Predicate logic:

- A *predicate* is like a propositional variable, but with *free variables*, and can be true or false depending on the value 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 $\neg \forall x P(x) \iff \exists x \neg P(x)$ and $\neg \exists x P(x) \iff \forall x \neg P(x)$.

- *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.

- *Reasoning in predicate logic* The *rule of universal instantiation* says that if some prperty 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, it is possible to check its truth value using Resolution method. For that, the formula is 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, and resolution can be used as in the propositional case.

- *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).

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*: $A = \{x\ inU|P(x)\}$ 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. A *power set* for a given set A, denoted 2^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). A power set is always larger than the original set, even in the infinite case (use diagonalization to prove that).

- 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 $A \times B = \{(a, b)|a \in A \text{ and } b \in B\}$.

- 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 $A \cup B$, $|A \cup B| = |A| + |B| - |A \cap B|$.

- *Zermelo-Fraenkel set theory* The foundations of mathematics, that is, the axioms from which all the mathematics is derived are several axioms about sets called Zermelo-Fraenkel set theory (together with the axiom of choice abbreviated as ZFC). For example, numbers can be defined from sets as follows: 0 is $\emptyset$. 1 is $\{\emptyset\}$. 2 is $\{\emptyset, \{\emptyset\}\}$ and in general n is $n-1 \cup \{n-1\}$. ZFC is carefully constructed to explicitly disallow *Russell's paradox* "if a barber shaves everybody who does not shave himself, who shaves the barber?" (in set theoretic terms, if X is a set of sets A such as $A \notin A$, is $X \in X$?) This kind of reasoning is used to prove that *halting problem* of checking whether a given piece of code contains an infinite loop is not solvable (an alternative way to prove this is diagonalization).

- **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 $^-, +, \cdot$.

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

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

Relations and Functions

1. 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.

2. 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, $a = b$, $a \iff b$.
   - *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.

3. *Cryptography* A relation used for cryptography is $a \equiv b \mod n$ relation, meaning $\exists k \in \mathbb{Z} \; a = b + kn$. Caesar cipher: if letters of the alphabet are numbered from $0 to 25$, then a letter i is encoded by the

letter j with the number $j \equiv i + 3 \mod 26$. Here, instead of 3 one can take any other number. To decode, take $i = j - 3 \mod 26$. In private-key cryptography, the code is obtained by doing a XOR of the binary string message with the binary string key. To decode, do another XOR with the key. In public-key cryptography such as RSA there are two parts of the key: the public part that gets published and is used to encode a message, and a private part used to decode the message. In this way, the sender does not need to know the secret in order to send message securely. The RSA algorithm uses $\mod n$ relation in to do encoding and decoding: you don't need to know the details of this for this course. Just note that the security of RSA is based on the assumption that given a product pq of two large prime numbers p and q there is no efficient way to find out the values of p and q separately. This is an open problem, whether it is possible to do (possible on quantum computers, but they don't handle large enough numbers).

4. A **function** $f\colon A \to B$ 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, $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$ 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*.

5. **Comparing set sizes**
   Theorem: 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).