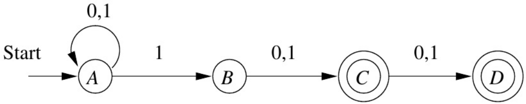
Example

The construction by state elimination works for every type of FA.
Consider NFA M



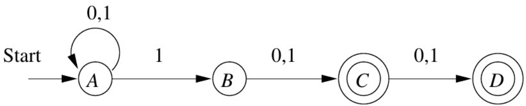
recognizing the language

Third or second to last must be a one

$$L(M) = \{w \mid w = x1b \text{ or } w = x1bc,\ x \in \{0,1\}^*,\ b, c \in \{0,1\}\}$$

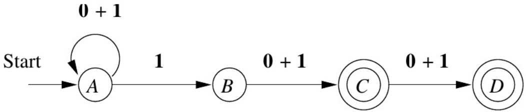
Construct from M a regular expression generating $L(M)$

Example



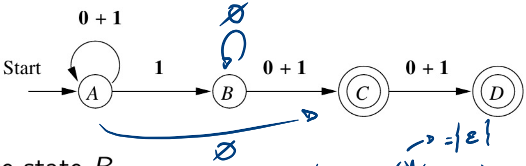
We transform M into an automaton with equivalent regular expressions at each transition

normali-
zation



Example

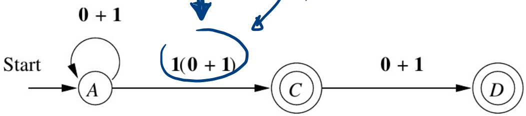
State elimination :



We eliminate state B

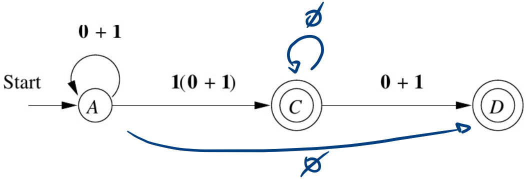
$= \{\varepsilon\}$

$\emptyset + 1\emptyset^*(0+1)$



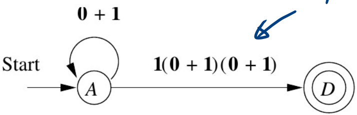
We have simplified the regular expression $\mathbf{1}\emptyset^*(\mathbf{0}+\mathbf{1})$ as $\mathbf{1}(\mathbf{0}+\mathbf{1})$, since $L(\emptyset^*) = \{\epsilon\}$

Example



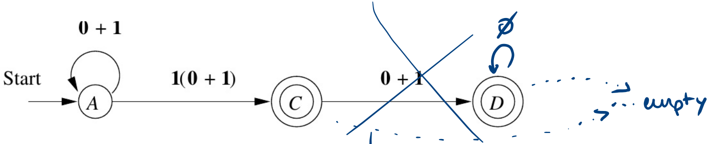
We eliminate state C resulting in M_D

$\emptyset + 1(0+1)\emptyset^*(0+1)$



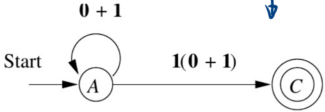
corresponding to the regular expression
$E_D = (\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$

Example



We eliminate state D resulting in M_C

just eliminate D



corresponding to the regular expression $E_C = (\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$

The desired regular expression is the sum of E_D and E_C :

$$(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1}) + (\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$$

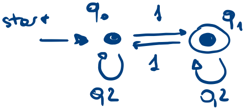
Exercise

Write a regular expression for the language L over $\Sigma = \{0, 1, 2\}$ such that, for each string in L, the sum of its digits is an odd number

Suggestion

- start specifying a DFA that accepts L
- then construct the equivalent regular expression

DFA :



REGEX :

$$(0+2)^*1\left((0+2)^*1(0+2)^*1(0+2)^*\right)^*$$

$$\Downarrow$$

$$(0+2)^*1\left\{\left[(0+2)^*1\right]^2\right\}^*(0+2)^*$$

From regular expression to ϵ-NFA

Theorem For every regular expression R we can construct an ϵ-NFA E such that $L(E) = L(R)$

Proof

We construct E with

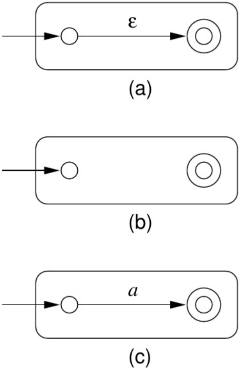
- only one final state
- no arc entering the initial state
- no arc exiting the final state

This will make it easier/safer to connect FAs

The construction uses structural induction

From regular expression to ϵ-NFA

Base Automata for regular expressions ϵ, $\emptyset$, and $\boldsymbol{a}$

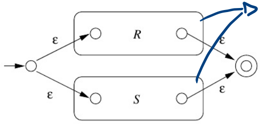


From regular expression to ϵ-NFA

Induction Automata for $\underset{1}{R+S}$, $\underset{2}{RS}$, e $\underset{3}{R}^*$ → 3 cases

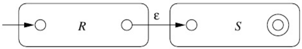
we already have this ε-NFA

1)



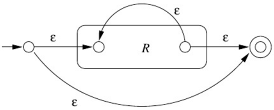
(a)

2)



(b)

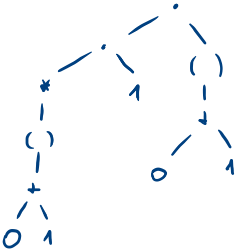
3)



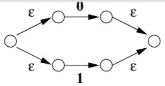
(c)

□

Example

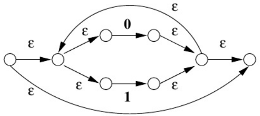


Construct ϵ-NFA for the regular expression $\underbrace{\underbrace{\underbrace{(\mathbf{0}+\mathbf{1})}_{a}^{*}}_{b}\mathbf{1}(\mathbf{0}+\mathbf{1})}_{c}$



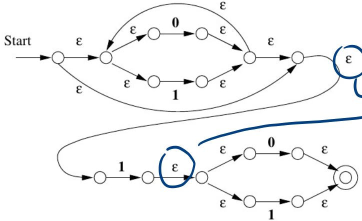
= (0+1)

(a)



= (0+1)*

(b)



those are the concatenations

(c)

Algebraic laws

There are some similarities between regular expressions and **arithmetic expressions**, if we consider the union as the sum and concatenation as the product

As for arithmetic expressions, there are similar properties for regular expressions (commutativity, distributivity, etc.)

There exists also **specific** properties for regular expressions, mainly related to Kleene's closure operator, which do not correspond to any laws of arithmetic

In the following slides, L, M, N are regular expressions, not languages

Commutativity and associativity

(generate the same language)

Union is **commutative** : $L + M = M + L$

Union is **associative** : $(L + M) + N = L + (M + N)$

Concatenation is **associative** : $(LM)N = L(MN)$

Concatenation is **not commutative** : there exist L and M such that $LM \neq ML$. **Example** : $\mathbf{10} \neq \mathbf{01}$

Identity and annihilators

Very useful in simplifying regular expressions :

$\emptyset$ is the **identity** for union: $\emptyset + L = L + \emptyset = L$

ϵ is the **left identity** and the **right identity** for concatenation : $\epsilon L = L\epsilon = L$

$\emptyset$ is the **left annihilator** and the **right annihilator** for concatenation : $\emptyset L = L\emptyset = \emptyset$

Distributivity and idempotence

Concatenation is **left distributive** over union :
$L(M + N) = LM + LN$

Concatenation is **right distributive** over union :
$(M + N)L = ML + NL$

Union is **idempotent** : $L + L = L$

Kleene closure & other operators

$(L^*)^* = L^*$ (proof in later slides)

$\varnothing^* = \epsilon$

$\epsilon^* = \epsilon$

At least one occurrence

$L^+ = LL^* = L^*L$

at least one occurrence

$L^* = L^+ + \epsilon$ or none

$L? = \epsilon + L$

At most one occurrence

Exercise with solution

Prove that the regular expressions $(R^*)^*$ and R^* are equivalent

$L((R^*)^*) = (L(R^*))^* = (((L(R))^*)^*$

$L(R^*) = (L(R))^*$

Assuming $L(R) = L_R$, we need to show $(L_R^*)^* = L_R^*$

Exercise with solution

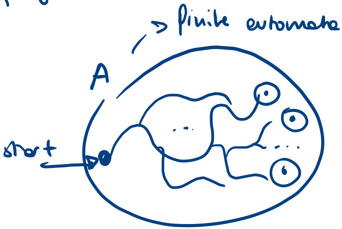
$$
\begin{aligned}
w \in (L_R^*)^* &\iff w \in \bigcup_{i=0}^{\infty} \left(\bigcup_{j=0}^{\infty} L_R^j \right)^i \\
&\iff \exists k, m \in \mathbb{N} \;:\; w \in (L_R^m)^k \\
&\iff \exists p \in \mathbb{N} \;:\; w \in L_R^p \\
&\iff w \in \bigcup_{i=0}^{\infty} L_R^i \\
&\iff w \in L_R^*
\end{aligned}
$$

In the right to left direction, choose $k = 1$ and $m = p$

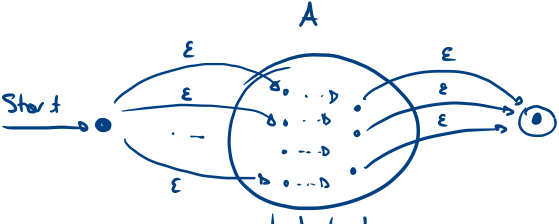
Es on an ε-NFA

$$\left(\begin{array}{l} x \in \Sigma^*, \quad \mathrm{Inf}(x) = \{ w \mid x = vwz, \; v,w,z \in \Sigma^* \} \\ \\ \mathrm{Inf}(L) = \bigcup_{x \in L} \mathrm{Inf}(x) \end{array} \right)$$

Show that if L is a regular language, then $\mathrm{Inf}(L)$ is a regular language



Imp(A):



each state that is not a trap state

each state reachable from the start state