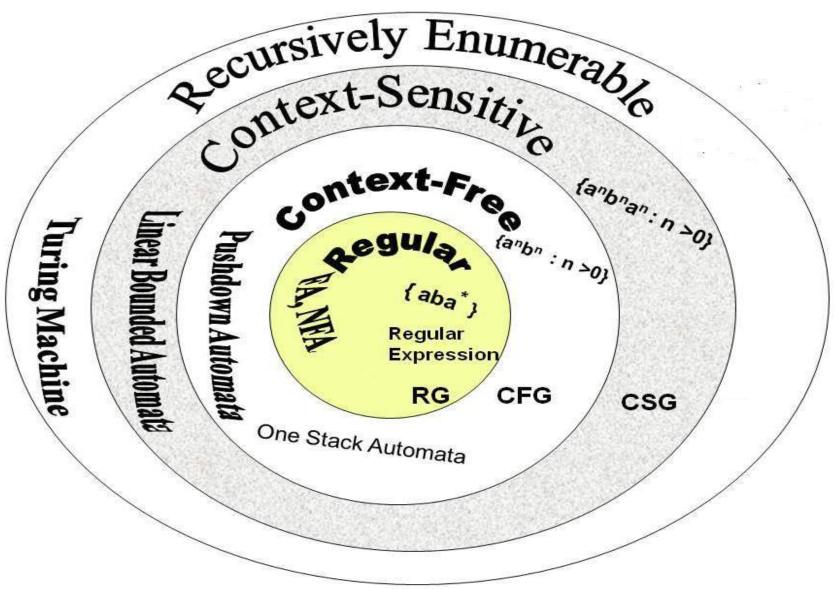
LANGUAGES

Languages

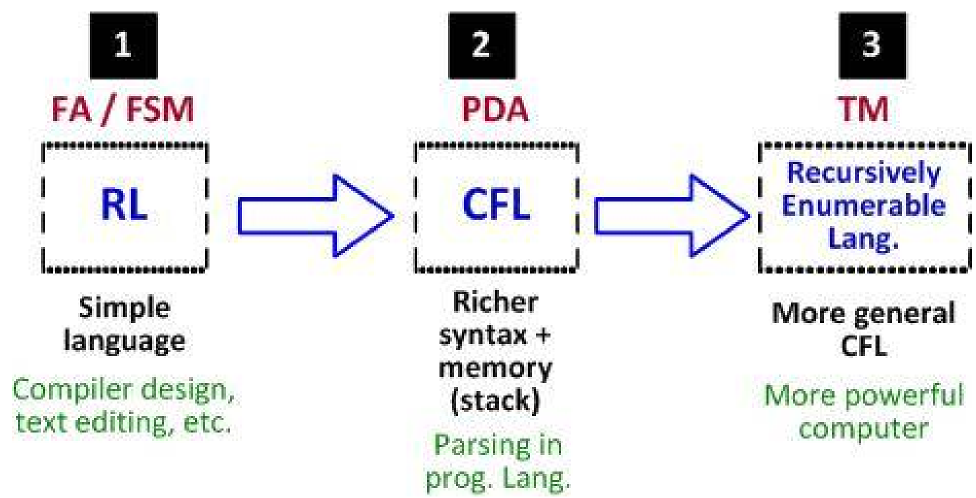
Contents:

1. Strings and Languages
2. Finite Specification of Languages
3. Regular Sets and Expressions

In a nutshell...



Languages and Model of computations



Formal Specification of Languages

- Generators
  - Grammars
    - Context---free
    - Regular
  - Regular Expressions
- Recognizers
  - Push---down Automata
    - Context Free Grammar
  - Finite State Automata
    - Regular Grammar
- A Finite Automata is:
  - a mechanism to recognize a set of valid inputs before carrying out an acKon.
  - a notation for describing a family of language recognition algorithms.

Strings and Languages

- Natural languages, computer languages, Mathematical languages
- A **language** is a set of **strings** over an **alphabet**.
- Syntax of the language: certain Properties that must be satisfied by strings.

Strings and Alphabets

- **Symbols** / Letters / Characters

  A single element of the alphabets that has a unique meaning, i.e.: symbol A and B which have different meanings.
- **Alphabet** (denoted by Σ in italic capital Letters)

  A finite set of symbols – indivisible objects.
- **String** / Word

  A finite sequence of symbols from alphabets.

Strings and Alphabets

- E.g.: C = {a, b, c, 1, 2, 3}
- **Alphabet** C with 6 units of **symbols**

  An example of a word / string from the alphabet C : acca, baca, 132, a12, etc.
- **String** acca and caac have different meanings.
- **String** acca, 121, abba are palindromes.

Strings and Alphabets

- For **alphabet** Σ :

  Σ^* is the set of all **strings** over Σ.

  Σ^n is the set of all strings of length n.

- A ***language*** over Σ is a set $L \subseteq \Sigma^*$.

  E.g.: $L = \{1, 01, 11, 001\}$ is a language over $\{0, 1\}$.

Length of String

- If w is a string over Σ, the **length of w**, Written **|w|**, is the number of symbols that it contains.

E.g.:

$\lvert\lambda\rvert$	=	0
$\lvert 0\rvert$	=	1
$\lvert 1\rvert$	=	1
$\lvert 1010\rvert$	=	4
$\lvert 001101\rvert$	=	6

Length of String (cont.)

- The set of strings, $\Sigma^* = \{a, b, c\}$ includes:

length 0:	λ
length 1:	a b c
length 2:	aa ab ac ba bb bc ca cb cc
length 3:	aaa aab aac aba abb abc aca acb acc baa bab bac bba bbb bbc bca bcb bcc caa cab cac cba cbb cbc cca ccb ccc
...	...

Concatenation of String

- If we have string x of length m and string y of length n, the **concatenation** of x and y Written xy is $x_1 \ldots x_m y_1 \ldots y_n$.

- E.g.:

	variable		string
E.g.:	x	=	*aba*
	y	=	*bbbab*
Then	xy	=	*ababbbab*
	yx	=	*bbbababa*
	xyx	=	*ababbbababa*

Self Concatenation

- If we have string x, the concatenation of x and x is self concatenation
  - E.g.: $x^0 = \lambda$
    $x^1 = x = aba$
    $x^2 = xx = abaaba$
    $x^3 = xxx = abaabaaba$
- $x^k = xx...x$: self-concatenated string *k* times

Substring

- A substring of a string is any sequence of consecutive symbols that appears in the string.
- Thus, substring is a subset of the symbols in a string.

E.g.: w = *abbaaababb*

bba is a substring of *w*
abab is a substring of *w*
baba is NOT a substring of *w*

Prefix

- A prefix of a string is any sequence of leading symbols of the string.
- A prefix can be seen as a special case of a substring.

E.g.: if w = *abbaaababb*

a is a prefix of w
abbaa is a prefix of w
bba is NOT a prefix of w

if w = *abaab*, *w* has 6 prefixes:

ε, *a*, *ab*, *aba*, *abaa* and *abaab.*

Suffix

- A suffix of a string is any sequence of trailing symbols of the string.
- A suffix also can be seen as a special case of a substring.

E.g.: if w = *abbaaababb*

abb is a suffix of w
babb is a suffix of w
bab is NOT a suffix of w

if *w* = *abaab*, *w* has 6 suffixes:

ε, *b*, *ab*, *aab*, *baab* and *abaab.*

Reverse

- The reverse of w, Written w^R or w^r is the string obtained by writing w in theopposite order.

E.g.:

if w = a w^R = a
if w = abb w^R = bba
if w = aba w^R = aba

if w = abbcd w^R = dcbba

Substring, Prefix and Suffix

- If *s* is a string and *s* = *tuv* for three strings *t*, *u*, and *v*, then *t* is a *prefix* of *s*, *v* is a *suffix* of *s*, and *u* is a *substring* of *s*. Because one or both of *t* and *u* might be λ, prefixes and suffixes are special cases of substrings.
- The string λ is a prefix of every string, a suffix of every string, and a substring of every string, and every string is a prefix, a suffix, and a substring of itself.

Finite Spec. of Languages

- Languages can be defined using 2 ways: either presented as
  - an alphabet and the exhaustive list of all valid words
  - an alphabet and a set of rules defining the acceptable words.

PALINDROME language

- E.g.: definiKon of a new language PALINDROME over the alphabet Σ = {a, b}
- Example --- (1) Exhaustive list :

PALINDROME = {ε, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, …}

- Example --- (2) Rule :

PALINDROME = {ε, and all string *x* such that reverse(*x*) = *x*}

Kleene Closure / Star

- Σ^* is the collection of all possible finite--- length strings generated from the strings in Σ.
- The definition of Kleene star on Σ is

$$\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^{i}{}_{i} = \{\lambda\} \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$

Kleene Closure / Star

- E.g.: if $\Sigma = \{x\}$, then

  $\Sigma^* = \{x\}^* = \{\varepsilon, x, xx, xxx\ldots\}$

  if $\Sigma = \{0, 1\}$, then

  $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
- Applied to set of strings:

  {ab, c}* = {λ, ab, c, abab, abc, cab, cc, ababab, ababc, abcab, abcc, cabab, cabc, ccab, ccc, ...}

Kleene Closure / Star

- Applied to set of characters:

  {a, b, c}* = {λ, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, ...}.
- Kleene star applied to the empty set:

$$\varnothing^* = \{\lambda\}.$$

Kleene star (cont.)

- We can think of the Kleene star as

an operation that makes an infinite language of strings of letter out of an alphabet.

Infinite language = infinitely many words, each of finite length.

DefiniKon of S*

- If S is a set of words, then by S*

  means the set of all finite strings formed by concatenating words from S,

  where any word may be used as frequently as we like, and where the null string is also included.

Example

- If S = {*aa*, *b*}, then

  S* = {λ plus any word composed of factors of *aa* and *b*}

  = {λ plus all strings of *a*'s and *b*'s in which the *a*'s occur in even clumps}

  = {λ, *b*, *aa*, *bb*, *aab*, *baa*, *bbb*, ...}

Example

- S* = {aa, b}*, where * = 0, 1, 2, 3, ...

  $\{aa, b\}^0 = \{\lambda\}$

  $\{aa, b\}^1 = \{aa, b\} = \{aa, b\}$

  $\{aa, b\}^2 = \{aa, b\}\,\{aa, b\} = \{aaaa, aab, baa, bb$

  $b\}^3 = \{aa, b\}\,\{aa, b\}\,\{aa, b\}$

  $= \{aaaaaa, aaaab, aabaa, aabb, baa$

  $baab, bbaa, bbb\}$

  ...

  $\{aa, b\}^* = \{\lambda, aa, b, aaaa, aab, baa, bb, aaaaaa, aaaabb,$

Length of String

- Thus, {*aa*, *b*}* = {λ, *aa*, *b*, *aaaa*, *aab*, *baa*, *bb*, *aaaaaa*, *aaaab*, *aabaa*, *aabb*, *baaaa*, *baab*, *bbaa*, *bbb*, ...}

length 0 :	λ
length 1 :	*b*
length 2 :	*aa*, *bb*
length 3 :	*aab*, *baa*, *bbb*
length 4 or more :	*aaaa*, *aaaaaa*, *aaaab*, *aabaa*, *aabb*, *baaaa*, *baab*, bbaa,
...	...

Example

- If S = {a, ab}, then

S* = {λ plus any word composed of factors of a and ab}

= {λ plus all strings of a's and b's except those that start with b and those that contain a double b}

= {λ, a, aa, ab, aaa, aab, aba, ...}

Example

- Consider the following languages

S = {a, b, ab} and T = {a, b, bb}

both S* and T* are languages of all strings of a's and b's since any string of a's and b's can be factored into syllables of either (a) or (b), both of which are in S and T.

Kleene Plus / PosiKve closure $^{+}$

- If we would like to refer to only the concatenation of some (not zero) strings from a set S, we use the notaKon + instead of *,
- E.g.: if $\Sigma = \{x\}$, then

$$\Sigma^{+} = \{x, xx, xxx, \ldots\}$$

- Kleene plus applied to the empty set:

$$\varnothing^{+} = \varnothing\varnothing^{*} = \{\} = \varnothing.$$

Regular Expressions

- Rules that define exactly the set of words that are valid in a formal language.

Regular Expressions

- Formally, the set of regular expressions can be defined by the following recursive rules:
  1. Every symbol of Σ is a regular expression
  2. ε is a regular expression
  3. if R1 and R2 are regular expressions, so are
     (R1) R1R2 R1 | R2 R1*
  4. Nothing else is a regular expression.

Regular Expressions

- Regular expressions are defined recursively
  a) Base case – simple regular expressions
  b) Recursive case – how to build more complex regular expressions from simple regular expressions

Formal definiKon of regular expressions

- A language is regular if it can be described by a regular expression.
- The Regular Languages is the set of all languages that can be represented by a regular expression
  – Set of set of strings
- Not every languages are able to be described by RE. Regular languages may also be described by another fine definiKons, besides the RE.

Regular Expression

- The symbols that appear in RE are
  – the Letters of the alphabet Σ
  – the symbol of null string ε or λ
  – parentheses ( )
  – star operator *
  – ∪ or + sign

$(a + b)^*$

$(a + b)^*$ $\quad * = 0, 1, 2, 3, \ldots$

$(a + b)^0 = \lambda$

$(a + b)^1 = (a + b) \quad = a, b$

$(a + b)^2 = (a + b)(a + b) \quad = aa, ab, ba, bb$

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

= aaa, aab, aba, abb, baa, bab, bba, bbb

$(a + b)^4 = (a + b)(a + b)(a + b)(a + b)$

= aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb, baaa, baab, baba, babb, bbaa, bbab, bbba, bbbb

Thus $(a + b)^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Regular Expressions

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

Regular Expressions

- When we design a regular expression, we need to imagine about the content of the string :
  - What the string will starts with
  - What in the middle
  - What the string will ends with

Regular Expressions

- $(a + b)b^*a$
  {aa, ba, aba, bba, abba, bbba, abbba, bbbba, . . .}
- $a(a + b)^*a$
  {aa, aaa, aba, aaaa, aaba, abaa, abba, . . .}
- a^*b^*
  {λ, a, b, aa, ab, bb, aaa, aab, abb, bbb, . . .}
- $(ab)^*$
  {λ, ab, abab, ababab, abababab, . . .}

Examples

- **ab***
- **(ab)***
- **(a*b*)**
- **What the difference?**
- **What is the shortest, IN and NOT IN?**

IN or NOT IN?

- **ab***
  IN a, ab, abb, abb, abbbb,
  NOT IN b, ba, aba, abab, bab, bbb, aab, baa, abba
- **(ab)***
  IN λ, ab, abab, ababab, abababab,
  NOT IN b, ba, bb, abb, baa, bba, bab, bbb
- **(a*b*)**
  IN λ, a, b, aa, ab, bb, aaa, aab, abb, bbb, aaaa
  NOT IN ba, bab, abab, bba, aba, bbba

Examples

- If Σ = {a, b}
- L_1 = all strings that begin and end with *aa*
  aa(a + b)*aa
- L_2 = all strings that begin or end with *aa*
  aa(a + b)* + (a + b)*aa
- L_3 = all strings that contain the substring *aa*
  (a + b)*aa(a + b)*
- L_4 = all strings that contain the substring *bb*
  (a + b)*bb(a + b)*
- L_5 = all strings that contain the substring *aa* or *bb* (a + b)*aa(a + b)* + (a + b)*bb(a + b)*

Examples

- If Σ = {a, b}
- L_1 = aa(a + b)*aa
- L_2 = aa(a + b)* + (a + b)*aa
- L_3 = (a + b)*aa(a + b)*
- L_4 = (a + b)*bb(a + b)*
- L_5 = (a + b)*aa(a + b)* + (a + b)*bb(a + b)*

Regular Expressions

- All strings over {a, b} that start with an a

  a (a + b)*
- All strings over {a, b} that are even in

  length ((a + b) (a + b))*
- All strings over {0, 1} that have an even number of 1's.

  0*(10*10*)*
- All strings over {a, b} that start and end with the

  same leter

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

Regular Expressions

- All strings over {*a, b, c*} that begin with *a*, contain exactly two *b*'s and end with *c.*

  a(*a* + *c*)**b*(*a* + *c*)**b*(*a* + *c*)**c*
- All strings over {0, 1} with no occurrences of 00

  1*(011*)*(0 + 1*)
- All strings over {0, 1} with exactly one occurrence of 00

  1*(011*)*00(11*0)*1*
- All strings over {0, 1} that contain 101

  (0 + 1)*101(0 + 1)*
- All strings over {0, 1} that do not contain 01

  1*0*

Regular Expressions

- We shall develop some new language--definiKon symbolism that will be much more precise than the ...

  E.g.:

  L_1 = {ε, x, xx, xxx, xxxx, ...}

  We can define it with closure

  Let $S = \{x\}$ Then $L_1 = S^*$

  or we can write $L_1 = \{x\}^*$

Language(x*)

- We can also define L_2 as

$$L_2 = (x^*)$$

Since x* is any string of x's, L_2 is then the set of all possible string of x's of any length (including λ)

Example

- Suppose we wish to describe the language L over the alphabet $\Sigma = \{a, b\}$ where $L = \{a, ab, abb, abbb, abbbb, ...\}$ "all words of the form one a followed by some number of b's (maybe no b's at all)"

  we may write $L = (ab^*)$

(ab)*

$(ab)^* = \varepsilon$ or ab or $abab$ or $ababab$ ...

- Parentheses are not Letters in the alphabet of this language, so they can be used to indicate factoring without accidentally changing the words.
- Like the powers in algebra

  ab^* means $a(b^*)$, not $(ab)^*$

(xx^*) vs. (x^+)

$L_1 = (xx^*)$

means ?

We start each word of L_1 by wriKng down an x and then we follow it with some string of x's (which may be no more x's at all.)

We can use the $^+$ notaKon and write

$L_1 = (x^+)$

$L_1 = (xx^*)$ and $L_1 = (x^+)$

- The language L_1 defined above can also be defined by any of these expressions:

xx^*	x^+	xx^*x^*	x^*xx^*
x^+x^*	x^*x^+	$x^*x^*x^*xx^*$	

Remember

x^* can always be λ

Example

ab*a

is the set of all string of a's and b's that have at least two Letters, that begin and end with a's, and that have nothing but b's inside.

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

Example

a*b*

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

(a*b*) = {ε, a, b, aa, ab, bb, aaa, aab, abb, ...}

noKce that

ba and aba are not in this language

a*b* vs. (ab)*

$$a^*b^* \neq (ab)^*$$

(ab)* can contain abab

but

a*b* can't contain abab

Regular OperaKons

- Let A and B be languages. We define the regular operaKons union, concatenation, and star as follows.
  - Union :

    $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
  - Concatenation : (simply no Written)

    $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
  - Star :

    $A^* = \{x_1x_2x_3 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Union (∪)

x ∪ y where x and y are strings of characters from an alphabet means "either x or y"

Also Written as x + y

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.

T = ((a ∪ c) b*)

= (either a or c then some b's)

Finite language L

- We can define any finite language by our new expression.
- E.g.: Consider a finite language L contains all the strings of a's and b's of length 3 exactly:

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

- The first letter can be either a or b. So do the 2nd and 3rd letter.

L = ((a ∪ b) (a ∪ b) (a ∪ b))

Finite language (cont.)

or we can simply write shortly as

$L = (a \cup b)^3$

if we write (a ∪ b)*, it means the set of all possible strings of Letters from the alphabet Σ = {a, b, c}

including the null string λ

Finite vs infinite language?

Examples

- If we write

$$a(a \cup b)^*$$

we can describe all words that begin with the letter a.

- If we would like to describe all words that begin with an a and end with b, we can define by the expression

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

Null Language

- ε or λ is the symbol of null string in regular expression.
- $\varnothing$ is the symbol for "Null Language"
- Don't confuse!
  - $R = \lambda$ represents the language containing a single string, the empty string. ⇨ $\{\lambda\}$
  - $R = \varnothing$ represents the language that doesn't contain any strings.

Example

- Let consider the language defined by

$$(a \cup b)^*a(a \cup b)^*$$

- What does it produce ?

The language of all words over the alphabet $\Sigma = \{a, b\}$ that have an *a* in somewhere. Only words which are not in this language are those that have only b's and the word ε .

Union of two languages

- Those words which compose of only b's are defined by the expression b*. (b* also includes the null string ε)
- Therefore, the language of all strings over the alphabet Σ = {a, b} are

all strings = (all strings with an a) ∪ (all string without an a)

(a ∪ b)* = (a ∪ b)*a(a ∪ b)* ∪ b*

Example

- How can we describe the language of all words that have at least two a's ?

(a ∪ b)*a(a ∪ b)*a(a ∪ b)*

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

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

Example

- Is there any other RE that can define the language with at least two a's ?

Yes. e.g.: b*ab*a(a ∪ b)*

= (some beginning of b's (if any))(the first a) (some middle of b's)(the second a) (some end)

Equivalent expressions

(a∪b)*a (a∪b)*a (a∪b)* = b*ab*a(a∪b)*

Both expressions are equivalent because they both describe the same item. We could write

((a∪b)*a (a∪b)*a (a∪b)*)

= (b*ab*a(a∪b)*)

= all words with at least two a's

= (a∪b)*ab*ab*

= b*a(a∪b)* ab*

Example

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

$$b^*ab^*ab^*$$

it can describes such words as

aab, baba, bbbabbbab, ...

Q: Can it make the word aab ?

A: Yes, by having the first and second $b^* = \lambda$

Example

- The language with at least one a and at least one b ?

$$(a \cup b)^*a(a \cup b)^*b(a \cup b)^*$$

It can only produce words which an a precede a b.
To produce words which have a b precede an a :

$$(a \cup b)^*b(a \cup b)^*a(a \cup b)^*$$

Thus, the set of all words :

$$(a \cup b)^*a\,(a \cup b)^*b\,(a \cup b)^* \cup (a \cup b)^*b\,(a \cup b)^*a\,(a \cup b)^*$$

Example

- $(a \cup b)^*a\,(a \cup b)^*b\,(a \cup b)^*$ can produce all words with at least one a and at least one b.
- However, it doesn't contain the words of the forms some b's followed by some a's.
- These excepKons are all defined by bb^*aa^*
- Thus, we have all strings over $\Sigma = \{a, b\}$

$$(a \cup b)^*a\,(a \cup b)^*b\,(a \cup b)^* \cup (a \cup b)^*b\,(a \cup b)^*a$$

$$(a \cup b)^* = (a \cup b)^*a\,(a \cup b)^*b\,(a \cup b)^* \cup bb^*aa^*$$

$(a \cup b)^*$

$$(a \cup b)^*a\,(a \cup b)^*b\,(a \cup b)^* \cup bb^*aa^*$$

- generates all words which have both a and b in them somewhere.
- Words which are not included in the above expression are words of all a's, all b's or ε ⇨⇨ a^*, b^*
- Now, we have all words which can be generated above the alphabet

$$(a \cup b)^* = (a \cup b)^*a\,(a \cup b)^*b\,(a \cup b)^* \cup bb^*aa^* \cup a^* \cup b^*$$

Note that:

- **Sets** $\varphi = \{\,\} \neq \{\lambda\}$
- **Set size** $|\{\,\}| = |\varphi| = 0$
- **Set size** $|\{\lambda\}| = 1$
- **String length** $|\lambda| = 0$

In class exercise

- For the alphabet {0,1} give RE for each language
  - i) All strings containing exactly two 0's
  - ii) All strings containing at least two 0's
  - iii) All strings containing 00 as substring
  - iv) All strings containing 00 as substring exactly once

RE vs Grammar vs TM

- **RE**

  Describes a set of **patterns** which form a language
- **Grammar**

  A set of **rule** describing a language
- **Turing machine**

  A **computational model** to recognize if a string is in a language

References

Α	α	alpha	a	"father"	Ν	ν	nu	n	
Β	β	beta	b		Ξ	ξ	xi	ks	"box"
Γ	γ	gamma	g		Ο	ο	omikron	o	"off"
Δ	δ	delta	d		Π	π	pi	p	
Ε	ε	epsilon	e	"end"	Ρ	ρ	rho	r	
Ζ	ζ	zêta	z		Σ	σ, ς	sigma	s	"say"
Η	η	êta	ê	"hey"	Τ	τ	tau	t	
Θ	θ	thêta	th	"thick"	Υ	υ	upsilon	u	"put"
Ι	ι	iota	i	"it"	Φ	ϕ	phi	f	
Κ	κ	kappa	k		Χ	χ	chi	ch	"Bach"
Λ	λ	lamda	l		Ψ	ψ	psi	ps	
Μ	μ	mu	m		Ω	ω	omega	ô	"grow"