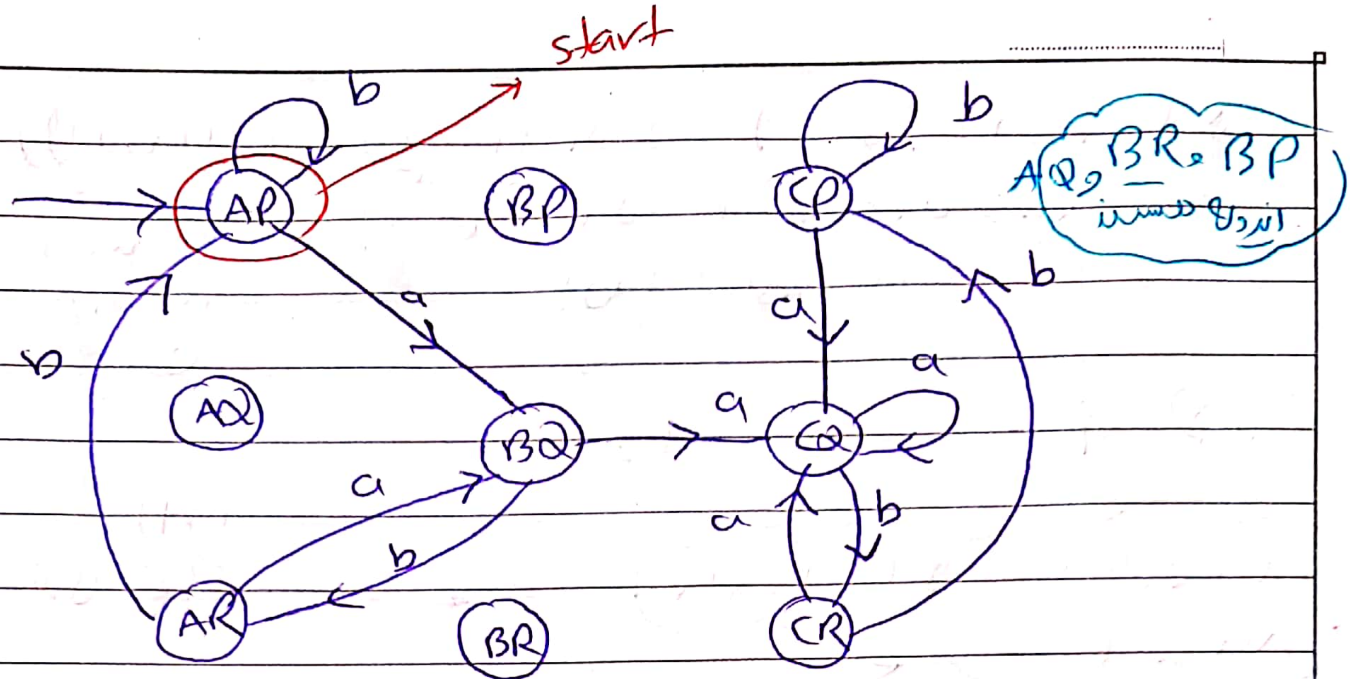
$L_1 = \{x \in \{a,b\}^* \mid aa \text{ is not a substring of } x\}$ &

$L_2 = \{x \in \{a,b\}^* \mid x \text{ ends with } ab\}$



$M_1 \cup M_2 = ?$





زوج مرتب هایی که مولفه ی اول آنها A یا B یا C — M1 U M2 ?
و مولفه ی دوم آنها P یا Q یا R

در M1 وقتی در C هستیم و a می خوانیم

$3 \times 3 = 9$

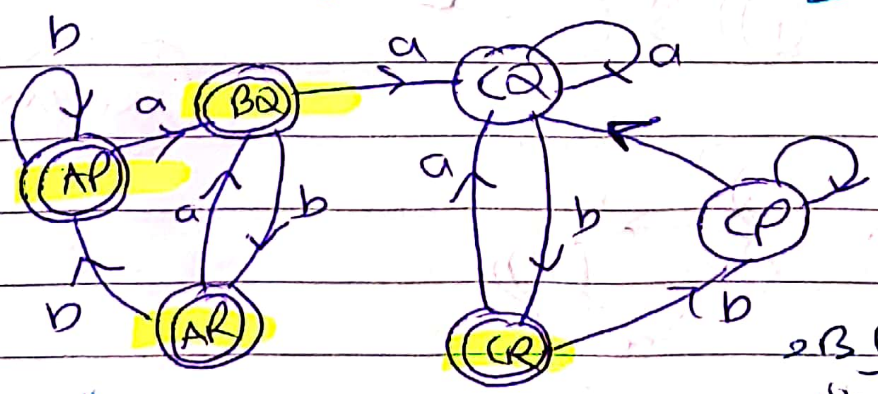
$\delta(CP, a) = (\delta_1(C, a), \delta_2(P, a))$ داریم state ۵

$= CQ$

در M2 وقتی در P هستیم و a می خوانیم

$\delta(AR, a) = (\delta_1(A, a), \delta_2(R, a)) = BQ$

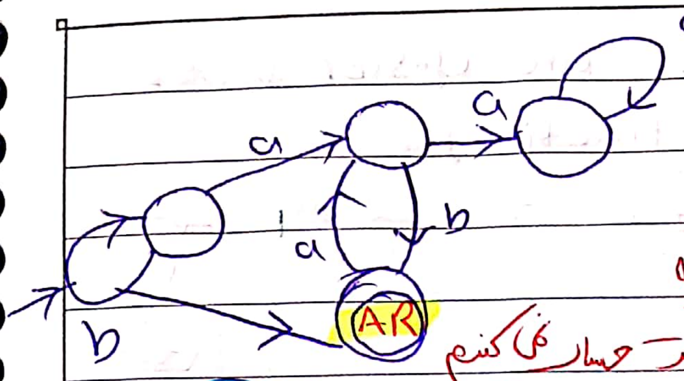
۳ تا state ها اندرو دسترس ← ۶ تا می شود



کدام state ها accept اند؟
مولفه اول A یا B و
مولفه ی دوم R باشد

4 تا

DFA ← تغییر اجتماع



DFA ـِ تقاطع اشتراک

برای اشتراک $F_1 \times F_2$

فقط AR accept میشه

BR هم شرط داره ولی چون از دوله الف حساب نمی‌کنیم

$L_1 \cap L_2$

$L_1 \setminus L_2$ ← تفاضل دو مجموعه ← $L_1 - L_2$

B عملگر باینری

$L_1 \cup L_2$ $\quad$ $L_1 \cap L_2$ $\quad$ $L_1 \setminus L_2$ (تفاضل) $\quad$ $L_1 \circ L_2$ (Concat)

دو عملگر unary: $\overline{L_1}$ (مکمل ← complement) $\quad$ L_1^*

complement

مکمل یک زبان

the complement of $L \subseteq \Sigma^*$ denoted by $\overline{L}$, is such that $\overline{L} = \Sigma^* \setminus L$

همه رشته‌هایی که روی الفبا قرار دارند ولی داخل زبان ان الفبا قرار ندارند

همه رشته‌هایی که داخل L قرار ندارند

مثال: اگر L دارای همه رشته‌هایی باشد که طول آنها زوج است

$\overline{L}$ ~ ~ است که ~ فرد ~

Σ^* → مجموعه همه رشته‌هایی که روی الفبای Σ تعریف می‌شوند

اگر L منتظم باشد ← $\overline{L}$ هم منتظم است Final ها را non final کن و برعکس

$M = (Q, \Sigma, \delta, q_0, F)$ $\quad$ $M' = (Q, \Sigma, \delta, q_0, Q \setminus F)$

→ $L(M') = \Sigma^* \setminus L = \overline{L}$

$Q - F$ را بعنوان final ها قبول میکنیم ← اگر زبان تشخیص M، L باشد $(L(M) = L)$

the regular languages are closed under set difference (subtraction)

$L_1 - L_2 = L_1 \cap \overline{L_2}$ → با اشتراک → متمم الف

متمم — متمم

راه دیگر برای اثبات متمم بودن اشتراک ؛

$\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$

$L_1 \to \overline{L_1}$, $L_2 \to \overline{L_2}$ → $\overline{L_1} \cup \overline{L_2}$ → $\overline{\overline{L_1} \cup \overline{L_2}}$ → متمم

متمم

متمم الف $L_1 \cap L_2$

Nondeterminism & NFAs

① DFA ها deterministic computations

when the machine is in a given state and reads the next input symbol, we know what the next state will be → it is determined

زمانی قطعی است که در هر state که هستیم بعدی رو میدونیم

② in a nondeterministic (غیرقطعی) machine, several choices may exist for the next state at any point

ممکن است تفاوتی نداشته باشیم یا در state ای هیچ حرکتی نداشته باشیم

ممکن است چندتا انتخاب بعدی

③ None determinism is a generalization of determinism, so every DFA is automatically an NFA

۴ جدا از NFA ها می‌توانیم استفاده کنیم وقتی غیرقطعی هستند!!

① NFA ها قدرت بیان ← خیلی بیشتری دارند نسبت به DFA ها

② طراحی NFA یک زبان خیلی راحت‌تر است نسبت به DFA

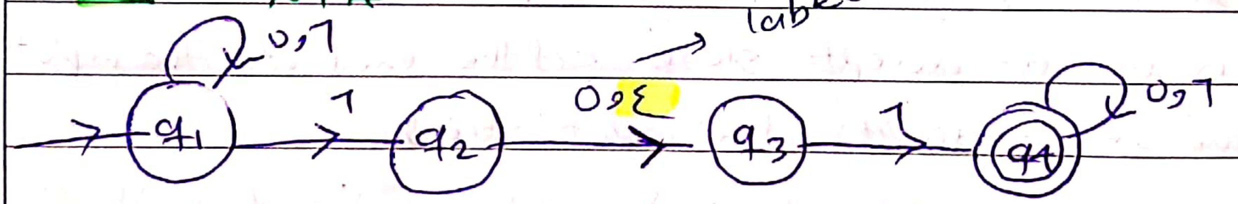
regular

③ تعداد state ها خیلی کمتر است و راحت‌تر درک می‌شود

④ از آنجایی که هر NFA می‌تواند به DFA تبدیل شود با ساختن یک NFA می‌توانیم خیلی راحت یک DFA را بسازیم.

---

ex) NFA



① در NFA، تغییر یک state، می‌توانیم یک ورودی را الفبا بگیریم یا بیش از ۲ یا هیچ ورودی نه الفبا بگیریم

② یک arc داریم که ε است

اپسیلون = ε

How does an NFA compute?

① we are running an NFA on an input and come to a state with multiple ways to proceed. after reading that symbol, the machine splits into multiple copies of itself and follows all the possibilities in parallel.

each copy of the machine takes one of the possible ways to proceed and continues as before. if there are subsequent choices, the machine splits again.

تقلید هر خروجی (حتی صفر) اگر ۱۰ تا خروجی بیش داشته باشیم ، باید ببینیم چه اتفاقی می‌افتد.

③ if the next input symbol doesnt appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of the computation associated with it.

④ if any one of these copies of the machine is in an accept state at the end of the input the NFA accepts the input string.

اگر یکی از این کپی‌ها ، ماشین‌های کپی‌شده ما را به accept state ببره ، رشته ورودی را قبول می‌کنیم در غیر این صورت نه.

★ اگر دیدی label یک arc ، ε است به معنی اینه به ابتدای مبدأ arc وارد شدیم می‌تونیم در انتهای arc هم باشیم.

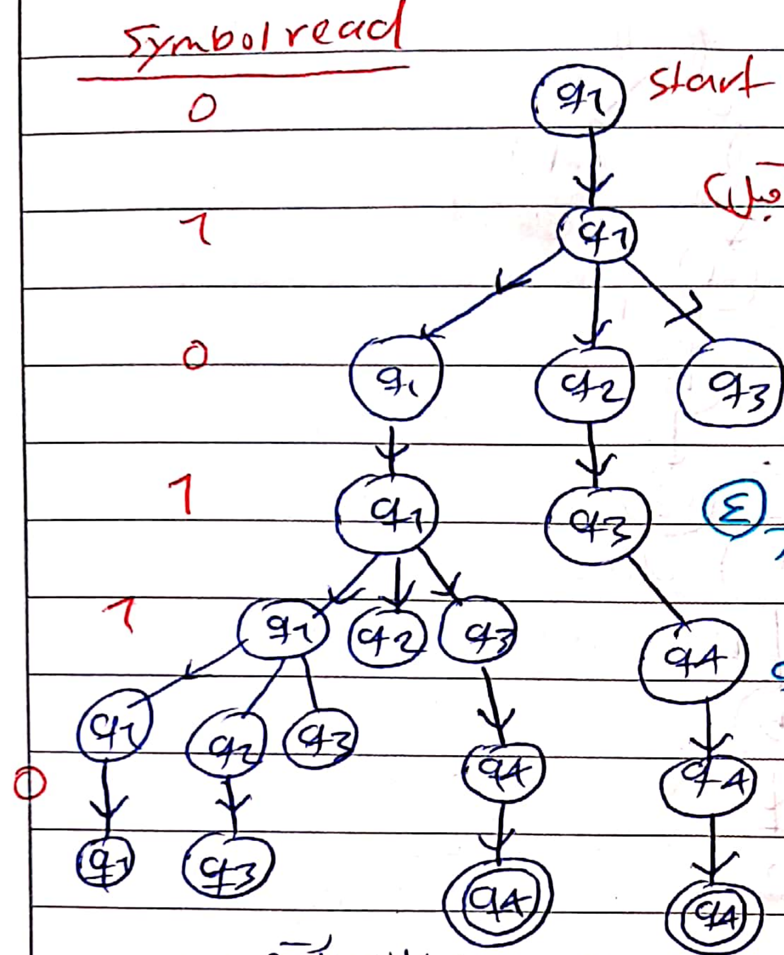
if an state with an ε symbol on an exiting arrow is encountered, then without reading any input, the machine splits into multiple copies

Tree of possibilities — درخت احتمال‌ها

درخت محاسبه Computation Tree

$N_1 \rightarrow 01011$

رشته ورودی مثال قبلی (دو صفحه قبل)



سه تا امکان داریم

وقتی در q_2 هستیم یه arc ε داریم

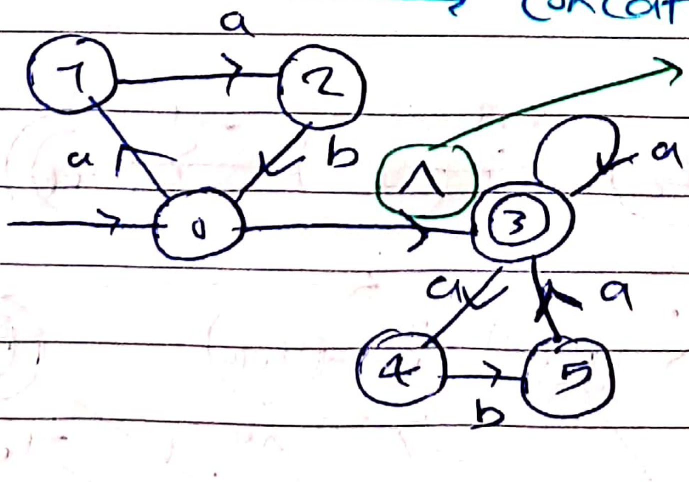
می‌توانیم بدون خواندن سمبل به q_3 بریم

نماد تهی ε

همه رشته‌هایی که ۱۰۱ یا ۱۱ را به عنوان زیررشته دارند.

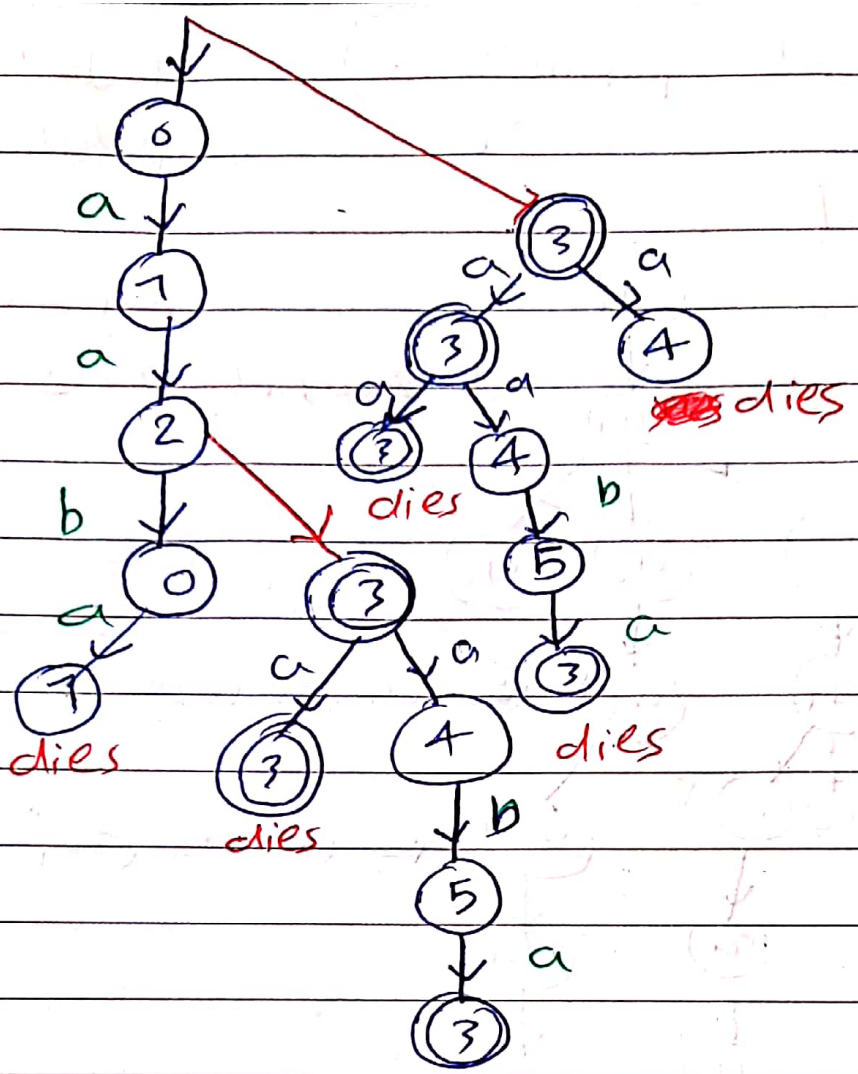
Ex1 $\{aab\}^* \{a, aba\}^*$ — concat

ما خوانیم aba

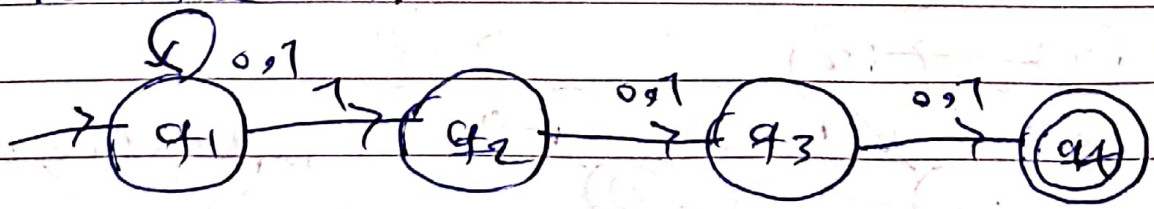


رشته تهی

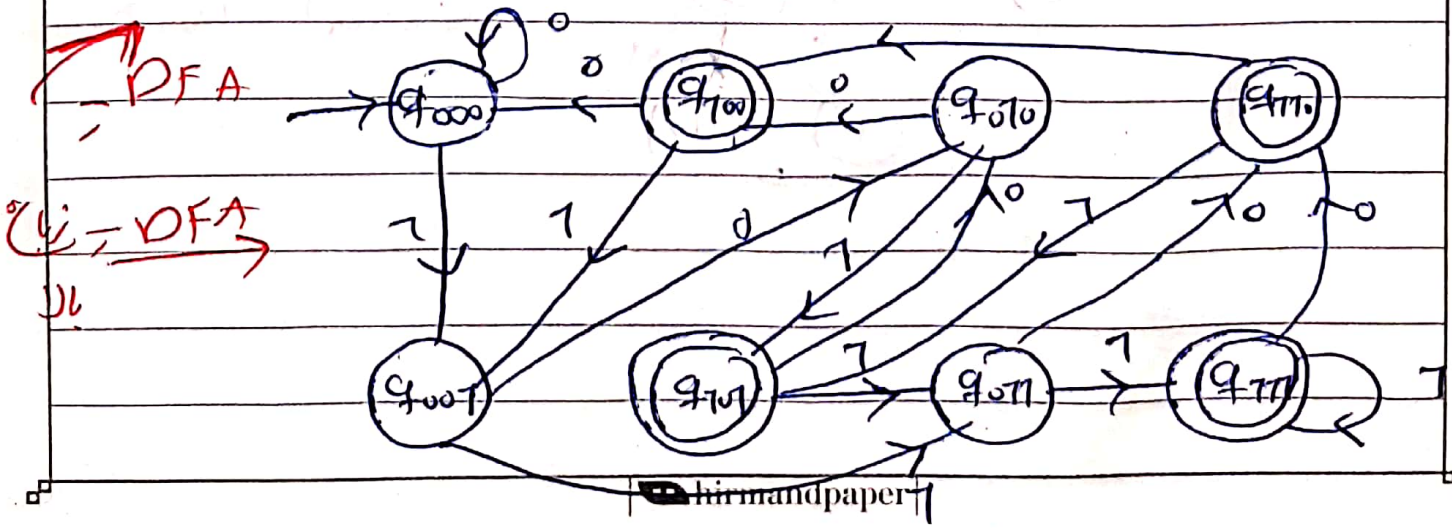
input string: aababa



ex: A is language consisting of all strings over {0,1} containing a 1 in the third position from the end.



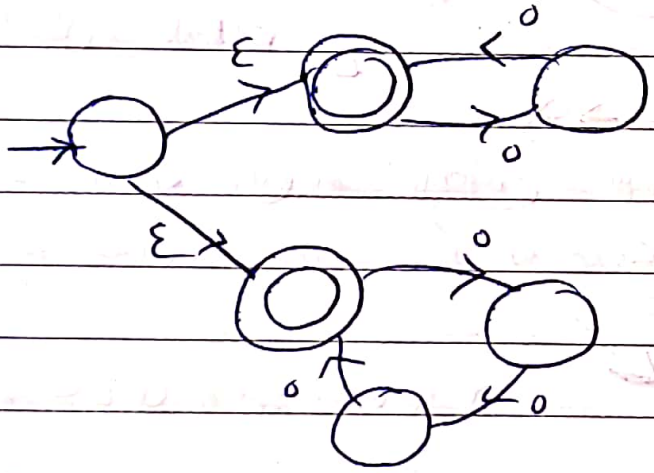
NFA



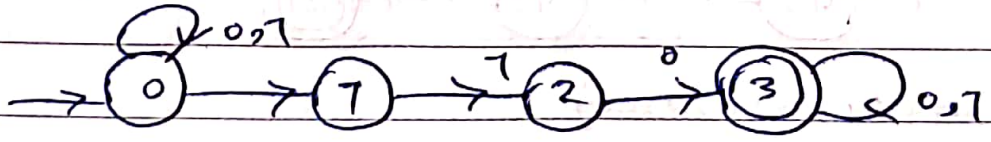
every NFA can be converted into an equivalent DFA

the smallest DFA for A contains 8 states.

ex1 this machine demonstrates the convenience of having ε arrows. it accepts all strings of the form of 0^k where k is a multiple of 2 or 3 it accepts strings ε, 00, 0000, ... but not 0, 00000



ex1 find an NFA that accepts the set of binary strings having a substring 010.



تعريف الى لـ "NFA"

a nondeterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

① Q is a finite set of states ② Σ is a finite alphabet

③ δ is $Q \times \Sigma_\varepsilon \longrightarrow P(Q)$ is the transition function

(5) $q_0 \in Q$ is the start state

(6) $F \subseteq Q$ is the set of accept states

* for any set Q we write $P(Q)$ to be the collection of all subsets of Q.
it can also be denoted by 2^Q → here $P(Q)$ is called the power set of Q.

$\Sigma_\epsilon \longrightarrow \Sigma \cup \{\epsilon\}$

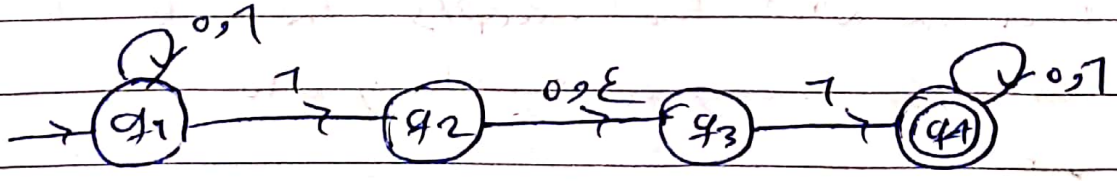
تابع انتقال در DFA

$\delta : Q \times \Sigma \to Q$

در یک NFA ممکن است با خواندن یک نماد چند جا برویم یا هیچ جا نرویم
← چنین اشاره دارد ← یک زیرمجموعه از Q است نه یک عضو

← powerset ← مجموعه همه زیرمجموعه‌های Q

$P(Q)$ یا 2^Q ← برد تابع transition



(1) $Q = \{q_1, q_2, q_3, q_4\}$

(2) $\Sigma = \{0, 1\}$ (3) δ is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

(4) q_1 is the start state

(5) $F = \{q_4\}$

تحويل NFA الى DFA

Deterministic and nondeterministic finite automata recognize the same class of languages.

① two machines are equivalent if they recognize the same language

② an NFA that accepts some language L → there exists an equivalent DFA that accepts L

③ خطوات تحويل NFA الى DFA

the ε-closure of a set of states

$M = (Q, \Sigma, \delta, q_0, F)$ is an NFA and $R \subseteq Q$ is a set of states,
the ε-closure of R is the $E(R)$ that can be defined recursively as follows

① $R \subseteq E(R)$

② for every $q \in E(R)$, $\delta(q, \varepsilon) \subseteq E(R)$

for any subset R of Q, we define E(R) to be the collection of states that can be reached from members of R by going only along ε arrows. for $R \subseteq Q$ → ε arrows.
E(R) = {q | q can be reached from R by traveling 0 or more

The Extended Transition Function δ^* for DFA

$\delta^* : Q \times \Sigma^* \rightarrow Q$

یک string
به جای اینکه symbol باشد

اگر $M = (Q, \Sigma, \delta, q_0, F)$ یک DFA باشد

extended function $\delta^* : Q \times \Sigma^* \rightarrow Q$:

① for every $q \in Q$, $\delta^*(q, \varepsilon) = q$ — base

② every $y \in \Sigma^*$ and every $\delta \in \Sigma$, $\delta^*(q, y\delta) = \delta(\delta^*(q, y), \delta)$ مثل یک تابع بازگشتی

اگر $M = (Q, \Sigma, \delta, q_0, F)$ یک DFA باشد و $x \in \Sigma^*$ ، رشته‌ی x توسط M قبول می‌شود اگر $\delta^*(q_0, x) \in F$ و در غیر این صورت رد می‌شود

از q_0 شروع کنیم در رشته‌ی x را ... بخوانیم ، اگر کار ما به یک state ختم شود که final باشد

the language accepted by M is the set

$L(M) = \{x \in \Sigma^* \mid x \text{ is accepted by } M\}$

زبان تفسیر M : string هایی که توسط M قبول می‌شوند ، آن string ها را قبول می‌کنیم

if L is a language over Σ, L is accepted by M if and only if $L = L(M)$

formal define of compution for NFA تعریف رسمی محاسبه برای

$N = (Q, \Sigma, \delta, q_0, F)$ is an NFA and w a string over the alphabet Σ. we say that N accepts w if $w = y_1 y_2 \dots y_m$ (y_i is a member of Σ_ε) and a sequence of states $r_0, r_1, \dots, r_m$ exists in Q with 3 conditions: ① $r_0 = q_0$ → کار را با state آغازین شروع کنیم

یا سمبلی از الفبا است یا ε

چرا عضویت داریم ولی در DFA تساوی داریم → چون اینجا یک set یا مجموعه داریم

② $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, 1, \ldots, m-1$

state ← به عبارت دیگر خارج از این یک set

③ $r_m \in F$

نباید وقتی توی r_0 هستی و y_1 را می‌خوانی
صرفاً توی r_1 ← r_1 باید داخل (مکان) r_0 و y_1 باشد

اگر $N = (Q, \Sigma, \delta, q_0, F)$ یک NFA باشد

we define extended transition function

$\delta^* : Q \times \Sigma^* \to 2^Q$ as follows:

کجا هستی — کجاها می‌توانی بری — تابع انتقال تعمیم‌یافته

نشان می‌دهد رشته را که بخوانی کجاها می‌توانی بری با یک string

اگر هیچی نخوانی ← همانجا هستی

① for every $q \in Q$, $\delta^*(q, \varepsilon) = E(\{q\})$. تعریف به شکل بازگشتی

② for every $q \in Q$, every $y \in \Sigma^*$, and every $\delta \in \Sigma$

سمبل آخر + یک رشته ← y = همهٔ سمبل‌ها به جز آخری

$$\delta^*(q, y\delta) = E(\cup \{\delta(p, \delta) \mid p \in \delta^*(q, y)\})$$

اجتماع ← ε-closure ← مجموعه جاهایی که به وسیله الفبا خوندن می‌توانیم بریم

q ← ε-closure: همه جاهایی که با کمک ε arrow ها می‌توانیم بریم

A string $x \in \Sigma^*$ is accepted by N if $\delta^*(q_0, x) \cap F \neq \emptyset$

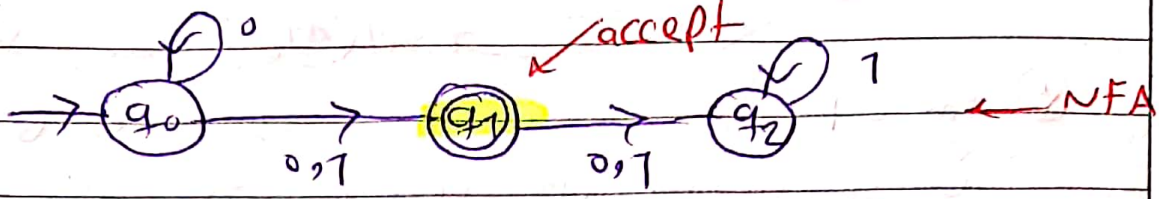
language $L(N)$ accepted by N is the set of all strings accepted by N.

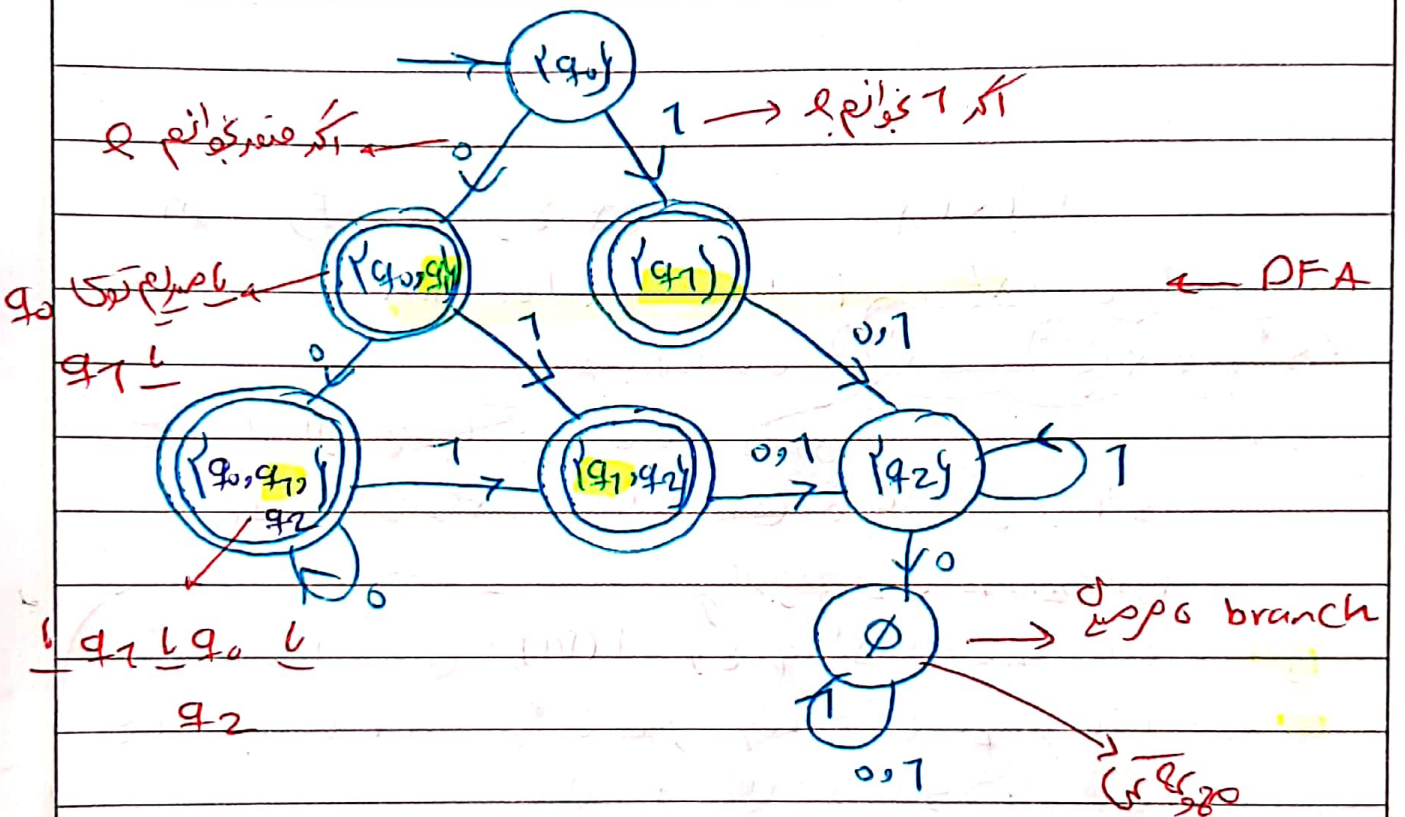
در مجموعه final قرار بگیرد ← x پذیرفته می‌شود

زبان تولید N — مجموعه همهٔ رشته‌هایی که توسط N پذیرفته می‌شود

مدت عبارتی NFA و DFA در یک جدول

ex





کام مجموعه ها accept میشن و اونایی که q_1 دارند

equivalence of DFA's & NFA's

every NFA has an equivalent DFA

for every language $A \subseteq \Sigma^*$ accepted by an NFA $N=(Q, \Sigma, \delta, q_0, F)$, there is a DFA $M=(Q', \Sigma, \delta', q_0', F')$ that also accepts A.

اول برای حالتی که arrow ε نداریم و انجام میکنیم بعدا ε را هم انجام میکنیم

مجموعه‌ی همه‌ی زیرمجموعه‌های Q

① $Q' = P(Q)$ every state of M is a set of states of N.

$P(Q)$ is the set of subsets of Q *

هر دارنده‌ی همه‌ی زیرمجموعه‌های مجموعه‌ی Q است

پس مثلاً اگر NFA ما ۳ تا state داشت، در DFA ما 2^3 تا داریم

اما همه‌ی آن‌ها لزوماً قابل دسترسی نیستند (state)

② for $R \in Q'$ and $a \in \Sigma$

یکی از state های Q'

$\delta'(R,a) = \{q \in Q \mid q \in \delta(r,a) \text{ for some } r \in R\}$

$\delta'(R,a) = \bigcup_{r \in R} \delta(r,a)$ به ازای هر کدام از r ها که در R هستند با سمبل a

$\delta'(R,a) = q_8 \; q_5 \; q_4$ ← اجتماع من به جاهای میرم

پس تابع transition DFA رو بر حسب NFA میسازیم

③ $q_0' = \{q_0\}$ ابتدای کار = مجموعه‌ی q_0

④ $F' = \{R \in Q' \mid R$ contains an accept state of $N\}$

مجموعه‌ی state های DFA در accept

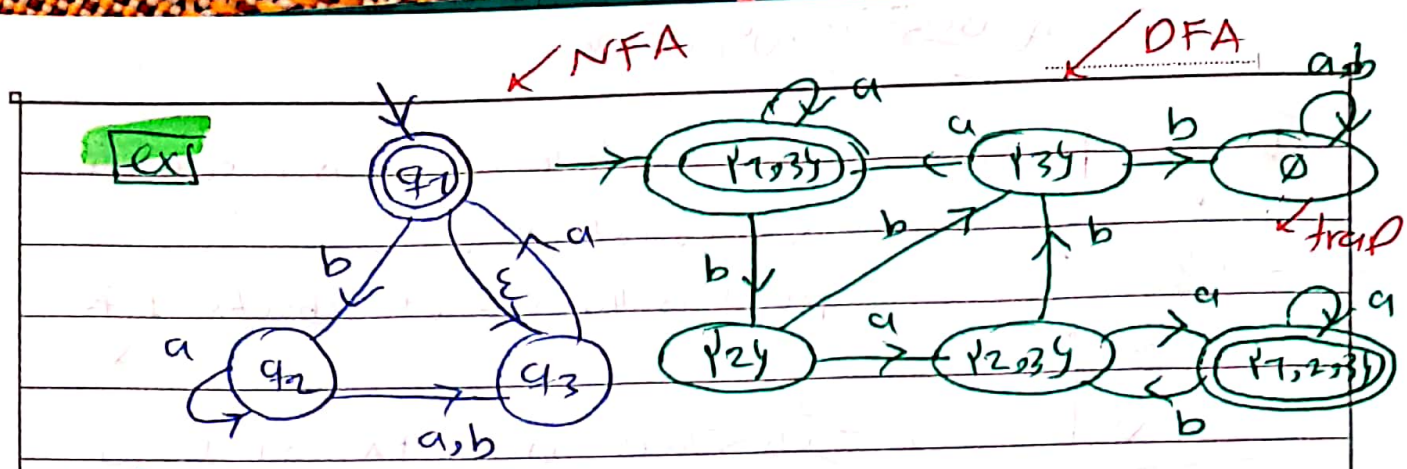
در نهایت کارمان به یک accept state ختم می‌شود.

A الان ε arrow اضافه می‌کنیم در انتها A

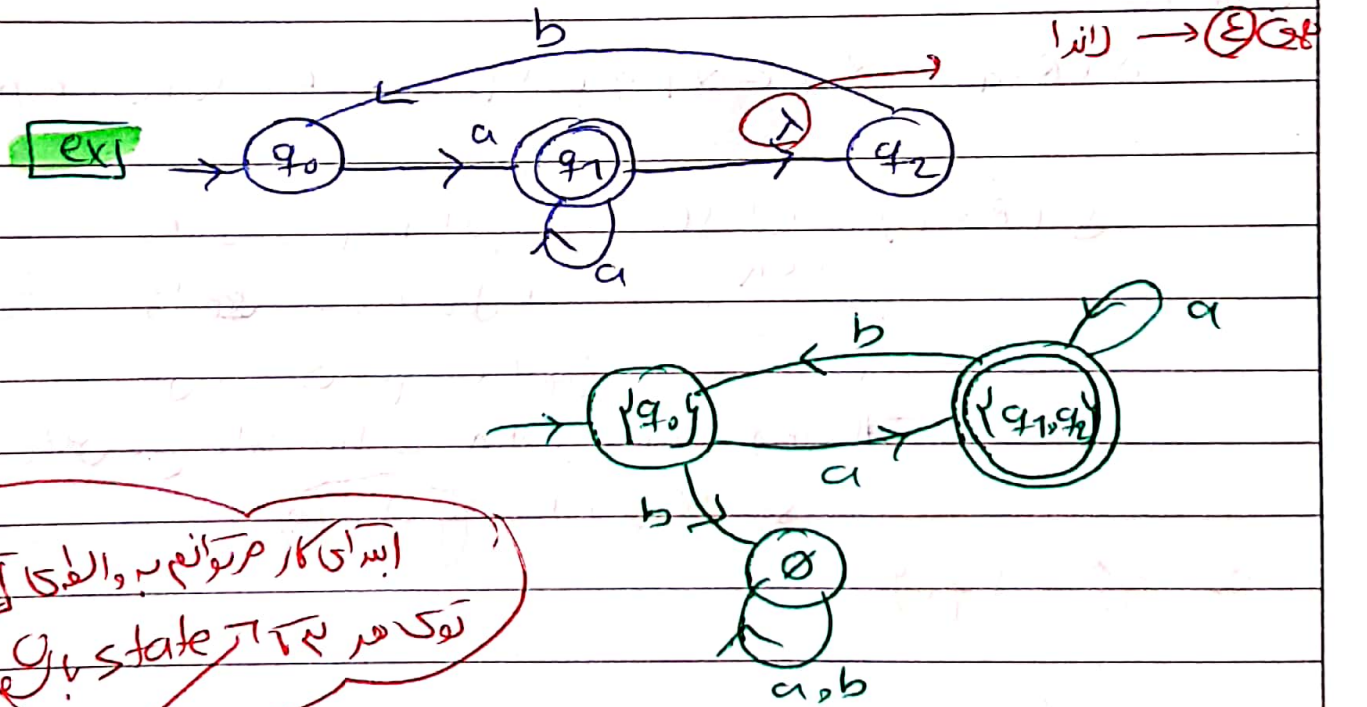
$\delta'(R,a) = \{q \in Q \mid q \in E(\delta(r,a)) \text{ for some } r \in R\}$

$= \bigcup_{r \in R} E(\delta(r,a))$. در ② جای $\delta(r,a)$ را $E(\delta(r,a))$ عوض کردیم

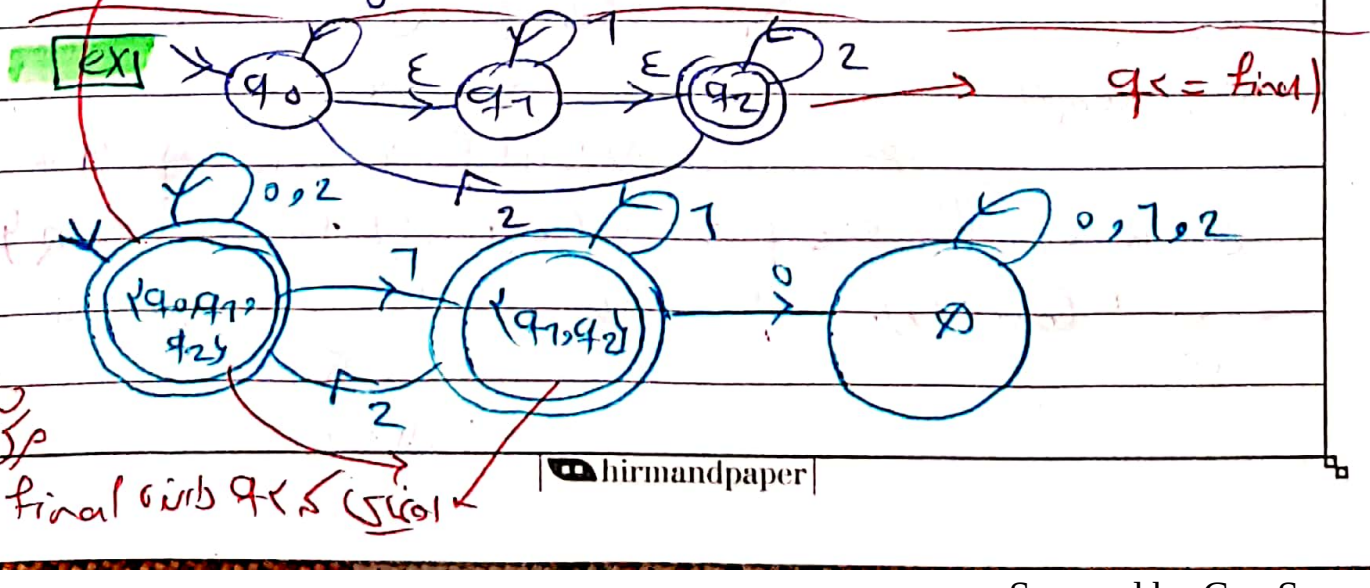
③ q_0 را $E(\{q_0\})$ تغییر می‌دهیم.



① ابتدای کار یا توی q_1 هستم یا q_3 ← اگه در q_1 یا q_3 باشم، $\delta(q_3, a) = q_1$ a نمیتونم چی بخونم؟

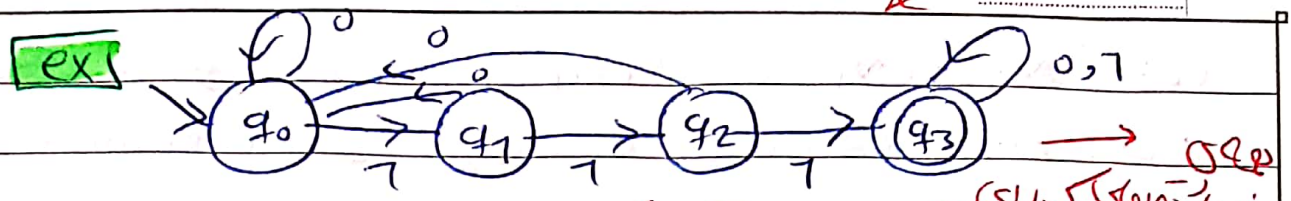


ابتدای کار میتونم به والدای ε توی هر state باشم

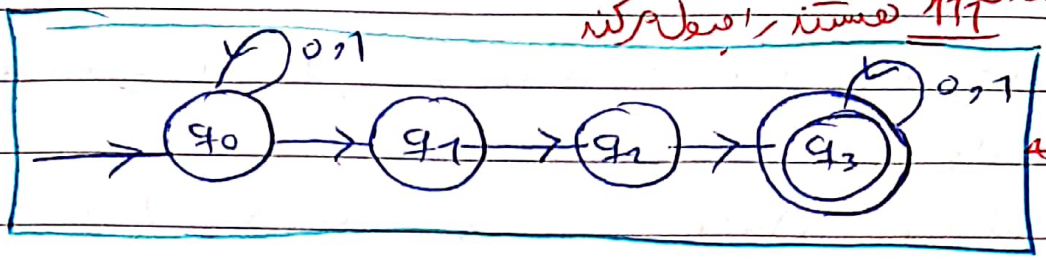


DFA

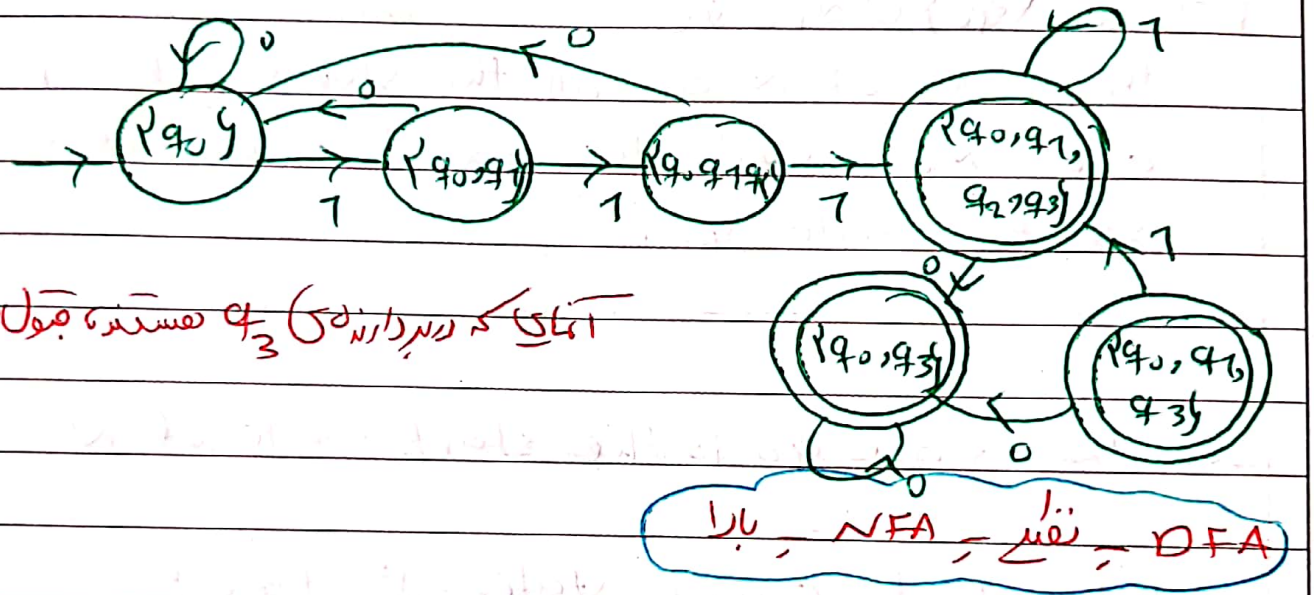
ex



زبانی که رشته‌هایی که دارای 111 هستند را قبول کند



NFA

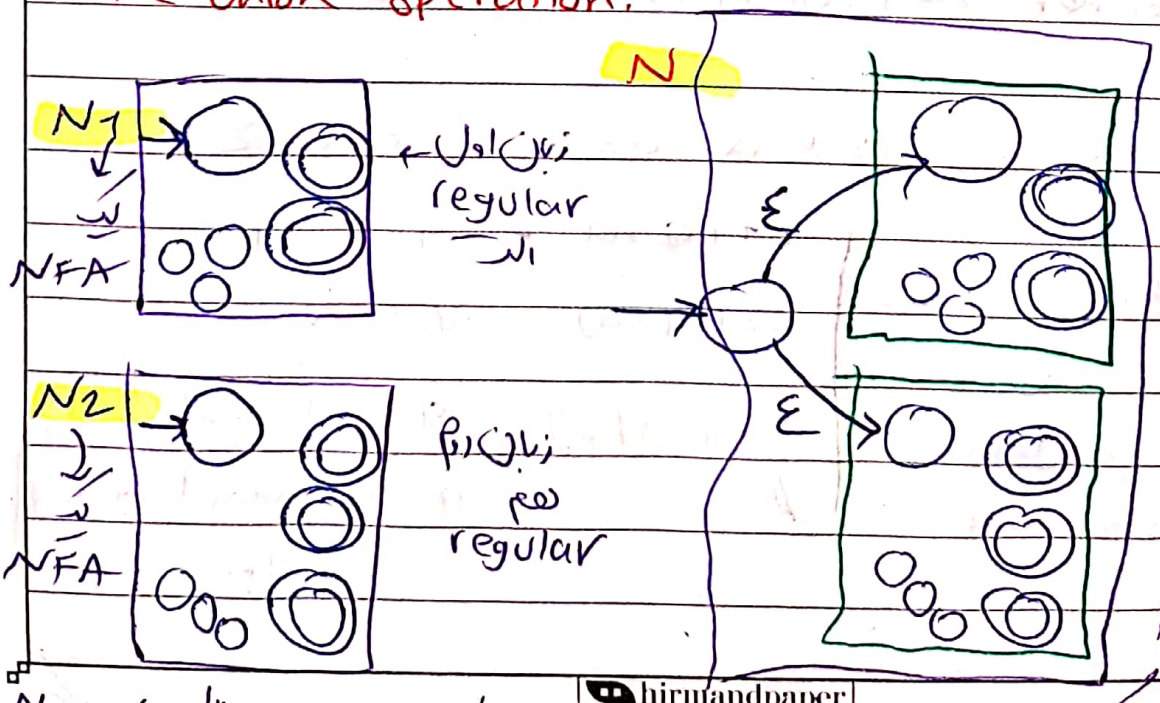


آنهایی که در پایان روی q_3 هستند قبول می‌شن

DFA ← نقیض ← NFA ← باز

---

the class of regular languages is closed under the union operation.



می‌خواهیم یک NFA داشته باشیم که زبان نقیض آن برابر اجتماع زبان نقیض N_1 و N_2 باشد

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$

① $Q = \{q_0\} \cup Q_1 \cup Q_2$

the states of N are all the states of N_1 and N_2, with the addition of a new start state q_0.

② the state q_0 is the start state of N

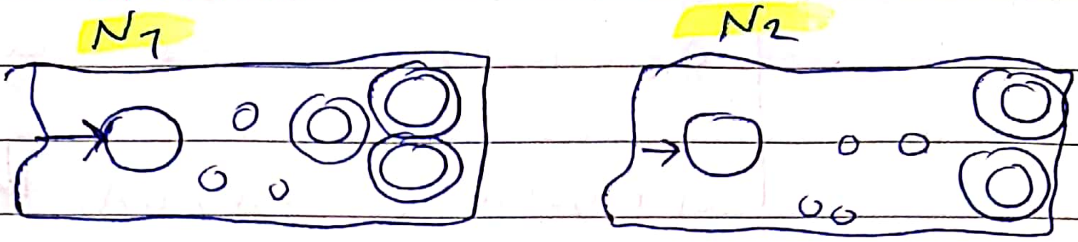
③ the set of accept states $F = F_1 \cup F_2$

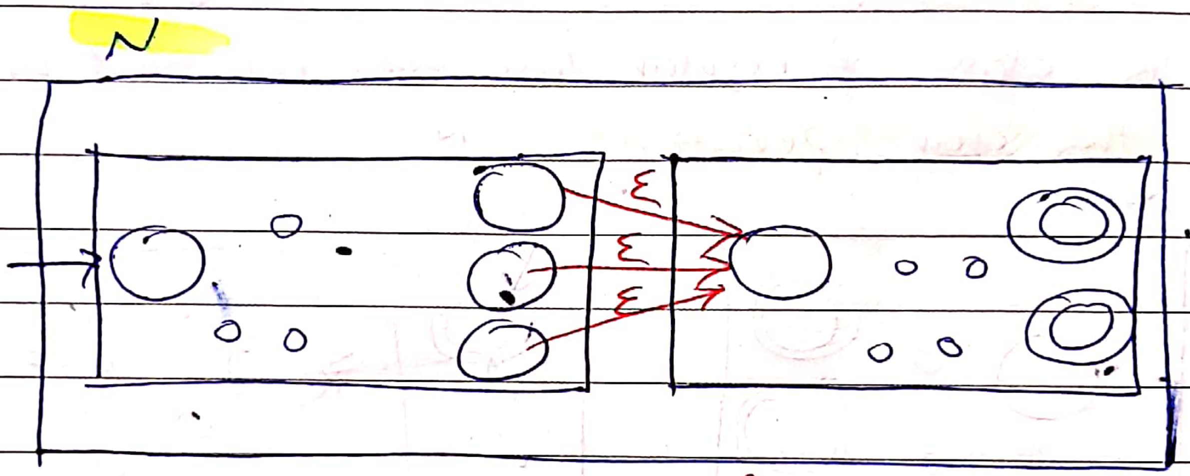
the accept states of N are all the accept states of N_1 and N_2. that way, N accepts if either N_1 accepts or N_2 accepts.

④ δ for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

The class of regular language is closed under the concatenation operation





رشته‌هایی که قسمت اولشون در N_1 الن، قسمت دوم هم توسط N_2، accept میشه. (باید توسط N_1، N_2 قبول بشن)

مجموعه فاینال‌های N، فاینال N2 الن

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 and

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ 〃 A_2

$N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

① $Q = Q_1 \cup Q_2$ start state

states of N are all the states of N_1 and N_2

② the state q_1 is the same as the start state of N_1.

③ the accept states F_2 are the same as the accept states of N_2.

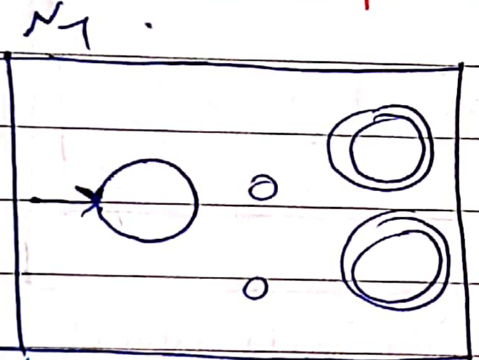
④ δ for any $q \in Q$ and any $a \in \Sigma_\varepsilon$

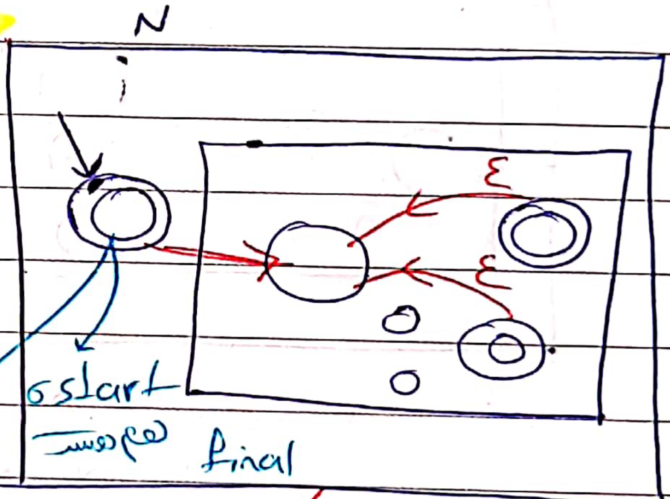
$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

(تابع ε فضا بعدی)

$A_1 \circ A_2$ یک زبان regular است.

the class of regular languages is closed under the star operation





باید تعلق A_1^* به یک NFA داریم (برای اثبات منظم بودن)
ثابت کنیم

از صفر تا ___ بار چسباندن یک رشته به خودش و این عکس کار می کند

A^* ستاره ی هر زبانی در بر دارنده ی ε هم هست ← صفر بار رشته های این زبان اصلی را به هم همواره بچسبانید $\boxed{\varepsilon}$ به دست میاد.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

$N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*

① $Q = \{q_0\} \cup Q_1$ the states of N are the states of N_1 plus a new start state

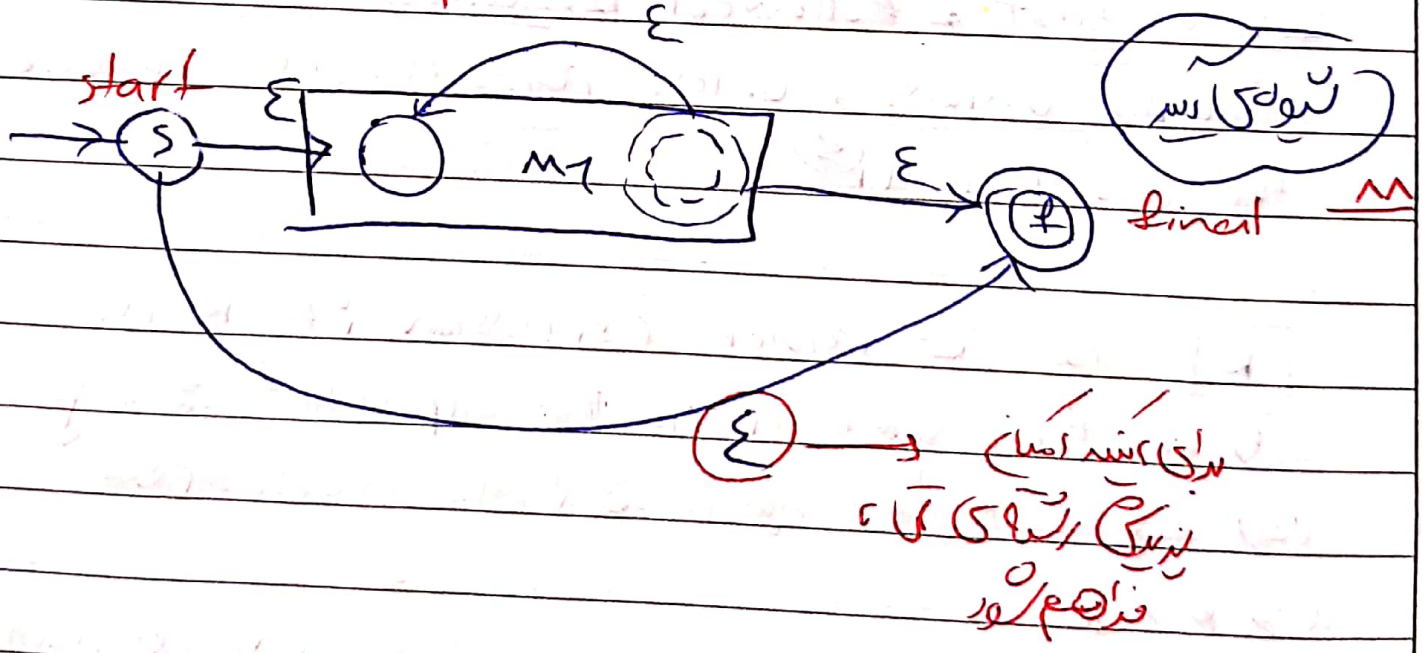
② the state q_0 is the new start state.

③ $F = \{q_0\} \cup F_1$

the accept states are the old accepts states plus the new start state.

④ δ for any $q \in Q$ and $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1,\ q \notin F_1 \\ \delta_1(q,a) & q \in F_1,\ a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1,\ a = \varepsilon \\ \{q_1\} & q = q_0,\ a = \varepsilon \\ \phi & q = q_0,\ a \neq \varepsilon \end{cases}$$



Scanned by CamScanner

«Regular expressions»

this notation involves a combination of strings of symbols from some alphabet Σ, parantheses, and the operators $\cup$, $\circ$ and $*$

we can use the regular ~~expressions~~ operations to build up expressions describing languages, which are called regular expressions. ex: $(0 \cup 1)0^*$

the value of a regular expression is a language

in regular expressions the star ① operation is done first, followed by concatenation ② and finally union, unless parantheses change the usual order ③

R is a regular expression if R is

① a for some a in the alphabet Σ,

② ε,

③ $\emptyset$,

④ $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions.

⑤ $(R_1 \circ R_2)$ where R_1 and R_2 are regular expressions, or

⑥ (R_1^*) where R_1 is a regular expression

the regular expressions a and ε represent the languages {a} and {ε}. regular expression ∅ represents the empty language

بجای عملگر (∪) می‌توان (+) هم گذاشت
و concat و نقطه (.) هم می‌گذارند.

* ∅, λ, a∈Σ are all regular expressions.
these are called primitive regular expressions.

* A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications

زبان نظیر عبارت منظم R را با L(R) نشان می‌دهیم.

(« زبان نظیر یک عبارت منظم »)

the language L(r) denoted by any regular expression r is defined by the following rules.

① ∅ is a regular expression denoting the empty set

زبان نظیر λ

② λ is a regular expression denoting {λ}

③ for every a∈Σ, a is a regular expression denoting {a}

if r_1 and r_2 are regular expressions then

④ $L(r_1+r_2) = L(r_1) \cup L(r_2)$

⑤ $L(r_1 \cdot r_2) = L(r_1)L(r_2)$

⑥ $L((r_1)) = L(r_1)$, ⑦ $L(r_1^*) = (L(r_1))^*$

$L(\emptyset) = \emptyset$

$L(a) = \{a\}$

$L(\lambda) = \{\lambda\}$

زبان نظیر

$a^* \cup b^*$ → رشته‌هایی که فقط a یا فقط b دارند

ex) $(0 \cup 1)^*$ → it starts with the language $(0 \cup 1)$ and applies the [*] operation. the value of this expression is the language consisting of all possible strings of 0s and 1s

همه‌ی رشته‌های متشکل از صفر و ۱

یک رشته‌هایی که از ۰ و ۱ درست کرده‌اند

if $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$.

if Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet, and Σ^* describes the language consisting of all strings over that alphabet.

ex) * $(\Sigma^* 1)$ is the language that contains all strings that end in a 1.

ex) * $(0\Sigma^*) \cup (\Sigma^* 1)$ consists of all strings that start with a 0 [or] end with a [1]

((ضرورت دقت هنگام مواجهه با عبارات منظم))

① * $a^* \cup b^* \neq (a \cup b)^*$

زبان سمت راست حاوی رشته‌های ab است در حالی که زبان سمت چپ حاوی فقط a ها یا فقط b ها است. زبان چپ نه و زبان راست نه.

② * $(ab)^* \neq a^* b^*$

زبان سمت چپ حاوی رشته‌ی abab است در حالی که زبان سمت راست حاوی رشته‌ی aaabbbbb است در حالی که زبان سمت چپ حاوی این رشته نیست.

if we let R be any regular expression, we have the following identities.

① $R \cup \emptyset = R$ → Adding the empty language to any other language will not change it.

② $R \circ \varepsilon = R$ joining the empty strings to any string will not change it.

③ $R \cup \varepsilon$ may not equal R. if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}$.

④ $R \circ \emptyset$ may not equal R. if $R = 0$, then $L(R) = \{0\}$ but $L(R \circ \emptyset) = \emptyset$.

$R \circ \emptyset = \emptyset$ → R ∘ ∅

R^*, R^+

we let R^+ be shorthand for RR^*.
whereas R^* has all strings that are 0 or more concatenations of strings from R,
the language R^+ has all strings that are 1 or more concatenations of strings from R.

→ $R^+ \cup \varepsilon = R^*$ we let R^k be shorthand for the concatenation of k R's with each other.

R^* → صفر او اكثر

R^+ → واحد او اكثر

$R^1 = 0$ او R^{10}

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

① $0^*10^* = \{w \mid w$ contains a single $1\}$ (یعنی یک 1 دارد)

② $\Sigma^*1\Sigma^* = \{w \mid w$ has at least one $1\}$ (حداقل یک 1 دارد)

③ $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string 001 as a substring$\}$

④ $1^*(01^+)^* = \{w \mid$ every 0 in w is followed by at least one $1\}$ (دنبال صفر نیست یا یکی داریم)

⑤ $(\Sigma\Sigma)^* = \{w \mid w$ is a string of even length$\}$ (جفت‌جفت رشته‌های با طول زوج)

⑥ $(\Sigma\Sigma\Sigma)^* = \{w \mid$ the length of w is a multiple of 3$\}$ (رشته‌های با طول مضرب 3)

⑦ $01 \cup 10 = \{01, 10\}$ (رشته‌های با طول 2 که یکی 0 و یکی 1)

⑧ $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w$ starts and ends with the same symbol$\}$

⑨ $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$

the expression $(0 \cup \varepsilon)$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^*.

⑩ $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$

⑪ $1^*\emptyset = \emptyset$ concatenating the empty set to any set yields the empty set.

(12) $\emptyset^* = \{\varepsilon\}$

The star operation puts together any number of strings from the language to get a string in the result. if the language is empty, the star operation can put together 0 strings, giving only the empty string.

ex1 $(0+1)^*110(0+1)^* = \{w \in \{0,1\}^* \mid$ w includes 110 as a substring $\}$

Concate

ex1 $(0+1)(0+1)(0+1)(0+1)^* = \{w \in \{0,1\}^* \mid$ the length of w is greater than or equal to 3 $\}$

ex1 $0+1+0(0+1)^*0+1(0+1)^*1 = \{w \in \{0,1\}^* \mid$ the first and the last symbols of w are the same $\}$.

ex1 $0^*1^*2^* = \{0^i 1^j 2^k \mid i \geq 0, j \geq 0, k \geq 0\}$

Regular expressions are useful tools in the design of compilers for programming languages. for example, a numerical constant that may include a fractional part and/or a sign may be described as a member of the language.

ε | _ | - | +   حداقل یک عدد

$$(+ \cup - \cup \varepsilon)\,(D^{+} \cup D^{+}.D^{*} \cup D^{*}.D^{+})$$ → ③

sign — علامت — concat

به صورت اولی عدد

توصیف می‌کنیم عدد‌های اعشاری را با یک عبارت منظم توصیف کنیم.

چهار ابزار اعشاری را ما زمانی

می‌توانیم نشان دهیم

② حالت صفر می‌تواند عددی نباشد / سمت چپ ممیز حداقل یک عدد باید باشد

③ سمت چپ ممیز عددی می‌تواند نباشد اما سمت راست باید حداقل یک عدد داشته باشد.

$D = \{0, 1, 2, \ldots, 9\}$ is the alphabet of decimal digits.

examples: 72, 3.14159, +7. , -07. (ممیز)

ex) the language of strings in $\{a, b\}^*$ with an odd number of a's

(فرد a)

جواب اشتباه: $b^*ab^*(ab^*a)^*b^*$ ✗ is incorrect, because it doesn't allow b's between the second a in one of the repeating pairs ab^*a and the first a in the next pair. one correct regular expression describing the language is

$b^*ab^*(ab^*ab^*)^*$ ✓ درست

hirmandpaper

Scanned by CamScanner

bas X

the expression $b^*a(b^*ab^*ab^*)^*$ is also not correct. because it doesn't allow strings with just one [a] to end with [b], and the expression $b^*a(b^*ab^*a)^*b^*$ corrects the mistake.

Another correct expression is $b^*a(b+ab^*a)^*$

All of these could also be written with the single [a] on the right, as in $(b+ab^*a)^*ab^*$

[ex] the language of strings in {a,b}* Ending with [b] and not containing [aa]

$(b+ab)^*(b+ab)$

$(b \cup ab)^+$ → حداقل یک بار ظاهر شود

چرا $(b+ab)^*$ نمی‌تواند جواب مسئله باشد؟

چون نمی‌تواند زبان بالا را توصیف کند کلا b خارج میشه

[ex] given a language, find a regular expression

Let $L = \{w \in \{a,b\}^* \mid |w| \text{ is even}\}$

regex is $((a \cup b)(a \cup b))^*$

this can be read as "Go through a loop zero or more times. each time through, choose an a or b then choose a second character (a or b)"

regex2: $(aa \cup ab \cup ba \cup bb)^*$

this can be read as " Go through a loop zero or more times, each time through, choose one of the two character sequences"

ex « more than one regex for a language »

let $L = \{w \in \{a,b\}^* : w$ contains an odd number of a's$\}$

مجموعه رشته های باینری که تعداد a های آن ها فرد باشد

two equally simple regular expressions that define L are:

a^{2k} یک واحد

1 روش $b^*(ab^*ab^*)^* a b^*$

2 روش $b^* a b^* (ab^*ab^*)^*$

(یک واحد a بیشتر)

* یک زبان منظم را با بی نهایت regex می توان توصیف کرد

ex for $\Sigma = \{0,1\}$ give a regex r such that

$L(r) = \{w \in \Sigma^* : w$ has at least one pair of consecutive zeros$\}$.

رشته های باینری که حداقل یک 00 باشد

$r = (0+1)^* 00 (0+1)^*$

complement مثال قبلی (نقیض مثال قبل)

ex find a regex for

$L = \{w \in \{0,1\}^* : w$ has no pair of consecutive zeros$\}$

رشته ای که دو تا صفر پشت سر هم نداشته باشد.

$r = (1^*011^*)^*(0+\varepsilon) + 1^*(0+\varepsilon)$

برای اینکه رشته هایی که تمام یک هم تو اش هست داشته باشیم ← برای اینکه صفر تو انتهای رشته حضور داشته باشد

→ concate شده

$r = (1 + 01)^* (0 + \varepsilon)$ مثال

* Generally, there are an unlimited number of regex for any given language *

Session 8

* regex and finite automata are equivalent in their descriptive power.

any regex can be converted into a finite automaton that recognize the language it describes.

Recall that a regex is one that is recognized by some finite automaton.

⇓

for every regex there is a regular language
And
for every regular language there is a regex.

if and only if

اثبات اینکه تقریبی هر عبارت منظم یک NFA دارد و به عبارت دیگر می‌توان

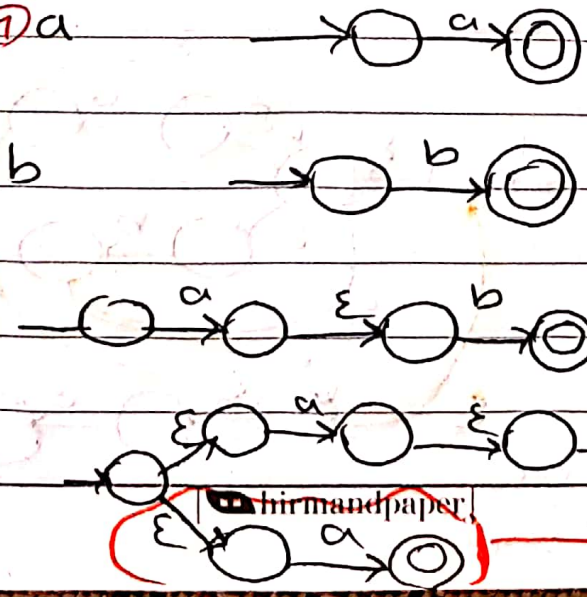
Ex

① a

② b

③ ab

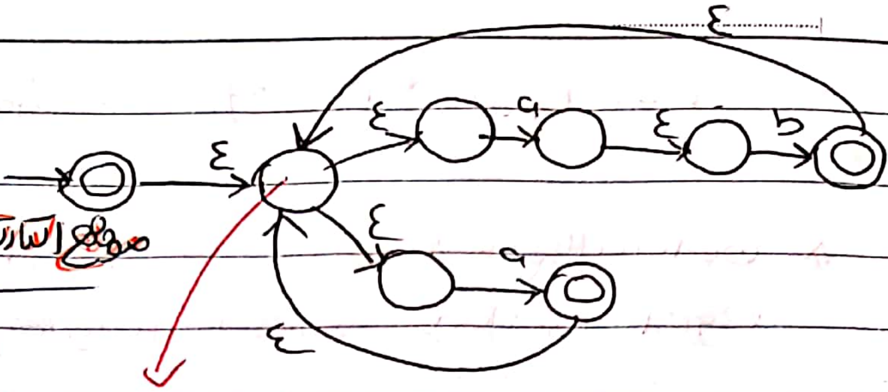
④ ab ∪ a



Building an NFA from the regex (ab ∪ a)*

تقریبی NFA

⑤ (ab ∪ a)*

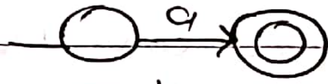


موقع استارت جدید

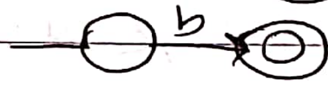
موقع استارت قدیم

Exp Building an NFA from regex (a∪b)*aba

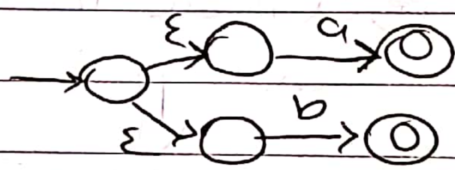
① a



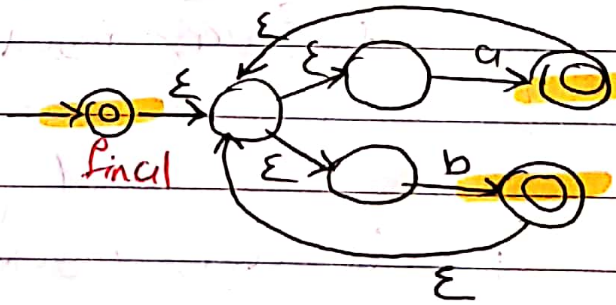
② b



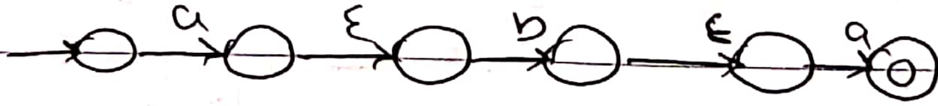
③ a∪b



④ (a∪b)*



⑤ aba

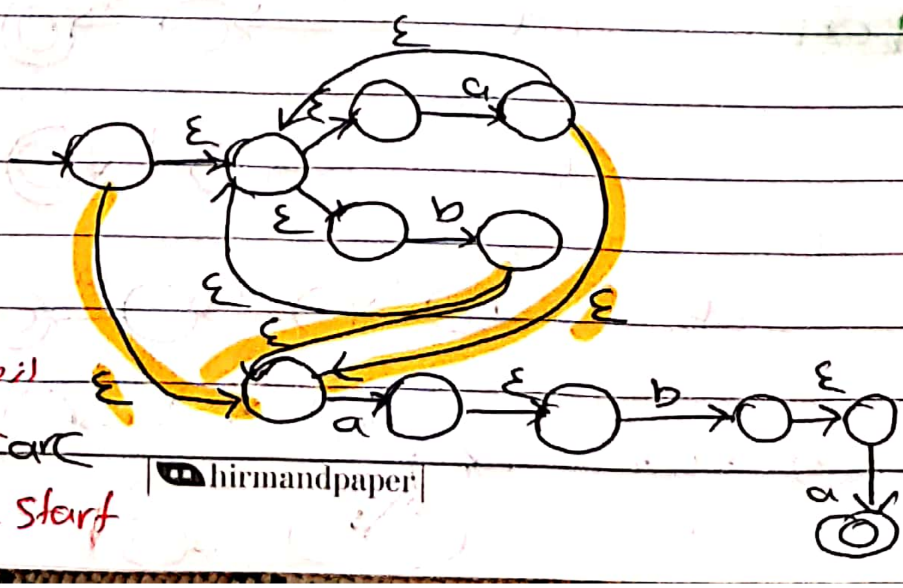


⑥ (a∪b)* aba

برای Concat کن

↓

ε یک از موقعیت فاینال اولی
arc ترسیم میکنیم به موقعیت
start = دوم



[ex] convert the regex to an NFA

$(abab)^* \cup (aaa^* \cup b)^*$

Lemma 8 if a language is regular, then it is described by a regex

* we need to show that if a language A is regular, a regular expression describes it

Because A is regular, it is accepted by a DFA.

We describe a procedure for converting DFAs into equivalent regex.

we break this procedure into two parts, using a new type of finite automaton called a generalized nondeterministic finite automaton

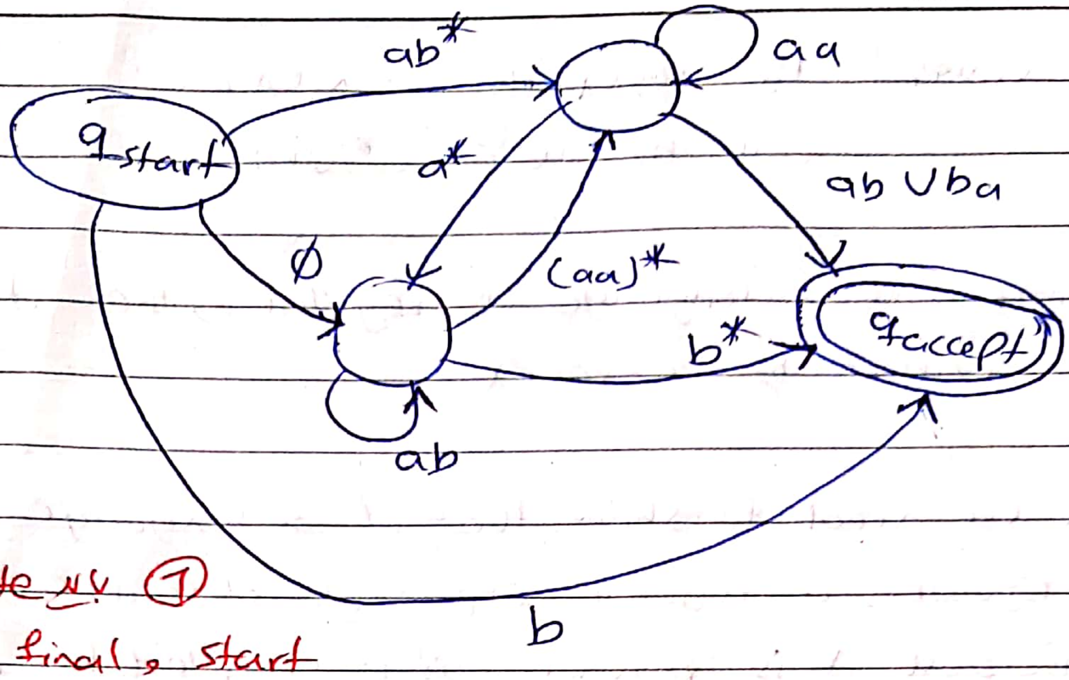
GNFA

① GNFA are nondeterministic finite automata wherein the transition arrows may have any regex as labels, instead of only members of the alphabet or ε.

* يختلف عن GNFA

② reads block of symbols from input, not just one symbol at a time as in an ordinary NFA.

③ A GNFA is nondeterministic and so may have different ways to process the same input string



① باید state ها
start و final یه واحد
داشته باشن

② به start یالی وارد نشود

③ از accept یالی خارج
نشود

④ از هر راس به راس دیگر
یال داشته باشیم

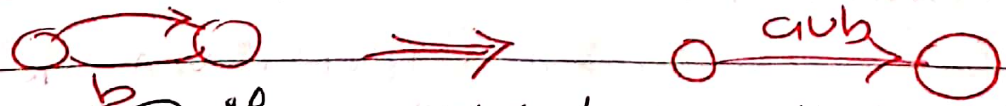
① تبدیل DFA به GNFA

② GNFA به یک regex معادل

* از start به همه state های دیگر
arrow داریم

How to ~~add~~ convert a DFA into a GNFA in the special form.

① we simply add a new start state with an ε arrow to the old start state and a new accept state with ε arrows from the old accept states



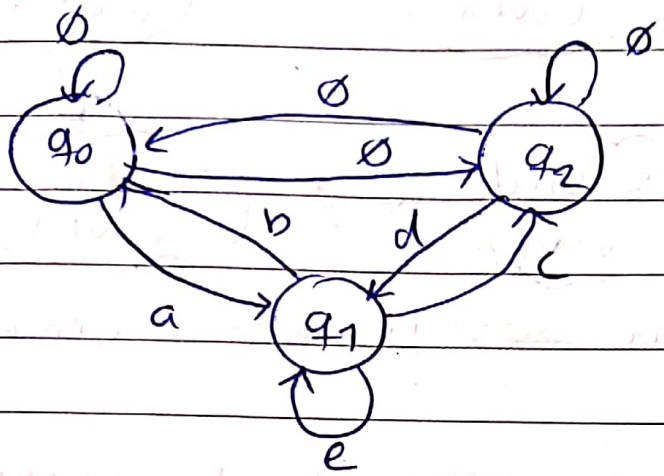
توضیح

② if any arrows have multiple labels (or if there are multiple arrows going between the same two states in the same direction), we replace each with a single arrow whose label is the union of the previous labels.

اجتماع می‌گیریم

* اضافه کردن label ∅ بین دو تا state که بال وجود ندارد

③ we add arrows labeled ∅ between states that had no arrows. this step won't change the language recognized because a transition labeled with ∅ can never be used.



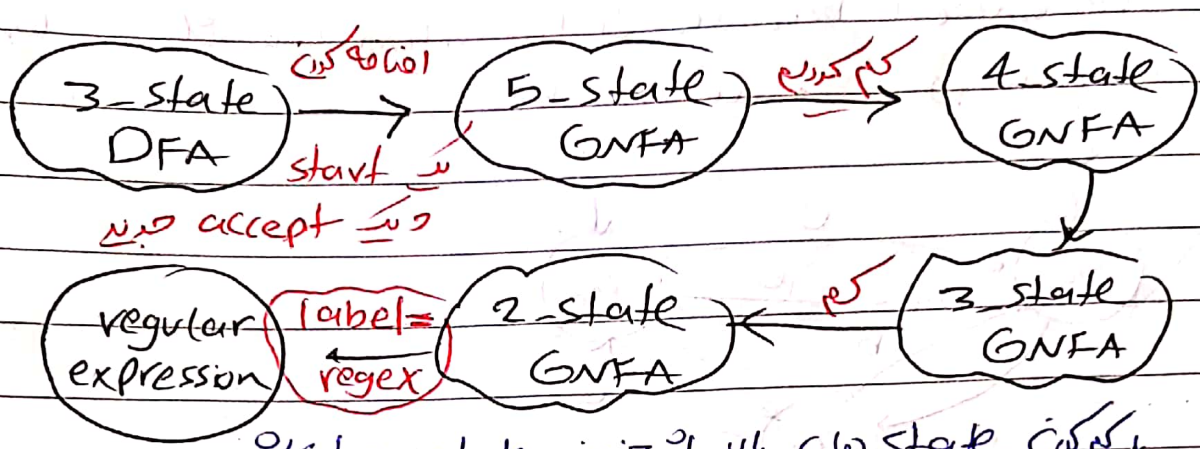
how to convert a GNFA to a regex?

① if GNFA has k states then because a GNFA must have a start and an accept state and they must be different from each other → $k \geq 2$

② if $k \geq 2$, we construct an equivalent GNFA with $k-1$ states

③ if $k=2$, the GNFA has a single arrow that goes from the start state to the accept state. the label of this arrow is the equivalent regex.

the stages in converting a DFA with three state to an equivalent regex:



با کم کردن state ها باید اثر حذف state را جبران کنیم.

the crucial step is constructing an equivalent GNFA with one fewer state when $k>2$.

* we do so by selecting (انتخاب می کنیم) a state, ripping it out (بیرون می اندازیم) of the machine, and repairing the remainder so that the same language is still recognized. باز هم بماند

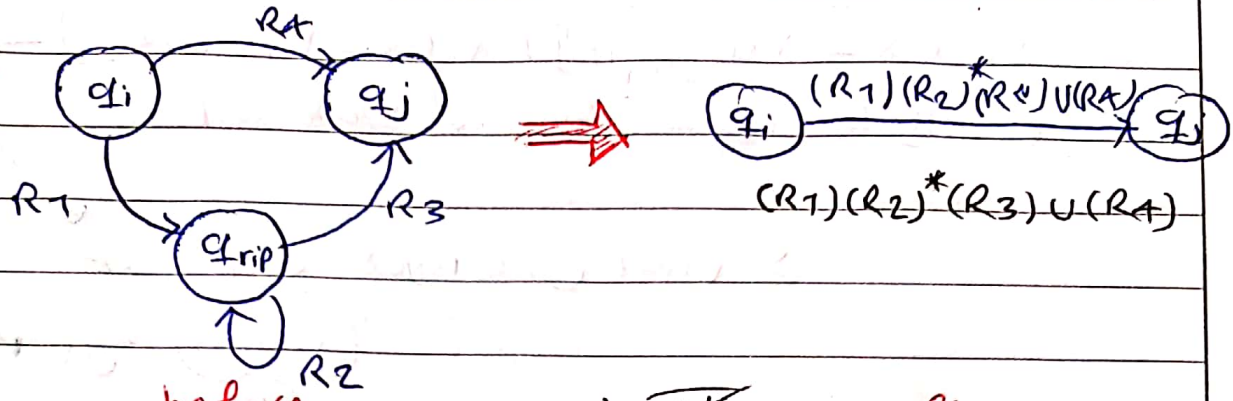
تا اثر حذف و جبران کنیم

* Any state will do, provided that it is not the start or accept state. we are guaranteed that such a state will exist because $k>2$. یک state را انتخاب می کنیم تا حذف کنیم باید غیر از start، accept باشد.

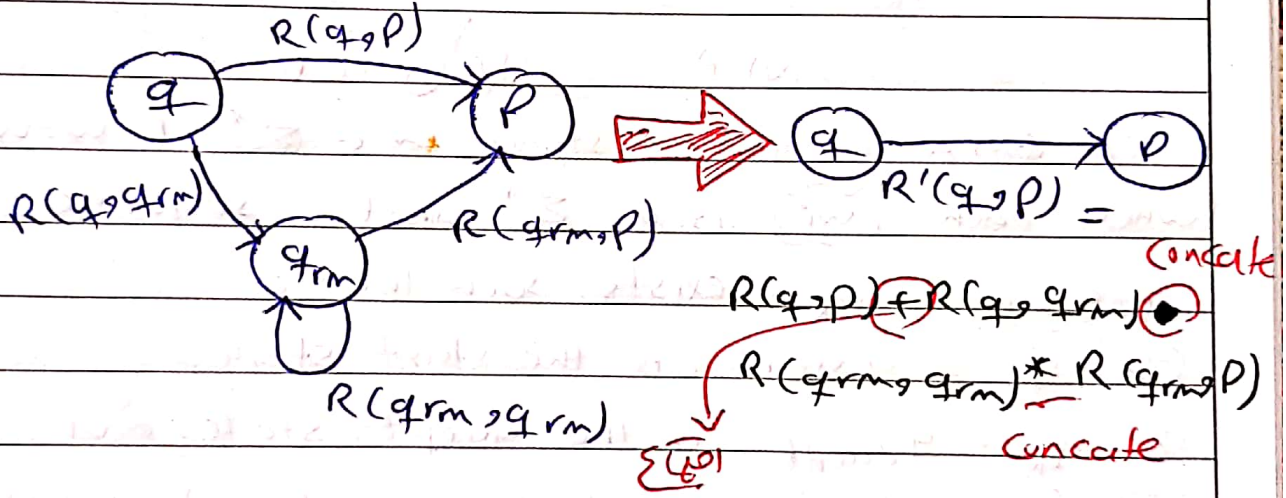
* lets call the removed state q_{rip}. After removing q_{rip} we repair the machine by altering the regex that label each of the remaining arrows. (state ها را که حذف می کنیم)

The new labels compensate for the absence of q_{rip} by adding back the lost computations.

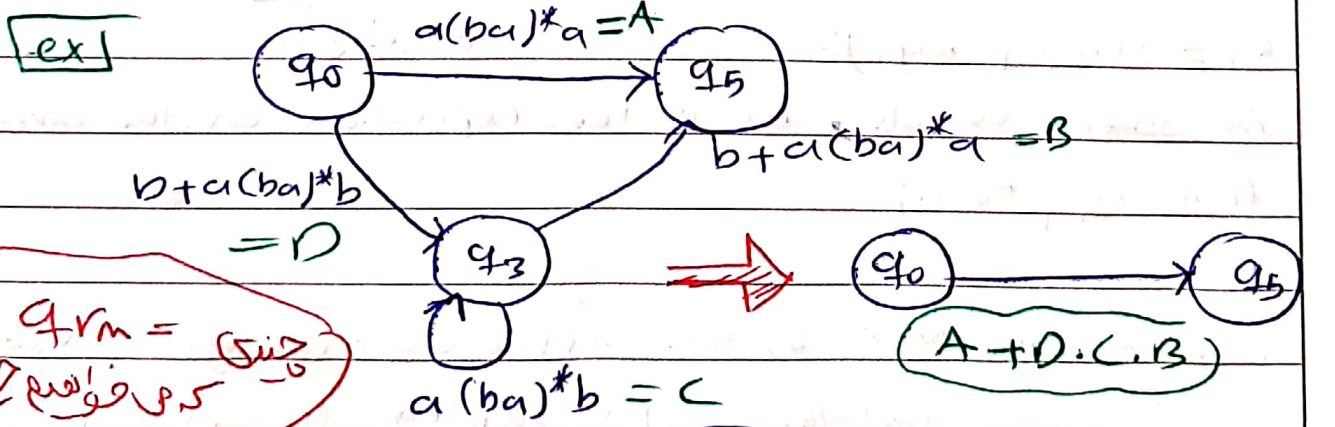
ex) Constructing an equivalent GNFA with one fewer state.



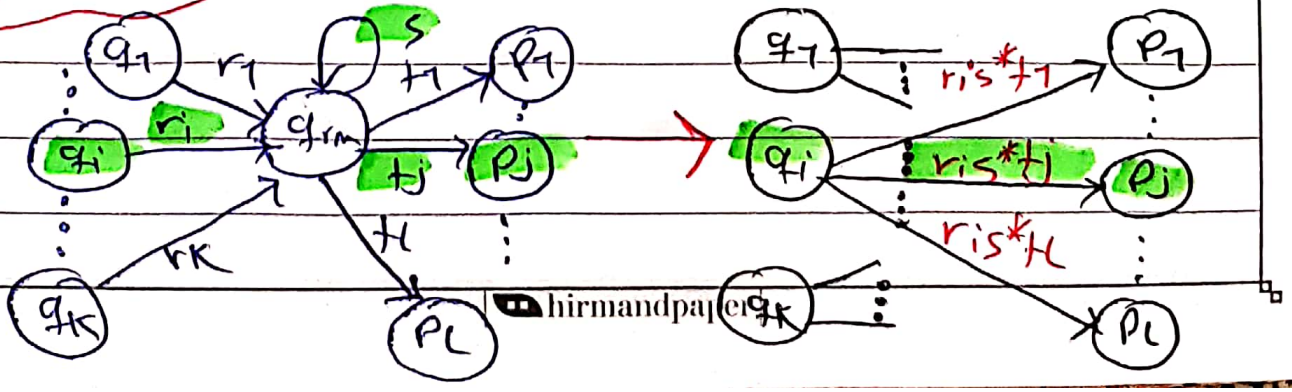
ex)



ex)



q_{rm} = جایی که حذف می‌شود



تابع انتقال یک GNFA به شکل زیر: formally:

state_پایان

state_آغاز

$\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \to R$

از هر state به state دیگر دقیقاً یک یال داریم

مجموعه همه regex ها روی الفبای Σ

R : یال (label)

تمرین: تابع انتقال یک GNFA را با تابع انتقال یک DFA و NFA ها مقایسه کنید

مفهوم پذیرش (رشته) برای یک GNFA

A GNFA accepts a string $w \in \Sigma^*$ if $w = w_1 w_2 \dots w_k$ where each w_i is in Σ^* and a sequence $q_0, q_1, \dots, q_k$ exists such that:

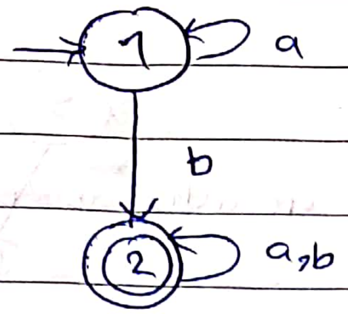
1. $q_0 = q_{start}$ is the start state.
2. $q_k = q_{accept}$ is the accept state, and
3. for each i, we have $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$; (زبان تفسیر R_i)

in other words, R_i is the expression on the arrow from q_{i-1} to q_i.

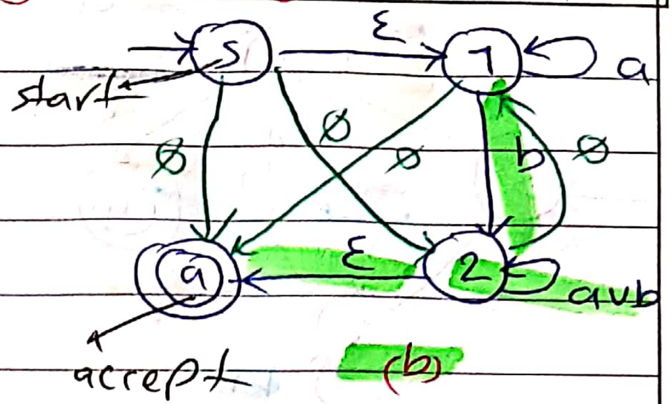
تمرین: تعریف رسمی بالا برای پذیرش یک رشته توسط GNFA را با تعریف رسمی مفهوم پذیرش برای DFA ها و NFA ها مقایسه کنید

DFA تبدیل به → NFA

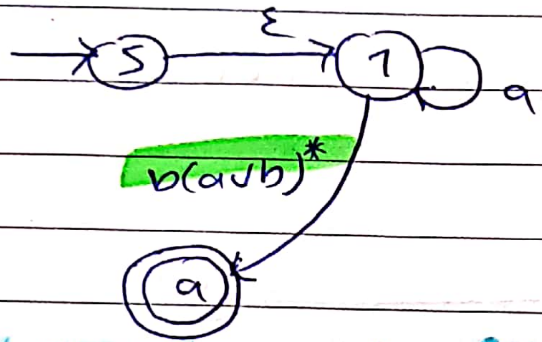
ex



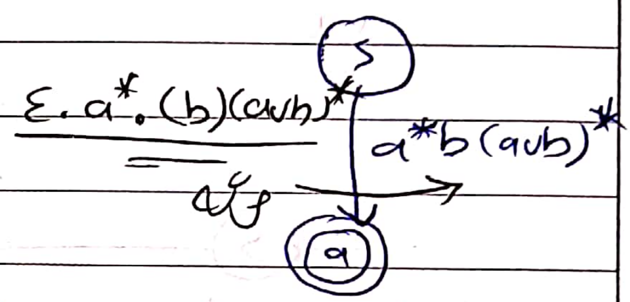
(a)



(b)



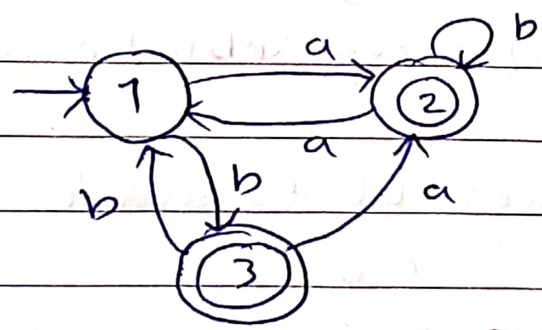
(c)



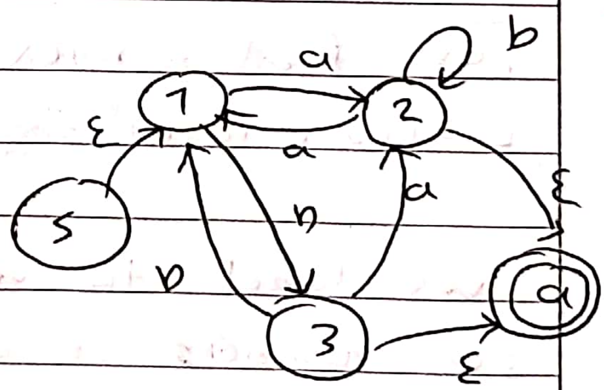
(d)

حذف state 2 • حذف state 1

ex



(a)



(b)

arc ها با label ε رسم نشده اند:

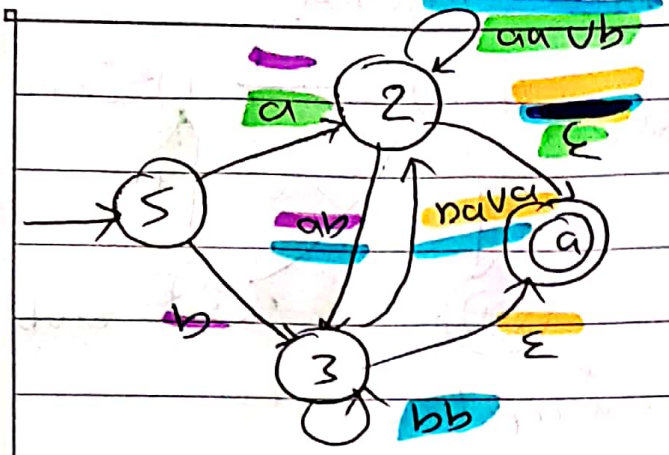
$\emptyset^* \Rightarrow \varepsilon$

مهم

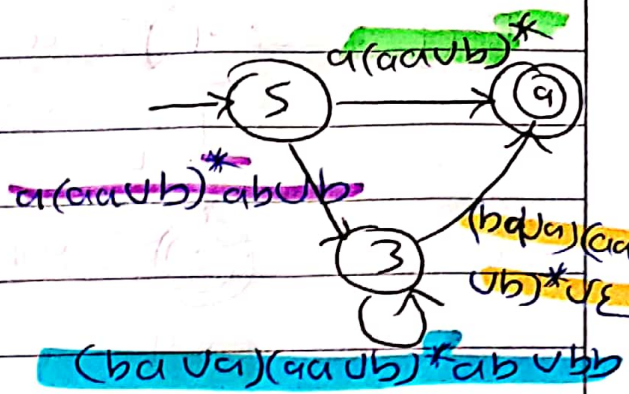
s → 2
s → 3
2 → 2
3 → 3
2 → 3
3 → 2

این arc ها باید اصلاح شود

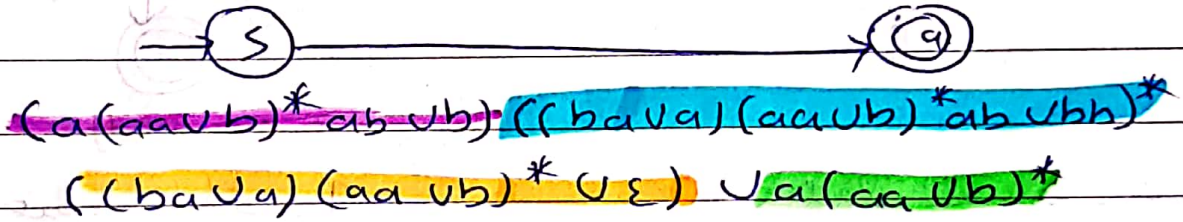
hirmandpaper



(c)



(d)



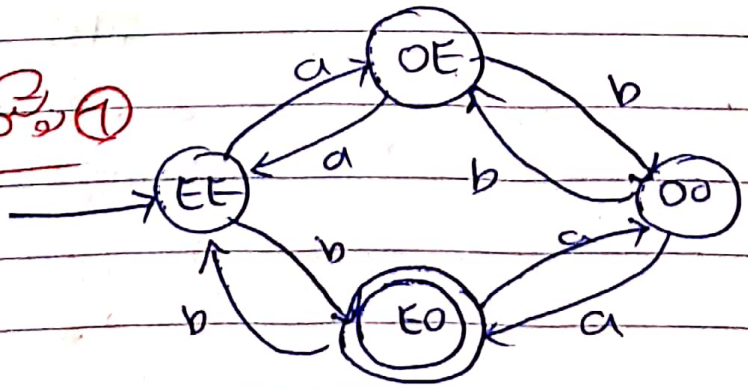
ex Find a regex for language

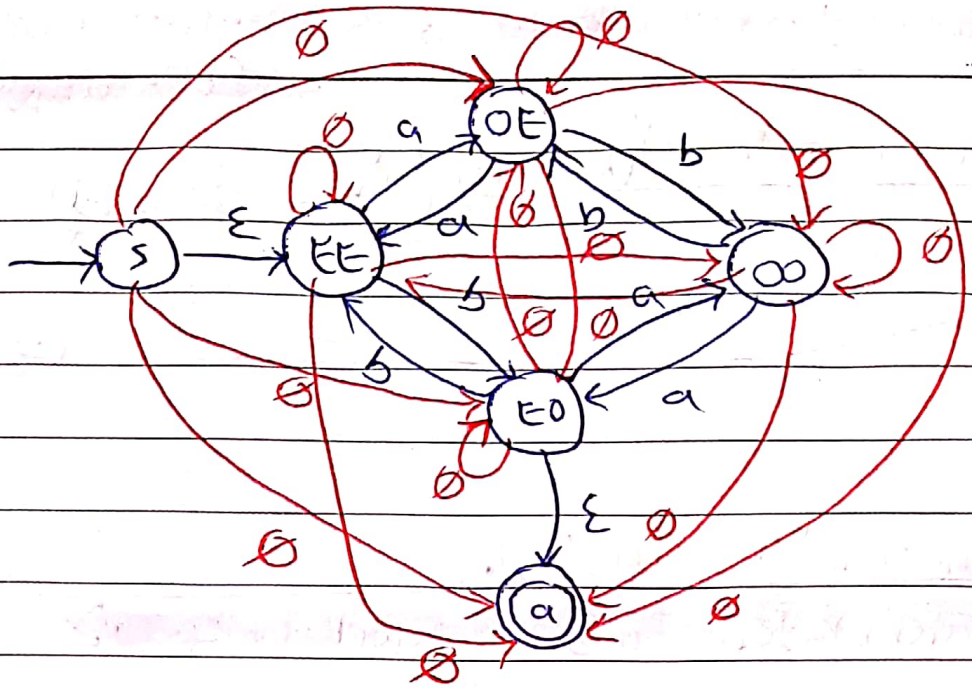
$L = \{w \in \{a,b\}^* \mid n_a(w) \text{ is even } n_b(w) \text{ is odd}\}$.

We label the vertices EE to denote an even number of a's and b's.

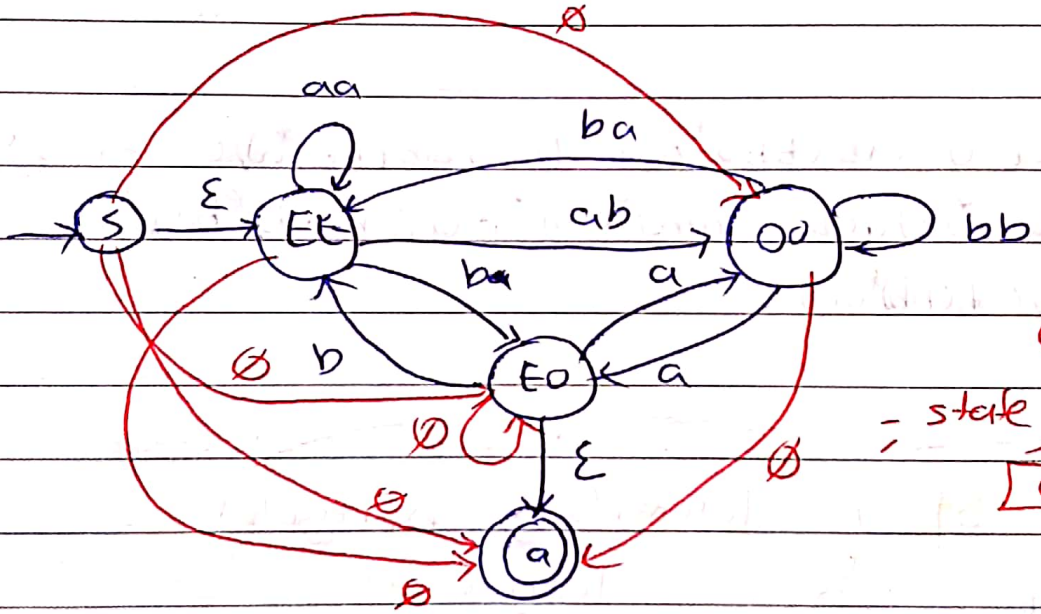
OE → odd number of a's & even number of b's

DFA ①





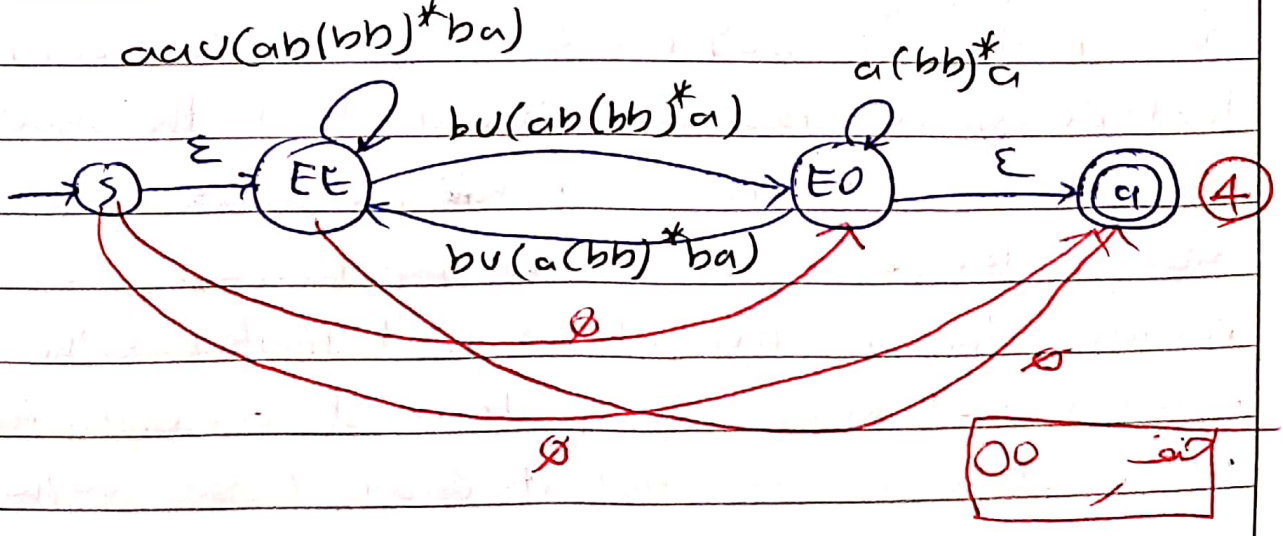
② تبدیل به DFA



③

- state حذف

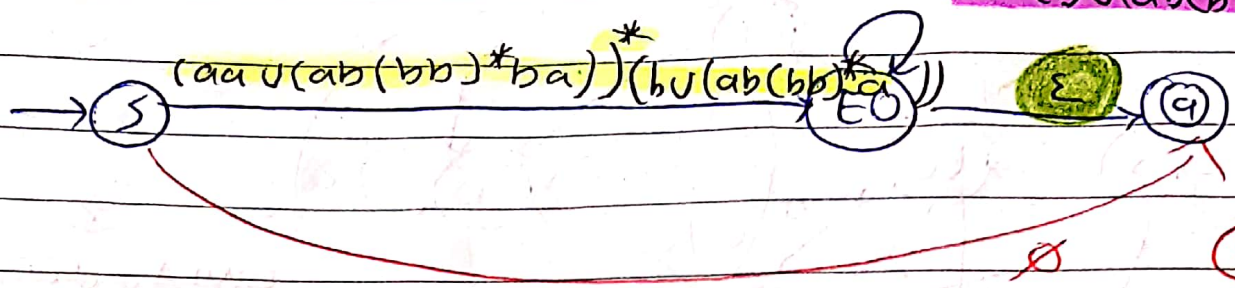
OE

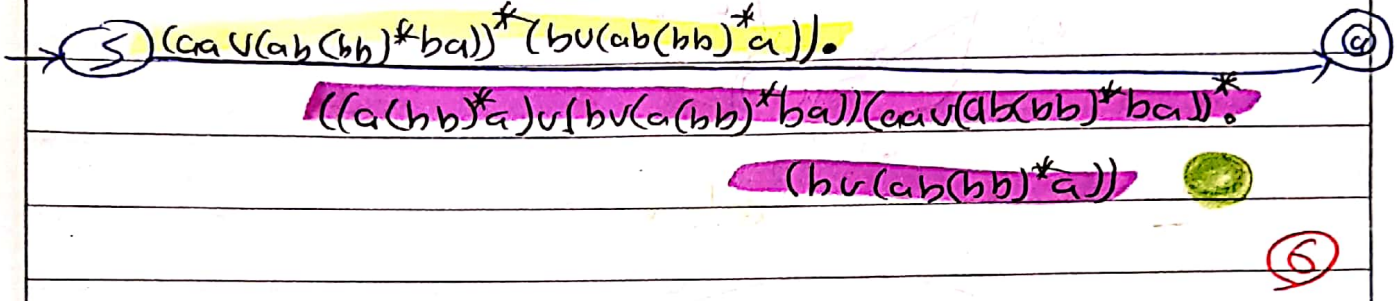


④

OO حذف

$(a(bb)^*a) \cup (bu(a(bb)^*ba))(aa \cup (ab(bb)^*ba))^*$
$(bu(ab(bb)^*a))$





$((a(bb)^*a) \cup (bu(a(bb)^*ba))(aa \cup (ab(bb)^*ba))^*.$
$(bu(ab(bb)^*a))$

6

$R = (aa \cup (ab(bb)^*ba))^*(bu(ab(bb)^*a))$ یا 7
$((a(bb)^*a) \cup (bu(a(bb)^*ba))(aa \cup (ab(bb)^*ba))^*.$
$(bu(ab(bb)^*a))$

session 8

Identifying nonregular languages

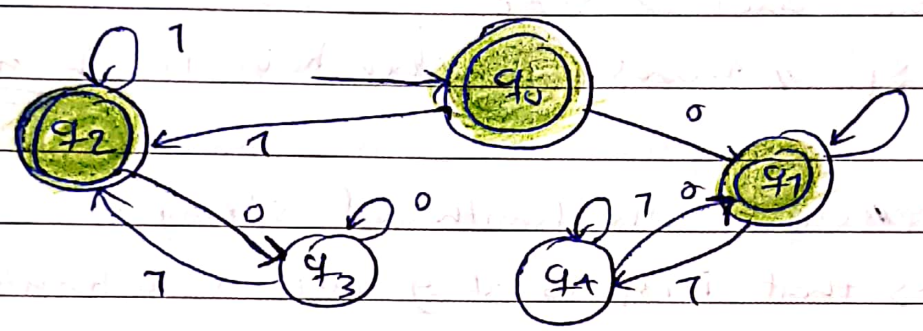
$B = \{0^n 1^n \mid n \geq 0\}$ if we attempt to find a DFA that recognizes B, we discover that the machine seems to need to remember how many 0's have been seen so far as it reads the input.
Because the number of 0's isn't limited → the machine will have to keep track of an unlimited number of possibilities. But it can't do so with any finite number of states.

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

ex) $C = \{w \mid w$ has an equal number of 0s and 1s$\}$ → is not regular

$D = \{w \mid w$ has an equal number of occurrences of 01 and 10 as substrings$\}$
→ is regular

D تفسیر DFA



the pumping Lemma for Regular languages

اثبات نامنظم بودن یک زبان با لم pumping به کمک برهان خلف

تناقض → contradiction

* اول فرض می کنیم زبان منظم است
سپس به یک تناقض می رسیم

ویژگی مشترک همه ی زبان های منظم :

all strings in the language can be "pumped" if they are at least as long as a special value called the pumping length.
that means each such string contains a section that can be repeated any number of times with the resulting string remaining in the language

کش میانی را می توانیم هی تکرار کنیم بدون اینکه خارج زبان بشه (همچنان جزو زبان حساب میشه)

★ ★ از برقرار نبودن لم تزریق می‌توانیم منظم نبودن یک زبان را نتیجه بگیریم ★

Pumping lemma : if A is a regular language, then there is a number p (the pumping length) where if s is string in A of length at least p, then s may be divided into 3 pieces $(s = xyz)$ satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$
2. $|y| > 0$ and // $y \neq \varepsilon$
3. $|xy| \leq p$ // x and y together have length at most p

* $|s|$ represents the length of string s
y^i means that i copies of y are concatenated together and y^0 equals to ε
when s is divided into xyz, either x or z may be ε but condition 2 says $y \neq \varepsilon$.

(نامتناهی) — یک بیان دیگر برای لم تزریق — (وجود دارد)

Let A be an infinite regular language then there exists some positive integer p such that any $s \in A$ with $|s| \geq p$ can be decomposed as $s = xyz$ with $|xy| \leq p$ and $|y| \geq 1$ such that xy^iz is also in A for all $i = 0, 1, 2, \ldots$

every sufficiently long string in A can be broken into three parts in a way that an arbitrary number of repetitions of the middle part (قسمت میانی y) yields another string in A → we say that the middle string is "pumped" → hence the term pumping lemma for this result.

* لم تزریق برای هر زبان منظمی برقرار است اما لزومی ندارد اگر لم تزریق برقرار بود ، زبان منظم باشد
عکس نقیض درست نیست

نکته مهم: ما حق تعیین تجزیه را نداریم. یعنی حق نداریم که x و y و z چی هست یا چی نیست

ما صرفاً می‌دانیم که یک چنین تجزیه‌ای که ۳ شرط را ارضا کند وجود دارد. اینکه دقیقاً چیست را نمی‌دانیم. ← استدلال ما نباید مبتنی بر تعیین تجزیه باشد

اثبات: اینکه لم تزریق برای زبان‌های منظم برقرار است. ؟!

صورت لم خیلی مهم است ***

proof: let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and $p = |Q|$. let $s = s_1 s_2 \ldots s_n$ be a string in A of length n where $n \geq p$.

let $r_1, r_2, \ldots, r_{n+1}$ be the sequence of states that M enters while processing s. so $r_{i+1} = \delta(r_i, s_i)$ for $1 \leq i \leq n$. this sequence has length $n+1$ which is at least $p+1$.

Among the first $p+1$ elements in the sequence, two must be the same state, by the Pigeonhole principle. ← اصل لانه کبوتری

we call the first of these r_j and the second r_l. Because r_l occurs among the first $p+1$ places in a sequence starting at r_1, we have $l \leq p+1$.

now let $x = s_1 \ldots s_{j-1}$, $y = s_j \ldots s_{l-1}$, $z = s_l \ldots s_n$

از z خارج می‌کنیم و به اولی می‌رویم

As x takes M from r_1 to r_j, y takes M from r_j to r_j, z takes M from r_j to r_{n+1}, which is an accept state → M must accept xy^iz for $i \geq 0$. we know that $j \neq l$ so $|y| > 0$; and $l \leq p+1$ so $|xy| \leq p$. →

همه‌ی شرط‌های لم تزریق را ارضا کردیم.

examples of pumping lemma

اثبات نامنظم بودن یک زبان به کمک لم تزریق

ex1 B is the $\{0^n 1^n \mid n \geq 0\}$ language

① فرض می‌کنیم B یک زبان منظم است (فرض خلف) ← ② یک p داریم که رشته‌های با طول حداقل p را می‌توان pump کرد

③ اگر $s = 0^p 1^p$ ← چون s داخل زبان B است و طولش بیشتر از p است (طول $s = 2p$) ← s را می‌توان pump کرد و به ۳ قسمت شکست ← ④ $s = xyz$ → for any $i \geq 0$ the xy^iz is in B

$|xy| \leq p$

$|y| > 0$

⑤ بخاطر شرط $|xy| \leq p$ ← $|xy|$ است از p سمبل اول s که 0s صفر هستند → all the symbols in x and y must be 0s.

⑥ در نتیجه ← $y = 0^k$ طبق شرط‌های بالا $k > 0$

⑦ طبق شرط یک که می‌گفت به ازای هر i، xy^iz باید داخل زبان باشد در حالی که اینجا xy^iz تعداد صفرها و یک‌ها با هم فرق می‌کند

مثلاً xy^2z ← $0^{p+k}1^p$ است ← با یک بار pump کردن 0y

باید رشته‌ی حاصل داخل زبان بیفتد اما نمی‌افتد ← تناقض داریم.

because the pumping lemma says $xy^2z \in L$ but $p+k \neq p$ → این تناقض نشان می‌دهد که فرض ما مبنی بر منظم بودن زبان درست نبوده ← فرض خلف باطل و در نتیجه به تناقض می‌رسیم.

زبان نامنظم است.

example2 let $C = \{w \mid w$ has an equal number of 0's and 1's$\}$ ①

assume (to the contrary) that C is regular. ② let p be the pumping length given by the pumping lemma and ③ let s be the 0^p1^p.

④ with s being a member of C and having length more than p, the pumping lemma guarantees that s can be split into three pieces → $s = xyz$ where for any $i \geq 0$ the string xy^iz is in C.

we would like to show that this outcome is impossible Because $|xy| \leq p$ (by condition 3) of y must consist only of 0's so $xyyz \notin C$ → therefore, s cannot be pumped → (تناقض) → (فرض خلف) → (نامتناهی است)

bic

(چون تعداد صفرها بیش از تعداد یک‌هاست)

* نکته: اینجا رشته مهم است *

منظور اینکه برای مثال s را $s = (01)^p$ در نظر گرفتیم و بررسی کردیم چون این رشته pump می‌شود و ما رشته‌ای می‌خواستیم که pump نشود منظور اگر $m = 2$، $y = 01$، $z = (01)^{p-1}$ ← برای این رشته $xy^iz \in C$ for every value of i

* Using closure properties *

An alternative way (method) of proving that C is nonregular follows from our knowledge that B is nonregular. if C were regular, $C \cap L(0^*1^*)$ also would be regular. → the reasons are that the language

$L(0^*1^*)$ is regular and that the class of regular language is closed under intersection.
But $C \cap L(0^*1^*)$ equals B. and we know that B is non-regular.

* گاهی اوقات کار بهتری راه برای اثبات نامنتظم بودن یک زبان ای است که از closure theorems استفاده کنیم. ← بسته بودن تحت اشتراک و complement
خواص بستاری
یکسه اول و دوم مثل هم

example 3 $F = \{ww \mid w \in \{0,1\}^*\}$ → مثلا $\boxed{001001}$

(1) فرض می‌کنیم F منتظم است ← (2) لم تزریق وعده می‌دهد که یک $\boxed{P}$ وجود دارد
که رشته‌های زبان به طول حداقل p را می‌شود pump کرد

(3) $S = 0^P 1 0^P 1$ ← این رشته داخل زبان هست چون تکه اول و تکه دوم برابر هستند
برابر

(4) طول رشته $S = 2P + 2 > P$

→ pumping lemma طبق ← در سه تکه می‌کنیم → $S = xyz$

تناقض ها چون y نمایی است و P تا کاراکتر اول رشته s همگی صفر هستند و طبق لم می‌گوید که $\boxed{|xy| \le P}$ ← پس x و y هر دو از صفر تشکیل شده اند ← y فقط از صفر درست شده

$xyyz \notin F$ چون $xyz = 0^P 1 0^P 1$

اگر $y = 0^K$ و $K \ge 1$

$xy^2z = 0^{P+K} 1 0^P 1$ ← xy می‌کنیم

به فرم ww نیست ← $\boxed{\notin F}$

تناقض است ← مخالف فرم ww

is a member of F

برای این مثال اگر s را 0^p0^p بگیریم و حاصل pumping داخل زبان می‌افتد ولی رشته را می‌توان پامپ کرد → برای اثبات نامنظم بودن فایده‌ای ندارد

در انتخاب رشته دقت کن

مکعوس رشته ← خود رشته

ww^R مثال: 011 | 110

example 4: $L = \{ww^R \mid w \in \{0,1\}^*\}$ is nonregular

Using the pumping lemma.

① let p be the pumping length given by the pumping lemma.

② Let s be the string 0^p110^p → طول $2p+2$

چون s عضو L است و طول آن بزرگتر از p است

لم تضمین می‌کند که s می‌تواند به سه قسمت تبدیل شود

$s = xyz$ → piece y must consist only of 0's → $xyyz \notin L$ → تناقض

که سه شرط را ارضا می‌کند

$y = 0^k$, $k \geq 1$

طبق این شرط که $|y| > 0$

→ $xy^2z = 0^{p+k}110^p \notin L$

در انتخاب s دقت کن

نکته: even though 0^p0^p is a member of L, it fails to demonstrate a contradiction because it can be pumped.

رشته‌های از 1 به طول مربع کامل

Example 5) Let $D = \{1^{n^2} \mid n \geq 0\}$.

D contains all strings of 1's whose length is a perfect square.

طول آنها

مربع کامل

از لم pumping استفاده می‌کنیم تا ثابت کنیم

فرض می‌کنیم طبق فرض خلف D منظم است ← ① p طول pumping باشد که طبق لم تزریق دارای شرایط

② $s = 1^{p^2}$ → چرا چون ← a) یک عضو از D است

b) ← طول آن حداقل p است

③ پس طبق لم تزریق s می‌تواند به سه بخش تقسیم شود ←

$s = xyz$ → for any $i \geq 0$ the string xy^iz is in D

④ $|s| = |xyz| = p^2$

$|xy^2z| = |xyz| + |y| \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2$

طول، طول، طول

$= p^2$, $\leq p$

① طبق شرط لم

$|xy| \leq p \rightarrow |y| \leq p$

② $|xy^2z| > p^2$

از 1 و 2 نتیجه می‌گیریم $\longrightarrow p^2 < |xy^2z| < (p+1)^2$

طول xy^2z به صورت اکید بین دو مربع کامل متوالی قرار می‌گیرد

بین دو مربع کامل متوالی

خودش نمی‌تواند مربع کامل باشد

مثلاً $64 < n < 81$

n هرگز نمی‌تواند مربع کامل باشد

پس → $xy^2z \notin D$ → زبان D نامنظم است

hirmandpaper

رشته‌هایی که طول آنها توانی از ۲ است

Example 6 ⊙ $A=\{a^{2^n} \mid n\geq 0\}$

زبان unary

① فرض خلف، فرض می‌کنیم A منظم است ← p را طول pumping آن در نظر می‌گیریم که توسط لم تزریق داده شده

② $s=a^{2^p}$ ← چون ① ← s عضو A و ② طول s بیشتر از p

③ طبق لم تزریق ← s can be split into 3 parts $s=xyz$ که سه شرط را ارضا می‌کند

④ شرط لم را بگوید که $|xy|\leq p$ و $y\neq\varepsilon$ ← $p<2^p$

$|x|\leq|xy|\leq p<2^p$

$|xy^2z|=|xyz|+|y|<2^p+2^p=2^{p+1}$

$|xy^2z|<2^{p+1}$ ، $|xy^2z|\geq 2^p$

$2^p<|xy^2z|<2^{p+1}$ ←

طول رشته‌ی xy^2z توان ۲ متوالی ← رشته‌ی xy^2z نمی‌تواند توانی از ۲ باشد ← تناقض ← ناقض اولیه ← A منظم نیست

example 7 $L=\{w\in\{a,b\}^* \mid n_a(w)<n_b(w)\}$ is not regular

((تعداد a ها اکیداً کمتر از تعداد b ها))

① p ← pumping length

② $s=a^p b^{p+1}$

③ $|xy|\leq p$ ← y فقط از a تشکیل شده

$y=a^k$ و $k\geq 1$ ← اگر $i=2$ ← $a^{p+k}b^{p+1}\notin L$

← تناقض ← L منظم نیست.

مثال ۶ با کمک لم تزریق نشان دهید که زبان زیر منظم نیستند

$L_1 = \{a^n b^m c^n \mid n \geq 0, m \geq 0\}$

$L_2 = \{a^n b^m : n \geq m\}$

Example 8 $B = \{0^m 1^n \mid m \neq n\}$ is not regular

روش اول ۶ برهان گیری از خودش بیتاری

فرض می‌کنیم که B، منظم است ← طبق خواص بیتاری ۶ مکمل B ($\overline{B}$) هم منظم است ← اشتراک‌گیری با $0^* 1^*$ هم باید منظم باشد regular

$\overline{B} \cap L(0^* 1^*) = \{0^k 1^k \mid k \geq 0\}$

می‌دانیم که این زبان منظم نیست ← B cannot be regular

روش دوم ۶ برهان گیری از لم تزریق

p = Pumping length ①

$p! = p(p-1)(p-2)\cdots 1$ $s = 0^p 1^{p+p!} \in B$ ②

$|s| \geq p$ →

$s = xyz$ can be ③ divided

$x = 0^a$

④ $y = 0^b \rightarrow |y| > 0 \rightarrow (b \geq 1)$

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

$\rightarrow a+b+c \leq p$

$s' = xy^{i+1}z$ ، $i = \frac{p!}{b}$ ⑤

$y^i = 0^{p!} \rightarrow y^{i+1} = 0^{b+p!}$

$(0^b)^{\frac{p!}{b}} = 0^{p!}$

$s' = 0^{\overbrace{a+b+c}^{\leq p} + p! - p} 1^{p+p!} = 0^{p+p!-p} 1^{p+p!}$

$s' = 0^{p+p!} 1^{p+p!} \notin B$

$i = 0$ ← تا تهی ← زبان نامنظم است

نکته مهم 8 اگر یک زبان منظم باشد ← لم تزریق قطعاً برای آن برقرار است.
اما اگر لم تزریق برای یک زبان برقرار باشد ← آن زبان الزاماً منظم نیست
پس برای اثبات منظم بودن زبان نمی‌توان از لم تزریق استفاده کرد.

ex) $F = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and if } i = 1 \text{ then } j = k\}$

(a) show that F is not regular

(b) show that F acts like a regular language in the pumping lemma.

(c) explain why parts (a) and (b) do not contradict the pumping lemma.

نکته مهم 9 شما حق تعیین m و x و y و z را ندارید چون با صور و جوانی مواجه نه بود.
مثلاً در اثبات نامنظم بودن زبان

$L = \{a^n b^m \mid \frac{m}{n} \text{ is an integer}\}$

رشته گیری از رشته‌ی $a^p b^p$ مناسب نیست
چون به ازای $y = a^p$ تناقض رخ نمی‌دهد (یعنی پامپ کردن میسر است)
شما هم حق ندارید تعیین کنید که y چه هست یا نیست.
اما رشته‌ی $s = a^{p+1} b^{p+1}$ مناسب است چون به ازای هر تجزیه‌ای
دلخواه از s که واجد دو شرط $|xy| \leq p$ و $|y| \geq 1$ باشد داریم
$1 \leq |y| \leq p \rightarrow xy^0z \notin L$
چون در رشته‌ی xy^0z تعداد a ها اکیداً کمتر از تعداد b ها است
و این تعداد برابر صفر هم نیست ← نسبت تعداد a ها به b ها
قطعاً 6 صحیح نیست)