COMP1002 exam study sheet

- *Propositional statement*: expression that has a truth value (true/false). It is a *tautology* if it is always true, *contradiction* if always false.
- *Logic connectives*: negation ("not") $\neg p$, conjunction ("and") $p \wedge q$, disjunction ("or") $p \vee q$, implication $p \to q$ (equivalent to $\neg p \vee q$), biconditional $p \leftrightarrow q$ (equivalent to $(p \to q) \wedge (q \to p)$). The order of precedence: $\neg$ strongest, $\wedge$ next, $\vee$ next, $\to$ and $\leftrightarrow$ the same, weakest.
- If $p \to q$ is an implication, then $\neg q \to \neg p$ is its *contrapositive*, $q \to p$ a *converse* and $\neg p \to \neg q$ an *inverse*. An implication is equivalent to its contrapositive, but not to converse/inverse or their negations. A negation of an implication $p \to q$ is $p \wedge \neg q$ (it is not an implication itself!)
- A *truth table* has a line for each possible values of propositional variables (2^k lines if there are k variables), and a column for each variable and subformula, up to the whole statement. The cells of the table contain T and F depending whether the (sub)formula is true for the corresponding values of variables.
- A *truth assignment* is a string of values of variables to the formula, usually a row with values of first several columns in the truth table (number of columns = number of variables). A truth assignment is *satisfying* the formula if the value of the formula on these variables is T, otherwise the truth assignment is *falsifying*. A truth assignment can be encoded by a formula that is a $\wedge$ of variables and their negations, with negated variables in places that have F (false) in the assignment, and non-negated that have T (true). For example, $x = T, y = F, z = F$ is encoded as $(x \wedge \neg y \wedge \neg z)$.It is an encoding in a sense that this formula is true only on this truth assignment and nowhere else.
- Finding a method for checking if a formula has a satisfying assignment that is always significantly faster than using truth tables (that is, better than brute-force search) is one of Clay institute millenium problems with a million dollar prize, known as "P vs. NP".
- Two formulas are *logically equivalent* if they have the same truth table. The most famous example of logically equivalent formulas is $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ (with a dual version $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$) where p and q can be arbitrary (propositional, here) formulas. These pairs of logically equivalent formulas are called *DeMorgan's law*.
- There are several other important pairs of logically equivalent formulas, called *logical identities* or *logic laws*. We will talk more about them when we talk about Boolean algebras. Here, just remember that $FALSE \wedge p \equiv p \wedge \neg p \equiv FALSE$, $FALSE \vee p \equiv TRUE \wedge p \equiv p$ and $TRUE \vee p \equiv p \vee \neg p \equiv TRUE$.
- A set of logic connectives is called *complete* if it is possible to make a formula with any truth table out of these connectives. For example, $\neg, \wedge$ is a complete set of connectives, and so is the Sheffer's stroke $|$ (where $p|q \equiv \neg(p \wedge q)$), also called NAND for "not-and". But $\vee, \wedge$ is not a complete set of connectives since then it is impossible to express a truth table with 0 when all variables are 1.
- An *argument* consists of several formulas called *premises* and a final formula called a *conclusion*. If we call premises $A_1 \dots A_n$ and conclusion B, then an argument is *valid* iff premises imply the conclusion, that is, $A_1 \wedge \dots \wedge A_n \to B$. We usually write them in the following format:

Today is either Thursday or Friday
On Thursdays I have to go to a lecture
Today is not Friday (alternatively, On Friday I have to go to the lecture)
———————————————
$\therefore$ I have to go to a lecture today

- A valid form of argument is called *rule of inference*. The most prominent such rule is called *modus ponens*.

  $p \to q$
  p ————
  $\therefore q$

- There are several main types of proofs depending on the types of rules of inference used in the proof. The main ones are *direct proof, by contraposition, by contradiction* and *by cases*.

- There are two main normal forms for the propositional formulas. One is called *Conjunctive normal form* (CNF) and is an $\wedge$ of $\vee$ of either variables or their negations (here, by $\wedge$ and $\vee$ we mean several formulas with $\wedge$ between each pair, as in $(\neg x \vee y \vee z) \wedge (\neg u \vee y) \wedge x$. A *literal* is a variable or its negation (x or $\neg x$, for example). A $\vee$ of (possibly more than 2) literals is called a *clause*, for example $(\neg u \vee z \vee x)$, so a CNF is true for some truth assignment whenever this assignment makes each of the clauses is true, that is, each clause has a literal that evaluates to true under this assignment. A *Disjunctive normal form* (DNF) is like CNF except the roles of $\wedge$ and $\vee$ are reversed. A $\wedge$ of literals in a DNF is called a *term*. To construct canonical DNF and a CNF, start from a truth table and then for every satisfying truth assignment $\vee$ its encoding to a DNF, and for every falsifying truth assignment $\wedge$ the negation of its encoding to the CNF, and apply DeMorgan's law. This may result in a very large CNFs and DNFs, comparable to the size of the truth table itself ($2^{\text{number of variables}}$).

- A *resolution proof system* is used to find a contradiction in a formula (and, similarly, to prove that a formula is a tautology by finding a contradiction in its negation). Resolution starts with a formula in a CNF form, and applies the rule "from clause $(C \vee x)$ and clause $(D \vee \neg x)$ derive clause $(C \vee D)$ until a falsity F (equivalently, empty clause ()) is reached (so in the last step one of the clauses being *resolved* contains just one variable and another clause being resolved contains just that variable's negation. Resolution can be used to check the validity of an argument by running it on the $\wedge$ of all premises (converted, each, to a CNF) $\wedge$ together with the negation of the conclusion.

- *Pigeonhole principle* If n pigeons sit in $n-1$ holes, so that each pigeon sits in some hole, then some hole has at least two pigeons. Can be used to show, for example, that there are two people in our class who carry the same number of pens. There is no small resolution proof of the pigeonhole principle.

- *Boolean functions* are functions which take as argument boolean (ie, propositional) variables and return 1 or 0 (or, the convention here is 1 instead of T, and 0 instead of F). Each Boolean function on n variables can be fully described by its truth table. A size of a truth table of a function on n variables is 2^n. Even though we often can have a smaller description of a function, vast majority of Boolean functions cannot be described by anything much smaller. Every Boolean function can be described by a CNF or DNF, using the above construction.

Predicate logic:

- A *predicate* is like a propositional variable, but with *free variables*, and can be true or false depending on the values of these free variables. A *domain* of a predicate is a set from which the free variables can take their values (e.g., the domain of $Even(n)$ can be integers).
- *Quantifiers* For a predicate $P(x)$, a quantified statement "for all" ("every", "all") $\forall x P(x)$ is true iff $P(x)$ is true for every value of x from the domain (also called universe); here, $\forall$ is called a *universal quantifier*. A statement "exists" ("some", "a") $\exists x P(x)$ is true whenever $P(x)$ is true for at least one element x in the universe; $\exists$ is an existential quantifier. The word "any" means sometimes $\exists$ and sometimes $\forall$. A domain (universe) of a quantifier, sometimes written as $\exists x \in D$ and $\forall x \in D$ is the set of values from which the possible choices for x are made. If the domain of a quantifier is empty, then if the quantifier is universal then the formula is true, and if quantifier is existential, false. A *scope* of a quantifier is a part of the formula (akin to a piece of code) on which the variable under that quantifier can be used (after the quantifier symbol/inside the parentheses/until there is another quantifier over a variable with the same name). A variable is *bound* if it is under a some quantifier symbol, otherwise it is free.
- *First-order formula* A predicate is a first-order formula (possibly with free variables). A $\wedge, \vee, \neg$ of first-order formulas is a first-order formula. If a formula $A(x)$ has a free variable (that is, a variable x that occurs in some predicates but does not occur under quantifiers such as $\forall x$ or $\exists x$), then $\forall x\ A(x)$ and $\exists x\ A(x)$ are also first-order formulas.
- *Negating quantifiers.* Remember that $\neg \forall x P(x) \equiv \exists x \neg P(x)$ and $\neg \exists x P(x) \equiv \forall x \neg P(x)$.
- *Reasoning in predicate logic* The *rule of universal instantiation* says that if some property is true of everything in the domain, then it is true for any particular object in the domain. A combination of this rule with modus ponens such as what is used in the "all men are mortal, Socrates is a man $\therefore$ Socrates is mortal" is called universal modus ponens.
- *Normal forms* In a first-order formula, it is possible to rename variables under quantifiers so that they all have different names. Then, after pushing negations into the formulas under the quantifiers, the quantifier symbols can be moved to the front of a formula (making their scope the whole formula).
- *Formulas with finite domains* If the domain of a formula is finite, a formula can be converted into a propositional formula by changing each $\forall x$ quantifier with a $\wedge$ of the formula on all possible values of x; an $\exists$ quantifier becomes a $\vee$. Then terms of the form $P(value)$ (e.g., $Even(5)$) are treated as propositional variables.
- *Limitations of first-order logic* There are concepts that are not expressible by first-order formulas, for example, transitivity ("is there a flight from A to B with arbitrary many legs?" cannot be a database query described by a first-order formula).

Proof strategies

- Existential statement: $\exists x F(x)$. Constructive proof: give an example satisfying the formula under the quantifier (e.g, exists x which is both even and prime: take $n = 2$), then conclude by the *existential generalization rule* that $\exists x F(x)$ is true. Non-constructive proof: If the proof says $\exists n P(n)$, show that assuming $\forall n \neg P(n)$ leads to contradiction.

- Universal statement: $\forall x F(x)$. To prove that it is false, give a counterexample. To prove that it is true, start with the *universal instantiation*: take an arbitrary element, give it a name (say n) , and prove that $F(n)$ holds without any additional assumptions. By *universal generalization*, conclude that $\forall x F(x)$ holds.

- To prove $F(n)$

  - *Direct proof:* show that $F(n)$ holds directly, using definition, algebra, etc. If $F(n)$ is of the form $G(n) \rightarrow H(n)$, then assume $G(n)$ and derive $H(n)$ from this assumption. Examples: sum of even integers is even, if $n \equiv m(\mod d)$ then there is $k \in \mathbb{Z}$ such that $n = m + kd$, if n is odd then $n^2 \equiv 1(\mod 8)$, Pythagores theorem.
  - *Proof by cases* If $F(x)$ is of the form $(G_1(x) \vee G_2(x) \vee \cdots \vee G_k(x)) \rightarrow H(x)$, then prove $G_1(x) \rightarrow H(x)$, $G_2(x) \rightarrow H(x) \ldots G_k(x) \rightarrow H(x)$. Examples: sum of two consecutive integers is odd, $\forall x, y \in \mathbb{R} |x + y| \leq |x| + |y|$, $\min(x, y) = (x + y - |x - y|)/2$, $k^2 + k$ is even.
  - *Proof by contraposition* If $F(n)$ is of the form $G(n) \rightarrow H(n)$, can prove $\neg H(n) \rightarrow \neg G(n)$ (that is, assume that $H(n)$ is false, and derive that $G(n)$ is false). Examples: pigeonhole principle, if a square of an integer is even, then integer itself is even.
  - *Proof by contradiction* To prove $F(n)$, show that $\neg F(n) \rightarrow FALSE$. Examples: $\sqrt{(2)}$ is irrational, there are infinitely many primes.

- Some definitions:

  - An $n \in \mathbb{Z}$ is *even* if $\exists k \in \mathbb{Z}$ such that $n = 2k$. An $n \in \mathbb{Z}$ is *odd* if $\exists k \in \mathbb{Z}$ such that $n = 2k + 1$. An $n \in \mathbb{Z}$ is *divisible by* $m \in \mathbb{Z}$ if $\exists k \in \mathbb{Z}$ such that $n = km$.
  - Modular arithmetic: for any $n, d \neq 0 \in \mathbb{Z}$ $\exists q, r \in \mathbb{Z}$ such that $n = qd + r$ and $0 \leq r < d$. Here, q is a *quotient* and r is a *remainder*. *Congruence:* for $n, m, d \neq 0 \in \mathbb{Z}$, $n \equiv m(\mod d)$ ("n is congruent to m mod d") iff $\exists q_1, q_2, r \in \mathbb{Z}$ such that $0 \leq r \leq d$, $n = q_1 d + r$ and $m = q_2 d + r$. That is, n and m have the same remainder modulo d.
  - Absolute value of $x \in \mathbb{R}$, denoted $|x|$, is x if $x \geq 0$ and $-x$ if $x < 0$.

Set Theory

- A *set* is a well-defined collection of objects, called elements of a set. An object x belongs to set A is denoted $x \in A$ (said "x in A" or "x is a member of A"). Usually for every set we consider a bigger "universe" from which its elements come (for example, for a set of even numbers, the universe can be all natural numbers). A set is often constructed using *set-builder notation*: $A = \{x \in U | P(x)\}$ where U is a universe , and $P(x)$ is a predicate statement; this is read as "x in U such that $P(x)$" and denotes all elements in the universe for which $P(x)$ holds. Alternatively, for a small set, one can list its elements in curly brackets (e.g., $A = \{1, 2, 3, 4\}$.)

- A set A is a *subset* of set B, denoted$A \subseteq B$, if $\forall x (x \in A \rightarrow x \in B)$. It is a *proper* subset if $\exists x \in B$ such that $x \notin A$. Otherwise, if $\forall x (x \in A \leftrightarrow x \in B)$ two sets are equal.

- Special sets are: *empty set* $\emptyset$, defined as $\forall x (x \notin \emptyset)$. Universal set U: all potential elements under consideration at given moment. Natural numbers $\mathbb{N}$ (here, $0 \in \mathbb{N}$), integers $\mathbb{Z}$, rationals $\mathbb{Q}$, reals $\mathbb{R}$.

Table 1: **Laws of boolean algebras, logic and sets**

Name	**Logic law**	**Set theory law**	**Boolean algebra law**
Double Negation	$\neg\neg p \equiv p$	$\overline{\overline{A}} = A$	$\overline{\overline{x}} = x$
DeMorgan's laws	$\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$	$\overline{A \cup B} = \overline{A} \cap \overline{B}$ $\overline{A \cap B} = \overline{A} \cup \overline{B}$	$\overline{x + y} = \overline{x} \cdot \overline{y}$ $\overline{x \cdot y} = \overline{x} + \overline{y}$
Associativity	$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$	$(x + y) + z = x + (y + z)$ $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
Commutativity	$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	$A \cup B = B \cup A$ $A \cap B = B \cap A$	$x + y = y + x$ $x \cdot y = y \cdot x$
Distributivity	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ $x + (y \cdot z) = (x + y) \cdot (x + z)$
Idempotence	$(p \vee p) \equiv p \equiv (p \wedge p)$	$A \cup A = A = A \cap A$	$x + x = x = x \cdot x$
Identity	$p \vee F \equiv p \equiv p \wedge T$	$A \cup \emptyset = A = A \cap U$	$x + 0 = x = x \cdot 1$
Inverse	$p \vee \neg p \equiv T$ $p \wedge \neg p \equiv F$	$A \cup \bar{A} = U$ $A \cap \bar{A} = \emptyset$	$x + \bar{x} = 1$ $x \cdot \bar{x} = 0$
Domination	$p \vee T \equiv T$ $p \wedge F \equiv F$	$A \cup U = U$ $A \cap \emptyset = \emptyset$	$x + 1 = 1$ $x \cdot 0 = 0$

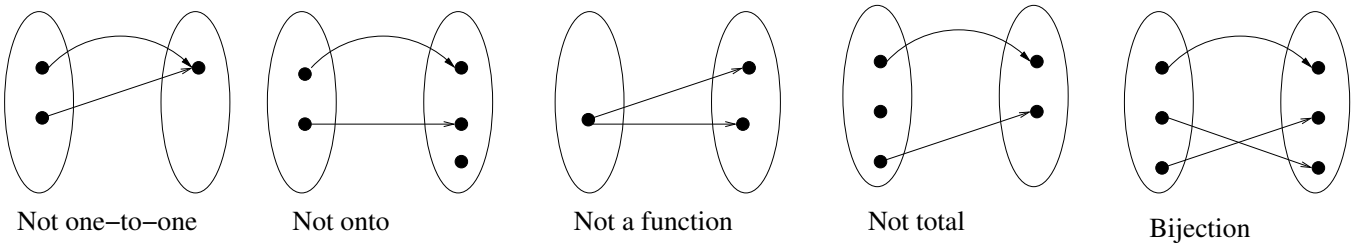
- A *power set* for a given set A, denoted 2^A or $\mathcal{P}(A)$, is the set of all subsets of A. If A has n elements, then 2^A has 2^n elements (since for every element there are two choices, either it is in, or not).
- Basic set operations are a *complement* $\bar{A}$, denoting all elements in the universe that are *not* in A, then *union* $A \cup B = \{x | x \in A \text{ or } x \in B\}$, and *intersection* $A \cap B = \{x | x \in A \text{ and } x \in B\}$ and *set difference* $A - B = \{x | x \in A \text{ and } x \notin B\}$. Lastly, the *Cartesian product* of two sets is a set of pairs $A \times B = \{(a, b) | a \in A \text{ and } b \in B\}$. Here, the parentheses notation means that the order matters, so a pair (a, b) is different from a pair (b, a), whereas with set notation $\{a, b\}$ order does not matter. This notation generalizes from pairs to *tuples*: $A_1 \times \cdots \times A_k = \{(a_1, \ldots, a_k) | a_1 \in A_1 \wedge \cdots \wedge a_k \in A_k\}$.
- To prove that $A \subseteq B$, show that if you take an arbitrary element of A then it is always an element of B. To prove that two sets are equal, show both $A \subseteq B$ and $B \subseteq A$. You can also use set-theoretic identities.
- A *cardinality* of a set is the number of elements in it. Two sets have the same cardinality if there is a bijection between them. If the cardinality of a set is the same as the cardinality of $\mathbb{N}$, the set is called *countable*. If it is greater, then *uncountable*.
- *Principle of inclusion-exclusion*: The number of elements in $A \cup B$, $|A \cup B| = |A| + |B| - |A \cap B|$. In general, add all odd-sized intersections and subtract all even-sized intersections.
- **Boolean algebra**: A set B with three operations $+, \cdot$ and $^-$, and special elements 0 and 1 such that $0 \neq 1$, and axioms of identity, complement, associativity and distributivity. Logic is a boolean algebra with F being 0, T being 1, and $^-, +, \cdot$ being $\neg, \vee, \wedge$,respectively. Set theory is a boolean algebra with $\emptyset$ for 0, U for 1, and $^-, \cup, \cap$ for $^-, +, \cdot$. Boolean algebra is sound and complete: anything true is provable (completeness) and anything provable is true (soundness).

- A k-ary **relation** R is a subset of Cartesian product of k sets $A_1 \times \cdots \times A_k$. We call elements of such R "k-tuples". A *binary* relation is a subset of a Cartesian product of two sets, so it is a set of *pairs* of elements. E.g., $R \subset \{2,3,4\} \times \{4,6,12\}$, where $R = \{(2,4),(2,6),(2,12),(3,6),(3,12),(4,4),(4,12)\}$ is a binary relation consisting of pairs of numbers such that the first number in the pair divides the second.

- *Database queries* A *query* in a relational database is often represented as a first-order formula, where predicates correspond to the relations occurring in database (that is, a predicate is true on a tuple of values of variables if the corresponding relation contains that tuple). A query "returns" a set of values that satisfy the formula describing the query; a Boolean query, with no free variables, returns true or false. For example, a relation $StudentInfo(x,y)$ in a university database contains, say, all pairs x, y such that x is a student's name and y is the student number of student with the name x. A corresponding predicate $StudentInfo(x,y)$ will be true on all pairs x, y that are in the database. A query $\exists x StudentInfo(x,y)$ returns all valid student numbers. A query $\exists x \exists y StudentInfo(x,y)$, saying that there is at least one registered student, returns true if there is some student who is registered and false otherwise.

- Binary relation R (usually over $A \times A$) can be:
  - *reflexive*: $\forall x \in A \;\; R(x,x)$. For example, $a \leq b$ and $a = b$.
  - *symmetric*: $\forall x, y \in A \;\; R(x,y) \rightarrow R(y,x)$. For example, $a = b$, $''sibling''$.
  - *antisymmetric*: $\forall x, y \in A \;\; R(x,y) \wedge R(y,x) \rightarrow x = y$. For example, $a < b$, $''parent''$.
  - *transitive*: $\forall x, y, z \in A \;\; (R(x,y) \wedge R(y,z) \rightarrow R(x,z)$. For example, $a = b$, $a < b$, $a|b$, $''ancestor''$.
  - *equivalence*: if R is reflexive, symmetric and transitive. For example, $a = b$, $a \equiv b$.
  - *order (total/partial)*: If R is antisymmetric, reflexive and transitive, then R is an order relation. If, additionally, $\forall x, y \in A \;\; R(x,y) \vee R(y,x)$, then the relation is a *total* order (e.g., $a \leq b$). Otherwise, it is a *partial* order (e.g., "ancestor", $a|b$.) An order relation can be represented by a Hasse diagram, which shows all connections between elements that cannot be derived by transitivity-reflexivity (e.g., "$p|n$" on $\{2,6,12\}$ will be depicted with just the connections 2 to 6 and 6 to 12.)
  - *transitive closure*: A transitive closure of R is a relation R^{tc} that contains, in addition to R, all x, y such that there are $k \in \mathbb{N}, v_1, \ldots, v_k \in A$ such that $x = v_1, y = v_k$, and for i such that $1 \leq i < k$, $R(v_i, v_{i+1})$. For example, an "ancestor" relation is the transitive closure of the "parent" relation.

- A **function** f: $A \rightarrow B$ is a special type of relation $R \subseteq A \times B$ such that for any $x \in A, y, z \in B$, if $f(x) = y$ and $f(x) = z$ then $y = z$. If $A = A_1 \times \ldots \times A_k$, we say that the function is k-ary. In words, a $k+1$-ary relation is a k-ary function if for any possible value of the first k variables there is at most one value of the last variable. We also say "f is a mapping from A to B" for a function f, and call $f(x) = y$ "f maps x to y".
  - A function is *total* if there is a value $f(x) \in B$ for every x; otherwise the function is *partial*. For example, $f : \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = x^2$ is a total function, but $f(x) = \frac{1}{x}$ is partial, because it is not defined when $x = 0$.



- If a function is $f\colon A \to B$, then A is called the *domain* of the function, and B a *codomain*. The set of $\{y \in B \mid \exists x \in A, f(x) = y\}$ is called the *range* of f. For $f(x) = y$, y is called the *image* of x and x a *preimage* of y.
- A *composition* of $f\colon A \to B$ and $g\colon B \to C$ is a function $g \circ f\colon A \to C$ such that if $f(x) = y$ and $g(y) = z$, then $(g \circ f)(x) = g(f(x)) = z$.
- A function $g\colon B \to A$ is an *inverse* of f (denoted f^{-1}) if $(g \circ f)(x) = x$ for all $x \in A$.
- A total function f is *one-to-one* if for every $y \in B$, there is at most one $x \in A$ such that $f(x) = y$. For example, the function $f(x) = x^2$ is not one-to-one when $f\colon \mathbb{Z} \to \mathbb{N}$ (because both $-x$ and x are mapped to the same x^2), but is one-to-one when $f\colon \mathbb{N} \to \mathbb{N}$.
- A total function $f\colon A \to B$ is *onto* if the range of f is all of B, that is, for every element in B there is some element in A that maps to it. For example, $f(x) = 2x$ is onto when $f\colon \mathbb{N} \to Even$, where $Even$ is the set of all even numbers, but not onto $\mathbb{N}$.
- A total function that is both one-to-one and onto is called a *bijection*.
- A function $f(x) = x$ is called the *identity* function. It has the property that $f^{-1}(x) = f(x)$. A function $f(x) = c$ for some fixed constant c (e.g., $f(x) = 3$) is called a *constant* function.

- **Comparing set sizes**

  Two sets A and B have the same cardinality if exists f that is a bijection from A to B.
  If a set has the same cardinality as $\mathbb{N}$, we call it a *countable* set. If it has cardinality larger than the cardinality of $\mathbb{N}$, we call it *uncountable*. If it has k elements for some $k \in \mathbb{N}$, we call it *finite*, otherwise *infinite* (so countable and uncountable sets are infinite). E.g: $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, Even$, set of all finite strings, Java programs, or algorithms are all countable, and $\mathbb{R}, \mathbb{C}$, power set of $\mathbb{N}$, are all uncountable. E.g., to show that $\mathbb{Z}$ is countable, we prove that there is a bijection $f\colon \mathbb{Z} \to \mathbb{N}$: take $f(x) = \begin{cases} 2x & x \geq 0 \\ 1 - 2x & x < 0 \end{cases}$.
  It is one-to-one because $f(x) = f(y)$ only if $x = y$, and it is onto because for any $y \in \mathbb{N}$, if it is even then its preimage is $y/2$, if it is odd $-\frac{y-1}{2}$. Often it is easier to give instead two one-to-one functions, from the first set to the second and another from the second to the third. Also, often instead of a full description of a function it is enough to show that there is an enumeration such that every element of, say, $\mathbb{Z}$ is mapped to a distinct element of $\mathbb{N}$. To show that one finite set is smaller than another, just compare the number of elements. To show that one infinite set is smaller than another, in particular that a set is uncountable, use *diagonalization*: suppose that a there is an enumeration of elements of a set, say, $2^{\mathbb{N}}$ by elements of $\mathbb{N}$. List all elements of $2^{\mathbb{N}}$ according to that enumeration. Now, construct a new set which is not in the enumeration by making it differ from the k^{th} element of the enumeration in the k^{th} place (e.g., if the second set contains element 2, then the diagonal set will not contain the element 2, and vice versa).

Mathematical induction and recursive definitions.

- **Mathematical induction** Let $a, n \in \mathbb{N}$, and $P(n)$ is a predicate with free variable n. Then the mathematical induction principle says:

$$(P(a) \wedge \forall n \geq a\ (P(n) \rightarrow P(n+1))) \rightarrow \forall n \geq a\ P(n)$$

  That is, to prove that a statement is true for all (sufficiently large) n, it is enough to prove that it holds for the smallest $n = a$ (*base case*) and prove that if it holds for some arbitrary $n > a$ (*induction hypothesis*) then it also holds for the next value of n, $n + 1$ (*induction step*).

  In *strong induction* we are assuming that the statement holds for *all* values smaller than $n + 1$: that is, the induction hypothesis becomes $(\forall i \in \{a, \ldots, k-1\} P(i)) \rightarrow P(k)$. Examples: generalized associative law for sets, every natural number ≥ 2 is a product of primes.

  We also can use multiple base cases, as long as there is a finite number of them (as in the coins problem).

  A *well-ordering principle* states that every set of natural numbers has the smallest element. It is used to prove statements by counterexample: e.g., "define set of elements for which $P(n)$ does not hold. Take the smallest such n. Show that it is either not the smallest, or $P(n)$ holds for it".

  These three principles, Induction, Strong Induction and Well-ordering are equivalent. If you can prove a statement by one of them, you can prove it by the others.

  The following is the structure of an induction proof.

  1. $P(n)$. State which assertion $P(n)$ you are proving by induction. E.g., $P(n)$: $2^n < n!$.
  2. Base case: Prove $P(a)$ (usually just put a in the expression and check that it works). E.g., $P(4)$: $2^4 < 4!$ holds because $2^4 = 16$ and $4! = 24$ and $16 < 24$.
  3. Induction hypothesis: "assume $P(k)$ for some $k \geq a$". I like to rewrite the statement for $P(k)$ at this point, just to see what I am using. For example, "Assume $2^k < k!$". Sometimes, to simplify calculations, use $k - 1$ instead of k here (and then k instead of $k + 1$ in the induction step).
  4. Induction step: prove $P(k + 1)$ under assumption that $P(k)$ holds. This is where all the work is. Start by writing $P(k + 1)$ (for example, $2^{k+1} < (k + 1)!$. Then try to make one side of the expression to "look like" (one side of) the induction hypothesis, maybe + some stuff and/or times some other stuff. For example, $2^{k+1} = 2 \cdot \mathbf{2^k}$, which is 2^k times additional 2. The next step is either to substitute the right side of induction hypothesis in the resulting expression with the left side (e.g., 2^k in $2 \cdot 2^k$ with $k!$, giving $2 \cdot k!$, or just apply the induction hypothesis assumption to prove the final result You might need to do some manipulations with the resulting expression to get what you want, but applying the induction hypothesis should be the main part of the proof of the induction step.

- A *recursive definition* (of a set) consists of 1) The base of recursion: "these several elements are in the set". 2) The recursion: "these rules are used to get new elements". 3) A restriction: "nothing extraneous is in the set". Examples: set of trees, set of even-length binary strings...

- A *Structural induction* is used to prove properties about recursively defined sets. The base case of the structural induction is to prove that $P(x)$ holds for the elements in the base, and the induction steps proves that if the property holds for some elements in the set, then it holds for all elements

obtained using the rules in the recursion. Examples: number of nodes in a full binary tree is bounded by log of the tree height, even number of odd-degree vertices in a tree.

- Recursive definitions of sequences and functions are similar to sets: define a function on 0 or 1 (or several), and then give a rule for constructing new values from smaller ones. Examples: arithmetic and geometric progression.

- A *recurrence relation* expresses subsequent elements of a sequence (or values of a function) in terms of previous ones. For example, definition of Fibonacci numbers as $F(n) = F(n-1) + F(n-2)$ or $s_n = s_{n-1} + d$ for arithmetic progression. A solution or *closed form* of a recurrence relation is an expression of n^{th} term in the sequence (or value of the function at n) in terms of just n itself, without the base case: for example, $s_n = c + nd$ for arithmetic progression.

- To compare grows rate of the functions, use $O()$-notation (big-Oh notation): $f(n) \in O(g(n))$ if $\exists n_0, c > 0$ such that $\forall n \geq n_0$ $f(n) \leq cg(n)$. In algorithmic terms, if $f(n)$ and $g(n)$ are running times of two algorithms for the same problem, $f(n)$ works faster on large inputs. If both $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$ then $f(n) \in \Theta(g(n))$ (f is in big-Theta of g).

- The Master Theorem for solving recurrences: Let $a, b, c, d \in \mathbb{R}$ with $a \geq 1, b \geq 2, c \geq 0$ and $d \geq 0$. Let $f(n) \in \Theta(n^c)$. Suppose $T(n)$ is defined by the recurrence $T(n) = aT(\lceil n/b \rceil) + f(n)$, with the basis $T(n) = d$. Then the growth rate of $T(n)$ is:

  1. If $\log_b a < c$ then $T(n) \in \Theta(f(n))$.
  2. If $\log_b a = c$ then $T(n) \in \Theta(f(n) \log n)$.
  3. If $\log_b a > c$ then $T(n) \in \Theta(n^{\log_b a})$.

Regular languages, finite automata and context-free grammars

- An *alphabet* is a finite set of symbols (e.g.: binary alphabet $\{0, 1\}$, English alphabet, etc). An alphabet is usually denoted Σ (not to be confused with the summation sign, this is a capital Greek letter "Sigma"). A (finite) *string* (also called *word*) is a (finite) sequence of letters from an alphabet. A special *empty string* is denoted ϵ or λ. A set of all strings is denoted Σ^* (pronounced "Sigma-star", star notation is explained below). A *language* L over an alphabet Σ is a (possibly infinite) set of words from this language: $L \subseteq \Sigma^*$.

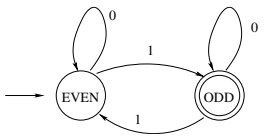
- A *context-free grammar* consists of 1) Finite set Σ of terminals (letters in the alphabet). 2) Finite set V of variables (also called non-terminals), including a specical starting variable. 3) Finite set of rules, each of the form $A \to w$ for some variable A and a string of variables and terminals w (several rules for the same variable can also be written using symbol "—" for "or": $A \to w_1|w_2|\ldots|w_k$ has the same meaning as $A \to w_1$, $A \to w_2$, ..., $A \to w_k$. For example, the following grammar defines natural numbers in decimal notation:
  $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $V = \{N, D\}$, with start variable N.
  $N \to 0|1D|2D|3D|4D|5D|6D|7D|8D|9D$
  $D \to \lambda|0D|1D|2D|3D|4D|5D|6D|7D|8D|9D$
  Note that this grammar avoids any number except for 0 starting with 0, and does not allow an empty number. A string is generated by a given grammar if it can be obtained by repeatedly applying the rules (represented by a parse tree); a language recognized by a grammar is the set of all strings generated by it. If there is a context-free grammar recognizing a given language, that language is called *context-free*.

- Since languages are sets, we talk about union, intersection and complement of a language (complement with Σ^* as a universe). Two additional operations are *concatenation* (somewhat similar to Cartesian product): if L_1 and L_2 are two languages, then the concatenation language $L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$. That is, it is a set of all strings consisting of a string from the first language concatenated with a string from the second language. Note that concatenation with λ does not change a string.

- The last operation on languages is called a *star* operation, or *Kleene star*. We define star operation recursively. Let L be a language. Define $L^0 = \{\lambda\}$ (set consisting of only the empty string), $L^1 = L$ and, recursively, $L^{n+1} = LL^n$ (that is, every string in L^{n+1} is a concatenation of $n+1$ strings from L). Now L^* is a union, for all n, of languages L^n. That is, a string is in L^* if it consists of zero (then it is λ) or more strings from L concatenated together.

- A *regular expression* is any expression defined from the symbols of an alphabet Σ by using concatenation, union and star operations. An empty set is a regular expression, so is λ. For example, a regular expression defining all strings over the binary alphabet ($\{0, 1\}$) that have 1 as their second symbol and 0 as their last symbol is $(0 \cup 1)1(0 \cup 1)^*0$. For example, a string 010001110 will match this regular expression, but the string 0010010110 or string 1 will not. A language L is called *regular* if there is a regular expression such that all and only strings in L are strings matched by this regular expression. There are languages that are not regular: for example, the language a^nb^n over the alphabet $\Sigma = \{a, b\}$, consisting of strings $\{\lambda, ab, aabb, aaabbb, \dots\}$.

- A *finite automaton*, formally, consists of a set of *states* (denoted as circles in a diagram), an input alphabet Σ (which provides the labels for the arrows), transition function that describes from which state on which input symbol which state is reached (drawn as arrows between states labeled by input symbols). There is one special state called *start state* (which has a little arrow pointing to it); this is the state from which the computation starts. There can be zero or more *accepting states*, which are denoted by double circles. If a computation finished in an accepting state, then we say that the automaton accepted its input string, otherwise it rejected that string. The set of all input strings accepted by an automaton is called a language of that automaton. If there is exactly one transition from every state on every symbol of the alphabet, the automaton is called *deterministic* (a DFA); otherwise it is non-deterministic (NFA). The automaton on this picture accepts the set of strings with an odd number of 1s; "even" is the start state and "odd" is the only accepting state.



- A set of languages accepted by finite automata is exactly the regular languages. That is, for every regular language there is an automaton that accepts it, and a language of any automaton can be described by a regular expression. Note that there can be different automata and different regular expressions accepting the same language.

- A *Turing machine* is like a finite automaton except it also has memory. the *Church-Turing thesis* states that everything that can be computed by any kind of computing procedure can be computed by a Turing machine. Though Turing machine cannot solve everything (for example, they cannot solve the Halting problem), they can compute more than context-free languages (and some context-free languages are not regular).

Combinatorics and probability

- *Rules of Sum and Product*: Choosing either one out of n *or* one out of m can be done $n + m$ ways. Choosing one out of n *and* one out of m can be done $n \cdot m$ ways.

- *Permutations*: The number of sequences of n distinct objects. Without repetition: $n!$, with repetition: n^k, where k is the length of the sequence.

- *Combinations*: The number of ways to choose k objects from n objects without repetition.

$$\text{Without order}: \quad C(n,k) = \binom{n}{k} = \frac{n!}{(n-k)!k!} \qquad \text{With order:} \quad P(n,k) = \frac{n!}{(n-k)!}$$

- *Combinations with repetition*: The number of ways to choose k elements out of n possibilities.

  Combinations of k elements from n categories ($n-1$ "dividers"'): $\binom{k+(n-1)}{k} = \frac{(k+(n-1))!}{k!(n-1)!}$.

- *Binomial theorem.* For a non-negative integer n, $(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i$

- *Pascal's identity:*

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$$

- *Pascal's triangle:* each row contains binomial coefficients for the power binomial expansion. Each coefficient is the sum of two above it (above-right and above-left), using Pascal's identity.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

- Other identities and corollaries of the binomial theorem:

$$\binom{n}{k} = \binom{n}{n-k} \qquad \sum_{i=1}^{n} \binom{n}{i} = 2^n \qquad \sum_{i=1}^{n} (-1)^i \binom{n}{i} = 0$$

- A *sample space* S is a set of all possible *outcomes* of an *experiment* ({heads, tails} for a coin toss, {1,2,3,4,5,6} for a die throw). An *event* is a subset of the sample space. If all outcomes are equally likely, probability of each is $1/|S|$ (uniform distribution). Otherwise, sum of probabilities of all outcomes is 1, and probability of each is between 0 and 1: probability distribution on the sample space. A probability of an event $Pr(A) = \Sigma_{a \in A} Pr(a)$. $Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$, so if events A and B are disjoint, then $Pr(A \cup B) = Pr(A) + Pr(B)$.

- *Birthday paradox*: about 23 people enough to have 1/2 probability that two have birthday the same day (if birthdays are uniformly random days of the year).

- *Conditional probability*: $Pr(A|B) = Pr(A \cap B)/Pr(B)$. If $Pr(A \cap B) = Pr(A) \cdot Pr(B)$, events A and B are *independent*. Better to switch the door in Monty Hall puzzle.

- *Bayes theorem:* $Pr(B|A) = Pr(A|B) \cdot Pr(B)/Pr(A) = Pr(A|B) \cdot Pr(B)/(Pr(A|B) \cdot Pr(B) + Pr(A|\bar{B}) \cdot Pr(\bar{B})$. Generalizes to partition into arbitary many events rather than just B and $\bar{B}$. For a medical test, let A be an event that the test came up positive, and B that a person is sick. If this medical test has a false positive rate $Pr(A|\bar{B})$ (healthy mistakenly labeled sick, specificity $1 - Pr(A|\bar{B})$) and false negative rate $Pr(\bar{A}|B)$ (sick labeled healthy, sensitivity $1 - Pr(\bar{A}|B)$, and probability of being sick is $Pr(B)$, then probability of a person being sick if the test came up positive is $Pr(B|A) = Pr(A|B) \cdot Pr(B)/Pr(A)$, where $Pr(A) = Pr(A|B) \cdot Pr(B) + Pr(A|\bar{B}) \cdot Pr(\bar{B})$.

- *Expectaton*: let X be a random variable for some event over a sample space $\{a_1, \ldots, a_n\}$ (e.g., X is the number of coin tosses that came up heads, amount won in a lottery or X is 1 iff some event happened (indicator variable)). Then $E(X) = \Sigma_{i=1}^{n} a_k Pr(X(a_k))$ (if outcomes are numbers, often write $X = a_k$ in the equation). *Linearity of expectation*: $E(X_1 + X_2) = E(X_1) + E(X_2)$, and $E(aX + b) = aE(X) + b$. Example: hat check problem.