Theory of Computation

النظرية الاحتسابية
المحاضرة الثانية والثالثة

كلية التربية للعلوم الصرفة / جامعة ديالى

اعداد
م.د. محمد سامي محمد

قسم علوم الحاسوب – المرحلة الثانية

RECURSIVE DEFINITIONS

One of the mathematical tools that we shall find extremely useful in our study,

A recursive definition is characteristically a three-step process:-

First, we specify some basic objects in the set.

Second, we give rules for constructing more objects in the set from the ones we already know.

Third, we declare that no objects except those constructed in this way are allowed in the set.

RECURSIVE DEFINITIONS

Example

Suppose that we are trying to define the set of positive even integers for someone who knows about arithmetic but has never heard of the even numbers.

Method 1:- EVEN is the set of all positive whole numbers divisible by 2.

Method 2:- EVEN is the set of all 2n where n = 1 2 3 4 ...

Method 3:-
The set EVEN is defined by these three rules:

Rule 1 2 is in EVEN.
Rule 2 If x is in EVEN then so is x + 2.
Rule 3 The *only* elements in the set EVEN are those that can be produced from the two rules above.

RECURSIVE DEFINITIONS

There is a reason that the third definition is less popular than the others: It is much harder to use in most practical applications.

suppose that we wanted to prove that 14 is in the set EVEN.

Using the first definition we divide 14 by 2 and find that there is no remainder. Therefore, it is in EVEN.

To prove that 14 is in EVEN by the second definition we have to somehow come up with the number 7 and then, since 14 = (2)(7), we know that it is in EVEN

By Rule 1, we know that 2 is in EVEN.
Then by Rule 2 we know that 2 + 2 = 4 is also in EVEN.
And, at last, by applying Rule 2 once more, to the number 12, we conclude that 12 + 2 = 14 is, indeed, in EVEN.

RECURSIVE DEFINITIONS

The set POLYNOMIAL is defined by these four rules:
Rule 1 Any number is in POLYNOMIAL.
Rule 2 The variable x is in POLYNOMIAL.
Rule 3 If *p* and q are in POLYNOMIAL, then so are p + q and (p) and *pq.*
Rule 4 POLYNOMIAL contains only those things which can be created by the three rules above.

RECURSIVE DEFINITIONS

Example

Some sequence of applications of these rules can show that $3x^2 + 7x - 9$ is in POLYNOMIAL.
By Rule 1 3 is in POLYNOMIAL
By Rule 2 x is in POLYNOMIAL
By Rule 3 (3)(x) is in POLYNOMIAL, call–it 3x
By Rule 3 *(3x)(x)* is in POLYNOMIAL, call it $3x^2$
By Rule 1 7 is in POLYNOMIAL
By Rule 3 (7)(x) is in POLYNOMIAL
By Rule 3 *3x′*+ 7x is in POLYNOMIAL
By Rule 1 –9 is in POLYNOMIAL
By Rule 3 3x 2 + 7x + (– 9) = 3x2 + 7x – 9 is in POLYNOMIAL.

REGULAR EXPRESSIONS

The language-defining symbols we are about to create are called Regular Expressions (RE).

We will define the term regular expression itself recursively. The languages that are associated with these regular expressions are called regular languages and are also said to be defined by finite representation.

ملاحظة : في هذا الفصل المجموعة محددة اي تحتوي على عناصر محددة وليست كالسابق

REGULAR EXPRESSIONS

Let us reconsider the language L4.

L4,={ Λ x xx *xxx xxxx . . .}*

In that chapter we presented one method for indicating this set as the closure of a smaller set.

Let S = {x}. Then L4 = *S**

As shorthand for this we could have written:
L4 = {x}*

We now introduce the use of the Kleene star applied not to a set but directly to the letter x and written as a superscript as if it were an exponent. x*

REGULAR EXPRESSIONS

This notation can be used to help us define languages by writing L4 = language (x*)

Suppose that we wished to describe the language L over the alphabet S = *{a,b}* where

L = {a ab abb abbb abbbb ... }

We could summarize this language by the English phrase "all words of the form one a followed by some number of b's (maybe no b's at all.)"

Using our star notation, we may write:
L = language (a b*) or without the space,
L = language (ab*)

REGULAR EXPRESSIONS

We can apply the Kleene star to the string *ab* if we want, as follows: (ab)* = Λ or *ab* or *abab* or *ababab* ...

EXAMPLE

The language defined by the expression ab*a

language (ab*a) = {aa *aba abba abbba abbbba* ... .}

REGULAR EXPRESSIONS

ملاحظة اذا احتوى القانون على نجمة هذا يعني ليس بالضرورة ان تكون موجودة

Remember x* can always be Λ.

EXAMPLE The language of the expression

a*b*

language (a*b*) = {Λ a *b* aa ab *bb* aaa aab *abb bbb* aaaa . . . }

Notice that ba and aba are not in this language. Notice also that there need not be the same number of a's and b's.

Here we should again be very careful to observe that

a*b* # (ab)*

REGULAR EXPRESSIONS

The language defined by the expression a*b*a* contains the word *baa* since it starts with zero a's followed by one b followed by two a's.

EXAMPLE

The following expressions both define the language

$$L2 = \{x^{Odd}\}$$

x(xx)* or (xx)*x but the expression x*xx* does not since it includes the word *(xx) x (x).*

REGULAR EXPRESSIONS

Question:-

Which one represent L = $\{x^{Even}\}$

- $x^* \; xx^*$
- $x^*(xx)^*$
- xx^*
- $(xx)^*$
- None
- All of above

REGULAR EXPRESSIONS

We now introduce another use for the plus sign. By the expression x + y, we mean "either x or y".

This means that x + y offers a choice, much the same way that x* does. Care should be taken so as not to confuse this with + as an exponent.

EXAMPLE

Consider the 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 an a or a c and then are followed by some number of b's.
Symbolically, we may write this as

T = language ((a + c) b*)
= language (either a or *c* then some b's)

REGULAR EXPRESSIONS

EXAMPLE

Now let us consider a finite language L

L = {aaa *aab aba abb baa bab bba bbb}*

The first letter of each word in L is either an a or a b. The second letter of each word in L is either an a or a *b.* The third letter of each word in L is either an a or a *b.*

So we may write

L = language ((a + b)(a + b)(a + b))

or for short,

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

REGULAR EXPRESSIONS

If we want to define the set of all seven letter strings of a's and b's, we could write $(a + b)^7$.
In general, if we want to refer to the set of all possible strings of a's and b's of any length whatsoever we could write, (a + b)*

This is the set of all possible strings of letters from the alphabet 2 = *{a, b}*

Again this expression represents a language. If we decide that * stands for 5, then (a + b)* gives

$(a + b)^5$= (a+b)(a+b)(a+b)(a+b)(a+b)

REGULAR EXPRESSIONS

We can describe all words that begin with the letter a simply as:

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

that is, first an a, then anything (as many choices as we want of either letter a or *b).*

All words that begin with an a and end with a b can be defined by the expression

$$a(a + b)^*b = a \text{ (arbitrary string) } b$$

REGULAR EXPRESSIONS

EXAMPLE

Let us consider the language defined by the expression

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

For example, the word *abbaab* can be considered to be of this form in three ways:

(Λ) a *(bbaab)* or *(abb)* a (ab) or (abba) a *(b)*

Question:- what about aaaaa and bbaa how many different expression can be described?

REGULAR EXPRESSIONS

EXAMPLE

The language of all words that have at least two a's can be described by the expression

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

= (some beginning)(the first important a)(some middle)(the second important a)(some end)

where the arbitrary parts can have as many a's (or b's) as they want.

In the last three examples we have used the notation $(a + b)^*$ as a factor to mean "any possible substring,"

REGULAR EXPRESSIONS

EXAMPLE

Another expression that denotes all the words with at least two a's is:

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

We scan through some jungle of b's (or no b's) until we find the first a, then more b's (or no b's), then the second a, then we finish up with anything.

In this set are *abbbabb* and aaaaa.
We can write:

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

where by the equal sign we do not mean that these expressions are equal algebraically in the same way as

$$x+x=2x$$

REGULAR EXPRESSIONS

We could write
language ((a + b)*a(a + b)*a(a + b)*)
= language (b*ab*a (a + b)*)
= all words with at least two a's.

To be careful about this point, we say that two regular expressions are equivalent if they describe the same language.

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

↑ next to last a ↑ last a

and

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

↑ first a ↑ last a

REGULAR EXPRESSIONS

EXAMPLE

If we wanted all the words with *exactly* two a's, we could use the expression

b*ab*ab*

which describes such words as *aab, baba,* and *bbbabbbbab.*

To make the word *aab,* we let the first and second b* become A and the last becomes b.

REGULAR EXPRESSIONS

EXAMPLE

The language of all words that have at least one a and at least one b is somewhat trickier. If we write

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

= (arbitrary) a(arbitrary) b(arbitrary)

we could define this set by the expression:

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

Here we are still using the plus sign in the general sense of disjunction (or). We are taking the union of two sets,

REGULAR EXPRESSIONS

يمكن التعبير عن الصيغ بصيغاخرى مشابهة

EXAMPLE

All temptation to treat these language-defining expressions as if they were algebraic polynomials should be dispelled by these equivalences:

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

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

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

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

REGULAR EXPRESSIONS

Usually when we employ the star operator, we are defining an infinite language.
We can represent a finite language by using the plus (union sign) alone. If the language L over the alphabet X = {a, *b*} contains only the finite list of words given below,

$$L = \{abba\ baaa\ bbbb\}$$

then we can represent L by the symbolic expression

$$L = \text{language (abba + baaa + bbbb)}$$

If L is a finite language that includes the null word Λ, then the expression that defines L must also employ the symbol Λ.
For example, if

$$L = \{\Lambda\ a\ aa\ bbb\}$$

then the symbolic expression for L must be

$$L = \text{language (}\Lambda + a + aa + bbb\text{)}$$

REGULAR EXPRESSIONS

EXAMPLE

Let V be the language of all strings of a's and b's in which the strings are either all b's or else there is an a followed by some b's. Let V also contain the word Λ.

$$V = \{\Lambda\ a\ bab\ bb\ abb\ bbb\ abbb\ bbbb\ \ldots.$$

We can define V by the expression

$$b^* + ab^*$$

where the word Λ is included in the term b^*. Alternatively, we could define V by the expression:

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

This would mean that in front of the string of some *b's* we have the option of either adding an a or nothing. Since we could always write $b^* = \Lambda\, b^*$,

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

REGULAR EXPRESSIONS

Let us reconsider the language
T = {a c ab cb abb cbb … }.

T can be defined as above by
$(a + c)b^*$
but it can also be defined by
$ab^* + cb^*$

This is another example of the distributive law.

REGULAR EXPRESSIONS

If r1 = aa + b then the expression r1* technically refers to the expression

$$r1^* = aa + b^*$$

which is the formal concatenation of the symbols for r, with the symbol *, but what we generally mean when we write r1* is actually (r1)*

$$(r1)^* = (aa + b)^*$$

REGULAR EXPRESSIONS

DEFINITION

If S and T are 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 }

EXAMPLE

If

$$S = \{a\ aa\ aaa\} \text{ and } T = \{bb\ bbb\}$$

then

$$ST = \{abb\ abbb\ aabb\ aabbb\ aaabb\ aaabbb\}$$

EXAMPLE

If $S = \{a\ bb\ bab\}$ $T = \{a\ ab\}$

Then $ST = \{aa\ aab\ bba\ bbab\ baba\ babab\}$

REGULAR EXPRESSIONS

EXAMPLE

If $P = \{a\ bb\ bab\}$ and $Q = \{\Lambda\ bbbb\}$

then

$PQ = \{a\ bb\ bab\ abbbb\ bbbbbb\ babbbbb\}$

ملاحظة :- فقط في حالة الصفر موجود في المجموعتين يكون في الناتج صفر

EXAMPLE

If $M = \{\Lambda\ x\ xx\}$ $N = \{\Lambda\ y\ yy\ yyy\ yyyy \ldots\}$

then

$MN = \{\Lambda\ y\ yy\ yyy\ yyyy$ **...**

x xy xyy xyyy xyyyy…

xx **xxy** xxyy xxyyy xxyyyy.. **.}.**

REGULAR EXPRESSIONS

Using regular expressions, these four examples can be written as:

$(a + aa + aaa)(bb + bbb) = abb + abbb + aabb + aabbb + aaabb + aaabbb$

$(a + bb + bab)(a + ab) = aa + aab + bba + bbab + baba + babab$

$(a + bb + bab)(A + bbbb) = a+bb+bab+abbbbb +bbbbbb + babbbbb$

$(\Lambda + x + xx)(y^*) = y^* + xy^* + xxy^*$

REGULAR EXPRESSIONS

THEOREM 5
If L is a finite language (a language with only finitely many words), then L can be defined by a regular expression.

PROOF

For example, the regular expression that defines the language

$$L = \{baa\ abbba\ bababa\}$$

is

$$baa + abbba + bababa$$

Another example If

$$L = \{aa\ ab\ ba\ bb\}$$

the algorithm described above gives the regular expression

$$aa + ab + ba + bb$$

Another regular expression that defines this language is

$$(a + b)(a + b)$$

REGULAR EXPRESSIONS

EXAMPLE

Let

$$L = \{\Lambda \text{ x xx } xxx\ xxxx\ xxxxx\}$$

The regular expression we get from the theorem is

$$\Lambda + x + xx + xxx + xxxx + xxxxx$$

A more elegant regular expression for this language is

$$(\Lambda + x)^5$$

Of course the 5 is, strictly speaking, not a legal symbol for a regular expression although we all understand it means

$$(\Lambda + x)(\Lambda + x)(\Lambda + x)(\Lambda + x)(\Lambda + x)$$

REGULAR EXPRESSIONS

EXAMPLE

Consider the expression:

$$(a + b)^*(aa + bb)(a + b)^*$$

This is the set of strings of a's and b's that at some point contain a double letter. We can think of it as

(arbitrary)(double letter)(arbitrary)

Example are: Λ a *b ab ba aba bab abab baba* …. The expression (ab)* covers all of these except those that begin with b or end in a. Adding these choices gives us the regular expression

(Λ+ b) (ab)* (Λ+ a)

REGULAR EXPRESSIONS

EXAMPLE

Consider the regular expression below:

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

= (arbitrary) a (arbitrary) [a or nothing] (arbitrary) a (arbitrary).

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

Before we analyze the second term let us make the observation that

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

REGULAR EXPRESSIONS

which occurs in the middle of the second term is only another way of saying "any string whatsoever" and could be replaced with the more direct expression

$$(a + b)^*$$

This would reduce the second term of the expression to

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

which we have already seen is a regular expression representing all words that have at least two a's in them. Therefore, the language associated with E is the union of all strings that have three or more a's with all strings that have two or more a's. But since all strings with three or more a's are themselves already strings with two or more a's, this whole language is just the second set alone.

REGULAR EXPRESSIONS

Example:- It is possible by repeated application of the rules for forming regular expressions to produce an expression in which the star operator is applied to a sub expression that already has a star in it. Some examples are:

$$(a + b^*)^* \; (aa + ab^*)^* \; ((a + bbba^*) + ba^*b)^*$$

In the first of these expressions, the internal * adds nothing to the language

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

$$(a^*)^* = a^*$$

However,

$$(aa + ab^*)^* \; \# \; (aa + ab)^*$$

REGULAR EXPRESSIONS

EXAMPLE

Consider the regular expression:

$$(a^*b^*)^*$$

The language defined by this expression is all strings that can be made up of factors of the form a*b*, but since both the single letter a and the single letter b are words of the form a*b*, this language contains all strings of a's and b's. It cannot contain more than everything, so

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

REGULAR EXPRESSIONS

EXAMPLE

One very interesting example, which we consider now in great detail is

$$E = [aa + bb + (ab+ba)(aa+bb)^*(ab+ba)]^*$$

This regular expression represents the collection of all words that are made up of "syllables" of three types:

type1 = aa
type2 = bb
type3 = (ab + ba)(aa + bb)*(ab + *ba)*

$$E = [\text{type}_1 + \text{type2} + \text{type 3}]^*$$

REGULAR EXPRESSIONS

EXAMPLE

Consider the language defined by the regular expression:

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

This is the language of all words without a double a.

"لا تُسيء اللّفظ وإن
ضَاق عليكَ الجَواب"

- الإمام علي بن ابي طالب