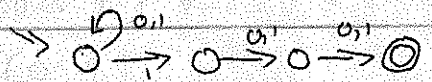
Goal: define sets

	machine	language	
Finite Sets	Write down all elements		set operations
Regular languages	DFAs $\Leftrightarrow$ NFAs	~~R.P.~~ RE	closure operations $(\cup, \cap, ^c, \circ, ^*)$



The regular P.L., it's programs (expression) come from the tree-Algebra

$$E := E \cup E \mid E \circ E \mid E^* \mid \emptyset \mid \varepsilon \mid a \in \Sigma$$

$\emptyset$ =

```
interface RP;
Empty()
```

```
class Empty impl RP
```

$E \circ E$

```
class Concat impl RP
Concat(RP L, RP R)
```

$w \in L(E)$

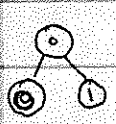
$a \in L(a)$

$\varepsilon \in L(\varepsilon)$

$$\frac{w \in L(E_1)}{w \in L(E_1 \cup E_2)} \qquad \frac{w \in L(E_2)}{w \in L(E_1 \cup E_2)}$$

$0 \circ 1 \neq \{01\}$

$$\frac{x \in L(E_L) \quad y \in L(E_R)}{xy \in L(E_L \circ E_R)} \qquad \frac{x \in L(E_1) \quad y \in L(E_1^*)}{xy \in L(E_1^*)}$$



$\varepsilon \in L(E_1^*)$

```
new Concat(new Char('0'),
           new Char('1'))
```

Σ^* '.' 'o' 'c'

"du -h *.c"

"emacs refl.txt"

Prove that every DFA matches a ~~R.P.~~ RE

① $\forall r \in RE.\ \exists d \in DFA.\ L(r) = L(d)$ — compiler

② $\forall d \in DFA.\ \exists r \in RE.\ L(d) = L(r)$ — decompiler

$\forall$ stuff $\exists$ other $\bullet$ (such that) Property $:=$ function from stuff to other where the property is asserted

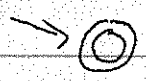
6-2 / $\forall r \in RE,\ \exists d \in DFA,\ L(r) = L(d)$

in: r out: $d = (Q, \Sigma, q_0, \delta, F)$

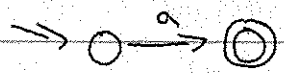
induction on cases of RE r:

case $\emptyset$:

case ϵ:

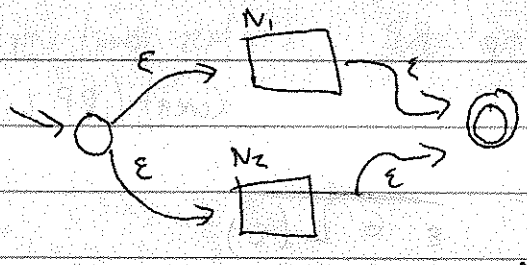


case $a \in \Sigma$:



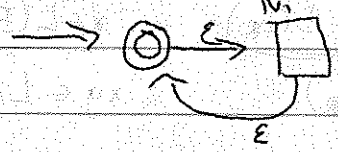
case $E_1 \cup E_2$:

assume $L(N_1) = L(E_1)$ $L(N_2) = L(E_2)$



case $E_1 \circ E_2$:

case E_1^*:



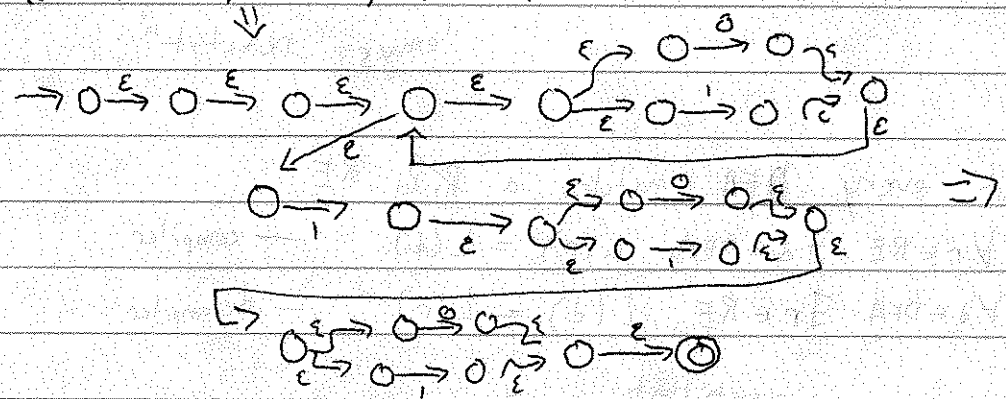
Natural = O
| succ Natural

Prove P by induction on Nat:

$\forall (P : Nat \to Prop).$
$(P\ O)$
$\wedge (\forall\ n : Nat.$
$P\ n \to P\ (S\ n))$
$\to$
$\forall n \in Nat.\ P\ n$

ind P P2 IC n
if n == 0 then P2
else IC (n-1)
(ind P P2 IC (n-1))

C $(((0 \cup 1)^* \circ 1) \circ (0 \cup 1)) \circ (0 \cup 1))$ ← *1??



6-3

$\forall d \in DFA.\ \exists r \in RE.\ L(r) = L(d)$

DECOMPILE = IN ∘ RIP$^\wedge$ ∘ OUT

IN: n-state DFA → (n+2)-state GNFA

RIP: (n+1)-state GNFA → n-state GNFA

OUT: 2-state GNFA → RE

GNFA := generalized NFA (NFA w/ RE on the edges)

$(Q, \Sigma, q_0, \Delta, q_f)$ + two rules

$q_0 \in Q$ — start state — You cannot go to the start state

$q_f \in Q$ — end state — You cannot leave the end state

$\Delta : (Q - q_f) \times (Q - q_0) \to RE$

$w \in L(q_j, q_k)$ iff $w \in L(\Delta(q_j, q_k))$

or $w = xy \wedge x \in L(\Delta(q_0, q_i)) \wedge y \in L(q, q_i)$

$w \in L(q)$ iff $w \in L(q, q_0)$

OUT: $(\{q_0, q_f\}, \Sigma, q_0, \Delta, q_f)$

$\Delta = \{((q_0, q_f), r)\}$

return r

IN: $(Q, \Sigma, q_0, \delta, F)$

$\Rightarrow (Q', \Sigma, q_0', \Delta, q_f')$

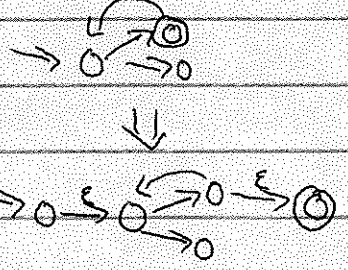
$Q' = Q \cup \{q_0', q_f'\}$

$\Delta(q_0', q_0) = \varepsilon$

$\forall q_f \in F,\ \Delta(q_f, q_f') = \varepsilon$

$\Delta(q_i, q_j) = a$ iff $\delta(q_i, a) = q_j$

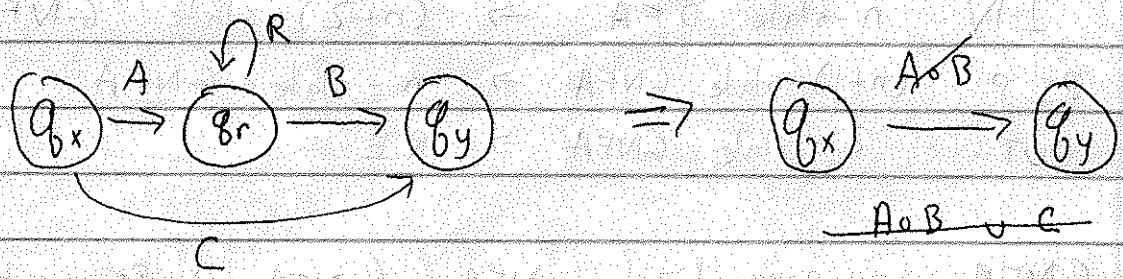
$\Delta = \emptyset$ in all other cases



6-4/ RIP : (n+1)-GNFA ⇒ n-GNFA

in: $(Q \cup \{q_r\}, \Sigma, q_0, \Delta, q_f)$

out: $(Q, \Sigma, q_0, \Delta', q_f)$



~~$A \circ B \cup C$~~

$A \circ R^* \circ B \cup C$

written, but not explained