Closure Properties of Regular Languages

In the previous chapters, we have covered the concept of regular expressions and regular languages in detail. In this chapter, we will take a look at the closure properties of regular languages. We need to learn this concept for theoretical computer science.

We'll explore the concept of closure and how it applies to various operations performed on regular languages here for a better understating.

What are Closure Properties?

The term "closure" means whether an operation performed on a set and it results an element that also belongs to the same set. If we consider a set of integers. If we take two integers and add them, the result is always another integer. This indicates that the set of integers is closed under addition.

Now, let's apply this concept to regular languages. A regular language is a language that can be described using a regular expression or recognized by a finite automaton. We'll examine if the output of certain operations performed on regular languages still belongs to the set of regular languages.

Key Operations and Their Closure Properties

We will see the following operations and determine whether regular languages are closed under them −

Operation	Description
Union	Combining two languages, L1 and L2, by taking all strings that are in either L1 or L2.
Concatenation	Combining two languages, L1 and L2, by creating all strings where a string from L1 is followed by a string from L2.
Closure (Kleene Star)	Creating a new language by taking all possible strings formed by concatenating zero or more copies of strings from the original language.
Complement	Creating a new language by taking all strings over the alphabet that are not present in the original language.
Intersection	Creating a new language by taking all strings that are present in both L1 and L2.

Difference	Creating a new language by taking all strings that are in L1 but not in L2.
Reversal	Creating a new language by reversing each string in the original language.
Homomorphism	Replacing each symbol in a language with another symbol according to a mapping rule.
Reverse Homomorphism	The reverse operation of homomorphism.
Quotient	Creating a new language by dividing one language by another.
Initiate	Creating a new language by taking all prefixes of strings in the original language.
Substitution	Replacing each symbol in a language with another language according to a mapping rule.
Infinite Union	Creating a new language by combining an infinite number of languages.

Closure Properties in Detail

In the above table, we highlighted the operations that can be applied on regular languages. Let us now take a look at them in detail, one by one.

Union Operation

Imagine L1 = {a, aa} and L2 = {b, bb}. The union of L1 and L2 is,

$$\mathrm{L1} \cup \mathrm{L2} = \{\mathrm{a, aa, b, bb}\}$$

Closure Property − Regular languages are closed under union. We can prove this in two ways −

- **Regular Expression** − If L1 and L2 are regular, they have regular expressions R1 and R2. The union of L1 and L2 can be expressed as R1 + R2, which is also a regular expression.
- **Finite Automata (DFA)** − If L1 and L2 are regular, they have DFAs D1 and D2. We can construct a DFA for L1 U L2 by combining D1 and D2, introducing a new start state with epsilon transitions to the start states of D1 and D2. This combined DFA represents L1 U L2, proving that it is regular.

Concatenation Operation

Continuing with our example, for concatenation,

$$\mathrm{L1} \cdot \mathrm{L2} \;=\; \{\mathrm{ab,\ aab,\ abb,\ aaab,\ aabb}\}$$

Closure Property – Regular languages are closed under concatenation.

- **Regular Expression** – If L1 and L2 are regular, they have regular expressions R1 and R2. The concatenation of L1 and L2 can be expressed as R1.R2, which is also a regular expression.
- **Finite Automata (DFA)** – We can construct a DFA for L1.L2 by connecting the final state of D1 to the start state of D2 using an epsilon transition. This combined DFA recognizes L1.L2, demonstrating its regularity.

Closure (Kleene Star)

L1* = {ε, a, aa, aaa, ...}. It includes all possible concatenations of zero or more strings from L1.

Closure Property – Regular languages are closed under closure (Kleene star).

- **Regular Expression** – If L1 is regular with a regular expression R1, then L1* can be expressed as R1*, which is also a regular expression.
- **Finite Automata (DFA)** – A DFA for L1* can be constructed by adding a new start state and making it a final state. Add epsilon transitions from this new start state to the original start state of L1 and also from all the final states of L1 back to its start state. This DFA recognizes all possible combinations of strings from L1, including the empty string.

Complement Operation

The complement of L1, denoted as L1', includes all strings over the alphabet that are not present in L1.

Closure Property – Regular languages are closed under complement.

DFA – If L1 is regular, it has a DFA D1. We can obtain the DFA for L1' by simply switching the final states and non-final states of D1. This modified DFA accepts all strings not accepted by D1, thus representing the complement of L1.

Intersection Operation

The intersection of L1 and L2, denoted as L1 ∩ L2, includes all strings that are present in both L1 and L2.

Closure Property – Regular languages are closed under intersection.

DFA – If L1 and L2 are regular, they have DFAs D1 and D2. The DFA for L1 ∩ L2 can be constructed using a product construction technique.

We create a new DFA where each state corresponds to a pair of states from D1 and D2. A transition from a state in this new DFA is determined by the transitions in both D1 and D2. A state is final if both its corresponding states in D1 and D2 are final.

Difference Operation

The difference of L1 and L2, denoted as L1 - L2, includes all strings that are in L1 but not in L2.

Closure Property – Regular languages are closed under difference.

Proof – We can express L1 - L2 as L1 ∩ L2', where L2' is the complement of L2. We've already established that both intersection and complement preserve regularity. Therefore, L1 - L2 is also regular.

Reversal of a Language

The reversal of a language, denoted as L^R, includes all strings in L with their characters reversed.

Closure Property – Regular languages are closed under reversal.

DFA – If L is regular, it has a DFA D. We can construct a DFA for L^R by reversing the direction of all transitions in D and swapping the start and final states. This reversed DFA recognizes the reversed strings of L.

Homomorphism

Homomorphism maps each symbol in a language to another symbol according to a predefined rule. For example, h(a) = b, h(b) = c.

Closure Property – Regular languages are closed under homomorphism.

- **Regular Expression** – If L is regular, it has a regular expression R. We can obtain a regular expression for h(L) by replacing each symbol in R with its corresponding image under the homomorphism h.
- **DFA** – If L is regular, it has a DFA D. We can construct a DFA for h(L) by simply renaming the labels of the transitions in D according to the homomorphism rule. This DFA recognizes the language obtained by applying the homomorphism to L.

Reverse Homomorphism

Reverse homomorphism is the reverse operation of homomorphism, where we replace symbols in the target language with their preimages according to the homomorphism rule.

Closure Property − Regular languages are closed under reverse homomorphism.

Proof − If L is regular, we can show that its preimage under a homomorphism is also regular. This is because the preimage can be constructed by using the same mapping rules but in reverse.

Since regular languages are closed under other operations, including union and concatenation, we can use these operations to construct the preimage of L, proving its regularity.

Quotient of Regular Language

The quotient of L1 by L2, denoted as L1 / L2, is a new language containing all strings that, when concatenated with any string from L2, result in a string in L1.

Closure Property − Regular languages are closed under quotient.

Proof − We can construct a DFA for L1 / L2 from the DFAs of L1 and L2. The states of the new DFA represent sets of states from L1. The transitions are defined based on the transitions in L1 and L2, ensuring that the new DFA accepts strings that, when concatenated with strings from L2, result in strings accepted by L1.

Initiate Operation

The **initiate** of L, denoted as **init(L)**, includes all prefixes of strings in L.

Closure Property − Regular languages are closed under initiate.

Proof − We can construct a DFA for init(L) from the DFA of L. We simply make all states in the original DFA final. This ensures that the new DFA accepts all prefixes of strings accepted by the original DFA.

Substitution Operation

Concept − Substitution replaces each symbol in a language with another language according to a mapping rule.

Closure Property − Regular languages are closed under substitution.

Proof − If L is regular and each language in the substitution rule is also regular, we can construct a DFA for the substituted language using a composition of DFAs.

The DFA for the substituted language would have states representing combinations of states from the original DFA and the DFAs for the languages in the substitution rule. The transitions would be defined based on the transitions in these component DFAs.

Infinite Union

The infinite union of languages combines an infinite number of languages.

Closure Property − Regular languages are **not** closed under infinite union.

Counter Example − Consider the infinite union of languages $L_i = \{a^i\}$, where i is a natural number. Each L_i is regular, but their infinite union represents the language of all strings consisting only of 'a', which is not regular.

Conclusion

In this chapter, we explained the closure properties for regular languages. We covered the operations on regular languages with descriptions and finally touched upon their properties along with examples.