# Discrete Math for Computer Science

Peter Schaefer

Freshman Fall

## Contents

# 1 Logic

## 1.1 Propositions and Logical Operations

**Proposition**: a statement that is either true or false.

Some examples include "It is raining today" and "$3 \cdot 8 = 20$".

However, not all statements are propositions, such as "open the door"

| **Name** | **Symbol** | **alternate name** |
|---|---|---|
| NOT | $\neg$ | negation |
| AND | $\wedge$ | conjunction |
| OR | $\vee$ | disjunction |
| XOR | $\oplus$ | exclusive or |

| $p$ | $q$ | $\neg p$ | $p \wedge q$ | $p \vee q$ | $p \oplus q$ |
|---|---|---|---|---|---|
| T | T | F | T | T | F |
| T | F | F | F | T | T |
| F | T | T | F | T | T |
| F | F | T | F | F | F |

XOR is very useful for encryption and binary arithmetic.

## 1.2 Evaluating Compound Propositions

$p$ : The weather is bad. $p \wedge q$ : The weather is bad *and* the trip is cancelled

$q$ : The trip is cancelled. $p \vee q$ : The weather is bad *or* the trip is cancelled

$r$ : The trip is delayed. $p \wedge (q \oplus r)$ : The weather is bad *and* either the trip is cancelled *or* delayed

**Order of Evaluation** $\neg$, then $\wedge$, then $\vee$, but parenthesis always help for clarity.

Example Truth Table:

| $p$ | $q$ | $p \wedge q$ | $\neg q$ | $(p \wedge q) \oplus \neg q$ |
|---|---|---|---|---|
| T | T | T | F | T |
| T | F | F | T | T |
| F | T | F | F | F |
| F | F | F | T | T |

## 1.3 Conditional Statements

$p \implies q$ where $p$ is the hypothesis and $q$ is the conclusion

| Format | Terminology |
|---|---|
| $p \implies q$ | given |
| $\neg q \implies \neg p$ | contrapositive |
| $q \implies p$ | converse |
| $\neg p \implies \neg q$ | inverse |

given $p \implies q \equiv \neg q \implies \neg p$ contrapositive

inverse $\neg p \implies \neg q \equiv q \implies p$ converse

| $p$ | $q$ | $p \implies q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

$p$ is a sufficient condition for $q$

$q$ is a necessary condition for $p$

| Phrase | Logic |
|---|---|
| $q$ if $p$ | $p \implies q$ |
| $q$ only if $p$ | $q \implies p$ |
| $q$ if and only if $p$ | $p \iff q$ |

**Order of Operations**: $p \wedge q \implies r \equiv (p \wedge q) \implies r$

## 1.4 Logical Equivalence

**Tautology**: a proposition that is always true **Contradiction**: a proposition that is always false

**Logically equivalent**: same truth value regardless of the truth values of their individual propositions

**DeMorgan's Laws**:

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$
$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

Verbally,
It is not true that the patient has migraines *or* high blood pressure ≡
≡ The patient does not have migraines *and* does not have high blood pressure
It is not true that the patient has migraines *and* high blood pressure ≡
≡ The patient does not have migraines *or* does not have high blood pressure

## 1.5 Laws of Propositional Logic

You can use **substitution** on logically equivalent propositions.

| **Law Name** | $\vee$ or | $\wedge$ and |
|---|---|---|
| Idempotent | $p \vee p \equiv p$ | $p \wedge p \equiv p$ |
| Associative | $(p \vee q) \vee r \equiv p \vee (q \vee r)$ | $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ |
| Commutative | $p \vee q \equiv q \vee p$ | $p \wedge q \equiv q \wedge p$ |
| Distributive | $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ | $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ |
| Identity | $p \vee \mathrm{F} \equiv p$ | $p \wedge \mathrm{T} \equiv p$ |
| Domination | $p \vee \mathrm{T} \equiv \mathrm{T}$ | $p \wedge \mathrm{F} \equiv \mathrm{F}$ |
| Double Negation | $\neg\neg p \equiv p$ | |
| Complement | $p \vee \neg p \equiv \mathrm{T}$ | $p \wedge \neg p \equiv \mathrm{F}$ |
| DeMorgan | $\neg(p \vee q) \equiv \neg p \wedge \neg q$ | $\neg(p \wedge q) \equiv \neg p \vee \neg q$ |
| Absorption | $p \vee (p \wedge q) \equiv p$ | $p \wedge (p \vee q) \equiv p$ |
| Conditional | $p \implies q \equiv \neg p \vee q$ | $p \iff q \equiv (p \implies q) \wedge (q \implies p)$ |

## 1.6 Predicates and Quantifiers

**Predicate**: a logical statement where truth value is a function of a variable.

P($x$): $x$ is an even number. P(5): false P(2): true

**Domain**: the set of all possible values for a variable in a predicate.

Ex. $\mathbb{Z}^+$ is the set of all positive integers.
*If domain is not clear from context, it should be given as part of the definition of the predicate.

**Quantifier**: converts a predicate to a proposition.

| Quantifier | Symbol | Meaning |
|---|---|---|
| Universal | $\forall$ | "for all" |
| Existential | $\exists$ | "there exists" |

$\exists x(x + 1 < x)$ is false.

**Counter Example**: universally quantified statement where an element in the domain for which the predicate is false. Useful to prove a $\forall$ statement false.

## 1.7 Quantified Statements

Consider the two following two predicates:

$$P(x) : x \text{ is prime, } x \in \mathbb{Z}^+$$
$$O(x) : x \text{ is odd}$$

Proposition made of predicates: $\exists x(\mathrm{P}(x) \wedge \neg \mathrm{O}(x))$
Verbally: there exists a positive integer that is prime but is not odd.

**Free Variable**: a variable that is free to be any value in the domain.
**Bound Variable**: a variable that is bound to a quantifier.

P($x$): $x$ came to the party
S($x$): $x$ was sick

$\mathrm{P}(x) \stackrel{?}{\equiv} \neg \mathrm{S}(x)$
$\mathrm{P}(x) \not\equiv \neg \mathrm{S}(x)$

| | P($x$) | S($x$) | $\neg$S($x$) |
|---|---|---|---|
| Joe | T | F | T |
| Theo | F | T | F |
| Gert | T | F | T |
| Sam | F | F | T |

## 1.8 DeMorgan's law for Quantified Statements

Consider the predicate: $\mathrm{F}(x)$ : "$x$ can fly", where $x$ is a bird. According to the DeMorgan Identity for Quantified Statements,

$$\neg\forall x\mathrm{F}(x) \equiv \exists x\neg\mathrm{F}(x)$$
"not every bird can fly $\equiv$ "there exists a bird that cannot fly

Example using DeMorgan Identities:

$$\begin{aligned}\neg\exists x(\mathrm{P}(x) \implies \neg\mathrm{Q}(x)) &\equiv \forall x\neg(\mathrm{P}(x) \implies \neg\mathrm{Q}(x)) \\ &\equiv \forall x(\neg\neg\mathrm{P}(x) \wedge \neg\neg\mathrm{Q}(x)) \\ &\equiv \forall x(\mathrm{P}(x) \wedge \mathrm{Q}(x))\end{aligned}$$

## 1.9 Nested Quantifiers

A logical expression with more than one quantifier that binds different variables in the same predicate is said to have **Nested Quantifiers**.

| Logic | Variable Boundedness |
|---|---|
| $\forall x\exists y\ \mathrm{P}(x,y)$ | $x$, $y$ bound |
| $\forall x\ \mathrm{P}(x,y)$ | $x$ bound, $y$ free |
| $\exists x\exists y\ \mathrm{T}(x,y,z)$ | $x$, $y$ bound, $z$ free |

| Logic | Meaning |
|---|---|
| $\forall x\forall y\ \mathrm{M}(x,y)$ | "everyone sent an email to everyone" |
| $\forall x\exists y\ \mathrm{M}(x,y)$ | "everyone sent an email to someone" |
| $\exists x\forall y\ \mathrm{M}(x,y)$ | "someone sent an email to everyone" |
| $\exists x\exists y\ \mathrm{M}(x,y)$ | "someone sent an email to someone" |

There is a two-player game analogy for how quantifiers work:

| Player | Action | Goal |
|---|---|---|
| Existential Player $\exists$ | selects value for existentially-bound variables | tries to make expression <u>true</u> |
| Universal Player $\forall$ | selects value for universally-bound variables | tries to make expression <u>false</u> |

Consider the predicate $\mathrm{L}(x,y)$ : "$x$ likes $y$".

$\exists x\forall y\mathrm{L}(x,y)$ means "there is a student who likes everyone in the school".
$\neg\exists x\forall y\mathrm{L}(x,y)$ means "there is no student who likes everyone in the school".

After applying DeMorgan's Laws,

$\forall x\exists y\neg\mathrm{L}(x,y)$ means "there is no student who likes everyone in the school".

## 1.10 More Nested Quantifiers

$\mathrm{M}(x,y)$ : "x sent an email to y". Consider $\forall x\forall y$ M $(x,y)$. It means that "email sent an email to everyone including themselves". Using $(x \neq y \implies \mathrm{M}(x,y))$ can fix this quirk.

$\forall x\forall y(x \neq y \implies \mathrm{M}(x,y))$ means "everyone sent an email to everyone else

### 1.10.1 Expressing Uniqueness in Quantified Statements

Consider $\mathrm{L}(x)$: $x$ was late to the meeting. If someone was late to the meeting, how could you express that that someone was the only person late to the meeting? You want to express that there is someone where everyone else was not late, which can be done with

$$\exists x(\mathrm{L}(x) \wedge \forall y(x \neq y \implies \neg\mathrm{L}(y)))$$

### 1.10.2 Moving Quantifiers in Logical Statements

Consider $\mathrm{M}(x, y)$: "$x$ is married to $y$" and $\mathrm{A}(x)$: "$x$ is an adult". One way of expressing "For every person $x$, if $x$ is an adult, then there is a person $y$ to whom $x$ is married to" is by this statement:

$$\forall x(\ \mathrm{A}(x) \implies \exists\ \mathrm{M}(x, y))$$

Since $y$ does not appear in $\mathrm{A}(x)$, "$\exists y$" can be moved so that it appears just after the "$\forall$", resulting with

$$\forall x \exists y(\ \mathrm{A}(x) \implies\ \mathrm{M}(x, y))$$

When doing this, keep in mind that $\forall x \exists y \not\equiv \exists y \forall x$:

$\forall x \exists y(\ \mathrm{A}(x) \implies\ \mathrm{M}(x, y))$ means
for every $x$, if $x$ is an adult, there exists $y$ who is married to $x$.
$\exists y \forall x(\ \mathrm{A}(x) \implies\ \mathrm{M}(x, y))$ means
There exists a $y$, such that every $x$ who is an adult is also married to $y$

## 1.11 Logical Reasoning

**Argument**: a sequence of propositions, called hypothesis, followed by a final proposition, called the conclusion.

An argument is **valid** if the conclusion is true whenever the hypothesis are all true, otherwise the argument is **invalid**.

An argument is denoted as: $\begin{array}{l} p_1 \\ p_2 \\ \vdots \\ \underline{p_n} \\ \therefore c \end{array}$ where $\begin{array}{l} p_1, p_2, \ldots, p_n \text{ are hypothesis} \\ c \text{ is the conclusion} \end{array}$

The argument is valid whenever the proposition $(p_1 \wedge p_1 \wedge \cdots \wedge p_n) \implies c$ is a tautology. Additionally, because of the commutative law, hypothesis can be reordered without changing the argument.

$$\begin{array}{l} p \\ \underline{p \implies q} \\ \therefore q \end{array} \equiv \begin{array}{l} p \implies q \\ \underline{p} \\ \therefore q \end{array}$$

### 1.11.1 The Form of an Argument

$$\begin{array}{l} \text{It is raining today.} \\ \underline{\text{If it is raining today, then I will not ride my bike to school.}} \\ \therefore \text{I will not ride my bike to school.} \end{array} \qquad \begin{array}{l} p \\ \underline{p \implies q} \\ \therefore q \end{array}$$

The argument is valid because its form, $\begin{array}{l} p \\ \underline{p \implies q} \\ \therefore q \end{array}$ is an valid argument.

$$\begin{array}{l} \text{5 is not an even number.} \\ \underline{\text{If 5 is an even number, then 7 is an even number.}} \\ \therefore \text{7 is not an even number.} \end{array} \qquad \begin{array}{l} p \\ \underline{p \implies q} \\ \therefore q \end{array}$$

The argument is invalid because its form, $\begin{array}{l} \neg p \\ \underline{p \implies q} \\ \therefore \neg q \end{array}$ is an invalid argument.

## 1.12 Rules of Inference with Propositions

Using truth tables to establish the validity of an argument can become tedious, especially if an argument uses a large number of variables.

| | | | |
|---|---|---|---|
| $p$<br>$p \implies q$<br>$\therefore q$ | Modus Ponens | $p$<br>$q$<br>$\therefore p \wedge q$ | Conjunction |
| $\neg q$<br>$p \implies q$<br>$\therefore \neg p$ | Modus Tollens | $p \implies q$<br>$q \implies r$<br>$\therefore p \implies r$ | Hypothetical Syllogism |
| $p$<br>$\therefore p \vee q$ | Addition | $p \vee q$<br>$\neg p$<br>$\therefore q$ | Disjunctive Syllogism |
| $p \wedge q$<br>$\therefore p$ | Simplification | $p \implies q$<br>$q \implies r$<br>$\therefore q \vee r$ | Resolution |

Example expressed in English:

| | |
|---|---|
| If it is raining or windy or both, the game will be cancelled.<br>The game will not be cancelled<br>$\therefore$ It is not windy. | $(r \vee w) \implies c$<br>$\neg c$<br>$\therefore \neg w$ |

Steps to Solve:

| | | |
|---|---|---|
| $(r \vee w) \implies c$ | Hypothesis | (1) |
| $\neg c$ | Hypothesis | (2) |
| $\neg(r \vee w)$ | Modus Tollens: 1, 2 | (3) |
| $\neg r \wedge \neg w$ | DeMorgan's Law: 3 | (4) |
| $\neg w \wedge \neg r$ | Commutative Law: 4 | (5) |
| $\neg w$ | Simplification: 5 | (6) |

## 1.13 Rules of Inference with Quantifiers

In order to apply the rules of quantified expressions, such as $\forall x \neg(\ \mathrm{P}(x) \wedge\ \mathrm{Q}(x))$, we need to remove the quantifier by plugging in a value from the domain to replace the variable x.

For example:

| | |
|---|---|
| Every employee who received a large bonus works hard.<br>Linda is an employee at the company.<br>Linda received a large bonus.<br>$\therefore$ Some employee works hard. | $\forall x(\mathrm{B}(x) \implies \mathrm{H}(x))$<br>Linda $\in x$<br>B(Linda)<br>$\therefore \exists x\ \mathrm{H}(x)$ |

**Arbitrary Element**: has no special properties other than those shared by all elements of the domain.

**Particular Element**: may have special properties that are not shared by all the elements of the domain. For example, if the domain is the set of all integers, $\mathbb{Z}$, a particular element is 3, because it is odd, which is not true for all integers.

**Rules of Inference for Quantified Statements**

| | | | |
|---|---|---|---|
| $c$ is an element<br>$\forall x$ P $(x)$<br>$\therefore$ P$(c)$ | Universal Instantiation | $\exists x$ P$(x)$<br>$\therefore$ $c$ is particular $\wedge$ P $(c)$ | Existential Instantiation* |
| $c$ is arbitrary<br>P $(c)$<br>$\therefore \forall$ P$(x)$ | Universal Generalization | $c$ is an element<br>P$(c)$<br>$\therefore \exists x$ P$(x)$ | Existential Generalization |

*Each use of Existential Instantiation must define a new element with its own symbol or name.

### 1.13.1 Example of using the Laws of Inference for Quantified Statements

Consider the following argument:

$\forall x(\mathrm{P}(x) \vee \mathrm{Q}(x))$
3 is an integer
$\neg$ P(3)
$\therefore$ Q(3)

Steps to Solve:

| | | |
|---|---|---|
| $\forall x(\mathrm{P}(x) \vee \mathrm{Q}(x))$ | Hypothesis | (1) |
| 3 is an integer | Hypothesis | (2) |
| $(\mathrm{P}(3) \vee \mathrm{Q}(3))$ | Universal Instantiation: 1, 2 | (3) |
| $\neg$ P(3) | Hypothesis | (4) |
| Q(3) | Disjunctive Syllogism: 3, 4 | (5) |

### 1.13.2 Showing an Argument with Quantified Statements is Invalid

Consider the following argument:

$\exists x\mathrm{P}(x)$
$\exists x\mathrm{Q}(x)$
$\therefore \exists x(\mathrm{P}(x) \wedge \mathrm{Q}(x))$

Using a supposed domain $\{c, d\}$, with truth values of

| | P | Q |
|---|---|---|
| c | T | F |
| d | F | T |

, the argument is invalid.

# 2 Proofs

# 3 Sets

## 3.1 Sets and Subsets

A **set** is a collection of objects. Objects in a set are called **elements**. Order does not matter, and there are no duplicates.

Roster notation:

$$A = \{2, 4, 6, 10\}$$
$$B = \{4, 6, 10, 2\}$$
$$A = B$$

To show membership, use the $\in$ symbol. For example, $2 \in A$, while $7 \notin A$. The empty set, which contains nothing, typically uses the $\emptyset$ symbol, or $\{\}$. Sets can be finite, or infinite. **Cardinality** of a set is the number of elements in a set. For example, the cardinality of A is 4.

$$|A| = 4$$

Cardinality can be infinite. Consider the set of all the integers, $\mathbb{Z}$. $|\mathbb{Z}| = \infty$

$\mathbb{N}$ : set of natural numbers
$= \{0, 1, 2, 3, \ldots\}$

$\mathbb{Z}$ : set of integers
$= \{\ldots, -2, -1, 0, 1, 2, \ldots\}$

$\mathbb{B}$ : set of rational numbers
$= \{x | x = \frac{a}{b} \text{ where } a, b \in \mathbb{Z}, b \neq 0\}$

$\mathbb{R}$ : set of real numbers
$= \{x | x \text{ has a decimal representation}\}$

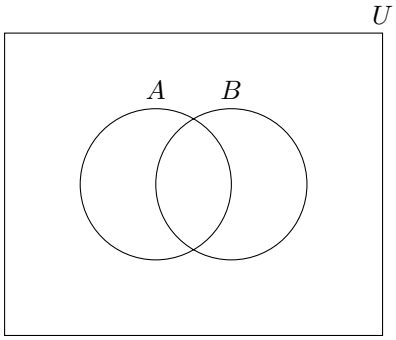
The subset operator is $\subseteq$

$$A \subseteq B \text{ if } \forall x(x \in A \implies x \in B)$$
$$A \subseteq A \text{ is true for any set}$$
$$\emptyset \subseteq A \text{ is true for any set}$$

Sometimes it is easier to define a set by defining properties that all the elements have. That is easy to do in **set builder notation**.

$$A = \{x \in S : A(x)\}, \text{ where } S \text{ is another set}$$
$$C = \{x \in \mathbb{Z} : 0 < x < 100 \text{ and } x \text{ is prime}\}.$$
$$D = \{x \in \mathbb{R} : |x| < 1\}$$

The **Universal Set**, usually called 'U', is a set that contains all elements mentioned in a particular context. For example, a discussion about certain types of real numbers, it would be understood that any element in the discussion is a real number. Sets are often represented pictorially with **Venn Diagrams**.



If $A \subseteq B$ and there is an element of $B$ that is not an element of $A$, meaning $A \neq B$, then $A$ is a **proper subset** of $B$, denoted as $A \subset B$. An important fact is that $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{B} \subset \mathbb{R}$

## 3.2 Sets of sets

Elements of sets can be sets themselves, consider $A = \{\{1,2\}, \emptyset, \{1,2,3\}, \{1\}\}$. The cardinality of $A$ is 4, $|A| = 4$. Additionally, $\{1,2\} \in A$, but $1 \notin A$.

The **Powerset** of A, denoted as $P(A)$ is the set of all subsets of $A$. For example,

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

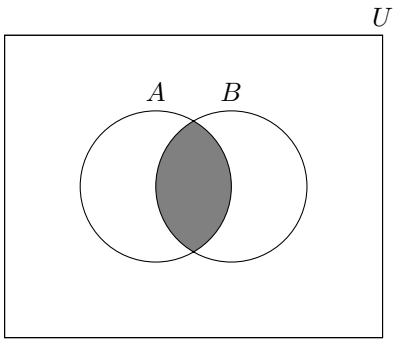
### 3.2.1 Cardinality of a Powerset

Let A be a finite set of cardinality $n$. Then the cardinality of the powerset of A is $2^n$.

$$|A| = n$$
$$|P(A)| = 2^n$$

## 3.3 Union and Intersection

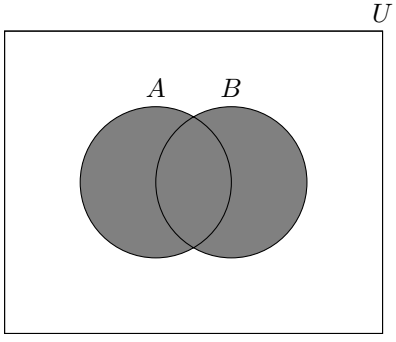
**Intersection** set operation: $\cap$. $A$ intersected with $B$ is defined to be the set containing elements which are in both $A$ and $B$. That is, $A \cap B = \{x : x \in A \wedge x \in B\}$.




Intersection $\cap$ can also apply to infinite sets:

$$A = \{x \in \mathbb{Z} : x \text{ is an integer multiple of } 2\}$$
$$B = \{x \in \mathbb{Z} : x \text{ is an integer multiple of } 3\}$$
$$A \cap B = \{x \in \mathbb{Z} : x \text{ is an integer multiple of } 6\}$$

**Union** set operation $\cup$. $A$ union with $B$ is defined to be the set containing elements which are in $A$ or $B$. That is, $A \cup B = \{x : x \in A \vee x \in B\}$.



A special notation, similar to $\sum$ or $\prod$ notation, allows for compound representation of the intersections or unions of a long sequence of sets.

$$\bigcap_{i=1}^{n} A_i = A_1 \cap A_2 \cap A_3 \cap \cdots \cap A_n = \{x : x \in A, \text{ for } \underline{\text{all}}\ 1 \leq i \leq n\}$$
$$\bigcup_{i=1}^{n} A_i = A_1 \cup A_2 \cup A_3 \cup \cdots \cup A_n = \{x : x \in A, \text{ for } \underline{\text{some}}\ 1 \leq i \leq n\}$$

Consider $A_j$ = a word with $j$ letters, with $U$ = is the Oxford English Dictionary.

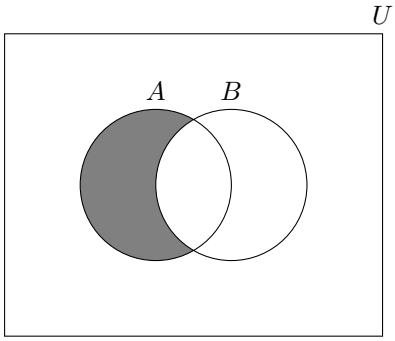
$\bigcup_{j=1}^{10} A_j =$ the set of all words with 10 letters or fewer in the OED

$\bigcap_{j=1}^{45} A_j = \emptyset$

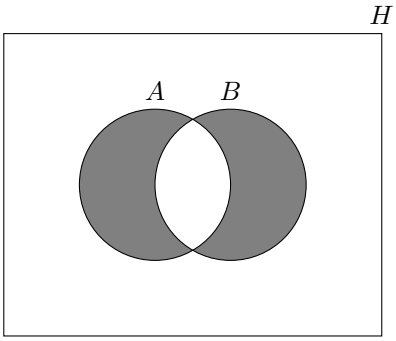
$\bigcup_{j=1}^{45} A_j =$ the set of all words in the OED.

## 3.4 More set operations

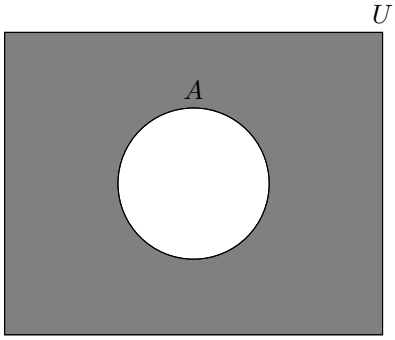
**Difference** set operation $-$. $A$ difference with $B$ is defined to be the set containing elements which are in $A$ but <u>not</u> $B$. That is, $A - B = \{x : x \in A \wedge x \notin B\}$. A set difference is <u>not</u> strictly commutative, often $A - B \neq B - A$.




**Symmetric Difference** set operation $\triangle$. $A$ symmetric difference with $B$ is defined to be the set containing elements which are in $A$ or $B$, but not $A$ and $B$. That is, $A \triangle B = \{x : x \in A \oplus x \in B\}$.




**Complement** set operation . complement $A$ is defined to be the set containing elements in $U$ which are not in $A$. That is, $\overline{A} = \{x : x \in U \wedge x \notin A\}$.




**Summary of Set Operations**

| Operation | Notation | Set Builder |
|---|---|---|
| Intersection | $A \cap B$ | $\{x : x \in A \wedge x \in B\}$ |
| Union | $A \cup B$ | $\{x : x \in A \vee x \in B\}$ |
| Difference | $A - B$ | $\{x : x \in A \wedge x \notin B\}$ |
| Symmetric Difference | $A \triangle B$ | $\{x : x \in A \oplus x \in B\}$ |
| Complement | $\overline{A}$ | $\{x : x \in U \wedge x \notin A\}$ |

## 3.5 Set identities

The laws of propositional logic can be used to derive corresponding set identities. A **set identity** is an equation involving sets that is true, regardless of the contents of the sets used in the expression.

| **Law Name** | $\cup$ Union | $\cap$ Intersection |
|---|---|---|
| Idempotent | $A \cup A = A$ | $A \cap A = A$ |
| Associative | $(A \cup B) \cup C = A \cup (B \cup C)$ | $(A \cap B) \cap C = A \cap (B \cap C)$ |
| Commutative | $A \cup B = B \cup A$ | $A \cap B = B \cap A$ |
| Distributive | $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ | $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ |
| Identity | $A \cup \emptyset = A$ | $A \cap U = A$ |
| Domination | $A \cup U = U$ | $A \cap \emptyset = \emptyset$ |
| Double Complement | $\overline{\overline{A}} = A$ | |
| Complement | $A \cup \overline{A} = \mathrm{T}$ | $A \cap \overline{A} = \mathrm{F}$ |
| DeMorgan | $\overline{A \cup B} = \overline{A} \cap \overline{B}$ | $\overline{A \cap B} = \overline{A} \cup \overline{B}$ |
| Absorption | $A \cup (A \cap B) = A$ | $A \cap (A \cup B) = A$ |

## 3.6 Cartesian products

An **ordered pair** of items is written $(x, y)$, where the first entry is $x$ and the second entry is $y$. The use of () instead of {} indicates that order matters.
**Cartesian Product** of $A$ and $B$, $A \times B = \{(a, b) : a \in A \wedge b \in B\}$

$$A = \{1, 2\} \qquad A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$$
$$B = \{a, b, c\} \qquad B \times A = \{(a, 1), (a, 2), (b, 1), (b, 2), (c, 1), (c, 2)\}$$

An ordered list of 3 items is called an **ordered triple**, denoted as $(x, y, z)$. For a size of $\geq 4$, use the term **n-tuple**. For example, $(u, w, x, y, z)$.

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) : a_i \in A \text{ for all } i \text{ such that} 1 \leq i \leq n\}$$

Another Example

$$A = \{a, b\} \qquad (a, 1, y, \beta) \in A \times B \times C \times D$$
$$B = \{1, 2\} \qquad (b, 1, x, \alpha) \in A \times B \times C \times D$$
$$C = \{x, y\} \qquad (1, b, x, \beta) \notin A \times B \times C \times D$$
$$D = \{\alpha, \beta\} \qquad \text{order matters}$$

$A \times A = A^2$, and in general,

$$A^k = \underbrace{A \times A \times \cdots \times A}_{k-times}$$

The **Cardinality of Cartesian Products**:

$$|A^n| = |A|^n$$
$$|A_1 \times A_2 \times \cdots| = |A_1| \cdot |A_2| \cdots$$

### 3.6.1 Strings

A sequence of characters is called a **string**. The set of characters used in a set of string is called the **alphabet** for the set of strings. The **length** of a string is the number of characters in the string. For example, the length of $'xxyxyx'$ is 6. The **empty string** is a string whose length is 0, and is usually denoted by $\lambda$. It is useful for $A^0$, for some alphabet $A$. $\{0,1\}^0 = \{\lambda\}$. If $s$ and $t$ are two strings, then the **concatenation** of $s$ and $t$ is the string obtained by putting $s$ and $t$ together.

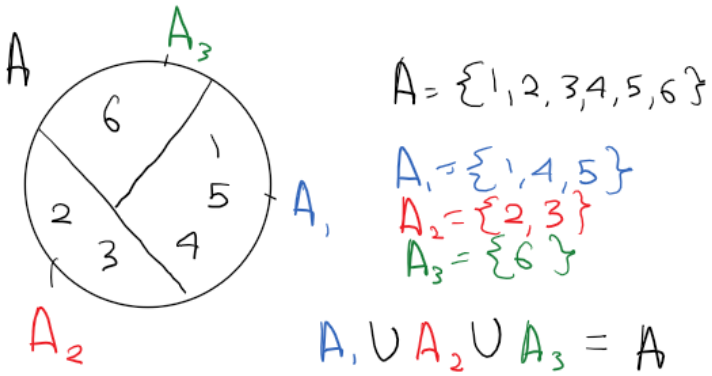
$$s = 010 \qquad st = 01011$$
$$t = 11 \qquad t0 = 110$$

Strings are used to specify passwords for computers or online accounts. Security systems vary with respect to the alphabet of characters allowed or required in a valid password. Strings also play an important rules in discrete mathematics as a mathematical tool to help count cardinality of sets.

## 3.7 Partitions

Two sets, $A$ and $B$, are said to be **disjoint** if their intersection is empty ($A \cap B = \emptyset$). A sequence of sets, $A_1, A_2, A_3, \ldots, A_n$, is **pairwise disjoint** if every pair of distinct sets in the sequence is disjoint. A **partition** of a non-empty set $A$ is a collection of non-empty subsets such that each element of $A$ is in exactly one of the subsets. $A_1, A_2, A_3, \ldots, A_n$ is a partition for a nonempty set $A$ if:

- For all $i, A_i \subseteq A$
- For all $i, A_i \neq \emptyset$
- $A_1, A_2, \ldots, A_n$ are pairwise disjoint
- $A = \bigcup_{i=1}^{n} A_i$, for some $n \in \mathbb{Z}^+$



# 4 Functions

## 4.1 Definition of functions

A **function** maps elements of one set $X$ to elements of another set $Y$. A function from $X$ to $Y$ can be viewed as a subset of $X \times Y : (x, y) \in f$ if $f$ maps $x$ to $y$. The notation for a function is:

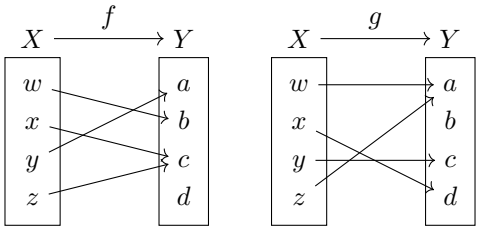
$$f : X \to Y, \text{ where } X \text{ is the } \textbf{domain} \text{ and } Y \text{ is the } \textbf{co-domain}.$$

*if $f$ maps an element of the domain to zero elements or more than one element of the target, then $f$ is not *well-defined*

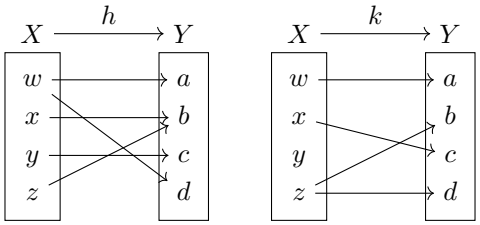
**Arrow Diagram**:

$$X = \{w, x, y, z\}$$
$$Y = \{a, b, c, d\}$$

Well-defined functions:




Not well-defined functions:




For function $f : X \to Y$, an element $y$ is in the **range** of $f$ iff there is an $x \in X$ such that $(x, y) \in f$.

$$\text{Range of } f = \{y : (x, y) \in f, \text{ for some } x \in X\}$$

Two functions, $f$ and $g$, are **equal** if $f$ and $g$ have the same domain and target and $f(x) = g(x)$ for every $x$ in the domain.

$$\forall x : f(x) = g(x) \implies f = g$$

## 4.2 Floor and Ceiling functions

The **Floor** function, $\lfloor x \rfloor$

$$\text{floor} : \mathbb{R} \to \mathbb{Z}, \text{ where floor}(x) = \text{the largest integer } y \text{ such that } y \leq x.$$

Notation: $\text{floor}(x) = \lfloor x \rfloor$

The **Ceiling** function, $\lceil x \rceil$

$$\text{ceiling} : \mathbb{R} \to \mathbb{Z}, \text{ where ceiling}(x) = \text{the smallest integer } y \text{ such that } y \geq x.$$

Notation: $\text{ceiling}(x) = \lceil x \rceil$

Examples of floor and ceiling:

$$\lceil 4.32 \rceil = 5 \qquad \lfloor 4.32 \rfloor = 4$$
$$\lceil -4.32 \rceil = -4 \qquad \lfloor -4.32 \rfloor = -5$$
$$\lceil 4 \rceil = 4 \qquad \lfloor 4 \rfloor = 4$$
$$\lceil -4 \rceil = -4 \qquad \lfloor -4 \rfloor = -4$$

## 4.3 Properties of functions

A function $f : X \to Y$ is **one-to-one** or **injective** if $x_1 \neq x_2$ implies that $f(x_1) \neq f(x_2)$. $f$ maps different elements in x to different elements in y.

A function $f : X \to Y$ is **onto** or **surjective** if the range of $f$ is equal to the target $Y$. That is, $\forall y \exists x (y \in Y \wedge x \in X \wedge f(x) = y)$

A function $f : X \to Y$ is **bijective** if it is both **injective** and **surjective**. A **bijective** function is called a **bijection**, or a **one-to-one correspondence**.

When the domain and target are finite sets, it is possible to infer information about their relative sizes based on whether a function is one-to-one or onto.

$$f : D \to T \text{ is } \textbf{one-to-one} \implies |D| \leq |T|$$
$$f : D \to T \text{ is } \textbf{onto} \implies |D| \geq |T|$$
$$f : D \to T \text{ is } \textbf{bijective} \implies |D| = |T|$$

## 4.4 The inverse of a function

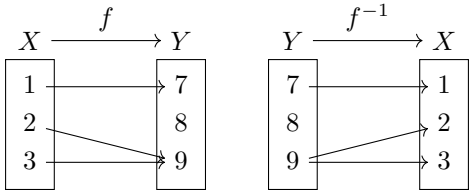
If a function $f : X \to Y$ is a *bijection*, then the **inverse** of f is obtained by exchanging the first and second entries in each pair in $f$.

$$\text{given } f : X \to Y$$
$$\text{inverse } f^{-1} : \{(y, x) : (x, y) \in f\}$$

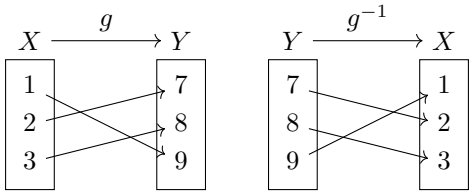
Reversing the cartesian pair does not always create a well-defined function. *Some functions do not have an inverse.*

**Examples**:

$$X = \{1, 2, 3\} \qquad f = \{(1, 7), (2, 9), (3, 9)\}$$
$$Y = \{7, 8, 9\} \qquad g = \{(1, 9), (2, 7), (3, 8)\}$$




$f^{-1}$ is not well defined, therefore $f$ does not have an inverse.




$g^{-1}$ is well defined, therefore $g$ does have an inverse.

## 4.5 Composition of functions

The process of applying a function to the result of another function is called **composition**.

$$\begin{aligned} f &: X \to Y \\ g &: Y \to Z \\ (g \circ f) &: X \to Z, \text{ such that } \forall x : x \in X, (g \circ f)(x) = g(f(x)) \end{aligned}$$

Remember that order matters, as often $(g \circ f)(x) \neq (f \circ g)(x)$. However, composition is associative:

$$(f \circ g \circ h)(x) = ((f \circ g) \circ h)(x) = (f \circ (g \circ h))(x) = f(g(h(x)))$$

### 4.5.1 Identity Function

The **Identity Function** maps a set onto itself and maps every element to itself. It is notated as $I_A : A \to A$, where $A$ is the set it maps. There are a number of identities about the Identity Function.

Let $f : A \to B$ be a bijection. Then,

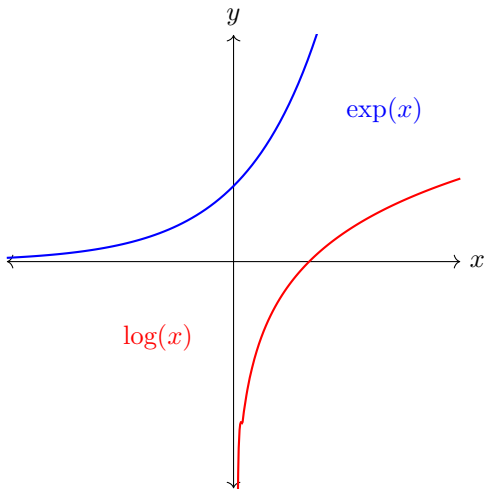
$$f \circ f^{-1} = I_B \text{ and } f^{-1} \circ f = I_A$$

## 4.6 Logarithms and exponents

The **Exponential** function, $\exp_b : \mathbb{R} \to \mathbb{R}^+, \exp_b(x) = b^x$. $b$ is the base of the exponent and $x$ is the exponent.

Properties of exponents:

$$\begin{aligned} b^x b^y &= b^{x+y} \\ (b^x)^y &= b^{xy} \\ \frac{b^x}{b^y} &= b^{x-y} \\ (bc)^x &= b^x c^x \end{aligned} \qquad \begin{aligned} &b \in \mathbb{R}^+ \\ &x \in \mathbb{R} \end{aligned} \qquad \begin{aligned} &c \in \mathbb{R}^+ \\ &y \in \mathbb{R} \end{aligned}$$




The **Logarithms** function, $\log_b : \mathbb{R} \to \mathbb{R}^+, \log_b(y) = x$. $b$ is the base of the logarithm and $x$ is the exponent.

Properties of exponents:

$$\log_b(xy) = \log_b x + \log_b y \qquad b \in \mathbb{R}^+$$
$$\log_b(\frac{x}{y}) = \log_b x - \log_b y \qquad c \in \mathbb{R}^+$$
$$\log_b(x^y) = y\log_b x \qquad x \in \mathbb{R}$$
$$\log_c x = \frac{\log_b x}{\log_b c} \qquad y \in \mathbb{R}$$

# 5 Boolean Algebra

## 5.1 An introduction to Boolean Algebra

**Boolean Algebra** is a set of rules/operations for working with variables whose values are either 0 or 1. It corresponds highly to propositional logic.

**Boolean Multiplication**, denoted by $\cdot$.

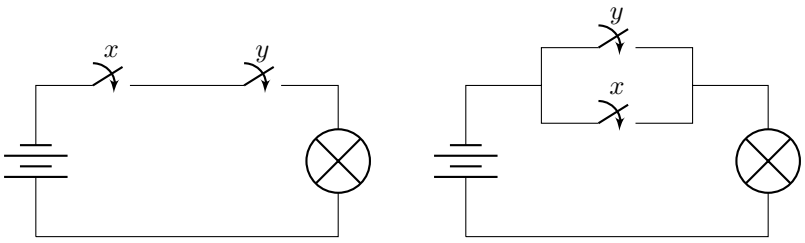
| Boolean $\cdot$ | Logic $\wedge$ |
|---|---|
| $0 \cdot 0 = 0$ | $\mathrm{F} \wedge \mathrm{F} = \mathrm{F}$ |
| $0 \cdot 1 = 0$ | $\mathrm{F} \wedge \mathrm{T} = \mathrm{F}$ |
| $1 \cdot 0 = 0$ | $\mathrm{T} \wedge \mathrm{F} = \mathrm{F}$ |
| $1 \cdot 1 = 1$ | $\mathrm{T} \wedge \mathrm{T} = \mathrm{T}$ |

**Boolean Addition**, denoted by $+$.

| Boolean $\cdot$ | Logic $\vee$ |
|---|---|
| $0 + 0 = 0$ | $\mathrm{F} \vee \mathrm{F} = \mathrm{F}$ |
| $0 + 1 = 1$ | $\mathrm{F} \vee \mathrm{T} = \mathrm{T}$ |
| $1 + 0 = 1$ | $\mathrm{T} \vee \mathrm{F} = \mathrm{T}$ |
| $1 + 1 = 1$ | $\mathrm{T} \vee \mathrm{T} = \mathrm{T}$ |

**Boolean Complement**, denoted by $\bar{}$.

| Boolean $\bar{}$ | Logic $\neg$ |
|---|---|
| $\bar{0} = 1$ | $\neg\mathrm{F} = \mathrm{T}$ |
| $\bar{1} = 0$ | $\neg\mathrm{T} = \mathrm{F}$ |




Shannon Circuit (AND $\cdot$) Switching Circuit (OR $+$)

Variables that can have a value of either 1 or 0 are called **Boolean Variables**. Boolean expressions are made of boolean variables. There are also common shorthand ways of notating operations.

$$x \cdot y + 1 \cdot \bar{z} = xy + \bar{z}$$
$$x + z + \overline{0 + y} = x + z \cdot \bar{y}$$

| **Law Name** | + OR | · AND |
|---|---|---|
| Idempotent | $x + x = x$ | $x \cdot x = x$ |
| Associative | $(x + y) + z = x + (y + z)$ | $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ |
| Commutative | $x + y = y + x$ | $x \cdot y = y \cdot x$ |
| Distributive | $x + (y \cdot z) = (x + y) \cdot (x + z)$ | $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ |
| Identity | $x + 0 = x$ | $x \cdot 1 = x$ |
| Domination | $x + 1 = 1$ | $x \cdot 0 = 0$ |
| Double Complement | $\overline{\overline{x}} = x$ | |
| Complement | $x + \overline{x} = 1$ | $x \cdot \overline{x} = 0$ |
| DeMorgan | $\overline{x + y} = \overline{x} \cdot \overline{y}$ | $\overline{x \cdot y} = \overline{x} + \overline{y}$ |
| Absorption | $x + (x \cdot y) = x$ | $x \cdot (x + y) = x$ |

## 5.2 Boolean functions

A **boolean function** is a function which maps $B^k \to B$, where $B = \{0, 1\}$. For example, consider $f : B^3 \to B$

| $x$ | $y$ | $z$ | $f(x,y,z)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

| $x$ | $y$ | $z$ | $f(x,y,z)$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | $\bar{x}yz$ |
| 1 | 0 | 0 | 1 | $x\bar{y}\bar{z}$ |
| 1 | 0 | 1 | 1 | $x\bar{y}z$ |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 1 | $xyz$ |

This function is equivalent to $f(x, y, z) = x\overline{y} + yz$. This can be determined using the boolean table and a strategy of combining the cases.

$$\begin{aligned} f(x,y,z) &= \bar{x}yz + x\bar{y}\bar{z} + x\bar{y}z + xyz \\ &= y(\bar{x}z + xz) + \bar{y}(x\bar{z} + xz) \\ &= y(z \cdot 1) + \bar{y}(x \cdot 1) \\ &= yz + \bar{y}x = \bar{y}x + yz \\ &= x\bar{y} + yz \end{aligned}$$

A **literal** is a boolean variable or the complement of a boolean variable, for example $x$ or $\bar{x}$. In a boolean function whose input variables are $v_1, v_2, \ldots, v_k$, a *mini-term* is a product of literals $u_1, u_2, \ldots, u_k$, such that $u_j$ is either $v_j$ or $\overline{v_j}$

## 5.3 Disjunctive and conjunctive normal form

A boolean expression that is the sum of literals is said to be in *disjunctive normal form*, **DNF**. It has the following form:

$$c_1 + c_2 + \cdots + c_m, \text{ where } c_j \text{ for } j \in \{1, \ldots, m\} \text{ is a product of literals.}$$

For example, $\bar{x}y\bar{z} + xy + w + y\bar{z}w$. The complement only applies to a single variable and no addition within a term.

A boolean expression that is the product of sums of literals is said to be in *conjunctive normal form*, **CNF**. It has the following form:

$$d_1 + d_2 + \cdots + d_m, \text{ where } d_j \text{ for } j \in \{1, \ldots, m\} \text{ is a sum of literals.}$$

Each $d_j$ is called a clause, and complements are only applied to a single variable. Additionally, there is no multiplication within variables. An example is $(\bar{x} + y + z)(x + \bar{y})(w)(y + \bar{z} + w)$.

## 5.4 Functional completeness

A set of operators is functionally complete if any boolean function can be expressed using only operations from the set. Two expressions can be added using only multiplication and complement.

$$x + y = \overline{\bar{x}\bar{y}} \text{ DeMorgan's Law}$$

DeMorgan's Law can be extended to more than two boolean variables:

$$x + y + w + z = \overline{\bar{x}\bar{y}\bar{w}\bar{z}}$$

The same can be said about the addition variant of the law:

$$xy = \overline{\bar{x} + \bar{y}}$$
$$xywz = \overline{\bar{x} + \bar{y} + \bar{w} + \bar{z}}$$

The **NAND** operation, $\uparrow$, and the **NOR** operation, $\downarrow$.

| $x$ | $y$ | $x \uparrow y$ | $x \downarrow y$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

{NAND} is functionally complete, it can create the complement, from which all other possible gates can be created.

| $x$ | $x \uparrow x$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

From the complement, AND can be created, and {AND, COMPLEMENT} has already been proven to be functionally complete.

$$xy = \overline{x \uparrow y} = (x \uparrow y) \uparrow (x \uparrow y)$$

## 5.5 Boolean satisfiability

The **Boolean Satisfiability problem**, called the *SAT* for short, takes the boolean expression as an input and asks whether it is possible to set the values of the variables so that the expression evaluates to 1.

If $\exists x \exists y \ldots B(x, y, \ldots)$, then the expression is **satisfiable**
If $\forall x \forall y \ldots \neg B(x, y, \ldots)$, then the expression is **unsatisfiable**

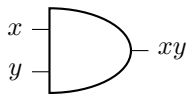
Expressions in DNF form are very easy to determine satisfiability. If there is <u>any</u> term which does *not* contain a variable and its complement, it is satisfiable. For example,

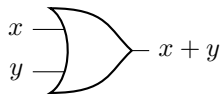
$$x\bar{y}z\bar{x} + \overbrace{\bar{w}xy\bar{z}}^{\text{no self-complements}} + \bar{w}xw\bar{x} + xy\bar{z}z$$

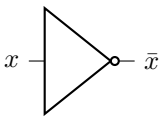
The above equation *is* satisfiable because there is a term which does not contain a self-complement.

## 5.6 Gates and circuits

Circuits are built from electrical devices called **gates**.

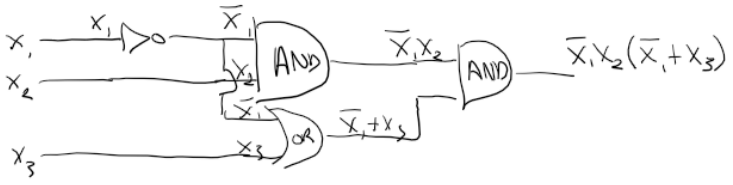
**AND**: 


**OR**: 


**NOT/INVERTER**: 

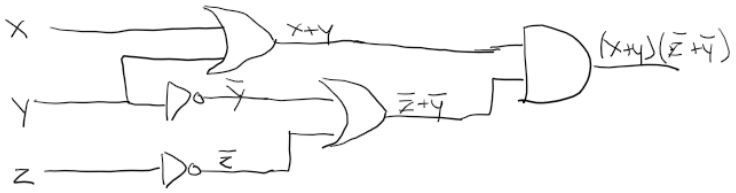
The boolean function $f(x_1, x_2) : (f(x_1, x_2) \cdot x_1) + x_2$. Yes, circuits can contain recursion.


Boolean expressions can also be constructed by following the logic or a circuit.




$$f(x_1, x_2, x_3) = \bar{x}_1 x_2 (\bar{x}_1 + x_3)$$

An example of constructing a circuit from a boolean expression, $f(x, y, z) : (x + y)(\bar{z} + \bar{y})$




### 5.6.1 Designing Circuits

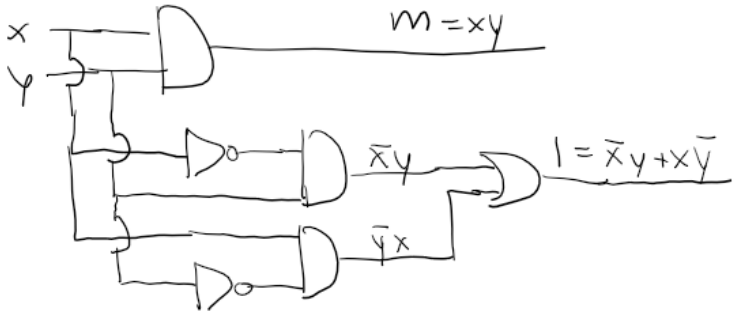
1. Build an input/output table with the desired output for every combination of input
2. Construct a boolean expression that computes the same function as the function specified in the input/output table
3. Construct a digital circuit that realizes the boolean expression

I/O for sum of two bits, $x, y$

| x | y | m | l |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

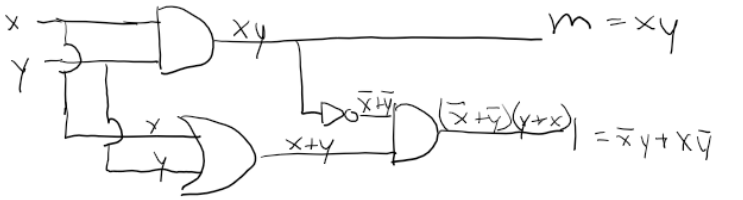
$m = xy$ most significant bit

$l = x\bar{y} + \bar{x}y$ least significant bit

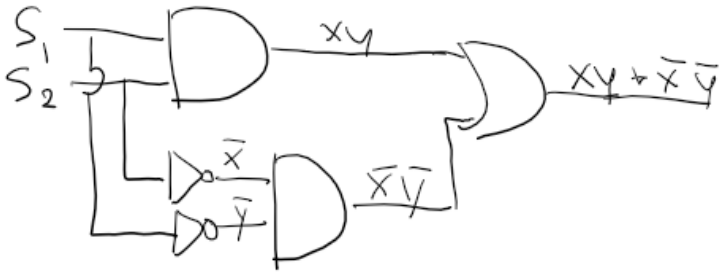



*this method does not necessarily give the most efficient circuit.

Circuits with fewer gates cost less to manufacture. Here is a simplified circuit to add two bits:




Here is another example with boolean logic for light switches:



# 6 Relation and Digraphs

## 6.1 Introduction to binary relations

A **Binary Relation** between two sets $A$ and $B$ is a subset $R$ of $A \times B$. It is binary because it is between two sets.

$$\text{for } a \in A \wedge b \in B, (a,b) \in R \text{ is denoted as } a\mathrm{R}b$$

For example, consider the relation C between $\mathbb{R}$ and $\mathbb{Z}$:

$$x\mathrm{C}y \text{ if } |x-y| \leq 1, \text{ where } x \in \mathbb{R} \text{ and } y \in \mathbb{Z}$$

If $A$ and $B$ are finite, then relation R between $A$ and $B$ can be represented by a set of ordered pairs.

### 6.1.1 Matrix Representation

$$P = \{\text{Sue}, \text{Harry}, \text{Sam}\}$$
$$\text{File} = \{\text{File A}, \text{File B}, \text{File C}, \text{File D}\}$$

$$\begin{array}{c} \\ \text{Sue} \\ \text{Harry} \\ \text{Sam} \end{array} \begin{array}{c} \begin{array}{cccc} \text{File A} & \text{File B} & \text{File C} & \text{File D} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$\text{An element is } \begin{array}{l} 1 \text{ if } p\mathrm{R}f \text{ is true} \\ 0 \text{ if } p\mathrm{R}f \text{ is false} \end{array}$$

## 6.2 Properties of binary relations

## 6.3 Directed graphs, paths, and cycles

## 6.4 Composition of relations

## 6.5 Graph powers and the transitive closure

## 6.6 Matrix multiplication and graph powers

## 6.7 Partial orders

## 6.8 Strict orders and directed acyclic graphs

## 6.9 Equivalence relations

## 6.10 N-ary relations and relational databases

# 7 Computation

# 8 Induction and Recursion

# 9 Integer Properties

# 10 Introduction to Counting

# 11 Advanced Counting

## 11.1 Generating permutations

## 11.2 Binomial coefficients and combinatorial identities

## 11.3 The pigeonhole principle

## 11.4 Generating functions

# 12 Discrete Probability

# 13 Graphs

# 14 Trees