CSE396 — Lecture Thu Feb 15 — Spring 2018

Top-Hat #0697

Formal Inductive Definition of Regexps r over an alphabet Σ together with their languages $L(r)$ and equivalent NFAs N_r.

Basis:

- $\tilde{\varnothing}$ is a regexp. $L(\tilde{\varnothing}) = \varnothing$ $N_{\tilde{\varnothing}} =$ 
- $\tilde{\varepsilon}$ is a regexp. $L(\tilde{\varepsilon}) = \{\varepsilon\}$ $N_{\tilde{\varepsilon}} =$ 

$L(N_{\tilde{\varnothing}}) = \varnothing$ but $L(N_{\tilde{\varepsilon}}) = \{\varepsilon\}$.

For any char $c \in \Sigma$,

- $\tilde{c}$ is a regexp. $L(\tilde{c}) = \{c\}$ $N_{\tilde{c}} =$  $L(N_{\tilde{c}}) = \{c\}$.

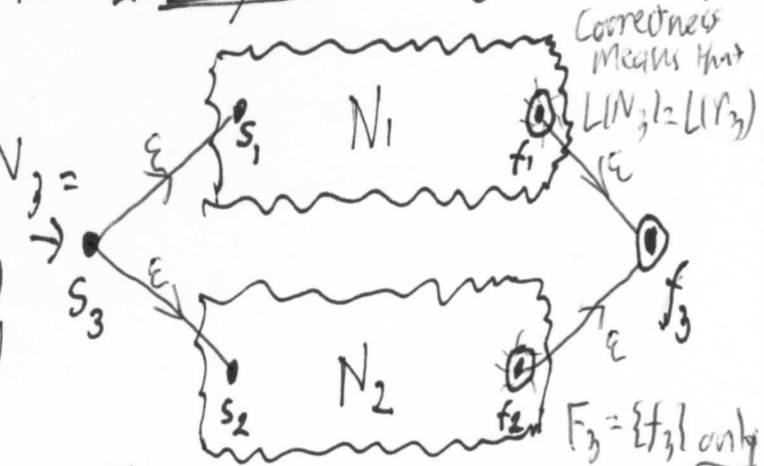
Note: The $\sim$ is temporary to distinguish regexps from chars, ε, or $\varnothing$.
Using a single acc. state f ⊙ different from s looks like a {ground terminus} in electrical diagrams
We will maintain the inductive invariant that no arcs go out of f and $F = \{f\}$.

Induction: Let any two regexps r_1 and r_2 be given. Then (we can define):

- $r_3 = r_1 \cup r_2$ $L(r_3) = L(r_1) \cup L(r_2)$

Also $= r_1 + r_2$, $r_1 | r_2$

To define the NFA N_3
We may suppose that NFAs N_1 s.t. $L(N_1) = L(r_1)$ and N_2 s.t. $L(N_2) = L(r_2)$ are already built. Build



To build $N_3 = (Q_3, \Sigma, \delta_3, s_3, F_3)$ in code terms, take:

$Q_3 = Q_1 \cup Q_2 \cup \{s_3, f_3\}$ s_3 new start $F_3 = \{f_3\}$ and:

understood that $s_3 \notin Q_1 \cup Q_2$, f_3

$\delta_3 = \delta_1 \cup \delta_2 \cup \{(s_3, \varepsilon, s_1), (f_1, \varepsilon, f_3), (s_3, \varepsilon, s_2), (f_2, \varepsilon, f_3)\}$.

The NFA N_3 is correct because: For any $x \in \Sigma^*$:

$x \in L(N_3) \Leftrightarrow N_3$ can process x from s_3 to f_3

$\Leftrightarrow$ (by diagram) N_1 can process x from s_1 to f_1 OR N_2 can process x from s_2 to f_2

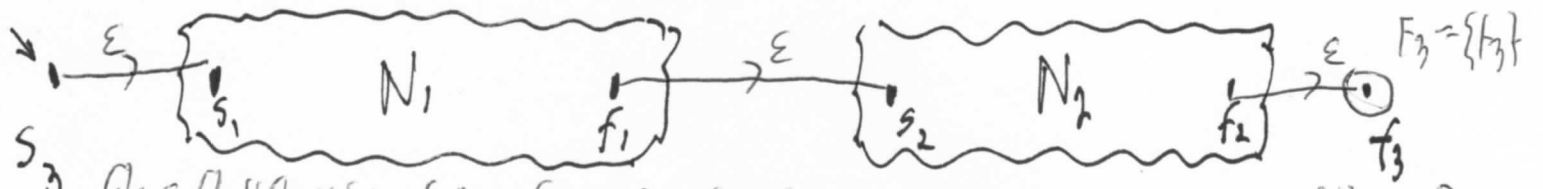
$\Leftrightarrow x \in L(N_1) \cup L(N_2)$ By induction hypothesis.

$\Leftrightarrow x \in L(r_1) \cup L(r_2) \Leftrightarrow x \in L(r_3)$ ☒

Then

• $r_3 = r_1 \circ r_2$ is a regexp, $L(r_3) = \underbrace{L(r_1)}_{z=x} \cdot \underbrace{L(r_2)}_{y} = \{z \in \Sigma^* : z \text{ can be broken as } z =: xy \text{ s.t. } x \in L(r_1) \text{ and } y \in L(r_2)\}$.

and given NFAs N_1 and N_2 (obeying invariants) st. $L(N_1) = L(r_1)$ and $L(N_2) = L(r_2)$, build:



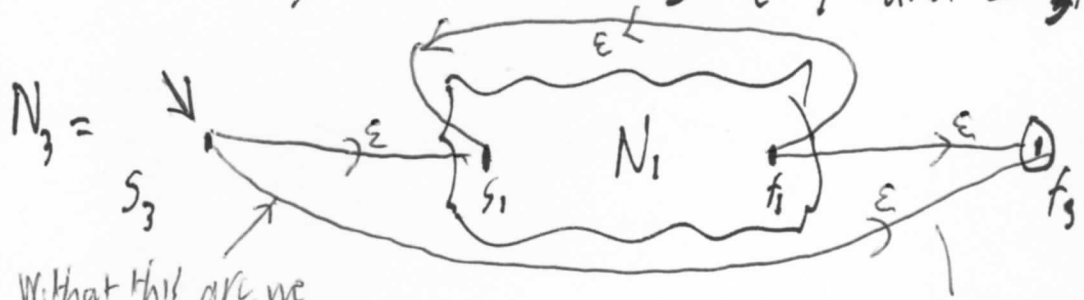
$Q_3 = Q_1 \cup Q_2 \cup \{s_3, f_3\}$, $\delta_3 = \delta_1 \cup \delta_2 \cup \{(s_3, \varepsilon, s_1), (f_1, \varepsilon, s_2), (f_2, \varepsilon, f_3)\}$. Done.

Interlude: Recall that for any language A, $A^2 = A \cdot A = \{xy : x \in A \text{ and } y \in A\}$

Then define $A^3 = A \cdot A \cdot A$, $A^4 = A \cdot A \cdot A \cdot A = A^2 \cdot A^2$ etc. NOT $\{xx : x \in A\}$!

And finally: $A^* = A^0 \cup A \cup A^2 \cup A^3 \ldots = \{\varepsilon\} \cup A \cup A^2 \ldots = \bigcup_{i=0}^{\infty} A^i$. "A-Star" using Kleene Star "clay-nuh"

• Given a regexp r_1, define $r_3 = (r_1)^*$ [for (r_1^*) dependency] and $L(r_3) = [L(r_1)]^*$. Given N_1, build



Without this arc, we would get $L(N_1)^+$ instead

FEEDBACK CIRCUIT (with bypass).

For N_3 to be correct, we need $L(N_3) = L(r_3) = L(N_1)^*$ (By Induction Hypothesis)

Then $L(N_3) = L(r_3)$, and this completes the induction for both definition and proof.

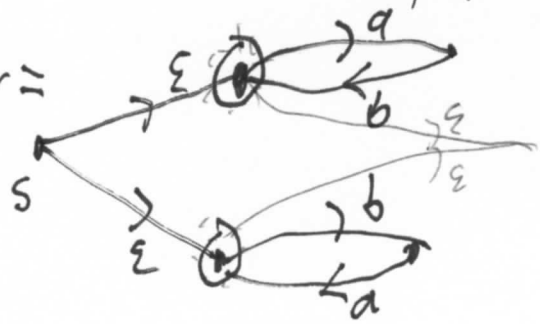
We have proved Theorem: For every regular expression r, we can build an NFA N_r s.t. $L(N_r) = L(r)$. ☒.

In practice, we can economize on many of the ε arcs in the proof.

Example

$r = (ab)^* \underbrace{\cup}_{\text{Outermost operator like if it is +}} (ba)^*$.

$N_r =$



We could make both accepting states go to a unique final one, but we don't need to.