

Closure Properties

Closure Properties

- A closure property is a statement that a certain operation on languages,
- when applied to languages in a class (Class of regular languages), produces a result that is also in that class.

Closure Properties

- To prove a closure property:-
- After the application of an operator, the Resultant language is regular , If we can use any of the following representations to describe the language-
 - 1) If a Finite Automata can be constructed
 - 2) If a RE can be written (Every RE has a corresponding Regular Language)
 - 3) If Regular Grammar can be written

Building Regular Expressions

Formal Recursive Definition of Regular Expression over Σ as follows-

1. Any terminal symbol (i.e. an element of Σ), Λ and \emptyset are regular expressions. When we view a in Σ as a regular expression, we denote it by a .
2. The union of two regular expressions R_1 and R_2 , written as $R_1 + R_2$, is also a regular expression.
3. The concatenation of two regular expressions R_1 and R_2 , written as R_1R_2 , is also a regular expression.
4. The iteration (or closure) of a regular expression R written as R^* , is also a regular expression.
5. If R is a regular expression, then (R) is also a regular expression.
6. The regular expressions over Σ are precisely those obtained recursively by the application of the rules 1-5 once or several times

Closure By RE Definition

By the Definition of regular expressions, the class of regular sets over Σ is closed under union, concatenation, and closure(iteration) by the conditions 2,3,4 of the definition

Closure Under Union (RE Method)

- If L_1 and L_2 are regular languages, so is $L_1 \cup L_2$.

Proof: Let L_1 and L_2 be the languages for regular expressions R_1 and R_2 , respectively, such that

- $L(R_1)=L_1$
- $L(R_2)=L_2$

Then R_1+R_2 is also a regular expression, By definition

Also, R_1+R_2 denotes $L_1 \cup L_2$.

Thus $L_1 \cup L_2$ is closed under the class of regular languages

Closure Under Concatenation (RE Method)

- If L_1 and L_2 are regular languages, so is $L_1.L_2$

Proof: Let L_1 and L_2 be the languages for regular expressions R_1 and R_2 , respectively, such that

- $L(R_1)=L_1$
- $L(R_2)=L_2$

Then $R_1.R_2$ is also a regular expression, By definition,
Thus $L_1.L_2$ is closed under the class of regular languages

Closure Under Kleene Closure (RE Method)

- If L is a regular languages, so is L^*

Proof: Let L be the languages for regular expressions R , such that

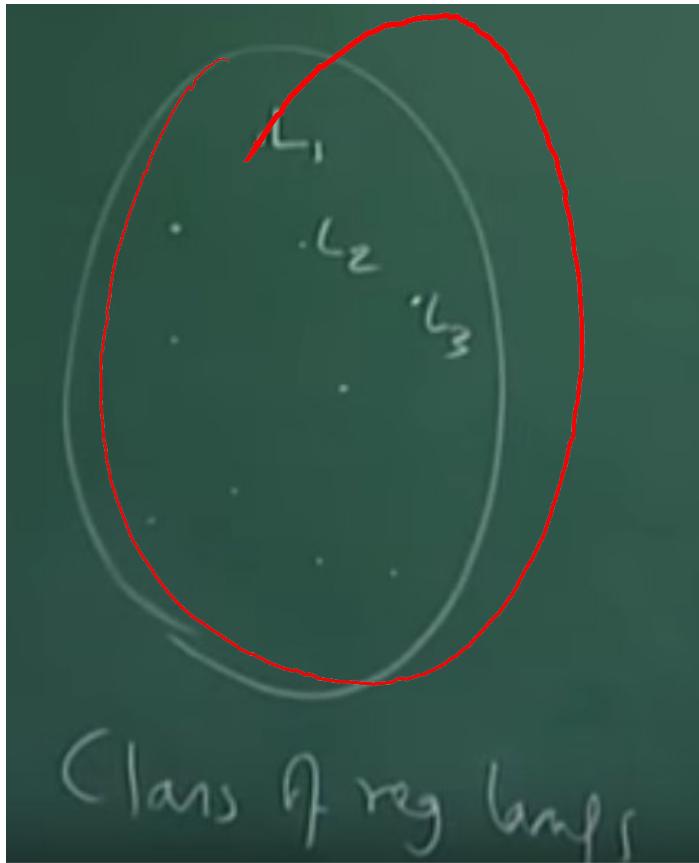
- $L(R)=L$

Then R^* is also a regular expression, By definition,
Thus L^* is closed under the class of regular languages

Closure Under and (RE Method)

- Same idea:
 - $R_1.R_2$ is a regular expression whose language is $L_1.L_2$
 - R^* is a regular expression whose language is L^* .

Closure Under Union

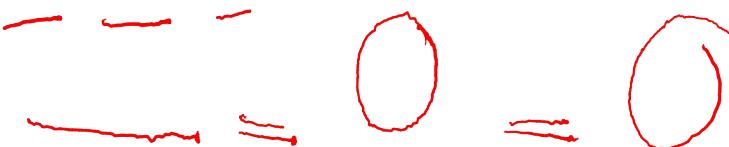


- Lets take a class of regular languages say $L_1, L_2, L_3\dots$
- Union is a binary Operation applied on this class
- If L_1 and L_2 are regular languages, so is $L_1 \cup L_2$, we get a new language.
- The New language generated is also regular and belongs to same class
- **Thus, This class is closed under Union**
- Union of Two regular Languages is Regular



Product Automaton for Union

- Lets take two DFA M1 and M2
- $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$
- $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$
- $M = (Q, \Sigma, \delta, q_0, F) = \text{Product Automaton of } M_1 \text{ and } M_2$
- ~~$Q = Q_1 \times Q_2$~~
- $\delta((q_1, q_2), a) = (\delta(q_1, a), \delta(q_2, a))$
- $q_0 = (q_0^1, q_0^2)$
- $\delta((q_0^1, q_0^2), x) = (\delta(q_0^1, x), \delta(q_0^2, x))$
- $F = \{(p_1, p_2) \mid p_1 \in F_1 \text{ or } p_2 \in F_2\}$



DFA can be constructed

Product Automaton for Union

- If Product Machine is in state F, It means either M1 machine has reached one of its final state or machine M2 has reached one of its final state
- Therefore, Any string that takes M1 to one of its final state or takes M2 to one of its final state, All such strings will be accepted by machine M
- Clearly $\text{Language}(M) = L(M1) \cup L(M2)$
- So any two regular languages for construction of Product Automaton
- Thus, Class of Regular Languages is closed under Union

Closure Under Intersection

- If L and M are regular languages, then so is $L \cap M$.

Product Automaton for Intersection

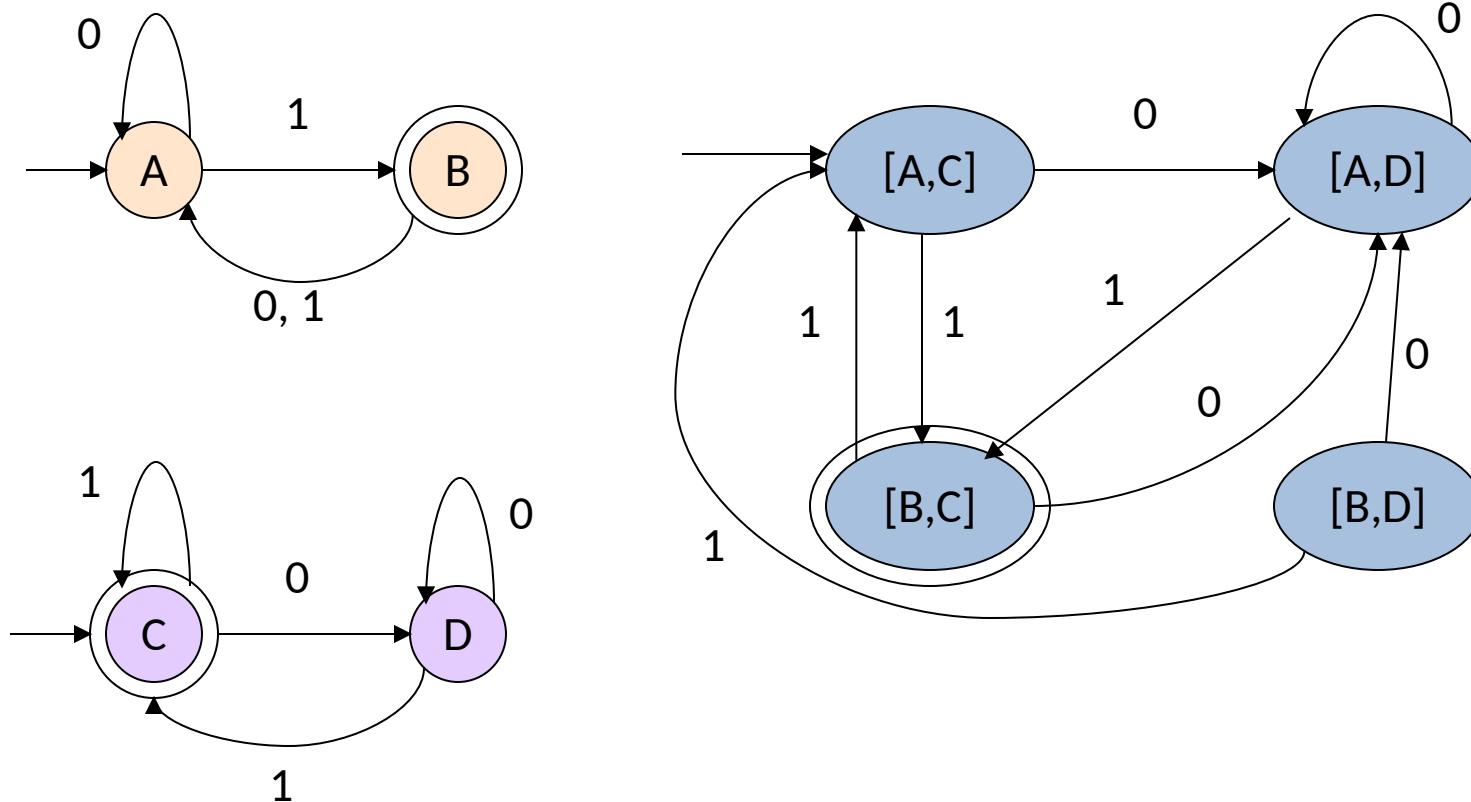
- To show $L_1 \cap L_2$ is also regular:-
- Which strings fall in $L_1 \cap L_2$?
- All strings that are in both L_1 and L_2
- What would such strings do ?
- Such strings will take the Product Automaton to a pair of states (p_1, p_2) such that both p_1 and p_2 are final states of M_1 and M_2 respectively.
- $\text{Language}(M) = L(M_1) \cap L(M_2)$

Product Automaton for Intersection

- Lets take two DFA M1 and M2
- $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$
- $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$
- $M = (Q, \Sigma, \delta, q_0, F) = \text{Product Automaton of } M_1 \text{ and } M_2$
- $Q = Q_1 \times Q_2$
- $\delta((q_1, q_2), a) = (\delta(q_1, a), \delta(q_2, a))$
- $q_0 = (q_0^1, q_0^2)$
- $\delta((q_0^1, q_0^2), x) = (\delta(q_0^1, x), \delta(q_0^2, x))$
- $F = \{(p_1, p_2) \mid p_1 \in F_1 \text{ and } p_2 \in F_2\}$

DFA can be constructed

Example: Product DFA for Intersection



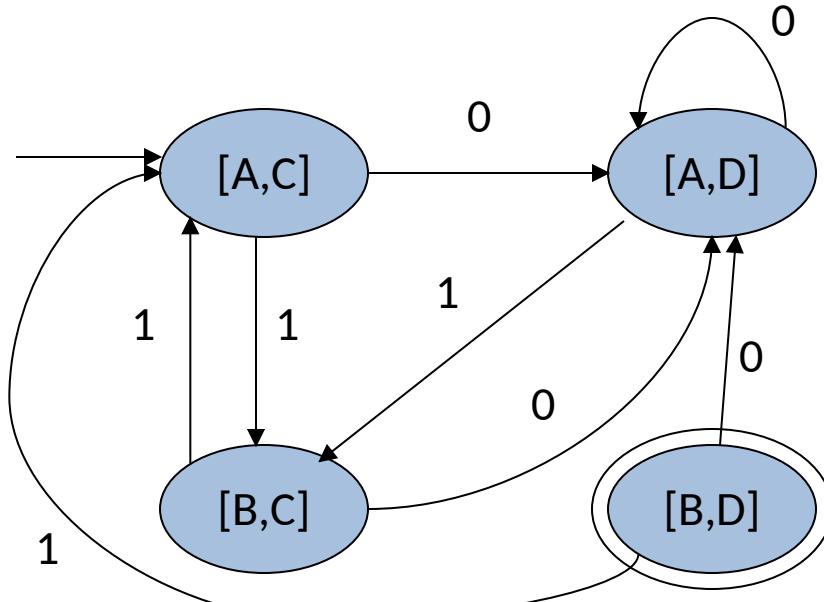
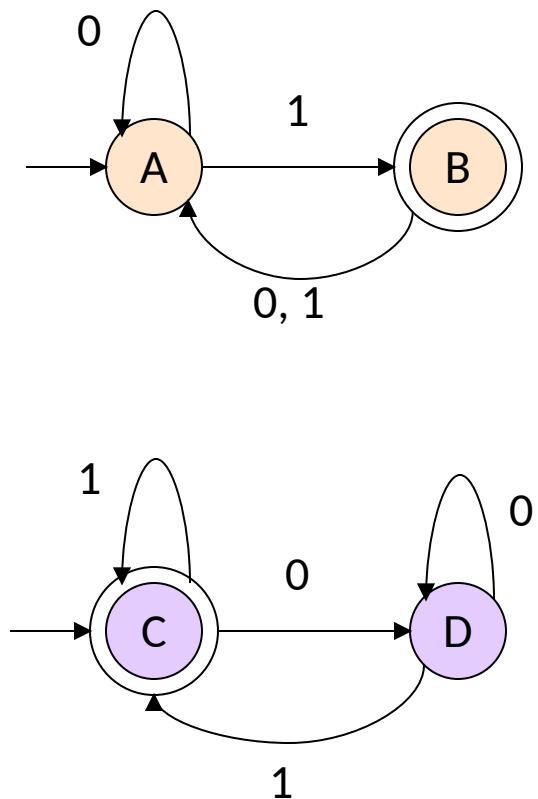
Closure Under Difference

- If L_1 and L_2 are regular languages, then so is $L_1 - L_2 =$ strings in L_1 but not L_2 .
- $L_1 - L_2 = \{x \mid x \in L_1 \text{ and } x \notin L_2\}$
- Proof: Let M_1 and M_2 be DFA's whose languages are L_1 and L_2 , respectively.
- Construct M , the product automaton of L_1 and L_2 .
- Make the final states of M be the pairs where M_1 -state is final but M_2 -state is not.

Product Automaton for Difference

- To show $L_1 - L_2$ is also regular:-
- Which strings fall in $L_1 - L_2$?
- What would such strings do ?
- Such strings will takes the Product Automaton to a pair of states (p_1, p_2) such that both p_1 is final state but p_2 is not a final states of M_1 and M_2 respectively.
- Such strings take M_1 to final state but does not take M_2 to final state,
- I.e. Product Automaton accepts all the strings accepted by L_1 but not accepted by L_2 .
- $\text{Language}(M) = L(M_1) - L(M_2)$

Example: Product DFA for Difference



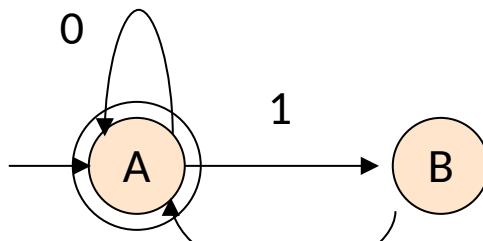
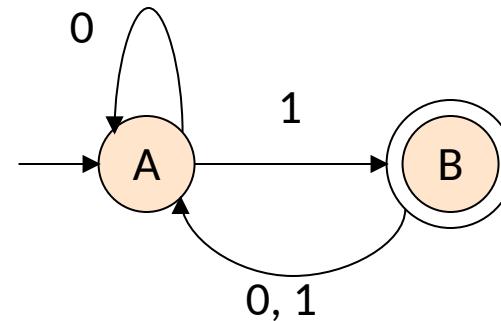
Notice: difference
is the empty language

Closure Under Complementation

- The *complement* of a language L (with respect to an alphabet Σ such that Σ^* contains L) is $\Sigma^* - L$.
- Since Σ^* is surely regular, the complement of a regular language is always regular.

Closure Under Complementation

- Initial State will Remain Initial
- Final State will become Non-Final
- Non-Final State will become Final
- Transitions will be same



- So RE for Complement= $(0^* \cdot 1 \cdot (0/1))^*$

DFA can be constructed,
RE can be written, So closed

Closure Under Reversal

- Recall example of a DFA that accepted the binary strings that, as integers were divisible by 3.
- Thus, the language of binary strings whose reversal was divisible by 3 was also regular,
- Good application of reversal-closure.

Closure Under Reversal - (2)

- Given language L , L^R is the set of strings whose reversal is in L .
- **Example:** $L = \{0, 01, 100\}$;
 $L^R = \{0, 10, 001\}$.
- **Proof:** Let E be a regular expression for L .
- We show how to reverse E , to provide a regular expression E^R for L^R .

Reversal of a Regular Expression

- **Basis:** If E is a symbol a , ϵ , or \emptyset , then $E^R = E$.
- **Induction:** If E is
 - $E=F+G$, then $E^R = F^R + G^R$.
 - $E=FG$, then $E^R = G^RF^R$
 - $E=F^*$, then $E^R = (F^R)^*$.

Example: Reversal of a RE

RE

- Let $E = \underline{01^*} + \underline{10^*}$.

~~RE~~ $E^R = (01^* + 10^*)^R = (01^*)^R + (10^*)^R$

~~RE~~ $= (1^*)^R 0^R + (0^*)^R 1^R$

~~RE~~ $= (1^R)^* 0 + (0^R)^* 1 =$

~~RE~~ $= 1^* 0 + 0^* 1.$

~~RE~~ $= =$

~~RE~~ $= =$

100
001

Induction: If E is

$E = F + G$, then $E^R = F^R + G^R$.

$E = FG$, then $E^R = G^R F^R$

$E = F^*$, then $E^R = (F^R)^*$

$$\Sigma' = \{0, 1\}$$

$$\Sigma' = \{a, b\}$$

Homomorphisms

- A **homomorphism** on an alphabet is a function that gives a string for each symbol in that alphabet.
- **Example:** $h(0) = ab; h(1) = \epsilon$.
- Extend to strings by $h(\bar{a}_1 \dots a_n) = h(a_1) \dots h(a_n)$.
- **Example:** $h(01010) = ababab$.

$$h(1010) = \epsilon ab \epsilon ab \underline{abab}$$

Closure Under Homomorphism

- If L is a regular language, and h is a homomorphism on its alphabet,
- then $h(L) = \{h(w) \mid w \text{ is in } L\}$ is also a regular language.
- **Proof:**
- Let E be a regular expression for L .
- Apply h to each symbol in E .
- Language of resulting RE is $h(L)$.

Example: Closure under Homomorphism

- Let $h(0) = ab$; $h(1) = \epsilon$.
- Let L be the language of regular expression $01^* + 10^*$.
 - Then $h(L)$ is the language of regular expression $ab\epsilon^* + \epsilon(ab)^*$.

Note: use parentheses to enforce the proper grouping.

$$Ea = a \cdot E = a$$

Example - Continued

- ~~$ab\varepsilon^* + \varepsilon(ab)^*$~~ can be simplified.
- ~~$\varepsilon^* = \varepsilon$~~ , so ~~$ab\varepsilon^* = ab\varepsilon$~~ .
- ~~ε is the identity under concatenation.~~
 - That is, $\varepsilon E = E\varepsilon = E$ for any RE E .
- Thus, ~~$ab\varepsilon^* + \varepsilon(ab)^* = ab\varepsilon + \varepsilon(ab)^* = ab + (ab)^*$~~
- ~~$ab + (ab)^*$~~ { ~~$a, ab, abab, ababab$~~ }
- Finally, $L(ab)$ is contained in $L((ab)^*)$, so a RE for $h(L)$ is ~~$(ab)^*$~~ .

Example - Continued

- $ab\varepsilon^* + \varepsilon(ab)^*$ can be simplified.
- $\varepsilon^* = \varepsilon$, so $ab\varepsilon^* = ab\varepsilon$.
- ε is the identity under concatenation.
 - That is, $\varepsilon E = E\varepsilon = E$ for any RE E .
- Thus, $ab\varepsilon^* + \varepsilon(ab)^* = ab\varepsilon + \varepsilon(ab)^* = ab + (ab)^*$.
- Finally, $L(ab)$ is contained in $L((ab)^*)$, so a RE for $h(L)$ is $(ab)^*$.

RE can be written, So closed



Inverse Homomorphisms

- Let h be a homomorphism and L a language whose alphabet is the output language of h .
- $h^{-1}(L) = \{w \mid h(w) \text{ is in } L\}$.

abab abab
↓
ε {010?}

Example: Inverse Homomorphism

- Let $h(0) = ab$; $h(1) = \epsilon$.
- Let $L = \{abab, baba\}$.
- $h^{-1}(L) = \text{the language with two 0's and any number of 1's} = L(1^*01^*01^*)$.

$\epsilon \epsilon \epsilon \epsilon ab \epsilon ab \epsilon \epsilon$
 \downarrow
 $|| \quad || \quad \underline{0} \quad \underline{1} \quad ||$

b a b a
 \downarrow
Notice: no string maps to baba; any string with exactly two 0's maps to abab.
cannot find

RE can be written, So closed

NPTEL – Theory of Computation, Assignment Questions

Q) Regular languages are closed over

- concatenation
- union
- intersection
- complement

NPTEL – Theory of Computation, Assignment Questions

Q) Regular languages are closed over

- concatenation
- union
- intersection
- complement

Accepted Answers:

union

intersection

complement

concatenation

NPTEL - Theory of Computation, Assignment Questions

9)

$DROP\text{-}ONE(L) = \{xz \mid xyz \in L \text{ where } x, z \in \Sigma^* \text{ and } y \in \Sigma\}$. Which of following are true?



Regular languages are closed under $DROP\text{-}ONE$.



For any regular language L , $DROP\text{-}ONE(L)$ is not regular.



For some regular language L , $DROP\text{-}ONE(L)$ are not regular.



For some regular languages L , $DROP\text{-}ONE(L)$ is regular but not all.

NPTEL - Theory of Computation, Assignment Questions

9)

$DROP\text{-}ONE(L) = \{xz \mid xyz \in L \text{ where } x, z \in \Sigma^* \text{ and } y \in \Sigma\}$. Which of following are true?

Regular languages are closed under $DROP\text{-}ONE$.

For any regular language L , $DROP\text{-}ONE(L)$ is not regular.

For some regular language L , $DROP\text{-}ONE(L)$ are not regular.

For some regular languages L , $DROP\text{-}ONE(L)$ is regular but not all.

Accepted Answers:

Regular languages are closed under $DROP\text{-}ONE$.