Theory of computation

Lecture 3: Automata and Regular languages

Adnane Saoud

Thursday, September 19, 2024

Summary of lecture 2

Lecture 2

- Nondeterministic finite automatas (NFAs)
- Closure under $\circ$ and $*$
- How to convert regular expressions to finite automata

Lecture 3

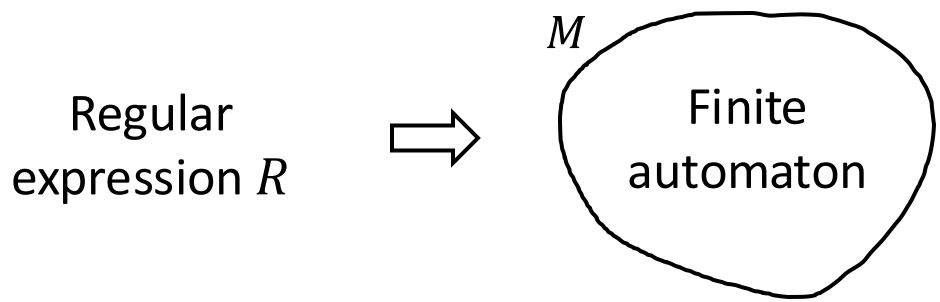
- From finite automata to regular expressions
- Proving some languages are not regular
- Context free grammars

From finite automata to regular expressions

DFAs → Regular Expressions

Recall Theorem: If a language is described by a regular expression then it is regular

Proof: Conversion $R \rightarrow$ NFA $M \rightarrow$ DFA M'



Today's Theorem: If a language A is regular then it can described as regular expression

Proof: Give conversion DFA $M \rightarrow R$

We need a new concept: GNFA.

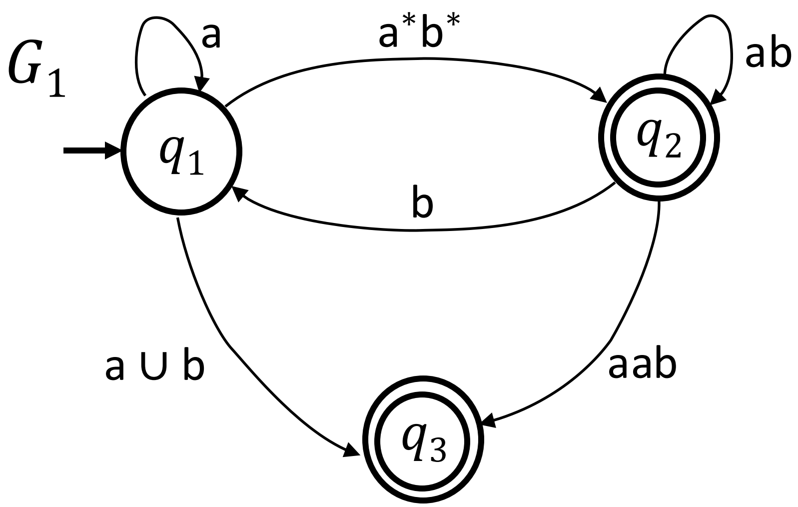
- From DFA to GNFA
- From GNFA to Regular expressions

Generalized NFAs

Definition: A Generalized Nondeterministic Finite Automaton (GNFA) is similar to an NFA, but allows regular expressions as transition labels

We first convert the GNFA to have this special form:

- The start state has transition arrows going to every other state but no arrows coming in from any other state.
- There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
- Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.



Generalized NFAs

Definition: A Generalized Nondeterministic Finite Automaton (GNFA) is similar to an NFA, but allows regular expressions as transition labels

How to convert the GNFA into the special form:

- The start state has transition arrows going to every other state but no arrows coming in from any other state.

➢ Add a new start state with an ε arrow to the old start state

- There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.

➢ Add a new accept state with ε arrows from the old accept states.

- Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

➢ If there are multiple arrows going between the same two states in the same direction, we replace each with a single arrow whose label is the union of the previous labels.

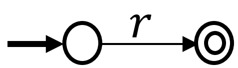
➢ We add arrows labeled ∅ between states that had no arrows

GNFA → Regular Expressions

Lemma: Every GNFA G has an equivalent regular expression R

Proof: By induction on the number of states k of G

<u>*Basis*</u> $(k = 2)$:

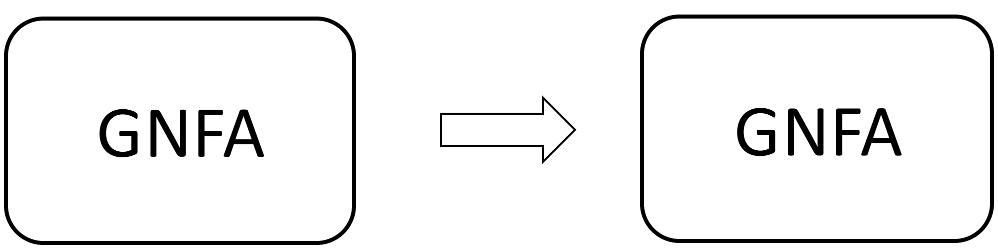
$G =$ 

G is in special form

Let $R = r$

Induction step $(k > 2)$: Assume Lemma true for $k - 1$ states and prove for k states

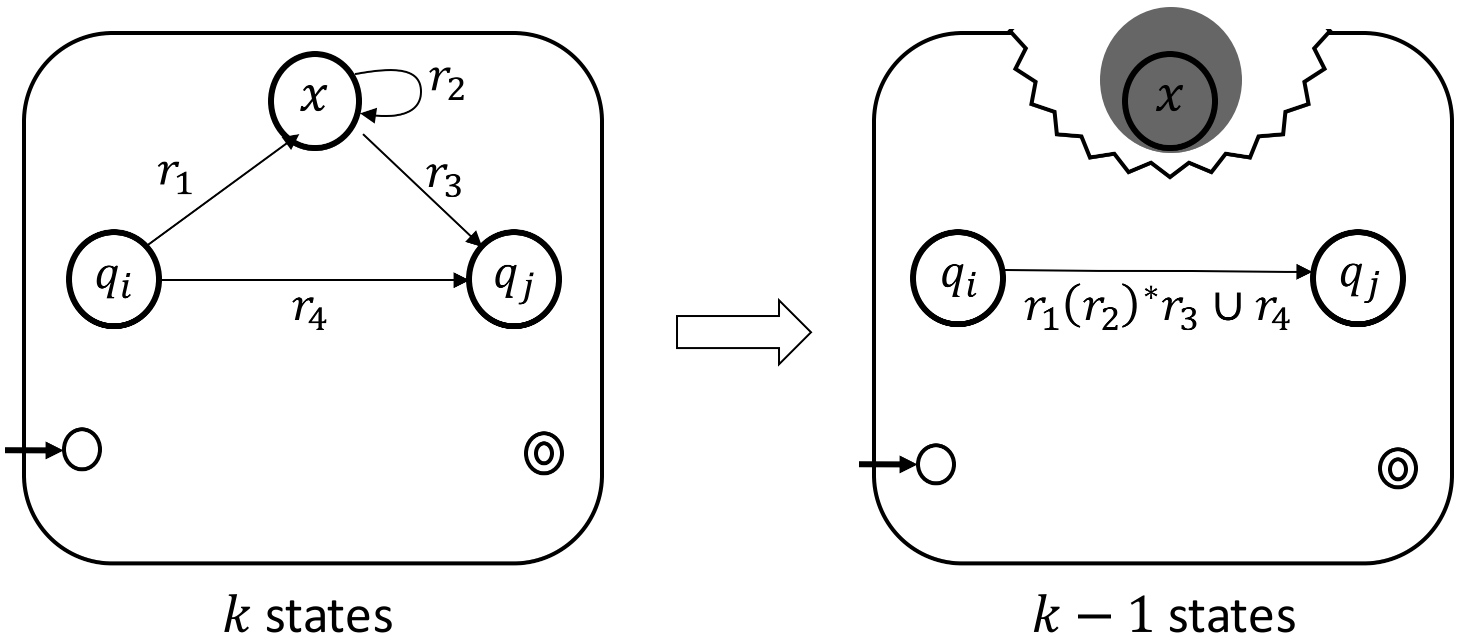
IDEA: Convert k-state GNFA to equivalent $(k - 1)$ -state GNFA



k states

$k - 1$ states

k-state GNFA $\rightarrow$ (k—1)-state GNFA



1. Pick any state x except the start and accept states.
2. Remove x.
3. Repair the damage by recovering all paths that went through x.
4. Make the indicated change for each pair of states q_i, q_j.

Pumping Lemma

Non-Regular Languages

How do we show a language is not regular?

- To show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

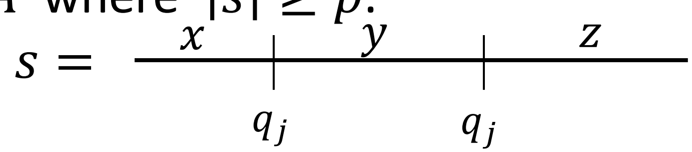
Method for Proving Non-regularity

Pumping Lemma: For every regular language A, there is a number p (the "pumping length") such that for any string $s \in A$ satisfying $|s| \geq p$ there exist x, y and z such that $s = xyz$ where

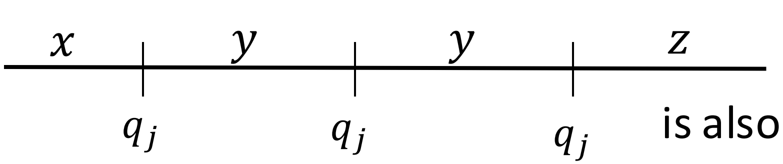
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

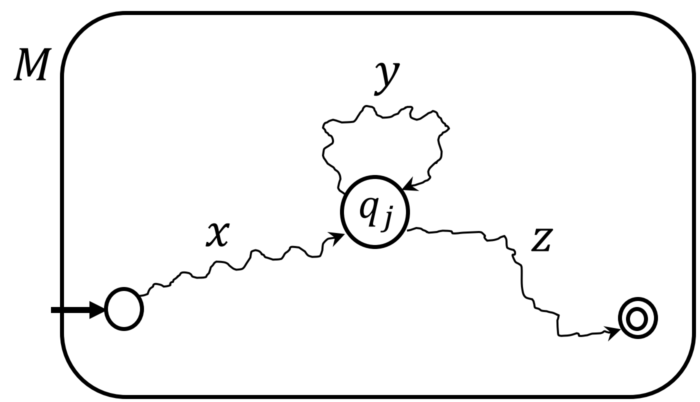
Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.



M will repeat a state q_j when reading s because s is so long.



is also accepted



The path that M follows when reading s.

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that for any string $s \in A$ satisfying $|s| \geq p$ there exist x, y and z such that $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Let $D = \{0^k1^k \mid k \geq 0\}$

Show: D is not regular

$$s = \underbrace{000\cdots000111\cdots111}_{x \,|\, y \,|\, z}$$

$$\leftarrow \leq p \rightarrow$$

Proof by Contradiction:

Assume (to get a contradiction) that D <u>is</u> regular.

The pumping lemma gives p as above. Let $s = 0^p1^p \in D$.

Pumping lemma says that can divide $s = xyz$ satisfying the 3 conditions.

But $xyyz$ has excess 0s and thus $xyyz \notin D$ contradicting the pumping lemma.

Therefore our assumption (D is regular) is false. We conclude that D is not regular.

Example 2 of Proving Non-regularity

Let $F = \{ww \mid w \in \Sigma^*\}$. Say $\Sigma^* = \{0,1\}$.

Show: F is not regular

Proof by Contradiction:
Assume (for contradiction) that F is regular.
The pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s = 0^p0^p \in F$. But that s can be pumped and stay inside F. Bad choice.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.
$xyyz \notin F$ Contradiction! Therefore F is not regular.

$$s = 000\cdots001000\cdots001$$

$x \mid y \mid z$

$\leftarrow \le p \rightarrow$

Example 3 of Proving Non-regularity

Let $B = \{w |\ w \text{ has equal numbers of 0s and 1s}\}$

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^*1^* is regular so $B \cap 0^*1^*$ is regular (closure under intersection).

But $D = B \cap 0^*1^*$ and we already showed D is not regular. Contradiction!

Therefore our assumption is false, so B is not regular.

Variant: Combine closure properties with the Pumping Lemma.

Context-free Languages

Context Free Grammars

G_1

$$S \rightarrow 0S1$$
$$S \rightarrow R$$
$$R \rightarrow \varepsilon$$

(Substitution) Rules

In G_1:

Rule:	Variable → string of variables and terminals	3 rules
Variables:	Symbols appearing on left-hand side of rule	R,S
Terminals:	Symbols appearing only on right-hand side	0,1
Start Variable:	Top left symbol	S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

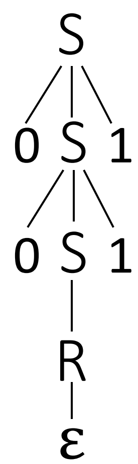
Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\}$$ (Substitution) Rules

Example of G_1 generating a string

Tree of substitutions



Resulting string

S

0S1

00S11

00R11

$0011 \in L(G_1)$

$$L(G_1) = \{0^k 1^k \mid k \geq 0\}$$

Summary

Summary

1. DFAs, NFAs, regular expressions are all equivalent

2. Proving languages not regular by using the pumping lemma and closure properties

3. Context Free Grammars