Regular Expressions

21 January 2021 00:33

We saw some of these already $01^* = 0 \cdot 1^*$

This is similar to math notation : $(3+5)*4$

Similarly, we allow combination of regular operations.

Precedence : Star, Concatenation, Union

Unless parenthesis are there

<u>Def 1.52 :</u> R is a regular expression over Σ

if R is

1. a for some $a \in \Sigma$.
2. ε
3. $\emptyset$
4. $R_1 \cup R_2$ where R_1 and R_2 are regular expressions.
5. $R_1 \circ R_2$ " "
6. R_1^* where R_1 is a reg. exp.

Regular expression denotes a set (or a language) and not a single string. For example, and not a single string.

Reg exp a denotes $\{a\}$

Reg exp ε denotes $\{\varepsilon\}$

Reg exp 01^* denotes $\{0, 01, 011, 0111, \ldots\}$

Reg exp ϕ denotes the empty language.

<u>Notation</u>: R^+ is repetition where R appears at least once.

$$R^+ = R R^*$$

$$R^+ \cup \{\varepsilon\} = R^*$$

R^k = k repeats of R.

$L(R)$ = language denoted by R.

<u>Examples</u>:

$0^* 1 0^*$: Has exactly one 1.

$\Sigma^* 1 \Sigma^*$: Has at least one 1.

$\Sigma^* 00 \Sigma^*$: Has 00 as a substring

$(\Sigma\Sigma)^*$: Strings of even length.

Read Examples 1.51 and 1.53 from the book.

$1^* \phi = \phi$. $1^* \varepsilon = 1^*$ $\phi^* = \{\varepsilon\}$

$(0 \cup \varepsilon) 1^* = 01^* \cup \varepsilon 1^* = \underline{01^* \cup 1^*}$.

$(0 \cup \varepsilon)1^* = 01^* \cup \varepsilon 1^* = \underline{01^* \cup 1^*}$.

$R \cup \phi = R$ $\qquad$ $R \cup \varepsilon$ need not be R

$R \circ \phi = \phi$ $\qquad$ $R \circ \varepsilon = R$.

If $R = 0$, $L(R) = \{0\}$

$\{xy \mid x \in A, y \in B\}$ $\qquad$ $L(R \cup \varepsilon) = \{0, \varepsilon\}$

$L(R \circ \phi) = \phi$.

A machine / compiler can parse a reg. exp. and analyse if it is in the correct form.

<u>Theorem 1.54</u>: A language is regular <u>iff</u> some regular expression describes it.

Yet another characterization for regular languages.

Proved in two parts IF and ONLY IF.
$(\Leftarrow)$ $\qquad$ $(\Rightarrow)$

<u>Lemma 1.55</u>: If a regular expression describes a language, then it is regular.

<u>Proof</u>: Given a regular expression R, we will NFA that recognizes $L(R)$.

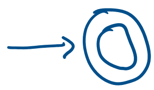
Proof. Given a

Construct an NFA that recognizes $L(R)$.

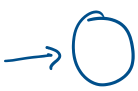
(1). $R = a$ for some $a \in \Sigma$. Then $L(R) = \{a\}$.

$N = \{ \{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\} \}$.

(2) $R = \varepsilon$. Then $L(R) = \varepsilon$.



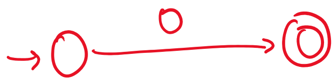
(3) $R = \phi$. Then $L(R) = \phi$.



(4) $R = R_1 \cup R_2$
(5) $R = R_1 \circ R_2$
(6) $R = R_1^*$

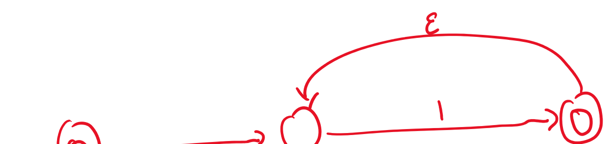
} Follows using closure rules.

Example: $01^* \cup 1$

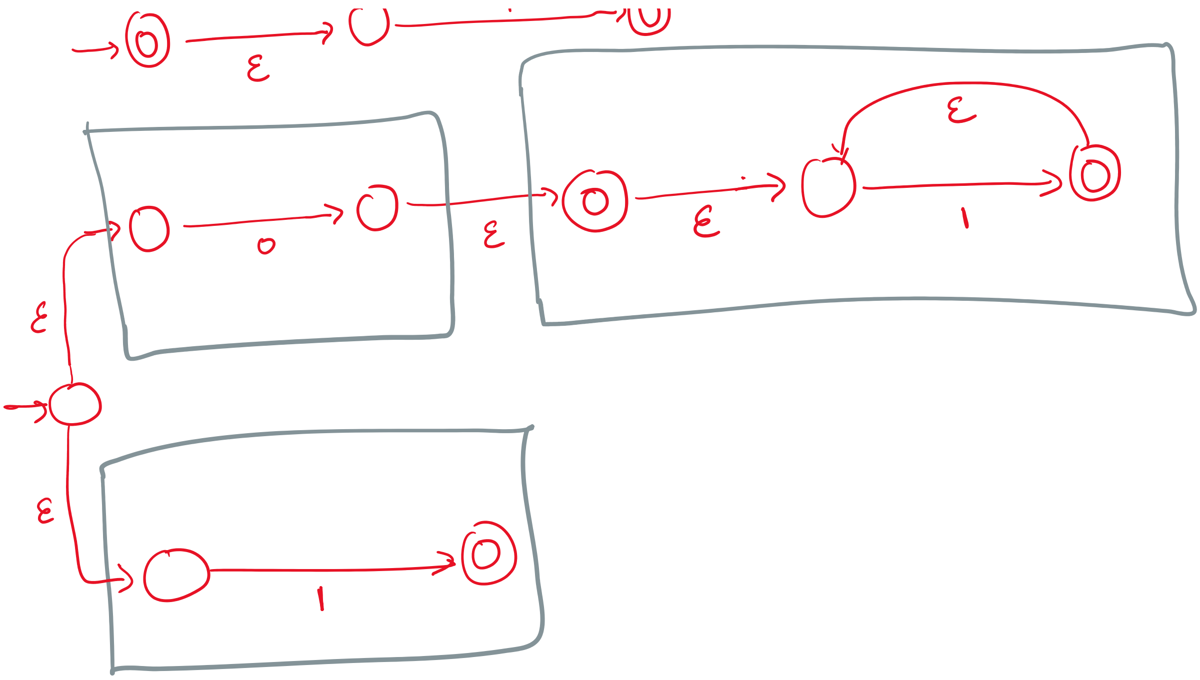


: 0

: 1



: 1^*



Read Examples 1.56 and 1.58.

Lemma 1.60: If a language is regular, then a regular expression describes it.

Regular $\Rightarrow$ Recognized by DFA

$\Rightarrow$ Recognized by GNFA.

$\Rightarrow$ Described by a regular expression.

GNFAs: NFA that has regular expressions on its transitions, not just symbols in Σ or ε.

Def 1.64: A generalized nondeterministic [illegible] automaton is a 5-tuple finite automaton is a 5-tuple

$(Q, \Sigma, \delta, q_{start}, q_{accept})$, where

$\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \to \mathcal{R}$

Set of all reg. exp. over Σ.

$q_i \xrightarrow{R} q_j$

If $\delta(q_i, q_j) = R$, this means that the automaton transitions from q_i to q_j when it sees $w \in R$.

A GNFA accepts $w \in \Sigma^*$ if w can be written as $w_1 w_2 \cdots w_k$ where $w_i \in \Sigma^*$ and there exists a sequence of states $q_0 q_1 q_2 \cdots q_k$ such that

(1) $q_0 = q_{start}$

(2) $q_k = q_{accept}$

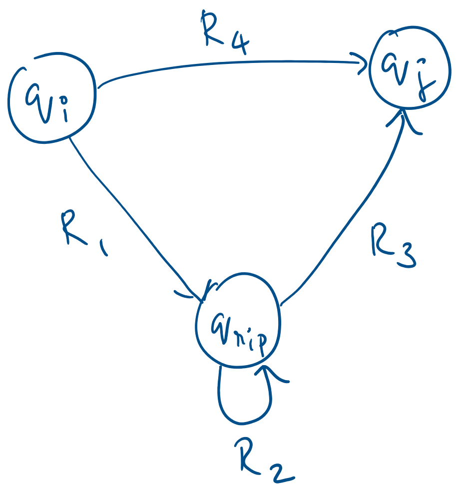
(3) For each i, $w_i \in L(R_i)$ where

$R_i = \delta(q_{i-1}, q_i)$.

Suppose A has a 6-state DFA. This can be converted into an 8 state GNFA, then to 7-state GNFA, 6-state GNFA, and so on till 2-state GNFA.

Then we get $L(R) = A$.

Main ideas of the proof:



Convert(G): Returns R or recursively calls itself.

1. k = no. of states of G.
2. If $k = 2$, then return R.

2. If $k = 2$, then return $\wedge$.

3. If $k > 2$, select a state q_{rip} such that $q_{rip} \in Q \setminus \{q_{start}, q_{accept}\}$. For all pairs q_i, q_j which is not q_{rip}, do the above transformation.

$$G' = (Q', \Sigma, \delta', q_{start}, q_{accept})$$

$$Q' = Q \setminus \{q_{rip}\}.$$

$$\delta'(q_i, q_j) = R_1 R_2^* R_3 \cup R_4.$$

4. Call Convert(G').

Claim 1.65: For any GNFA G, Convert(G) is equivalent to G.

Proof: If $k = 2$, this is true by the definition of GNFA. If $k > 2$, we will show that G' is equivalent to G.

Suppose G accepts w. Let G go through the states $q_{start}\, q_1\, q_2 \ldots q_{accept}$.

If q_{rip} is not in the above sequence, then G' accepts w because all the transitions of the old transitions.

If q_{rip} is present between q_i and q_{i+1},

$$q_i\; q_{rip}\; q_{rip}\; q_{rip}\; q_{i+1}$$

then $R_1 R_2^* R_3 \cup R_4$ describes this as well.

Q: Is this enough to show that G and G' are equivalent?

By induction hypothesis, convert (G') is equivalent to G'. Thus convert (G) is equivalent to G.

Read Example 1.66 and 1.68.