Regular Expressions

Narges Khakpour

Department of Computer Science,
Linnaeus University

Table of Content

Proving Irregularity

- To show a language is regular, we should find an automaton that accepts it.
- How to show that a language is NOT regular? Use pumping lemma
- Pumping lemma is usually used for infinite languages

Pumping Lemma

- Any string with a length greater than a threshold, can be divided into three parts: x, v and y
- If we pump (repeat) v any number of times, the result is still in L

Theorem

For any regular language L, there exists an integer n, such that for all $x \in L$ *with* $|x| \geq n$*, there exist* $u, v, w \in \Sigma^*$*, such that*

C1 $x = uvw$

C2 $|uv| \leq n$

C3 $|v| \geq 1$

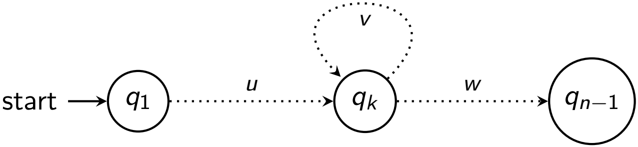
C4 *for all* $i \geq 0 : uv^i w \in L$

Pumping Lemma

- L is regular $\Rightarrow$ it satisfies conditions,
- If L satisfies the conditions, we **CANNOT always** conclude that L is regular,
- If L does not satisfy conditions $\Rightarrow$ L is NOT regular
- To prove that a languages is not regular, we should prove that the conditions holds for **NO** n.

Proof Sketch

- Let A be a DFA that is equivalent to L and has $n-1$ states.
- For all strings x with $|x| \geq n$, there is at least one state that is visited twice, because A has only $n-1$ states.



We can see that all the conditions in the theorem hold.

Example

Example

Prove that $a^n b^{2n}$ is not regular.

We should show that there is NO integer n such that the lemma's conditions hold, i.e. for all n, lemma's conditions do NOT hold.
Let $x = a^n b^{2n}$ ($|x| \geq n$) and such n exist.
If (C1-C3) hold, then $x = uvw$, $|uv| \leq n$ and $|v| \geq 1$.
Then, $u = a^j$, $v = a^k$ and $w = a^m b^{2n}$ where $j + k \leq n$ and $j + k + m = n$.
For C4 to hold, $a^j (a^k)^i a^m b^{2n} = a^{j+ki+m} b^{2n}$ should belong to L which does not hold because $j + ki + m \neq n$ for any arbitrary i (Observe that $k \geq 1$ according to C3).

Table of Content

Language Operations

Let L_1 and L_2 be two languages defined over an alphabet Σ.

- Union
  $L_1 \cup L_2 = \{w | w \in L_1 \vee w \in L_2\}$

Example

L_1 is the set of all number dividable by 5. L_2 is the set of all number dividable by 2. $L_1 \cup L_2$ is the set of all numbers either dividable by 2 or 5.

- Concatenation
  $L_1.L_2 = \{w_1 w_2 | w_1 \in L_1 \wedge w_2 \in L_2\}$

Example

If L_1 is the set of all 2-bit binary numbers and L_2 is the set of all 3-bit binary numbers. $L_1 L_2$ is the set of all 5-bit binary numbers.

Language Operations

- Power
  $L^0 = \{\epsilon\},\ L^1 = L\ ,\ L^{k+1} = L^k.L$

Example

Let $L_1 = \{a, b\}$. $L_1^2 = \{aa, ab, ba, bb\}$,
$L_1^3 = \{aaa, aba, baa, bba, aab, abb, bab, bbb\}$

- Kleene Closure
  $L^* = \bigcup_{0 \leq k} L^k$

Example

If $L_1 = \{0, 1\}$, then L_1^* is the set of all binary numbers including ϵ.

Is L^* always infinite?

Table of Content

Regular Expressions

- An algebraic way to describe languages.
- A RE is defined over an alphabet Σ
- The language that an expression E expresses is denoted by $L(E)$.
- Definition is recursive.

Regular Expressions

Definition

- **Base Cases**
  - For any symbol $a \in \Sigma$, a is a regular expression and $L(a) = \{a\}$
  - ϵ is a regular expression and $L(\epsilon) = \{\epsilon\}$
  - $\emptyset$ is a regular expression and $L(\emptyset) = \emptyset$
- **Induction Steps**
  - Union(denoted by $+$ or $|$) If E_1 and E_2 are two regular expressions, then $E_1 + E_2$ is a regular expression too and $L(E_1 + E_2) = L(E_1) \cup L(E_2)$
  - Concatenation If E_1 and E_2 are two regular expressions, then E_1E_2 is a regular expression and $L(E_1E_2) = L(E_1).L(E_2)$
  - Closure If E is a regular expressions, then E^* is a regular expression and $L(E^*) = L(E)^*$

Precedence of Operators

- Order of precedence is $*$ (highest), then concatenation, and finally $+(|)$ (lowest).
- Use parenthesis wherever needed.

RE Examples

Example

- The set of all even decimal numbers

$$(0|1|2.....|9)^*(0|2|4|6|8)$$

.

- The set of all strings over $\Sigma = \{a, b, c\}$ where each a is followed by at least one b:

$$(((c|b)^*ab)^*|(c|b)^*)^*$$

.

- Strings over $\Sigma = \{a, b\}$ where the number of a's is divisible by 3?

Regular Languages

Definition

A language ℓ defined over Σ is regular, if there is a regular expression E that can express ℓ, i.e. $L(E) = \ell$.

Example

Which one(s) of the following languages defined over $\{a, b\}$ are regular? Why?

- All strings over $\{a, b\}$ where the sum of a's and b's is odd.
- All hexadecimal even numbers.
- All strings of the form $a^n b^{2n}$

Table of Content

Algebraic Laws for Regular Expressions

- Union and concatenation behave kind of like addition and multiplication.
- Union is commutative and associative, i.e. $E_1 + E_2 = E_2 + E_1$ and $(E_1 + E_2) + E_3 = E_1 + (E_2 + E_3)$
- Concatenation is associative but not commutative, i.e. $E_1 E_2 \neq E_2 E_1$ and $(E_1 E_2) E_3 = E_1 (E_2 E_3)$.
- Concatenation distributes over $+$, i.e. $E_1(E_2 + E_3) = E_1 E_2 + E_1 E_3$.

Algebraic Laws for Regular Expressions

- $\emptyset$ is the identity for $+$, i.e.
  $E + \emptyset = E$.
- ϵ is the identity for concatenation, i.e.
  $\epsilon E = E\epsilon = E$ and
- $\emptyset$ is the annihilator for concatenation, i.e.
  $\emptyset R = R\emptyset = \emptyset$

Algebraic Laws for Regular Expressions

- $E^*E^* = E^*$
- $\emptyset^* = \epsilon$
- $\epsilon^* = \epsilon$
- $(E^*)^* = E^*$
- $EE^* = E^+$
- $(E_1 + E_2)^* = (E_1^* E_2^*)^*$

BonusPoint

Which one(s) is correct, if any? Prove it.

$\epsilon + E.E^* = (\epsilon + E)^*$

$(\epsilon + E)^* = E^*$

$(E^+ + E)^* = E^+$

Algebraic Laws for Regular Expressions

- $E + E = E$
- $\epsilon + E.E^* = E^*$
- $(\epsilon + E)^* = E^*$

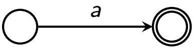
Table of Content

Equivalence of Regular Expressions and NFAs

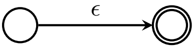
- We already showed that NFAs, ϵ-NFAs and DFAs are equivalent.
- RE and NFAs are equivalent. We should prove that
  - (i) For each regular expression E, there is an ϵ-NFA A where $L(A) = L(E)$, and
  - (ii) For each NFA A, there is a regular expression E where $L(A) = L(E)$

Regular Expression to ϵ-NFA

- To prove the existential goal (i), we first define a candidate procedure that we think produces the automaton with the language $L(E)$
- Then, we prove that the language of the produced automaton is $L(E)$ by induction on the number of operators
- That's a recursive procedure
- **Base Cases**
  - if the expression is the symbol a, then

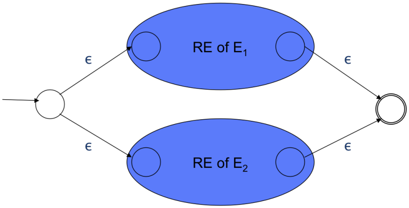


  - if the expression is ϵ, then



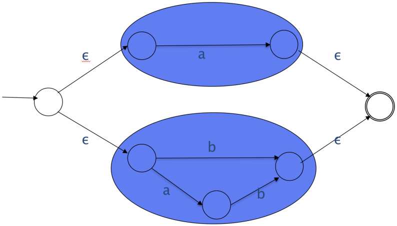
  - if the expression is $\emptyset$, then

Regular Expression to ϵ-NFA

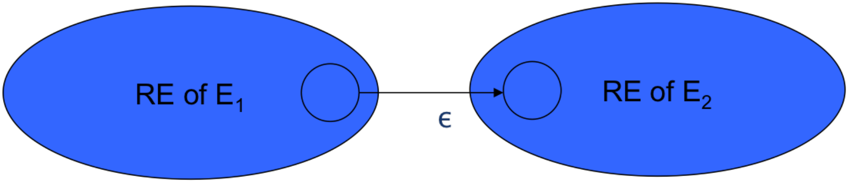


ϵ-NFA for $E_1 + E_2$

Example: The regular expression $a|(b|ab)$

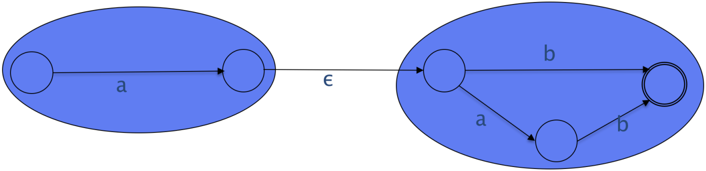


Regular Expression to ϵ-NFA

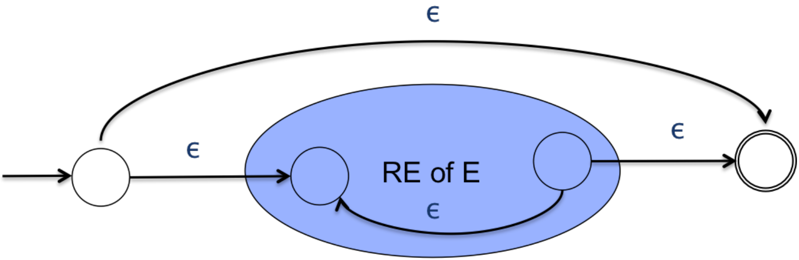


ϵ-NFA for $E_1 E_2$

Example: The regular expression $a(b|ab)$



Regular Expression to ϵ-NFA

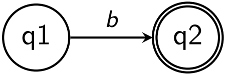


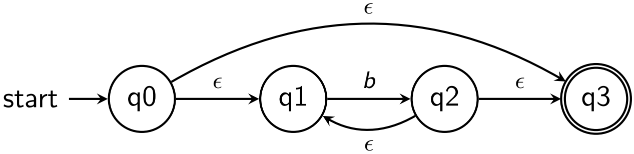
ϵ-NFA for E^*

Example

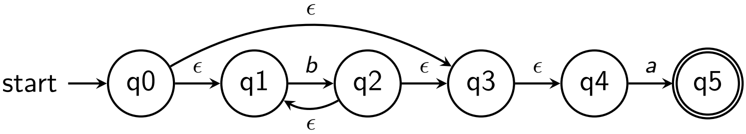
$(b^* a b^* a)^*$

Step 1- b





Step 2- b^*

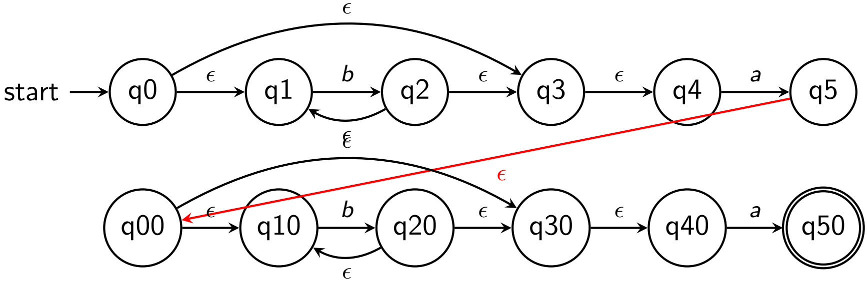


Step 3- $b^* a$

Example

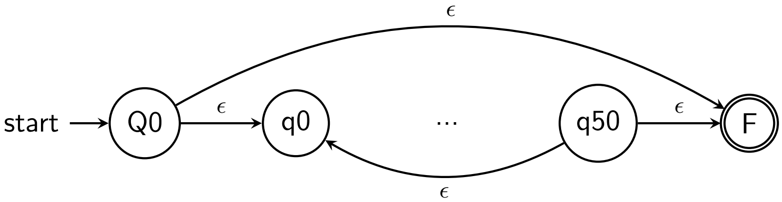
$(b^*ab^*a)^*$

Step 4- b^*ab^*a



Example

Step 5- $(b^*ab^*a)^*$



NFA to Regular Expressions

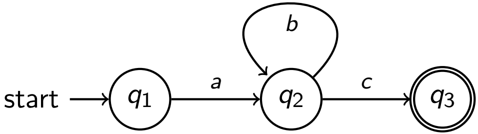
- Several methods: state-removal, transitive closure method and the Brzozowski algebraic method
- We focus on the state-removal method

Generalized NFA

- A Generalized NFA is an NFA whose labels are regular expressions
- An edge can be taken, if a prefix of the unread input matches the RE of the edge
- We transform an NFA to a GNFA that includes only initial and final states!

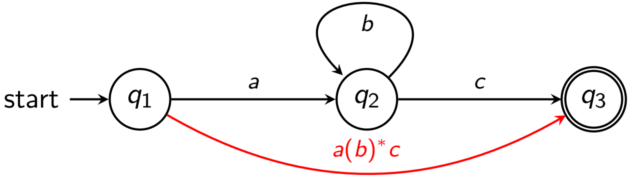
NFA to GNFA

Step 1- Choose an intermediate node (a node that is neither initial nor final state), .e.g q_2



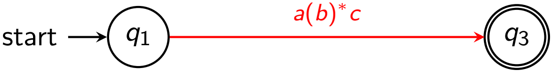
NFA to GNFA

Step 2- Change the transitions through q_2 into direct transitions, i.e. for each pair of transitions $\langle q_1, a, q_2\rangle$ and $\langle q_2, c, q_3\rangle$, add a new transition $\langle q_1, a(b)^*c, q_3\rangle$.



NFA to GNFA

- Step 3- Remove the node q_2

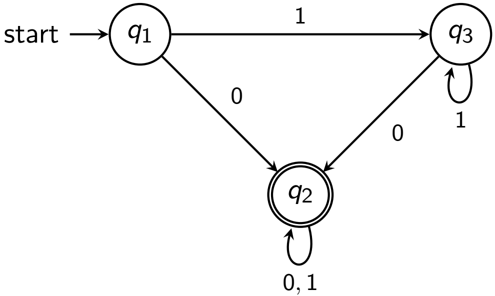


- Repeat Step 1-3 for all intermediate nodes

NFA to Regular Expressions

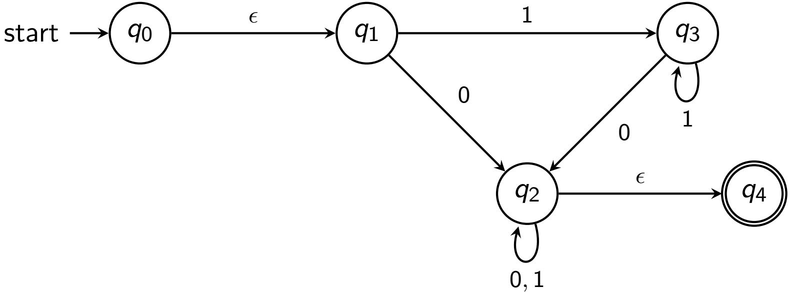
- Add a special initial state q_0' that is connected to the old initial state via an ϵ-transition
- Add a special final state q_f, such that all the final states are connected to q_f via and ϵ-transition
- Convert the NFA to a GNFA
- Output is the label on the (single) transition left in the GNFA.

Example

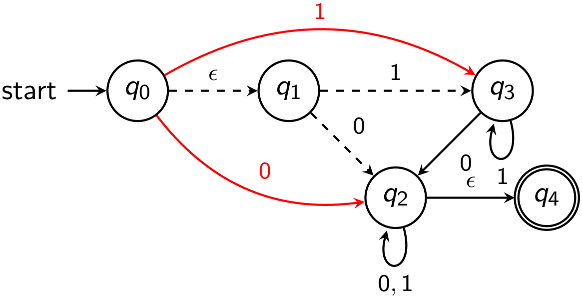


Example

Add initial and final states

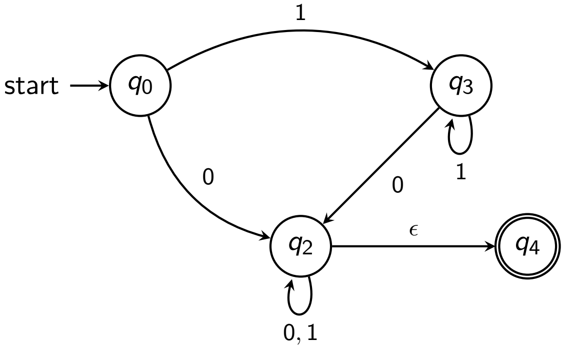


Example



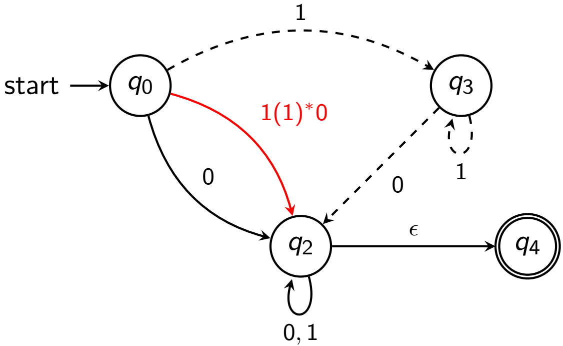
Remove q_1

Example



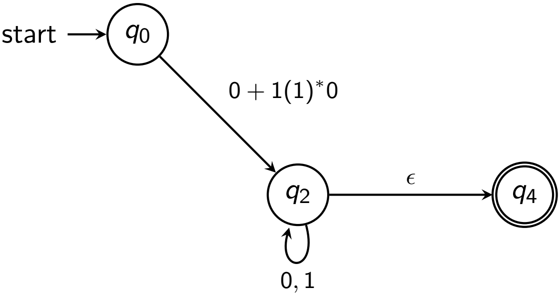
Remove q_1

Example



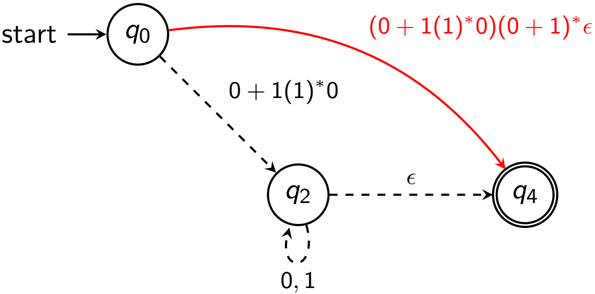
Remove q_3

Example



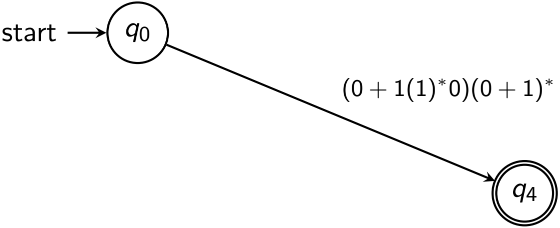
Remove q_3

Example



Remove q_2

Example



Remove q_2

The corresponding regular expression is $(0+1(1)^*0)(0+1)^*$.

Table of Content

Summary

- Languages Operations
- Regular expressions and their algebraic laws
- Equivalence of REs and NFAs
- Pumping Lemma
- Next lecture is on context-free grammars