CSCI 340: Computational Models

Regular Expressions

Chapter 4

Department of Computer Science
Yet Another New Method for Defining Languages

Given the Language:

\[ L_1 = \{ x^n \text{ for } n = 1 \ 2 \ 3 \ \ldots \} \]

We could easily change the sequence for \( n \):

\[ L_2 = \{ x^n \text{ for } n = 1 \ 3 \ 5 \ 7 \ \ldots \} \]

But if we change the sequence for \( n \) it can be difficult:

\[ L_3 = \{ x^n \text{ for } n = 1 \ 4 \ 9 \ 16 \ \ldots \} \]

Or just unwieldy / non-definitive:

\[ L_3 = \{ x^n \text{ for } n = 3 \ 4 \ 8 \ 22 \ \ldots \} \]

We need a notation for something more precise than the ellipsis
Reappearance of Kleene Star

Reconsider the language from Chapter 2:

\[ L_4 = \{ \lambda, x, xx, xxx, xxxx, \ldots \} \]

We presented one method for indicating this set as a closure:

Let \( S = \{ x \} \). Then \( L_4 = S^* \)

Or in shorthand:

\[ L_4 = \{ x \}^* \]

Let’s now introduce a Kleene star applied to a letter rather than a set:

\[ x^* \]

We can think of the star as an unknown or undetermined power.
Defining Languages

- We should not confuse $x^*$ with $L_4$ as they are not equivalent.
- $L_4$ is semantically a language, $x^*$ is a language defining symbol.
- We can define a language as follows: $L_4 = \text{language}(x^*)$.

Example

$$
\Sigma = \{a\ b\} \\
L = \{a\ ab\ abb\ abbb\ abbbb\ \ldots\} \\
L = \text{language}(a\ b^*) \\
L = \text{language}(ab^*)
$$

Note: the Kleene star is applied to the letter immediately preceding
Applying Kleene Star to an Entire String

- Closure to entire substrings requires forced precedence
- We can accomplish this by grouping with parentheses
- For example: (ab)* = λ or ab or abab or ababab...

We can also use + to represent one-or-more

**Theorem**

\[ xx^* = x^+ \]

**Proof.**

\[ L_1 = \text{language}(xx^*) \quad L_2 = \text{language}(x^+) \]

\[ \text{language}(x^*) = \lambda \; x \; xx \; xxx \; \ldots \]

\[ \text{language}(x \; x^*) = x\lambda \; xx \; xxx \; xxxx \; \ldots \]

\[ \text{language}(xx^*) = x \; xx \; xxx \; xxxx \; \ldots \]

\[ \text{language}(xx^*) = \text{language}(x^+) = x \; xx \; xxx \; xxxx \; \ldots \]
Language Examples

Example

The language $L_1$ can be defined by any of the expressions below:

$$xx^* \quad x^+ \quad xx^*x^* \quad x^*xx^* \quad x^+x^* \quad x^*x^*x^*xx^*$$

Remember: $x^*$ can always be $\lambda$

Example

The language defined by the expression

$$ab^*a$$

is the set of all strings of $a$’s and $b$’s that have at least two letters that

1. start and end with $a$
2. only have $b$’s in between
## Language Examples

### Example

The language of the expression

\[ a^* b^* \]

contains all of the strings of a’s and b’s in which all the a’s (if any) come before all the b’s (if any)

\[
\text{language}(a^* b^*) = \{ \lambda, a, b, aa, ab, bb, aaa, aab, abb, bbb, aaaa, \ldots \}
\]

### Note

It is *very* important to note that

\[ a^* b^* \neq (ab)^* \]
Language Examples

Example

Consider the language $T$ defined over the alphabet $\Sigma = \{a, b, c\}$

$$T = \{a\ c\ ab\ cb\ abb\ cbb\ abbb\ cbbb\ abbbb\ cbbbb\ \ldots\}$$

We may formally define the language as follows:

$$T = \text{language}((a + c)b)$$

Or in English as:

$$T = \text{language(either } a \text{ or } c \text{ followed by some } b\text{’s})$$

Note: parens force precedence change: selection before concatenation
Consider the language \( L \) defined over the alphabet \( \Sigma = \{a, b\} \)

\[
L = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}
\]

• What is the pattern?
• How can we write a language expression for this?
• How can we generalize this?
• How can we represent “choose any single character” from \( \Sigma \)?
**Regular Expressions**

*Regular Language* — a language which can be expressed as a regular expression

**Definition for Regular Expression**

1. Every letter of $\Sigma$ can be made into a regular expression. $\lambda$ is a regular expression.
2. If $r_1$ and $r_2$ are regular expressions, then so are:
   - (i) $(r_1)$
   - (ii) $r_1r_2$
   - (iii) $r_1 + r_2$
   - (iv) $(r_1^*)$
3. Nothing else is a regular expression

**Note:** we could add $r_1^+$ but we can rewrite it as $r_1r_1^*$
Defining Some Regular Expressions

Chalkboard Problems

1. All words that begin with an $a$ and end with a $b$
2. All words that contain exactly two $a$’s
3. All words that contain exactly two $a$’s and start with $b$
4. All words that contain two or more $a$’s
5. All words that contain two or more $a$’s that end in $b$
6. All words of length 3 or higher which contain two $a$’s in a row
A More Complicated Example

Language of all words that have at least one $a$ and one $b$

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

which can also be expressed as

$$<\text{arbitrary}> \ a \ <\text{arbitrary}> \ b \ <\text{arbitrary}>$$

This mandates that $a$ must be found before $b$. The unhandled case can be matched with:

$$bb^*aa^*$$

One of these must be true for our expression to be matched:

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

Consider from the last slide

\[(a + b)^*a(a + b)^*b(a + b)^* + bb^*aa^*\]

If we wanted to include strings of all \(a\)'s or \(b\)'s we would use:

\[a^* + b^*\]

This would mean that we could define a regular expression which accepts any sequence of \(a\)'s and \(b\)'s:

\[(a + b)^*a(a + b)^*b(a + b)^* + bb^*aa^* + a^* + b^*\]

but this is simply just

\[(a + b)^*\]

These are not obviously equivalent
Algebraic Equivalence Need Not Apply

An Analysis of \((a + b)^*\)

\[(a + b)^* = (a + b)^* + (a + b)^*\]
\[(a + b)^* = (a + b)^*(a + b)^*\]
\[(a + b)^* = a(a + b)^* + b(a + b)^* + \lambda\]
\[(a + b)^* = (a + b)^*ab(a + b)^* + b^*a^*\]

All of these are equal — O_o
Some Algebra Works!

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 $\lambda$.

$$V = \{\lambda\ a\ b\ ab\ bb\ abb\ bbb\ abbb\ bbbb\ \ldots\}$$

We can then define $V$ by the expression:

$$b^* + ab^*$$

Where $\lambda$ is embedded into the term $b^*$. Alternatively, we could define $V$ by the expression

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

This gives us an option of having a $a$ or nothing! Since we could always write $b^* = \lambda b^*$, we demonstrate the distributive property

$$\lambda b^* + ab^* = (\lambda + a)b^*$$
_concatenation

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

$$ST = \{ \text{all combinations of all string } S \text{ followed with a string from } T \}$$

Example

$$S = \{a \ a a \ a a a\} \quad T = \{b b \ b b b\}$$

$$ST = \{a b b \ a b b b \ a a b b \ a a b b b \ a a a b b \ a a a b b b\}$$

Rewritten as a Regular Expression

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

$$= a b b + a b b b + a a b b + a a b b b + a a a b b + a a a b b b$$
### Concatenation

#### 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

$$ST = \{ \text{all combinations of all string } S \text{ followed with a string from } T \}$$

#### Example

- $S = \{a\ bb\ bab\}$
- $T = \{a\ ab\}$
- $ST = \{aa\ aab\ bba\ bbab\ baba\ babab\}$

#### Rewritten as a Regular Expression

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

What are the regular expressions for the concatenation of the two sets in each example? Give both the simple and “distributed” forms.

**Example**

\[ P = \{a\ bb\ bab\} \]
\[ Q = \{\lambda\ bbb\} \]

**Example**

\[ M = \{\lambda\ x\ xx\} \]
\[ N = \{\lambda\ y\ yy\ yyy\ yyyyy\ \ldots\} \]
Associating a Language with Every RE

The rules below define the **language associated** with any RE:

1. The language associated with the regular expression that is just a single letter is that one-letter word alone and the language associated with $\lambda$ is just $\{ \lambda \}$, a one-word language.

2. If $r_1$ is a regular expression associated with language $L_1$ and $r_2$ is a regular expression associated with the language $L_2$ then:
   - i. RE $(r_1)(r_2)$ is associated with $L_1 \times L_2$
     \[
     \text{language}(r_1r_2) = L_1L_2
     \]
   - ii. RE $r_1 + r_2$ is associated with $L_1 \cup L_2$
     \[
     \text{language}(r_1 + r_2) = L_1 + L_2
     \]
   - iii. RE $r_1^*$ is $L_1^*$ (the Kleene closure)
     \[
     \text{language}(r_1^*) = L_1^*
     \]
Exprsing a Finite Language as RE

Theorem

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

Proof.

To make one RE that defines the language $L$, turn all the words in $L$ into **boldface** type and stick pluses between them. Violá. For example, the RE defining the language

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

is

$aa + ab + ba + bb \ OR \ (a + b)(a + b)$

The reason this “trick” only works for *finite* languages is that an infinite language would yield an infinitely-long regular expression (which is forbidden). □
EVEN-EVEN

\[ E = \left[ aa + bb + (ab + ba)(aa + bb)^* (ab + ba) \right] \]

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

- \( \text{type}_1 = \text{aa} \)
- \( \text{type}_2 = \text{bb} \)
- \( \text{type}_3 = (ab + ba)(aa + bb)^* (ab + ba) \)

\[ E = \left[ \text{type}_1 + \text{type}_2 + \text{type}_3 \right] \]

**Question 1**

What does this Regular Expression “do”?

**Question 2**

What are the first 12 strings matched by this RE?
Homework 2a

1. For each of the problems below, give a regular expression which only accepts the following. Assume $\Sigma = \{a, b\}$
   1. All strings that begin and end with the same letter
   2. All strings in which the total number of $a$'s is divisible by 3
   3. All strings that end in a double letter

2. Show the following pairs of regular expressions define the same language
   1. $(ab)^*a$ and $a(ba)^*$
   2. $(a^*bbb)^*a^*$ and $a^*(bbba)^*$

3. Describe (in English phrases) the languages associated with the following regular expressions
   1. $(a + b)^*a(\lambda + bbbb)$
   2. $(a(aa)^*b(bb)^*)^*$
   3. $((a + b)a)^*$