Pre-lecture brain teaser

Assume L is any regular language. Let's define a new language:

Definition
$\text{Flip}(L) = \{\bar{w} \mid w \in L, x \in \Sigma^*\}$

CS/ECE-374: Lecture 6 - Regular Languages - Closure Properties

Lecturer: Nickvash Kani
Chat moderator: Samir Khan

February 11, 2021

University of Illinois at Urbana-Champaign

Pre-lecture brain teaser

Assume L is any regular language. Let's define a new language:

Definition
$\text{Flip}(L) = \{\bar{w} \mid w \in L, x \in \Sigma^*\}$

Pre-lecture brain teaser

Assume L is any regular language. Let's define a new language:

Definition
$\text{Flip}(L) = \{\bar{w} \mid w \in L, x \in \Sigma^*\}$

Yes

Pre-lecture brain teaser

Assume L is any regular language. Let's define a new language:

Definition
$\text{Flip}(L) = \{\bar{w} \mid w \in L, x \in \Sigma^*\}$

Yes Next problem.

Pre-lecture brain teaser

Assume L is any regular language. Let's define a new language:

Definition
$L^R = \{w^R \mid w \in L\}$

Pre-lecture brain teaser

Assume L is any regular language. Let's define a new language:

Definition
$L^R = \{w^R \mid w \in L\}$

Also yes.

Closure propeties

Definition
(Informal) A set A is **closed** under an operation **op** if applying **op** to any elements of A results in an element that also belongs to A.

Closure propeties

Definition
(Informal) A set A is **closed** under an operation **op** if applying **op** to any elements of A results in an element that also belongs to A.

Examples:

- *Integers:* closed under $+$, $-$, $*$, but not division.
- *Positive integers:* closed under $+$ but not under $-$
- *Regular languages:* closed under union, intersection, Kleene star, complement, difference, homomorphism, inverse homomorphism, reverse, $\ldots$

Closure properties of Regular Languages

How do we prove that regular languages are closed under some new operation?

Closure properties of Regular Languages

How do we prove that regular languages are closed under some new operation?

Three broad approaches

- Use existing closure properties

Closure properties of Regular Languages

How do we prove that regular languages are closed under some new operation?

Three broad approaches

- Use existing closure properties
  - L_1, L_2, L_3, L_4 regular implies $(L_1 - L_2) \cap (\overline{L_3} \cup L_4)^*$ is regular

Closure properties of Regular Languages

How do we prove that regular languages are closed under some new operation?

Three broad approaches

- Use existing closure properties
    - L_1, L_2, L_3, L_4 regular implies $(L_1 - L_2) \cap (\overline{L_3} \cup L_4)^*$ is regular
- Transform regular expressions

Closure properties of Regular Languages

How do we prove that regular languages are closed under some new operation?

Three broad approaches

- Use existing closure properties
  - L_1, L_2, L_3, L_4 regular implies $(L_1 - L_2) \cap (\overline{L_3} \cup L_4)^*$ is regular
- Transform regular expressions
- Transform DFAs to NFAs — versatile technique and shows the power of nondeterminism

Homomorphism closure

Let's look back at the pre-lecture teaser. Define a function

$$h(x) = \begin{cases} 1 & x = 0 \\ 0 & x = 1 \end{cases}$$

This is known as a homomorphism - A cipher that is a one-to-one mapping to one character set to another.

How do we prove $h(L)$ is regular if L is regular?

Homomorphism closure

Proof Idea:

1. Suppose R is a regular expression for L.
2. We define $Flip(L) = L^F$ as a regular expression based off the regular expression for L (using a finite number of concatenations, unions and Kleene Star)
3. Thus L^F is regular because it has a regular expression.

Thus we reduce the argument to $L(h(R)) = h(L(R))$

Homomorphism closure

Let's define the regular expression inductively by transforming the operations in R. We see that:

- **Base Case:** Zero operators in R means that $R =: a \in \Sigma$, ε, $\emptyset$. In any case we define $R^F = h(R)$
- Otherwise R has three potential types of operators to transform. Splitting R at an operator we see:
  - $h(R_1 R_2) = h(R_1) \cdot h(R_2)$
  - $h(R_1 \cup R_2) = h(R_1) \cup h(R_2)$
  - $h(R^*) = (h(R))^*$

Hence, since we can define L^F via a regular language, L^F is regular.

Regular Languages

Regular languages have three different characterizations

- Inductive definition via base cases and closure under union, concatenation and Kleene star
- Languages accepted by DFAs
- Languages accepted by NFAs

Regular Languages

Regular languages have three different characterizations

- Inductive definition via base cases and closure under union, concatenation and Kleene star
- Languages accepted by DFAs
- Languages accepted by NFAs

Regular language closed under many operations:

- union, concatenation, Kleene star via inductive definition or NFAs
- complement, union, intersection via DFAs
- homomorphism, inverse homomorphism, reverse, ...

Different representations allow for flexibility in proofs.

Closure problem - Reverse

Example: REVERSE

Given string w, w^R is reverse of w.

For a language L define $L^R = \{w^R \mid w \in L\}$ as reverse of L.

Theorem
L^R is regular if L is regular.

Example: REVERSE

Given string w, w^R is reverse of w.

For a language L define $L^R = \{w^R \mid w \in L\}$ as reverse of L.

Theorem
L^R is regular if L is regular.

Infinitely many regular languages!

Proof technique:

- take some finite representation of L such as regular expression r
- Describe an algorithm A that takes r as input and outputs a regular expression r' such that $L(r') = (L(r))^R$.
- Come up with A and prove its correctness.

REVERSE via regular expressions

Suppose r is a regular expression for L. How do we create a regular expression r' for L^R?

REVERSE via regular expressions

Suppose r is a regular expression for L. How do we create a regular expression r' for L^R? Inductively based on recursive definition of r.

- $r = \emptyset$ or $r = a$ for some $a \in \Sigma$
- $r = r_1 + r_2$
- $r = r_1 \cdot r_2$
- $r = (r_1)^*$

REVERSE via regular expressions

- $r = \emptyset$ or $r = a$ for some $a \in \Sigma$
  $r' =$
- $r = r_1 + r_2$.
  If r_1', r_2' are reg expressions for $(L(r_1))^R, (L(r_2))^R$ then
  $r' =$
- $r = r_1 \cdot r_2$.
  If r_1', r_2' are reg expressions for $(L(r_1))^R, (L(r_2))^R$ then
  $r' =$
- $r = (r_1)^*$.
  If r_1' is reg expressions for $(L(r_1))^R$ then
  $r' =$

$r = (0 + 10)^*(001 + 01)1$ then $r' =$

REVERSE via machine transformation

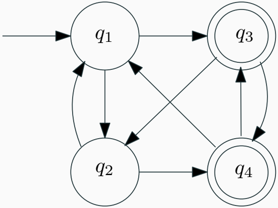
Given DFA $M = (Q, \Sigma, \delta, s, A)$ want NFA N such that $L(N) = (L(M))^R$.

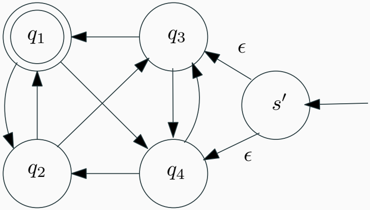
N should accept w^R **iff** M accepts w

M accepts w iff $\delta^*_M(s, w) \in A$

Idea:

REVERSE via machine transformation





Caveat: Reversing transitions may create an NFA.

REVERSE via machine transformation

Proof (DFA to NFA): Let $M = (\Sigma, Q, s, A, \delta)$ be an arbitrary DFA that accepts L. We construct an NFA $M^R = (\Sigma, Q^R, s^R, A^R, \delta^R)$ with ε-transitions that accepts L^R, intuitively by reversing every transition in M, and swapping the roles of the start state and the accepting states. Because M does not have a unique accepting state, we need to introduce a special start state s^R, with ε-transitions to each accepting state in M. These are the only ε-transitions in M^R.

$$\begin{aligned} Q^R &= Q \cup \{s^R\} \\ A^R &= \{s\} \\ \delta^R(s^R, \varepsilon) &= A \\ \delta^R(s^R, a) &= \varnothing && \text{for all } a \in \Sigma \\ \delta^R(q, \varepsilon) &= \varnothing && \text{for all } q \in Q \\ \delta^R(q, a) &= \left\{ p \;\middle|\; q \in \delta(p, a) \right\} && \text{for all } q \in Q \text{ and } a \in \Sigma \end{aligned}$$

Routine inductive definition-chasing now implies that the reversal of any sequence $q_0 \to q_1 \to \cdots \to q_\ell$ of transitions in M is a valid sequence $q_\ell \to q_{\ell-1} \to \cdots \to q_0$ of transitions in M^R. Because the transitions retain their labels (but reverse directions), it follows that M accepts any string w if and only if M^R accepts w^R.

We conclude that the NFA M^R accepts L^R, so L^R must be regular. $\square$

REVERSE via machine transformation

Formal proof: two directions

- $w \in L(M)$ implies $w^R \in L(N)$. Sketch. Let $\delta^*_M(s, w) = q$ where $q \in A$. On input w^R N non-deterministically transitions from its start state s' to q on an ϵ transition, and traces the reverse of the walk of M on w^R and hence reaches s which is an accepting state of N. Thus N accepts w^R

- $u \in L(N)$ implies $u^R \in L(M)$. Sketch. If $u \in N$ it implies that s' transitioned to some $q \in A$ on ϵ transition and

Closure Problem - Cycle

A more complicated example: CYCLE

$CYCLE(L) = \{yx \mid x, y \in \Sigma^*, xy \in L\}$

Theorem
$CYCLE(L)$ is regular if L is regular.

Example: $L = \{abc, 374a\}$

$CYCLE(L) =$

A more complicated example: CYCLE

$$CYCLE(L) = \{yx \mid x, y \in \Sigma^*, xy \in L\}$$

Theorem
$CYCLE(L)$ is regular if L is regular.

A more complicated example: CYCLE

$CYCLE(L) = \{yx \mid x, y \in \Sigma^*, xy \in L\}$

Theorem
$CYCLE(L)$ is regular if L is regular.

Given DFA M for L create NFA N that accepts $CYCLE(L)$.

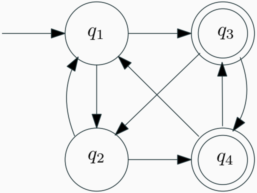
- N is a finite state machine, cannot know split of w into xy and yet has to simulate M on x and y.
- Exploit fact that M is itself a finite state machine. N only needs to “know” the state $\delta_M^*(s, x)$ and there are only finite number of states in M

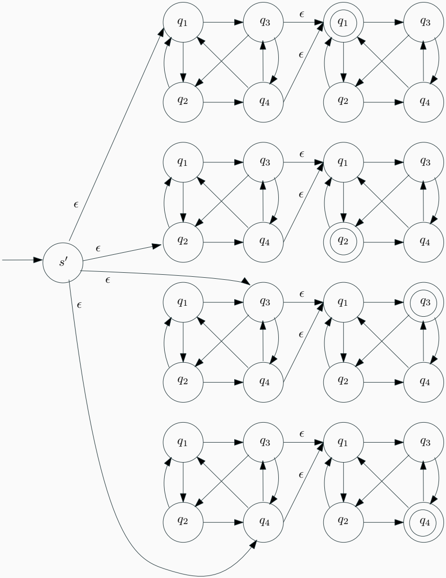
Construction for CYCLE

Let $w = xy$ and $w' = yx$.

- *N guesses* state $q = \delta_M^*(s, x)$ and simulates M on w' with start state q.
- *N guesses* when y ends (at that point M must be in an accept state) and transitions to a copy of M to simulate M on remaining part of w' (which is x)
- N accepts w' if after second copy of M on x it ends up in the guessed state q

Construction for CYCLE





Proving correctness

Exercise: Write down formal description of N in tuple notation starting with $M = (Q, \Sigma, \delta, s, A)$.

Need to argue that $L(N) = CYCLE(L(M))$

- If $w = xy$ accepted by M then argue that yx is accepted by N
- If N accepts w' then argue that $w' = yx$ such that xy accepted by M.

Closure Problem - Prefix

Example: PREFIX

Let L be a language over Σ.

Definition
$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

Example: PREFIX

Let L be a language over Σ.

Definition
$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

Theorem
If L is regular then PREFIX(L) is regular.

Example: PREFIX

Let L be a language over Σ.

Definition
$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

Theorem
If L is regular then PREFIX(L) is regular.

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes L

Example: PREFIX

Let L be a language over Σ.

Definition
$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

Theorem
If L is regular then PREFIX(L) is regular.

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes L

$X = \{q \in Q \mid s \text{ can reach } q \text{ in } M\}$

Example: PREFIX

Let L be a language over Σ.

Definition
$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

Theorem
If L is regular then PREFIX(L) is regular.

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes L

$X = \{q \in Q \mid s \text{ can reach } q \text{ in } M\}$

$Y = \{q \in Q \mid q \text{ can reach some state in } A\}$

Example: PREFIX

Let L be a language over Σ.

Definition
$\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

Theorem
If L is regular then $PREFIX(L)$ is regular.

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes L

$X = \{q \in Q \mid s \text{ can reach } q \text{ in } M\}$

$Y = \{q \in Q \mid q \text{ can reach some state in } A\}$

$Z = X \cap Y$

Create new DFA $M' = (Q, \Sigma, \delta, s, Z)$

Example: PREFIX

Let L be a language over Σ.

Definition
$\mathrm{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$

Theorem
If L is regular then PREFIX(L) *is regular.*

Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA that recognizes L

$X = \{q \in Q \mid s \text{ can reach } q \text{ in } M\}$
$Y = \{q \in Q \mid q \text{ can reach some state in } A\}$
$Z = X \cap Y$

Create new DFA $M' = (Q, \Sigma, \delta, s, Z)$

Claim: $L(M') = \mathrm{PREFIX}(L)$.

Exercise: SUFFIX

Let L be a language over Σ.

Definition
$\text{SUFFIX}(L) = \{w \mid xw \in L, x \in \Sigma^*\}$

Prove the following:

Theorem
If L is regular then SUFFIX(L) is regular.

Exercise: SUFFIX

Let L be a language over Σ.

Definition
$\text{SUFFIX}(L) = \{w \mid xw \in L, x \in \Sigma^*\}$

Prove the following:

Theorem
If L is regular then SUFFIX(L) is regular.

Same idea as PREFIX(L)

$X = \{q \in Q \mid s \text{ can reach } q \text{ in } M\}$

$Y = \{q \in Q \mid q \text{ can reach some state in } A\}$

$Z = X \cap Y$

With one major **difference**:

Application of closure properties to non-regularity

We can also prove non-regularity using the techniques above. For instance:

Application of closure properties to non-regularity

We can also prove non-regularity using the techniques above. For instance:

$L_1 = \{0^n 1^n \mid n \geq 0\}$

$L_2 = \{w \in \{0,1\}^* \mid \#_0(w) = \#_1(w)\}$

$L_3 = \{0^i 1^j \mid i \neq j\}$

Application of closure properties to non-regularity

We can also prove non-regularity using the techniques above. For instance:

$L_1 = \{0^n 1^n \mid n \geq 0\}$

$L_2 = \{w \in \{0,1\}^* \mid \#_0(w) = \#_1(w)\}$

$L_3 = \{0^i 1^j \mid i \neq j\}$

L_1 is not regular. Can we use that fact to prove L_2 and L_2 are not regular without going through the fooling set argument?

Application of closure properties to non-regularity

We can also prove non-regularity using the techniques above. For instance:

$L_1 = \{0^n 1^n \mid n \geq 0\}$

$L_2 = \{w \in \{0, 1\}^* \mid \#_0(w) = \#_1(w)\}$

$L_3 = \{0^i 1^j \mid i \neq j\}$

L_1 is not regular. Can we use that fact to prove L_2 and L_2 are not regular without going through the fooling set argument?

$L_1 = L_2 \cap 0^*1^*$ hence if L_2 is regular then L_1 is regular, a contradiction.

Application of closure properties to non-regularity

We can also prove non-regularity using the techniques above. For instance:

$L_1 = \{0^n 1^n \mid n \geq 0\}$

$L_2 = \{w \in \{0,1\}^* \mid \#_0(w) = \#_1(w)\}$

$L_3 = \{0^i 1^j \mid i \neq j\}$

L_1 is not regular. Can we use that fact to prove L_2 and L_2 are not regular without going through the fooling set argument?

$L_1 = L_2 \cap 0^*1^*$ hence if L_2 is regular then L_1 is regular, a contradiction.

$L_1 = \bar{L_3} \cap 0^*1^*$ hence if L_3 is regular then L_1 is regular, a contradiction