Chapter 4: Regular Expressions

What are the languages with a finite representation?

We start with a simple and interesting class of such languages.

A new method to define languages

- alphabet → language

  $S = \{x\}$ $S^* = \{\Lambda, x, xx, xxx, \ldots\}$

  or directly $\{x\}^* = \{\Lambda, x, xx, xxx, \ldots\}$

- language → language

  $S = \{xx, xxx\}$ $S^* = \{\Lambda, xx, xxx, xxxx, \ldots\}$

  or directly $\{xx, xxx\}^* = \{\Lambda, xx, xxx, xxxx, \ldots\}$

- "letter" → language

  x* (written in bold)

  language(**x***) = $\{\Lambda, x, xx, xxx, \ldots\}$

  or informally **x*** = $\{\Lambda, x, xx, xxx, \ldots\}$

- L1 = {a, ab, abb, abbb, …} or simply (**ab***)
- L2 = {Λ, ab, abab, ababab, …} or simply (**ab**)*

Several ways to express the same language

- {x, xx, xxx, xxxx, …}

  xx* **x**$^+$ **xx*x*** **x*xx*** **(x**$^+$**)x*** **x*(x**$^+$**)** **x*x*xx***

- L3= {Λ, a, b, aa, ab, bb, aaa, aab, abb, bbb, aaaa, …} or simply (**a*b***)

  (a's before b's)

Remark: language(**a*b***) ≠ language(**(ab)***)

Example: S-ODD

- Rule 1: $x \in$ S-ODD
- Rule 2: If w is in S-ODD then xxw is in S-ODD

- S-ODD = language(**x(xx)***)
- S-ODD = language(**(xx)*x**)
- But not: S-ODD = language(**x*xx***)

xx|x|x

- A useful symbol to simplify the writing:
  - **x + y**   choose either x or y

- Example:

  S = {a, b, c}

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

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

(defines the language whose words are constructed from either a or c followed by some b's)

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

  all words of exactly three letters from the alphabet {a, b}

  L = **(a+b)(a+b)(a+b)**
- **(a+b)*** all words formed from alphabet {a,b}
- **a(a+b)*** = ?
- **a(a+b)*b** = ?

- <u>Definition:</u> Given an alphabet S, the set of regular expressions is defined by the following rules.
  1. For every letter in S, the letter written in bold is 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. $\mathbf{r_1\,r_2}$
     3. $\mathbf{r_1+r_2}$
     4. $\mathbf{r_1{}^*}$
  3. Nothing else is a regular expression.

- Remark: **Notice that $r_1^+ = r_1 r_1^*$**
- $\mathbf{r_1}=\mathbf{r_2}$ if and only if language($\mathbf{r_1}$) = language($\mathbf{r_2}$)
- Example: **(a+b)*a(a+b)***

  All words that have at least one a.

abbaab: (Λ)a(bbaab) (abb)a(ab) (abba)a(b)

- Words with no a's? **b***
- All words formed from {a,b}?

  (a+b)*a(a+b)* + b*

Thus: **(a+b)* = (a+b)*a(a+b)* + b***

- Example: The language of all words that have at least two a's.

  (a+b)*a(a+b)*a(a+b)*

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

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

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

- Example: The language of all words that have exactly two a's.

  b*ab*ab*

Another Example: At least one a and one b?

- First solution:

  (a+b)*a(a+b)*b(a+b)* +   (a+b)*b(a+b)*a(a+b)*

- But **(a+b)*a(a+b)*b(a+b)*** expresses all words except words of the form some b's (at least one) followed by some a's (at least one).   **bb*aa***

- Second solution:

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

- Thus: **(a+b)*a(a+b)*b(a+b)* +   (a+b)*b(a+b)*a(a+b)***

  = (a+b)*a(a+b)*b(a+b)* +   bb*aa*

- The only words that do not contain both an a and b in them are the words formed from all a’s or all b’s:

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

- Thus:

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

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

- <u>Example:</u> The language of all words formed from some b's (possibly 0) and all words where an a is followed by some b's (possibly 0):

  $\{\Lambda, a, b, ab, bb, abb, bbb, abbb, bbbb, \ldots\}$

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

- In general: concatenation is distributive over the + operation.

  $$\mathbf{r_1(r_2 + r_3) = r_1r_2 + r_1r_3}$$

  $$\mathbf{(r_1 + r_2)\, r_3 = r_1r_3 + r_2r_3}$$

- <u>Example of the distributivity rule:</u> **(a+c)b* = ab*+cb***

- 2 operations: language(s) → language

  If S and T are two languages from the same alphabet S,

  1. S+T: the union of languages S and T defined as $S \cup T$
  2. ST: the product set is the set of words x written vw with v a word in S and w a word in T.

- <u>Example:</u> S = {a, bb} T = {a, ab}

  ST = {aa, aab, bba, bbab}

Language associated with a regular expression is defined by the following rules.

1. The language associated with a regular expression that is just a single letter is that one-letter word alone. The language associated with $\mathbf{\Lambda}$ is $\{\Lambda\}$.
2. If L_1 is the language associated with the regular expression $\mathbf{r_1}$ et L_2 is the language associated with the regular expression $\mathbf{r_2}$:

   (i) The product L_1L_2 is the language associated with the regular expression $\mathbf{r_1r_2}$, that is: language($\mathbf{r_1r_2}$) = L_1L_2

   (ii) The union L_1+L_2 is the language associated with the regular expression $\mathbf{r_1}+\mathbf{r_2}$, that is: language($\mathbf{r}+\mathbf{r_2}$) = L_1+L_2

   (iii) The Kleene closure of L_1, written L_1^*, is the language associated with the regular expression $\mathbf{r_1^*}$, that is language($\mathbf{r_1^*}$) = L_1^*

- Remark: For all regular expressions, there is some language associated with it.

- ## Finite Languages are Regular
- Let L be a finite language. There is a regular expression that defines it.
- Algorithm (and proof)

  Write each letter in L in bold, and write a + between regular expressions

Example: L = {baa, abbba, bababa}

baa + abbba + bababa

- The regular expression that is defined by this algorithm is not necessarily unique.

  Example: L = {aa, ab, ba, bb}

  aa + ab + ba + bb or **(a+b)(a+b)**

- Remark: This algorithm does not work for infinite languages. Regular expressions must be finite, even if the language defined is infinite.

- Kleene star applied to a subexpression with a star

  (a+b*)* **(aa+ab*)***

  (a+b*)* = (a+b)* **(aa+ab*)* ≠ (aa+ab)*** abb|abb

- **(a*b*)***

  The letter a and the letter b are in language(**a*b*).**

  (a*b*)* = (a+b)*

- Is it possible to determine if two regular expressions are equivalent?
  - With a set of algebraic rules? Unknown.
  - With an algorithm? Yes.

- Examples
  - Words with a double letter: $(a+b)^*(aa+bb)(a+b)^*$
  - Words without a double letter: $(ab)^*$

    But not words that begin with b or end with a:

    $$(\Lambda+b)(ab)^*(\Lambda+a)$$
  - $(a+b)^*(aa+bb)(a+b)^* \quad + \quad (\Lambda+b)(ab)^*(\Lambda+a)$

Language EVEN-EVEN defined by the expression:

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

Every word in EVEN-EVEN has an even number of a’s and b’s.

Every word that contains an even number of a’s and b’s is a member of EVEN-EVEN.