Basic concepts for Foundations of Computer Science

Vedran Dunjko

Fall 2019

Contents

Chapter 1

Sets

1.1 Concepts & definitions

Specification of sets: extensive ($\{1, 2, 3, 4...\}$), intensive $\{x|x$ has property P$\}$

Examples: $R = \{a, b, c, d\}$, $S = \{x|x$ is an integer divisible by 17$\}$;

Special sets: empty set $\emptyset$, universe or universal set U. Sets of numbers $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$

Operations (on sets A, B): intersection $A \cap B$, union $A \cup B$, set difference $A - B$ *or* $(A \setminus B)$, symmetric difference $A \oplus B$ *or* $(A \triangle B)$, complement A^c.

Basic set relations (on sets A, B) subset $A \subseteq B$, equality $A = B$, strict subset $A \subset B$ *or* $(A \subsetneqq B)$, disjointness (no special symbol, but means $A \cap B = \emptyset$)

Venn diagrams: Venn diagram factors, or "regions" [4 for 2 sets (and universe), 8 for 3 sets (and universe)].

Set cardinality: $|A|$,$card(A), \#(A)$

Powerset $\mathcal{P}(A)$

Example: $A = \{1, 2\}$, $\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$. Note, $A \in \mathcal{P}(A)$.

Principle of inclusion and exclusion

Main properties of operations: commutativity, associativity, distributivity (general, and of set operations). Correct expression evaluation; order and parentheses (brackets) matter.

Laws of set algebra: (Theorem 6.5, Schaum) (commutativity, associativity, distributivity (both), idempotence, De Morgan, nul (identity) elements, double complement (involution), complementation rules.

Duality: true expression involving unions intersections and complements, empty sets and universal sets remain true if we exchange unions for intersections (and converse) and empty sets with universes.

1.2 Other topics & useful mathematical concepts

Various theorems, formal proofs, equivalences of statements, implication . Example: $A \cup B = B$ is equivalent to $A^c \cup B = U$ (in other words, if two sets satisfy any of the two expressions, then they necessarily satisfy the other as well.)

Implication ($\Rightarrow$), bi-implication $\Leftrightarrow$. Note A implies B, means if A is true, then so is B. Implication is transitive:(A implies B and B implies C), implies that A implies C (if if rains, the streets are wet. If the streets are wet, driving is dangerous. Then if it rains, driving is dangerous.). **"if and only if", iff**: A if and only if B means that A implies B and B implies A. Careful: A if B (A happens if B happens) means B implies A. A only if B means A implies B (A only if B means that if A happens, B must (have) happen(ed)... in other words, A implies B).

Other symbols: There exists $\exists$, for all $\forall$

Evaluating expressions, priorities, importance of parentheses

Binary numbers, bit-strings, and counting subsets Specifically, the proof that the cardinality of the powerset of set of n elements is the same as the number of all bit-strings of length n.

Chapter 2

Relations

Tuples: n-tuples and ordered pairs (from a family of sets), tuple equality

Cartesian product of sets $A_1, A_2, \ldots, A_n$: $A_1 \times A_2 \times, \ldots, \times A_n$.

Relations: subsets of Cartesian products, binary relations

Cardinality of Cartesian products

2.1 Binary relations

Inverse relation and inverse relation : R^{-1} for R; identity relation on A: id_A, or Δ_A or $\mathbf{1}_A$) .

Relation representations: Arrow diagrams, directed graph, matrix, graph (plot)

Domain, range of a relation ($Dom, Range$). **Image, preimage** of a set, under a relation.

Main types of relations (I): functional, total, injective, surjective

Composition of relations R, S, denoted $R \circ S$.

Main types/properties of relations (II) : reflexive, symmetric, antisymmetric, transitive, irreflexive. Note, not symmetric $\neq$ antisymmetric, irreflexive $\neq$ not reflexive.

Main types of relations (III) : equivalence relation, partial order

Main characterization of main properties via set-theoretic expressions (equivalent statements) Example: R is symmetric if and only if $R^{-1} \subseteq R$.

Closure or property P (e.g., P= reflexive, symmetric,...) the P-closure of a relation R is "smallest" relation containing R which has property P.

Comment: Note that the properties of reflexivity, symmetricity, and transitivity, can always be achieved by *adding* certain pairs to the relation; to see this note that these properties are violated only when certain pairs are missing from the relation.

Examples:

- $R \subseteq \{a, b\} \times \{a, b\}, R = \{(a, a)\}$. R is not reflexive as (b, b) is missing. However, $R \cup \{(b, b)\}$ is reflexive.
- $S \subseteq \{a, b\} \times \{a, b\}, S = \{(a, b)\}$. S is not symmetric as (b, a) is missing. So $S \cup \{(b, a)\}$ is symmetric.
- $T \subseteq \{a, b, c\} \times \{a, b, c\}, T = \{(a, b), (b, c)\}$ is not transitive as (a, c) is missing. $T \cup \{(a, c)\}$ is transitive.
- More involved example for transitive closure:

  $T \subseteq \{a, b, c, d\} \times \{a, b, c, d\}, T = \{(a, b), (b, c), (c, d)\}$ is not transitive as (a, c), and (b, d) are missing. However, $T \cup \{(a, c), (b, d)\}$ is still not transitive, as it now includes (a, c) and (c, d), but not (a, d). But, $T \cup \{(a, c), (b, d)\} \cup \{(a, d)\}$ is the transitive closure.

  Note $T \circ T = \{(a, c), (b, d)\}$; $T \circ T \circ T = (T \circ T) \circ T = \{(a, d)\}$.

Expressions for reflexive, symmetric and transitive closure Provided in slides, e.g. the transitive closure of the relation T, is $T^+ = \bigcup_{k=1}^{\infty} T^{\circ k}$.

Theorems: unions and intersections preserve reflexivity (r), symmetricity (s), transitivity (t) : in other words if R, S are both $r/s/t$ then both $R \cup S$ and $R \cap S$ are $r/s/t$.

P-closure of R: intersection of all relations R' which have the property P and contain R . By above theorems, this intersection will have the property, and be the smallest one such (no element can be removed without violating property, or loosing the containment of R).

Chapter 3

Functions

Function is synonymous to: mapping, map, transformation.

Main representations; arrow diagrams, tables, graphs. For a function $f : A \to B$, its graph (*grafiek*) is the set $\{(x, f(x))|x \in A\}$.

Formal definition A function $f : A \to B$ is a functional total relation from A to B, specifically $f \subseteq A \times B$. Note f is now identified with its graph.

Comment: recall that total means that $\forall y \in B$ there exists $x \in A$ such that $(x, y) \in f$, or, equivalently $f(x) = y$. Functional means that $f(x) = y$ and $f(x) = z$, imply that $y = z$ (no 1-to-many) (relationally, we would write this $(x, y) \in f$ and $(x, y) \in f \Rightarrow z = y$.)

Domain, range, codomain, image, preimage of a function. (Same as in the case of relations). Comment. Let $f : X \to Y$ be a function, and let $V \subseteq V$, $W \subseteq Y$. By abuse of notation, with $f(V)$ we denote the following set: $f(V) = \{f(x)|x \in V\}$, that is the set of all elements of Y reached by applying f to the elements of V. This symbol reads as if the function f takes a subset as an argument, but it is just the (natural) notation of a subset of the codomain Y (note $f(V) \subseteq Y$). One can think of this as a generalisation of the function f onto the domain $\mathcal{P}(X)$ with range in $\mathcal{P}(Y)$, but it is better to simply understand as special notation, defined once the function f is defined. More importantly, and what can cause more confusion, with $f^{-1}(W)$ we denote the following subset of the domain X: $f^{-1}(W) = \{x \in X|f(x) \in W\}$; this is the set of all elements of the domain X which are mapped into the subset W. Do not mistake this set for the inverse function $f^{-1}(y)$ – the sets $f^{-1}(W)$ are always defined, even when the function f has no inverse.

Surjective, injective and bijective functions. .Comment: proving that a function (or function family) is not surjective, injective and bijective is often easily done by providing a *counterexample*. E.g., a difference of two bijective functions f, g from $\mathbb{R}$ to.$\mathbf{R}$ $(f-g)(x) := f(x)-g(x)$ is not necessarily bijective. Proof: take f to be the identity and $f = g$. Then $(f-g)(x) = 0$ for all x (constant function). A constant function is not bijective, as it is (e.g.) not injective: take any $x_1 \neq x_2$, yet $(f-g)(x_1) = (f-g)(x_2)$, which is in contradiction with the definition of injectivity.

Inverse function. If f is a function the inverse relation of f (understood as a relation) need not be a function.

Function composition For $f : A \to B$, $g : B \to C$, then $g \circ f : A \to C$ is defined with $(g \circ f)(x) = g(f(x))$.
Comment: this notation is opposite to the case of relations. Consider the relations $R_f \subseteq A \times B$, $S_g \subseteq B \times C$, which are just the relational representations of f and g (i.e. their graphs), so, $R_f = \{(x, f(x))|x \in A\}$ and $S_g = \{(y, g(y))|y \in B\}$. The composed relation $R_f \circ S_g$ is a relation from A to C, and is the relational representation (the graph) of the composed function $g \circ f$. Note the reversal of the order g and f. The relational composition is read "first R_f then S_g", whereas the functional is read g after f. [We have introduced the notation R_f and S_g for the relational versions of f and g for clarity; we could have used the symbols f and g to mean both, but in this case, the ordering would be more confusing.]

Sequences and series. Sequences and tuples; Series and sums; Summation and product symbols $\sum_{i=k}^{l} a_i$, $\sum_{i<k} a_i$, $\sum_{i\in S} a_i$, analogously for products, unions, intersections ($\prod_{i=k}^{l} a_i$, $\bigcup_{i=k}^{l} a_i$,$\bigcap_{i=k}^{l} a_i$). Arithmetic and geometric series. Basic summation identities (e.g. what is $\sum_{i=4}^{34} i$?)

Chapter 4

Graph theory 1

Graphs: definitions and basic concepts; Vertices, edges, undirected, directed; representation: graph, adjacency matrix; Simple graphs; connectedness, adjacency, incidence, neighbourhood. Degree.

Sum-degree formula and handshaking lemma. Sum of all degrees is twice the edge number. Number of vertices with odd degree is even.

Graph equality; graph isomorphism.

Subgraphs. induced subgraphs, vertex and edge removal. Connected components,

Path, simple path, trail, cycle, circuit. If there is a path between vertices, there is a simple path. Distance between vertices, graph diameter.

Seven Bridges of Köningsberg.

Eulerian graph: has Eulerian circuit (each edge once, end where started). Eulerian trail. For Euler circuit, necessary and sufficient condition: all vertices even degree. For trail: exactly 2 or 0 (then also circuit) odd degree vertices.

Hamilton cycle. Closed path with each vertex traversed exactly once.

Concepts with directed graphs, and topological ordering. Topological sorting (ordering) exists if and only if the graph is directed without cycles. Theorems 9.2 and 9.3 (Schaum): a) strongly connected if and only a closed spanning path exists b) weakly connected if and only if a spanning semi path exists; A directed graph G without cycles has a source and a sink.

Planar graphs.

Complete graphs.

Bipartite graphs. Graph is bipartite if and only if if it has no cycles of odd length.

Weighted and labeled graphs.

Chapter 5

Basic combinatorics

Permutations. $n! = \prod_{i=1}^{n} i$. Sampling without replacement.

Ordered lists. Length n, k elements $= k^n$. Sampling with replacement.

Subsets. Number of k sized subsets out of n distinct elements: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$, ("n-choose-k"). Binomial coefficients.

Chapter 6

Recursion and induction

Inductive definition, recursive definition E.g., Fibonacci sequence.

Mathematical induction. Proofs via induction (prove that some property – equality, inequality, congruence, etc., holds for all numbers (larger than some constant)): (1) base, (2) inductive step;

Mathematical induction over inductively defined sets. E.g.. all trees have $n-1$ edges... E.g., inductively defined languages. E.g., Blurpsen language and proving properties.

Chapter 7

Trees

Basic definitions. Undirected, directed, rooted, ordered tree...

Terminology. Leaf, internal vertex (node), edge, root, child, sibling, parent ancestor, descendant. Depth of a vertex. Height of a vertex. Depth of tree. Subtrees.

Main properties. If G is a tree, then $G-e$ is not connected. $|E| = |V|-1$. Inductive proof.

(Equivalent) characterizations : T is a tree; T is maximally acyclic; There exists a unique simple path between any two vertices of T; T is acyclic with $n-1$ edges; T is minimally connected; T is connected with $n-1$ edges.

Binary trees Definition (recursive). Types: complete, full; extended binary tree.

First child- right sibling encoding (Knuth transformation) ; Encoding arbitrary trees into binary trees.

Tree traversals. Preorder (NLR for binary). Postorder (LRN for binary). Inorder (symmetric ordering: only binary, LNR).

Arithmetic expressions and tree traversals. Polish and reverse Polish notation.

Chapter 8

Modulo computation and equivalence relations

Congruence modulo n. Definition. Is an equivalence relation.

Equivalence classes.

Residue classes modulo n: equivalence classes of the relation of congruence modulo n.

Modulo arithmetic: if $a \equiv b \ (mod\ n)$, and $c \equiv d \ (mod\ n)$, then: (1) $a \pm c \equiv b \pm c \ (mod\ n)$; (2) $a \times c \equiv b \times c \ (mod\ n)$; Also if $a \equiv b \ (mod\ n)$ then $a^k \equiv b^k \ (mod\ n)$. Due to these rules, and the fact that the congruence relations are transitive, it is easy to compute modulo computations; mod can essentially be taken at any point (in exponents, in products and sums)...

Computing with residue classes.

Chapter 9

Languages

Basic definitions; Alphabet, word/string, empty string (λ), word length, set of all words, Kleene star. Language. Empty language.

Languages as sets, and set operations.

Operations on words. Concatenation, powering, Kleene star, mirroring. Basic properties (length).

Operation on languages – derived from operations on words. Concatenation, powering, Kleene star, mirroring (of languages).

Specifying languages.

Regular expression. Recursive definition. Language defined by a regular expression.

Regular language. Generated by union, concatenation and Kleene star from singlet sets. Regular languages are exactly those specified by a regular expression. Set of regular languages is closed under mirroring.

From inductive definition of a regular language to a defining regular expression (set expression)

Chapter 10

Automata

Basic definitions Finite state machine/automaton. States, transition table, transition graph, labels. Terminal states. Deterministic automaton (exactly one arrow with each label from each state). Non-deterministic automaton.

Automata and accepting words. Labelling of a walk. Accepts word if ends in a terminal (accepting) state (for non-deterministic, one of the labelings ends in an accepting state).

A language is representable by a deterministic finite state automaton if and only if it is representable by a non-deterministic finite state automaton. Not all languages are recognised/accepted by a finite state automaton (counterexample?).

Theorem (Kleene): A language is representable as a finite state automation if and only if it is a regular language.