BITS Pilani
Hyderabad Campus

Theory of Computation (CS F351)

Prof.R.Gururaj
CS&IS Dept.

Sets, Relations, and Languages (Ch.1 of T1)

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Sets, Relations, and Functions (Ch.1 of T1; Sec- 1.1 to 1.3)

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Sets

1. A Set is a collection of objects.
2. *L= {a, b, c, d}*
3. The objects comprising a set are called as *elements* or *members*.
4. The elements need not be related to each other.
5. L= { a, b, c, d}   we say that b belongs to L   and p does not belong to L
6. The order of the elements is not significant
7. We do not distinguish repetitions of the elements.

Hence set { 4, 3, 8} and { 3, 8, 3, 4, 8} are same.

1. A Set may contain other sets. Ex: {2, { b, k}, 8}

2. A set which has only one element is called a *singleton*. Ex: {1}

3. A set can have no elements and is called *empty set* Ø.

4. A set may be infinite. Ex: N= {0, 1, 2, …} *-three dots means list is infinite*

1. Describing sets with reference to other sets

   Ex: K= {1,3,9} G= {3,9}

   Now, G may be described as G= {x : x ϵ K and x is greater than 2}

2. If **P** is a property that the elements of a set **A** may have then we can describe a new set- B= {x : x ϵ A and x has the property P}

Subset - If each element in A is also there in B then we say that A subset of B

Proper subset-

If each element in A is also there in B and further A and B are not same then we say that A proper subset of B.

Note: An empty set is subset of every set. Ø is subset of every set.

Any set is subset of itself.

To prove that A and B are equal we must prove that

A is subset of B and B is subset of A then A=B

We represent set of all natural numbers as N and Z for set of all integers.

There are two forms to represent sets

1. Roster form- all elements listed by separating each by a comma and enclosed with in braces. { 2, 6, 9, 43}
2. Set builder form-

   set of all integers less than 40 and divided by 5

   {x : x is less than 40 and divided by 5}

the same can be written in roster form as {5, 10, 15, 20, 25, 30, 35}

Empty set G= {x : x is a cat and has wings}

Set operations

UNION A U B = { x: x ϵ A or x ϵ B}

INTERSECTION A ∩ B = { x: x ϵ A and x ϵ B}

DIFFERENCE A – B = { x: x ϵ A and x does not belong to B}

Laws of Set operations

Idempotent $A \cup A = A$ $A \cap A = A$

Commutative $A \cap B = B \cap A$ and $A \cup B = B \cup A$

Associative $(A \cup B) \cup C = A \cup (B \cup C)$

$(A \cap B) \cap C = A \cap (B \cap C$

Distributive $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$

$(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$

Absorption $(A \cup B) \cap A = A$

$(A \cap B) \cup A = A$

Demorgan's law $A-(B \cup C)= (A-B) \cap (A-C)$

$A- (B \cap C) = (A-B) \cup (A-C)$

Disjoint sets:

1. Two sets are disjoint if they have no common elements.
2. Hence the intersection of disjoint sets is empty.

To describe the union of More than two sets:

S= { {a,b}, {b,c}, {c,d}}

$\bigcup S$ = {a,b,c,d}

Powerset:

If A is a set, then collection of all subsets of A is called as power set of A.

A= {1, 2}   subsets are: Ø, {1}, {2}, {1,2}

2^A= { {1}, {2}, {1,2}, Ø}

Partition of a non-empty set:

1. Is a subset $\prod$ of 2^A such that $\emptyset$ is not an element of $\prod$ and each element of A is in only one set of $\prod$

Hence

1. Each element of $\prod$ is non-empty
2. Distinct members of $\prod$ are disjoint
3. $\bigcup_{\prod} = A$

Ex: A= {a, b, c, d}

Which of the following is partition of A

{{a,b}, {a,c}, {d}}

{{a,b}, {c}, {d}}

Relations

Ordered pair .

1. (a, b)
2. Here order matters (a, b) is not same as (b, a)
3. Two components of an ordered pair need not be different, but this is not possible in a set.
4. Two ordered pairs (a, b) and (c, d) are equal only when a=c and b=d.

Cartesian product of two sets

1. A= {1,3,9} B={a, b, c}
2. Then AXB = { (1,a), (1,b), (1,c),
   (3,a), (3,a), (3,a),
   (9,a), (9,a), (9,a) }
3. A X B is set of all pairs (a, b) where a ϵ A and b ϵ B
4. We have ordered 2-tuples (a, b)
   3-tuples (2, 5, 8) triples
   4- tuple quadruple
   5-tuple quintuple
   n-tuple

Binary Relation

If we have a subset of Cartesian product on two sets (different) it is called binary relation.

A= {10,26,40}   B={14, 32, 48}

Then AXB = { (10,14), (10,32), (10,48),
(26,14), (26,32), (26,48),
(40,14), (40,32), (40,48) }

A X B is set of all pairs (a, b) where $a \in A$ and $b \in B$

The subset of A x B yields a set of ordered pairs

$R = \{ (a, b) : a \text{ is less than } b \;\; AND \;\; a \in A \;\; and \;\; b \in B \}$

$R = \{ (10,14), (10,32), (10,48), (26,32), (26,48), (40,48) \}$

Relation *R* from a non-empty set *A* to a non-empty set *B* is a subset of Cartesian product *A x B* .

The subset is derived by describing a relationship between the first element and the second element of the ordered pairs in *A X B*

The inverse of a binary relation R is subset of AXB denoted by R^{-1} is simply the relation $\{ (b,a) : (a,b) \in R\}$

Functions

1. A function from set $A \rightarrow B$ is a binary relation R on A and B such that every element of A there is exactly one ordered pair in R with first component as a.

Image of $a \in A$

Domain of f

Range of *f*

CoDomain of f

Functions

1. *We write $f: A \rightarrow B$ where $f(x)$ is y*
2. *A function $f: A \rightarrow B$ is said to be onto B if each element in B is an image of some element in A*
3. *A function $f: A \rightarrow B$ is said to be one-to-one B if for every two distinct elements a and a' in A, $f(a) \neq f(a')$*
4. *A function $f: A \rightarrow B$ is said to be Bijection between A and B if it is both one-to-one and onto B.*

Special types of Binary Relations

Let A be a set, R is a subset of *AXA*.

1. The relation R can be represented as a directed Graph.
2. Each element is represented by a small circle called as a Node.
3. An arrow is drawn from *a* to *b* if and only iff *(a, b)* ϵ *R.*
4. These arrows are edges of the Graphs.

Let *A = {a, b, c, d}*

AXA= { }

If R subset of AXA

R= {(a,b), (b,a), (a,d), (b,c), (c,c), (c,a)}

– //Graph representation is

Reflexive Relations.

1. A relation is reflexive if R is a subset of *AXA* *if (a,a)* ϵ *R* *for each a* ϵ *A*
2. A directed Graph that represents a reflexive relation has a loop from each node to itself.

Symmetric Relations.

A relation R is a subset of *AXA* is symmetric if *if (b,a)* ϵ *R* *whenever (a, b)* ϵ *R*

1. In the corresponding directed Graph Whenever there is an arrow between tow nodes, we see arrows between those nodes in both the directions.

2. A symmetric relation without pairs of the form *(a, a)* is represented as an undirected Graph or simple graph.

Antisymmetric Relations.

A relation R is a subset of *AXA* is anti symmetric if *if (a, b)* ∈ *R, a and b are distinct, then (b, a) does not belong to R .*

Ex: if P is set of persons

The Parent-child relationship can be described as -

{ (a, b) : a, b ∈ *P and a is father of b}*

Some relations are neither symmetric nor antisymmetric.

Partial order Relations.

A relation R is a *partial order relation,* if it is reflexive, antisymmetric and transitive.

Ex: if P is set of persons

The ancestral relationship can be described as – (we consider that a person is ancestor of himself)

{ (a, b) : a, b ∈ P and a is ancestor of b}

Total order Relations.

1. A Partial order is a *Total order*

for all *whenever a, b* $\in$ *A, either (a, b)* $\in$ *R, or (b, a)* $\in$ *R.*

Ex: if P is set of persons

The ancestral relationship when we have siblings it is not total.

ex: Less-than-or-equal-to relation is total.

Alphabets and Languages
T1- Ch.1 ; Section- 1.7

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Alphabets

Data is encoded in the computer's memory as strings of bits or other symbols, appropriate for manipulation by a computer.

We look at the mathematics of strings of symbols.

An Alphabet *is* a finite set of symbols.

Ex: Roman alphabet { a, b, c, d, …,z}

An alphabet pertinent to the computer is- Binary alphabet {0, 1}.

An alphabet can be of any sort – { $, *, #}

String

A string over an alphabet is a finite sequence of symbols from the alphabet.

We use *u*, *v*, *w*, *x*, *y*, *z*, and *Greek letters* to denote strings.

Ex: *elephant* is a string over the roman alphabet

{*a, b, c, d, …,z*}

00001110 is a string over the binary alphabet {0,1}

A string may contain no symbols and is called as an *empty string* or *null string*. An empty denoted by *e .*

We can also give name to string.

Ex: The string *tiger* may be named as *w .*

The set of all strings including the empty string, over an alphabet Σ is denoted by Σ^* .

The length of the string is its length as a sequence.

Length of *tiger* is 5

If the string *tiger* is named as *w*.

The length of the string *w* is represented by $|w| = |tiger| = 5$

and $|e|=0$

The value $w(j)$ is j^{th} symbol in the string w where j is between 1 and $|w|$

If the string *tiger* is named as *w, then* $w(3)=g$ *and* $w(5)=r$

A string may contain a symbol that repeats at different positions.

We refer to them as different occurrences of the symbol.

Concatenation : Two strings over the same alphabet can be combined to form the third by the operation of concatenation.

The concatenation of the strings x and y is written as $x \circ y$ or simply xy.

Formally, $w = x \circ y$ if and only if $|w| = |x| + |y|$, $w(j) = x(j)$ for $j = 1, \ldots |x|$ and $w(|x|+j) = y(j)$ for $j = 1, \ldots |y|$.

Ex: $tiger \circ cub = tigercub$ and

$0110 \circ 110 = 0110110$

Further $w \circ e = e \circ w = w$

Concatenation is associative. $(x \circ y) \circ z = x \circ (y \circ z)$

A string *v* is a substring of *w* if and only if there are strings *x* and *y* such that *w=xvy*

Both *x* and *y* could be *e*.

An empty string is substring of any string.

A string is substring of itself .

Suffix:If *w=xv* for some *x*, then *v* is a suffix of *w*

Prefix: If *w=vy* for some *y*, then *v* is a prefix of *w*

For each string w and each natural number i, the string w^i is defined as $w^0=e$, the empty string $w^{i+1}=w^i \circ w$ for $i>= 0$

Thus $w^1=w$

$come^2 = comecome$

The reversal of a string w is denoted by w^R is the string spelled backwards.

Like how we represent infinite sets , the same format can be used to represent infinite languages.

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

Σ^* denotes all strings that can be formed using Σ.

Hence a language is any set of strings over an alphabet Σ that is any subset of Σ^* .

Hence Σ^* , Σ, and $\emptyset$ also are languages.

Most of the languages are infinite.

Ex: List all binary strings that are formed over {0, 1}

It is not possible.

We can represent by,

$L = \{w : w \in \Sigma^* \}$

The general form is $L = \{w : w \in \Sigma^*$ *and* w *has property* $P\}$

List all the strings over {0, 1} that start with two zeroes and end with even no. of ones.

Set of all integers that are divisible by 4. Ex: {4, 8, 12,…}

If *n* is the no. symbols, and *k* is the length of the string, we can have n^k strings.

The infinite languages can be represented using set builder notation.

Since languages are sets we can apply set operations to them.

Union, *Intersection*, *Difference*.

Another important operation is *concatenation*.

If L_1 and L_2 are languages then concatenation of L_1 and L_2 resulting in L

$L = L_1 \circ L_2$ or simply $L_1 L_2$

Where $L = \{w \in \Sigma^* : w = xy \text{ for some } x \in L_1 \text{ and } y \in L_2\}$

Ex: $\Sigma = \{0, 1\}$

$L_1=\{w \in \Sigma^* : w \textit{ has even number of zeros}\}$

$L_2=\{w \in \Sigma^* : w \textit{ starts with zero and rest are ones}\}$

Now, $L_1 L_2 = \{ w \in \Sigma^* : w \textit{ has odd no. of zeros}\}$

What is a Palindrome?

Kleene star of a language *L* is denoted by *L*.*

*L** is the set of all strings obtained by concatenating zero or more strings from *L*.

Concatenating zero times is empty string.

BITS Pilani
Hyderabad Campus

Finite Representations of Languages & Regular Expressions

T1- Ch.1 ; Section- 1.8

Prof.R.Gururaj
CS&IS Dept.

- We use Regular Expressions as means of representing certain subsets of strings over Σ .
- Regular Expressions are used to describe languages that consist of set of strings.
- They describe languages exclusively by means of single symbols and $\cup$ and $*$.
- They are useful for representing certain sets of string in algebraic fashion.
- Actually these describe the languages accepted by FA .
- We see $\Sigma \cup \{ (,), \cup, \Phi, * \}$ in Regular Expressions .
- Every regular expression represents a language .

We give recursive definition of RE over Σ as follows.

1. Φ and each member of Σ is a RE.
2. If α and β are REs then then so is $(\alpha\beta)$
3. If α and β are REs then then so is $(\alpha \cup \beta)$
4. If α is a REs then then so is α^*
5. Nothing is a RE unless it follows from (1) through (4)

Regular expressions are precisely those obtained recursively by application of rules 1-4 once or several times.

The relationship between RE and Language is established by a function *L*, such that if α is a RE the *L*(α) is a language represented by α.

That is *L* is a function from strings to languages.

We define the *L* function as follows.

1. $L(\Phi) = \Phi$ and $L(a) = \{a\}$ for each a of Σ.
2. If α and β are REs then then $L((\alpha\beta)) = L(\alpha) . L(\beta)$
3. If α and β are REs then then $L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$
4. If α is a REs then then $L(\alpha^*) = L(\alpha)^*$

The class of Regular Language over Σ is defined as all languages L such that the

L= $L(\alpha)$ for some RE α over Σ.

That is *Regular languages* are all languages that are described by REs.

Summary:

1. *Sets, relations, Functions*
2. *Alphabets and representation*
3. *Regular expressions, languages and representation*