CSE 396 Lecture Thu 2/11 Spring 2016

Given a regular expression R, <u>over</u> an alphabet Σ, define

$$\underline{L(R)} = \{x \in \Sigma^* : x \text{ \underline{matches} } R\}.$$

Intent / Intension: <u>how</u> does a string x match R?

<u>Extension</u>: concentration on the language as a whole.

Formal Inductive Definition of Regular Expressions, Their Languages, With NFA "Pictures" Too:

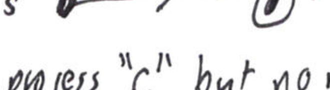
<u>Basis</u>: (Lecture: Temporarily use ~ to say something is a symbol for a language or string.)

$\underset{\sim}{\emptyset}$ is a regexp, $L(\underset{\sim}{\emptyset}) = \emptyset$ $N_\emptyset$:


$\underset{\sim}{\varepsilon}$ is a regexp, $L(\underset{\sim}{\varepsilon}) = \{\varepsilon\}$ $N_{\underset{\sim}{\varepsilon}}$:


Cannot process any nonempty string.

For any char $c \in \Sigma$,

$\underset{\sim}{c}$ is a regexp, $L(\underset{\sim}{c}) = \{\text{"c"}\}$ N_c: $L(N_c) = \{c\}$


Can process "c" but no more
<u>Must</u> process one c, so $\varepsilon \notin L(c)$

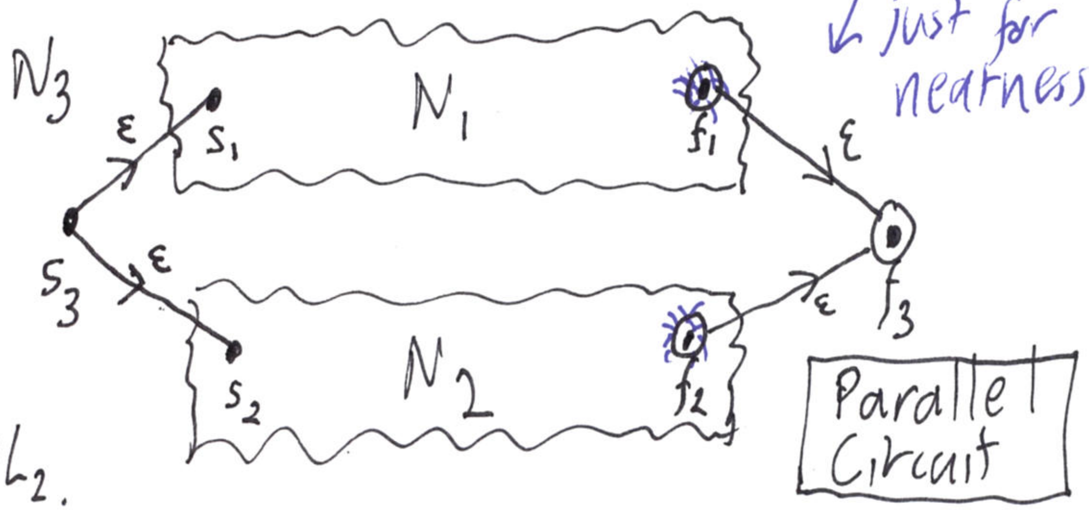
<u>Induction</u>: Let any two regexps R_1 and R_2 be given, along with:

- Their languages $L_1 = L(R_1)$ and $L_2 = L(R_2)$
- Their NFA "pictures" N_1 and N_2 such that $L(N_1) = L_1$ and $L(N_2) = L_2$.

Then: $R_3 = (R_1 \cup R_2)$ is a regexp [also write $R_3 = R_1 + R_2$] (2)
and denotes $L(R_1) \cup L(R_2) =: L(R_3)$.
From the given NFAs N_1 s.t. $L(N_1) = L(R_1)$
and N_2 s.t. $L(N_2) = L(R_2)$ build $N_3 =$

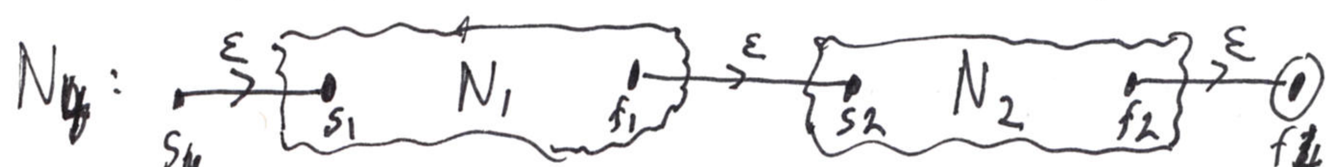
The text skips this part, which is ↙ just for neatness.

Build N_3 and complete the induction by showing $L(N_3) = L_3$ $= L(R_3) = \underline{L_1 \cup L_2}$.



∴ N_3 can process a string x from s_3 to f_3 ⟺
N_1 can process x from s_1 to f_1 **OR** N_2 can process x from s_2 to f_2
∴ $\underbrace{L(N_3) = L(N_1) \cup L(N_2)}_{\text{by machine construction}} = \underbrace{L(R_1) \cup L(R_2)}_{\text{by induction hyp.}} = L_1 \cup L_2 \equiv L_3$ ☒ ∴ $L(N_3) = L(R_3)$.

Then $R_4 = (R_1 \bullet R_2)$ is a regexp, [Parentheses and dot optional like how we do $\bullet, +$ in math]
$L(R_4) =_{def} L(R_1) \bullet L(R_2)$. Given the NFAs N_1 and N_2, build



Then N_4 can process a string x from s_4 to f_4 ⟺ x can be broken as $x =: yz$ such that N_1 can process y from s_1 to f_1 and N_2 can process z from s_2 to f_2.
∴ $L(N_4) = L(N_1) \bullet L(N_2)$ (By Series Machine Construction:)

Recall: $A \cdot B = \{ X : X$ can be broken as $X =: Y \cdot Z$ st. $Y \in A \wedge Z \in B \}$.

So $L(R_4) \equiv_{def} L(R_1) \cdot L(R_2) = \{ X :$ X can be broken as $X =: Y \cdot Z$ st. $Y \in L(R_1) \wedge Z \in L(R_2) \}$

By induc. hyp: $L(R_1) = L(N_1)$
$L(R_2) = L(N_2)$.

$\therefore L(R_4) = \{ X :$ X can be broken as $X =: YZ$ st. ~~$Y \in L(N_1)$~~ $Y \in L(N_1) \wedge Z \in L(N_2) \}$.

$\therefore = L(N_1) \cdot L(N_2) = \underline{L(N_4)}$. $\boxed{\therefore L(N_4) = L(R_1) \cdot L(R_2) = L(R_4) \;. \boxtimes}$

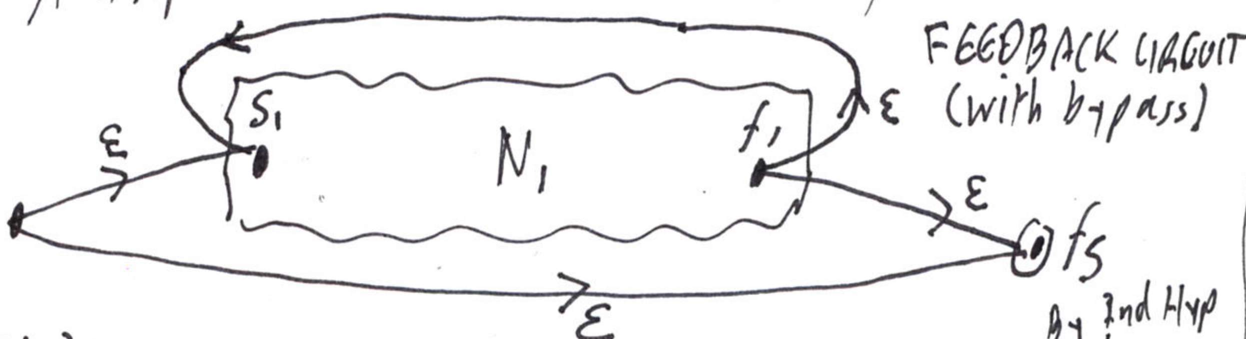
☆

Finally define $\underline{R_5 = (R_1)^*}$ [R_2 not involved] And define

$L(R_5) = L(R_1)^* = \{\varepsilon\} \cup L(R_1) \cup L(R_1)^2 \cup \cdots$ to ∞.

$= \{ X \in \Sigma^* :$ X can be broken as $X =: \underbrace{X_1 \cdot X_2 \cdots X_K}_{\text{substrings}}$ such that $X_1 \in L(R_1) \wedge \cdots \wedge X_K \in L(R_1)$ (ε rides by default with $K = 0$)

Build N_5 given N_1 like so:

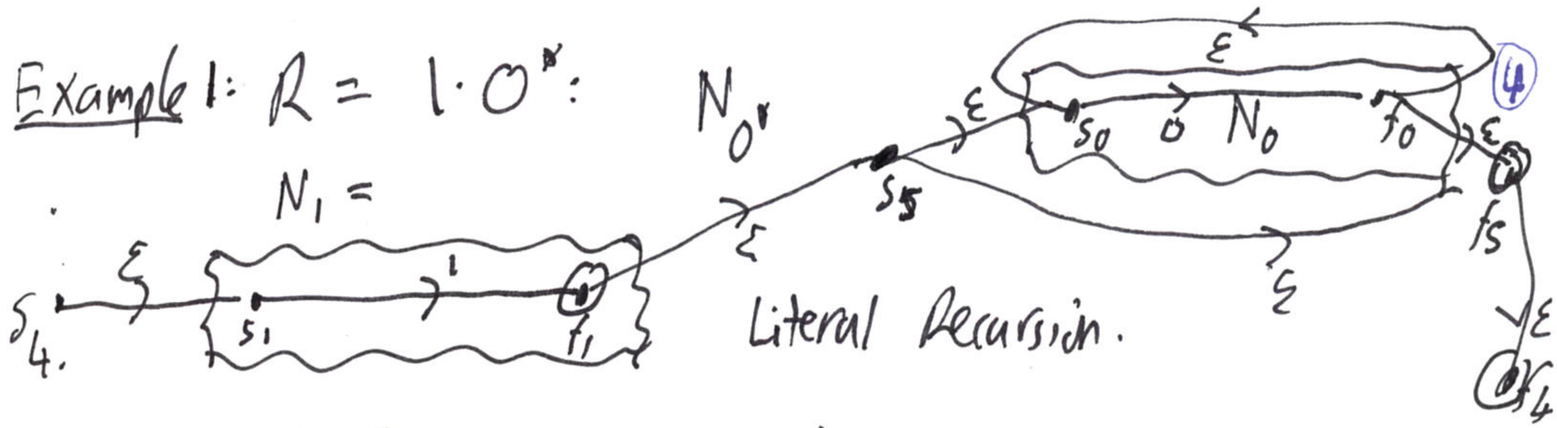


Then $L(N_5) = \{ x :$ X can be broken into zero or more substrings each processed by $N_1 \} \overset{\text{By Ind Hyp}}{=} L(R_1)^* = L(R_5)$

by Diagram construction

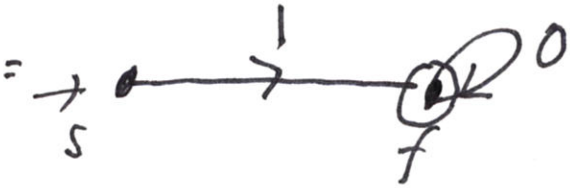
$\therefore$ <u>Theorem</u>: For every regular expression R, we can (recursively!) build an NFA N_R such that $L(N_R) = L(R)$.

Example 1: $R = 1 \cdot 0^*$:



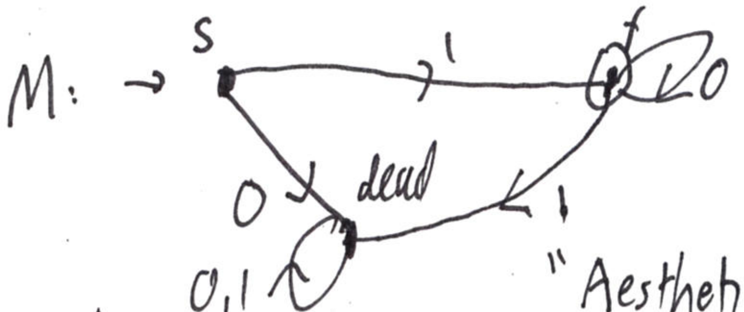
Much simpler (closer to text)

$N =$



Formally an NFA but has no actual nondeterminism.

M:

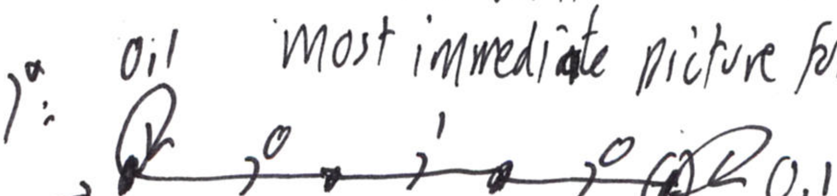


Hence can complete to a DFA.

"Aesthetic Point": The NFA N is the most immediate picture for 10^*, not M.

Example 2:

$(0+1)^* 010 (0+1)^*$:



Has actual nondeterminism.

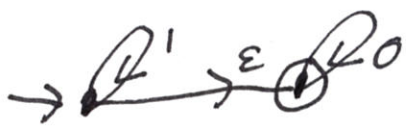
So DFA more complicated...

Extra — added after lecture:

Example 3: $R = 1^* 0^*$: Wrong is

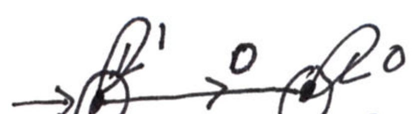
— that is $(1+0)^*$ i.e. $(1 \cup 0)^*$

Correct is



Thus ε-arcs are sometimes helpful.

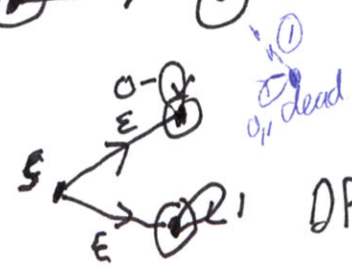
Also good is



This needs the start state to be accepting too. Can be completed to a DFA as shown in blue.

Example 4:

$R = 1^* \cup 0^*$:



DFA is

Again, should complete DFA as shown in blue.