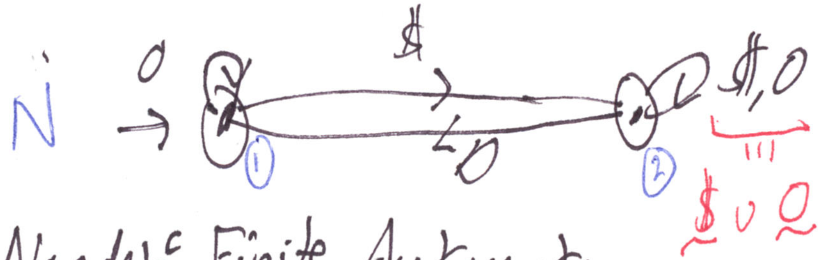
CSE 396 Lecture Thu Feb 22 Spring 2018 ①

"Turing Kit Demo" (used first 20 minutes)



Drawn as NFA without dead state $\Sigma = \{0, \$, D\}$.

Defn: A Generalized Nondet^c Finite Automaton (GNFA) is a 5-tuple $N = (Q, \Sigma, \delta, s, F)$ where $s \in Q$, $F \subseteq Q$ as with NFAs, but

$$\delta \subseteq (Q \times \underline{\underline{Regexp(\Sigma)}}) \times Q$$

$Regexp(\Sigma) \equiv$ the set of regular expressions with Σ as its char base

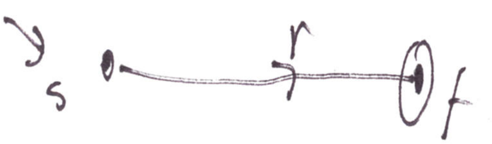
Defn: A computation trace of a GNFA from state p to state q on an input $x \in \Sigma^*$ of length m is a sequence $\vec{c} = (q_0, \beta_1, q_1, \beta_2, \ldots, q_{n-1}, \beta_n, q_n)$ st. $q_0 = p$, $q_n = q$, and x can be broken into substrings $x =: u_1 \cdot u_2 \cdots u_m$ s.t. for each j, $1 \le j \le m$:

$$\boxed{(q_{j-1}, \beta_j, q_j) \in \delta \text{ and } u_j \in L(\beta_j), \text{ ie matches } \beta_j.}$$

$L_{pq} = \{x : N$ can process x from p to q / has a valid trace with $q_0 = p, q_m = q\}$. $L(N) =_{defn} \bigcup_{f \in F} L_{sf}$.

Theorem: For every GNFA N (and hence every NFA N and DFA M) we can build a regexp r st. $L(N) = L(r)$.

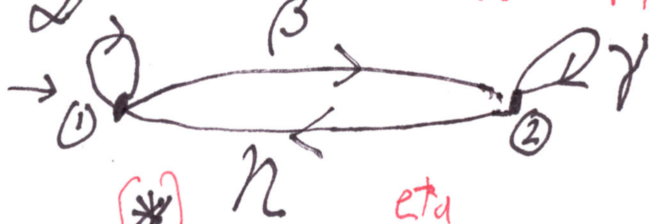
And vice-versa: Given any regexp r, there is the trivial GNFA

$N_r =$ 

Text proof arranges that you get this kind of GNFA at the end, but it gets nasty so we will use a shortcut.

above on p1, $\gamma = \$ \cup 0$ (2)

Proof: Use a general 2-state GNFA as a basis:



gamma

eta

Then $L_{11} = \Big(\underbrace{\alpha \cup \beta\cdot\gamma^*\cdot\eta}_{\text{one time around the track (or in the pitstop)}}\Big)^{*}$

zero or more times around the track

Above example: $\alpha = 0$, $\beta = \$$, $\gamma = \$+0$, $\eta = D$ — no outer *

$L_{11} = (0 + \$(\$+0)^* D)^*$

$$L_{12} = L_{11}\cdot(\beta\cdot\gamma^*) = (\alpha \cup \beta\gamma^*\eta)^*\cdot\beta\gamma^*$$

home stretch and victory spins

$L_{12} = L_{11}\cdot\$(0+\$)^*$.

$L(N) = L_{11} \cup L_{12} = L_{11} + L_{11}\cdot\$(0+\$)^* = L_{11}\cdot(\varepsilon + \$(0+\$)^*)$

$= (0+\$(\$+0)^* D)^*\cdot(\varepsilon + \$(0+\$)^*)$ = answer in the Turing Kit Exam

Alternative: $L_{11} = \underline{\alpha^*\beta\cdot L_{22}} = \alpha^*\beta\cdot(\underbrace{\gamma + \eta\alpha^*\beta}_{\text{"L22 once"}})^*$

This works for any 2 state GNFA as the basis.

Induction: N has $n \geq 3$ states.

Let's assume (2) is the only accepting state different from s (if any).

Strategy: Eliminate all non-accepting states different from s until we get $n = 2$

(NEVER need a new start state)

* Unless there are ≥ 2 accepting states different from s we're good. If so, then add a new accepting state f with arcs from all the old ones. Number it (2

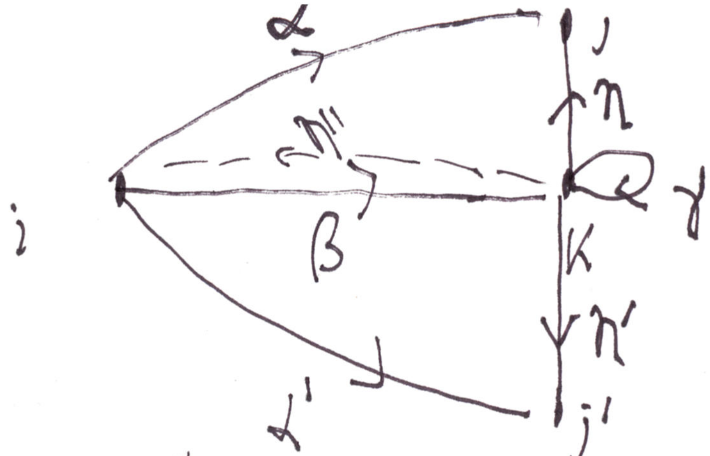
Alg^m: for $K = n$ down to 3:

eliminate state K.

done $\Rightarrow$ read answer using basis for 2-state machine.

Because $K \notin F$, any processing that goes into K from some state i must go out via some state j. This says: For all i and j, bypass (i, K) to j via

Diagram for Bypass



③

Can also have j

α from i to j

New α = old $\alpha + \beta\gamma^*\eta$. ← body of

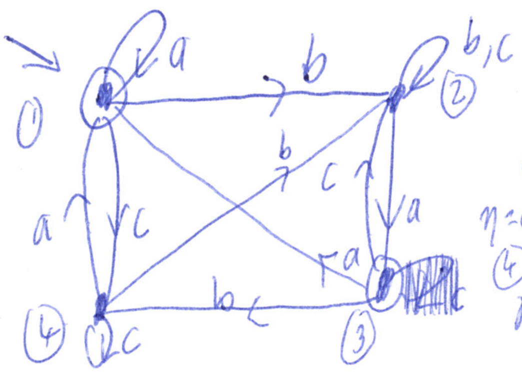
$\alpha' \mathrel{+}= \beta\gamma^*\eta'$

for k = n to 3
for i = 1 to k-1
for j = 1 to k-1

Doing for all exits j from i bypasses edge β

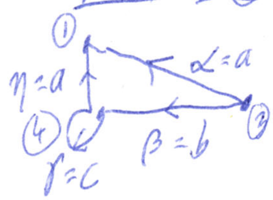
Bypassing all incoming edges allows you to eliminate state k. ☒

Added: Here is an example done "graphically" — Tuesday's lecture will do it "in code style". Consider the following DFA. It has only one accepting state besides the start state, so we need not add any more states.

M = 

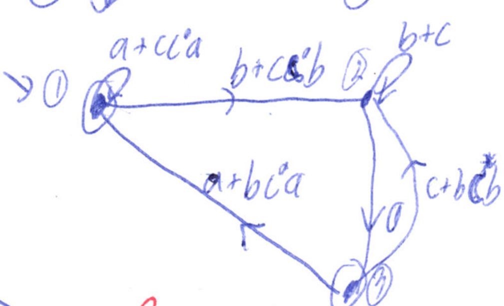
Eliminate ④: Incoming: (3, b), (1, c) Out: (a, 1), (b, 2)

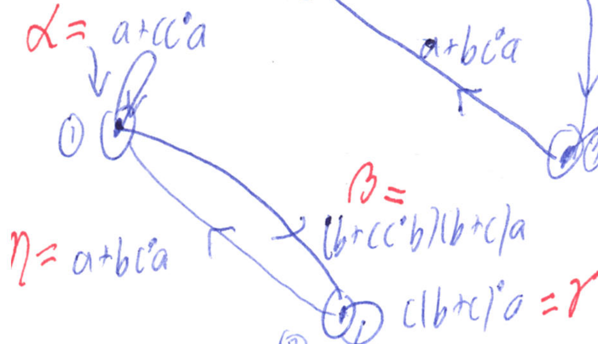
∴ Update: (3, 1), (1, 1), (3, 2), (1, 2)



new α = old $\alpha + \beta\gamma^*\eta$
$= a + bc^*a$

New loop at ① = old loop $+ cc^*a = a + cc^*a$
New (3, 2) = old (3, 2) $+ bc^*b = c + bc^*b$
New (1, 2) = old (1, 2) $+ cc^*b = b + cc^*b$. Then elim ④





Eliminate ②: In from ① and ③ Out only to ③.
So we only need to update (1, 3) and (3, 3). There was no (1, 3) but now there's $(b+cc^*b)(b+c)^*a$. And (3, 3) becomes $c(b+c)^*a$. Now down to $L(M) = L_{11} \cup L_{13}$ wi[th]

$L_{11} = (a+cc^*a + (b+cc^*b)(b+c)a \cdot (c(b+c)^*a)^*(a+bc^*a))^*$

and $L_{13} = L_{11} \cdot \beta\gamma^*$ Saves writing more of that grunge!