CSE322 Theory of Computation(L10)

Recap of last lecture

Quiz on next class on Regex

$\leftarrow$ $Half(L) = \{x : \exists y, |x| = |y|, xy \in L\}$ $\exists y \in L$

Today

Regular Expressions II

7(b)

Kleene's Theorem

Regular language: Languages of Regexes

Thm: Regular languages are equivalent to DFA

Regex is not as powerful as Java

Regex is okay for verifying patterns ...

like C variable names, URLs ... (lexical an.)

Can Regex verify syntax of C programs?

Thm: If L is accepted by an NFA, then L is decribed by a regex.

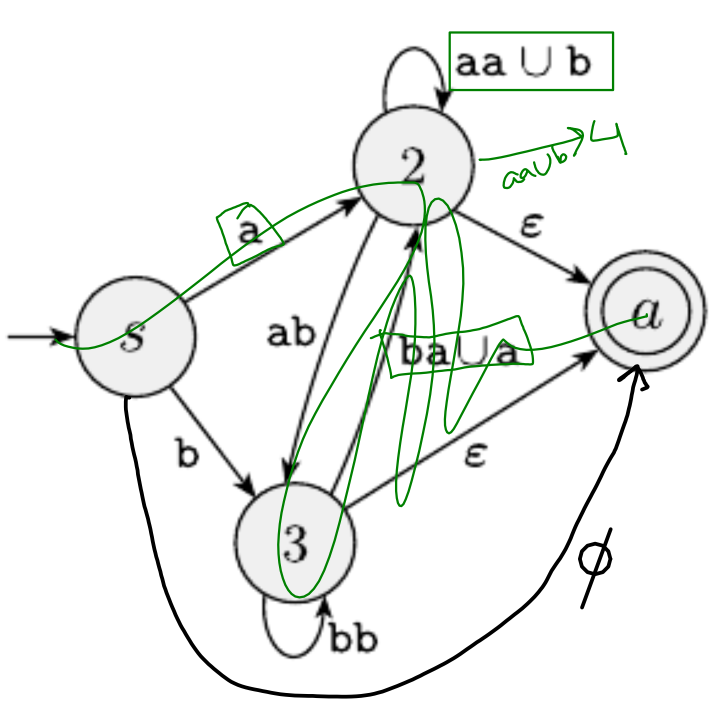
(Proof uses GNFA -- coming up)

Corollary:

Regular expression = DFA = NFA

Generalized NFA (GNFA)

* arcs between states are labeled with regex
* input read by multiple alphabets at a time



$2 \to 3$

Σ'^*

$\delta(2, ab) = 3$

Is abababab ε accepted?

$s \to 2\ 2\ 3\ 2\ 2\ 3$ a

	s	2	3	a
→ s	x	a	b	∅
2	x	aa∪b	ab	ε
3	x	ba∪a	bb	ε
*a	x	x	x	x

~~Convention (easy to satisfy)~~

1. Unique start state with no incoming <-

2. Unique accept state with no outgoing ->

q, q' $\quad q \to q'$ except when $q = q_f$, or $q' = q_a$

3. -> between every other pair, incl. self-loops

GNFA Formalization

GNFA is described by $(Q, \Sigma, \delta, q0, qf)$

* set of states Q
* set of input alphabets Σ
* start state $q0 \in Q$
* final state $qf \in Q$
* transition function δ: $(Q - \{q_f\} \times Q - \{q_0\}) \longrightarrow R$: set of all possible regular expression

$q \xrightarrow{a} ?$ $\qquad$ $q \xrightarrow{?} q'$

(there is a Regex between every source and dest. states)

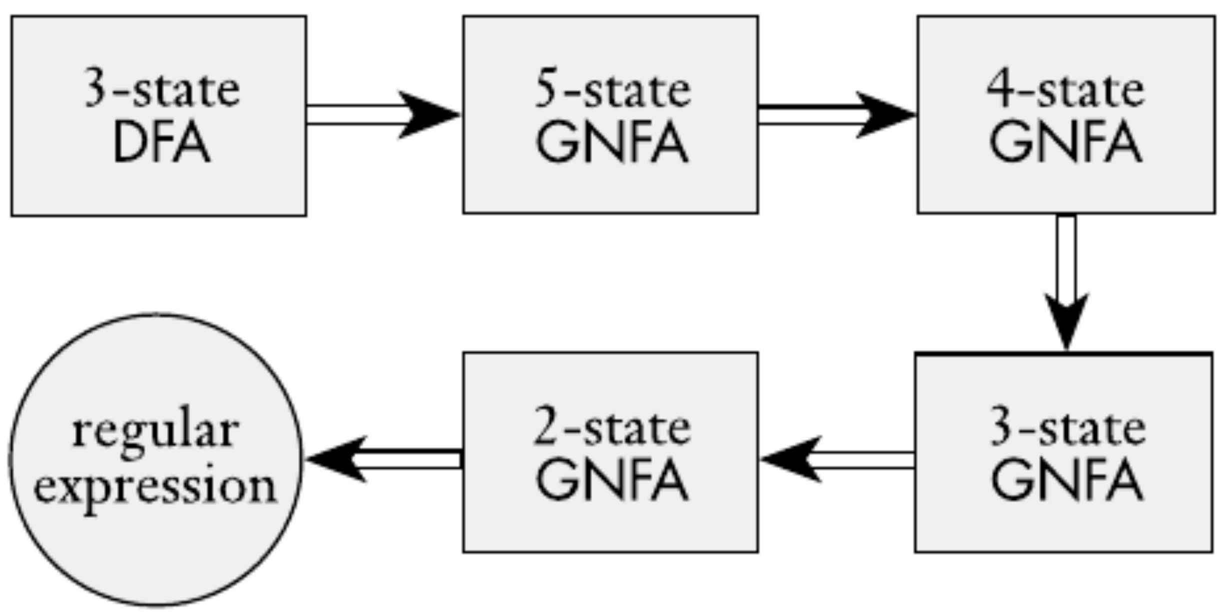
GNFA accepts $s \in \Sigma^*$ if $s = s1 \ldots sk$ and there exists a seq. of states $r0 \ldots rk$

$s = \; r_0 \xrightarrow{s_1} r_1 \xrightarrow{s_2} r_2 \xrightarrow{s_3} \cdots \; r_{k-1} \xrightarrow{s_k} r_k$

1. $si \in \Sigma^*$
2. $r0 = q0$
3. $rk = qf$
4. for all $i = 1 \ldots k$ $\; si \in L(\delta(r_{i-1}, r_i))$ $\qquad$ s_i matches $\delta(r_0, r_1)$

Proof of theorem:

1. DFA/NFA -> GNFA
2. GNFA -> Reduced GNFA
3. Reduced GNFA -> Regex



DFA/NFA -> GNFA

if D accepts w, then N accepts w.

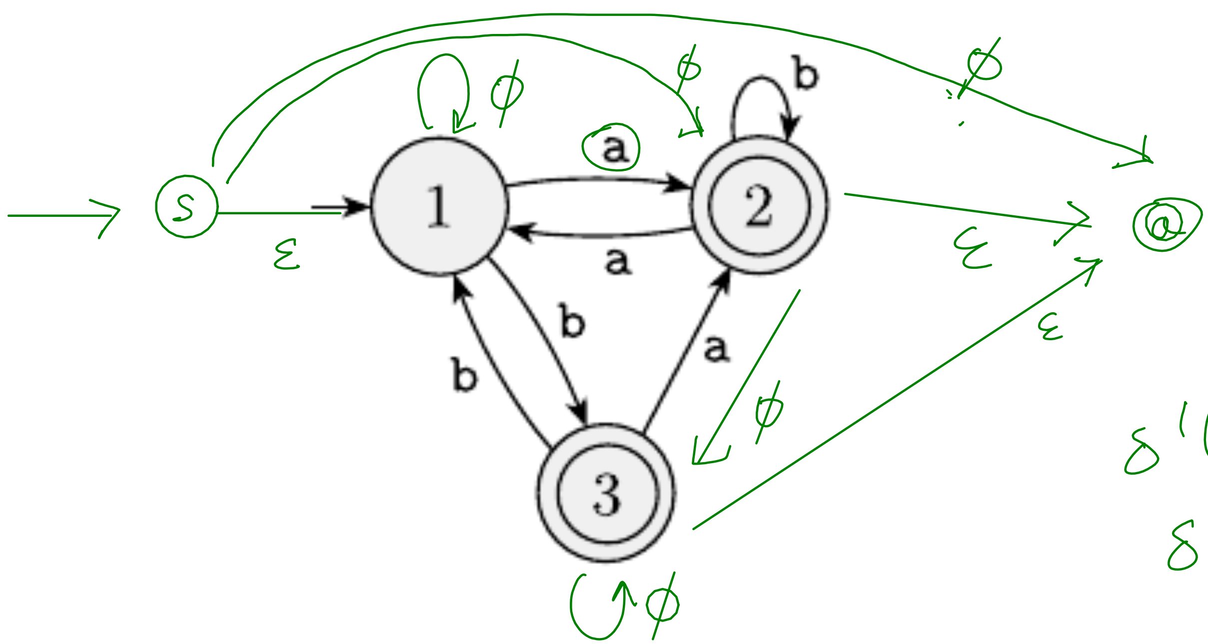
$q_0 \cdots q_f \in F$

$q_0 \; q_1 \cdots$

$s \xrightarrow{\varepsilon} q_0 \rightarrow \cdots \rightarrow q_f \xrightarrow{\varepsilon} a$

Given FA N, L = L(N)...

1. Add unique start and end states:



DFA $D = \langle Q, \Sigma, \delta, q_0, F \rangle$

GNFA $N = \langle Q \cup \{s\} \cup \{a\}, \Sigma, \delta', s, a \rangle$

$$\delta'(p,q) = \begin{cases} a & \text{if unique s.t. } \delta(p,a) = q \text{ if } p,q \in Q \\ \emptyset & \text{o/w} \\ \boxed{\bigcup_{a \in A} a} & \text{if } A = \{a : \delta(p,a) = q\} \end{cases}$$

$D: p \xrightarrow{a} q$

$\delta'(1,2) = a$

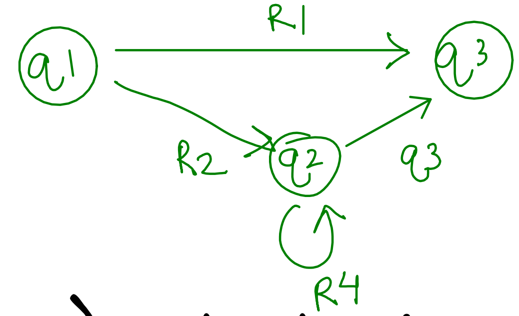
$\delta'(s,2) = \emptyset$

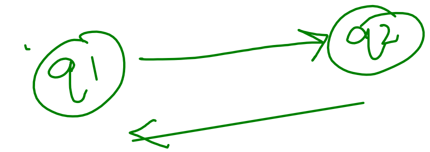
"$a \cup b \cup c$"

To prove: $L(N) = L(D)$

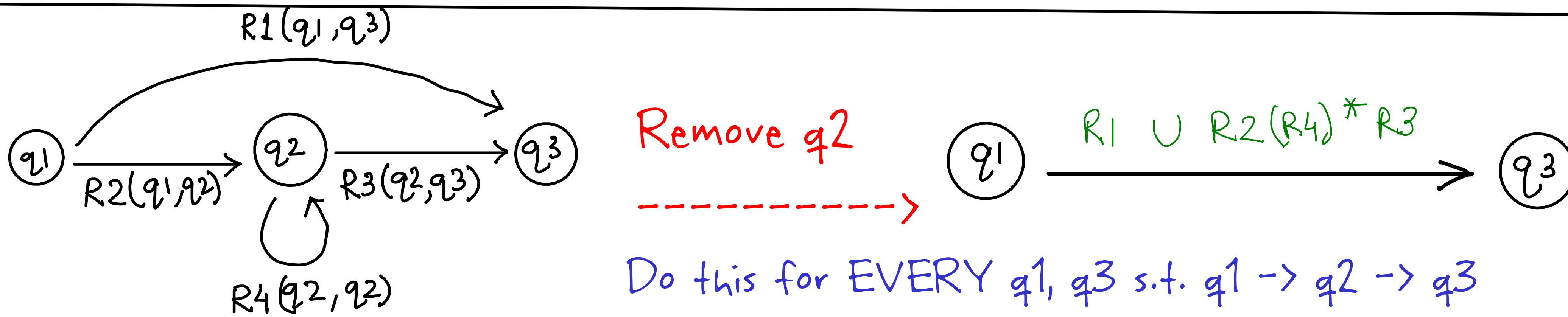
Extend to other states

Reduce GNFA





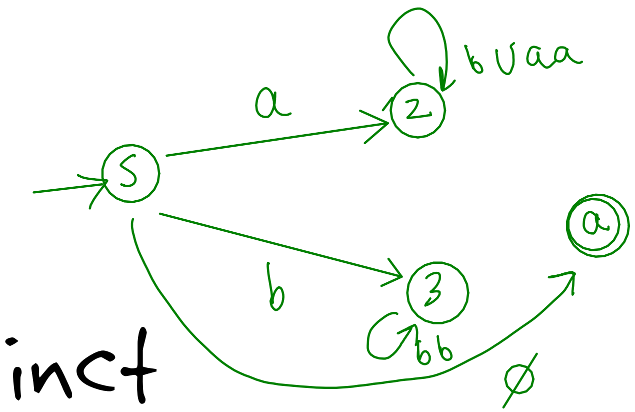
2. Remove any (non-terminal) state by modifying regexes on affected arcs, in an iterative manner



Do this for EVERY q1, q3 s.t. q1 -> q2 -> q3

Do this for EVERY q1 s.t. q1 -> q2 -> q1

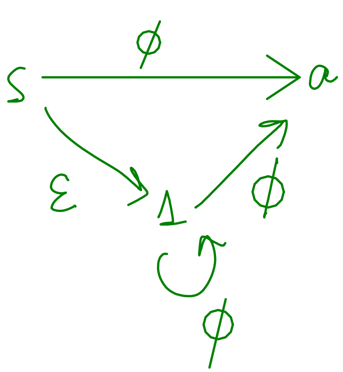
```
while (#states > 2):
        q <- some intermediate state
        // remove q
        for every ordered pair (qi, qj):
                // qi qj need not be distinct
                // qi cannot be final state
                // qj cannot be start state
                create regex for qi -> qj by bypassing q
```



$R\phi = \phi$

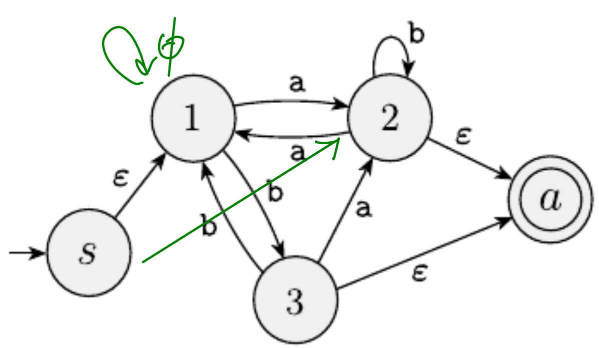
$\phi \cup \phi = \phi$

$\phi^* = \{\varepsilon\}$



$\phi \cup \varepsilon . \phi^* . \phi = \phi$

$s \longrightarrow a$



Remove 1 --------->

s,2
s,3
s,a
2,2
3,3
3,2
3,a
2,a
2,3

$s \longrightarrow 2$

$\phi \cup \varepsilon . \phi^* . a = a$

$\phi \cup \varepsilon . \phi^* b = b$

$\phi \cup \varepsilon . \phi^* . \phi = \phi$

$2 \longrightarrow 1 \longrightarrow 2$ $\quad b \cup a\phi^* a$

$= b \cup aa$

G = <Q,S,d,q0,qf> -> G' = <Q' $= Q - \{q_m\}$, S, d', q0' $= q_0$, qf' $= q_f$> in which qm $\neq q_0, q_f$ is removed

$p \xrightarrow{?} q$

$$\delta'(p,q) = \delta(p,q) \cup \delta(p,q_m) \cdot \delta(q_m,q_m)^* \cdot \delta(q_m,q)$$

Prove that L(G) = L(G')

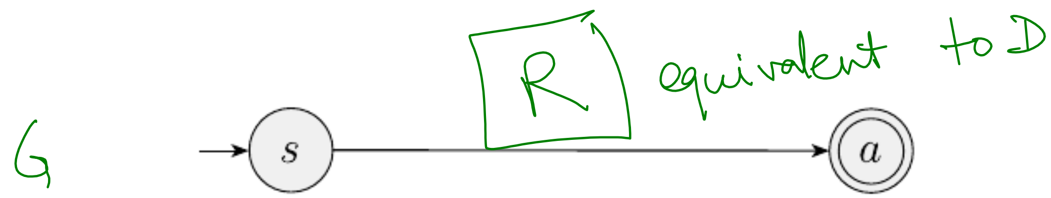
=> If G accepts w then G' accepts w.

Read from Sipser

<= If G' accepts w then G accepts w.

GNFA -> Regex

3. Stop when 2 terminal states are left.
   Return regex on arc between them.



$L(D) = L(G) = L(R)$

Lemma: Suppose R is the final regex left.
Then, L(N) = L(R).

Identifying C comments

Design NFA to identify valid multiline comments?

```
/* I am a simple but
 * three-line/3-line
 * *long* **multi-line**
 * comment
 */
```

What is a lexer?

See http://www.cs.man.ac.uk/~pjj/cs211/flexdoc.html

Homomorphism

$X = \{a, b\}$ $Y = \{1, 0, 2\}$

$f(a) = 102$

$f(b) = 222$

concatenation respecting function

$f(aabb) = 102\,102\,222222$

Consider alphabets X and Y.

Let f() be a homomorphism from strings over X to strings over Y.

$f(xy) = f(x)f(y)$ for any strings x, y over X.

Show that,

1. $f(e) = e$

If $f(e) = y$ over Y, then $y = f(e) = f(ee) = f(e)f(e) = yy$.

$$\overset{Y}{\frown}$$

2. If L over X is regular, then f(L) is regular.

$R = (0 \cup 1)^* 001 (0 \cup 1)^*$, $f(0) = ab$, $f(1) = bc$

$\therefore \quad f(R) = (ab \cup bc)^* ababbc (ab \cup bc)^* = R'$ is a regex

How to prove this fact? $L(R') = f(L)$ Tut.

4. If L over Y is regular, $\quad f^{-1}(L) = L'$

then { x over X : f(x) in L } is regular.

Tut

$x \in f^{-1}(L)$ iff $f(x) \in L$

$q_0 \overset{f(x)}{\rightsquigarrow}$

$q \xrightarrow{a}$

$X = \{a, b\}$

$Y = \{0, 1\}$

$f(a) = 01$

$f(b) = 10$