Difference between DFA and NFA.

DFA: each possible input determines the resulting state q' uniquely. Each input causes a state change, and the new state is completely determined by the input, moreover, the automaton can change state only after reading an input.

If start a DFA in its initial state and input some word w, the state q in which the DFA ends up is completely determined by w; this is meant by calling it deterministic, and we say that the word is accepted if the state is an accepted state

NFA:
- some inputs may allow a choice of resulting states → many possibilities.
- some may cause the automaton to choke → no new state corresponding to that input
- May change state to some new state q' without reading any input → λ.

Therefore, if start an NFA in its initial state and input word w, there may be several possible states in which it can end up, consequently you can't predict from w alone in exactly which state the automaton will finish, this is meant by calling it nondeterministic

But as long as at least one of them is an accepted state, we say automaton accepts the word.

DFA cannot use λ transition while NFA can.

Properties of Regular Language.

For regular language L_1 and L_2,

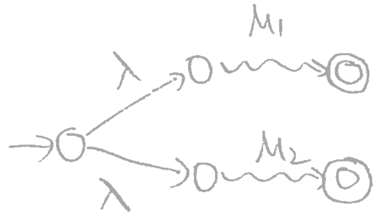
- Union : $L_1 \cup L_2$
- concatenation : $L_1 L_2$
- star : L_1^*
- Reversal : L_1^R
- complement : $\bar{L}_1$
- intersection : $L_1 \cap L_2$

} Are regular languages

Take two languages : L_1, L_2, $L(M_1) = L_1$, $L(M_2) = L_2$.

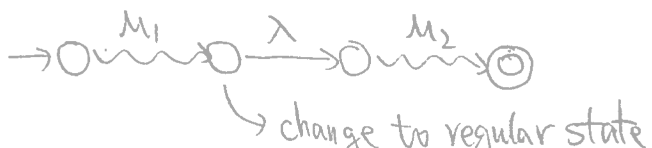
Union :

i). NFA for $L_1 \cup L_2$.



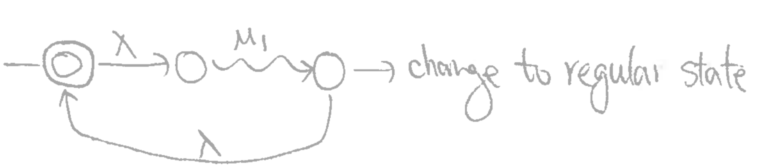
$w \in L_1 \cup L_2 \Leftrightarrow w \in L_1$ or $w \in L_2$

ii). Concatenation : $L_1 L_2$:



$w \in L_1 L_2 \Leftrightarrow w = w_1 w_2 : w_1 \in L_1$ and $w_2 \in L_2$

iii). Star : L_1^* :



$w \in L^* \Leftrightarrow w = w_1 w_2 \cdots w_k : w_i \in L$ or $w = \lambda$

iv) Reverse : L^R. i). Reverse all transitions ii). { initial state → accepting state, accepting state → initial state.

$L(M') = L^R$

V). Compliment : $\overline{L_1}$. i). Take the $\boxed{\text{DFA}}$ that accepts L.

ii). Make all accepting states regular, and make all regular states accepting states.

NFAs cannot be used for complement

VI) Intersection : $L_1 \cap L_2$.

$$L(M) = L(M_1) \cap L(M_2)$$

Regular Expressions

Regular Expressions describe regular languages.

Given regular expressions r_1 and r_2 , $\left\{ \begin{array}{l} r_1+r_2 \\ r_1 \cdot r_2 \\ r_1^* \\ (r_1) \end{array} \right.$ Are regular expressions.

Primitive regular expression: ϕ, λ, α ?

$L(r)$: language of regular expression.

e.g.: $L((a+b\cdot c)^*) = \{\lambda, a, bc, aa, abc, \dots\}$

For primitive regular expressions: $L(\phi)=\phi$, $L(\lambda)=\{\lambda\}$, $L(a)=\{a\}$.

For regular expressions r_1 and r_2: $L(r_1+r_2) = L(r_1) \cup L(r_2) = \{r_1, r_2\}$

$L(r_1 \cdot r_2) = L(r_1)L(r_2) = \{r_1\}\cdot\{r_2\}$

$L(r_1^*) = (L(r_1))^* = \{r_1\}^*$

$L((r_1)) = L(r_1) = \{r_1\}$

$L((a+b)\cdot a^*) = L(a+b)\cdot L(a^*) = (L(a) \cup L(b))(L(a))^*$

$= (\{a\} \cup \{b\})(\{a\})^* = \{a, b\}\cdot\{\lambda, a, aa, \cdots\}$

$= \{a, aa, aaa, \cdots, b, ba, baa, \cdots\}$

Regular expressions r_1 and r_2 are equivalent if $L(r_1) = L(r_2)$.

e.g.: $r_1 = (1+01)^*(0+\lambda)$

$r_2 = (1^*011^*)^*(0+\lambda) + 1^*(0+\lambda)$

$L(r_1) = L(r_2) = L = \{$all strings without substring 00$\}$

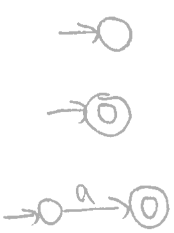
Theorem: $\left\{\begin{matrix}\text{Languages generated}\\ \text{by regular Expressions}\end{matrix}\right\} = \{\text{Regular languages}\}$

Proof: i). $\left\{\begin{matrix}\text{Languages generated}\\ \text{by regular expressions}\end{matrix}\right\} \subseteq \{\text{regular languages}\}$

Namely, for any regular expression r, the language $L(r)$ is regular.

Proof by induction on the size of r.

Induction Basis: given primitive regular expressions ϕ, λ, a.



$\left.\begin{matrix}\phi & : L(\phi) = \phi \\ \lambda & : L(\lambda) = \{\lambda\} \\ a & : L(a) = \{a\}\end{matrix}\right\}$ regular languages

Inductive Hypothesis: suppose that for regular expressions r_1 and r_2, $L(\lambda_1)$ and $L(\lambda_2)$ are regular languages.

We will prove: $L(r_1+r_2)$, $L(r_1 r_2)$, $L(r_1^*)$. $L((r_1))$ are regular languages

Since we know regular languages are closed under: union, concatenation, star, they are regular languages.

ii). {regular languages} ⊆ {languages generated by regular expressions}

Namely, for any regular language L, there is a regular expression r with $L(r)=L$

We will convert an NFA that accepts L to a regular expression.

Since L is regular, there is a NFA M that accepts it.

$L(M)=L$

From M construct the equivalent generalized transition graph, in which transition labels are regular expressions. $\rightarrow \bigcirc \xrightarrow{a,b} \circledcirc \;\Rightarrow\; \rightarrow \bigcirc \xrightarrow{a+b} \circledcirc$

Then reducing the states, and the resulting state r will have: $L(r)=L(M)=L$

By repeating the process until two states are left, the resulting expression r has:

$$L(r)=L(M)=L.$$

Standard representations of regular languages:

- DFAs
- NFAs
- Regular expressions.