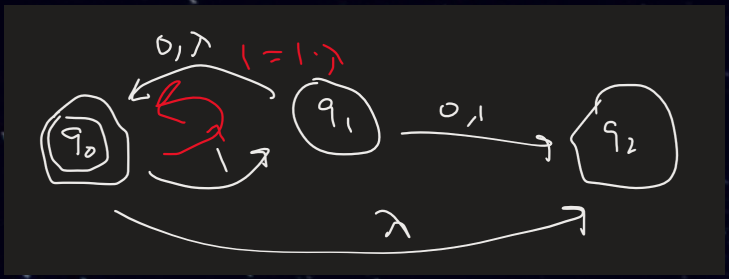
1 Review



We can see 0 is not accepted, but 1 is accepted.

$\delta : Q \times (\Sigma \cup \{\lambda\}) \to P(Q)$

For all $q \in Q$, let $E(q)$= "set of states reachable from q by following 0 or more λ-transitions"

$E(q_0) = \{q_0, q_2\}$, $E(q_1) = \{q_0, q_1, q_2\}$, $E(q_2) = \{q_2\}$

extended transition function $\delta^* : Q \times \Sigma^* \to P(Q)$

$\delta^*(q, \lambda) = E(q)$ for all $q \in Q$

$\delta^*(q, xa) = \bigcup\{E(q'') \mid q'' \in \delta(q', a)$ for some $q' \in \delta^*(q, x)$ for all $x \in \Sigma^*, q \in Q, a \in \Sigma\}$

$q' \in \delta^*(q, x)$ if and only if there is a path from q to q' labelled by x.

$$\begin{aligned} L(M) &= \{x \in \Sigma^* \mid \delta^*(q_0, x) \cap F \neq \varnothing\} \\ &= \{x \in \Sigma^* \mid \text{ there is a path from } q_0 \text{ to } a \text{ final state (in F) labelled by } x\} \end{aligned}$$

2 Extend δ^*

We can also extend δ^* to $P(Q) \times \Sigma^*$

$\delta^*(Q', x) = \bigcup\{\delta^*(q, x) \mid q \in Q'\}$ for all $Q' \in P(Q)$ and $x \in \Sigma^*$.

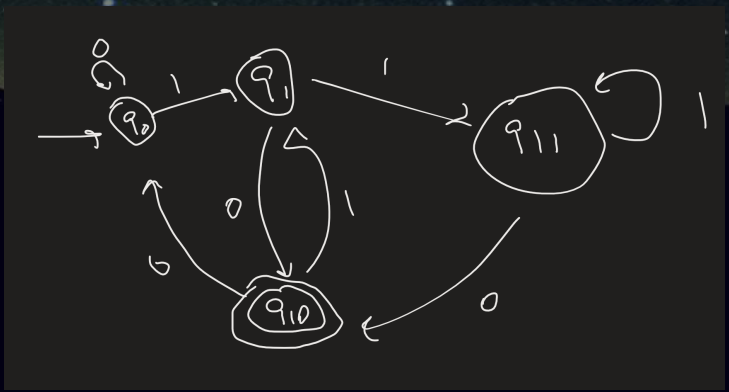
Alternatively we can also give a recursive definition (or defined recursively by):

$$\delta^*(Q', \lambda) = \bigcup\{E(q) \mid q \in Q'\}$$

for all $a \in \Sigma, x \in \Sigma^*$,

$$\delta^*(q_0, xa) = \bigcup\{E(q'') \mid q'' \in \delta(q', a) \text{ for some } q' \in \delta^*(Q', x)\}$$

Why is this useful? We can prove by induction $\forall x \in \Sigma^*.\forall y \in \Sigma^*.\forall Q' \in P(Q).\delta^*(Q', xy) = \delta^*(\delta^*(Q', x), y)$

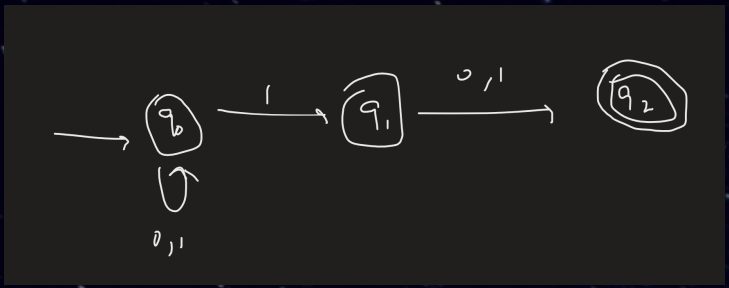


$L(D) = \{x \in \{0,1\}^* \mid$ the second last letter of x is 1$\}$

3 Convert DFA to NFA and NFA to DFA

Given an NFA N such that $L(N) = L(D)$.

1. View the DFA as an NFA: change $\delta(q, a) = q'$ to $\delta(q, a) = \{q'\}$ for all $q \in Q$. for all $a \in \Sigma$; and add $\delta(q, \lambda) = \varnothing$.



2. we can see this is a easier NFA compared to the previous one.

We can always convert DFA to NFA, but how about the converse.

Theorem 1

For every NFA $N = (Q, \Sigma, \delta, q0, F)$, there is a DFA $D = (Q', \Sigma, \delta', q_0, F')$ such that $L(D) = L(N)$.

We are going to prove this by construction, take an arbitrary NFA and then we will construct the DFA one.

And this proof technique is ok well we will use generalization.

Proof by generalization.
Let $N = (Q, \Sigma, \delta, q_0, F)$ be an arbitrary NFA. Now we will construct a DFA as we are reading the input strings.
That is, construct a DFA that keeps track of the subset of states N could be in as it reads the input.
Subset construction:
Let $D = (Q', \Sigma, \gamma, q_0, F')$
$Q' = P(Q)$
$q_0' = E(q_0)$
$\gamma(S, a) = \bigcup\{\delta^*(q, a) \mid q \in S\}, S \in P(Q).x \in \Sigma$
$F' = \{S \in P(Q) \mid S \cap F \neq \varnothing\}$
This isn't very hard, in fact all the definitions are in there.
Now prove $L(D) = L(N)$.
Let $p(w) =$"$\gamma^*(q_0', w) = \delta^*(q_0, w)$ for $w \in \Sigma^*$."
We will prove $\forall w \in \Sigma^*.p(w)$ by structural induction.
Base Case $w = \lambda$.
$\gamma^*(q_0', \lambda) = q_0')$ since D is deterministic.
$\delta^*(q_0, \lambda) = E(q_0)$ since N is nondeterministic.
By construction $q_0' = E(q_0)$, so $p(\lambda)$ is true.
Constructor Case:
Let $x \in \S^*, a \in \Sigma$, assume $p(x)$ is true.

$$\begin{aligned}
\gamma^*(q_0', x) &= \delta^*(q_0, \lambda) \\
\gamma^*(q_0', xa) &= \gamma(\gamma^*(q_0', x), a) \\
&= \bigcup\{\delta^*(q, a) \mid q \in \gamma^*(q_0', x)\} \text{ by construction} \\
&= \bigcup\{\delta^*(q, a) \mid q \in \delta^*(q_0, x)\} \text{ by induction hypothesis / substitution} \\
&= \delta^*(q_0, xa) \\
&= \delta^*(q_0, w)
\end{aligned}$$

Hence $p(xa)$ is true.
By structural induction, $\forall w \in \Sigma^*.p(w)$ is true.

Quod Erat Demo■

So, $w \in L(D)$

if and only if $\gamma^*(q_0', w) \in F'$ since D is a DFA, and so $L(D) = \{w \in \Sigma^* \mid \gamma^*(q_0', w) \in F'\}$, and this is true

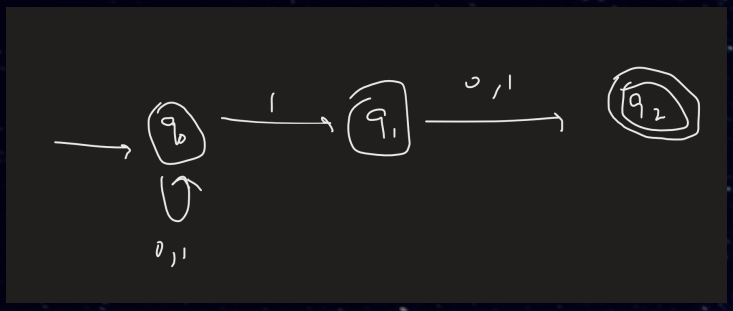
if and only if $\gamma^*(q_0', w) \cap F \neq \varnothing$ since $F' = \{S \in P(Q) \mid S \cap F \neq \varnothing\}$,

if and only if $\delta^*(q_0, w) \cap F \neq \varnothing$ since $p(w)$ is true so $\gamma^*(q_0', w) = \delta^*(q_0, w)$,

if and only if $w \in L(N)$ since $L(n) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \varnothing\}$

Hence $L(D) = L(N)$.

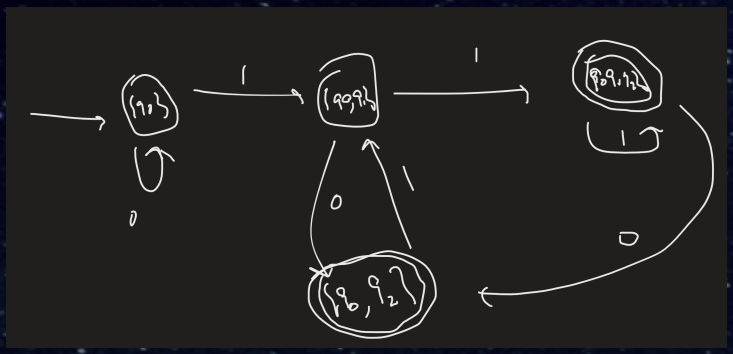
Now consider



We want to convert this NFA to DFA.

We can see $q_0' = \{q_0\}$,

...



We first find the paths, the find the final states, we can see this DFA is the exact same as our original one.

Now we are going to talk about closure results.

4 Closure Results

Let FA denote a finite automata.

Suppose $L_1, L_2 \subseteq \Sigma^*$ are accepted by FA.

Then $\overline{L_1} = \Sigma^* - L_1 = \{x \in \Sigma^* \mid x \notin L_1\}$ being the complement of L_1.

$L_1 \cup L_2 = \{x \mid (x \in L_1) \text{ OR } (x \in L_2)\}$ being the union

$L_1 \cap L_2 = \{x \mid (x \in L_1) \text{ AND } (x \in L_2)$ being the intersection

$L_1 - L_2 = \{x \in \Sigma^* \mid ((x \in L_1) \text{ AND } (x \notin L_2))\}$ are accepted by FA.

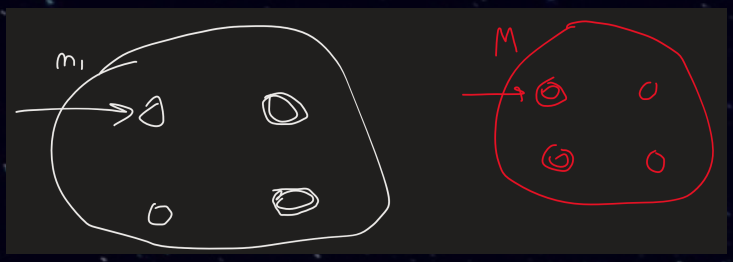
Claim 1

The family of language $\{L \in \Sigma^* \mid L \text{ is accepted by a FA}\}$ is closed under complement, union, intersection, and difference.

Let $M_1 = (Q_1, \Sigma, \delta, q_1, F_1)$ be a DFA such that $L(M_1) = L_1$ and let $M_2 = (Q_2, \Sigma, \delta, q_2, F_2)$ be a DFA such that $L(M_2) = L_2$.

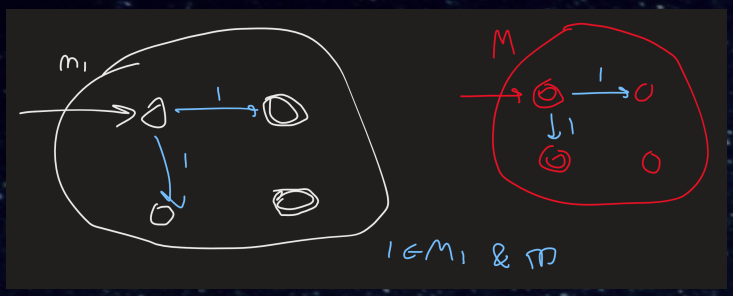
W.l.o.g. suppose $Q_1 \cap Q_2 = \varnothing$ (otherwise we can rename them).

We will first prove other things.



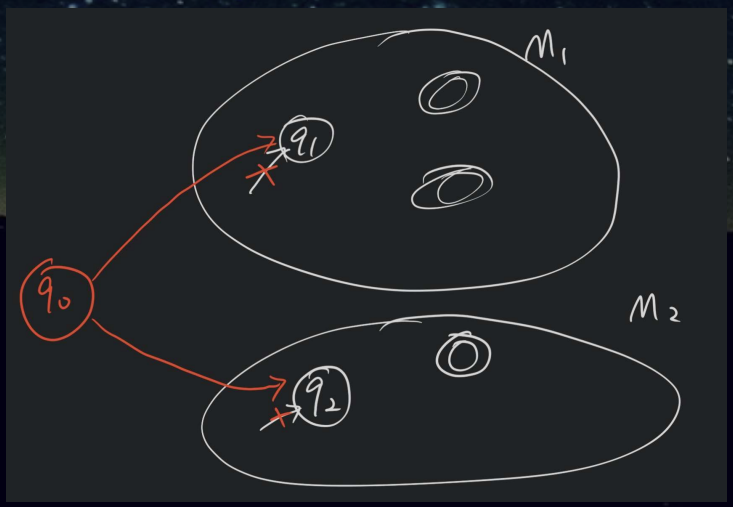
Proof of Complement. By switching all states in M_1 from final to non-final and non-final to final, we can claim M aceepts the strings oppositely compared to M_1.
That is, let $M = (Q, \Sigma, \delta_1, q_1, Q_1 - F_1)$, then we can claim $L(M) = \Sigma^* - L(M_1)$.
Let $x \in \Sigma^*$ Then $x \in L(M)$
if and only if $\delta_1^*(q_1, x) \in Q_1 - F_1$
if and only if (This is true only for DFA) $\delta_1^*(q_1, x) \notin F_1$
if and only if $\delta_1^*(q_1, x) \notin L(M_1) = L_1$

QUOD ERAT DEM■



We can see 1 is in both M_1 and M, thus not works for NFA.

Remark 1. The construction may not work if M_1 is not an NFA.
$\delta_1^*(q_0, x) \cap (Q_1 - F_1) \neq \varnothing$ and $\delta_1^*(q_1, x) \cap F_1 \neq \varnothing$, for an NFA both can be true.



proof of union. More formally, let q_0 be a new state,

$$M = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup F_2)$$

where $q_0 \notin Q_1 \cup Q_2$
$\delta(q_0, \lambda) = \{q_1, q_2\}$
$\delta(q_0, a) = \varnothing$ for all $a \in \Sigma$
$\delta(q, a) = \{\delta_1(q, a)\}$ if $q \in Q_1, a \in \Sigma$
$\delta(q, a) = \{\delta_2(q, a)\}$ if $q \in Q_2, a \in \Sigma$
$\delta(q, \lambda) = \varnothing$ for all $q \in Q_1 \cup Q_2$
Then we can claim that $L(M) = L_1 \cup L_2$.

QUOD ERAT DEM■

For intersection, we can see $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Another way is to run both machines in parallel, accept only if both accept.

Formally,

Proof of intersection. $M = (Q \times Q, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$
$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ for all $p_1 \in Q_1, p_2 \in Q_2, a \in \Sigma$
Then we can claim $L(M) = L(M_1) \cap L(M_2)$.

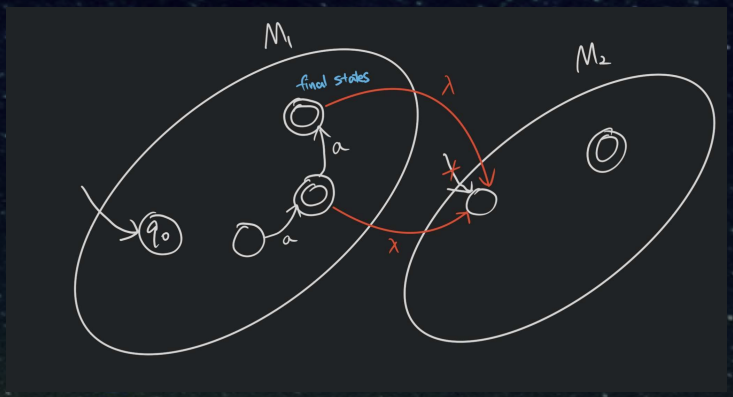
QUOD ERAT DEM■

For difference,

Proof of difference.
We can simply use intersection and construct $L_1 - L_2 = L_1 \cap \overline{L_2}$.

QUOD ERAT DEM■

For concatenation,



Proof of concatenation.
$L_1 \cdot L_2 = \{x \cdot y \mid x \in L_1 \text{ AND } y \in L_2\}$
if $L_1 = \{a, bb\}$, $L_2 = \{\lambda, c\}$
Then

$$L_1 \cdot L_2 = \{a, ac, bb, bbc\}$$
$$\neq L_2 \cdot L_1$$
$$L_1 \cdot \varnothing = L_1 = \varnothing \cdot L_1$$

To prove the family of languages accepted by FA is closed under concatenation, we want to construct 2 machines.
From the figure, we can construct $M = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$
$\delta(q, a) = \delta_1(q, a)$ if $q \in Q_1$

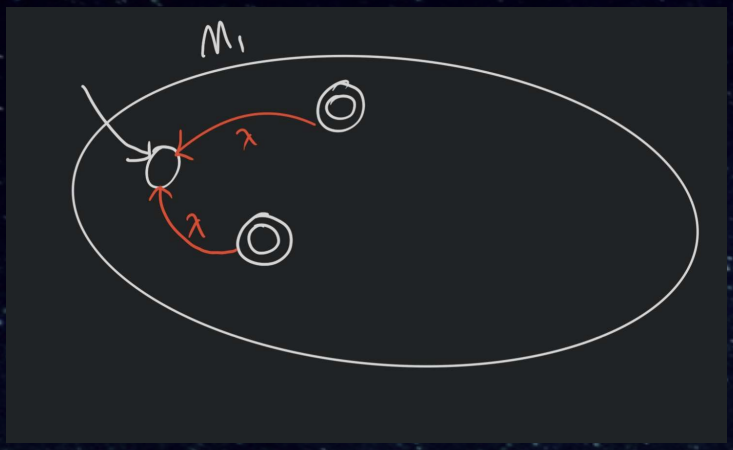
$\delta(q, \lambda) = \{q_2\}$ if $q \in F_1$
$\delta(q, a) = \delta_2(q, a)$ if $q \in Q_2$
$\delta(q, \lambda) = \varnothing$ if $q \in (Q_1 - F_1) \cup Q_2$

QUOD ERAT DEM■

5 Positive Closure

$$L_1^1 = L_1$$
$$L_1^{i+1} = L_1^i \cdot L_1 = L_1 \cdot L_1^i$$
$$L_1^+ = \bigcup_{i \geq 1} L_1^i \mid L^i = \{x_1 \cdot \ldots \cdot x_i \mid x_1 \cdot \ldots \cdot x_i \in L_1\}$$

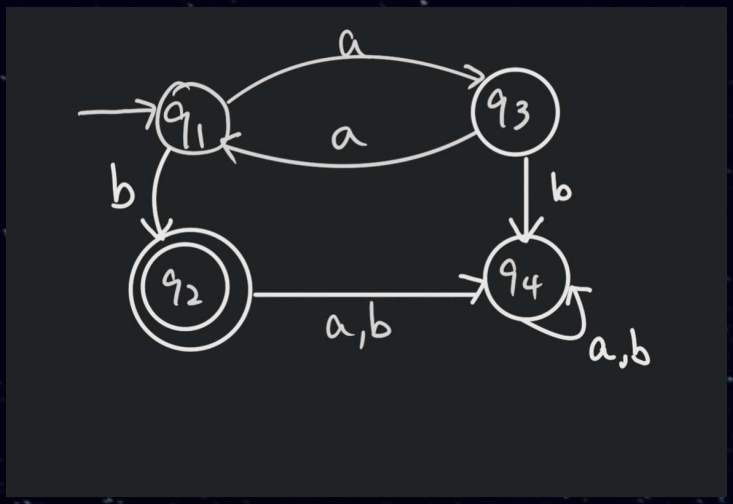
$L_1 = \{aab, c\}$
$L_1^+ = \{aab, c, aabc, caab, cc, aabaab\}$



$L(M) = (Q, \Sigma, q_1, \delta, F_1)$
$\delta(q, a) = \{\delta_1(q, a)\}$ for $q \in Q_1, a \in \Sigma$
$\delta(q_1, \lambda) = \{q_1\}$ for $q \in F_1$
$\delta(q_1, \lambda) = \varnothing$ for $q \notin F_1$
$L(M_1)^+ = L(M)$

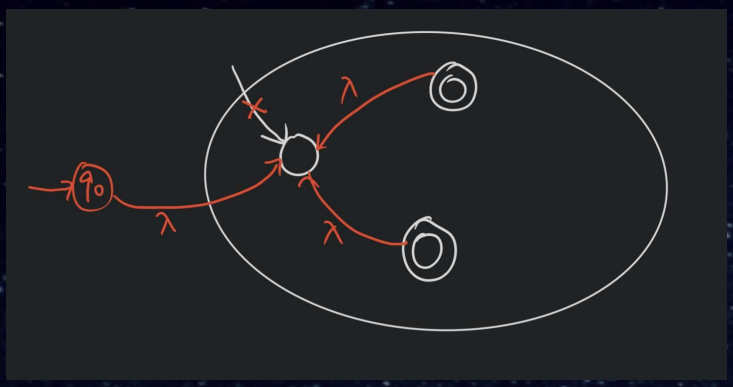
6 Star Closure

$L_1^0 = \{\lambda\} \neq \lambda$
$L_1^* = \bigcup_{i \geq 0} L^i = L_1^+ \cup \{\lambda\}$
$L_1^* = L_1^+$ if and only if $\lambda \in L_1$



Is $L(M) = (L(M_1))^*$ true?

Not always:

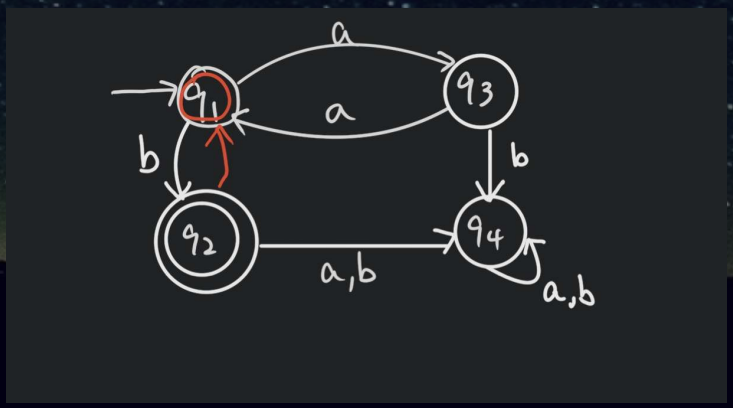


$M_1 a, aa \notin L(M_1)$ so $aa \notin L(M_1)^*$

$aa \in L(M)$

$L(M) \neq L(M_1)^*$

To resolve this,



7 Regular Expressions

Let Σ be a finite alphabet, R is a set of strings defined inductively:

Base Cases:

$\varnothing \in R, \lambda \in R$

$\Sigma \subseteq R$

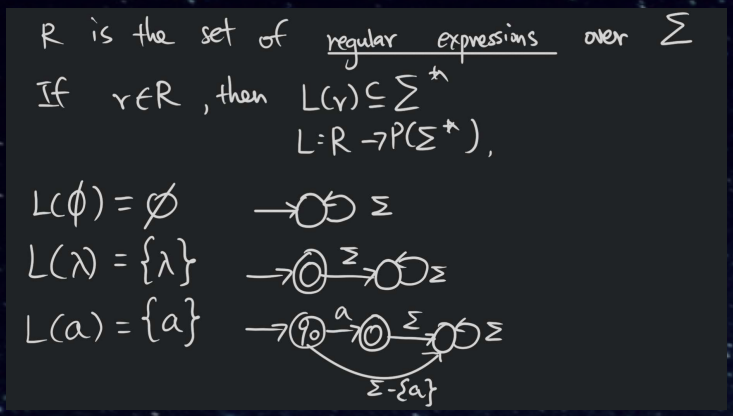
Constructor Cases:

If $r, r' \in R$, then $r \cdot r', r + r', r^* \in R$

Here R is the set of regular expressions over Σ.

If $r \in R$, then $L(r) \subseteq \Sigma^*$ is the function $L : R \to P(\Sigma^*)$,

$$\begin{aligned} L(\varnothing) &= \varnothing \\ L(\lambda) &= \{\lambda\} \\ L(a) &= \{a\} \text{ for all } a \in \Sigma \\ L(r \cdot r') &= L(r) \cdot L(r') \\ L(r + r') &= L(r) \cup L(r') \\ L(r^*) &= L(r)^* \end{aligned}$$



This is regular expression, we can also talk about generalized regular expressions.

8 Generalized Regular Expressions

Generalized regular expressions allow $\cap$, difference and complement:

$$\begin{aligned} L(r \cap r') &= L(r) \cap L(r') \\ L(r - r') &= L(r) - L(r') \\ L(\overline{r}) &= \overline{L(r)} = \Sigma^* - L(r) \end{aligned}$$

$abc + cc$ means $((a \cdot (b \cdot c)) + (c \cdot c))$

A language A is **regular** if and only if $A = L(r)$ for some $r \in R$.

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

Remark 2. For strings over $\{a, b, c\}$ that start with ab:

$$\begin{array}{ccc} L(a{\cdot}b \cdot (a + b + c)^*) & \neq & L(a{\cdot}b \cdot (a \cdot b \cdot c)^*) \\ \Downarrow & & \Downarrow \\ L(a{\cdot}b \cdot \overline{\phi}) & & \{ab, ababc, ababcabc, \ldots\} \end{array}$$

Theorem 2

Every regular language can be accepted by a finite automaton (FA).