Recitation - Week 3

CSE 355: Theory of Computation

Today's Agenda

1. Define Regular Expressions
2. Convert a Language to a Regular Expression
3. Convert Regular Expressions to a Language
4. Convert a Regular Expression to a Finite Automaton
5. Show equivalence between DFA/NFA and Regular Expressions

What are Regular Expressions?

Regular **Expressions** use special characters to represent a set of strings. A regular **Expression** can be one of the following:

1. a ϵ Σ (Alphabet Set) / ε (Empty String) / ϕ (Empty Set)
2. $R_1 \cup R_2$; where R_1 and R_2 are any Regular Expressions
3. $R_1 \circ R_2$; where R_1 and R_2 are any Regular Expressions
4. R * ; where R is any Regular Expression

UNION - $\cup$

Union combines two sets of strings into one set. Recall how union works in sets

For example,

L_1 = {“010”, “10110”}

L_2 = {“11”, “0010”}

and if $L_3 = L_1 \cup L_2$ then, L_3 = {“010”, “10110”, “11”, “0010”}

CONCATENATION - ∘

Concatenation combines two set(s) of strings by joining them together into one string. Think about adding two strings

For example,

L_1 = {“010”, “10110”}

L_2 = {“11”, “0010”}

and if $L_3 = L_1 \circ L_2$ then, L_3 = {“01011”, “0100010”, “1011011”, “101100010”}

or if string s_1 = “abcd” & s_2 = “1234” then,

$s_1 \circ s_2$ = “abcd1234”

Kleene Star - *

It is a unary operator and it means 0 or more concatenated iterations of the operand (input).

For example,

Let string s_1 = “ab” then,

s_1* = {ε, “ab”, “abab”, “ababab”, “abababab”, …}

Let L_1 = {“c”, “d”} then,

L_1* = all string possible over “c” and “d”, including the empty string!

Language to Regular Expression

Consider the Language L = “ { w ϵ Σ | w = “ 1^n0^m ” where n >= 2, m >= 0 }

Solution : “1111…000…”

Regular Expression :

=> 11 ○ 1* ○ 0* => 111*0*

Explanation :

We need the string to begin with 2 1’s and then have any amount of 1’s followed by any amount of 0’s. Result : 2 mandatory 1’s (11) + >=0 1’s (1*) + >=0 0’s (0*)

Regular Expression to Language

Consider the Language regex : R = $0110^* \cup \Sigma^*(101)^*\Sigma^*$

Solution : L = { w ϵ $\mathbf{\Sigma}$ | w begins with 011 followed by 0 or more 0's OR "101" is a substring of w }

Explanation :

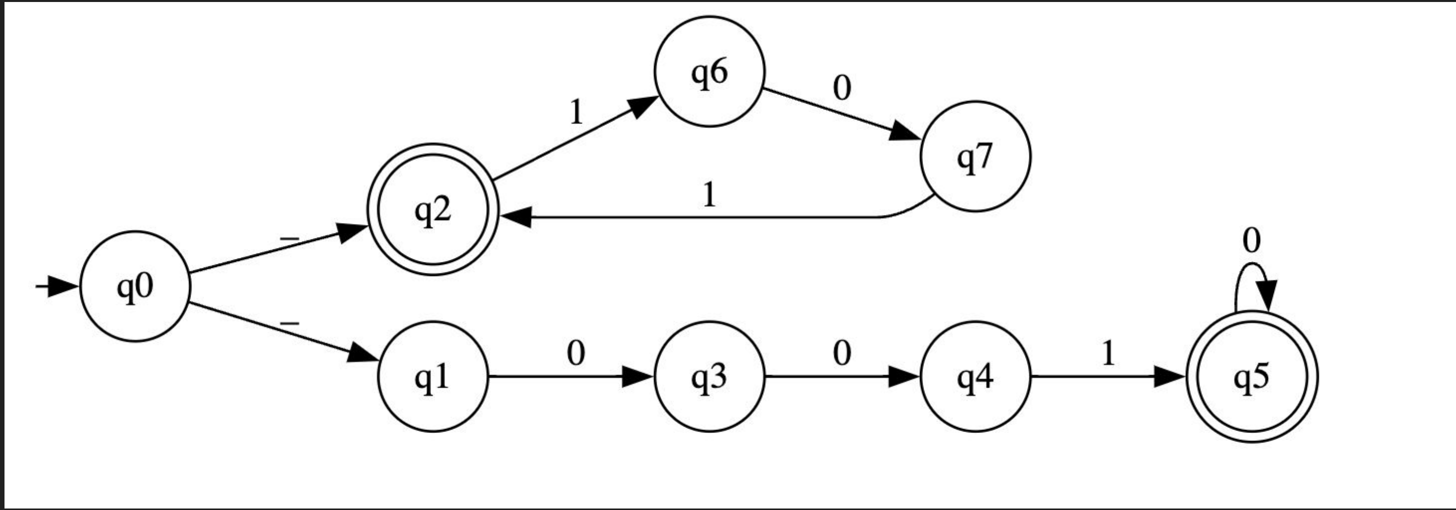
The language must accept either strings that follow the regex 0110^* OR $\Sigma^*(101)^*\Sigma^*$

0110^* means that the string must start with 011 and proceed to have 0 or more 0's

$\Sigma^*(101)^*\Sigma^*$ means "any string over the alphabet" + $(101)^*$ + "any string over the alphabet" => 101 must be a substring.

Regular Expression to Finite Automaton

Consider a Regular Expression R = 0110* U (101)*



Look at Module 4: Readings and Videos for more information!

Equivalence between NFA and Regular Expressions

I showed you one example where I created an NFA from a regular expression. Can you think of any that cannot be converted to an NFA?

Equivalence between NFA and Regular Expressions

I should be able to convert any regular expression to an NFA with the help of transformations (Look at Module 4: Readings and Videos for the transformations).

So, a regular expression is equivalent to an NFA in power.

And we know that an NFA is equivalent to a DFA in power.

This means that a regular expression is equivalent to power in DFA and thus, all languages derived from a regular expression are regular!

Conclusion

Lemma

If a language is regular, there exists a regular expression that describes it.

Fetch recitation material

View any recitation related material on this github repo

https://github.com/tanay-jaiman/CSE355_Recitations/tree/main/Module2

Any Questions?