2014

Regular expressions: A language for describing patterns in strings. We define regular expressions inductively and then give a semantics.

Fix an alphabet Σ of symbols.

(1) ϕ is a regular expression

(2) ε is a regular expression

(3) $a (\in \Sigma)$ is a regular expression

(4) If R, S are regular expressions so is $R \bullet S$

(5) If R, S are regular expressions so is $R + S$

(6) If R is a regular expression so is R^*

Examples : Fix $\Sigma = \{a, b\}$

(i) $ab + \varepsilon$ (ii) $(a^* b)^*$ (iii) $a^* + b^*$ (iv) $a a^* b$ (v) ϕ

What do they mean? This is another way of describing a language. Each expression denotes a set of strings. We give the meaning through an inductive definition :

Each expression defines a subset of Σ^*:

(i) ϕ defines the set $\phi \subseteq \Sigma^*$

(ii) ε defines the set $\{\varepsilon\}$

(iii) a defines the set $\{a\}$

~~(iii)~~ If R defines the set $\hat{R}$ and S defines $\hat{S}$ then

(iv) $R \bullet S \stackrel{def}{=} \{ w_1 \cdot w_2 \mid w_1 \in \hat{R}, w_2 \in \hat{S} \}$

(v) $R + S \stackrel{def}{=} \hat{R} \cup \hat{S}$

(vi) $R^* \stackrel{def}{=} \{ w_1 w_2 \cdots w_k \mid \text{each } w_i \in \hat{R} \} \cup \{\varepsilon\}$

$ab+\varepsilon \longrightarrow \{\varepsilon, ab\}$

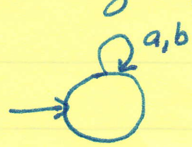
$(a^*b)^* \longrightarrow \{\varepsilon, b, bb, \dots\ ab, abab, \ .\ aab, aabab, \dots\}$

$a^*+b^* \longrightarrow \{\varepsilon, a, aa, aaa, \dots\} \cup \{\varepsilon, b, bb, bbb, \dots\}$

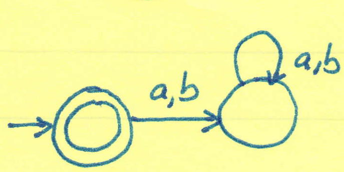
It would be hard to write verbal or even mathematical descriptions of these sets without the notation. Our regular expressions are <u>almost</u> the same as reg. exp used in systems. Why are they called regular expressions?

<u>Thm</u> (Kleene) The language defined by any regular expression is a regular language i.e. it can be recognized by an NFA+ε (or NFA or DFA). Furthermore, every regular language can be described by a regular expression.

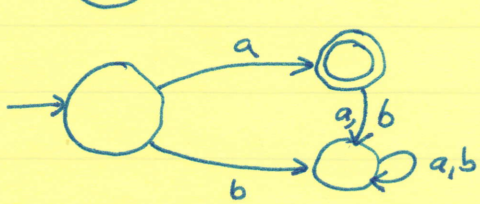
Proof (I) From reg. exp to NFA+ε:

(i) 

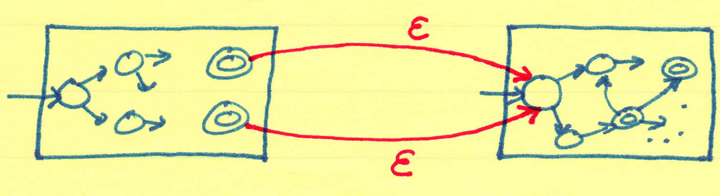
recognizes $\emptyset$

(ii) 

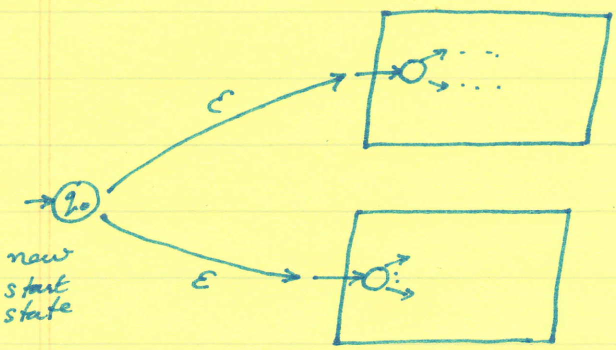
recognizes ε

(iii) 

recognizes a

(iv) 

(v) Given machines to recognize $\hat{R}$ and $\hat{S}$ we construct an NFA-ε to recognize $\hat{R} \cup \hat{S}$:



$M_1 = (S_1, s_1, \delta_1, F_1)$

$M_2 = (S_2, s_2, \delta_2, F_2)$

New machine (NFA+ε)

States $= S_1 \cup S_2 \cup \{q_0\}$

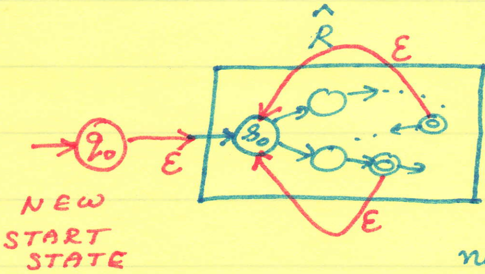
Start state $= q_0$

$$\Delta(q,a) = \begin{cases} \{\delta_1(q,a)\} \text{ if } q \in S_1 \\ \{\delta_2(q,a)\} \text{ if } q \in S_2 \\ \{s_1, s_2\} \text{ if } q = q_0 \ \& \ a = \varepsilon. \end{cases}$$

$a \in \Sigma \cup \{\varepsilon\}$

Final states $= F_1 \cup F_2$

(vi) Given a DFA to recognize $\hat{R}$ we construct a m/c to recognize $\hat{R^*}$: The new states and new transitions are in red.



Add a new start state and make it an accept state (ε is <u>always</u> in $\hat{R^*}$). Put new ε-moves from the old accept states to the <u>old</u> start state.

EXERCISE : Formalize this as in case (v)

REMARK: It does not work to just make s_0 an accept state & not bother to put in a new accept state.

EXERCISE : Explain why not. Give an example.

<u>EXAMPLE</u> : $L_1 = ab^*a$ $\quad L_2 = (ba)^*$ $\quad L_1 \cdot L_2 = ab^*a(ba)^*$