Closure Properties

Lecture 23.2.2024

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

Λ / ε empty string

Formal Recursive Definition of Regular Expression over ∑ as follows-

1. Any terminal symbol (i.e. an element of ∑), Λ and Ø are regular expressions. When we view *a* in ∑ as a regular expression, we denote it by **a**. RE symbol/string
2. The union of two regular expressions R1 and R2 , written as R1 + R2, is also a regular expression.
3. The concatenation of two regular expressions R1and R2, written as R1R2, is also a regular expression. R1·R2 R1R2
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∑ are precisely those obtained recursively by the application of the rules *1-5* once or several times

$a \cdot b^* \Rightarrow abbb\ldots\ldots$ $(a \cdot b)^* = abab\ldots\ldots$

Building Regular Expressions

Formal Recursive Definition of Regular Expression over ∑ as follows-

1. Any terminal symbol (i.e. an element of ∑), Λ and ∅ are regular expressions. When we view *a* in ∑ as a regular expression, we denote it by **a**.
2. The union of two regular expressions R1 and R2 , written as R1 + R2, is also a regular expression.
3. The concatenation of two regular expressions R1and R2, written as R1R2, 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∑ 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 $\sum$ is closed under union, concatenation, and closure(iteration) by the conditions 2,3,4 of the definition

Closure Under Union (RE Method)

- If L1 and L2 are regular languages, so is L1 ∪ L2.

Proof: Let L1 and L2 be the languages for regular expressions R1 and R2, respectively, such that

- L(R1)=L1
- L(R2)=L2 Union

Then R1+R2 is also a regular expression, By definition

Also, R1+R2 denotes L1 ∪ L2.

Thus L1 ∪ L2 is closed under the class of regular languages

Closure Under Concatenation (RE Method)

- If L1 and L2 are regular languages, so is L1.L2

Proof: Let L1 and L2 be the languages for regular expressions R1 and R2, respectively, such that

- L(R1)=L1
- L(R2)=L2

Then R1.R2 is also a regular expression, By definition,

Thus L1.L2 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

$R \to R^*$

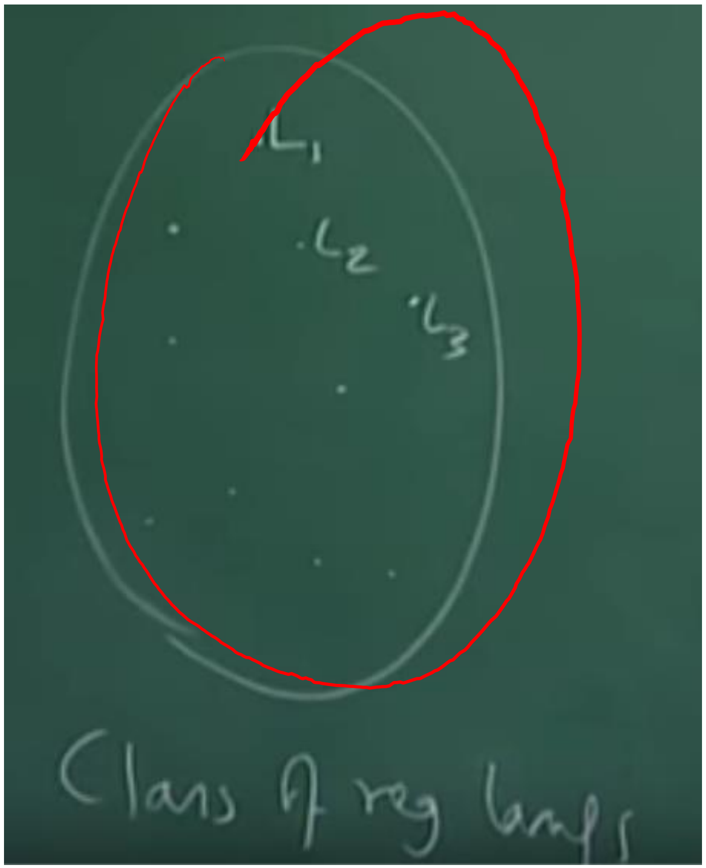
Then R* is also a regular expression, By definition,

Thus L* is closed under the class of regular languages

Closure Under and
(RE Method)

- Same idea:
  - R1.R2 is a regular expression whose language is L1.L2
  - R* is a regular expression whose language is L*.

Closure Under Union



- Lets take a class of regular languages say L1,L2,L3…
- Union is a binary Operation applied on this class
- If L1 and L2 are regular languages, so is L1 ∪ L2, 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
- $M1=(Q1, \Sigma, \delta 1, q_0^{1}, F1)$
- $M2=(Q2, \Sigma, \delta 2, q_0^{2}, F2)$
- $M= (Q, \Sigma, \delta, q_0, F)$ =Product Automaton of M1 and M2
- $Q=Q1XQ2$
- $\delta((q1,q2),a)= (\delta(q1,a), \delta(q2,a))$ input
- $q0=(q_0^{1}, q_0^{2})$
- $\delta((q_0^{1}, q_0^{2}),x)= (\delta(q_0^{1},x), \delta(q_0^{2},x))$
- $F=\{(p1,p2) \mid p1 \in F1 \text{ or } p2 \in F2\}$

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 Language(M)=L(M1) U 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

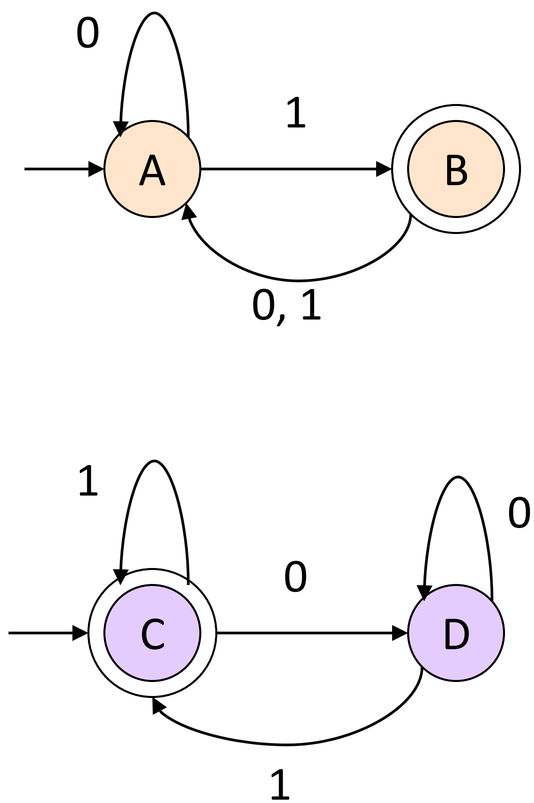
- To show $L1 \cap L2$ is also regular:-
- Which strings fall in $L1 \cap L2$?
- All strings that are in both L1 and L2
- What would such strings do ?
- Such strings will take the Product Automaton to a pair of states (p1,p2) such that both p1 and p2 are final states of M1 and M2 respectively.
- Language(M)=L(M1) $\cap$ L(M2)

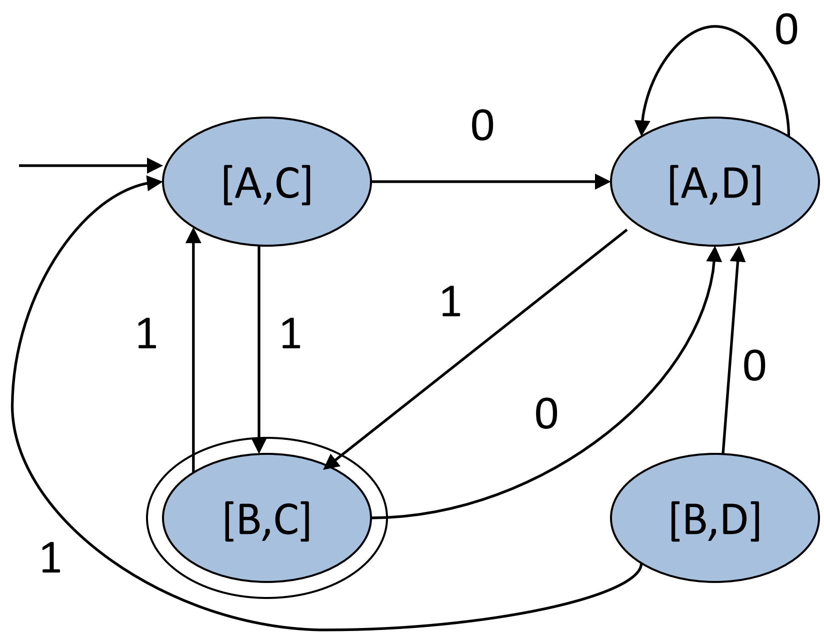
Product Automaton for Intersection

- Lets take two DFA M1 and M2
- $M1=(Q1, \sum, \delta 1, q_0^1, F1)$
- $M2=(Q2, \sum, \delta 2, q_0^2, F2)$
- $M= (Q, \sum, \delta, q_0, F)$ =Product Automaton of M1 and M2
- $Q=Q1XQ2$
- $\delta((q1,q2),a)= (\delta(q1,a), \delta(q2,a))$
- $q0=(q_0^1, q_0^2)$
- $\delta((q_0^1, q_0^2),x)= (\delta(q_0^1,x), \delta(q_0^2,x))$
- $F=\{(p1,p2) \mid p1 \in F1 \text{ and } p2 \in F2\}$

DFA can be constructed

Example: Product DFA for Intersection





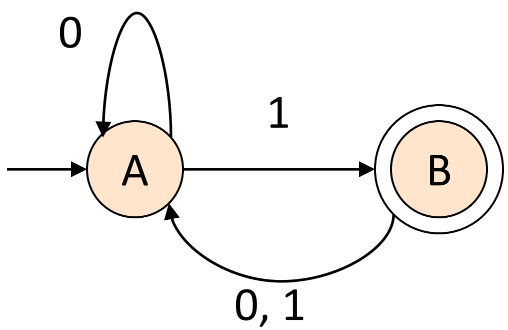
Closure Under Difference

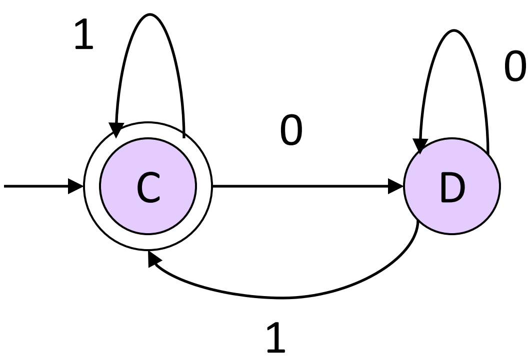
- If L1 and L2 are regular languages, then so is *L1 – L2* = strings in L1 but not L2.
- L1-L2={x | x ∈ L1 and x ∉ L2 }
- Proof: Let M1 and M2 be DFA's whose languages are L1 and L2, respectively.
- Construct M, the product automaton of L1 and L2.
- Make the final states of M be the pairs where M1-state is final but M2-state is not.

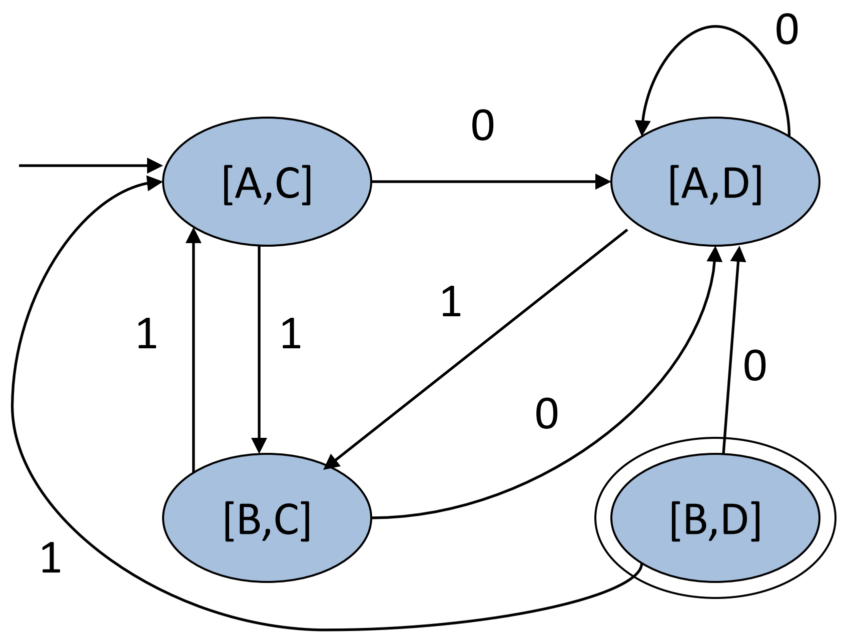
Product Automaton for Difference

- To show L1 - L2 is also regular:-
- Which strings fall in L1 - L2 ?
- What would such strings do ?
- Such strings will takes the Product Automaton to a pair of states (p1,p2) such that both p1 is final state but p2 is not a final states of M1 and M2 respectively.
- Such strings take M1 to final state but does not take M2 to final state,
- I.e. Product Automaton accepts all the strings accepted by L1 but not accepted by L2.
- Language(M)=L(M1) - L(M2)

Example: Product DFA for Difference







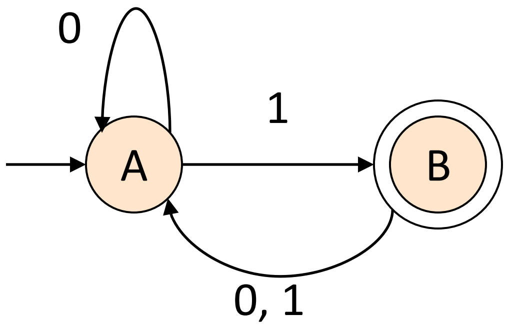
Notice: difference
is the empty language

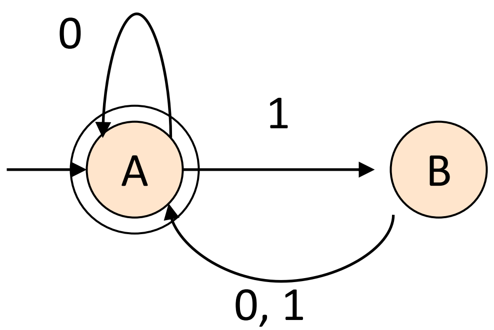
Closure Under Complementation

- 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*.1.(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 $\varnothing$, then $E^R = E$.
- 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 $E^R = (F^R)^*$.

Example: Reversal of a RE

RE

- Let $E = \mathbf{01^* + 10^*}$.
- RE $E^R = (\mathbf{01^* + 10^*})^R = (\mathbf{01^*})^R + (\mathbf{10^*})^R$
- $= (\mathbf{1^*})^R\mathbf{0}^R + (\mathbf{0^*})^R\mathbf{1}^R$
- $= (\mathbf{1}^R)^*\mathbf{0} + (\mathbf{0}^R)^*\mathbf{1}$
- $= \mathbf{1^*0 + 0^*1}$.

$E = F + G \Rightarrow E^R = F^R + G^R$

100

001

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 $E^R = (F^R)^*$.

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

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) = ϵ.
- Extend to strings by $h(a_1 \ldots a_n) = h(a_1) \ldots h(a_n)$.
- Example: h(01010) = ababab.

$h(1010) = \epsilon\, ab\, \epsilon\, ab = abab$

Closure Under Homomorphism

- If L is a regular language, and h is a homomorphism on its alphabet,
- then h(L) = {h(w) | w 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) = ϵ.
- Let L be the language of regular expression **01*** + **10***.
- Then h(L) is the language of regular expression **ab**ϵ* + ϵ(**ab**)*.

Note: use parentheses to enforce the proper grouping.

$\epsilon a = a \cdot \epsilon = a$

Example – Continued

- **ab**ϵ* + ϵ(**ab**)* can be simplified.
- ϵ* = ϵ, so **ab**ϵ* = **ab**ϵ.
- ϵ is the identity under concatenation.
  - That is, ϵE = Eϵ = E for any RE *E*.
- Thus, **ab**ϵ* + ϵ(**ab**)* = **ab**ϵ + ϵ(**ab**)* = **ab** + (**ab**)*.
- **ab** + (**ab**)*

  $\{\Lambda, ab, abab, ababab\}$
- Finally, L(**ab**) is contained in L((**ab**)*), so a RE for h(L) is (**ab**)*.

Example – Continued

- $\mathbf{ab}\epsilon^* + \epsilon(\mathbf{ab})^*$ can be simplified.
- $\epsilon^* = \epsilon$, so $\mathbf{ab}\epsilon^* = \mathbf{ab}\epsilon$.
- ϵ is the identity under concatenation.
  - That is, $\epsilon E = E\epsilon = E$ for any RE E.
- Thus, $\mathbf{ab}\epsilon^* + \epsilon(\mathbf{ab})^* = \mathbf{ab}\epsilon + \epsilon(\mathbf{ab})^* = \mathbf{ab} + (\mathbf{ab})^*$.
- 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 | h(w) is in L}.

abab abab

{0101}

Example: Inverse Homomorphism

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

ϵ ϵ ϵ ϵ ab ϵ ab ϵ ϵ

11 11 0 1 0 11

b a b a

0

cannot find h^{-1}

Notice: no string maps to baba; any string with exactly two 0's maps to abab.

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)

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

- [ ] Regular languages are closed under *DROP-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)

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

- Regular languages are closed under *DROP-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-ONE*.