Discrete Math Review

Last Updated: August 8th, 2026

1 Sets

Definition 1.1. A **set** represents a collection of items, where each item is called a **member** or **element** of the set.

List Notation the most common way to represent a set is to list the set members one-by-one, and delimit the list with curly braces.

For example,

$$\{2, 3, 5, 7, 11\}$$

uses list notation to describe the set consisting of all prime numbers that do not exceed 11. Note that the order in which the members are listed does not matter. Indeed the sets $\{2, 3, 5, 7, 11\}$ and $\{3, 11, 5, 2, 7\}$ are two different ways of writing the same set. Also, each member occurs only *once* in the set, meaning that no item can be listed more than once. Note: a **multiset** is a set for which each member may occur more than once. When using list notation to write a multiset, then we list an item a number of times that is equal to its number of occurrences in the set. For example, $\{1, 2, 2, 3, 3, 3\}$ is a multiset for which 1 occurs once, 2 occurs twice, 3 occurs thrice.

Informal List Notation uses **ellipsis** ... to indicate that a pattern is to be continued in the list, either indefinitely or up to some value.

Common Numerical Sets

Natural Numbers $\mathcal{N} = \{0, 1, 2, \ldots\}$

Integers $\mathbb{I} = \{0, \pm 1, \pm 2, \ldots\} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$

Rational Numbers (i.e. Fractions) $\mathbb{Q} = \{p/q \mid p, q \in \mathbb{I} \wedge q \neq 0\}$

Empty Set the set having no members and denoted by $\emptyset$.

Membership Symbol $x \in A$ indicates that item x is a member of set A, while $x \notin A$ means that x is not a member of A.

Containment Symbol $A \subseteq B$ means that A is a **subset** of B. In other words, every member of A is also a member of B. Note: trivially, $\emptyset \subseteq B$ for every set B, but $A \subseteq \emptyset$ is only true when $A = \emptyset$.

Proper Containment Symbol $A \subset B$ means that A is a **proper subset** of B, meaning that there is some member of B that is not a member of A. For example, $\mathcal{N} \subset \mathbb{Q}$ since there is a rational number, e.g. $\frac{1}{2}$, that is not an natural number. Similarly,

$$\mathcal{N} \subset \mathbb{I} \subset \mathbb{Q}.$$

Set Equality $A = B$ iff $A \subseteq B$ and $B \subseteq A$ are both true statements.

Set Cardinality $|A|$ denotes the number of items of A. We also refer to $|A|$ as the size of A. Note: $|\mathcal{N}| = |\mathbb{I}| = |\mathbb{Q}| = \infty$.

Powerset of a set A, denoted $\mathcal{P}(A)$, is the set consisting of all subsets of A. For example, if $A = \{1, 2, 3\}$, then

$$\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

In general,

$$|\mathcal{P}(A)| = 2^{|A|}$$

since there is a one-to-one correspondence between subsets of A and binary strings of length $|A|$.

Cartesian Product Given sets A and B, $A \times B$ is the set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$. For example, if $A = \{1, 2, 3\}$ and $B = \{a, b\}$, then

$$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}.$$

Example 1.2. Which of the following are true statements?

a. $63 \in \{2, 3, 5, 7, 11, \ldots, 97\}$

b. $39 \notin \{2, 3, 5, 7, 11, \ldots, 97\}$

c. $\{3\} \in \{\emptyset, \{1\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$

d. $3 \notin \{\emptyset, \{1\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$

e. $\{7, 23, 59\} \subset \{2, 3, 5, 7, 11, \ldots, 97\}$

f. $\{7, 23, 59\} \not\subseteq \{2, 3, 5, 7, 11, \ldots, 97\}$

g. $\{3\} \in \{\emptyset, \{1\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$

h. $\{3\} \subset \{\emptyset, \{1\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$

i. $|\{\emptyset\}| = 0$

j. $|\emptyset| = 0$

k. $|\{\emptyset, \{1\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}| = 6$

□

2 Functions

Definition 2.1. A function $f : A \to B$ is a set A, a set B, and a rule that assigns each member $a \in A$ to exactly one member $b \in B$. We write this as $f(a) = b$.

Some terminology related to function $f : A \to B$

Function Name f is an identifier that names the function

Domain set A is the set of **inputs** that get assigned by f

Co-Domain set B is the set of possible **outputs** to which an input can be assigned by f. Simply put, $a \in A$ is the input and $f(a) = b \in B$ is the output associated with a.

Example 2.2. Consider the sequence of numbers $24, 43, 87, 86, 68, 62, 51, 17$ and the function

$$\text{order} : A \to B,$$

where $A = \{17, 24, 43, 51, 62, 68, 86, 87\}$, $B = \{1, 2, 3, 4, 5, 6, 7, 8\}$, and order$(x)$ equals the order that x appears in the above sequence. Provide a **function table** that shows the correspondence between each input x and its corresponding output order(x) for every $x \in A$.

$a \in A$	order$(a) \in B$
17	8
24	1
43	2
51	7
62	6
68	5
86	4
87	3

Example 2.3. Consider the sets $A = \{-9, -6, 2, 5\}$ and $B = \{-1, 8\}$. Let $C = A \times B$ and define the function $\texttt{prod} : C \to \mathbb{I}$ by $\texttt{prod}(a, b) = a \cdot b$, the product of a with b. Provide a function table for prod.

$C = \{(-9,-1), (-9,8), \ldots, (5,-1), (5,8)\}$

prod(a,b): a \ b	-1	8
-9	9	-72
-6	6	-48
2	-2	16
5	-5	40

onto: every codomain member is mapped to

one-to-one: every codomain member is mapped to at most once

$\text{prod}(-6, 8) = -48$

3 Propositional Logic

3.1 Boolean Variables and Assignments

Boolean Variable A variable is said to be **Boolean** iff its domain equals $\{0, 1\}$. We use lowercase letters, such as $x, y, z, x_1, x_2, \ldots$, etc., to denote a Boolean variable.

Assignment An **assignment** over a set of V of Boolean variables is a function $\alpha : V \to \{0, 1\}$ that assigns to each variable $x \in V$ a value in $\{0, 1\}$. We may represent α using function notation, or as a labeled tuple.

Example: for the assignment α that assigns 1 to both x_1 and x_2, and 0 to x_3, we may use function notation and write $\alpha(x_1) = 1$, $\alpha(x_2) = 1$, and $\alpha(x_3) = 0$, or we may use tuple notation and write

$$\alpha = (x_1 = 1, x_2 = 1, x_3 = 0),$$

or

$$\alpha = (1, 1, 0),$$

if the associated variables are understood.

Variable Negation If x is a Boolean variable, then $\overline{x}$ is called its **negation**.

Example: Suppose assignment α satisfies $\alpha(x_1) = 0$. Then (extending α to include negation inputs) $\alpha(\overline{x}_1) = 1$.

Literal A **literal** is either a variable or the negation of a variable .

Example: x_1, x_3, $\overline{x}_3$, are $\overline{x}_5$ all examples of literals.

Consistent A set R of literals is said to be **consistent** iff no variable and its negation are both in R. Otherwise, R is said to be **inconsistent**.

Example: $\{x_1, \overline{x}_2, x_4, \overline{x}_7, \overline{x}_9\}$ is a consistent set, but $\{x_1, \overline{x}_2, x_4, \overline{x}_7, x_7\}$ is an inconsistent set.

Induced Assignment If $R = \{l_1, \ldots, l_n\}$ is a consistent set of literals, then α_R is called the **(partial) assignment induced by R** and is defined as $\alpha_R(x) = 1$ if $x \in R$, $\alpha_R(x) = 0$ if $\overline{x} \in R$, and $\alpha_R(x)$ is undefined for any for which $x, \overline{x} \notin R$.

Example: the assignment induced by $R = \{x_1, \overline{x}_2, x_4, \overline{x}_7, \overline{x}_9\}$ is

$$\alpha_R = (x_1 = 1, x_2 = 0, x_4 = 1, x_7 = 0, x_9 = 0).$$

3.2 Logical Operations

Connective	**Symbol**	**Example**
NOT	-	"It is **not** raining outside."
AND	$\wedge$	"It is raining **and** the sidewalk is wet."
OR (Inclusive)	$\vee$	"You may board with either a passport **or** state-issued ID."
OR (Exclusive)	$\oplus$	"He was born in either Phoenix **or** Tuscon."
IF-THEN	$\rightarrow$	"**If** it rains **then** the sidewalk gets wet."
EQUIVALENCE	$\leftrightarrow$	"Passing the exam is **equivalent** to scoring at least 75 points."

3.3 Truth Tables

Each logical operation may be formally defined as a function that takes one or two Boolean values as input, and outputs a Boolean value. Below are function tables (also called truth tables) for each of the operations.

x	$\overline{x}$
0	1
1	0

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

x	y	$x \leftrightarrow y$
0	0	1
0	1	0
1	0	0
1	1	1

We now comment on the $x \to y$ truth table.

1. x being true is a cause for y to be true. Hence $1 \to 1$ is true, while $1 \to 0$ is false.
2. However, if $x = 0$ then it is OK if either $y = 0$ or $y = 1$.
3. Example: if x evaluates the truth of the statement "it is raining", and y evaluates "the sidewalk is wet", then $0 \to 0$ is true since it is common for a sidewalk to be dry when it is not raining. Furthermore, $0 \to 1$ is also true, since it might not be raining, but the sprinklers could be getting the sidewalk wet.
4. Therefore, $x \to y$ means that $x = 1$ causes $y = 1$, but it may not be the only possible cause.

Definition 3.1. Similar to an arithmetic expression that uses variables, numbers, parentheses, and the set of operations $A = \{+, -, \times, \div, \text{ mod }\}$, a **Boolean expression**, also referred to as a **Boolean formula**, is the same as an arithmetic expression, but with numbers replaced with Boolean values, and A replaced with $B = \{\text{-}, \wedge, \vee, \oplus, \rightarrow, \leftrightarrow\}$.

Example 3.2. Show some examples of both arithmetic and Boolean expressions.

4 Computational Problems

Informally, when we think of a problem, we think of a situation that needs to be resolved (i.e. solved). Moreover, in computer science we think of a computing problem as having the following properties.

Definition 4.1. A **computing problem** is a collection of situations that share a common theme, and each of which requires a solution.

- Each situation is referred to as a **problem instance**, and represents a concrete example of the general problem.
- Each problem instance can be represented by a unique word over some alphabet, meaning that no two instances map to the same word. The word may then be encoded into a binary word so that the problem instance can be stored in computer memory.

Three types of computational problems in relation to this course

Decision The solution to a problem instance is either Yes or No, equivalently True (1) or False (0). An instance x for which the solution is 1 (respectively, 0) is called a **positive instance** (respectively, **negative instance**).

Optimization The solution to a problem instance x is a number that represents the greatest (or least) quantity of some entity that is associated with x

Miscellaneous Any computation problem that is neither a decision nor an optimization problem

Example 4.2. For each of the following problems, determine if it is a decision, optimization, or miscellaneious problem.

a. An instance of the `Prime` problem is a natural number $n \geq 0$, and the problem is to decide if n is a prime number, meaning that its only natural divisors are 1 and n.

b. An instance of the `Maximum Subsequence Sum (MSS)` is a sequence (array) a of integers. The problem is to determine the greatest sum that can be made by any subsequence of a. For example, determine the maximum subsequence sum for any subsequence of

$$3, -4, 2, 5, -2, -4, 0, 2, 3, -2, 1.$$

c. An instance of `Sort` is an array a of integers. The problem is to produce another array b whose members are the members of a, but in sorted order.

d. An instance of `Fallible` is a Boolean formula F. Is there an assignment that can be made to the variables of F so that F evaluates to 0? For example, given the logical formula $F(x, y) = x \to (y \to x)$, can x and y be assigned Boolean values that force F to evaluate to 0?

Problems as Sets

Notice that every problem is given a name consisting of one or more words along with an associated acronym if necessary. We often write a problem's name to represent the *set* of all problem instances. For example,

$$f : \texttt{Sort} \to \{0, 1\}$$

is a function whose inputs are instances of problem `Sort`, and whose outputs are 0 or 1. For example, we may define $f(a) = 1$ iff integer array a is a sorted array, and $f(a) = 0$ otherwise.