Foundations of Computing

Lecture 5

Arkady Yerukhimovich

January 28, 2025

Outline

Lecture 4 Review

- More NFAs
- Equivalence of NFAs and DFAs
- NFAs for union, composition, and star – closure of regular languages

Outline

A Useful Corollary

Recall that:

Definition

A language L is regular if and only if there is a DFA that recognizes it

A Useful Corollary

Recall that:

Definition

A language L is regular if and only if there is a DFA that recognizes it

Since we now know that NFAs and DFAs are equal:

Corollary

A language L is regular if and only if there is an NFA that recognizes it

A Useful Corollary

Recall that:

Definition

A language L is regular if and only if there is a DFA that recognizes it

Since we now know that NFAs and DFAs are equal:

Corollary

A language L is regular if and only if there is an NFA that recognizes it

We can now use NFAs to argue the properties of regular languages

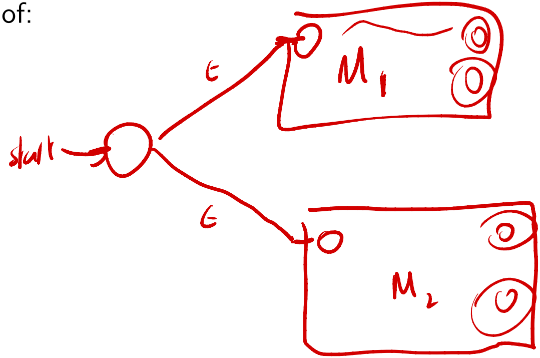
Closure Under Union

Closure Under Union

If L_1 and L_2 are both regular languages then $L_1 \cup L_2$ is also regular

$L_1 \cup L_2$ is the language consisting of all strings either in L_1 or L_2

Proof:



Closure Under Concatenation

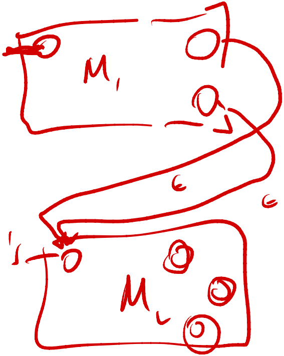
Closure Under Concatenation

If L_1 and L_2 are both regular languages then $L_1 \circ L_2$ is also regular

$L_1 \circ L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$

Proof:

00101



$x = y \in L_1 \,||\, z \in L_2$

L_1 = strings of all 0's

L_2 = strings 01 01 ..

Closure Under the Star Operation

Closure Under Star Operation

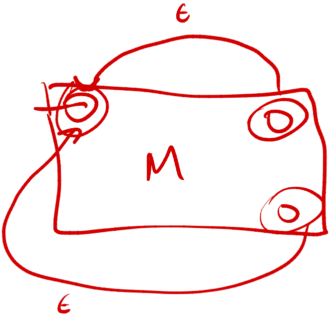
If L is a regular languages then L^* is also regular

$L^* = \{0 \text{ or more strings from } L\}$

0^*

Proof:

$\{0, 1\}^*$



$L = \{01, 10\}$

$L^* = \{\epsilon, 01, 10, 0101,$

0110

1001

Outline

Regular Expressions

- Strings that describe a language

Regular Expressions

- Strings that describe a language
- They consist of:
  - Symbols (e.g., 0,1)
  - Parentheses
  - $\cup$ - representing union
  - $*$ - representing repetition 0 or more times

Regular Expressions

- Strings that describe a language
- They consist of:
    - Symbols (e.g., 0,1)
    - Parentheses
    - $\cup$ - representing union
    - $*$ - representing repetition 0 or more times
- Examples:
    - $0^*10^* = \{w|w \text{ has exactly one } 1\}$

Regular Expressions

- Strings that describe a language
- They consist of:
  - Symbols (e.g., 0,1)
  - Parentheses
  - $\cup$ - representing union
  - $*$ - representing repetition 0 or more times
- Examples:
  - $0^*10^* = \{w|w \text{ has exactly one } 1\}$
  - $01 \cup 10 = \{01, 10\}$

Regular Expressions

- Strings that describe a language
- They consist of:
  - Symbols (e.g., 0,1)
  - Parentheses
  - $\cup$ - representing union
  - $*$ - representing repetition 0 or more times
- Examples:
  - $0^*10^* = \{w|w \text{ has exactly one } 1\}$
  - $01 \cup 10 = \{01, 10\}$
  - $\Sigma^*1\Sigma^* = \{w|w \text{ has at least one } 1\}$

Regular Expressions

- Strings that describe a language
- They consist of:
  - Symbols (e.g., 0,1)
  - Parentheses
  - $\cup$ - representing union
  - $*$ - representing repetition 0 or more times
- Examples:
  - $0^*10^* = \{w|w \text{ has exactly one } 1\}$
  - $01 \cup 10 = \{01, 10\}$
  - $\Sigma^*1\Sigma^* = \{w|w \text{ has at least one } 1\}$

You've seen this before

Regular expressions very useful in compilers, and string search (e.g., grep)

Formal Definition

R is a regular expression if R is

1. a for some a in the alphabet Σ (or Σ)

Formal Definition

R is a regular expression if R is

1. a for some a in the alphabet Σ (or Σ)
2. ϵ – the empty string

Formal Definition

R is a regular expression if R is

1. a for some a in the alphabet Σ (or Σ)
2. ϵ – the empty string
3. $\emptyset$ – the empty set

Formal Definition

R is a regular expression if R is

1. a for some a in the alphabet Σ (or Σ)
2. ϵ – the empty string
3. $\emptyset$ – the empty set
4. $(R_1 \cup R_2)$ – R_1 or R_2 where R_1 and R_2 are regular expressions

Formal Definition

R is a regular expression if R is

1. a for some a in the alphabet Σ (or Σ)
2. ϵ – the empty string
3. $\emptyset$ – the empty set
4. $(R_1 \cup R_2)$ – R_1 or R_2 where R_1 and R_2 are regular expressions
5. $(R_1 \circ R_2)$ – R_1 concatenated with R_2 where R_1 and R_2 are regular expressions

Formal Definition

R is a regular expression if R is

1. a for some a in the alphabet Σ (or Σ)
2. ϵ – the empty string
3. $\emptyset$ – the empty set
4. $(R_1 \cup R_2)$ – R_1 or R_2 where R_1 and R_2 are regular expressions
5. $(R_1 \circ R_2)$ – R_1 concatenated with R_2 where R_1 and R_2 are regular expressions
6. (R_1^*) – 0 or more repetitions of R_1 where R_1 is a regular expression

Some More Examples

- $(\Sigma\Sigma)^* =$

Some More Examples

- $(\Sigma\Sigma)^* = \{w | w \text{ is a string of even length}\}$
- $(0 \cup \epsilon)(1 \cup \epsilon) =$

Some More Examples

- $(\Sigma\Sigma)^* = \{w | w \text{ is a string of even length}\}$
- $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$
- $1^*\emptyset =$

Some More Examples

- $(\Sigma\Sigma)^* = \{w | w \text{ is a string of even length}\}$
- $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$
- $1^*\emptyset = \emptyset$ $(\{\epsilon, 1, 11, 111, \ldots\})\ \emptyset$
- $\emptyset^* =$

Some More Examples

- $(\Sigma\Sigma)^* = \{w | w \text{ is a string of even length}\}$
- $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$
- $1^*\emptyset = \emptyset$
- $\emptyset^* = \{\epsilon\}$
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$

Some More Examples

$1^* \epsilon = 1^*$

- $(\Sigma\Sigma)^* = \{w | w \text{ is a string of even length}\}$
- $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$
- $1^*\emptyset = \emptyset$
- $\emptyset^* = \{\epsilon\}$
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w | w \text{ starts and ends with the same symbol}\}$

$\phi = \{\}$ $\{\epsilon\}$

Languages to Regular Expressions Examples

Consider languages over the alphabet $\{0, 1, 2\}$

1. $L_1 = \{w | w \text{ has 2 consecutive 0's}\}$

$$\Sigma^* 00 \Sigma^*$$

Languages to Regular Expressions Examples

Consider languages over the alphabet $\{0, 1, 2\}$

1. $L_1 = \{w | w \text{ has 2 consecutive 0's}\}$

2. $L_2 = \{w | w \text{ has a substring 101 and ends in 22}\}$

$\Sigma^* 101 \Sigma^* 22$

Languages to Regular Expressions Examples

Consider languages over the alphabet $\{0, 1, 2\}$

1. $L_1 = \{w | w \text{ has 2 consecutive 0's}\}$

2. $L_2 = \{w | w \text{ has a substring 101 and ends in 22}\}$

3. $L_3 = \{w | w \in L_1 \text{ or } w \in L_2\}$

$R_1 \cup R_2$

Languages to Regular Expressions Examples

Consider languages over the alphabet $\{0, 1, 2\}$

1. $L_1 = \{w | w \text{ has 2 consecutive 0's}\}$

2. $L_2 = \{w | w \text{ has a substring 101 and ends in 22}\}$

3. $L_3 = \{w | w \in L_1 \text{ or } w \in L_2\}$

Question:

What does this have to do with FAs and regular languages?

Outline

Regular Expressions == Regular Languages == NFA

Theorem

A language L is regular if and only if some regular expression describes it.

Regular Expressions == Regular Languages == NFA

Theorem

A language L is regular if and only if some regular expression describes it.

Proof (Part 1): If L is described a regular expression then it is regular.
Enough to show how to construct NFA to recognize L

Regular Expressions == Regular Languages == NFA

Theorem

A language L is regular if and only if some regular expression describes it.

Proof (Part 1): If L is described a regular expression then it is regular.
Enough to show how to construct NFA to recognize L

1. $R = a$ for some $a \in \Sigma$

2. $R = \epsilon$

3. $R = \emptyset$

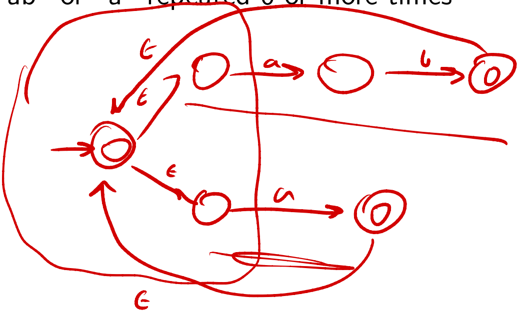
4. $R = R_1 \cup R_2$
5. $R = R_1 \circ R_2$
6. $R = R_1^*$

An Example

Problem: Convert $(ab \cup a)^*$ to an NFA

In English: Either "ab" or "a" repeated 0 or more times

- a:
- b:
- ab:
- $ab \cup a$:
- $(ab \cup a)^*$:



Regular Expressions == Regular Languages

Theorem

A language L is regular if and only if some regular expression describes it.

Regular Expressions == Regular Languages

Theorem

A language L is regular if and only if some regular expression describes it.

Proof (Part 2): If L is regular then it can be described by a regular expression.

Enough to show how to build regular expression corresponding to a NFA

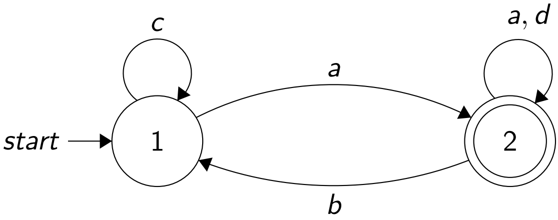
Regular Expressions == Regular Languages

Theorem

A language L is regular if and only if some regular expression describes it.

Proof (Part 2): If L is regular then it can be described by a regular expression.

Enough to show how to build regular expression corresponding to a NFA



Question

How do we represent L by a regular expression?

Step 1: NFA $\rightarrow$ generalized NFA

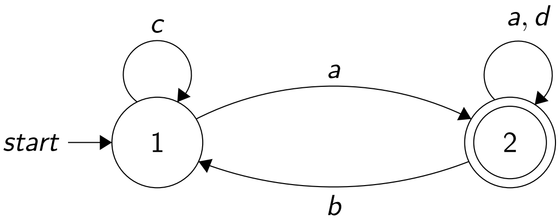
A generalized NFA has 3 important properties:

1. Start state has no incoming edges
2. Only one accept state, and it has no outgoing edges
3. Edges labeled by regular expressions

Step 1: NFA → generalized NFA

A generalized NFA has 3 important properties:

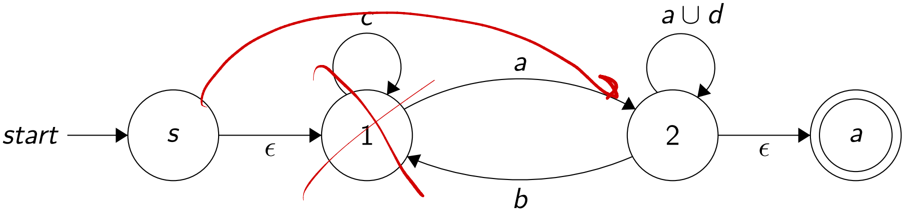
1. Start state has no incoming edges
2. Only one accept state, and it has no outgoing edges
3. Edges labeled by regular expressions



Step 2: Node Elimination – Remove Node 1

Remove nodes one-by-one (in any order) until only start and accept states left:

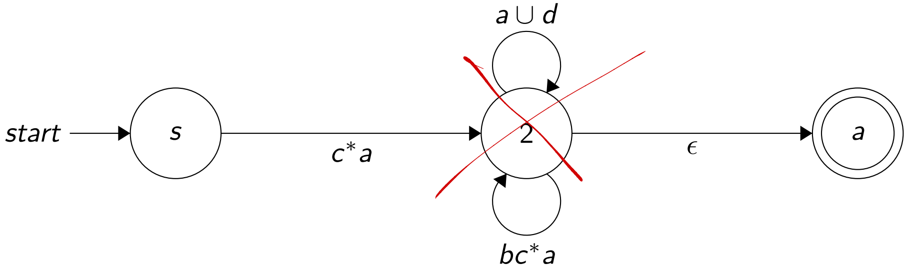
- Need to update reg. exp.'s for all paths through removed nodes



Step 2: Node Elimination – Remove Node 2

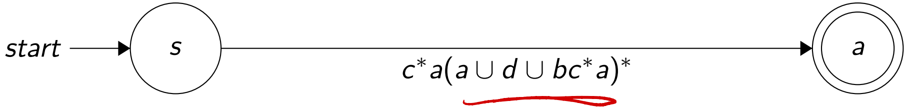
Remove nodes one-by-one (in any order) until only start and accept states left:

- Need to update reg. exp.'s for all paths through removed nodes

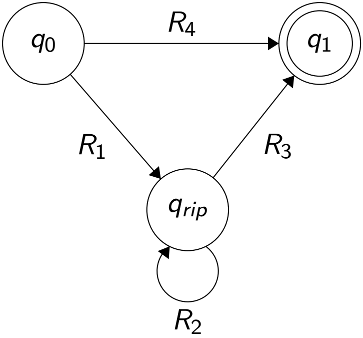


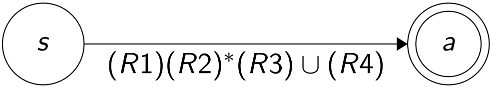
We are Done

Output label of final edge from start to accept state.



Generalized Node Elimination





Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Base Case: For $|G| = 2$, G consists of start and accept states and arrow between them. The label on this arrow exactly describes the language of strings accepted by G.

Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Inductive step: Assume true for $|G| = k - 1$, prove true for $|G| = k$. (i.e., prove that $G' = G$)

- Assume some w s.t. $G(w) = 1$, then on input w, G goes through

$$q_{start}, q_1, q_2, \ldots, q_{accept}$$

Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Inductive step: Assume true for $|G| = k - 1$, prove true for $|G| = k$. (i.e., prove that $G' = G$)

- Assume some w s.t. $G(w) = 1$, then on input w, G goes through

$$q_{start}, q_1, q_2, \ldots, q_{accept}$$

  - If q_{rip} is not on this path, clearly $G'(w) = 1$

Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Inductive step: Assume true for $|G| = k - 1$, prove true for $|G| = k$. (i.e., prove that $G' = G$)

- Assume some w s.t. $G(w) = 1$, then on input w, G goes through

$$q_{start}, q_1, q_2, \dots, q_{accept}$$

  - If q_{rip} is not on this path, clearly $G'(w) = 1$
  - If q_{rip} is on this path, then the q_i and q_j nodes before and after q_{rip} have a new reg. exp. in G' describing all paths through q_{rip}

Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Inductive step: Assume true for $|G| = k - 1$, prove true for $|G| = k$. (i.e., prove that $G' = G$)

- Assume some w s.t. $G(w) = 1$, then on input w, G goes through

$$q_{start}, q_1, q_2, \ldots, q_{accept}$$

  - If q_{rip} is not on this path, clearly $G'(w) = 1$
  - If q_{rip} is on this path, then the q_i and q_j nodes before and after q_{rip} have a new reg. exp. in G' describing all paths through q_{rip}
- Assume some w s.t. $G'(w) = 1$, then $G'(w)$ stops in q_{accept}.

Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Inductive step: Assume true for $|G| = k - 1$, prove true for $|G| = k$. (i.e., prove that $G' = G$)

- Assume some w s.t. $G(w) = 1$, then on input w, G goes through

$$q_{start}, q_1, q_2, \ldots, q_{accept}$$

  - If q_{rip} is not on this path, clearly $G'(w) = 1$
  - If q_{rip} is on this path, then the q_i and q_j nodes before and after q_{rip} have a new reg. exp. in G' describing all paths through q_{rip}
- Assume some w s.t. $G'(w) = 1$, then $G'(w)$ stops in q_{accept}.
  - If it would have gone through q_{rip} then the modified edge accepts w, so there is a path through q_{rip} in G that accepts w.

Proof

Theorem

For any GNFA G, G'=NODE-ELIMINATE(G) is equivalent to G

Inductive step: Assume true for $|G| = k - 1$, prove true for $|G| = k$. (i.e., prove that $G' = G$)

- Assume some w s.t. $G(w) = 1$, then on input w, G goes through

$$q_{start}, q_1, q_2, \ldots, q_{accept}$$

  - If q_{rip} is not on this path, clearly $G'(w) = 1$
  - If q_{rip} is on this path, then the q_i and q_j nodes before and after q_{rip} have a new reg. exp. in G' describing all paths through q_{rip}
- Assume some w s.t. $G'(w) = 1$, then $G'(w)$ stops in q_{accept}.
  - If it would have gone through q_{rip} then the modified edge accepts w, so there is a path through q_{rip} in G that accepts w.
  - If the accepting path would have not gone through q_{rip}, then G must also have the same path to accept w

Outline

Properties of Regular Expressions

By equivalence between regular expressions and NFAs, we have:

1. Regular expressions are closed under complement
2. Regular expressions are closed under union
3. Regular expressions are closed under star
4. ...

Properties of Regular Expressions

By equivalence between regular expressions and NFAs, we have:

1. Regular expressions are closed under complement
2. Regular expressions are closed under union
3. Regular expressions are closed under star
4. ...

Proof:

Properties of Regular Expressions

By equivalence between regular expressions and NFAs, we have:

1. Regular expressions are closed under complement
2. Regular expressions are closed under union
3. Regular expressions are closed under star
4. ...

Proof:

- Build NFA M corresponding to each clause

Properties of Regular Expressions

By equivalence between regular expressions and NFAs, we have:

1. Regular expressions are closed under complement
2. Regular expressions are closed under union
3. Regular expressions are closed under star
4. ...

Proof:

- Build NFA M corresponding to each clause
- Since we already showed how to build NFA to show closure, can convert that to regular expression to prove the claim.