CSC 222: AUTOMATA THEORY

Lecture IV: Regular Expressions

Recap

- A precise definition aids in carrying out proofs
- A concise definition assists in understanding
- Recursive definitions are precise and concise
- Regular expressions ➔ another method of defining languages precisely and concisely

Need for more precision!

- We have used the following symbols to define languages e.g.

  L_1={x^n for n=1 2 3 ...}

  L_{ODD}={x^n for n=1 3 5 7...}

  L_5={x^n for n=1 4 9 16 ...}

  L_6={x^n for n=3 7 8 29 ...} ?
- For computation, more precise language-defining symbolism is required.

Applying a closure directly to alphabet symbols

Consider L_4

L_4 = {Λ x xx xxx xxxx ...}

possible (previous) definitions:

- As a closure of a smaller set:

  Let S = {x}. Then L_4 = S*.

  shorthand representation: L_4 = {x}*
- Now, apply kleene star directly to *x* i.e.

  X*

  this simple expression, ***x**** indicates some sequences of *x*'s (or none at all).

Applying a closure directly to alphabet symbols

- This notation is helpful in defining languages:

  L_4 = Language(**x***).

  since **x*** is any string of *x*'s, then L_4 is the set of all possible strings of *x*'s of any length (including Λ).
- Note:

  x* is a language-defining symbol, while L_4 is the name we give to a certain language.

Example 1

Let us define language *L* over the alphabet Σ = {a b}

L = {a ab abb abbb abbbb abbbbb …}

What is the descriptive definition of *L*?

L = language(ab^*)

Example 2

L_1 = {Λ ab abab ababab ...}

What is the descriptive definition of L_1?

L_1 = *Language*(ab)*

Note!

ab* ≠ (ab)*

Example 3

L_1 = {x^n *for n = 1 2 3 …*}

- How would you define the language L_1 using this notation?

  Remember that x^* generates Λ!

Example 3 Solutions

L_1 = language(**xx***)

L_1 = language($\mathbf{x}^+$)

L_1 = language(**xx*x***)

L_1 = language(**x*xx***)

L_1 = language($\mathbf{x}^+$**x***)

L_1 = language(**x***$\mathbf{x}^+$)

L_1 = language(**x*x*x*xx***)

...

Some more languages

- ab*a
- a*b*
- a*b*a*
- What is the difference between the language denoted by abb*a and ab*a?
- What are some alternative ways that you might define the Language L_{odd} using this notation?.

The plus sign (+) = Either .. or

- Consider the following language *T* defined over the alphabet Σ = {a b c}

  T = {a c ab cb abb cbb abbb cbbb abbbb cbbbb…}

- All the words in *T* begin with either an ***a*** or a ***c*** followed by any number of ***b***'s (including no ***b***'s).
- We can denote this language by using the following regular expression:

  T = language((a + c)b^*)

- The set of all strings that can be produced using this expression is the language of the expression.
- How can you define T recursively?

Defining finite languages

- L = {aaa aab aba abb baa bab bba bbb} *i.e. all strings over the alphabet {a b} of length 3.*
- We can define this language in different ways using regular expressions
- L = language ((a + b)(a + b)(a + b))
- A shorthand way of expressing the same thing is:

  L = language $((a + b)^3)$

- A whole class of languages of strings of a's and b's of various lengths can be defined using the regular expression L = language $((a + b)^n)$
- Finally the infinite language that contains all possible strings of a's and b's of finite length can be defined using the regular expression $(a + b)^*$

How could we define the following languages?

Assuming Σ = {a b},

- How do you define the language of all words that begin with the letter ***a*** followed by ***a***'s and ***b***'s of any length?
- How do you define the language of all words that begin with an ***a***, then anything (any combination of ***a***'s and ***b***'s) and ends with a ***b***?
- The language of all strings that have at least one b in them?
- The language of all strings of b's?

Formal Definition of Regular Expressions

- The languages defined by regular expressions are referred to as regular languages.
- A regular language is one that can be defined by a regular expression even though it may also have many other fine definitions.
- Regular expressions, though powerful and concise, are of limited capacity since there are many languages that cannot be defined by regular expressions.
- The symbols that appear in regular expressions are: the letters of the alphabet Σ, Λ, parentheses, star operator (*) and plus sign (+).

Recursive definition of Regular Expressions

Rule 1: Every letter of Σ is a regular expression. Λ itself is a regular expression.

Rule 2: If r_1 and r_2 are regular expressions then so are (r_1) r_1r_2 r_1+r_2 r_1^*

Rule 3: Nothing else is a regular expression (exclusion rule)

why don't we have r_1^+ as part of the definition?

Union of Regular Expressions

- Consider the language defined by the regular expression $(a + b)^*a(a + b)^*$
- What is the descriptive definition of this language?
- Is *abbaab* a string in this language?
- What regular expression defines those words omitted from this language?

Union of Regular Expressions

- If we combine the two regular expressions, we should produce the language of all strings, since:

all strings = (all strings with an *a*) + (all strings without an *a*)

$(a + b)^* = (a + b)^*a(a + b)^* + b^*$

Here, we have added two language-defining expressions to produce an expression that defines the union of the two languages defined by the individual expressions.

Note! Not similar to addition in the algebraic sense, since

a* = a* + a*
a* = a* + a* + a*
a* = a* + aaa
...

More language definitions

- What about the language with exactly two *a*'s?
- What about the language with at least one *a* and at least one *b*?
  - What is another way that we could define this language?
- Using the regular expressions defined above, how can you define all possible strings of *a*'s and *b*'s i.e. $(a + b)^*$?

More on equivalence

$(a + b)^* = (a + b)^* + (a + b)^*$

$(a + b)^* = (a + b)^*(a + b)^*$

$(a + b)^* = \Lambda + a(a + b)^* + b(a + b)^*$

$(a + b)^* = (a + b)^* ab(a + b)^* + b^* a^*$

Note! These are not algebraic polynomials!

Distributive Law

Let V = {Λ a b ab bb abb bbb abbb bbbb ...}

What is the descriptive definition of this language?

Distributive Law

Let V = {Λ a b ab bb abb bbb abbb bbbb ...}

What is the descriptive definition of this language?

The regular expression defining V is:
$b^* + ab^*$

Alternatively, V can be defined by $(\Lambda + a)b^*$

since $b^* = \Lambda b^*$, applying distributive law we get:

$\Lambda b^* + ab^* = (\Lambda + a)b^*$

Here, we have factored out b* by taking advantage of the distributive law.

Distributive Law

Let T = {a c ab cb abb cbb ...}

What is the descriptive definition of this language?

What alternate regular expressions define this language?

Note: Distributive law must be used with care since expressions may be distributed but operators cannot!

e.g. $(ab)^* \neq a^*b^*$

The Product Set of two Languages

Definition:

- If S and T are two sets of strings of letters (whether they are finite or infinite sets) we define the product set of strings of letters to be:

  ST = {all combinations of a string from S concatenated with a string from T in that order}

Example:

S = {a aa aaa} T = {bb bbb}, then ST = {abb abbb aabb aabbb aaabb aaabbb}

(Note that these words are not in lexicographic order, but in size ordering!)

If S = {aba bab} and T = {a b} what is the product set ST?

If P = {a bb bab} and Q = {Λ bbbb} what is the product set PQ?

Note: if L is any language, then LΛ = ΛL = L

An important definition

- The rules for associating a language with a regular expression can be rigorously defined using a recursive definition:
- Rule 1: The language associated with Λ is just {Λ}, a one-word language.
- Rule 2: The language that is associated with a single letter regular expression is that one letter word alone.
- Rule 3: If r_1 is a regular expression associated with the language L_1 and r_2 is a regular expression associated with the language L_2, then:

An important definition

- The regular expression $(r_1)(r_2)$ is associated with the product L_1L_2:

  $$\text{language}(r_1 r_2) = L_1 L_2$$

- The regular expression $r_1 + r_2$ is associated with the language formed by the union of sets L_1 and L_2:

  $$\text{language}(r_1 + r_2) = L_1 + L_2$$

- The regular expression $(r_1)^*$ is assocated with the language L_1^*, the kleene closure of set L_1:

  $$\text{language}(r_1)^* = L_1^*$$

Rule 4: Exclusion rule

Recap

- Recursive definition proves that there is some language associated with every regular expression.
- As we build up a regular expression from the rules, we simultaneously build up the corresponding language.
- How do we determine if two regular expressions define the same language?
- Every regular expression is associated with some language; is it also true that every language can be described by a regular expression?

Finite Languages are Regular

Theorem: If *L* is a finite language (i.e. has finitely many words), then *L* can be defined by a regular expression.

Proof:

if *L* = {baa abbba baba} then baa + abbba + baba is the regular expression that defines it.

If *T* = {aa ab ba bb}, then using the algorithm above, the regular expression that defines *T* is aa + ab + ba + bb.

What is another regular expression that defines *T*?

Regular expressions for a language need not be unique, we need only to show that one exists!

The algorithm above does not work for infinite languages. Why?

Finite languages are regular

Using the theorem, what is the regular expression for L, where L = {Λ x xx xxx xxxx}?

Cryptic Regular Expressions

- Consider the following languages defined over the alphabet Σ = {a b}:
  - What regular expression defines the language of strings of a's and b's that at some point contain a double letter?
  - How can you expand this expression to be one that defines all strings?
- Clearly, the result is not obvious!

Another Example

- Consider the language EVEN-EVEN with Σ = {a b} whose strings have an even number of a's and an even number of b's
- What is the regular expression that defines E?

Exercises

- Construct a regular expression defining the following languages over the alphabet {0 1}.

  1. All words in which 0 appears quadrupled, if at all. This means that every clump of 0's contains 4 or 8 or 12 or 16 .... 0's.
  2. All words that contain exactly two 1's or exactly three 0's, not more.
  3. All strings that end in a double letter.

- alphabet $\Sigma = \{a, b, c\}$:

  1. All strings containing exactly one ***a***.
  2. All strings containing at least one occurrence of each symbol in Σ.