CSCI 340: Computational Models

Regular Expressions

Chapter 4

Department of Computer Science

Yet Another New Method for Defining Languages

Given the Language:

$$L_1 = \{x^n \text{ for } n = 1 \;\; 2 \;\; 3 \quad \ldots\}$$

We could easily change the sequence for *n*:

$$L_2 = \{x^n \text{ for } n = 1 \;\; 3 \;\; 5 \;\; 7 \quad \ldots\}$$

But if we change the sequence for *n* it can be difficult:

$$L_3 = \{x^n \text{ for } n = 1 \;\; 4 \;\; 9 \;\; 16 \quad \ldots\}$$

Or just unwieldy / non-definitive:

$$L_3 = \{x^n \text{ for } n = 3 \;\; 4 \;\; 8 \;\; 22 \quad \ldots\}$$

We need a notation for something **more precise than the ellipsis**

Reappearance of Kleene Star

Reconsider the language from Chapter 2:

$$L_4 = \{\lambda \quad x \quad xx \quad xxx \quad xxxx \quad \ldots\}$$

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

$$\text{Let } S = \{x\}. \text{ Then } L_4 = S^*$$

Or in shorthand:

$$L_4 = \{x\}^*$$

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

$$\mathbf{x}^*$$

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

Defining Languages

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

Example

$$\Sigma = \{a\ b\}$$

$$L = \{a\ ab\ abb\ abbb\ abbbb\ \ldots\}$$

$$L = \text{language}(\mathbf{a} \quad \mathbf{b}^*)$$

$$L = \text{language}(\mathbf{ab}^*)$$

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: $(\mathbf{ab})^* \;=\; \lambda \quad \text{or} \quad ab \quad \text{or} \quad abab \quad \text{or} \quad ababab \ldots$

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

Theorem

$\mathbf{xx}^* = \mathbf{x}^+$

Proof.

$L_1 = \text{language}(\mathbf{xx}^*) \qquad L_2 = \text{language}(\mathbf{x}^+)$
$\text{language}(\mathbf{x}^*) = \lambda \quad x \quad xx \quad xxx \quad \ldots$
$\text{language}(\mathbf{x} \quad \mathbf{x}^*) = x\lambda \quad xx \quad xxx \quad xxxx \quad \ldots$
$\text{language}(\mathbf{xx}^*) = x \quad xx \quad xxx \quad xxxx \quad \ldots$
$\text{language}(\mathbf{xx}^*) = \text{language}(\mathbf{x}^+) = x \quad xx \quad xxx \quad xxxx \quad \ldots$ $\square$

Language Examples

Example

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

$$\mathbf{xx}^* \qquad \mathbf{x}^+ \qquad \mathbf{xx}^*\mathbf{x}^* \qquad \mathbf{x}^*\mathbf{xx}^* \qquad \mathbf{x}^+\mathbf{x}^* \qquad \mathbf{x}^*\mathbf{x}^*\mathbf{x}^*\mathbf{xx}^*$$

Remember: $\mathbf{x}^*$ can always be λ

Example

The language defined by the expression

$$\mathbf{ab}^*\mathbf{a}$$

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

$$\mathbf{a}^*\mathbf{b}^*$$

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)

$$\text{language}(\mathbf{a}^*\mathbf{b}^*) = \{\lambda \quad a \quad b \quad aa \quad ab \quad bb \quad aaa \quad aab \quad abb \quad bbb \quad aaaa \quad \ldots$$

Note

It is *very* important to note that

$$\mathbf{a}^*\mathbf{b}^* \neq (\mathbf{ab})^*$$

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:

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

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*

Language Examples

Example

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 Σ?

Regular Expressions

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

Definition for Regular Expression

1. Every letter of Σ can be made into a regular expression. λ is a regular expression.
2. If $\mathbf{r}_1$ and $\mathbf{r}_2$ are regular expressions, then so are:
   1. $(\mathbf{r}_1)$
   2. $\mathbf{r}_1\mathbf{r}_2$
   3. $\mathbf{r}_1 + \mathbf{r}_2$
   4. $(\mathbf{r}_1{}^*)$
3. Nothing else is a regular expression

Note: we could add $\mathbf{r}_1{}^+$ but we can rewrite it as $\mathbf{r}_1\mathbf{r}_1{}^*$

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*

$$(\mathbf{a}+\mathbf{b})^*\mathbf{a}(\mathbf{a}+\mathbf{b})^*\mathbf{b}(\mathbf{a}+\mathbf{b})^*$$

which can also be expressed as

<arbitrary> **a** *<arbitrary>* **b** *<arbitrary>*

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

$$\mathbf{bb}^*\mathbf{aa}^*$$

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

$$(\mathbf{a}+\mathbf{b})^*\mathbf{a}(\mathbf{a}+\mathbf{b})^*\mathbf{b}(\mathbf{a}+\mathbf{b})^* + \mathbf{bb}^*\mathbf{aa}^*$$

Confusing Equivalences

Consider from the last slide

$$(\mathbf{a}+\mathbf{b})^*\mathbf{a}(\mathbf{a}+\mathbf{b})^*\mathbf{b}(\mathbf{a}+\mathbf{b})^* + \mathbf{bb}^*\mathbf{aa}^*$$

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

$$\mathbf{a}^* + \mathbf{b}^*$$

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

$$(\mathbf{a}+\mathbf{b})^*\mathbf{a}(\mathbf{a}+\mathbf{b})^*\mathbf{b}(\mathbf{a}+\mathbf{b})^* + \mathbf{bb}^*\mathbf{aa}^* + \mathbf{a}^* + \mathbf{b}^*$$

but this is simply just

$$(\mathbf{a}+\mathbf{b})^*$$

These are not obviously equivalent

Algebraic Equivalence Need Not Apply

An Analysis of $(\mathbf{a}+\mathbf{b})^*$

$$(\mathbf{a}+\mathbf{b})^* = (\mathbf{a}+\mathbf{b})^* + (\mathbf{a}+\mathbf{b})^*$$
$$(\mathbf{a}+\mathbf{b})^* = (\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{b})^*$$
$$(\mathbf{a}+\mathbf{b})^* = \mathbf{a}(\mathbf{a}+\mathbf{b})^* + \mathbf{b}(\mathbf{a}+\mathbf{b})^* + \lambda$$
$$(\mathbf{a}+\mathbf{b})^* = (\mathbf{a}+\mathbf{b})^*\mathbf{ab}(\mathbf{a}+\mathbf{b})^* + \mathbf{b}^*\mathbf{a}^*$$

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 λ.

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

We can then define V by the expression:

$$\mathbf{b}^* + \mathbf{ab}^*$$

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

$$(\lambda + \mathbf{a})\mathbf{b}^*$$

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

$$\lambda\mathbf{b}^* + \mathbf{ab}^* = (\lambda + \mathbf{a})\mathbf{b}^*$$

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 = { all combinations of all string S followed with a string from T }

Example

$$S = \{a \quad aa \quad aaa\} \qquad T = \{bb \quad bbb\}$$

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

Rewritten as a Regular Expression

$$(\mathbf{a} + \mathbf{aa} + \mathbf{aaa})(\mathbf{bb} + \mathbf{bbb})$$

$$=$$

$$\mathbf{abb} + \mathbf{abbb} + \mathbf{aabb} + \mathbf{aabbb} + \mathbf{aaabb} + \mathbf{aaabbb}$$

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 = { all combinations of all string S followed with a string from T }

Example

$$S = \{a \quad bb \quad bab\} \qquad T = \{a \quad ab\}$$

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

Rewritten as a Regular Expression

$$(\mathbf{a} + \mathbf{bb} + \mathbf{bab})(\mathbf{a} + \mathbf{ab})$$

$$=$$

$$\mathbf{aa} + \mathbf{aab} + \mathbf{bba} + \mathbf{bbab} + \mathbf{baba} + \mathbf{babab}$$

Concatenation

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

Example

$$P = \{a \;\; bb \;\; bab\}$$

$$Q = \{\lambda \;\; bbbb\}$$

Example

$$M = \{\lambda \;\; x \;\; xx\}$$

$$N = \{\lambda \;\; y \;\; yy \;\; yyy \;\; yyyy \;\; \ldots\}$$

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 λ is just $\{ \lambda \}$, a one-word language
2. If $\mathbf{r}_1$ is a regular expression associated with language L_1 and $\mathbf{r}_2$ is a regular expression associated with the language L_2 then
   i. RE $(\mathbf{r}_1)(\mathbf{r}_2)$ is associated with $L_1 \times L_2$

$$\text{language}(\mathbf{r}_1\mathbf{r}_2) = L_1 L_2$$

   ii. RE $\mathbf{r}_1 + \mathbf{r}_2$ is associated with $L_1 \cup L_2$

$$\text{language}(\mathbf{r}_1 + \mathbf{r}_2) = L_1 + L_2$$

   iii. RE $\mathbf{r_1}^*$ is $L_1{}^*$ (the Kleene closure)

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

Expressing 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

$$\mathbf{aa} + \mathbf{ab} + \mathbf{ba} + \mathbf{bb} \qquad \text{OR} \qquad (\mathbf{a} + \mathbf{b})(\mathbf{a} + \mathbf{b})$$

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

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

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

$$\begin{aligned}
\text{type}_1 &= \mathbf{aa} \\
\text{type}_2 &= \mathbf{bb} \\
\text{type}_3 &= (\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^{*}(\mathbf{ab} + \mathbf{ba}) \\
E &= \left[\text{type}_1 + \text{type}_2 + \text{type}_3\right]
\end{aligned}$$

Question 1

What does this Regular Expression "do" ?

Question 2

What are the first 12 strings matched by this RE?