

Computer Science 173

Course Notes

Professor L. Pitt

scribe: Bardia Sadri

Note: These notes are likely to have typos and errors. The errors will be fixed and posted online as they are found.

January 30, 2003

Contents

1	Propositional Logic	5
2	Predicate Logic	15
3	Proof Techniques	23
4	Set Theory	33
5	Boolean Functions	53
6	Familiar Functions and Growth Rate	61
7	Mathematical Induction	71
8	Running Time of Algorithms	85
9	Relations	105
10	Equivalence Relations and Partial Orders	117

Chapter 1

Propositional Logic

1.1 Propositions

Definition 1 A *proposition* is a declarative statement that is either *true* (T) or *false* (F) (but not both).

EXAMPLE ▷ Propositions:

- “*CS 173 is taught in 124 Burrill.*”
- “ $3 + 2 = 5.$ ”
- “ $3 + 2 = 32.$ ”
- “*Prof Pitt is the queen of England.*”
- “*The earth is flat.*”
- “*The earth is an oblate spheroid.*”
- “*There is no gravity.*”

The earth sucks.

EXAMPLE ▷ Non-propositions:

- “*Eat your peas.*”
- “*Will you study hard?*”
- “ $3 + 2.$ ”

- “ $x + y = z$.”
- “Your place or mine?”

Definition 2 *negation* of a proposition p is written \bar{p} (or $\neg p$ or $\sim p$) and has meaning “It is not the case that p .”

EXAMPLE \triangleright “It is not the case that CS 173 is taught in 1243 Burrill.”
(CS 173 is *not* taught in 124 Burrill).

EXAMPLE \triangleright “It is not the case that $3 + 2 = 5$.” ($3 + 2 \neq 5$).

EXAMPLE \triangleright “It is not the case that Prof Pitt is the queen of England.”
(Prof Pitt is *not* the queen of England).

Note.

- \bar{p} is true *exactly* when p is false.
- \bar{p} is false *exactly* when p is true.

Truth table:

p	\bar{p}
T	F
F	T

1.2 Compound Propositions

Let p and q be propositions. We form new propositions from p , q , and *Logical Operators*: $\neg, \wedge, \vee, \oplus, \rightarrow, \leftrightarrow$ as follows.

Conjunction (And)

English: “and”

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

EXAMPLE \triangleright “*John is tall and slim*” is false if John is short, fat, or both.

Disjunction (Or)

English: “*or*” (inclusive)

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

$p \vee q$ is true if p , or q , or both are true. $p \vee q$ is false *only if* both p and q are false.

EXAMPLE \triangleright Compare with English:

- “*You will give me \$20 or I will punch you.*” (“exclusive or” is meant).
 - “*To ride the roller-coaster, you must be 12 years old or taller than 48”.*” (“inclusive or” is meant).
-

Exclusive Or

English: “*either ... or ...*”

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

$p \oplus q$ is true if *exactly one* of p and q are true and false otherwise.

Implication

English: “if . . . then . . .”/“implies”

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

EXAMPLE ▷ “If it is raining then it is cloudy.” (Raining \rightarrow Cloudy).

$p \rightarrow q$ is true unless $p = \text{T}$ and $q = \text{F}$. $p \rightarrow q$ is true when $p = \text{F}$ regardless of q and is true when $q = \text{T}$ regardless of p .

There need not be any causal relationship between p and q for $p \rightarrow q$ to be true.

EXAMPLE ▷ “If there are 500 people in the room, then I am the Q.O.E.”

p is false, so the whole implication is true.

EXAMPLE ▷ “If p then $2 + 2 = 4$.” is always true (why?).

1.3 Logical Equivalence

Definition 3 If p and q are (possibly compound) propositions, then $p \equiv q$ (p is equivalent to q) if the truth tables for p and q are the same.

EXAMPLE ▷ $p \rightarrow q \equiv \bar{p} \vee q$

p	q	\bar{p}	$p \rightarrow q$	$\bar{p} \vee q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Note. The book uses the notation $p \Leftrightarrow q$ instead of $p \equiv q$.

Definition 4 The *contrapositive* of $p \rightarrow q$ is $\bar{q} \rightarrow \bar{p}$.

EXAMPLE \triangleright The contrapositive of “If it is raining then it is cloudy” is “If it is not cloudy then it is not raining.”

Theorem 1 $p \rightarrow q$ is logically equivalent to $\bar{q} \rightarrow \bar{p}$.

Proof.

p	q	\bar{p}	\bar{q}	$p \rightarrow q$	$\bar{q} \rightarrow \bar{p}$
T	T	F	F	T	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Hence $p \rightarrow q \equiv \bar{q} \rightarrow \bar{p}$. ■

Definition 5 The *converse* of $p \rightarrow q$ is $q \rightarrow p$.

EXAMPLE \triangleright The converse of “If it is raining, then it is cloudy” is “If it is cloudy, then it is raining.”

Is the converse of an implication logically equivalent to the original implication?

p	q	$p \rightarrow q$	$q \rightarrow p$
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

So, it can be cloudy *but not* raining and then the original statement of the previous example is true but the converse is false.

Definition 6 The *inverse* of $p \rightarrow q$ is $\bar{p} \rightarrow \bar{q}$.

EXAMPLE \triangleright The inverse of “If it is raining, then it is cloudy” is “If it is not raining, then it is not cloudy.”

Fact 1 The inverse of an implication is the contrapositive of its converse. Hence, the inverse and the converse of every implication are logically equivalent. In formula:

$$\bar{p} \rightarrow \bar{q} \equiv q \rightarrow p$$

Remark. The converse of an implication is not logically equivalent to the implication itself. Since the inverse and the converse are logically equivalent, then the inverse is not equivalent to the implication.

Biconditional

English: “if and only if”

p	q	$p \leftrightarrow q$	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$	$p \oplus q$
T	T	T	T	T	T	F
T	F	F	F	T	F	T
F	T	F	T	F	F	T
F	F	T	T	T	T	F

EXAMPLE \triangleright “I will punch you if and only if you do not give me \$20.”

$p \leftrightarrow q$ is logically equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$ and $\overline{p \oplus q}$.

Definition 7 A *tautology* is a compound proposition that is always true, regardless of truth values of component propositions.

Definition 8 A *contradiction* is a proposition that is always false.

EXAMPLE \triangleright $p \vee \overline{p}$ is a tautology and $p \wedge \overline{p}$ is a contradiction.

p	\overline{p}	$p \vee \overline{p}$	$p \wedge \overline{p}$
T	F	T	F
F	T	T	F

1.4 Famous Logical Equivalences

Identity

$$\begin{aligned} p \wedge \mathbf{T} &\equiv p \\ p \vee \mathbf{F} &\equiv p \end{aligned}$$

Domination

$$\begin{aligned} p \vee \mathbf{T} &\equiv \mathbf{T} \\ p \wedge \mathbf{F} &\equiv \mathbf{F} \end{aligned}$$

<i>Idempotent</i>	$p \vee p \equiv p$
	$p \wedge p \equiv p$

<i>Double Negation</i>	$\neg(\neg p) \equiv p$
------------------------	-------------------------

<i>Commutative</i>	$p \vee q \equiv q \vee p$
	$p \wedge q \equiv q \wedge p$

<i>Associative</i>	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

<i>Excluded Middle Uniqueness</i>	$p \vee \bar{p} \equiv \text{T}$
	$p \wedge \bar{p} \equiv \text{F}$

<i>Distributive</i>	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

<i>De Morgan's</i>	$\overline{(p \vee q)} \equiv \bar{p} \wedge \bar{q}$
<i>De Morgan's</i>	$\overline{(p \wedge q)} \equiv \bar{p} \vee \bar{q}$

Note.

1. *Do not memorize these, understand them!*
2. *Similar rules for other systems — There is a general theory encompassing all of this (boolean algebras).*

Proof of the first distributive law:

To show:

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

Proof. Make the truth table for every possible combination of values of p , q , and r and show that columns for the above two expressions are identical.

p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$p \vee q$	$p \vee r$	$(p \vee q) \wedge (p \vee r)$
T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	F
F	F	T	F	F	F	T	F
F	F	F	F	F	F	F	F

The fifth and the last columns are identical. ■

Proof of the second distributive law is similar.

Proof of the first De Morgan's law:

To show:

$$\overline{(p \vee q)} \equiv \bar{p} \wedge \bar{q}$$

Proof. Truth table:

p	q	\bar{p}	\bar{q}	$p \vee q$	$\overline{(p \vee q)}$	$\bar{p} \wedge \bar{q}$
T	T	F	F	T	F	F
T	F	F	T	T	F	F
F	T	T	F	T	F	F
F	F	T	T	F	T	T

Proof of the second De Morgan's law using the first De Morgan's law:

To show:

$$\overline{(p \wedge q)} \equiv \bar{p} \vee \bar{q}$$

Proof.

$$\begin{aligned} \overline{(p \wedge q)} &\equiv \overline{(\bar{\bar{p}}) \wedge (\bar{\bar{q}})} && \text{by double negation} \\ &\equiv \overline{(\bar{p}) \wedge (\bar{q})} && \text{by De Morgan's law 1} \\ &\equiv (\bar{\bar{p}}) \vee (\bar{\bar{q}}) && \text{by double negation} \\ &\equiv \bar{p} \vee \bar{q} \end{aligned}$$



EXAMPLE ▷ We want to show that $[p \wedge (p \rightarrow q)] \rightarrow q$ is a tautology.

Foreshadow: Modus Ponens; A proof method

Solution 1. Build the truth tables and show that the above compound proposition is always true.

Solution 2. Use equations to show that $[p \wedge (p \rightarrow q)] \rightarrow q \equiv \text{T}$

$$\begin{array}{ll}
 [p \wedge (p \rightarrow q)] \rightarrow q \equiv [p \wedge (\bar{p} \vee q)] \rightarrow q & \text{because } p \rightarrow q \equiv \bar{p} \vee q \\
 \equiv [(p \wedge \bar{p}) \vee (p \wedge q)] \rightarrow q & \text{distributive} \\
 \equiv [\text{F} \vee (p \wedge q)] \rightarrow q & \text{uniqueness} \\
 \equiv (p \wedge q) \rightarrow q & \text{identity} \\
 \equiv \overline{(p \wedge q)} \vee q & \text{because } p \rightarrow q \equiv \bar{p} \vee q \\
 \equiv (\bar{p} \vee \bar{q}) \vee q & \text{De Morgan} \\
 \equiv \bar{p} \vee (\bar{q} \vee q) & \text{associative} \\
 \equiv \bar{p} \vee \text{T} & \text{exclude middle} \\
 \equiv \text{T} & \text{domination}
 \end{array}$$

Chapter 2

Predicate Logic

Remember that “ $x > 3$ ” or “ $x - y = y - x$ ” are *non-propositions* because their truth value depends on the the value of x and y .

2.1 Predicates

Definition 9 A *propositional function* or *predicate* is a function that takes some variables (such as x or (x, y) in the above examples) as arguments and returns a truth value (true or false).

EXAMPLE $\triangleright P(x) = “x > 3”$ is a predicate (a true/false valued function of x):

$$P(1) = \text{F} \quad P(3) = \text{F} \quad P(3.001) = \text{T}$$

EXAMPLE $\triangleright Q(x, y) = “x - y = y - x”$.

$$Q(x, y) = \begin{cases} \text{F} & \text{if } x \neq y \\ \text{T} & \text{if } x = y \end{cases}$$

Fact 2 A predicate becomes a proposition when all of its variables are assigned values.

EXAMPLE \triangleright Let $Q(x, y) = “x > y”$.

- $Q(x, y)$ is *not* a proposition.

- $Q(3, 4)$ is a proposition.
- $Q(x, 5)$ is *not* a proposition (but it is a predicate of one variable x).

2.2 Quantifiers

When interpreting a predicate we assume a “*universe of discourse*”. In the above example x and y were understood to be real numbers. In $Q(x, y) =$ “ x is the mother of y ”, x and y are assumed to be people.

Note. Usually the universe of discourse is understood from context. Sometimes we state it explicitly though.

We now study quantifiers that can turn a predicate into a proposition.

Universal Quantifier

Definition 10 If $P(x)$ is a predicate, the *universal quantification* of $P(x)$ is the *proposition*:

“ $P(x)$ is true for all x in the universe of discourse”

and is written $\forall x P(x)$ or sometimes $(\forall x \in U)P(x)$ when U is the universe of discourse.

$\forall x P(x)$ is a *true* proposition *if and only if* $P(x)$ is true for every single x in the universe of discourse.

$\forall x P(x)$ is *false* if there is at least one x such that $P(x)$ is false.

False!

EXAMPLE $\triangleright \forall x$ x will receive an A.

True.

EXAMPLE $\triangleright \forall x$ if x does well, x will receive an A.

EXAMPLE $\triangleright \forall x$ x is awake now.

Fact 3 If the universe of discourse U is finite ($U = \{x_1, x_2, \dots, x_n\}$) then $\forall x P(x)$ is equivalent to the proposition $P(x_1) \wedge P(x_2) \wedge \dots \wedge P(x_n)$.

To determine if $P(x)$ is true when the universe of discourse is finite we can use the following algorithm.

```

1  for  $i \leftarrow 1$  to  $n$  do
2      if  $P(x_i)$  is false then
3          Halt and output " $\forall x P(x)$  is false."
4  endfor
5  output " $\forall x P(x)$  is true."

```

Existential Quantifier

Definition 11 If $P(x)$ is a predicate, the *existential quantification* of $P(x)$ is the *proposition*:

"There exists (at least one) element x of the universe of discourse such that $P(x)$ is true "

and is written $\exists x P(x)$ or sometimes $(\exists x \in U)P(x)$ when U is the universe of discourse.

$\exists x P(x)$ is a *true* proposition *if and only if* $P(x)$ is true for at least one value of x in the universe of discourse.

$\exists x P(x)$ is *false* if and only if $P(x)$ is false for *every* value of x .

EXAMPLE $\triangleright \exists x x$ will receive an A.

True.

EXAMPLE $\triangleright \exists x x$ will receive an E.

EXAMPLE $\triangleright \exists x x$ will receive an \exists .

EXAMPLE $\triangleright \exists x x$ awake now!

Notation. We use the notation $\exists! x$ to denote *unique existence*. It is read "*There is one and only one x such that ...*".

Fact 4 If universe is finite ($U = \{x_1, x_2, \dots, x_n\}$), then

$$\exists x P(x) \equiv P(x_1) \vee P(x_2) \vee \dots \vee P(x_n).$$

In such cases as above, we can use the following algorithm to check if $\exists x P(x)$.

```

1  for  $i \leftarrow 1$  to  $n$  do
2      if  $P(x_i)$  is true then
3          halt and output " $\exists xP(x)$  is true."
4  endfor
5  output " $\exists xP(x)$  is false."

```

EXAMPLE \triangleright Let the universe U be the set of all creatures and denote by $L(x)$, $F(x)$, and $C(x)$ the followings:

- $L(x)$: " x is a lion."
- $F(x)$: " x is fierce."
- $C(x)$: " x drinks coffee."

English	Math
All lions are fierce	$\forall x(L(x) \rightarrow F(x))$
Some lions don't drink coffee	$\exists x(L(x) \wedge \overline{C(x)})$
Some fierce creatures don't drink coffee	$\exists x(F(x) \wedge \overline{C(x)})$

Remark. The proposition $\exists x(L(x) \rightarrow \overline{C(x)})$ is not a correct equivalent of the second example. In fact, this proposition is true for any x that is an elephant.

EXAMPLE \triangleright Let U again be the set of all creatures and let $B(x)$, $L(x)$, $H(x)$, and $R(x)$ denote the followings:

- $B(x)$: x is a hummingbird.
- $L(x)$: x is a large bird.
- $H(x)$: x lives on honey.
- $R(x)$: x is richly colored.

English	Math
All hummingbirds are richly colored	$\forall x(B(x) \rightarrow R(x))$
No large birds live on honey	$\neg \exists x(L(x) \wedge H(x))$
Birds that do not live on honey are dull in color	$\forall x(\neg H(x) \rightarrow \neg R(x))$

For the second example above $\forall x(L(x) \rightarrow \neg H(x))$ is equally a correct translation.

Moral: Be careful; English is imprecise!

2.3 Negation of a quantified predicate

$\forall xP(x)$ means:

$P(x)$ is true for every x .

So, $\neg(\forall xP(x))$ is

not [$P(x)$ is true for every x].

This is not the same as “ $P(x)$ is false for all x .”

This means:

For *some* x , $P(x)$ is false.

or in formula: $\exists x\neg P(x)$.

So, in general to negate universal quantifiers: *Move the negation to the right, flipping the quantifiers from \forall to \exists .*

$\exists xP(x)$ means:

$P(x)$ is true for some x .

So, $\neg(\exists xP(x))$ is

not [$P(x)$ is true for some x].

This is not the same as “ $P(x)$ is false for some x .”

This means:

$P(x)$ is false for all x .

or in formula: $\forall x\neg P(x)$.

So, in general to negate existential quantifiers: *Move the negation to the right, flipping the quantifiers from \exists to \forall .*

EXAMPLE \triangleright “No large birds live on honey.”

$$\begin{aligned} \neg\exists x(L(x) \wedge H(x)) &\equiv \forall x\neg(L(x) \wedge H(x)) && \text{negation of quantifier rule} \\ &\equiv \forall x(\neg L(x) \vee \neg H(x)) && \text{De Morgan} \\ &\equiv \forall x(L(x) \rightarrow \neg H(x)) && p \rightarrow q \equiv \bar{p} \vee q \end{aligned}$$

2.4 Free and bound variables

Definition 12 A variable is *bound* if its value is known or is in the scope of some quantifier. A variable is *free* if it is not bound.

EXAMPLE \triangleright In “ $P(x)$ ”, x is free. However in “ $P(5)$ ”, x is bound to 5.

EXAMPLE ▷ In “ $\forall xP(x)$ ”, x is bound (has been quantified).

Remark. In propositions all variables must be bound (assigned or quantified).

Some predicates involve many variables and to bind them we use many quantifiers.

EXAMPLE ▷ Let $P(x, y) = “x > y”$. Then:

- “ $\forall xP(x, y)$ ” is not a proposition (y is free).
- “ $\forall x\forall yP(x, y)$ ” is a false proposition.
- “ $\forall x\exists yP(x, y)$ ” is a true proposition.
- “ $\forall xP(x, 3)$ ” is a false proposition (both x and y are bound).

2.5 Meaning of multiply quantified predicates

1. “ $\forall x\forall yP(x, y)$ ”: $P(x, y)$ is true for every combination of x and y values.
2. “ $\exists x\exists yP(x, y)$ ”: There is at least one choice of x and y , such that $P(x, y)$ is true.
3. “ $\forall x\exists yP(x, y)$ ”: For each x we can find at least one y (possibly a different y for each x) such that $P(x, y)$ is true.
4. “ $\exists x\forall yP(x, y)$ ”: There is one x in particular (there may be more) such that for *every* y , $P(x, y)$ is true.

Note that there is a big difference between 3, and 4.

EXAMPLE ▷ Let the universe U be people in this class and $N(x, y)$ denote “ x is sitting next to y .”

- | | |
|---------------|---|
| <i>True.</i> | • $\exists x\exists yN(x, y)$: There are two people next to each other. |
| <i>False.</i> | • $\forall x\forall yN(x, y)$: Everybody is sitting next to everybody. |
| <i>False.</i> | • $\exists x\forall yN(x, y)$: There is a particular person whom everybody is sitting next to. |
| <i>True?</i> | • $\forall x\exists yN(x, y)$: Every person is sitting next to somebody. |

EXAMPLE ▷ Let the universe U be the set of real numbers \mathbb{R} .

- $\forall m \exists n (n > 2^m)$ (True). To show this, we need to show that for every number m there is a number (that can depend on m) that is greater than 2^m . Well, $n = 2^m + 1$ is greater than 2^m . So $\forall m (n = 2^m + 1 > 2^m)$. Hence $\forall m \exists n (n > 2^m)$.
- $\exists n \forall m (n > 2^m)$. Is there a number n such that $n > 2^m$ regardless of m ? ∞ is not a number. We show the answer to this question is “No” by showing its negation is true.

$$\neg(\exists n \forall m (n > 2^m)) \equiv \forall n \neg(\forall m (n > 2^m)) \equiv \forall n \exists m \neg(n > 2^m) \equiv \forall n \exists m (n \leq 2^m)$$

This is true by letting $m = n$. Clearly, $n \leq 2^n$ for all n .

Check table 2 at page 31 of the text.

EXAMPLE \triangleright Let $U = \mathbb{R}$ and $Q(x, y, z) = “x + y = z”$.

- $\forall x \forall y \exists z Q(x, y, z)$: True.
- $\exists z \forall x \forall y Q(x, y, z)$: False.
- $\forall x \exists y \exists z Q(x, y, z)$: True.

EXAMPLE \triangleright Translating from English. Let $A(x, y)$ denote “Person x receives an A on assignment y .”

- There is a person in this class who will receive an A on every assignment: $\exists x \forall y A(x, y)$.
- Everyone will receive an A on some assignment: $\forall x \exists y A(x, y)$.

If the universe is finite ($U = \{x_1, x_2, \dots, x_n\}$), to determine if $\forall x \exists y P(x, y)$ is true we use the algorithm below:

```

1  for  $i \leftarrow 1$  to  $n$  [choose  $x$ ]
2      for  $j \leftarrow 1$  to  $n$  [choose  $y$ ]
3          if  $P(x_i, y_j)$  then go to 6
4      endfor
5      Halt and output “ $\forall x \exists y P(x, y)$  is not true.”
6      [Found: A “ $y$ ” was found for the current “ $x$ ”; so check the next  $x$ ]
7  endfor
8  Halt and output “ $\forall x \exists y P(x, y)$  is true.”

```

Similar nested loop will do the job for $\forall x \forall y$, $\exists x \forall y$, and $\exists x \exists y$.

How about $\forall x \exists y \forall z \forall w \exists v$?

Chapter 3

Proof Techniques

Definition 13 A *theorem* is a statement that can be shown to be true.

A *proof* is a sequence of statements, forming an argument to show that a theorem is true.

The statements used in a proof can include *axioms* or *postulates*, which are the underlying assumptions about the mathematical structure, the hypotheses of the theorem to be proved, and previously proved theorems. The *rules of inference*, which are the means used to draw conclusions from other assertions, tie together the steps of a proof.

EXAMPLE ▷ An ideal proof should look like the following:

1. ⟨ expression 1 ⟩ Axiom 14
2. ⟨ expression 2 ⟩ Axiom 23
3. ⟨ expression 3 ⟩ Lines 1 and 2, and rule 28
4. ⟨ expression 4 ⟩ Lines 2 and 3, and rule 4
5. ⟨ expression 5 ⟩ Axiom 6
6. ⟨ expression 6 ⟩ Lines 3, 5, and 6, and rule 2
- ⋮
73. ⟨ expression 73 ⟩ *What we wanted to prove.*

3.1 Rules of Inference

We use the following notation for a rule of inference:

$$\begin{array}{c}
 \textit{Hypothesis 1} \\
 \textit{Hypothesis 2} \\
 \vdots \\
 \textit{Hypothesis n} \\
 \hline
 \therefore \textit{Conclusion}
 \end{array}$$

The sign “ \therefore ” is read “therefore”.

The most famous inference rule is *Modus Ponens*:

$$\begin{array}{c}
 p \\
 p \rightarrow q \\
 \hline
 \therefore q
 \end{array}$$

Modus Ponens corresponds to $[p \wedge (p \rightarrow q)] \rightarrow q$ which is a tautology.

Fact 5 All our rules of inference should be tautologies.

EXAMPLE \triangleright (Modus Ponens)

$$\begin{array}{c}
 \text{If I am Bonnie Blair then I win a gold medal.} \\
 \text{I am Bonnie Blair} \\
 \hline
 \therefore \text{I win a gold medal.}
 \end{array}$$

3.2 Important Rules of Inference

<i>Rule</i>	<i>Corresponding Tautology</i>	<i>Name</i>
$ \begin{array}{c} p \\ \hline \therefore p \vee q \end{array} $	$p \rightarrow (p \vee q)$	addition
$ \begin{array}{c} p \wedge q \\ \hline \therefore p \end{array} $	$(p \wedge q) \rightarrow p$	simplification
$ \begin{array}{c} p \\ p \rightarrow q \\ \hline \therefore q \end{array} $	$[p \wedge (p \rightarrow q)] \rightarrow q$	modus ponens
$ \begin{array}{c} \neg q \\ p \rightarrow q \\ \hline \therefore \neg p \end{array} $	$[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$	modus tollens
$ \begin{array}{c} p \rightarrow q \\ q \rightarrow r \\ \hline \therefore p \rightarrow r \end{array} $	$[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$	hypothetical syllogism

$p \vee q$		
$\frac{\neg p}{\therefore q}$	$[(p \vee q) \wedge \neg p] \rightarrow q$	disjunctive syllogism

EXAMPLE ▷ (Modus Tollens)

If I am Bonnie Blair then I win a gold medal.
 I do not win a gold medal.

 ∴ I am not Bonnie Blair.

EXAMPLE ▷ (Hypothetical Syllogism)

If I am Bonnie Blair then I skate quickly.
 If I skate quickly then I win a gold medal.

 ∴ If I am Bonnie Blair then I win a gold medal.

EXAMPLE ▷ (Disjunctive Syllogism)

I am tired of talking or you are tired of listening.
 I am *not* tired of talking

 ∴ You are tired of listening.

EXAMPLE ▷ Consider the following propositions:

- M : She is a math major.
- C : She is a CS major.
- D : She know discrete math.
- S : She is smart.

Now consider the following statements:

1. She is a math major or a CS major.
2. If she does not know discrete math, she is not a math major.
3. If she knows discrete math, she is smart.

4. She is not a CS major.

Now we can conclude that “*She is smart*” by first writing the above sentence as logical propositions (lines 1–4) and then applying the inference rules (lines 5–8).

1. $M \vee C$
2. $\neg D \rightarrow \neg M$.
3. $D \rightarrow S$
4. $\neg C$
5. $\therefore M$ (By 1 and 4 — Disjunctive Syllogism).
6. $\therefore M \rightarrow D$ (Contrapositive of 2).
7. $\therefore D$ (By 5 and 6 — Modus ponens).
8. $\therefore S$ (By 7 and 3 — Modus ponens).

EXAMPLE \triangleright Consider the following set of propositions.

1. Rainy days make gardens grow.
2. Gardens don’t gro if it is not hot.
3. Whenever it is cold outside, it rains.

We rewrite the above setence using the following variables:

- R : It rains.
- G : The gardens grow.
- H : It is hot (we use \overline{H} for it is cold).

Thus we get the following translations:

1. $R \rightarrow G$
2. $\overline{H} \rightarrow \overline{G}$
3. $\overline{H} \rightarrow R$

Suppose we want to show that it is hot. We may try a proof by contradiction.

4. \overline{H} (We assume).
5. $\therefore R$ (By 3 and 4 — Modus ponens).
6. $\therefore \overline{G}$ (By 2 and 4 — Modus ponens).
7. $\therefore G$ (By 1 and 5 — Modus ponens).

But 6 and 7 together say that “*Gardens are growing and they are not growing,*” a contradiction showing that our assumption was wrong and therefore “*it is hot.*”

Since we derive the conclusion in our proof methods from assumptions and inference rules, the conclusions are true only when both

1. the initial propositions (assumptions) are true, and
2. the rules of inference are tautologies.

Our conclusions however could be true or false if propositions used are false, or if inference rules used are fallacies.

3.3 Fallacies

Fallacies are *non-rules* of inference often used. Some famous fallacies are *affirming the conclusion*, *denying the hypothesis*, *begging the question*, and *circular reasoning*.

3.3.1 Fallacy of affirming the conclusion

EXAMPLE ▷ Here is an example of affirming the conclusion:

If I am Bonnie Blair then I skate fast.	
I skate fast.	
\therefore I am Bonnie Blair.	

I can be Erk Heiden.

EXAMPLE ▷ Another one:

If you don't give me \$10, I will punch you in the nose.
 I punch you in the nose.

 \therefore You didn't give me \$10.

I can be sadist!

Affirming the conclusion is a fallacy because

$$[(p \rightarrow q) \wedge q] \not\rightarrow p.$$

This is because p might be false while q be true and this makes the whole implication false.

3.3.2 Fallacy of denying the hypothesis

EXAMPLE \triangleright Here is an example of denying the hypothesis:

What if it is cloudy but it doesn't rain?

If it rains then it is cloudy.
 It does not rain.

 \therefore It is not cloudy.

EXAMPLE \triangleright Another one:

What if it is a 4-wheeled truck?

If it is a car, then it has 4 wheels.
 It is not a car.

 \therefore It does not have 4 wheels.

Denying the hypothesis is a fallacy because

$$[(p \rightarrow q) \wedge \neg p] \not\rightarrow \neg q.$$

This is because p might be false while q be true and this makes the whole implication false.

3.3.3 Other mistakes

To prove $\forall xP(x)$: We must not rely on any special properties of x . We must choose x *arbitrarily* and show $P(x)$ holds.

To prove $\exists xP(x)$: We only need to find some value of x and show that $P(x)$ holds for that value.

To prove $\exists y\forall xP(x, y)$: We need to show that there is *one single* y such that for *any* x , $P(x, y)$ is true. It is not sufficient to show that $\forall x\exists yP(x, y)$.

3.4 Proof methods

3.4.1 Vacuous proof

To show that $p \rightarrow q$ is true, it is sufficient to show that p is false.

EXAMPLE $\triangleright p$: Elephants are small.

q : Bush likes to Quayle-hunt.

Since p is false, $p \rightarrow q$ is true.

3.4.2 Trivial Proof

To show that $p \rightarrow q$ is true, it is sufficient to show that q is true.

EXAMPLE $\triangleright p$: Elephants are small.

q : Bush likes to quail-hunt.

3.4.3 Direct Proof

To show that $p \rightarrow q$ is true, we assume that p is true, and use the inference rules to conclude that q is true.

EXAMPLE \triangleright (Direct Proof)

To be proved: If $n \equiv 3 \pmod{4}$ then $n^2 \equiv 1 \pmod{4}$.

Proof. We assume that $n \equiv 3 \pmod{4}$. This means that $n = 4k + 3$ for some $k \in \mathbb{Z}$. But then,

$$\begin{aligned}n^2 &= (4k + 3)^2 \\&= 16k^2 + 24k + 9 \\&= 16k^2 + 24k + 8 + 1 \\&= 4(4k^2 + 6k + 2) + 1 \\&= 4k' + 1 \quad \text{for some } k' \in \mathbb{Z}\end{aligned}$$

Hence $n^2 \equiv 1 \pmod{4}$.

3.4.4 Indirect proof

Instead of showing $p \rightarrow q$ is true, we can show that $\neg q \rightarrow \neg p$ (the contrapositive) is true. This establishes $p \rightarrow q$ as true since

$$p \rightarrow q \equiv \neg q \rightarrow \neg p.$$

3.4.5 Proof by contradiction

To prove that p is true, it is sufficient to prove that $\neg p \rightarrow q$ is true where q is clearly false. This is like saying “If $\neg p \rightarrow q$ is true, and q is false, then p must be true.” This is in fact that same as the the disjunctive syllogism

$$\frac{p \vee q \quad \neg q}{\therefore p}$$

Similar to indirect proof, to show that $p \rightarrow q$ is true, we show that $\neg(p \rightarrow q)$ results in a contradiction. That the same as showing that $\neg(\neg p \vee q) \equiv p \wedge \neg q$ results in a contradiction. In fact, the latter is the most common form of argument.

EXAMPLE \triangleright Classic proof that $\sqrt{2}$ is irrational by contradiction.

p : $\sqrt{2}$ is irrational.

$\neg p$: $\sqrt{2}$ is rational.

We show that $\neg p$ makes a contradiction, i.e. we assume $\neg p$ and derive something false.

This is the definition of a rational numbers. By requiring a and b to be relatively prime we look for the fraction $\frac{a}{b}$ is in “lowest terms”.

Lemma. *If a^2 is even then a is also even. [Prove this as an exercise.]*

If $\sqrt{2}$ is rational, then $\exists a, b : \gcd(a, b) = 1$, and $\sqrt{2} = a/b$. Then it follows that $2 = a^2/b^2$ or equivalently $2b^2 = a^2$.

Since a^2 is even (definition of even) we conclude that a is also even, i.e. $\exists k \in \mathbb{Z} : a = 2k$.

Plugging this $2k$ instead of a in $2b^2 = a^2$ we get $2b^2 = (2k)^2 = 4k^2$ or $b^2 = 2k^2$. Thus b^2 is even and so b is even.

But now we have shown that a and b are both even, i.e. $2|a$ and $2|b$. This contradicts the assumption that $\gcd(a, b) = 1$.

3.4.6 Proof by cases

Suppose $p \equiv p_1 \vee p_2 \vee \dots \vee p_n$. Then proving $p \rightarrow q$ is *equivalent* to proving

$$[p_1 \vee p_2 \vee \dots \vee p_n] \rightarrow q,$$

which is *equivalent* to

$$(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \wedge \dots \wedge (p_n \rightarrow q).$$

Proof. We show the proof for $n = 2$. For larger n the proof is similar.

$$\begin{aligned} (p_1 \vee p_2) \rightarrow q &\equiv \neg(p_1 \vee p_2) \vee q \\ &\equiv (\neg p_1 \wedge \neg p_2) \vee q \\ &\equiv (\neg p_1 \vee q) \wedge (\neg p_2 \vee q) \\ &\equiv (p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \end{aligned}$$

Note. $[p_1 \vee p_2 \vee \dots \vee p_n] \rightarrow q \not\equiv (p_1 \rightarrow q) \vee (p_2 \rightarrow q) \vee \dots \vee (p_n \rightarrow q)$.

3.4.7 Constructive vs. non-constructive existence proofs

To show $\exists x P(x)$ we can find a value of x for which $P(x)$ is true. This is called a *constructive* proof because x is *given* or *constructed* in the proof.

But we may also be able to show that $\exists x P(x)$ without actually finding an x such that $P(x)$ holds.

EXAMPLE \triangleright (Constructive proof) Show that $\forall n \in \mathbb{N} \exists n$ consecutive composite integers.

A composite is an integer which is not a prime.

Proof. Let n be an arbitrary positive integer. The following n consecutive integers are all composite:

$$\begin{array}{ll} (n+1)! + 2 & \text{divisible by 2} \\ (n+1)! + 3 & \text{divisible by 3} \\ \dots & \dots \\ (n+1)! + n + 1 & \text{divisible by } n + 1 \end{array}$$

■

EXAMPLE ▷ (Non-constructive proof) Show that $\forall n \in \mathbb{N} \exists$ a prime p such that $p > n$.

Proof. Let n be arbitrary. Consider $n! + 1$. If $n! + 1$ is a prime then we are done. Otherwise, $n! + 1$ is composite and therefore it has a prime factor. Can any of $2, 3, \dots, n$ be a factor for $n! + 1$?

$$i > 1 \rightarrow \gcd(i, i + 1) = 1$$

Thus there exists a prime factor of $n! + 1$ that is greater than n (although we do not know what it is).

Note. $n! + 1$ is not necessarily prime. For example $5! + 1 = 121 = 11 \times 11$. ■

Chapter 4

Set Theory

Definition 14 A *set* is an *unordered* collection of *elements* or objects.

Primitive Notation

EXAMPLE $\triangleright \{1, 2, 3\}$ is a set containing 3 elements: “1”, “2”, and “3”.

EXAMPLE $\triangleright \{1, 2, 3\} = \{3, 2, 1\}$ (order irrelevant).

EXAMPLE $\triangleright \{1, 1, 3, 2, 2\} = \{1, 2, 3\}$ (number of occurrences irrelevant).

EXAMPLE $\triangleright \{0, 1, 2, \dots\}$ is the set of natural numbers \mathbb{N} (infinite set).

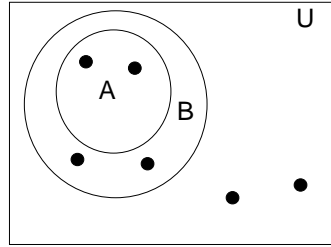
EXAMPLE $\triangleright \emptyset = \{\}$ (empty set — contains no elements).

EXAMPLE $\triangleright \emptyset \neq \{\emptyset\} = \{\{\}\}$.

Definition 15 We denote

- “ x is an element of the set S ” by $x \in S$.
- “ x is not an element of the set S ” by $x \notin S$.
- “every element of the set A is also an element of the set B ” by $A \subseteq B$,
i.e. $(\forall x)x \in A \rightarrow x \in B$.

The Venn diagram for $A \subseteq B$



Definition 16 A is a *superset* of B , denoted $A \supseteq B$ if and only if $B \subseteq A$.

Definition 17 $A = B$ if and only if A and B have exactly the same elements.

By the definition of equality of the sets A and B :

$$\begin{aligned} A = B & \text{ iff } \forall x(x \in A \leftrightarrow x \in B) \\ & \text{ iff } A \subseteq B \text{ and } A \supseteq B \\ & \text{ iff } A \subseteq B \text{ and } B \subseteq A \end{aligned}$$

If you show only one of these, **Note.** Showing two sets begin equal involves showing the following 2 things: you're half done!

1. $\forall x(x \in A \rightarrow x \in B)$, which gives us $A \subseteq B$, and
2. $\forall x(x \in B \rightarrow x \in A)$, which gives us $B \subseteq A$.

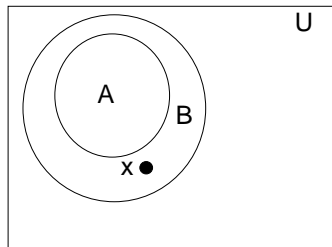
4.1 Proper Subset

Also written $A \subset B$ by some authors, although this is misguided because other (really misguided) authors use $A \subset B$ to mean $A \subseteq B$.

Definition 18 A is a *proper subset* of B , denoted $A \subsetneq B$, if $A \subseteq B$ but not $B \subseteq A$, i.e. $A \subseteq B$ but $A \neq B$.

A is a proper subset of B iff:

$$\begin{aligned} \forall x(x \in A \rightarrow x \in B) \wedge \neg \forall x(x \in B \rightarrow x \in A) & \equiv \\ \forall x(x \in A \rightarrow x \in B) \wedge \exists x \neg (\neg(x \in B) \vee x \in A) & \equiv \\ \forall x(x \in A \rightarrow x \in B) \wedge \exists x(x \in B \wedge \neg(x \in A)) & \equiv \\ \forall x(x \in A \rightarrow x \in B) \wedge \exists x(x \notin A \wedge x \in B). & \end{aligned}$$



EXAMPLE $\triangleright \{1, 2, 3\} \subseteq \{1, 2, 3, 5\}$
 $\{1, 2, 3\} \subsetneq \{1, 2, 3, 5\}$

Question: Is $\emptyset \subseteq \{1, 2, 3\}$?

Answer: Yes:

$$\forall x[(x \in \emptyset) \rightarrow (x \in \{1, 2, 3\})]$$

Since $x \in \emptyset$ is false, the implication is true.

Fact 6 \emptyset is a subset of *every* set. \emptyset is a *proper subset* of every set except \emptyset .

EXAMPLE \triangleright Is $\emptyset \in \{1, 2, 3\}$?

No.

EXAMPLE \triangleright Is $\emptyset \subseteq \{\emptyset, 1, 2, 3\}$?

Yes.

EXAMPLE \triangleright Is $\emptyset \in \{\emptyset, 1, 2, 3\}$?

Yes.

4.2 Ways to Define Sets

1. List the elements explicitly (works only for finite sets).

EXAMPLE $\triangleright \{John, Paul, George, Ringo\}$

2. List them “implicitly.”

EXAMPLE $\triangleright \{1, 2, 3, \dots\}$

EXAMPLE $\triangleright \{2, 3, 5, 7, 11, 13, 17, 19, 23, \dots\}$

3. Describe a property that defines the set (“set builder” notation).

EXAMPLE $\triangleright \{x : x \text{ is prime}\}$

EXAMPLE $\triangleright \{x \mid x \text{ is prime}\}$

EXAMPLE $\triangleright \{x : P(x) \text{ is true.}\}$ (where P is a predicate). Let $D(x, y)$ be the predicate “ x is divisible by y .” Then this set becomes

$$D(x, y) \equiv \exists k(k \cdot y = x).$$

$$\{x : \forall y[(y > 1) \wedge (y < x)] \rightarrow \neg D(x, y)\}$$

Question: Can we use any predicate P to define a set $S = \{x : P(x)\}$.

4.2.1 Russell's Paradox

Let

$$S = \{x : x \text{ is a set such that } x \notin x\}.$$

Is $S \in S$?

- If $S \in S$ then by the definition of S , $S \notin S$.
- If $S \notin S$ then by the definition of S , $S \in S$.

ARGHHH!!@#!

EXAMPLE \triangleright There is a town with a barber who shaves all and only the people who don't shave themselves.

Question: Who shaves the barber?

4.3 Cardinality of a Set

Definition 19 IF S is finite, then the *cardinality* of S (written $|S|$) is the number of (distinct) elements in S .

EXAMPLE \triangleright $S = \{1, 2, 5, 6\}$ has cardinality $|S| = 4$.

EXAMPLE \triangleright $S = \{1, 1, 1, 1, 1\}$ has cardinality $|S| = 1$.

EXAMPLE \triangleright $|\emptyset| = 0$.

EXAMPLE \triangleright $S = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}$ has cardinality $|S| = 3$.

We will discuss infinite cardinality later.

EXAMPLE \triangleright The cardinality of $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ is infinite.

4.4 Power Set

In other words, the power set of S is the set of all subsets of S

Definition 20 If S is a set, then the *power set* of S , written 2^S or $\mathcal{P}(S)$ or $P(S)$ is defined by:

$$2^S = \{x : x \subseteq S\}$$

EXAMPLE $\triangleright S = \{a\}$ has the power set $2^S = \{\emptyset, \{a\}\}$.

EXAMPLE $\triangleright S = \{a, b\}$ has the power set $2^S = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$.

EXAMPLE $\triangleright S = \emptyset$ has the power set $2^S = \{\emptyset\}$.

EXAMPLE $\triangleright S = \{\emptyset\}$ has the power set $2^S = \{\emptyset, \{\emptyset\}\}$.

EXAMPLE $\triangleright S = \{\emptyset, \{\emptyset\}\}$ has the power set $2^S = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$.

Fact 7 If S is finite, then $|2^S| = 2^{|S|}$.

Question. Who cares if $\{\emptyset, \{\emptyset, \{\emptyset\}\}, \{\{\emptyset\}\}$ is a subset of the power set of

$$\{\{\emptyset\}, \emptyset, \{\emptyset, \{\emptyset\}\}, \{\{\emptyset\}\}\}?$$

Answer. No body!

Question. Then why ask the question?

Answer. We must learn to pick nits.

4.5 Cartesian Product

Definition 21 The *Cartesian product* of two sets A and B is the set

$$A \times B = \{\langle a, b \rangle : a \in A \wedge b \in B\}$$

of ordered pairs. Similarly, the Cartesian product of n sets A_1, \dots, A_n is the set

$$A_1 \times \dots \times A_n = \{\langle a_1, \dots, a_n \rangle : a_1 \in A_1 \wedge \dots \wedge a_n \in A_n\}$$

of ordered n -tuples.

EXAMPLE \triangleright If $A = \{\text{john, mary, fred}\}$ and $B = \{\text{doe, roe}\}$, then

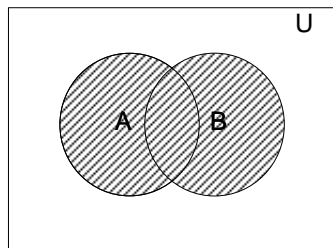
$$A \times B = \{\langle \text{john, doe} \rangle, \langle \text{mary, doe} \rangle, \langle \text{fred, doe} \rangle, \langle \text{john, roe} \rangle, \langle \text{mary, roe} \rangle, \langle \text{fred, roe} \rangle\}$$

Fact 8 If A and B be finite, then $|A \times B| = |A| \cdot |B|$.

4.6 Set Operations

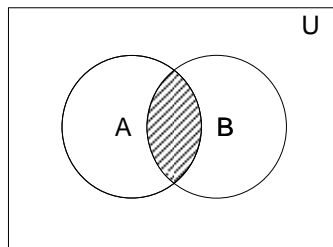
4.6.1 Union of Two Sets

Definition 22 For sets A and B , the *union* of A and B is the set $A \cup B = \{x : x \in A \vee x \in B\}$.



4.6.2 Intersection of Two Sets

Definition 23 For sets A and B , the *intersection* of A and B is the set $A \cap B = \{x : x \in A \wedge x \in B\}$.



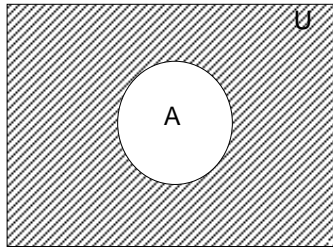
EXAMPLE $\triangleright \{\text{presidents}\} \cap \{\text{deceased}\} = \{\text{deceased presidents}\}$

EXAMPLE $\triangleright \{\text{US presidents}\} \cap \{\text{people in this room}\} = \emptyset$

Note. If for sets A and B , $A \cap B = \emptyset$, A and B are said to be *disjoint*.

4.6.3 Complement of a Set

Definition 24 Assume U be the universe of discourse (U contains all elements under consideration). Then the *complement* of a set A is the set $\overline{A} = \{x \in U : x \notin A\}$ which is the same as the set $\{x : x \in U \wedge x \notin A\}$ or $\{x : x \notin A\}$ when U is understood implicitly.

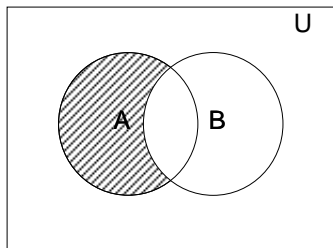


Note. $\overline{\emptyset} = U$ and $\overline{U} = \emptyset$.

EXAMPLE \triangleright IF $U = \{1, 2, 3\}$ and $A = \{1\}$ then $\overline{A} = \{2, 3\}$.

4.6.4 Set Difference of two Sets

Definition 25 For sets A and B the *set difference* of A from B is defined as the set $A - B = \{x : x \in A \wedge x \notin B\} = A \cap \overline{B}$.



EXAMPLE \triangleright $\{\text{people in class}\} - \{\text{sleeping people}\} = \{\text{awake people in class}\}$.

EXAMPLE \triangleright $\{\text{US presidents}\} - \{\text{baboons}\} = \{\text{US presidents}\}$.

?

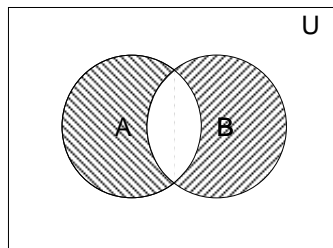
Note. $A - \overline{A} = A$.

4.6.5 Symmetric Difference (Exclusive-Or) of two Sets

Definition 26 For sets A and B , the *symmetric difference* of A and B is the set $A \oplus B = \{x : (x \in A \wedge x \notin B) \vee (x \notin A \wedge x \in B)\}$.

From the definition:

$$\begin{aligned}
 A \oplus B &= \{x : (x \in A \wedge x \notin B) \vee (x \notin A \wedge x \in B)\} \\
 &= \{x : (x \in A - B) \vee (x \in B - A)\} \\
 &= \{x : x \in (A - B) \cup (B - A)\} \\
 &= (A - B) \cup (B - A).
 \end{aligned}$$



4.7 Famous Set Identities

<i>Identity</i>	$A \cap U = A$
	$A \cup \emptyset = A$

<i>Domination</i>	$A \cup U = U$
	$A \cap \emptyset = \emptyset$

<i>Idempotent</i>	$A \cup A = A$
	$A \cap A = A$

<i>Double Negation</i>	$\overline{(\overline{A})} = A$
------------------------	---------------------------------

<i>Commutative</i>	$A \cup B = B \cup A$
	$A \cap B = B \cap A$

<i>Associative</i>	$(A \cup B) \cup r = A \cup (B \cup r)$
	$(A \cap B) \cap r = A \cap (B \cap r)$

<i>Excluded Middle Uniqueness</i>	$A \cup \bar{A} = U$
	$A \cap \bar{A} = \emptyset$

<i>Distributive</i>	$A \cup (B \cap r) = (A \cup B) \cap (A \cup r)$
	$A \cap (B \cup r) = (A \cap B) \cup (A \cap r)$

<i>De Morgan's</i>	$\overline{(A \cup B)} = \bar{A} \cap \bar{B}$
<i>De Morgan's</i>	$\overline{(A \cap B)} = \bar{A} \cup \bar{B}$

◇ *Don't memorize these; understand them!* ◇

4.8 4 Ways to Prove Identities

To show that two sets A and B are equal we can use one of the following four techniques. 1. *Show that $A \subseteq B$ and $B \subseteq A$.*

EXAMPLE ▷ We show that $\overline{A \cup B} = \bar{A} \cap \bar{B}$ (first proof).

(\subseteq) If $x \in \overline{A \cup B}$, then $x \notin A \cup B$. So, $x \notin A$ and $x \notin B$, and thus $x \in \bar{A} \cap \bar{B}$.

(\supseteq) If $x \in \bar{A} \cap \bar{B}$ then $x \notin A$ and $x \notin B$. So, $x \notin A \cup B$ and therefor $x \in \overline{A \cup B}$.

2. *Use a "membership table".*

Works the same way as a truth table in which we use 0s and 1s to indicate the followings.

0: x is *not* in the specified set.

1: x is in the specified set.

EXAMPLE ▷ We show that $\overline{A \cup B} = \bar{A} \cap \bar{B}$ (second proof).

A	B	\overline{A}	\overline{B}	$A \cup B$	$\overline{A \cup B}$	$\overline{A} \cap \overline{B}$
1	1	0	0	1	0	0
1	0	0	1	1	0	0
0	1	1	0	1	0	0
0	0	1	1	0	1	1

3. Use previous identities.

EXAMPLE ▷ We show that $\overline{A \cup B} = \overline{A} \cap \overline{B}$ (third proof).

$$\begin{aligned}
 \overline{(A \cup B)} &\equiv \overline{\overline{\overline{A}} \cup \overline{\overline{B}}} && \text{by double complement} \\
 &\equiv \overline{\overline{A} \cap \overline{B}} && \text{by De Morgan's law 2} \\
 &\equiv \overline{\overline{A}} \cap \overline{\overline{B}} && \text{by double complement} \\
 &\equiv \overline{A} \cap \overline{B}
 \end{aligned}$$

4. Use logical equivalences of the defining propositions.

EXAMPLE ▷ We show that $\overline{A \cup B} = \overline{A} \cap \overline{B}$ (fourth proof).

$$\begin{aligned}
 \overline{A \cup B} &= \{x : \neg(x \in A \vee x \in B)\} \\
 &= \{x : (\neg x \in A) \wedge (\neg x \in B)\} \\
 &= \{x : (x \in \overline{A}) \wedge (x \in \overline{B})\} \\
 &= \{x : (x \in \overline{A} \cap \overline{B})\} \\
 &= \overline{A} \cap \overline{B}
 \end{aligned}$$

4.9 Generalized Union and Intersection

Definition 27

$$\begin{aligned}
 \bigcup_{i=1}^n A_i &= A_1 \cup A_2 \cup \dots \cup A_n \\
 &= \{x : x \in A_1 \vee x \in A_2 \vee \dots \vee x \in A_n\}
 \end{aligned}$$

$$\begin{aligned}
 \bigcap_{i=1}^n A_i &= A_1 \cap A_2 \cap \dots \cap A_n \\
 &= \{x : x \in A_1 \wedge x \in A_2 \wedge \dots \wedge x \in A_n\}
 \end{aligned}$$

4.10 Representation of Sets as Bit Strings

Definition 28 Let $U = \{x_1, x_2, \dots, x_n\}$ be the universe and A be some set in this universe. Then we can represent A by a sequence of bits (0s and 1s) of length n in which the i -th bit is 1 if $x_i \in A$ and 0 otherwise. We call this string the *characteristic vector* of A .

EXAMPLE \triangleright If $U = \{x_1, x_2, x_3, x_4, x_5\}$ and $A = \{x_1, x_2, x_4\}$, then the characteristic vector of A will be 11010.

Fact 9 To find the characteristic vector for $A \cup B$ we take the *bitwise “or”* of the characteristic vectors of A and B .

To find the characteristic vector for $A \cap B$ we take the *bitwise “and”* of the characteristic vectors of A and B .

EXAMPLE \triangleright If $U = \{x_1, x_2, x_3, x_4, x_5\}$ and $A = \{x_1, x_2, x_4\}$ and $B = \{x_1, x_3, x_4, x_5\}$, then the characteristic vector of A will be 11010 and the characteristic vector of B will be 10111. Then for the characteristic vector of $A \cup B$ and $A \cap B$ we have get.

$$\begin{array}{rcccccc}
 A: & 1 & 1 & 0 & 1 & 0 \\
 B: & 1 & 0 & 1 & 1 & 1 \\
 \hline
 A \cup B & 1 & 1 & 1 & 1 & 1 \\
 A \cap B & 1 & 0 & 0 & 1 & 0
 \end{array}$$

4.11 Principle of Inclusion/Exclusion

We start with an example.

EXAMPLE \triangleright How many people wear hats or gloves?

$$\begin{aligned}
 \text{Number of people wearing hats or gloves} &= \text{Number of people wearing hats} + \\
 &\quad \text{Number of people wearing gloves} - \\
 &\quad \text{Number of people wearing both} \\
 &\quad \text{hats and gloves.}
 \end{aligned}$$

In general it is easy to see that $|A \cup B| = |A| + |B| - |A \cap B|$.

EXAMPLE \triangleright If we have 217 CS majors, 157 taking CS 125, 145 taking CS 173, and 98 taking both, how many are not taking either?

Answer: The number of people who are not taking either is 217 minus the

number of people who are taking CS 125 or CS 173 which is $217 - (157 + 145 - 98) = 217 - 204 = 13$.

4.12 General Inclusion/Exclusion

It can be checked easily when we have three sets A , B , and C that

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

This can be generalized as follows.

Fact 10 For sets A_1, A_2, \dots, A_n the inclusion/exclusion principle is stated as follows:

$$\begin{aligned} |A_1 \cup A_2 \cup \dots \cup A_n| &= \sum_{1 \leq i \leq n} |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| \\ &\quad + \dots + (-1)^n |A_1 \cap A_2 \cap \dots \cap A_n| \end{aligned}$$

4.13 Functions

Definition 29 A function $f : A \rightarrow B$ is a subset of $A \times B$ such that $\forall a \in A \exists! b \in B \langle a, b \rangle \in f$.

From the above definition we *cannot* have $\langle a, b \rangle \in f$, and $\langle a, c \rangle \in f$ for $b \neq c$. Also there must be for each a , some b such that $\langle a, b \rangle \in f$.

Didn't say that $\forall b \exists! a \langle a, b \rangle \in f$.

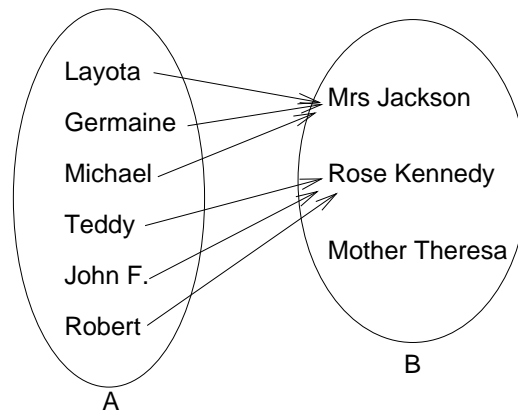
Note. There can be $\langle a, b \rangle \in f$ and $\langle c, b \rangle \in f$.

We write $f(a) = b$ to indicate that b is the unique element of B such that $\langle a, b \rangle \in f$.

though this intuition fails . . . more later.

Intuitively we can think of f as a *rule, assigning* to each $a \in A$ a $b \in B$.

EXAMPLE $\triangleright A = \{\text{Latoya, Germaine, Michael, Teddy, John F., Robert}\}$
 $B = \{\text{Mrs Jackson, Rose Kennedy, Mother Theresa}\}$
 $f : A \rightarrow B, f(a) = \text{mother}(a)$



f is a function because everyone has exactly one (biological) mother. Mrs Jackson has more than 1 child. Functions do not require $\forall b \in B \exists! a \in A f(a) = b$.

Mother Theresa (paradoxically) is not a mother. Functions do not require $\forall b \in B \exists a \in A f(a) = b$.

Definition 30 For a function $f : A \rightarrow B$, A is called the *domain* of f and B is called the *co-domain* of f .

For a set $S \subseteq A$ the *image* of A is defined as

$$\text{image}(A) = \{f(a) : a \in S\}$$

Note. When I was a lad we used “range” instead of “co-domain”. Now range is used sometimes as “co-domain” and sometimes as “image”.

EXAMPLE \triangleright In the previous example:

$$\text{image}(\{\text{Latoya, Michael, John}\}) = \{\text{Mrs Jackson, Rose Kennedy}\}$$

and

$$\text{image}(\{\text{Latoya, Michael}\}) = \{\text{Mrs Jackson}\}.$$

Note.

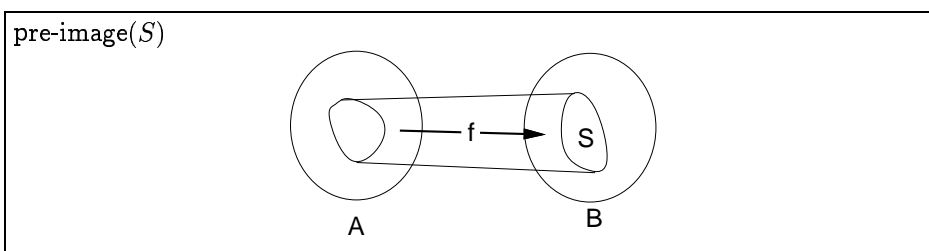
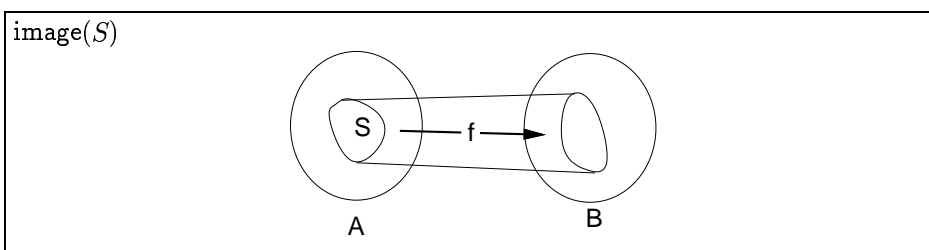
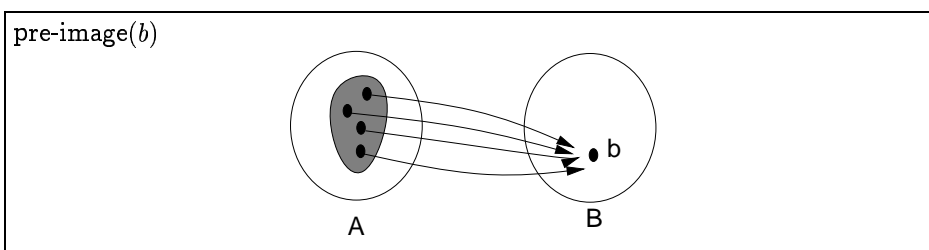
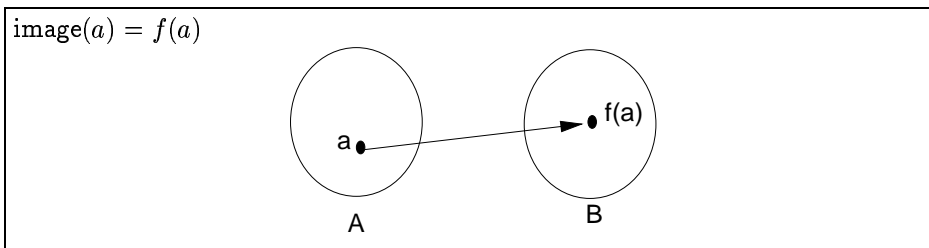
$$\text{image}(S) = \{f(a) : a \in S\} = \{b \in B : \exists a \in A f(a) = b\}$$

Definition 31 The *range* of a function $f : A \rightarrow B$ is $\text{image}(A)$.

$a \in A$ is a *pre-image* of $b \in B$ if $f(a) = b$. For a set $S \subseteq B$, the pre-image of S is defined as

$$\text{pre-image}(S) = \{\text{pre-image}(b) : b \in S\}$$

The above concepts are represented pictorially below. Suppose $f : A \rightarrow B$ is a function.



Questions. What is the relationship between these two?

$$\text{image}(\text{pre-image}(S)) \quad \text{and} \quad \text{pre-image}(\text{image}(S))$$

Definition 32 A function f is *one-to-one*, or *injective*, or *is an injection*, if

$$\forall a, b, c (f(a) = b \wedge f(c) = b) \rightarrow a = c.$$

In other words each $b \in B$ has at most one pre-image.

EXAMPLE ▷ The functions $f(x) = \text{mother}(x)$ in the previous example was *not* one-to-one because $f(\text{Latoya}) = f(\text{Michael}) = \text{Mrs Jackson}$.

EXAMPLE ▷ $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, where $f(x) = x^2$ is one-to-one, since each positive real x has exactly one positive pre-image (\sqrt{x}).

EXAMPLE ▷ $f : \mathbb{R} \rightarrow \mathbb{R}$, where $f(x) = x^2$ is *not* one-to-one. $f(2) = 4 = f(-2)$.

Note. In terms of diagrams, one-to-one means that no more than one arrow go to the same point in B .

Definition 33 A function $f : A \rightarrow B$ is *onto*, or *surjective*, or *is a surjection* if

$$\forall b \in B \exists a \in A \quad f(a) = b$$

In other words, everything in B has at least one pre-image.

EXAMPLE ▷ In the *mothers* example, the function $f(x) = \text{mother}(x)$ is *not* surjective because Mother Theresa is not the mother of anyone in A .

EXAMPLE ▷ $f : \mathbb{R} \rightarrow \mathbb{R}$, where $f(x) = x^2$ is *not* surjective, since -1 has no squared roots in \mathbb{R} .

EXAMPLE ▷ $f : \mathbb{R} \rightarrow \mathbb{R}^+$, where $f(x) = x^2$ is surjective, since every positive number has at least one squared root.

In fact 2 squared roots.

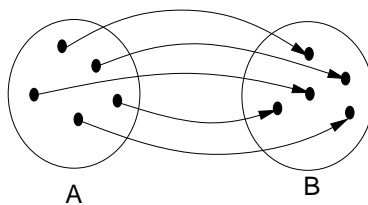
4.14 Bijections

Definition 34 A function $f : A \rightarrow B$ is a *bijection*, or a *one-to-one correspondence*, if and only if f is one-to-one and onto (injective and surjective).

In other words, $f : A \rightarrow B$ is a bijection iff:

- every $b \in B$ has *exactly one* pre-image, and
- every $a \in A$ has *exactly one* image.

This is always true by the definition of a functions.



Definition 35 The *inverse* function of a bijection f , denoted by f^{-1} is the function $f^{-1} : B \rightarrow A$ where $f^{-1}(b)$ is defined as the unique a such that $f(a) = b$.

4.15 Cardinality of Infinite Sets

Motivating Questions:

- Are there more even numbers, or odd numbers?
- Are there more natural numbers than evens?
- Are there more evens than numbers divisible by 3?

The answer to the above questions depends on what we mean by “more than”.

Definition 36 Two sets A and B have the same cardinality if and only if there exists a bijection between A and B (written $A \sim B$).

Note that this makes sense for finite sets.

A set is *countably infinite* if it can be put into one-to-one correspondence with \mathbb{N} (the natural numbers).

A set is *countable* if it is finite, or countably infinite.

EXAMPLE $\triangleright \{2, 4, 6, 8, \dots\} \sim \{1, 3, 5, 7, 9, \dots\}$: $f(x) = x - 1$.

EXAMPLE $\triangleright \mathbb{N} = \{0, 1, 2, 3, \dots\} \sim \{0, 2, 4, 6, \dots\}$: $f(x) = 2x$.

EXAMPLE $\triangleright \{0, 2, 4, 6, \dots\} \sim \{2, 4, 6, \dots\}$: $f(x) = x + 2$.

EXAMPLE $\triangleright \{\text{perfect squares}\} \sim \mathbb{N}$: $f(x) = \sqrt{x}$.

Fact 11 An *infinite* set S is countably infinite iff it is possible to *list* all elements of S without repetitions, so that each $s \in S$ appears *with* or *without* repetitions in the list.

A list is another representation of a bijection.

EXAMPLE \triangleright Evens: 0, 2, 4, 6, 10, 12, ...

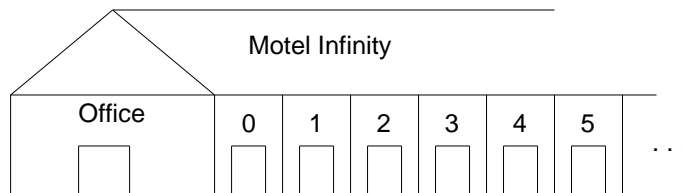
EXAMPLE \triangleright Odds: 1, 3, 5, 7, 9, 11, ...

EXAMPLE \triangleright Squares: 0, 1, 2, 4, 9, 16, ...

EXAMPLE \triangleright Rationals: Hmmm, we'll see!

EXAMPLE ▷ Reals: No! this can't be done (not countably infinite).

Now consider the Motel Infinity shown below with no vacancy:



-1 can go to room 0 if everybody shifts down a room. Thus $\mathbb{N} \cup \{-1\} \sim \mathbb{N}$.

Imagine instead of -1 we needed room for "camp ∞ " (A countably infinite set). Still we could free up enough room by moving for every i the occupant of room i to room $2i$ and giving the i -th person of the camp ∞ bus the room $2i + 1$.

Fact 12 If A and B are countably infinite, then $A \cup B$ is countably infinite.

What happens if a countably infinite number of camp ∞ buses need room in Motel Infinity?

4.15.1 Rational Numbers are Countably Infinite

By definition, rational numbers are those that can be written in the form $\frac{p}{q}$ where p and q are natural numbers and $q \neq 0$.

A naive attempt to enumerate all the rationals then can be to list them as

$$\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \dots, \frac{2}{1}, \frac{2}{2}, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}, \dots, \frac{3}{1}, \frac{3}{2}, \frac{3}{3}, \frac{3}{4}, \frac{3}{5}, \dots$$

But this does not work because the first part of the list $(\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \dots)$ never finishes and therefore $\frac{2}{1}$ is never reached.

A second approach will be to use the table below.

		$p \rightarrow$								
		0	1	2	3	4	5	6	...	
$q \downarrow$	1	0	2	5	9	14				
	2	1	4	8	13					
	3	3	7	12						
	4	6	11							
	5	10								
	6	...								
	7									
	⋮									

There is still a small problem with the above approach: some rationals are listed twice or more.

Solution: Skip over those not in the “lowest term”. This way every positive rational appears in some box and we enumerate the boxes:

$$\frac{0}{1}, \frac{1}{2}, \frac{1}{1}, \frac{2}{3}, \frac{1}{2}, \frac{2}{1}, \frac{3}{4}, \frac{1}{3}, \frac{2}{2}, \frac{3}{1}, \dots$$

4.15.2 Real Numbers are NOT Countably Infinite

This is due to George Cantor.

We will show that $[0, 1]$ cannot be enumerated.

Recall that each real number in $[0, 1]$ can be represented as an infinite decimal $0.d_1d_2d_3d_4d_5d_6\dots$, where d_i 's are digits in $\{0, \dots, 9\}$.

EXAMPLE $\triangleright \pi - 3 = 0.1415926535897\dots$

EXAMPLE $\triangleright \frac{1}{2} = 0.500000000000\dots$

Assume that there *was* an enumerate of reals in $[0, 1]$: r_1, r_2, r_3, \dots such that all reals in $[0, 1]$ would appear exactly once.

r_1	0.	1	4	1	5	9	2	6	3	...
r_2	0.	5	0	0	0	0	0	0	0	...
r_3	0.	1	6	1	7	8	0	3	4	...
r_4	0.	1	2	3	4	5	6	7	8	...
⋮						⋮				
r_n	0.	d_1	d_2	d_3	d_4	...	d_n	d_{n+1}	d_{n+2}	...

Consider the real number r defined by *diagonalizing*.

$$r = .b_1b_2b_3b_4b_4\dots$$

where b_i is 1 plus the i -th digit of r_i ($9 + 1 = 0$).

EXAMPLE \triangleright For the table above we get $r = 0.2125\dots$

r is a real number between 0 and 1 but was not listed because $r \neq r_i$ for any i .

Why? Because r and r_i differ in decimal position i .

Thus the enumeration wasn't one after all, contradicting the existence of such an enumeration.

4.15.3 Can All Functions Be Computed by Programs?

Suppose our alphabet have 1000 characters. We can list all strings of this alphabet in order of their length.

- length 0 strings: 1
- length 1 strings: 1,000
- length 2 strings: 1,000,000
- length 3 strings: 10^9
- ...

Some *but not all* of these strings are computer programs. The shortest computer program that looks like this

```
begin
end.
```

requires 10 characters. Thus we can enumerate all programs (plus some meaningless strings) and therefore the set of programs is countably infinite, i.e.

$$\{\text{all programs}\} \sim \mathbb{N}.$$

We like to show that there exists a function f that no program can compute it. We show even the stronger statement that there is some function that outputs a single digit and still cannot be computed by any program. Let

$$\mathcal{F} = \{f : \mathbb{N} \rightarrow \{0, 1, \dots, 9\}\}.$$

Let us give some examples of such functions.

EXAMPLE $\triangleright f(x) = 1$ (The constant function).

EXAMPLE $\triangleright f(x) = x \bmod 10$ (remainder of x divided by 10).

EXAMPLE $\triangleright f(x) = \text{first digit of } x$.

EXAMPLE $\triangleright f(x) = \begin{cases} 0 & \text{if } x \text{ is even} \\ 1 & \text{if } x \text{ is odd} \end{cases}$

How many such function do we have? Each function $f : \mathbb{N} \rightarrow \{0, \dots, 9\}$ can be represented by a single real number in $[0, 1]$. This number is exactly $0.f(0)f(1)f(2)f(3)\dots$.

EXAMPLE \triangleright The function f shown be the table below

x	0	1	2	3	4	5	6	7	8	...
$f(x)$	1	4	1	5	9	2	6	5	3	...

can be represented by the real number $0.141592653\dots$.

Conversely, each number in $[0, 1]$ can be thought of as a function from the natural numbers to single digits.

EXAMPLE \triangleright The number 0.72 corresponds to the function f where $f(0) = 7$, $f(1) = 2$, and $f(x) = 0$ for $x \geq 2$.

Thus we have shown that

$$\mathcal{F} \sim [0, 1] \sim \mathbb{R}.$$

This means that there are much more many functions as there are programs and therefore, some functions are not computed by any programs.

Chapter 5

Boolean Functions

5.1 Propositions vs. Boolean Functions

A proposition is really a special kind of function.

EXAMPLE $\triangleright f : \{T, F\} \rightarrow \{T, F\} \quad f(p) = \neg p.$

p	$f(p)$
F	T
T	F

EXAMPLE $\triangleright f(p, q, r) = (p \wedge r) \vee \bar{q}$

$f : \{T, F\} \times \{T, F\} \times \{T, F\} \rightarrow \{T, F\}$ or $f : \{T, F\}^3 \rightarrow \{T, F\}.$

$f(F, F, F) = T$

$f(F, T, F) = F$

5.2 New Notation

Old	New
T	1
F	0
\wedge	.
\vee	+
\neg	$\bar{\quad}$

EXAMPLE $\triangleright f(p) = \bar{p}$

p	$f(p)$
0	1
1	0

EXAMPLE $\triangleright f(p, q) = p + q$

p	q	$p + q$
0	0	0
0	1	1
1	0	1
1	1	1

EXAMPLE $\triangleright f(p, q) = pq$

p	q	pq
0	0	0
0	1	0
1	0	0
1	1	1

5.3 Boolean Functions of n variables

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

EXAMPLE $\triangleright f(x_1, x_2, \dots, x_n) = ((x_1 + x_2)x_3 + \overline{x_4})(x_5 + \overline{x_6x_7})$

x_1	x_2	x_3	\dots	x_n	$f(x_1, x_2, \dots, x_n)$
0	0	0	\dots	0	0
0	0	0	\dots	1	0
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
1	1	1	\dots	1	1

Question. How many rows are needed to completely define a boolean function of n variables?

Of course, it is often much easier (shorter) to describe the function symbolically.

Answer. 2^n

Question. How many different functions $f(x_1, \dots, x_n)$ can be defined?

If $n = 1$

x_1	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

There are 2 rows and 4 different columns.

If $n = 2$

x_1	f_0	f_1	f_2	f_3	f_4	f_5	...	f_{15}
0	0	0	0	0	0	0	...	1
0	1	0	0	0	1	1	...	1
1	0	0	1	1	0	0	...	1
1	1	0	1	0	1	1	...	1

There are 4 rows and 16 different columns. In general if there are k rows, there will be as many columns as the number of rows a truth table with k variables would need, i.e. 2^k . Since $k = 2^n$ there will be 2^{2^n} columns.

Question. Can we write all of them, using only $+$, $.$, and \neg , i.e. Is the operator set $\{+, ., \neg\}$ functionally complete?

5.4 Standard Sum of Products

Also called Disjunctive Normal Form (DNF): Look at the rows which the values are 1. For each of these rows, construct the logical AND (\wedge) to yield a 1. Then, OR (\vee) all of these products together to get a sum of products.

EXAMPLE \triangleright For the following truth table,

p	q	r	b	<i>minterms</i>
0	0	0	1	$\neg p \neg q \neg r$
0	0	1	0	
0	1	0	0	
0	1	1	1	$\neg pqr$
1	0	0	0	
1	0	1	0	
1	1	0	1	$pq\neg r$
1	1	1	1	pqr

we get:

$$b = \neg p \neg q \neg r + \neg pqr + pq\neg r + pqr.$$

5.5 Standard Product of Sums

Also called Conjunctive Normal Form (CNF): Look at the rows which the values are 0. For each of these rows, construct the logical OR of the negations to yield a 0. Then, AND all of these products together to get product of sums.

EXAMPLE ▷ For the following truth table,

p	q	r	b	<i>minterms</i>
0	0	0	1	
0	0	1	0	$p + q + \neg r$
0	1	0	0	$p + \neg q + r$
0	1	1	1	
1	0	0	0	$\neg p + q + r$
1	0	1	0	$\neg p + q + \neg r$
1	1	0	1	
1	1	1	1	

we get:

$$b = (p + q + \neg r)(p + \neg q + r)(\neg p + q + r)(\neg p + q + \neg r).$$

Since

- *Every boolean function is representable as a truth table, and*
- *Every table is representable using +, ., and ¬,*

therefore *Every boolean function is representable using +, ., and ¬, i.e. {+, ., ¬} is functionally complete.*

Homework. Show that {+, ¬} and {., ¬} are also functionally complete but {+, .} is not.

5.6 Karnaugh Maps

The goal of this section is to find *small* sum-of-products for given boolean functions. There is a graphical technique for designing minimum sum-of-product expressions from truth tables. This technique puts truth tables into two-dimensional arrays called *Karnaugh Maps*.

Each entry in the Map represents a row of the corresponding truth table. This method works well for Boolean functions up to four variables. For example, the Karnaugh Map for $p \rightarrow q$ looks like this

	q	0	1
p	0	1	1
1	0	0	1

- Each 1×1 cell containing 1 gives a minterm.

$$\overline{p}q + \overline{p}q + pq$$

- Two adjacent 1 cells gives a shorter term.

$$\overline{p} + pq$$

- The boxes can overlap and the goal is to cover the 1 cells with boxes.

$$\overline{p} + q$$

EXAMPLE ▷ Here is an example of a Karnaugh Map for three variables. Note that the order of labels on the cells is selected in such a way that adjacent cells correspond to variable settings that differ in exactly *one* position.

	qr	00	01	11	10
p	0	0	1	1	1
1	0	0	1	0	0

The table is for the following truth table.

p	q	r	b
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

The given map simplifies to $\bar{p}q + \bar{q}r$.

EXAMPLE ▷ Another three variable Karnaugh Map. The simplified function by this map is $yz + xz + xy$.

		yz			
		00	01	11	10
x	0	0	0	1	0
	1	0	1	1	1


EXAMPLE ▷ And this is a four variable Karnaugh Map that gives us the function $\bar{p}\bar{s} + \bar{q}\bar{s} + pqrs$.

		rs			
		00	01	11	10
pq	00	1	0	0	1
	01	1	0	0	1
	11	0	0	1	0
	10	1	0	0	1

5.7 Design and Implementation of Digital Networks

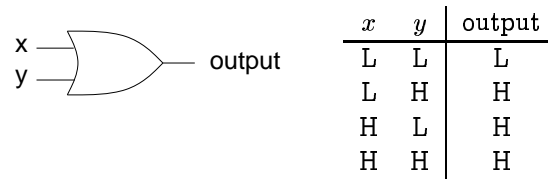
- 0: Low voltage (0V)
- 1: High voltage (6V or 2.5V)

The AND Gate

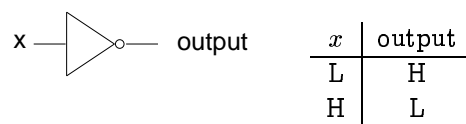
x		output
y		

x	y	output
L	L	L
L	H	L
H	L	L
H	H	H

The OR Gate



The Inverter



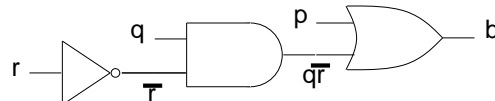
EXAMPLE ▷ We want to activate a sound buzzer that if

1. temperature of engine exceeds 200 degrees, or
2. automobile is in gear and driver's seat-belt is not buckled.

Let us represent the above signals with the following variables

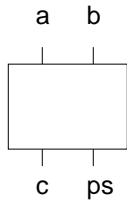
- b : sound buzzer.
- p : temperature of the engine exceeds 200 degrees.
- q : automobile is in gear.
- r : driver's seat-belt is buckled.

We want to in fact make a network for $b = p + q\bar{r}$.



5.7.1 One Bit Adder

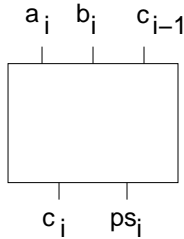
$$\begin{array}{r}
 0 \\
 + 0 \\
 \hline
 0
 \end{array}
 \quad
 \begin{array}{r}
 0 \\
 + 1 \\
 \hline
 1
 \end{array}
 \quad
 \begin{array}{r}
 1 \\
 + 0 \\
 \hline
 1
 \end{array}
 \quad
 \begin{array}{r}
 1 \\
 + 1 \\
 \hline
 1 \ 0
 \end{array}$$



a	b	ps	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

5.7.2 Half Adder

	1	0	1	1
+	0 ₀	0 ₁	0 ₁	1
	1	1	0	0



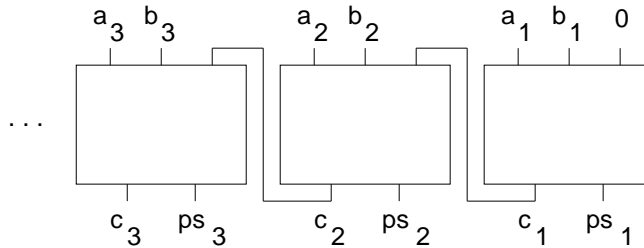
a_i	b_i	c_{i-1}	ps_i	c_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Simplifying using Karnaugh maps we get

$$ps_i = \overline{a_i} \overline{b_i} c_{i-1} + \overline{a_i} b_i \overline{c_{i-1}} + a_i b_i c_{i-1} + a_i \overline{b_i} \overline{c_{i-1}}$$

$$c_i = b_i c_{i-1} + a_i b_i + a_i c_{i-1} = (b_i + a_i) c_{i-1} + a_i b_i.$$

We make an n -bit *full adder* from n half adders as in the figure below.



Chapter 6

Familiar Functions and Growth Rate

6.1 Familiar Functions

Polynomials

$$f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_nx^0$$

EXAMPLE $\triangleright f(x) = x^3 - 2x^2 + 14$.

Exponentials

$$f(x) = c^{dx}$$

EXAMPLE $\triangleright f(x) = 3^{10x}$.

EXAMPLE $\triangleright f(x) = e^x$.

Logarithms

$$\log_2 x = y \quad \text{such that} \quad 2^y = x$$

Read Appendix 1

Note. In this course \lg is used for \log_2 and unless stated otherwise \log is also used for \log_2 .

$$\ln x = \log_e x = \int_1^x \frac{1}{t} dt$$

Ceiling

$$\lceil x \rceil = \text{least integer } y \text{ such that } x \leq y$$

EXAMPLE $\triangleright \lceil 1.7 \rceil = 2$.

EXAMPLE $\triangleright \lceil -1.7 \rceil = -1$.

Floor

$$\lfloor x \rfloor = \text{greatest integer } y \text{ such that } y \leq x$$

EXAMPLE $\triangleright \lfloor 1.7 \rfloor = 1$.

EXAMPLE $\triangleright \lfloor -1.7 \rfloor = -2$.

6.2 Sequences

Definition 37 A *sequence* $\{a_i\}$ is a function $f : \mathbb{N} \rightarrow \mathbb{R}$ where we write a_i to indicate $f(i)$.

EXAMPLE \triangleright The sequence $\{a_i\}$ where $a_i = i$ is: $a_1 = 1, a_2 = 2, \dots$

EXAMPLE \triangleright The sequence $\{a_i\}$ where $a_i = i^2$ is: $a_1 = 1, a_2 = 4, a_3 = 9, \dots$

6.3 Summation

Definition 38 We use the notation

$$\sum_{i=1}^k a_i$$

to indicate

$$a_1 + a_2 + \cdots + a_k.$$

Similarly we use the notation

$$\sum_{i=1}^{\infty} a_i$$

to indicate

$$\lim_{n \rightarrow \infty} \sum_{i=1}^n a_i.$$

It is easy to verify the validity of the *linearity* in summation:

$$\sum_{k=1}^n ca_k + b_k = c \sum_{k=1}^n a_k + \sum_{k=1}^n b_k.$$

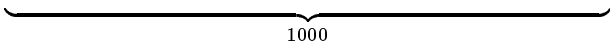
6.3.1 Famous Sums

Question. What is S defined as the following sum?

$$S = 1 + 2 + 3 + 4 + 5 + 6 + \cdots + 1000.$$

Answer. Let us add S to itself but in reverse order.

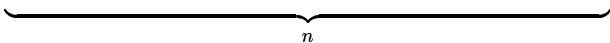
$$\begin{array}{r} S = 1 + 2 + 3 + \cdots + 1000 \\ +S = 1000 + 999 + 998 + \cdots + 1 \\ \hline 2S = 1001 + 1001 + 1001 + \cdots + 1001 \end{array}$$



Therefore $2S = 1000 \times 1001$ and $S = \frac{1000 \times 1001}{2} = 500,500$.

Let's generalize this:

$$\begin{array}{r} S = 1 + 2 + 3 + \cdots + n \\ +S = n + n-1 + n-2 + \cdots + 1 \\ \hline 2S = n+1 + n+1 + n+1 + \cdots + n+1 \end{array}$$



Thus we get

$$S = \sum_{k=1}^n k = \frac{n(n+1)}{2}.$$

EXAMPLE ▷ What is the sum $1 + 3 + 5 + 7 + \cdots + 2n - 1$?

$$\begin{aligned}
 \sum_{k=1}^n 2k - 1 &= 1 + 3 + 5 + 7 + \cdots + 2n - 1 \\
 &= 2 + 4 + 6 + \cdots + 2n - n \\
 &= 2(1 + 2 + 3 + \cdots + n) - n \\
 &= 2 \frac{n(n+1)}{2} - n \\
 &= n(n+1) - n \\
 &= n^2 + n - n \\
 &= n^2
 \end{aligned}$$

6.3.2 Geometric Series

$$\begin{aligned}
 \sum_{k=0}^n r^k &= 1 + r + r^2 + r^3 + \cdots + r^n \\
 r \sum_{k=0}^n r^k &= r + r^2 + r^3 + r^4 + \cdots + r^{n+1}
 \end{aligned}$$

Subtracting top from bottom

$$\begin{aligned}
 (r-1) \sum_{k=0}^n r^k &= r^{n+1} - 1 \\
 \sum_{k=0}^n r^k &= \frac{r^{n+1} - 1}{r - 1}
 \end{aligned}$$

If $r < 1$ then the series converges as n approaches infinity.

$$\begin{aligned}
 \sum_{k=0}^{\infty} r^k &= 1 + r + r^2 + r^3 + \cdots + r^n + \cdots \\
 r \sum_{k=0}^{\infty} r^k &= r + r^2 + r^3 + r^4 + \cdots + r^{n+1} + \cdots
 \end{aligned}$$

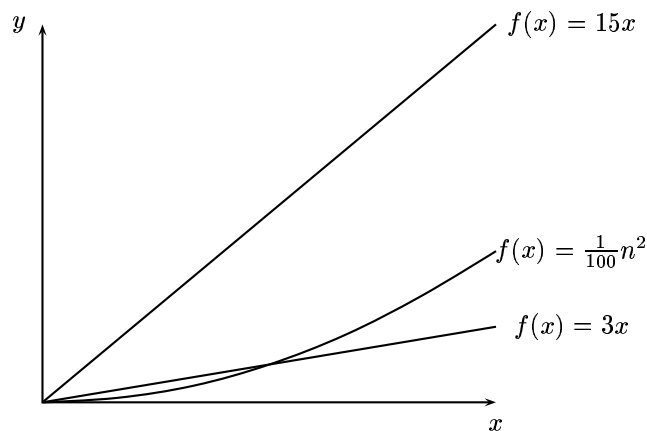
Subtracting bottom from top

$$\begin{aligned}
 (1-r) \sum_{k=0}^{\infty} r^k &= 1 \\
 \sum_{k=0}^{\infty} r^k &= \frac{1}{1-r}
 \end{aligned}$$

6.4 Growth of Functions

Running time of algorithms is defined by *functions* describing how many steps they take to complete in terms of *input size*.

EXAMPLE ▷ It might take $3n$ steps to add a list of n numbers, or $15n$ steps? or $\frac{1}{100}n^2$ steps?



- We would like to ignore constant factors (since different machines run at different speeds/cycle).
- We would also like to ignore small values of x .
- In fact we want to see which function *eventually* is the largest.

Definition 39 The Big Oh Notation. For functions f and g , we write

$$f(x) = O(g(n))$$

to denote:

$$\exists c, k \geq 0 \text{ such that } \forall x > k \quad f(x) \leq cg(x).$$

EXAMPLE ▷ $3n$ is $O(15n)$ since $\forall n > 0 \quad 3n \leq 1 \times 15n$.

EXAMPLE ▷ $15n$ is $O(3n)$ since $\forall n > 0 \quad 15n \leq 5 \times 3n$.

EXAMPLE ▷ x^2 is $O(x^3)$ since $\forall x > 1 \quad x^2 \leq x^3$.

EXAMPLE ▷ x^2 is $O(\frac{x^2}{1,000,000})$ since $\forall x > 0 \quad x^2 \leq 1,000,000 \frac{x^2}{1,000,000}$.

EXAMPLE \triangleright $x^2 + 100x + 100$ is $O(\frac{1}{100}x^2)$ since $\forall x > 2$,

$$\begin{aligned} 100x^2 + 100x + 100 &\leq 2 \times (100)^2 \times \frac{1}{100}x^2 \\ 100x^2 + 100x + 100 &\leq 200x^2 \\ 100x + 100 &\leq 100x^2 \\ x + 1 &\leq x^2 \end{aligned}$$

The last inequality is not true for $x = 0$ and $x = 1$ but is obviously true for $x \geq 2$.

EXAMPLE \triangleright We want to show that $5x + 100$ is $O(x/2)$. In other words we want to fill in the blanks below

$$\forall x \geq \underline{\quad} : \quad 5x + 100 \leq \underline{\quad} \times \frac{x}{2}.$$

Attempt 1. Let $c = 10$, so that $10 \times \frac{x}{2} = 5x$ and then get rid of the $5x$ term. But we will be left with

$$\forall x \geq \underline{\quad} : \quad 5x + 100 \leq 5x$$

and this obviously won't work.

Insight. We must end up with something else on the right that is larger than 100 (if x is large enough).

Let $c = 11$ (this way we will make an extra $x/2$ on the right).

$$\forall x \geq \underline{\quad} : \quad 5x + 100 \leq 5x + \frac{x}{2}$$

and therefore we need to fill in the blank such that,

$$\forall x \geq \underline{\quad} : \quad 100 \leq \frac{x}{2}$$

and this is easy!

$$\forall x \geq 200 : \quad 100 \leq \frac{x}{2}$$

6.4.1 Guidelines

- In general, only the *largest* term in a sum matters. i.e.,

$$a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n \quad \text{is} \quad O(x^n).$$

See proof of Theorem 3 of page 81 in the text.

- n dominates $\lg n$. So, as an example $n^5 \lg n$ is $O(n^6)$.
- Here is a list of common functions in increasing $O()$ order.

$$1 \quad \lg n \quad n \quad n \lg n \quad n^2 \quad n^3 \quad \dots \quad 2^n \quad n!$$

EXAMPLE $\triangleright \sum_{k=1}^n k = \frac{n(n+1)}{2}$ is $O(n^2)$.

EXAMPLE $\triangleright \sum_{k=1}^n k^2 = 1^2 + 2^2 + \dots + n^2 \leq \overbrace{n^2 + n^2 + \dots + n^2}^n = n^3$. So, $\sum_{k=1}^n k^2$ is $O(n^3)$.

EXAMPLE \triangleright For $n!$ we have

$$\begin{aligned} n! &= n(n-1)(n-2)(n-3)\dots 3.2.1 \\ &\leq \underbrace{n \times n \times n \times n \times \dots \times n}_n \\ &= n^n \end{aligned}$$

So, $n!$ is $O(n^n)$.

EXAMPLE $\triangleright \lg n! \leq \lg n^n \leq n \lg n$. So, $\lg n!$ is $O(n \lg n)$.

We'll see this later.

An important observation is that n^3 is *not* $O(100n^2)$. If this were not true, then for some integers k and c we would have

$$\forall n \geq k : \quad n^3 \leq c \times 100n^2.$$

But this would simplify to

$$\forall n \geq k : \quad n \leq c \times 100$$

which is a contradiction.

Theorem 2 If $f_1 = O(g_1)$ and $f_2 = O(g_2)$, then $f_1(x) + f_2(x) = O(\max\{g_1(x), g_2(x)\})$.

Proof. Let $h(x) = \max\{g_1(x), g_2(x)\}$.

Note that for different values of x , $h(x)$ can be $g_1(x)$ or $g_2(x)$.

Clearly, $f_1 = O(h)$ since $g_1(x) \leq h(x)$ for all x . Therefore,

$$\exists c_1, k_1 : \quad f_1(x) \leq c_1 h(x) \quad \forall x \geq k_1.$$

Similarly $f_2 = O(h)$ and therefore

$$\exists c_2, k_2 : \quad f_2(x) \leq c_2 h(x) \quad \forall x \geq k_2.$$

Let $k = \max\{k_1, k_2\}$ and $c = c_1 + c_2$. Then

$$f_1(x) + f_2(x) \leq c_1 h(x) + c_2 h(x) \leq ch(x) \quad \forall x \geq k.$$

■

Corollary 1 If f_1 is $O(g)$ and f_2 is $O(g)$, then $f_1 + f_2$ is $O(g)$.

A similar theorem can be proved for the product of two functions:

Theorem 3 If $f_1 = O(g_1)$ and $f_2 = O(g_2)$, then $f_1(x) \cdot f_2(x) = O(g_1(x) \cdot g_2(x))$.

Proof. From the assumptions we have

$$\exists c_1, k_1 : \quad f_1(x) \leq c_1 g_1(x) \quad \forall x \geq k_1.$$

And similarly for f_2 we have

$$\exists c_2, k_2 : \quad f_2(x) \leq c_2 g_2(x) \quad \forall x \geq k_2.$$

But then we will have

$$\forall x \geq \max\{k_1, k_2\} : \quad f_1(x) \cdot f_2(x) \leq c_1 \cdot c_2 \cdot g_1(x) \cdot g_2(x).$$

Therefore setting $k = \max\{k_1, k_2\}$ and $c = c_1 \cdot c_2$ we will have

$$\forall x \geq k : \quad f_1(x) \cdot f_2(x) \leq c \cdot g_1(x) \cdot g_2(x)$$

as desired. ■

Definition 40 For functions f and g , we write $f = \Omega(g)$ if $g = O(f)$.

In other words, $f = \Omega(g)$ if

$$\exists k, c : \quad \forall x \geq k \quad g(x) \leq cf(x)$$

or

$$\exists k, c' = \frac{1}{c} \quad \forall x \geq k \quad f(x) \geq c'g(x).$$

Definition 41 For functions f and g , we write $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$ (or equivalently if $f = O(g)$ and $g = O(f)$).

This means that we can “trap” g between some $c'f$ and cf .

Intuitively,

<i>When we write</i>	<i>it is similar to</i>
$f = O(g)$	$f \leq g$
$f = \Omega(g)$	$f \geq g$
$f = \Theta(g)$	$f = g$

6.4.2 Other Big-Oh Type Estimates

Definition 42 For functions f and g , $f = o(g)$ if

$$\forall c > 0 \exists k \quad f(x) \leq cg(x) \quad \text{when } x \geq k.$$

Compare to the definitions of $O(g)$.

In other words, no matter *how small* c is, cg eventually dominates f . $f = o(g)$ can alternatively be defined as

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$$

EXAMPLE \triangleright We want to show that n^2 is $o(n^2 \lg n)$. That will be to show that

$$\forall c > 0 \exists k \quad n^2 \leq cn^2 \lg n \quad \text{if } n \geq k.$$

Proof. Let c be given (think of c as $\frac{1}{1,000,000}$). How large does n have to be to ensure that

$$n^2 \leq cn^2 \lg n$$

or $1 \leq c \lg n$?

The answer is $\lg n \geq \frac{1}{c}$ or $n \geq 2^{\frac{1}{c}}$ ($n \geq 2^{1,000,000}$). Therefore,

$$\forall c > 0 \quad n^2 \leq cn^2 \lg n \quad \text{when } n \geq 2^{\frac{1}{c}}.$$

■

EXAMPLE \triangleright We want to show that $10n^2 = O(n^3)$. Equivalently we want to show that

$$\forall c > 0 \exists k \quad 10n^2 \leq cn^3 \quad \text{when } n \geq k.$$

How large n must be so that $10n^2 \leq cn^3$? Simplifying we get $10 \leq cn$ or $\frac{10}{c} \leq n$? So,

$$\forall c > 0 \exists k = \frac{10}{c} \quad 10n^2 \leq cn^3 \quad \text{when } n \geq k.$$

Definition 43 For functions f and g , we write $f = \omega(g)$ if $g = o(f)$.

Chapter 7

Mathematical Induction

Mathematical Induction is an extremely important technique in proving theorems, especially in discrete mathematics and computer science. It is used often to prove that $\forall n \in \mathbb{N} P(n)$ holds where $P()$ is a predicate.

In general to show that $\forall n \in \mathbb{N} P(n)$ is true, we can either

1. Let $n \in \mathbb{N}$ be arbitrary and then show that $P(n)$ is true, or
2. Show that $P(0) \wedge P(1) \wedge P(2) \wedge \dots$ (*but this would take forever*).

This simultaneously proves $P(n)$ for all $n \in \mathbb{N}$.

Mathematical Induction is a way to show that for each n , we could eventually prove that $P(n)$ is true.

Definition 44 Principle of Mathematical Induction:

Let $P()$ be a predicate defined on \mathbb{N} . Then if

- (1) $P(0)$ is true, *and*
- (2) $\forall n \in \mathbb{N} [P(n) \rightarrow P(n + 1)]$,

then

$\forall n \in \mathbb{N} P(n)$ is true.

7.1 Why it works

Suppose (1) and (2) hold. Then

$P(0)$	By (1)
$P(0) \rightarrow P(1)$	By (2)
$P(1)$	By Modus Ponens
$P(1) \rightarrow P(2)$	By (2)
$P(2)$	By Modus Ponens
$P(2) \rightarrow P(3)$	By (2)
$P(3)$	By Modus Ponens
\vdots	

7.2 Well Ordering Property

Definition 45 A set S is said to have the *well ordering property*, if every non-empty subset of S has a *least* element.

EXAMPLE $\triangleright \mathbb{N}$ is well-ordered.

Fact 13 We take the *intuitive* fact that the natural numbers are well-ordered as a “*given axiom*”.

EXAMPLE \triangleright Are the integers (\mathbb{Z}) well-ordered?

Answer. No, the subset $S = \{\dots, -3, -2, -1, 0\}$ of \mathbb{Z} has no least element.

EXAMPLE \triangleright Are the non-negative reals well-ordered?

Answer. No, the subset $S = \{x \mid x > 1\}$ has no least element.

Fact 14 Any set can be well-ordered using a different definition of “ $<$ ”.

Proof. (of Mathematical Induction)

Proof by contradiction: Assume that

- (1) $P(0)$ is true, and
- (2) $\forall n [p(n) \rightarrow P(n+1)]$,

but NOT $\forall n P(n)$.

Then $\exists n \in \mathbb{N}$ such that $\neg P(n)$. Let

$$S = \{n \in \mathbb{N} : \neg P(n)\}.$$

S is non-empty since by assumption $\exists n \in \mathbb{N} \neg P(n)$. By well ordering property of \mathbb{N} , S has a least element k . This means that

- $P(k)$ is false and $k \neq 0$ because we already know by (1) that $P(0)$ is true, and
- $P(k - 1)$ is true, since k is the least element in S .

But then by (2), $P(k - 1) \rightarrow P(k)$ contradicting $P(k - 1)$ true and $P(k)$ false (Q.E.D).

■ *Homework: Show that Mathematical Induction Principle implies Well Ordering Principle.*

EXAMPLE ▷ **Sum of the first n odd numbers:** What is the sum of the first n odd numbers?

$$1 + 3 + 5 + \cdots + 2n - 1 = ?$$

We first try to *guess* the answer.

$$\begin{aligned} 1 &= 1 \\ 1 + 3 &= 4 \\ 1 + 3 + 5 &= 9 \\ 1 + 3 + 5 + 7 &= 16 \\ 1 + 3 + 5 + 7 + 9 &= 25 \\ &\dots = \dots \\ 1 + 3 + 5 + 7 + 9 + \cdots + 2n - 1 &= n^2?? \end{aligned}$$

Now we prove it by induction. Let

$$P(n) = \text{"The sum of the first } n \text{ odd numbers is } n^2\text{"}$$

We must show that (1) $P(0)$ is true (*base case*), and (2) $\forall n \in \mathbb{N} P(n) \rightarrow P(n+1)$ (*inductive step*).

Base Case.

$$P(0) = \text{"The sum of the first 0 odd numbers is 0."}$$

This is clearly true. We add nothing and get 0.

Inductive Step.

$$\forall n \in \mathbb{N} [P(n) \rightarrow P(n + 1)].$$

To show this, we do a direct proof. We assume $P(n)$ and show that $P(n + 1)$ holds.

Inductive Hypothesis: "The sum of the first n odd numbers is n^2 .", i.e.

$$1 + 3 + 5 + \cdots + 2n - 1 = n^2.$$

To show: “The sum of the first $n + 1$ odd numbers is $(n + 1)^2$.”, i.e.

$$1 + 3 + 5 + \cdots + 2n - 1 + (2n + 1) = (n + 1)^2.$$

$$\begin{aligned} \text{Sum of the first } n + 1 \text{ odds} &= 1 + 3 + 5 + \cdots + 2n - 1 + (2n + 1) \\ &= n^2 + (2n + 1) \quad (\text{Applying the Induction Hypothesis}) \\ &= (n + 1)^2. \end{aligned}$$

Thus, $P(n + 1)$ holds.

Important Notes.

- Book uses the following principle instead:

$$\frac{P(1) \\ \forall n [P(n) \rightarrow P(n + 1)] \\ \therefore \forall n \geq 1 P(n)}$$

i.e. it starts at $P(1)$. So, the text’s base case is $P(1)$ not $P(0)$.

- The following principle is also valid:

$$\frac{P(k) \\ \forall n \geq k [P(n) \rightarrow P(n + 1)] \\ \therefore \forall n \geq k P(n)}$$

EXAMPLE ▷ Prove that $1.1! + 2.2! + 3.3! + \cdots + n.n! = (n + 1)! - 1$.

Proof. *Base Case.* Show the above statement holds when $n = 1$.

$$1.1! = 1 = (1 + 1)! - 1.$$

Inductive Step. Show $\forall n [P(n) \rightarrow P(n + 1)]$ where $P(n)$ is the predicate stating that the above equality holds.

Inductive Hypothesis:

$$1.1! + 2.2! + \cdots + n.n! = (n + 1)! - 1.$$

Now consider

$$\begin{aligned} \underbrace{1.1! + 2.2! + \cdots + n.n!}_{(n+1)!-1 \text{ by induction hypothesis}} + (n + 1).(n + 1)! &= (n + 1)! - 1 + (n + 1).(n + 1)! \\ &= [(n + 1) + 1](n + 1)! - 1 \\ &= (n + 2)(n + 1)! - 1 \\ &= (n + 2)! - 1 \\ &= ((n + 1) + 1)! - 1. \end{aligned}$$

■

EXAMPLE ▷ If S has n elements, then $\mathcal{P}(S)$ has 2^n elements.

Proof. *Base Case.* If S has 0 elements, then $S = \emptyset$ and $\mathcal{P}(S) = \{\emptyset\}$ which has $1 = 2^0$ elements.

Inductive Step. We want to show that

$$\forall n [(\text{If } |S| = n \text{ then } |\mathcal{P}(S)| = 2^n) \rightarrow (\text{If } |S| = n + 1 \text{ then } |\mathcal{P}(S)| = 2^{n+1})].$$

Inductive Hypothesis. Assume that $\mathcal{P}(S)$ has 2^n elements for any n -element set S .

Let S' be any set with $n + 1$ elements. Then $S' = S \cup \{a\}$ for some $a \in S'$ and $S \subseteq S'$ of cardinality n .

Every subset of S' either contains a , or does not. We will count separately those subsets of S' that don't contain a , and those that do contain a .

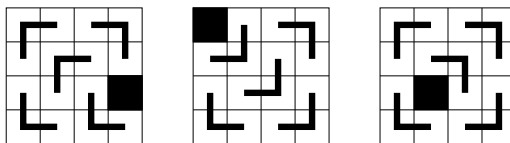
$$\begin{aligned} \text{Number of subsets of } S' \text{ not containing } a &= \text{Number of subsets of } S \\ &= 2^n \quad (\text{by induction hypothesis}) \end{aligned}$$

On the other hand, subsets of S' containing a can all be made by taking some subset of S' not containing a and adding a to it. Therefore, there are as many subsets of S' containing a as there are subsets of S' not containing a . As every subset of S' either contains a or not, and we just saw that the number of subsets of S' not containing a is 2^n , we conclude that the total number of subsets of S' (containing or not containing a) is $2^n + 2^n = 2 \cdot 2^n = 2^{n+1}$. ■

EXAMPLE ▷ Any $2^n \times 2^n$ grid with one cell blacked out can be tiled with L-shaped pieces shown below without overlap.



See the following $2^2 \times 2^2 = 4 \times 4$ grids for example.



Proof. Proof by induction on n . Let $T(n)$ be the predicate: "A $2^n \times 2^n$ missing one cell can be tiled with the L-shaped piece."

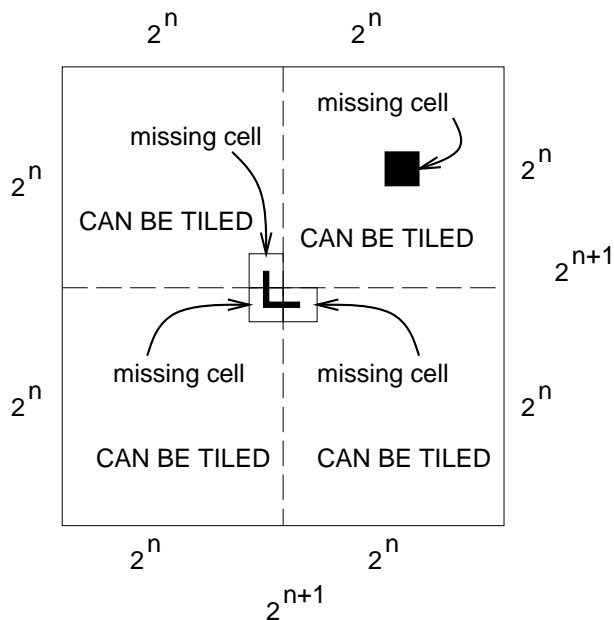
Base Case. $T(0)$: A $2^0 \times 2^0$ (1×1) missing one cell (but the missing cell is the whole grid!) can be tiled with L-shaped piece. This is true because no cell remains to be tiled.



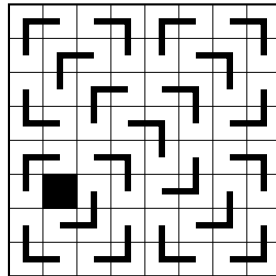
$T(1)$ (more satisfying): A 2×2 missing one cell can be tiled.



Inductive Step. Show that $\forall n [T(n) \rightarrow T(n+1)]$. We assume inductively that a $2^n \times 2^n$ (missing one) can be tiled and show how to tile a $2^{n+1} \times 2^{n+1}$ as in the following figure.



Below is an example of tiling an 8×8 missing one.



7.3 Strong Mathematical Induction

Definition 46 Strong Mathematical Induction Principle

If $P(0)$ and

$$\forall n \geq 0 [(P(0) \wedge P(1) \wedge \dots \wedge P(n)) \rightarrow P(n+1)]$$

then $\forall n P(n)$.

Using the above (stronger) principle, to prove $P(n+1)$ we can rely on inductive hypothesis that *all* of $P(0), P(1), \dots, P(n)$ are true.

The reason this principle works, and the proof that it does (based on well-ordering principle) is *identical* to the *weak* induction.

EXAMPLE \triangleright Given n black points and n white points in the plane, no three of them on a line, prove that there exists a way to *match* them one-to-one, such that the line segments between the matched pairs do not intersect.

Proof. *Base Case.* $n = 1$.



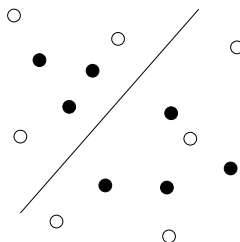
Just join them together; there is only one segment so clearly it doesn't intersect any others.

Inductive Step. Show

$$\forall n \geq 0 [(P(0) \wedge P(1) \wedge \dots \wedge P(n)) \rightarrow P(n+1)].$$

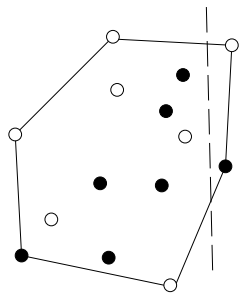
Assume as inductive hypothesis that there exists a matching as stated for any set of $1, 2, \dots, n$ points. We now show that there exists a matching for any set of $n+1$ points.

If there exists a line, separating the group into one smaller group of k black and k white points and a smaller group of $(n + 1) - k$ black and $(n + 1) - k$ white points, then we can apply the inductive hypothesis to each side and note that a pair from one side cannot interfere with a pair from the other side (why not?)



We now show that there is always such a separating line. For this consider the *convex hull* of the points¹. There are two possibilities:

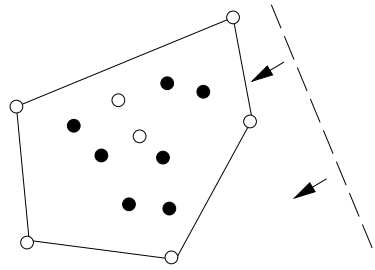
- *Case 1.* There exists a black point and a white point on the convex hull. In this case, there exists an adjacent white and black points on the convex hull (why?) Then, we can separate this pair from the rest of the points and we are done.



- *Case 2.* The convex hull is all of the same color. Then take a line with slope unequal to the slope of any pair of points and *sweep* it across the figure. There must be a place where k white and k black points fall on one side of the sweeping line and $(n + 1) - k$ white and $(n + 1) - k$ black points fall on the other side of it.

This is because if for example the convex hull is all black, then the first point the line passes is black. Also the last point that line passes is black. Somewhere in between this process, the white points must have “caught up” to the black points behind the line.

¹The convex hull of a set of points in plane is the smallest convex polygon containing all of the points (on or inside it).



■.

7.4 Inductive (recursive) Definitions

Similar to proving theorems inductively, we can sometimes define mathematical structures inductively (recursively).

EXAMPLE ▷ The non-recursive definition of factorial is

$$n! = 1.2.3. \dots .n.$$

But we can define factorial in the following (recursive) way as well.

$$n! = \begin{cases} 1 & \text{if } n = 1 \\ n.(n-1)! & \text{if } n > 1 \end{cases}$$

Note that this suggests an obvious recursive algorithm.

An inductive or recursive definition has a *base case* which gives the definition for $n = 0$, and an *induction step* which gives the definition for $n + 1$ in terms of $1, 2, \dots, n$.

EXAMPLE ▷ **Fibonacci numbers.**

$$f(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ f(n-1) + f(n-2) & \text{if } n > 1 \end{cases}$$

The first few numbers in the Fibonacci sequence ($f(0), f(1), f(2), \dots$) are then

$$0, 1, 1, 2, 3, 5, 8, 13, 21, \dots$$

7.4.1 Sets can Be Defined Inductively

EXAMPLE ▷ Let S be the set of numbers divisible by 3: We define S as follows

- (1) $3 \in S$.
- (2) If $x, y \in S$, then $x + y \in S$.
- (3) If $x, y \in S$, then $x - y \in S$.
- (4) No other numbers are in S except as described by rules 1, 2, and 3.

It can be easily seen that $\dots, -9, -6, -3, 0, 3, 6, 9, 12, \dots$ are all in S . In fact, it can be proven by induction that *all* multiples of 3 are in S and that *no* other numbers are in S as follows.

Proof. We want to prove that $S = \{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\}$. For this we will prove two things:

1. $\{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\} \subseteq S$, and
2. $S \subseteq \{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\}$.

Part 1: Proving that $\{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\} \subseteq S$. We show that for all integers k that $3k \in S$. We prove this by induction on $|k|$.

Base Case. $0 \in S$ because

- By (1), $3 \in S$
- Then by (3), $3 - 3 = 0 \in S$.

Inductive Step. Assuming that $3n, -3n \in S$, we show that $3(n+1), -3(n+1) \in S$.

- By inductive assumption $3n \in S$
- By (1), $3 \in S$.
- Then by (2), $3n + 3 = 3(n + 1) \in S$.
- By base case $0 \in S$
- (3), $0 - 3(n + 1) = -3(n + 1) \in S$.

Thus, $\{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\} \subseteq S$.

Part 2: Proving that $S \subseteq \{\dots, -9, -6, -3, 0, 3, 6, 9, \dots\}$. For this part, we observe that by (4), a number is in S only because it is possible to build it using a finite number of applications of rules

1. $3 \in S$
2. $x, y \in S \rightarrow x + y \in S$
3. $x, y \in S \rightarrow x - y \in S$

For example

- To show that $3 \in S$ we need to apply only a single rule (rule (1)).
- To show that $0 \in S$ we need two rule applications $3 \in S$ and $3 - 3 \in S$.
- To show that $9 \in S$ we need three rule applications: rule (1) once and rule (2) twice.

We will prove by induction on the number of rules needed to show that $x \in S$ that x must be of the form $3k$ for some $k \in \mathbb{Z}$.

Base case. If $x \in S$ by one application of the rules, then it must be rule (1) ($3 \in S$). Thus $x = 3$ and clearly $x = 3k$ for $k = 1$.

Induction Step. We assume that the statement we want to prove holds for every x whose membership in X can be shown by at most n application of the given rules.

Let $x \in S$ via $n + 1$ rule applications. Then suppose that the last application was rule (2). Thus $x = a + b$ where a, b are already known to be in S . Then $a, b \in S$ via less than or equal to n rule applications each and so $a = 3k_a$ and $b = 3k_b$ for some k_a and k_b . But then $x = a + b = 3k_a + 3k_b = 3(k_a + k_b)$, a multiple of 3 as desired.

The case where the last rule-application is rule (3) is similar. ■

EXAMPLE \triangleright **Recursive definition of Well-Formed-Formulas.**

- (1) T is a well-formed-formula, F is a well-formed-formula, and p is a well-formed-formula for any propositional variable p .
- (2) If p is a well-formed-formula, then $(\neg p)$ is a well-formed-formula.
- (3) If p and q are well-formed-formulas, then $(p \wedge q)$ and $(p \vee q)$ are well-formed-formulas.

Therefore, using the above definition, to see that $((p \vee q) \wedge (\neg r))$ is a well-formed-formula, we show inductively that $(p \vee q)$ and $(\neg r)$ are well-formed-formulas and apply rule (3).

7.4.2 Strings and Recursive Definitions

Let Σ be a finite set. We will call Σ an *alphabet*. The set of *strings* over Σ denoted by Σ^* is defined inductively as follows

- (1) $\lambda \in \Sigma^*$ where λ denotes the *null* or string.
- (2) If $x \in \Sigma$ and $w \in \Sigma^*$, then $xw \in \Sigma^*$ where xw is the concatenation of strings w with symbol x .

EXAMPLE \triangleright If $\Sigma = \{a, b, c\}$, then

$$\Sigma^* = \{\lambda, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \dots\}.$$

Questions. 1. Does Σ^* have any strings of infinite length?
 2. Is there a bound on how large strings in Σ^* can be? (i.e. Is that a maximum-length string?)

Note that Σ^* is infinite.

EXAMPLE \triangleright **Inductive Definition of Length of Strings.** We denote by $|w|$ the length of the string w .

- (1) $|\lambda| = 0$.
- (2) If $w \in \Sigma^*$ and $x \in \Sigma$ then $|xw| = |w| + 1$.

EXAMPLE \triangleright **Inductive Definition of Reversal of Strings.** By $(w)^R$ we denote the string we get from w if list the symbols of w in the reverse order. For example $(abc)^R = cba$. This is defined inductively as follows.

- (1) $(\lambda)^R = \lambda$.
- (2) If $w \in \Sigma^*$ and $x \in \Sigma$, then $(xw)^R = x(w)^R$.

For example, by the given rules

$$\begin{aligned} (abc)^R &= ((ab)c)^R \\ &= c(ab)^R \\ &= c((a)b)^R \\ &= c(b(a)^R) \\ &= c(b((\lambda)a)^R) \\ &= c(b(a(\lambda)^R)) \\ &= c(b(a(\lambda))) \\ &= cba \end{aligned}$$

Theorem 4 $\forall x, y \in \Sigma^*$

$$(xy)^R = y^R x^R$$

Proof. By induction on $|y|$.

Base Case. $|y| = 0$ then $y = \lambda$. Therefore,

$$(xy)^R = (x\lambda)^R = x^R = \lambda x^R = y^R x^R.$$

Inductive Step. Assume that for every y of length at most n we have $\forall x (xy)^R = y^R x^R$.

Let y be a string of length $n + 1$. Then $y = ua$ for some $u \in \Sigma^*$ and some $a \in \Sigma$ and furthermore $|u| = n$. Then

$$\begin{aligned} (xy)^R &= (x(ua))^R \\ &= ((xy)a)^R \\ &= a(xu)^R && \text{by inductive definition of reverse} \\ &= au^R x^R && \text{by IH since } |u| \leq n \\ &= (ua)^R x^R && \text{by inductive definition of reverse} \\ &= y^R x^R \end{aligned}$$

■

Definition 47 A string w over alphabet Σ is a *palindrome* if

$$w^R = w.$$

EXAMPLE \triangleright Over alphabet $\Sigma = \{0, 1\}$, the string $w = 1011101$ is a palindrome.

EXAMPLE \triangleright **Inductive Definition of Palindromes.** The following two rules inductively define the set of all palindromes S over an alphabet $\Sigma = \{0, 1\}$.

1. $\lambda \in S$, $0 \in S$, and $1 \in S$.
2. If $w \in S$, then so are $0w0$, and $1w1$.

Proof. First we show that if w is a palindrome, then $w \in S$. We use induction on $|w|$ (length of w).

Base Case. For $|w| = 0$ and $|w| = 1$, the only strings are λ , 0 , and 1 which are all in S by rule 1.

Inductive Step. Suppose every palindrome of length less than n is in S and suppose w be a palindrome of length n . Since w is a palindrome, its first and last symbol should be the same. Therefore, either $w = 0w'0$ or $w = 1w'1$ for some palindrome w' of length less than n . Then by induction hypothesis, $w' \in S$ and then by rule 2, $w \in S$. This proves that every palindrome is in S .

Now we want to prove that every string in S is a palindrome. Note that every string in S can be shown to be in S by applying a sequence of applications of rules 1 and 2. We use induction on the number of applications of rules 1 and 2 needed in showing that $w \in S$ to prove that w is a palindrome.

Base Case. The only strings that can be shown to be in S with a single application of rules 1 and 2 are λ , 0, and 1 which are derived using a single application of rule 1 and are all palindromes.

Inductive Step. Suppose that every string derived by fewer than n applications of rules 1 and 2 is a palindrome and let w be a string that happens to be in S by n applications of rules 1 and 2. Clearly, the last rule applied to show that $w \in S$ must have been rule 2 because rule 1 can be applied only once. Therefore, w is obtained by adding 0's or 1's to the beginning or end of a string $w' \in S$ which can be derived in $n - 1$ application of the above rules. By induction hypothesis, w' is a palindrome and since rule 2 adds either 0's or 1's to the beginning and end of the string, w is also a palindrome. ■

Chapter 8

Running Time of Algorithms

Definition 48 An *algorithm* is a clearly defined, step-by-step procedure for solving a problem. An algorithm has the following properties:

We will use pseudo-codes for describing algorithms.

- The input and output are from specified domain.
- Each step is precisely described.
- Each step is executable in finite time.
- Is applicable to a *large* range of inputs, not just a particular one.
- Must halt with output in finite time.

Generality

Halting: This does not follow from the finiteness of each step.

Note. It is a very difficult (in fact *unsolvable*) problem to determine in general if the last property, i.e. determining given an algorithm and an input to the algorithm, whether the algorithm halts on its input.

EXAMPLE \triangleright Nobody knows if the algorithm below halts for every input. It is known that for input x up to billions, it halts.

```

algorithm ALG
input:  $x$ , a positive integer
output:  $y$ , a positive integer

1  begin
2       $y := 0$ ;
3      if  $x = 1$  then
4          halt and output  $y$ 
5      else if  $x$  is even then
6           $x := \frac{x}{2}$ ;
7           $y := y + 1$ ;
8      else if  $x$  is odd then
9           $x := 3x + 1$ 
10          $y := y + 1$ 
11     goto 6;
12  end

```

Note that y , the output of the algorithm, is the number of steps it takes for x to become 1. The following table traces the values x takes before becoming 1.

initial x	subsequent values
5	16, 8, 4, 2, 1
3	3, 10, 5, 16, 8, 4, 2, 1
6	6, 3, ...
7	7, 22, 11, 34, 17, 52, 26, 13

8.1 Iterative Algorithms

The name *iterative algorithm* is applied to any algorithm that repeats the same sequence of steps any number of times. Typical programming constructs for iterations include

- **for** loops
- **while** loops
- **repeat** loops

8.1.1 Complexity (Running Time)

To compute the complexity of an iterative algorithm, we usually determine the execution times of the loops of the algorithm. For nested loops, we multiply the times the loops take.

EXAMPLE ▷ In this code

```
1   for i := 1 to n
2       for j := 1 to m
3           Twiddle-Thumbs
```

the inner loop (2) executes “Twiddle-Thumbs” m times, one time for each value of j . The outer loop (1) executes the inner loop n times (one time for each value of i). The total number of execution of “Twiddle-Thumbs” is therefore mn times.

8.1.2 Correctness of Iterative Algorithms

- Is usually proved by induction.
- The induction is usually on the number of iterations of loops.
- Typically, we find a *property* that asserts *progress* is made. Then we show that
 1. this property holds for only one execution of the loop (Base Case) , and
 2. assuming the property holds after k iterations, it then holds after $k + 1$ iterations (Inductive Step).

EXAMPLE ▷ The following algorithm finds the largest element of a list.

```

algorithm LARGEST
input:  $x_1, \dots, x_n$ , a list (array) of positive integer
output:  $x_n$ , when done will contain the largest

1  begin
2      for  $j := 1$  to  $n - 1$ 
3          if  $x_j > x_{j+1}$  then [swap  $x_j$  and  $x_{j+1}$ ]
4               $temp := x_j$ ;
5               $x_j := x_{j+1}$ ;
6               $x_{j+1} := temp$ ;
7  end

```

Below is an example of the execution of the above algorithm with 4 inputs. The two \triangleleft 's show the elements that are compared at each step.

variable	initial value	$j = 1$	$j = 2$	$j = 3$	after halting
x_1	3	3 \triangleleft	2	2	2
x_2	2	2 \triangleleft	3 \triangleleft	3	3
x_3	4	4	4 \triangleleft	4 \triangleleft	1
x_4	1	1	1	1 \triangleleft	4

If we assume that

1. each comparison takes constant time c ,
2. each addition takes constant time s , and
3. each assignment takes constant time a ,

then the lines 3, 4, 5, and 6 of the above algorithm, take times $c + s$, a , $a + s$ and a respectively. Adding these up, each execution of the loop of line (2) needs time equal to $\alpha = c + 3a + 3s$. Since the loop is executed $n - 1$ times, the total running time of the above algorithm will be $(c + 3a + 3s)(n - 1) = \alpha(n - 1) = O(n)$.

Note that here, the constant α depends only on the machine that runs the algorithm and not n .

Now we give the proof of the correctness:

Proof. *Assertion:* After the j -th execution of the loop,

$$x_{j+1} = \max\{x_1, x_2, \dots, x_j, x_{j+1}\}.$$

We prove the above claim by induction on j . Note that for $j = n - 1$, the above assertion states that the algorithm has successfully found the largest element of the input.

Base Case. $j = 0$, i.e. no iterations of the loop yet. Certainly $x_1 = \max\{x_1\}$.

Inductive Step. Assume that after j -th iteration,

$$x_{j+1} = \max\{x_1, x_2, \dots, x_j, x_{j+1}\}.$$

Now on $(j + 1)$ -th execution of the loop, x_{j+1} is compared with x_{j+2} and the larger is put in x_{j+2} . Thus,

$$\max\{x_{j+1}, x_{j+2}\} = \max\{x_1, x_2, \dots, x_{j+1}, x_{j+2}\} = x_{j+2}.$$

■

EXAMPLE ▷ **Bubble Sort.** Now consider the following algorithm that sorts its input list of integers in increasing order.

EXAMPLE ▷ The following algorithm finds the largest element of a list.

```

algorithm BUBBLE SORT
input:  $x_1, \dots, x_n$ , a list (array) of positive integer
output: input list sorted

1   begin
2       for  $j := n$  down to 2
3           LARGEST( $x_1, x_2, \dots, x_j$ )
4   end

```

Below is an example of the execution of the above algorithm with 4 inputs. The two \triangleleft 's show the elements that are compared at each step.

variable	initial value	$j = 4$			$j = 3$		$j = 1$	after halting
x_1	3	3 \triangleleft 2	2	2	2 \triangleleft 2	2	2 \triangleleft	1
x_2	2	2 \triangleleft 3	3	3	3 \triangleleft 1	1	1 \triangleleft	2
x_3	4	4	4 \triangleleft 4	4	1	3	3	3
x_4	1	1	1	1 \triangleleft 4	4	4	4	4
		L(x_1, x_2, x_3, x_4)			L(x_1, x_2, x_3)		L(x_1, x_2)	

We now prove the correctness of the given algorithm.

Proof. *Assertion:* "For $k = n, n - 1, \dots, 2$, after the iteration where $j = k$, the largest $n - k + 1$ numbers are sorted in positions x_n, x_{n-1}, \dots, x_k ."

Base Case. $k = n$: After 1st iteration, the largest number is in position x_n , since LARGEST is correct.

Inductive Step. Assume after iteration with $j = k$, the largest $n - k + 1$ numbers are sorted in order in x_n, \dots, x_k .

Now the iteration with $j = k - 1$ calls LARGEST(x_1, \dots, x_{k-1}). So (by correctness of largest), the next largest is put into x_{k-1} . Thus after this iteration, the largest $n - (k - 1) + 1$ numbers are in positions $x_n, x_{n-1}, \dots, x_k, x_{k-1}$. ■

8.2 Complexity

Often the running time of an algorithm depends not only on the number of inputs (as with Bubble Sort), but also on what the particular inputs are (not like Bubble sort).

Two common ways to analyze such algorithms are

1. *Worst case complexity*: Analyze what worst possible input could be.
2. *Average case complexity*: Consider the *typical* input.

EXAMPLE ▷ Linear Search Algorithm.

The following algorithm finds the largest element of a list.

```

algorithm LINEAR SEARCH
input: Sorted list  $a_1 < a_2 < \dots < a_n$  of distinct integers and number  $x$ 
output: Position  $i$  that  $a_i = x$ , if  $x$  is in the list.

1  begin
2       $i := 1$ 
3      while ( $i \leq n$  and  $x \neq a_i$ )
4           $i := i + 1$ 
5      if  $i = n + 1$  then
6          output " $x$  not in list"
7      else
8          output " $x$  is in position  $i$ "
9  end

```

Complexity (counting only comparisons with a'_i 's): there will be i comparisons only if $x = a_i$. However, if $x = a_n$ or x is not in list at all, n comparisons will be made.

8.2.1 Average Case Complexity

EXAMPLE ▷ `LINEAR SEARCH`(a_1, \dots, a_n) takes at the worst case n comparisons.

For the “average case”

- Assume x is in list (a_1, \dots, a_n) and x be equally likely to be in any position.
- If x is in position i , then i comparisons are used.
- Average over $i = 1, 2, \dots, n$ is $(1 + 2 + \dots + n)/n$ comparisons.

$$\frac{1 + 2 + \dots + n}{n} = \frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2}.$$

For other problems, the assumption that all inputs are equally likely may be inappropriate or even meaningless.

8.2.2 Binary Search

Consider the following *math version* of the “20 questions” game: I pick a number x between 1 and 2^{20} . You get 20 questions to find x . Each question has a “yes-no” answer.

EXAMPLE ▷ Find $x \in \{1, 2, \dots, 64\}$ (suppose the answer is 40).

Question 1: Is $x \leq 32$? — Answer: no

Question 2: Is $x \leq 48$? — Answer: yes

Question 3: Is $x \leq 40$? — Answer: yes

Question 4: Is $x \leq 36$? — Answer: no

Question 5: Is $x \leq 38$? — Answer: no

Question 6: Is $x \leq 39$? — Answer: no

Aha! then it is 40!

- Each comparison eliminates $\frac{1}{2}$ of the remaining possibilities.

- The number of comparisons needed is the number of times we need to halve 64 until it becomes 1.

$$\left(\frac{1}{2}\right)^x \times 64 = 1 \quad \Rightarrow \quad 64 = 2^x \quad \Rightarrow \quad x = \log_2 64 = 6$$

- If we are search among n numbers (n a power of 2), with the same calculation as above we need $\lg n$ comparisons.

```

algorithm BINARY SEARCH
input: Sorted list  $a_1 < a_2 < \dots < a_n$  of distinct integers and number  $x$ 
output: Position  $i$  that  $a_i = x$ , if  $x$  is in the list.

1  begin
2       $i := 1$ 
3       $j := n$ 
4      while  $i < j$ 
5           $m := \lfloor \frac{i+j}{2} \rfloor$  [middle element of the current range]
6          if  $x > a_m$  then
7               $i := m + 1$ 
8          else
9               $j := m$ 
10         if  $x = a_i$  then
11             output " $x$  is in position  $i$ "
12         else
13             output " $x$  not in list"
14     end

```

The Complexity

- If n is a power of two (say $n = 2^k$) then at the beginning $i = 1$ and $j = 2^k$ which will make m at line 5 to become $\lfloor (i + j)/2 \rfloor = 2^{k-1}$.

The number of elements left after s iterations of the while loop (i.e., after 2^k has been “halved” s times) is 2^{k-s} . Thus after $k = \lg n$ iterations, there are $2^{k-k} = 2^0 = 1$ elements will be left. Thus the complexity of the algorithm is $O(\lg n)$ since constant time is done within the “while” loop.

- If n is *not* a power of two, let k be the unique number such that $2^k < n \leq 2^{k+1}$. Pretend there are 2^{k+1} of a_i 's. If a comparison about a_i

with $i > n$ is made, answer “ $x < a_i$ ”. Then the number of iterations is $k + 1 \leq \lg n + 1 = O(\lg n)$.

8.3 The Complexity of an Algorithm

To talk about time complexity of algorithms we need to have a clear understanding of the notion of *input size*. Many algorithms are designed to answer questions on inputs that are not necessarily in form of arrays, lists, or strings. The algorithm designer *encodes* these inputs in the form of data structures that computer programs can understand. Therefore, the input size in fact depends on how we encode the input.

As a simple example suppose the input to the algorithm be a number. We can represent a number in decimal, binary or even unary. For example, number 1024 has binary encoding 1000000000 or unary encoding

$$\underbrace{1111 \dots 111}_{1024 \text{ times}}.$$

When we talk about the size of input, we consider a reasonably efficient encoding of the input.

We will denote by $\text{worst}_A(n)$ the worst case time complexity of an algorithm A on an input of size n .

EXAMPLE $\triangleright \text{worst}_{\text{LINEAR SEARCH}}(n) = O(n)$.

EXAMPLE $\triangleright \text{worst}_{\text{BINARY SEARCH}}(n) = O(\lg n)$.

The following table compare the amount of time that an algorithm with time complexity $f(n)$ takes to complete if the computer that executes the algorithm runs at 1,000,000 steps per second.

Time Complexity	Input Size n					
	10	20	30	40	50	60
n	.00001s	.00002s	.00003s	.00004s	.00005s	.00006s
n^2	.0001s	.0004s	.0009s	.0016s	.0025s	.0036s
n^5	.1s	3.2s	24.3s	1.7m	5.2m	13m
3^n	.059s	58m	6.5y	3855c	$2 \times 10^8 c$	$1.3 \times 10^{13} c$

s: seconds, *m*: minutes, *y*: years, *c*: centuries

One might ask the following questions: *The computers are getting faster.*

How much bigger of a problem can we solve if computers get faster?

The following table gives us some clue.

Time Complexity	size of largest problem solvable in 1 hour using	
	today's technology	100 × today's technology
n	N_1	$100 \times N_1$
n^2	N_2	$10 \times N_2$
n^5	N_3	$2.5 \times N_3$
3^n	N_4	$N_4 + 4.19$

8.3.1 Why Worst Case?

Advantages:

- Easy to analyze
- Gives guarantee about the running time, regardless of what input might be encountered
- Don't have to assume or decide what typical inputs are

Disadvantages:

- Bizarre inputs could cause worst case performance to look awful, when in fact most "typical" inputs cause the algorithm to terminate quickly.

8.4 Recursive Algorithms

Definition 49 A *recursive algorithm* is one which calls *itself* from within its own body.

A recursive algorithm usually breaks the problem into one or more related sub-problems, solves (recursively) the sub-problems, and obtains the final solution by some simple computation using the solution(s) to the sub-problem(s). Such algorithms are sometimes called "*Divide And Conquer*" algorithms.

These algorithms must satisfy two requirements:

1. The subproblems should be easier (smaller) than the original problem.
2. After a finite number of sub-divisions, we should reach a sub-problem that can be solved outright (terminating case).

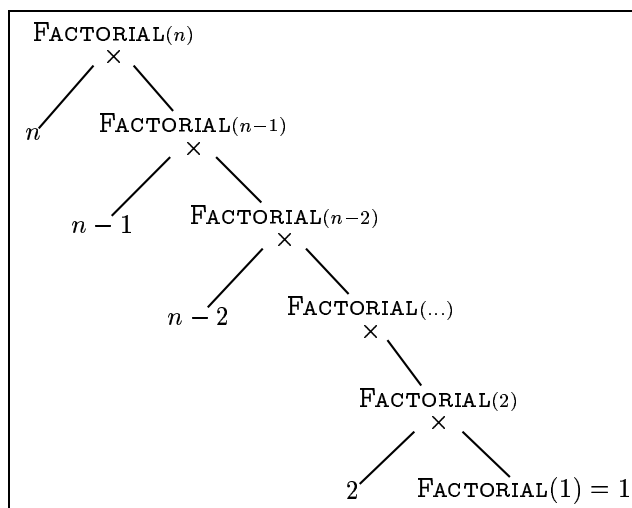
EXAMPLE ▷ Calculating Factorial

```

algorithm FACTORIAL
input: integer  $n$ 
output:  $n!$ 

1  begin
2      if  $n = 1$  then
3          output "1"
4      else
5          output " $n \times \text{FACTORIAL}(n - 1)$ "
6  end

```



Below we give a proof of correctness for the given algorithm.

Proof. By induction on n we show that $\text{FACTORIAL}(n) = n!$

Base Case. Clearly on input 1, FACTORIAL outputs $1 = 1!$.

Inductive Step. We show that if the output of the algorithm for $n - 1$ is $(n - 1)!$, then its output for n is $n!$.

Assume inductively that $\text{FACTORIAL}(n - 1) = (n - 1)!$. Then $\text{Factorial}(n)$ outputs $n \times \text{FACTORIAL}(n - 1) = n \times (n - 1)! = n!$. ■

The correctness of a recursive algorithm is often easily proved by induction, since the algorithm itself has

- A base case (non-recursive) showing what to do if input is 1.
- An inductive (recursive) step, showing how to solve current problem *assuming* that the recursive calls have correctly returned the solutions to the subproblems. This of course looks like inductive step in a proof using induction.

EXAMPLE ▸ **Merge-Sort.** Recursive algorithm for sorting n numbers.

We first write a *subroutine* to merge two sorted list into one.

```

algorithm MERGE
input: Sorted lists  $a_1, \dots, a_k, b_1, \dots, b_l$ 
output: Sorted list combining the inputs lists into one

1  begin
2       $i := 1$ 
3       $j := 1$ 
4      while  $i + j < k + l$ 
5          if  $i = k$  then
6              append  $b_j$  to the output list;  $j := j + 1$ 
7          else if  $j = l$  then
8              append  $a_i$  to the output list;  $i := i + 1$ 
9          else if  $a_i < b_j$  then
10             append  $a_i$  to the output list;  $i := i + 1$ 
11         else
12             append  $b_j$  to the output list;  $j := j + 1$ 
13     end

```

The following table shows how MERGE works for input lists 1, 2, 5, 8 and 3, 4, 6, 9, 12.

output list	a_1, \dots, a_k	b_1, \dots, b_l
1	1 < 2 5 8	3 < 4 6 9 12
1, 2	2 < 5 8	3 < 4 6 9 12
1, 2, 3	5 < 8	3 < 4 6 9 12
1, 2, 3, 4	5 < 8	4 < 6 9 12
1, 2, 3, 4, 5	5 < 8	6 < 9 12
1, 2, 3, 4, 5, 6	8 <	6 < 9 12
1, 2, 3, 4, 5, 6, 8	8 <	9 < 12
1, 2, 3, 4, 5, 6, 8, 9		9 < 12
1, 2, 3, 4, 5, 6, 8, 9, 12		12 <

MERGE-SORT is a divide-and-conquer sorting algorithm. On input x_1, \dots, x_n , we

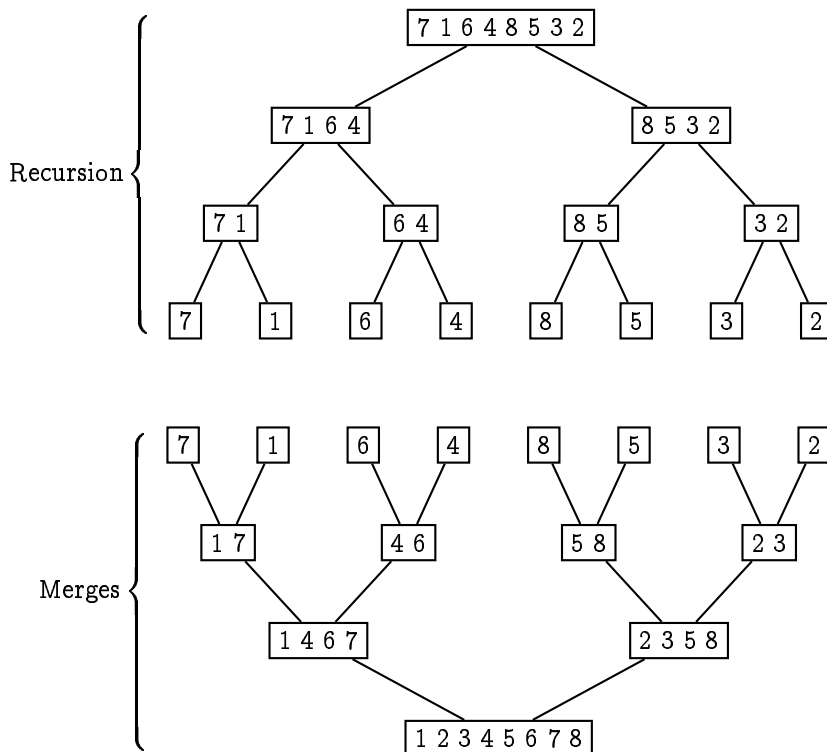
- divide x_1, \dots, x_n into two groups,
- recursively MERGE-SORT the two separate groups, and
- call MERGE to merge the two sorted lists.

```

algorithm MERGE-SORT
input: sub-list  $x_l, \dots, x_h$ 
output: Sorted sub-list

1  begin
2      if  $l \geq h$  then
3          return
4      else
5          MERGE-SORT( $x_l, \dots, x_{\lfloor \frac{l+h}{2} \rfloor}$ )
6          MERGE-SORT( $x_{\lceil \frac{l+h}{2} \rceil}, \dots, x_h$ )
7          MERGE( $(x_l, \dots, x_{\lfloor \frac{l+h}{2} \rfloor}), (x_{\lceil \frac{l+h}{2} \rceil}, \dots, x_h)$ )
8  end

```



Correctness of MERGE-SORT

Lemma 1 If MERGE is given two sorted lists, then it correctly returns a single sorted list containing elements of both lists.

This should be intuitive.

The proof for the above lemma is omitted.

Now we prove the correctness of MERGE-SORT

Proof. We use induction on n .

Base Case. $n = 1$. If there is only one element to be sorted, then clearly the single element returned is in sorted order.

Inductive Assumption. For all $n \leq k$, MERGE-SORT correctly sorts n numbers.

Inductive Step. To sort $k + 1$ numbers, MERGE-SORT divided into lists L_1 and L_2 , each with approximately $\lfloor \frac{k+1}{2} \rfloor \leq k$ numbers.

By inductive assumption, the recursive calls to `MERGE-SORT` return L_1 and L_2 in sorted order.

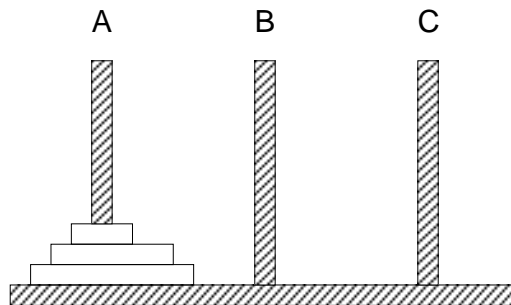
By the above lemma, `MERGE` correctly combines these into a single, sorted lists, which is then output. ■

EXAMPLE ▷ **The towers of Hanoi.**

Rules:

1. Move only one disc at a time.
2. Can only put smaller on larger.

Goal: Transfer all n from A to B .



In the following algorithm `MOVE(n, A, B, C)` corresponds to move n discs from A to B using C as spare.

```

algorithm MOVE( $n, A, B, C$ )
1  begin
2      if  $n = 1$  then
3          transfer the disc from  $A$  to  $B$ 
4      MOVE( $n - 1, A, C, B$ )
5      transfer  $n$ -th disc to  $B$ 
6      MOVE( $n - 1, C, B, A$ )
7  end

```

We now prove the correctness of the above algorithm.

Proof. Proof by induction on n .

Base Case. IF $n = 1$, it just moves the ring.

Inductive Hypothesis. Assume that $\text{MOVE}(k, X, Y, Z)$ correctly moves k discs from X to Y using Z as a spare.

Inductive Step. $\text{MOVE}(k + 1, A, B, C)$ calls $\text{MOVE}(k, A, B, C)$. By inductive hypothesis, we have largest disc left on A , and the stack of k on pole C . Now the largest disc is moved to B . Then $\text{MOVE}(k, C, B, A)$ is called and by inductive hypothesis, the stack of k on C are moved onto B . ■

8.4.1 Complexity of Recursive Algorithms

We usually use *recurrence equations* or *recurrence relations* to find the complexity of recursive algorithm.

We won't show you how to solve these in this class — take CS 273, 373,

To find a recurrence relation, we set up equations that express the complexity of a recursive algorithm.

- First, we write down boundary value(s): The running time for base case $n = 1$.
- Next, we express the time $T(n)$ to solve a problem of size n in terms of values $T(k)$ where $k < n$.
- Finally, we can guess the solution, and prove it correct via induction. Otherwise, solve with techniques from CS 273, 373,

EXAMPLE ▷ Let $T(n)$ be the time to do binary search on n elements (in sorted order).

$$\begin{aligned} T(1) &= 1 \\ T(n) &= c + T(n/2) \end{aligned}$$

where c is the time to determine which half x is in (some small constant) and $T(n/2)$ is then time to search a sorted list of half size.

$$\begin{array}{rcl}
 T(n) & = & c + \overbrace{T(n/2)} \\
 & = & c + c + \overbrace{T(n/4)} \\
 & = & c + c + c + T(n/8) \\
 & \vdots & \vdots \\
 & = & c + c + c + \cdots + T(1)
 \end{array}
 \left. \vphantom{\begin{array}{rcl} T(n) \\ & = & c + c + c + T(n/8) \\ & \vdots & \vdots \\ & = & c + c + c + \cdots + T(1) \end{array}} \right\} \begin{array}{l} \text{depth} \\ \text{of} \\ \text{recursion} \\ = \lg n \end{array}$$

Therefore, total time is $c(\lg n) = O(\lg n)$.

EXAMPLE ▷ Let $M(n)$ be the time to run MERGE-SORT on n numbers.

$$\begin{aligned}
 M(1) &= 1 \\
 M(n) &= M(n/2) + M(n/2) + cn
 \end{aligned}$$

where each $M(n/2)$ is the time to sort half of the array and cn is the time to merge the sorted half-arrays.

Again, intuitively this recurrence relation can be solved as follows.

$$\begin{array}{rcl}
 M(n) & = & 2(M(n/2)) + cn \\
 & = & 2(2(M(n/4)) + c(n/2)) + cn \\
 & = & 4M(n/4) + cn + cn \\
 & = & 4(2(M(n/8) + c(n/4)) + cn + cn \\
 & = & 8M(n/8) + cn + cn + cn \\
 & \vdots & \\
 & = & nM(n/n) + cn + cn + cn + \cdots + cn
 \end{array}
 \left. \vphantom{\begin{array}{rcl} M(n) \\ & = & 2(2(M(n/4)) + c(n/2)) + cn \\ & = & 4M(n/4) + cn + cn \\ & = & 4(2(M(n/8) + c(n/4)) + cn + cn \\ & = & 8M(n/8) + cn + cn + cn \\ & \vdots & \\ & = & nM(n/n) + cn + cn + cn + \cdots + cn \end{array}} \right\} \begin{array}{l} \text{depth} \\ \text{of} \\ \text{recursion} \\ = \lg n \end{array}$$

Therefore, the total time is $n + cn(\lg n) = O(n \lg n)$

EXAMPLE ▷ Let $H(n)$ be the number of moves used by the towers of Hanoi MOVE algorithm on n discs.

$$\begin{aligned}
 H(1) &= 1 \\
 H(n) &= H(n-1) + 1 + H(n-1) \\
 &= 2H(n-1) + 1
 \end{aligned}$$

Again by expanding we can intuitively find the answer to the recurrence relation

above.

$$\begin{array}{r}
 H(n) = 2(H(n-1)) + 1 \\
 = 2(2H(n-2) + 1) + 1 \\
 = 4H(n-2) + 2 + 1 \\
 = 4(2H(n-3) + 1) + 2 + 1 \\
 = 8H(n-3) + 4 + 2 + 1 \\
 \vdots \\
 = 2^{n-1}(H(1)) + 2^{n-2} + \cdots + 4 + 2 + 1 \\
 = \sum_{i=0}^{n-1} 2^i \\
 = 2^n - 1
 \end{array}
 \left. \vphantom{\begin{array}{r} H(n) \\ = \\ = \\ = \\ = \\ = \\ \vdots \\ = \\ = \\ = \end{array}} \right\} \begin{array}{l} \text{depth} \\ \text{of} \\ \text{recursion} \\ = n \end{array}$$

The last equality is just the sum of a geometric series.

8.5 Typical Divide & Conquer Recurrences — Cheat Sheet

Fact 15 If you come up with the recurrence relation consisting of a base case and the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d$$

then the solution to this recurrence relations is

$$T(n) = \begin{cases} O(n^d) & \text{if } a < b^d \\ O(n^d \lg n) & \text{if } a = b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Note that the above recurrence relation is usually obtained whenever solving the problem of size n can be done by

- breaking the problem into a sub-problems,
- with each sub-problem of size $\frac{n}{b}$,
- and we need cn^d (typically $d = 1$) amount of work to put the results of the sub-problems into a solution for the original problem (of size n).

8.5. TYPICAL DIVIDE & CONQUER RECURRENCES — CHEAT SHEET103

EXAMPLE ▷ MERGE-SORT: $T(n) = 2T(n/2) + cn$. Using the above fact, we obtain $T(n) = O(n \lg n)$.

EXAMPLE ▷ BINARY-SEARCH: $T(n) = T(n/2) + cn^0$. Therefore, $T(n) = O(n^0 \lg n) = O(\lg n)$.

Chapter 9

Relations

Recall that for sets A and B , the Cartesian product of A and B is

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

Definition 50 A *relation* R from A to B is a subset $R \subseteq A \times B$.

Note that by the above definition, a relation is a set of ordered pairs.

EXAMPLE \triangleright If A is the set of students and B is the set of courses, we can define the relation R from A to B be

$$R = \{(a, b) : \text{student } a \text{ taking course } b\}.$$

EXAMPLE \triangleright Let $A = B$ be the set of people. Then a relation R from A to B can be

$$R = \{(a, b) : a \text{ and } b \text{ have the same mother}\}.$$

Note. Whenever we define a relation R from a set A to the same set A , we simply say that R is a *relation on* A .

EXAMPLE \triangleright Any function $f : A \rightarrow B$ is a relation R defined as

$$R = \{(a, b) : f(a) = b\}.$$

Since f is a function, we know that this relation has the property that

$$\forall a \in A \exists! b (a, b) \in R$$

9.0.1 Properties of Relations on Sets

Definition 51 A relation R on set A is *reflexive* if

$$\forall a \in A \quad (a, a) \in R.$$

EXAMPLE \triangleright If $A = \{1, 2, 3, 4\}$, the relation $R = \{(1, 2), (3, 4), (1, 1), (3, 3)\}$ on A is not reflexive but the relation $R = \{(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (3, 4)\}$ is reflexive.

EXAMPLE \triangleright Let A be the set of people. The the relation R defined on A as

$$R = \{(a, b) : a \text{ and } b \text{ have the same mother } \}$$

is reflexive because everybody has the same mother as himself!

Definition 52 A relation R on A is *symmetric* if

$$\forall a, b \in A \quad (a, b) \in R \rightarrow (b, a) \in R.$$

EXAMPLE \triangleright If A is the set of people, then $R = \{(a, b) : a \text{ and } b \text{ have the same mother } \}$ is symmetric.

EXAMPLE \triangleright On $A = \{1, 2\}$, the relation \emptyset is symmetric because for all pairs (a, b) :

$$(a, b) \in \text{emptyset} \rightarrow (b, a) \in \text{emptyset}$$

EXAMPLE \triangleright The relation $R = \{(a, b) : a|b\}$ is not symmetric on \times .

Definition 53 A relation R on A is *antisymmetric* if

$$\forall a, b \in A \quad [(a, b) \in R \wedge (b, a) \in R] \rightarrow a = b.$$

EXAMPLE \triangleright On $A = \mathbb{N}$ the “divides” relation $R = \{(a, b) : a|b\}$ is antisymmetric because if for integers a and b , $a|b$ and $b|a$ we can conclude that $a = b$.

EXAMPLE \triangleright “divides” is not antisymmetric on \mathbb{Z} because $-1|a$ and $1|-1$ but $1 \neq -1$.

EXAMPLE \triangleright $A = \{1, 2, 3\}$ and $R = \{(1, 1), (2, 2), (3, 3)\}$ is *both* symmetric and antisymmetric.

Definition 54 A relation R on A is *transitive* if

$$\forall a, b, c \in A \quad [(a, b) \in R \wedge (b, c) \in R] \rightarrow (a, c) \in R.$$

EXAMPLE ▷ If A is the set of people and $R = \{(a, b) : a \text{ and } b \text{ are sisters}\}$, then clearly if a is a sister of b and b is a sister of c , then a is a sister of c and therefore R is transitive but R is none of reflexive, symmetric, or antisymmetric (why?) .

EXAMPLE ▷ $A = \{1, 2, 3\}$ and $R = \{(1, 2), (3, 1)\}$. Then R is not transitive because otherwise $(3, 2)$ should be in.

EXAMPLE ▷ $A = \{1, 2, 3\}$ and $R = \{(1, 2), (2, 1)\}$. Then R is not transitive because otherwise $(1, 1)$ and $(2, 2)$ should be in.

EXAMPLE ▷ $A = \{1, 2, 3\}$ and $R = \{(1, 2)\}$ is transitive (vacuously).

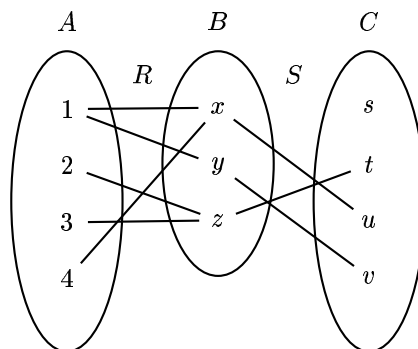
9.0.2 Combining Relations

Let R_1 and R_2 be two relations. Since relations are sets, $R_1 \cup R_2$, $R_1 \cap R_2$, $R_1 - R_2$, $R_2 - R_1$, and \bar{R}_1 are all subsets of $A \times B$ and hence relations.

Definition 55 Let R be a relation from A to B (i.e, $R \subseteq A \times B$) and S be a relation from B to C (i.e, $S \subseteq B \times C$). Then the *composition* of R and S denoted by $R \circ S$ and defined as

$$S \circ R = \{(a, c) : \exists B \quad (a, b) \in R \text{ and } (b, c) \in S\}$$

EXAMPLE ▷ In the following figure, lines between elements of A and B show the pairs in the relation R and those between B and C show the pairs in the relation S .



Then $S \circ R = \{(1, u), (1, v), (2, t), (3, t), (4, u)\}$.

Definition 56 If R is a relation on A , then (inductively define)

$$\begin{aligned} R^1 &= R \\ R^{n+1} &= R^n \circ R \end{aligned}$$

Theorem 5 If R is transitive, then

$$\forall n \quad R^n \subseteq R$$

Proof. By induction on n .

Base Case. $n = 1$: clearly $R^1 = R \subseteq R$.

Inductive Step. Assume as inductive hypothesis that $R^n \subseteq R$ show that $R^{n+1} \subseteq R$.

Let $(a, b) \in R^{n+1} = R^n \circ R$. By definition of composition, there exists an x in the set A on which R is defined such that

$$(a, x) \in R \quad \text{and} \quad (x, b) \in R^n.$$

By inductive hypothesis $R^n \subseteq R$ and therefore $(x, b) \in R$ also. Thus $(a, x) \in R$ and $(x, b) \in R$. By transitivity of R , $(a, b) \in R$ and thus $R^{n+1} \subseteq R$. ■

9.1 Representing Relations

9.1.1 As Matrices

We can represent a relation from $A = \{a_1, \dots, a_n\}$ to $B = \{b_1, \dots, b_n\}$ by an $n \times m$ matrix $M = [m_{ij}]$ in which $m_{ij} = 1$ if $(a_i, b_j) \in R$ and 0 otherwise.

EXAMPLE ▷ The following matrix represent the relation

$$R = \{(a_1, b_1), (a_1, b_3), (a_1, b_4), (a_1, b_6), (a_2, b_1), \dots, (a_5, b_7)\}$$

from $A = \{a_1, \dots, a_5\}$ to $B = \{b_1, \dots, b_7\}$.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7
a_1	1	0	1	1	0	1	0
a_2	1	1	1	0	1	0	1
a_3	1	0	1	0	0	1	0
a_4	1	1	1	1	0	0	0
a_5	0	0	1	0	1	1	1

The value 1 in the boxed cell in row 4 and column 3 indicates that $(a_4, b_3) \in R$ and the value 0 in the boxed cell in row 5 and column 4 indicates that $(a_5, b_4) \notin R$.

Let R be a relation on A and M_R be the matrix representation of R . Then

R is reflexive iff the diagonal of M_R is all 1's,

and

R is symmetric iff M_R is visually symmetric across the diagonal.

If R_1 and R_2 are relations on A with matrices M_{R_1} and M_{R_2} , then

- $R_1 \cup R_2$ has matrix $M_{R_1} \vee M_{R_2}$ (obtained by “or”ing the corresponding bits).
- $R_1 \cap R_2$ has matrix $M_{R_1} \wedge M_{R_2}$ (obtained by “and”ing the corresponding bits).
- $R_2 \circ R_1$ has matrix $M_{R_1} \odot M_{R_2}$ (boolean product of the two matrices as defined below).

Definition 57 The *boolean matrix multiplication* of boolean matrices M_1 and M_2 is the same as their product except that “+” is interpreted as “ \vee ” and “ \cdot ” is interpreted as “ \wedge ”.

EXAMPLE \triangleright Check the multiplication below:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

To see why the matrix representation of $S \circ R$ is $M_R \odot M_S$ Let $M_R = [r_{ij}]$ be the matrix for relation $R \subseteq A \times B$ and $M_S = [s_{ij}]$ be the matrix for relation $S \subseteq A \times B$ and assume that $M_R \odot M_S = [t_{ij}]$. Observe that

$$\begin{aligned} t_{ij} = 1 & \quad \text{iff} & \quad \exists k : r_{ik} = 1 = s_{kj} \\ & \quad \text{iff} & \quad \exists b_k \in B : (a_i, b_k) \in R \text{ and } (b_k, c_j) \in S \\ & \quad \text{iff} & \quad (a_i, c_j) \in S \circ R \end{aligned}$$

Therefore $M_{S \circ R} = M_R \odot M_S$.

EXAMPLE \triangleright $A = \{1, 2\}$, $B = \{a, b, c\}$, and $C = \{x, y, z, w\}$. Suppose $R \subseteq A \times B$ be defined as

$$R = \{(1, a), (1, c), (2, b), (2, c)\}.$$

Then the matrix representation of R will be

$$M_R = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Now suppose $S \subseteq B \times C$ be the relation

$$S = \{(a, x), (a, y), (a, w), (b, y), (b, z), (c, x), (c, w)\}$$

with the following matrix representation:

$$M_S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Then $S \circ R$ will have the matrix representation:

$$M_R \odot M_S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

and therefore

$$S \circ R = \{(1, x), (1, y), (1, w), (2, x), (2, y), (2, z), (2, w)\}.$$

9.1.2 As Directed Graphs

Definition 58 A *directed graph* $G = (V, E)$ consists of a set V (*vertices*) and a set $E \subseteq V \times V$ (*directed edges*). In drawing a graph $G = (V, E)$ we represent vertices with points and put directed edge (an arrow) from vertex a to vertex b if $(a, b) \in E$.

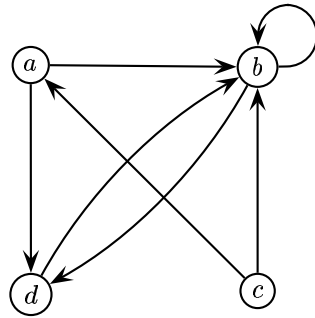
EXAMPLE \triangleright The graph $G = (V, E)$ where

$$V = \{a, b, c, d\}$$

and

$$E = \{(a, b), (a, d), (b, b), (b, d), (c, a), (c, b), (d, b)\}$$

can be depicted as

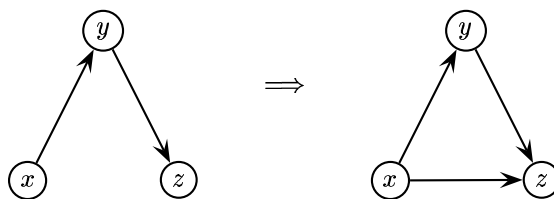


Notice that a directed graph is a relation E on a set V .

Given a relation R on a set A , the corresponding directed graph is one with vertex set $V = A$ and edge set $E = R$.

Let us denote by G_R the graph of the relation R .

- R is reflexive if and only if every vertex in G_R has a self-loop.
- R is symmetric if and only if all edges come in pairs (one in one direction and the other in the opposite direction).
- R is antisymmetric if and only if no edges come in pair (but self-loops are allowed).
- R is transitive if and only if all triangle are completed like this:



9.2 Closure

Let R be a relation and P be a property (reflective, symmetric, transitive, etc.). If R has property P , then R is closed under P . If R does not have property P , we may want to “close” R , by *adding* as few pairs as possible to yield a relation R' such that $R \subseteq R'$ and R' has property P .

Definition 59 *Closure of R with respect to P* is a relation R' such that

1. $R \subseteq R'$
2. R' has property P
3. $\forall S$ such that (1) and (2) hold for S we have $R' \subseteq S$ we have $R' \subseteq S$.

Note.

1. Closure of a relation with respect to a property may not exist.
2. If a relation R already has property P , then the closure of R with respect to P will be R itself.

EXAMPLE \triangleright R a relation on A . We will denote by $r(R)$ the reflexive closure of R .

$$r(R) = R \cup \{(a, a) : a \in A\}.$$

Clearly,

1. $R \subseteq r(R)$ is true,
2. $r(R)$ is reflexive, and
3. If $R \subseteq S$ and S reflexive then $\{(a, a) : a \in A\} \subseteq S$, so $r(R) \subseteq S$.

EXAMPLE \triangleright Let us denote by $s(R)$ the symmetric closure of R . Then

$$s(R) = R \cup \underbrace{\{(b, a) : (a, b) \in R\}}_{R^{-1}}.$$

Clearly,

1. $R \subseteq s(R)$ is true,
2. $s(R)$ is symmetric, and
3. If $R \subseteq S$ with S symmetric then $s(R) \subseteq S$ because $R \subseteq S$ and everything in $s(R) - R$ must be in S also. Otherwise S will not be symmetric.

9.2.1 Transitive Closure

Using a similar technique as in the previous two examples we may come up with the following definition for $c(R)$, the transitive closure of R :

$$c(R) = R \cup \{(a, c) : \exists b (a, b) \in R, (b, c) \in R\}.$$

But this doesn't work as shown in the following example.

EXAMPLE \triangleright $A = \{1, 2, 3, 4, 5\}$ and $R = \{(1, 2), (2, 3), (3, 4), (4, 5)\}$. Then applying the definition above, we get

$$c(R) = \{(1, 2), (2, 3), (3, 4), (4, 5), (1, 3), (2, 4), (3, 5)\}.$$

But this relation is not transitive because it contains $(1, 3)$ and $(3, 4)$ but does not contain $(1, 4)$.

Solution. Do it over and over and over and . . .

Definition 60 A *path* in a directed graph is a sequence of edges

$$(x_0, x_1), (x_1, x_2), (x_2, x_3), \dots, (x_{n-1}, x_n).$$

The path as defined above:

- is from x_0 to x_n .
- has length n .
- if $x_0 = x_n$ is a *cycle* or *circuit*.
- if all x_i 's are different is a *simple path*.

EXAMPLE \triangleright In the graph of top of page 86, adb is a simple path, $bdbb$ is a circuit, and $abdbb$ is a path.

Since relations on a set are *really* graphs, the notion of *path* makes sense also.

Definition 61 A *path* in a relation R on set A from a to b is a sequence of elements

$$a, x_1, x_2, \dots, x_{n-1}, b$$

such that

$$\begin{aligned} (a, x_1) &\in R \\ \forall 1 \leq i < n-1 & \quad (x_i, x_{i+1}) \in R \\ (x_{n-1}, b) &\in R \end{aligned}$$

Lemma 2 If $t(R)$ is the transitive closure of R and if R contains a path from a to b , then $(a, b) \in t(R)$.

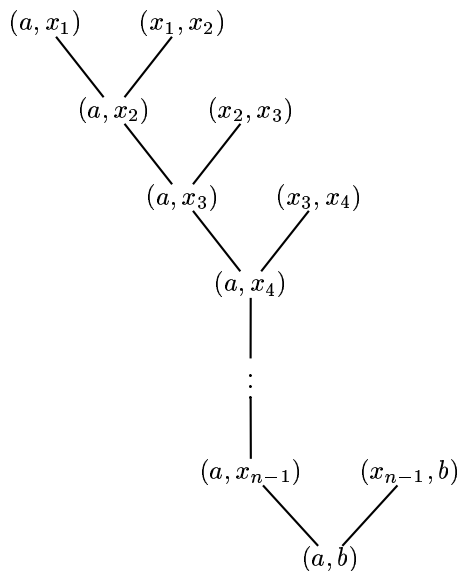
Proof. Let R contain a path

$$a \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_{n-1} \rightarrow b$$

from a to b . In other words R contains the pairs

$$(a, x_1), (x_1, x_2), \dots, (x_{n-2}, x_{n-1}), (x_{n-1}, b).$$

Then since R is transitive it will contain the other following pairs.



■

EXAMPLE $\triangleright R = \{(a, b), (a, c), (b, c), (c, d), (d, e), (d, b)\}$. Then $abcde$ is a path and therefore by the above theorem $(a, e) \in t(R)$.

To determine the transitive closure of a relation:

1. Represent the relation as a graph.
2. For each pair of vertices (a, b) , determine if there exists a path from a to b .
 - If so, then $(a, b) \in t(R)$.
 - If not, then $(a, b) \notin t(R)$.

Note.

1. The above method uses problem of checking whether there is a path between a pair of vertices (or equivalently the graph connectivity problem). You will see efficient algorithms for these problems in later classes.
2. There is a more efficient, though more confusing method. Read about *Warshall's Algorithm* in the text.
3. We can also use the Boolean Matrix Multiplication to solve this problem. The intuition is to compute $R \cup R^2 \cup \dots \cup R^n$.

Chapter 10

Equivalence Relations and Partial Orders

10.1 Equivalence Relations

The motivation to study equivalence relations is to focus only on *properties* of objects instead of objects themselves. All objects with the same property will be “equivalent”.

Definition 62 An *equivalence relation* R on a set S is a relation that is

1. **reflexive**, i.e. $\forall a \in S \ aRa$.
2. **symmetric**, i.e. $\forall a, b \in S \ aRb \leftrightarrow bRa$.
3. **transitive**, i.e. $\forall a, b, c \in S \ aRb \wedge bRc \rightarrow aRc$.

we will use the notation aRb where R is a relation to denote $(a, b) \in R$.

EXAMPLE \triangleright Let $S = \{ \text{people in the class} \}$ and

$$R = \{ (a, b) : a \text{ has the same number of coins in pocket/purse as } b \}$$

Clearly R is reflexive, symmetric, and transitive and thus R is an equivalence relation.

EXAMPLE \triangleright $S = \mathbb{Z}$ and $R = \{ (a, b) : a \equiv b \pmod{4} \}$. Thus

- $0 \equiv 4 \equiv 8 \equiv 12 \equiv 16 \cdots$ (all with remainder 0),

- $1 \equiv 5 \equiv 9 \equiv 13 \equiv 17 \cdots$ (all with remainder 1),
- $2 \equiv 6 \equiv 10 \equiv 14 \equiv 18 \cdots$ (all with remainder 2),
- $3 \equiv 7 \equiv 11 \equiv 15 \equiv 19 \cdots$ (all with remainder 3).

EXAMPLE \triangleright Let $S = \{\text{people of this class}\}$ and

$$R = \{(a, b) : \text{total cash on } a \text{ is within } \$1.0 \text{ of total cash on } b\}$$

This relation is reflexive and symmetric by it is not transitive because a , b , and c can have \$20, \$21 and \$22 respectively. Then we have aRb and bRc but $a \not R c$.

10.2 Equivalence Classes

Definition 63 Let R be an equivalence relation on S . The *equivalence class* of $a \in S$ written $[a]_R$ is defined by

$$[a]_R = \{b \in S : (a, b) \in R\}.$$

Thus $[a]_R$ is the set of all elements of S that a relates to, i.e. is equivalent to.

EXAMPLE \triangleright Taking the equivalence relation on a few example ago based on the number of coins in pocket/purse we have

$$[\text{Prof. Pitt}]_R = \{\text{people with 5 coins in pocket/purse}\}.$$

Note that the collection of all equivalence classes for this relation are

- $\{\text{people with 0 coins in pocket/purse}\}$
- $\{\text{people with 1 coins in pocket/purse}\}$
- $\{\text{people with 2 coins in pocket/purse}\}$
- $\{\text{people with 3 coins in pocket/purse}\}$
- \dots

EXAMPLE \triangleright $S = \mathbb{Z}$ and $R = \{(a, b) : a \equiv b \pmod{4}\}$. Then $[17]_R = \{\dots, -7, -3, 1, 5, 9, 13, 17, \dots\}$. All classes are $[0]$, $[1]$, $[2]$, and $[3]$.

Definition 64 We call a a *representative* for equivalence class $[a]_R$.

Note that if aRb (thus, if $b \in [a]_R$), then b is also a representative of $[a]_R$.

EXAMPLE \triangleright 17 is a representative of the class $\{\dots, -7, -3, 1, 5, 9, 13, 17, \dots\}$. In fact every number of the form $4k + 1$ is a representative for this class.

Question. What do the equivalence classes of a relation R on set S look like?

We will show that they *subdivide* S into *disjoint pieces*. **In particular, we can't have two different equivalence classes with non empty intersection.**

To see this, consider the following lemma.

Lemma 3 Let R be an equivalence relation on a set S . Then

- a) If aRb then $[a]_R = [b]_R$ (the equivalence classes are identical).
- b) If $a \not R b$ then $[a]_R \cap [b]_R = \emptyset$ (the equivalence classes are completely disjoint).

Proof. a) Suppose aRb . Now for $x \in S$:

$$\begin{array}{ll}
 x \in [a]_R \Leftrightarrow aRx & \text{definition of } [a]_R \\
 \Leftrightarrow xRa & \text{symmetry of } R \\
 \Leftrightarrow xRb & xRa \text{ and } aRb \\
 & \text{and } R \text{ is transitive} \\
 \Leftrightarrow bRx & \text{symmetry of } R \\
 \Leftrightarrow x \in [b]_R & \text{definition of } [b]_R
 \end{array}$$

Thus $x \in [a]_R \Leftrightarrow x \in [b]_R$. So, $[a]_R = [b]_R$.

b) Proof by contradiction. Suppose to the contrary that

$$\exists x \in [a]_R \cap [b]_R.$$

Then by the definition of $[a]_R$ and $[b]_R$ we know that aRx and bRx . By symmetry these become xRa and xRb . But then by transitivity of R we obtain aRb contradicting the assumption that $a \not R b$. \blacksquare

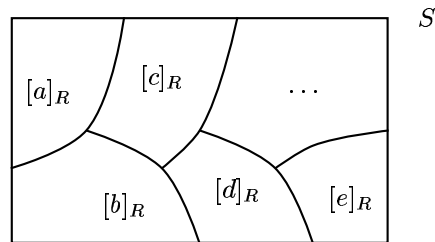
The two parts of the previous lemma together show that $[a]_R$ and $[b]_R$ are always **either** identical **or** disjoint.

Note that if R is an equivalence relation on S then

why?

$$S = \bigcup_{a \in S} [a]_R.$$

Thus the equivalence classes of R “completely cover” S in the following way: S is the union of disjoint equivalence classes of R .



Such a cover of a set has a different name.

Definition 65 A *partition* of a set S is a (perhaps infinite ... perhaps uncountably infinite) collection of sets $\{A_i\}$ such that

- each A_i is non-empty.
- each $A_i \subseteq S$.
- $\cup A_i = S$.
- $\forall i, j \quad A_i \cap A_j = \emptyset$.

Each A_i is called a *block* of the partition.

EXAMPLE $\triangleright \mathbb{R} = \{\text{rationals}\} \cup \{\text{irrationals}\}$ (2 blocks).

EXAMPLE $\triangleright \mathbb{R} = \{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{\text{other reals}\}$.

Theorem 6 a) If R is an equivalence relation on S , then

$$\{[a]_R : a \in S\}$$

is a partition of S , i.e., the equivalence classes of R partition S , and each equivalence class is a block of this partition.

b) If $\{A_i\}$ is *any* partition of S , then there exists an equivalence relation whose equivalence classes are exactly the blocks A_i of S .

Proof. a) We’ve already noted that $S = \cup_{a \in S} [a]_R$, and each $[a]_R$ is nonempty (because it at least contains a itself) and $[a]_R \cup [b]_R = \emptyset$ (unless aRb).

b) If $\{A_i\}$ partitions S , then define the equivalence relation R on S by

$$R = \{(a, b) : \exists i \ a \in A_i \text{ and } b \in A_i\}.$$

(Thus aRb if and only if a and b are both in the same block A_i together). Clearly, R is reflexive, symmetric. We now show that R is transitive as well.

Suppose aRb and bRc . Then by the definition of R given above:

1. There exists an i such that a and b are both in A_i , and
2. There exists a j such that b and c are both in A_j .

But $b \in A_i \cap A_j$, so A_i must equal A_j (else wasn't a partition). So, $a, b, c \in A_i$ and in particular aRc . ■

10.3 Partial Orders

Definition 66 If R is a relation on a set S , then (S, R) is a *partial order* and S is called a *partially ordered set* (or a *poset*) if and only if R is

1. **reflexive**, i.e. $\forall a \ aRa$
2. **transitive**, i.e. $\forall a, b, c \ aRb \wedge bRc \rightarrow aRc$
3. **antisymmetric**, i.e. $\forall a, b \ aRb \wedge bRa \rightarrow a = b$.

EXAMPLE $\triangleright (\leq, \mathbb{R})$, i.e. the “less than or equal” relation on \mathbb{R} is a partial order.

EXAMPLE \triangleright “divides” on \mathbb{Z}^+ , i.e. $(\mathbb{Z}^+, |)$ is a partial order. This because

- *reflexive*: $a|a$
- *transitive*: If $a|b$ and $b|c$ then $a|c$ (easy)
- *antisymmetric*: If $a|b$ and $b|a$, then there exist k_1, k_2 both positive such that $k_1a = b$ and $k_2b = a$. Thus $k_1a = a/k_2$, or $k_1k_2 = 1$. So $k_1 = k_2 = 1$ and thus $a = b$.

EXAMPLE $\triangleright (\mathbb{Z}, |)$ is not a partial order because $1|-1$ and $-1|1$ but $1 \neq -1$.

EXAMPLE \triangleright Let S be any set. Then $(2^S, \subseteq)$ is a partial order.

Definition 67 Notation. The sign “ \leq ” will be used to define an arbitrary relation that is reflexive, transitive, and antisymmetric. We will write $a \leq b$ to mean aRb . We will also write $a < b$ if $a \leq b$ but $a \neq b$.

EXAMPLE \triangleright A standard partial order on $\{0, 1\}^n$ (the set of bit strings of length n) is defined as

$$a_1a_2 \dots a_n \leq b_1b_2 \dots b_n$$

if and only if

$$\forall i \quad a_i \leq b_i.$$

Thus $0110 \leq 1110$ and 0110 is incomparable to 1011 . The above relation defines a partial order which corresponds exactly with the partial order of $(2^S, \subseteq)$ for a set S of size n .

For example for a set $S = \{a, b\}$ of size two, the power set of S consists of

$$\emptyset, \{b\}, \{a\}, \{a, b\}$$

Think of bit-strings as characteristic vectors of the elements of the power set.

$$00, 01, 10, 11$$

. In the relation we defined on bit strings $00 < 10$ because each bit of 00 is less than or equal to the corresponding bit of 10 . In other words 10 has at least all of the “1” bits as 00 , and perhaps more.

So the set 10 represents *contains* the set represented by 00 and in fact we can check that $\emptyset \subseteq \{a\}$.

Definition 68 Let (S, \leq) be a partial order. Then a and b are *comparable* if either $a \leq b$ or $b \leq a$ (or both).

Otherwise, they are *incomparable*.

EXAMPLE \triangleright For $(\mathbb{Z}, |)$: 6 and 3 are comparable, 3 and 3 are comparable, 3 and 5 are incomparable, and $8, 12$ are incomparable.

In fact we call a partially order set so since not all things need to be comparable.

Definition 69 A *total order* is a partial order in which every pair of elements are comparable.

EXAMPLE \triangleright (\mathbb{Z}, \leq) is a total order.

A total order used a lot in CS.

EXAMPLE \triangleright **Lexicographic ordering of strings.** This is defined below.

Definition 70 Having fixed a usual order on an alphabet Σ , for example the order $a < b < c < d < \dots < z$ if the alphabet are English letters, for any pair of given strings $\alpha = x_1x_2\dots x_m$ and $\beta = y_1y_2\dots y_n$ in Σ^* we define the following ordering:

To determine if $\alpha \leq \beta$ let $t = \min\{m, n\}$ (shorter length). If

$$x_1x_2\dots x_t = y_1y_2\dots y_t$$

then shorter of α and β comes first. Else let $i \leq t$ be the smallest number such that $x_i \neq y_i$ (i is the first position where they differ). If $x_i < y_i$ then $\alpha < \beta$ and otherwise $\beta < \alpha$.

EXAMPLE \triangleright For the usual ordering on English alphabet we have
discreet $<$ discreetness $<$ discrete $<$ discretion

10.3.1 Hasse Diagrams

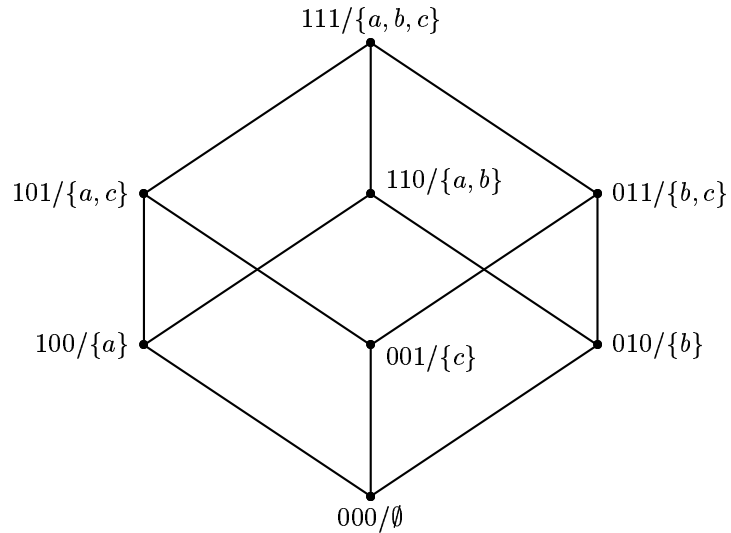
We can represent posets as graphs. Consider the partial order $(\{1, 2, 3, 4\}, \leq)$.



In drawing Hasse diagrams

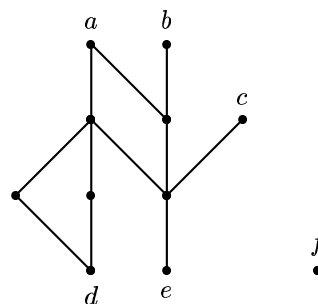
- We draw an edge from a to b if and only if $a \leq b$
- We do not draw the direction of the edges and assume that they all point up.
- We do not draw self-loops, since we know they are there (all posets are reflexive).
- We do not bother drawing transitive edges (transitive edges are eliminated).

EXAMPLE $\triangleright (2^{\{a,b,c\}}, \subseteq)$ (same as bitwise \leq on $\{0, 1\}^3$).



Definition 71 Let (S, \leq) be a partial order. Then an element $a \in S$ is a *minimal element* if there exists no b such that $b < a$. Similarly, $a \in S$ is called a *maximal element* if there exists no b such that $a < b$.

EXAMPLE \triangleright Consider the poset represented by the following Hasse diagram



Maximal elements of the above poset are $\{a, b, c, f\}$ and the minimal elements are $\{d, e, f\}$.

Note. maximal and minimal elements need *not* be unique and a poset can have several maxima and minima. Some points can be both maximal and minimal.

Question. Must there always exist some maximal or minimal elements?

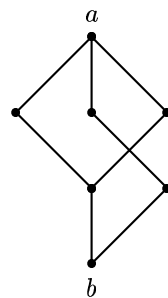
Answer. Only if set is finite (consider (\mathbb{Z}, \leq)).

Definition 72 In a poset S an element a is a *minimum element* if $\forall b \in S \ a \leq b$. Similarly, an element a is a *maximal element* if $\forall b \in S \ b \leq a$.

Note the difference of mum and mal.

Note. Intuitively when we say a is *maximal* it means that *nothing beats a* . But when we say that a is *maximum* it means that *a beats everything*.

EXAMPLE \triangleright In the following poset a is maximum and b is minimum.



Question. Must maximum and minimum exist? **Answer.** No, same reason.

Note. If a is a maximum element, then a is a maximal element too (why?). Similarly, if a is a minimum element, then a is a minimal element too (why?).

Question. If maximum exists, is it unique i.e., can there be two different maximums?

Theorem 7 In every poset if the maximum element exists, it is unique. Similarly, in every poset if the minimum element exists, it is unique.

Proof. If $a_1 \neq a_2$ and both are maximum, then $a_1 \leq a_2$ because a_2 is maximum and $a_2 \leq a_1$ because a_1 is maximum. But then since " \leq " is antisymmetric, this implies that $a_1 = a_2$ contradicting $a_1 \neq a_2$. So, only one maximum may exist.

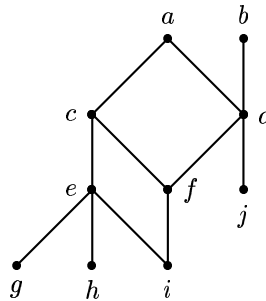
The argument for the uniqueness of minimum is similar. ■

10.3.2 Upper and Lower Bounds

Definition 73 Let (S, \leq) be a partial order. If $A \subseteq S$, then an *upper bound* for A is any element of S (perhaps in A also) such that $\forall a \in A \ a \leq x$.

Similarly, a lower bound for A is any $x \in S$ such that $\forall a \in A \ x \leq a$.

EXAMPLE ▷ consider the following poset.



Here the only upper bound of $\{g, j\}$ is a , upper bounds of $\{g, i\}$ are a, c , and e . The upper bounds of $\{i, j\}$ are a, b , and d . The only upper bound of $\{a, d\}$ is a and $\{a, b\}$ have no upper bound.

The lower bounds of $\{a, b\}$ are d, f, i , and j . The lower bounds of $\{c, d\}$ are f , and i . and $\{g, j\}$ has no lower bounds.

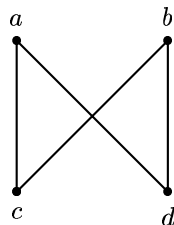
Definition 74 Given a poset (S, \leq) , and $A \subseteq S$, an element $x \in S$ is a *greatest lower bound* for A if x is a lower bound and for lower bound y of A , $y \leq x$.

Similarly, x is a *least upper bound* for A if x is an upper bound and if $x \leq y$ for every upper bound y of A .

EXAMPLE ▷ In the previous example $\text{lub}(g, j) = a$, $\text{lub}(g, i) = e$, $\text{lub}(i, j) = d$, $\text{glb}(g, j)$ does not exist. $\text{glb}(c, d) = f$ and $\text{glb}(a, b) = d$.

Note. Sometimes lub and glb don't exist, even though lower and upper bounds do exist.

EXAMPLE ▷ In the following poset



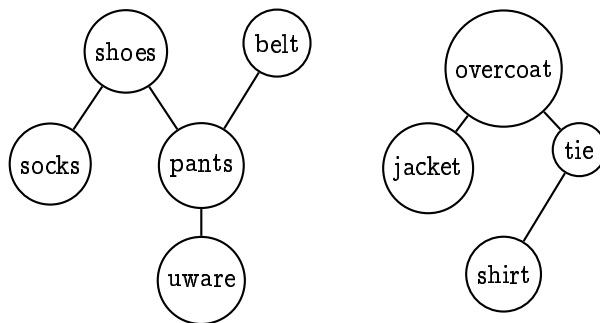
c and d are both lower bounds for $\{a, b\}$ but neither is a glb. Also a and b are both upper bounds for $\{c, d\}$ but neither is a lub.

10.3.3 From Partial Orders to Total Orders

Consider the problem of getting dressed.

Who said this course has no practical applications?

Precedence constraints are modeled by a poset in which $a \leq b$ if and only if you must put on a before b .



Problem. how to get dressed, while respecting the constraints?

Clearly many solutions are possible. This is the problem of *extending* a partial order to a total order.

Let (S, \leq) be a partial order with S finite. We show how to extend \leq to a total order on S . (i.e. can decide for all incomparable pair a and b whether to make $a \leq b$ or vice versa, without violating transitivity, reflexivity, or antisymmetry.)

Lemma 4 Every finite nonempty poset (S, \leq) has *at least* one minimal element.

Proof. Choose $a_0 \in S$. If a_0 was not minimal, then there exists an $a_1 \leq a_0$, so choose a_1 . If this is not minimal, choose $a_2 \leq a_1$, and so on until a minimal element is found. (why must this work?) ■

Lemma 5 If (S, \leq) is a poset with a minimal, then $(S - \{a\}, \leq)$ is also a poset.

Proof. If you remove minimal a , reflexivity still holds and antisymmetry also still holds. If $x, y, z \in S - \{a\}$, with $x \leq y$ and $y \leq z$, then $x \leq z$ also, since (S, \leq) was transitive. ■

Think about Lemmas 1 and 2:

1. Always there exists a minimal element.
2. If you remove it, you still have a poset.

This suggests:

```

algorithm TOPOLOGICAL SORT
Input: poset  $(S, \leq)$ 
Output: elements of  $S$  in some total order.

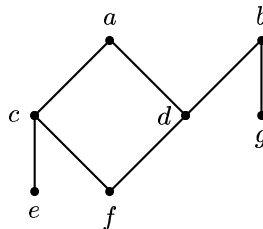
1  begin
2      Remove any minimal element from  $S$  and output it
3      If  $S \neq \emptyset$  goto 2
7  end

```

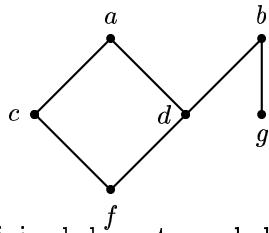
Depending on *which* minimal element is chosen at step 2 each time, a different total order may be obtained. But all obtainable total orders will be consistent with the original partial order.

Probably an easy inductive proof shows this.

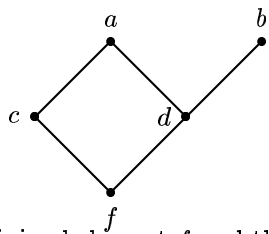
EXAMPLE ▷ For the following poset



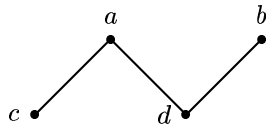
We first remove the minimal element e and obtain



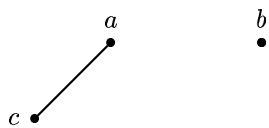
Then we remove the minimal element g and obtain



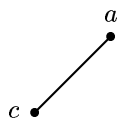
Then we remove the minimal element f and the poset becomes



Then we remove d and we get



Then we remove b and get



The next minimal element will be c and the resulting poset will be

a
•

Eventually we remove c and the algorithm terminates. The resulting total order is then

$$e \leq g \leq f \leq d \leq b \leq c \leq a.$$