Regular Expressions $\overset{\text{KLEENE}}{\Longleftrightarrow}$ Languages

① KLEENE Theorem $e \mapsto A_e$ (DFA)

② Reversed direction. $A_e \mapsto e$

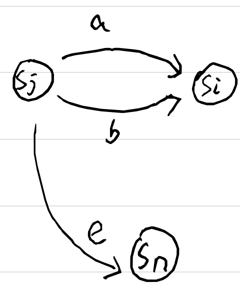
Focus ②:

DFA

$S \qquad \underline{S \to S^A}$

transition function as matrix.

$$\begin{array}{cc} & \begin{matrix} S_0 & \cdots & i & \cdots & S_k \end{matrix} \\ \begin{matrix} S_0 \\ \vdots \\ j \\ S_k \end{matrix} & \begin{bmatrix} & & & & \\ & & & & \\ & & a+b & & \\ & & & & \end{bmatrix} \end{array}$$



Data Structure for representing DFAs

then treat matrix as regex.

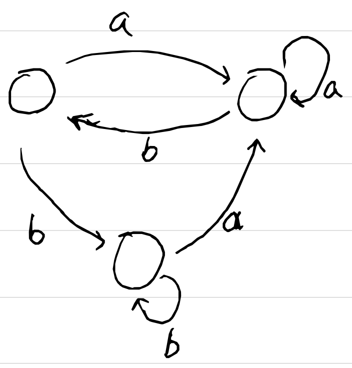
How do we calculate the $*$ of matrix?

$$\begin{array}{cc} & \begin{matrix} S_0 & \cdots & i & \cdots & S_k \end{matrix} \\ \begin{matrix} S_0 \\ \vdots \\ j \\ S_k \end{matrix} & \begin{bmatrix} & & & & \\ & & & & \\ & & a+b & & \\ & & & & \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \end{array}$$

← vector for indicating final states.

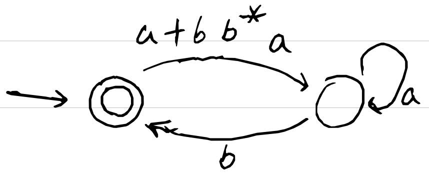
classic proof: state elemination for proving DFA → regex

At least 3 states to start with.



Q: starting state and final state.

If deleting a state, send the transition related to that state should be transitioned elsewhere. See below:



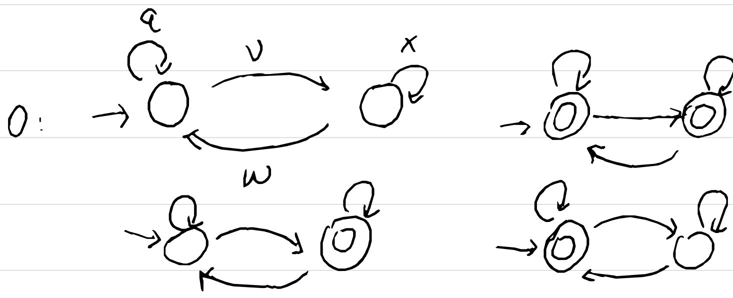
starting from the left most and come back to the final state, we can use the following regex:

$$((a+bb^*a)a^*b)^*$$

If the automaton has more than one final state we can use ε transition on all final states and have them transition to one final state.

Then we can keep doing the state-elimination until there are only 2 states left.

Below are the 4 possibilities of the final states.



Two regexes are equal ($e_1 \equiv e_2$) iff the two DFAs are equal.

KLEENE wondered if there were a set of equations that can decide syntactically whether 2 regexes are equal.

KLEENE Algebra: $K, 0, 1, +, ;, (\_)^*$

a set K that satisfies the following laws:

$$\left.\begin{array}{l} 1.\ e+e \equiv e \\ 2.\ e+f \equiv f+e \\ 3.\ (e+f)+g \equiv e+(f+g) \\ 4.\ e+0 = e \end{array}\right\} \text{join semi-lattice}$$

$$\left.\begin{array}{l} (e;f);g \equiv e;(f;g) \\ e;1 \equiv 1;e \\ e;0 \equiv 0 \equiv 0;e \end{array}\right\} \text{monoid}$$

$e;(f+g) \equiv e;f + e;g$

$(e+f);g \equiv e;g + f;g$

semi-ring

$e^* \equiv 1 + e;e^*$

$e^* \equiv 1 + e^*;e$

natural order on semiring

$e \leq f \iff e+f \equiv f$

partial order

Axiom Scheme $\left\{ \dfrac{e;x+f \leq x}{e^*;f \leq x} \right.$ ← iteratively replace x with $(e;x+f)$ to get $e^*;f$

e^* is a Least Fix Point. f is the stop branch. "+" is like branching: with the left branch, recursion continues; with the right branch, it stops.

Exercises for this lecture:

1. $x^* x^* \equiv x^*$
2. $x^* \equiv (x^*)^*$
3. $xy = y^2 \Rightarrow x^* y \equiv y z^*$
4. $(a+b)^* \equiv (a^* b)^* a^*$ (denesting rule)

Examples for Kleene Algebra:

$(2^{A^*}, \emptyset, \{\varepsilon\}, \cup, \bullet, (-)^*)$

2^{A^*} ← Regular language. $\bullet$ ← concatenation.

$(BRel, \emptyset, \Delta, \cup, \circ, (-)^*)$

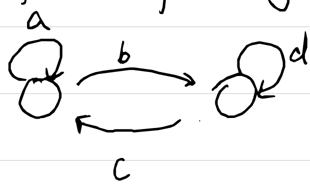
BRel ← Binary Relation; $\circ$ ← composition; $(-)^*$ ← transitive closure

$(MAT(K), 0, 1, +, \times, (\cdot)^*)$

$+$ ← pointwise plus

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} (a+bd^*c)^* & (a+bd^*c)^*bd^* \\ \cdots & (d+ca^*b)^* \end{bmatrix}$$

is the matrix star for the following DFA:



How do we use the matrix above for calculating the star of matrix of any size? See breaking down of a square matrix of size 3 below:

$$\left[\begin{array}{cc|c} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \end{array}\right]$$

From Kozen '93:

soundness

$$[\![e]\!] = [\![f]\!] \iff e \equiv_{KA} f$$

completeness.

To prove

$$\begin{tikzcd}
e \arrow[d, "\text{Thompson construction}"'] \arrow[dr, squiggly] & \equiv & f \arrow[dl, squiggly] \arrow[d] \\
\boxed{A_e} \arrow[d, "\text{def}"'] & e & A_f \arrow[d] \\
A_e' \arrow[dr, "\text{min}"'] & & A_f' \arrow[dl, "\text{min}"] \\
& M_e \equiv M_f &
\end{tikzcd}$$

DFA → unique minimal automata

$$e \xrightarrow{\text{KLEENE Algebra Axioms}} M_e$$

Do this syntactically for every arrow in the diagram.

This is helpful for verifying simple imperative programs

If b then P else q

$b_0 \vee b_1$

$b_0 \wedge \overline{b_1}$

we need to combine kleene algebra and boolean algebra: ($B \subseteq K$, or more precisely B is a sub-algebra of K)

$(K, 0, 1, +, ;, (-)^*)$ $(B, 0, 1, \overset{(\vee, \wedge)}{+, ;}, \overset{\text{not}}{\overline{(\ )}})$ in BA

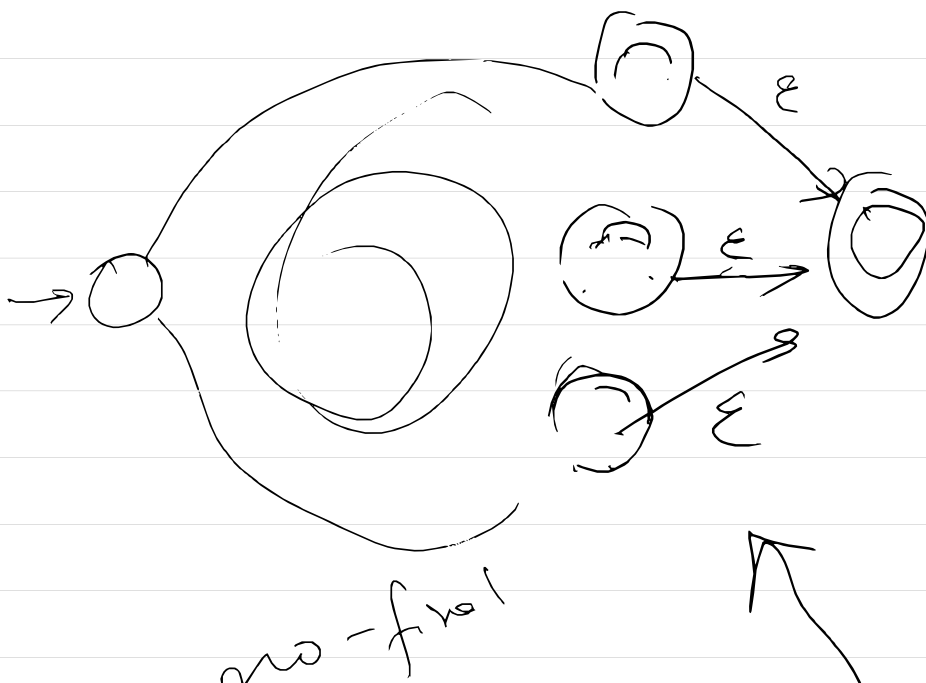
2^{A^*} $A = \{p, q, r, s, t, b_0, b_1, \cdots b_n$
$\bar{b}_0, \bar{b}_1, \cdots \bar{b}_n\}$

what should not $(\bar{\ })$ satisfy?

- $\overline{(a+b)} \equiv \bar{a} ; \bar{b}$
- $\overline{(a;b)} \equiv \bar{a} + \bar{b}$
- $\bar{0} \equiv 1$
- $a;b \equiv b;a$ conjunction is commutative
- $\bar{\bar{a}} \equiv a$

if b then p else q $\Rightarrow b;p + \bar{b};q$

while b do p $\Rightarrow (b;p)^*;\bar{b}$



Assumption : one final state

Any automaton

adding one extra state + ε ◎

eliminate until 2 st.