CS 396 Lecture Thu 2/14 Spr 2019

TopHat #1018 Stephen

Kleene's Theorem: For any language L over a finite alphabet Σ, the following statements are equivalent:

(a) There is a DFA $M = (Q, \Sigma, \delta, s, F)$ such that $L = L(M)$ ← Text definition

(b) There is an NFA N s.t. $L = L(N)$

(c) There is a regular expression r s.t. $L = L(r)$. ← Historical defn of "L is a regular language".

Defn of Regexps over Σ and beginning proof with (b) ⇒ (c).

Basis

- $\emptyset$ is a regexp, $L(\emptyset) = \emptyset$ $N_\emptyset$ =

$N_\emptyset = (Q, \Sigma, \delta, s, F)$ where δ is empty. $\therefore L(N_\emptyset) = \emptyset = L$. ✓

For "show", we will give each NFA we build exactly one acc. state f different from s.

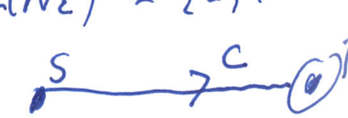
INDUCTION INVARIANT.

- ε is a regexp, $L(\varepsilon) = \{\varepsilon\}$, N_ε =

$\therefore L(N_\varepsilon) = \{\varepsilon\}$. ✓

For any char $c \in \Sigma$

- c is a regexp, $L(c) = \{c\}$, N_c =



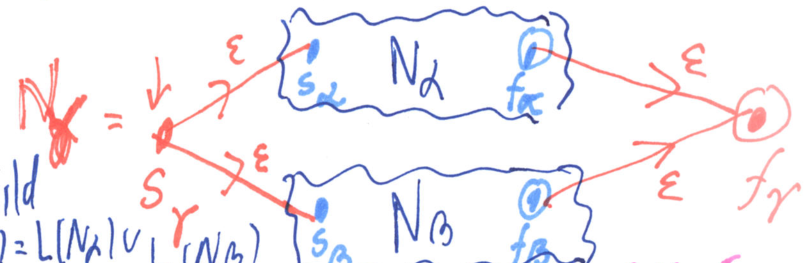
$\therefore L(N_c) = \{c\}$

Induction: Let any two regexps α and β be given. Then:

- $\gamma =_{def} \alpha + \beta$ is a regexp with $L(\gamma) = L(\alpha) \cup L(\beta)$

By induction hypothesis there are NFAs N_α and N_β such that $L(N_\alpha) = L(\alpha)$ and $L(N_\beta) = L(\beta)$. Build

N_γ =



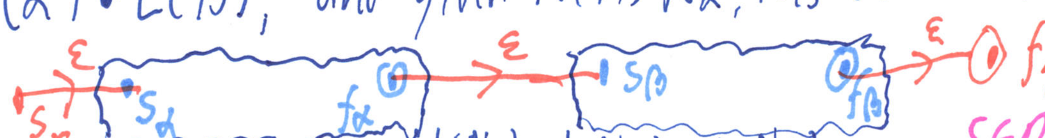
$\therefore L(N_\gamma) = L(N_\alpha) \cup L(N_\beta)$

PARALLEL CIRCUIT

- $\gamma =_{def} \alpha \cdot \beta$ is a regexp with

$L(\gamma) = L(\alpha) \cdot L(\beta)$, and given NFAs N_α, N_β as above, we build

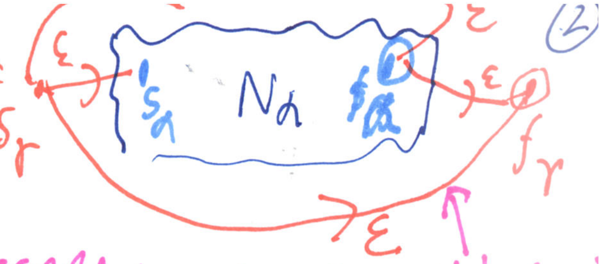
N_γ =



$L(N_\gamma) = L(N_\alpha) \cdot L(N_\beta)$

SERIES CIRCUIT

• $\gamma =_{def} (\alpha^*)$ is a regexp, $N_\gamma =$



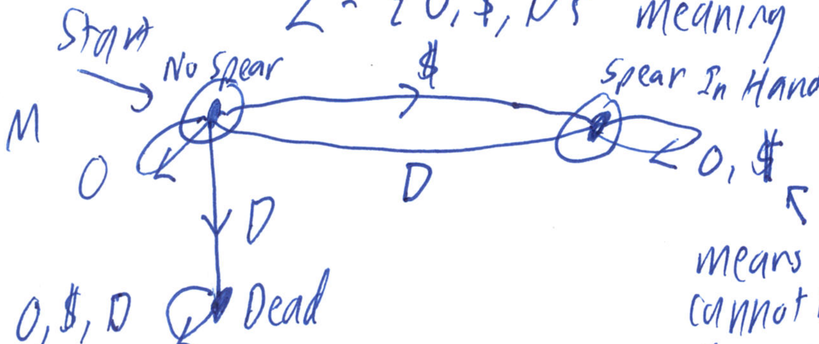
$L(\gamma) = L(\alpha)^*$, and given N_α, build

FEEDBACK CIRCUIT (with bypass)

Then $L(N_\gamma) = L(N_\alpha)^*$. ⊠

The lecture included a long demo of the "Turing Kit" software (a still-working JAR file from 1998). Here is a drawing of the DFA that was shown:

$\Sigma = \{0, \$, D\}$ meaning "empty room", "spear", and "dragon".



means that you cannot hold more than one spear

$L(M) = \{x \in \Sigma^* : x$ represents a "dungeon" in which the player exits alive$\}$.

$L(M) = \underline{(0 + \$(0+\$)^*D)^* \cdot (\epsilon + \$(0+\$)^*)}$

The regular expression "means" that to come back to the start state without being killed, either we see an empty room (0) or pick up a spear ($). We can't pick up any more spears, and the only way to be back in the no-spear state is if — after zero or more rooms with 0 or $ — we get and kill a dragon D. This last sentence represents the $\$(0+\$)^*D$ part. Since we can come back to start any number of times and stay alive, what we will soon call $\underline{\underline{L_{ss}}}$ equals $(0 + \$(0+\$)^*D)^*$. If the dungeon ends so we exit there (ϵ), fine. Or if we pick up a spear and get any number of dragon-free rooms, then we exit with a spear in hand ($\$(0+\$)^*$).