

COURSE MA1015 – DISCRETE MATHEMATICS.

(This is one half of the mathematics course for Computer Science, the other half is MA1010 – Calculus).

1) Lecturer: Dr. A.K. Seda, Department of Mathematics, Room 268, Aras na Laoi.

2) Timetable: Monday 4–5 LL2 and Thursday 9–10 LL3.

3) Tutorials: timetable to follow.

4) Set textbook: “Discrete Mathematics” 4th Edition, by Richard Johnsonbaugh. Published by Prentice Hall (available in Waterstones or maybe second hand; there are some copies in the library). This book **MUST** be obtained.

Additional reading (not necessary to buy):

- “The Essence of Discrete Mathematics” by Neville Dean. Published by Prentice Hall.

- “Logic and Discrete Mathematics – a Computer Science Perspective” by W. Grassman and J.P. Tremblay. Published by Prentice Hall.

5) Exercise sheets: these will be issued regularly – to be done and discussed in tutorials.

6) Mid-Term Exam – worth 20% of Summer Mark.

7) Difficulties? – approach me, your tutor or the class rep. (when appointed).

COURSE CONTENTS

For exact syllabus, see Calendar (Book of Modules). This course is intended to aid studies in Computer Science concerned with issues such as:

- Analysis of algorithms. E.g. is bubble sort “better” than quicksort i.e. does it use less memory, is it faster, is it more efficient etc.?
- Verification of software (Formal Methods). How can we prove that a piece of software is correct? This is done by use of systems such as Z, VDM, B etc. and these depend on logic and discrete mathematics.
- Use of logic in computer science: checking correctness (as already mentioned), use of logic in artificial intelligence – automating reasoning, expert systems, programming languages such as Prolog and Parlog.
- The problem of giving semantics (machine-independent meaning) to computer programs.

Etc., etc.

Full details of the course can be found by clicking on *Details of the contents of the course* on my home page.

DISCRETE MATHEMATICS

Page references are to Johnsonbaugh unless otherwise stated.

1 SETS AND FUNCTIONS (pp 64–72)

1.1 Definition A *set* is any collection of objects. The objects are usually called the *elements* or *members* of the set.

We denote sets usually by capital letters $A, B, C, \dots, S, T, \dots, X, Y, Z, A_0, B_1, \dots$ etc; elements of sets are usually denoted by lower case letters $a, b, c, \dots, s, t, \dots, x, y, z, a_0, b_1$ etc.

Sets can be represented pictorially by Venn diagrams:

If a is an element of A , we write $a \in A$; if a is not an element of A we write $a \notin A$ (“ \in ” is the symbol for membership). If each element b of a set B is s.t. (such that) also $b \in A$, we call B a subset of A and write $B \subseteq A$. If $B \subseteq A$ and furthermore there is at least one $a \in A$ s.t. $a \notin B$, then B is called a proper subset of A , written $B \subset A$. So the above diagram shows $B \subset A$.

1.2 Example Examples and notation

(i) $A = \{1, 2, 3, 4\}$ is a set containing the four elements 1, 2, 3 and 4. Thus $1 \in A, 2 \in A, 3 \in A, 4 \in A$ but $0 \notin A, 5 \notin A$ etc. If $B = \{1, 2\}$, then $B \subset A$

(ii) $N = \{0, 1, 2, 3, \dots\}$ – the set of natural numbers.

(iii) $Z = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$ – the set of all integers.

(iv) $Q = \{\frac{p}{q}; p, q \in Z, q \neq 0 \text{ and } p \text{ and } q \text{ have no common divisors}\}$ – the set of all rational numbers (fractions).

Note this very important way of specifying sets, especially infinite sets which cannot be listed:

$A = \{ \text{symbol e.g. } x; \text{some condition(s) on } x \}$.

(v) $R = \text{set of all } \underline{\text{real numbers}}$

We have $N \subset Z \subset Q \subset R$.

(vi) $X = \{a, b, c, \dots, x, y, z\}$ – the set of Arabic symbols.

(vii) \emptyset denotes the empty or null set i.e. “the” set which contains no elements. We have (a) $\emptyset \subseteq A$, and (b) $A \subseteq A$ for all sets A .

(viii) Lists over a set A (an important data structure especially in logic and functional programming and an example of inductive definitions of sets). Let A be a non-empty set:

1. The empty list (the empty subset of A) is a list, denoted by $[\]$.

2. If $h \in A$ and T is a list, then $h.T$ or $[h \mid T]$ is a list – we call h the head of the list $h.T$ and T its tail.

3. Nothing else is a list.

Note that 1., 2. and 3. are called rules of formation.

Examples of lists. Let $A = \{1, 2, 3, 4\}$.

Then $[\]$ is a list

$[3 \mid [\]]$ is a list.

$[2 \mid [3]]$ is a list.

$[4 \mid [2, 3]]$ is a list.

$[1 \mid [4, 2, 3]]$ is a list.

$[1 \mid [1, 4, 2, 3]]$ is a list

and so on. Let L_A denote the set of all lists over A . Then $[\] \in L_A, [1, 1, 1] \in L_A, [1, 2, 1] \in L_A$ etc.

1.3 Definition A set A is called *finite* if it contains only finitely many elements, and is called *infinite* otherwise. If A is finite, we denote by $|A|$ the number of elements it contains and call this number the *order* of A .

1.4 Example (1) $|\emptyset| = 0$.

(2) Let a be any object. We write $\{a\}$ for the set whose only element is a – called the singleton set containing a . Clearly we have $a \in \{a\}$ and $|\{a\}| = 1$ always. Notice that, in particular, $\emptyset \in \{\emptyset\}$, $\emptyset \subset \{\emptyset\}$ and $|\{\emptyset\}| = 1$.

(3) $|\{1, 2, 3, 4\}| = 4$.

(4) N, Z, Q and R are infinite sets and their orders are not defined (not here anyway – they can be). Nor is $|L_A|$ defined since L_A is also an infinite set.

(5) If X is the set of arabic symbols, then $|X| = 26$.

1.5 Definition (Important definition) Two sets A and B are said to be *equal*, written $A = B$, if they have the same elements. Therefore, $A = B$ iff $A \subseteq B$ and $B \subseteq A$.

Consequences of this definition.

(1) To show A and B are equal, we show $A \subseteq B$ and $B \subseteq A$ i.e. we show “whenever $x \in A$, we have $x \in B$ ” and “whenever $x \in B$, we have $x \in A$ ” – see examples of this later.

(2) The sets $\{1, 2, 3, 4\}$ and $\{1, 1, 2, 3, 4\}$ are equal because they contain the same elements. Thus, $|A|$ means the number of distinct elements in A .

(3) The sets $\{1, 2, 3, 4\}$ and $\{1, 3, 2, 4\}$ are equal because they contain the same elements. Therefore the order in which el-

ements occur in a set does not matter.

1.6 Construction The power set of a set. For any set X we denote by $\mathcal{P}(X)$ the power set of X i.e. the set of all subsets of X .

1.7 Example (1) Take $X = \emptyset$ – the empty set. The only subset of \emptyset is \emptyset ($\emptyset \subseteq \emptyset$). Thus, $\mathcal{P}(\emptyset) = \{\emptyset\}$. Note that $|\mathcal{P}(\emptyset)| = 1 = 2^0$.

(2) Take $X = \{a\}$ – a set containing one element. Possible subsets of X are \emptyset and $\{a\}$. So $\mathcal{P}(X) = \{\emptyset, \{a\}\}$. Note that $|\mathcal{P}(X)| = 2 = 2^1$.

(3) Take $X = \{a, b\}$ (or any other set containing two elements). Possible subsets are: $\emptyset, \{a\}, \{b\}, \{a, b\}$. So $\mathcal{P}(X) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ and $|\mathcal{P}(X)| = 4 = 2^2$.

(4) Take $X = \{a, b, c\}$ (any set of three elements). Then $\mathcal{P}(X) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$. In this case $|\mathcal{P}(X)| = 8 = 2^3$.

(5) If X is infinite, then also $\mathcal{P}(X)$ is infinite.

1.8 Proposition If $|X| = n$, then $|\mathcal{P}(X)| = 2^n$.

Proof. Recall the binomial expansion

$$(a + b)^n = a^n + C(n, 1)a^{n-1}b + C(n, 2)a^{n-2}b^2 + \dots + C(n, r)a^{n-r}b^r + \dots + C(n, n-1)ab^{n-1} + b^n,$$

where $C(n, r) = \text{binomial coefficient} = \frac{n!}{r!(n-r)!} = \text{number of combinations of } n \text{ things taken } r \text{ at a time. Put } a = b = 1$
to get $2^n = (1 + 1)^n$
 $= 1^n + C(n, 1)1^{n-1}.1 + C(n, 2)1^{n-2}.1^2 + \dots +$
 $C(n, r)1^{n-r}.1^r + \dots + C(n, n - 1)1.1^{n-1} + 1^n$
 $= 1 + C(n, 1) + C(n, 2) + \dots + C(n, r) + \dots + C(n, n - 1) + 1.$

The first “1” corresponds to \emptyset

$C(n, 1)$ corresponds to number of subsets of X of size 1

$C(n, 2)$ corresponds to number of subsets of X of size 2

and in general

$C(n, r)$ corresponds to number of subsets of X of size r .

The last “1” corresponds to the subset X itself. Therefore we get $|\mathcal{P}(X)| = 2^n$ as required. ■

Another proof can be given using mathematical induction.

OPERATIONS ON SETS

Given two sets X and Y , there are various ways to combine them to obtain a new set:

(1) The set $X \cup Y = \{z; z \in X \text{ or } z \in Y\}$ called the union of X and Y . Note the inclusive “or” here i.e. z here can be in both X and Y :

Example If $X = \{1, 2, 3, 4\}$ and $Y = \{3, 4, 6, 7\}$, then $X \cup Y = \{1, 2, 3, 4, 6, 7\}$.

(2) The set $X \cap Y = \{z; z \in X \text{ and } z \in Y\}$ called the intersection of X and Y .

Example With X and Y as in (1), $X \cap Y = \{3, 4\}$.

Note We call X and Y disjoint if $X \cap Y = \emptyset$.

(3) The set $X \setminus Y = \{z; z \in X \text{ and } z \notin Y\}$ called the difference (or relative complement) of X and Y .

Example With X and Y as in (1) and (2), $X \setminus Y = \{1, 2\}$ and $Y \setminus X = \{6, 7\}$.

Thus, we see that $X \setminus Y \neq Y \setminus X$ in general.

Sometimes we are dealing with sets all of which are subsets of a set U . This set U is called a universal set or universe. We write \bar{X} for $U \setminus X$ and call \bar{X} the complement of X in U .

1.9 Theorem (see Theorem 2.1.8 on page 67). Let U be a universal set and let A , B , and C be subsets of U . Then the following hold:

(a) Associative laws

$$(A \cup B) \cup C = A \cup (B \cup C), \quad (A \cap B) \cap C = A \cap (B \cap C)$$

(b) Commutative laws

$$A \cup B = B \cup A, \quad A \cap B = B \cap A$$

(c) Distributive laws

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

(d) Identity laws

$$A \cup \emptyset = A, \quad A \cap U = A$$

(e) Complement laws

$$A \cup \bar{A} = U, \quad A \cap \bar{A} = \emptyset$$

(f) Idempotent laws

$$A \cup A = A, \quad A \cap A = A$$

(g) Bound laws

$$A \cup U = U, \quad A \cap \emptyset = \emptyset$$

(h) Absorption laws

$$A \cup (A \cap B) = A, \quad A \cap (A \cup B) = A$$

(i) Involution law

$$\bar{\bar{A}} = A$$

(j) 0/1 laws

$$\bar{\emptyset} = U, \quad \bar{U} = \emptyset$$

(k) De Morgan's laws for sets

$$\overline{A \cup B} = \bar{A} \cap \bar{B}, \quad \overline{A \cap B} = \bar{A} \cup \bar{B}.$$

Proof Exercise 70 page 72. We give some sample proofs. We prove the first of the distributive laws in (c) and the first of De Morgan's laws in (k).

To prove $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

We must show that $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ and $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$.

Let $x \in A \cap (B \cup C)$ be arbitrary. Then $x \in A$ and $x \in (B \cup C)$. Therefore $x \in A$ and either $x \in B$ or $x \in C$. Therefore $x \in A$ and $x \in B$ or $x \in A$ and $x \in C$. Therefore $x \in A \cap B$ or $x \in A \cap C$ and hence $x \in (A \cap B) \cup (A \cap C)$. Since x is arbitrary we now have the inclusion $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Conversely, let $x \in (A \cap B) \cup (A \cap C)$ be arbitrary. Then $x \in A \cap B$ or $x \in A \cap C$. Therefore $x \in A$ and $x \in B$ or $x \in A$ and $x \in C$. Therefore $x \in A$ and either $x \in B$ or $x \in C$. Hence $x \in A$ and $x \in (B \cup C)$ and so $x \in A \cap (B \cup C)$.

Since x is arbitrary we therefore have $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$ and hence $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

To prove $\overline{A \cup B} = \bar{A} \cap \bar{B}$.

We must show $\overline{A \cup B} \subseteq \bar{A} \cap \bar{B}$ and $\bar{A} \cap \bar{B} \subseteq \overline{A \cup B}$.

Let $x \in \overline{A \cup B} = U \setminus (A \cup B)$ be arbitrary. Then $x \notin (A \cup B)$. Hence $x \notin A$ and $x \notin B$. Therefore $x \in \bar{A}$ and $x \in \bar{B}$ so that $x \in \bar{A} \cap \bar{B}$. Hence $\overline{A \cup B} \subseteq \bar{A} \cap \bar{B}$.

Conversely, let $x \in \bar{A} \cap \bar{B}$ be arbitrary. Then $x \in \bar{A}$ and $x \in \bar{B}$. Therefore $x \notin A$ and $x \notin B$. Hence $x \notin A \cup B$ and therefore $x \in \overline{A \cup B}$. Since x is arbitrary we get $\bar{A} \cap \bar{B} \subseteq \overline{A \cup B}$.

Hence $\overline{A \cup B} = \bar{A} \cap \bar{B}$ as required. ■

Families of sets

Suppose A_1, A_2, \dots, A_n are sets. We denote by \mathcal{S} the set $\{A_1, A_2, \dots, A_n\}$ whose elements are the sets A_1, A_2, \dots, A_n . We write $\mathcal{S} = \{A_i, i = 1, \dots, n\}$ or $\mathcal{S} = \{A_i; i \in \{1, 2, \dots, n\}\}$ or $\mathcal{S} = \{A_i\}$ etc. and call \mathcal{S} a family of sets.

More generally, if I is any set (called an index set) and A_i is a set for each $i \in I$, we write $\mathcal{S} = \{A_i; i \in I\}$ or $\mathcal{S} = \{A_i\}_{i \in I}$ for the set whose elements are the sets A_i . We call \mathcal{S} a family of sets indexed by I.

Example

Suppose $I = \{1, 2, 3\}$ and $A_1 = \{0, 1\}$, $A_2 = \emptyset$, $A_3 = Z$.
Then $\mathcal{S} = \{A_i; i \in I\} = \{A_1, A_2, A_3\} = \{\{0, 1\}, \emptyset, Z\}$.

If $\mathcal{S} = \{A_i; i \in I\}$ is a family of sets, we can define the union of the family, written $\cup_{i \in I} A_i$, by

$\cup_{i \in I} A_i =$ set of those elements a s.t. a belongs to at least one A_i in the family $= \{a; a \in A_i \text{ for at least one } i\}$.

We also define the intersection of the family, written $\cap_{i \in I} A_i$, by $\cap_{i \in I} A_i =$ set of those elements a s.t. a belongs to every A_i in the family $= \{a; a \in A_i \text{ for every } i \in I\}$.

1.10 Definition A *partition* $\mathcal{S} = \{A_i; i \in I\}$ of a set X is a family \mathcal{S} of sets s.t. every member of X belongs to precisely one set in the family \mathcal{S} ie

(1) $\cup_{i \in I} A_i = X$.

(2) The sets A_i are *pairwise disjoint* ie if $i, j \in I$ and $i \neq j$, then $A_i \cap A_j = \emptyset$. Thus, no pair of distinct sets A_i, A_j “overlap”.

Pictorially:

so each $x \in X$ belongs to one and only one of the A_i s.

Example

(1) Let $A_1 =$ set of all even integers,

let $A_2 =$ set of all odd integers.

Then $\mathcal{S} = \{A_1, A_2\}$ is a partition of Z (set of all integers).

(2) Let $A_1 = Z^+$ be the set of all positive integers,

let $A_2 = \{0\}$,

let $A_3 = Z^-$ be the set of all negative integers.

Then $\mathcal{S} = \{A_1, A_2, A_3\}$ is a partition of Z .

(3) Consider the plane:

given $x \in R$, let L_x be the vertical line (parallel to y-axis) passing through x . We note that any point (x, y) in the plane belongs to L_x and that two distinct lines L_x and $L_{x'}$ have no points in common (ie $L_x \cap L_{x'} = \emptyset$ if $x \neq x'$). Thus, the family $\mathcal{S} = \{L_x; x \in R\}$ of lines partitions the plane.

Note that the index set I in this case is R and is infinite.

Exercise

If X is a finite set and $\mathcal{S} = \{A_i; i = 1, 2, \dots, n\}$ is a partition of X , then $|X| = |A_1| + |A_2| + \dots + |A_n|$.

Cartesian product of sets.

Let X and Y be sets. We define $X \times Y = \{(x, y); x \in X, y \in Y\}$ - called the cartesian product of X and Y . We call the elements (x, y) of $X \times Y$ ordered pairs.

Note Order matters in ordered pairs ie we define $(x_1, y_1) = (x_2, y_2)$ iff $x_1 = x_2$ and $y_1 = y_2$. So in general $X \times Y \neq Y \times X$.

Example

Let $X = \{1, 2, 3\}$ and $Y = \{a, b\}$.

$$X \times Y = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$$

$$Y \times X = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$$

$$X \times X = \{(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), \\ (3, 1), (3, 2), (3, 3)\}$$

$$Y \times Y = \{(a, a), (a, b), (b, a), (b, b)\}$$

Note that $(1, a) \neq (a, 1)$ and indeed $(1, a) \notin Y \times X$ so $X \times Y \neq Y \times X$ etc.

Note that if X and Y are finite sets, then

$$|X \times Y| = |X||Y| = |Y||X| = |Y \times X| \quad (\text{exercise}).$$

So, even though $X \times Y \neq Y \times X$ these sets have the same number of elements.

1.11 The Principle of Mathematical Induction (See pp 46-51 of Textbook).

This is a very powerful method of proof and states:

Suppose $S(n)$ is a statement or proposition depending on the natural number n (so, for each n , $S(n)$ is either true or false). Suppose:

- (i) Basis step $S(n)$ is true for $n = 1$ ie $S(1)$ is true.
- (ii) Inductive step $S(k + 1)$ is true whenever $S(k)$ is true.

Then $S(n)$ is true for all n .

To apply this principle we do:

Step 1 Check that $S(1)$ is true.

Step 2 *Assume* $S(k)$ is true and from this *prove* that $S(k+1)$ is true.

This establishes the truth of $S(n)$ for all n because:

$n = 1$. $S(1)$ is true by Step 1.

$n = 2$. $S(2) = S(1 + 1) = S(k + 1)$ with $k = 1$. But $S(k) = S(1)$ is true. Therefore $S(2)$ is true by Step 2.

$n = 3$. $S(3) = S(2 + 1) = S(k + 1)$ with $k = 2$. Since $S(2)$ is true, $S(3)$ is true by Step 2.

$n = 4$. $S(4) = S(3 + 1) = S(k + 1)$ with $k = 3$. Since $S(3)$

is true, $S(4)$ is true by Step 2, etc. Hence $S(n)$ true all n .

Notes

(1) The basis step (i) can be taken as “ $S(0)$ is true ie verify $S(0)$ is true” (assuming $S(0)$ makes sense).

(2) There are other forms of this principle (strong form, structural induction etc.).

Example 1

Prove that $1 + 3 + 5 + \dots + (2n - 1) = n^2$ for each natural number n .

Solution

Here $S(n)$ is the statement: $1 + 3 + 5 + \dots + (2n - 1) = n^2$ and, for each n , $S(n)$ is clearly either true or false.

Step 1 Check $S(1)$ is true.

LHS with $n = 1$ is just 1

RHS is $1^2 = 1$. Thus LHS = RHS and so $S(1)$ is true.

Step 2 Assume $S(k)$ is true ie assume that

$$1 + 3 + 5 + \dots + (2k - 1) = k^2 \dots (*)$$

We have to *prove* that $S(k + 1)$ is true ie that

$$1 + 3 + 5 + \dots + (2k - 1) + (2(k + 1) - 1) = (k + 1)^2 \dots (**)$$

(note that $(**)$ is $(*)$ with k replaced by $k + 1$)

We are assuming $(*)$ true, so add $2(k + 1) - 1$ to both sides of $(*)$ to get

$$\begin{aligned}
& 1 + 3 + 5 + \dots + (2k - 1) + (2(k + 1) - 1) \\
&= k^2 + 2(k + 1) - 1 \\
&= k^2 + 2k + 1 \\
&= (k + 1)^2
\end{aligned}$$

Thus $(**)$ true ie $S(k + 1)$ true.

Therefore $S(n)$ true for all n by principle of induction ie.

$$1 + 3 + 5 + \dots + (2n - 1) = n^2 \text{ for all } n.$$

Example 2

Prove that $1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{1}{4}n^2(n + 1)^2$ for each natural number n .

Solution Let $S(n)$ be the statement that

$$1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{1}{4}n^2(n + 1)^2.$$

Step 1 Check $S(1)$ true.

With $n = 1$, LHS = $1^3 = 1$, and RHS = $\frac{1}{4}1^2(1 + 1)^2 = \frac{1}{4} \cdot 4 = 1$. So LHS = RHS and $S(1)$ is true.

Step 2 Assume $S(k)$ is true ie that

$$1^3 + 2^3 + 3^3 + \dots + k^3 = \frac{1}{4}k^2(k + 1)^2 \dots (*).$$

We have to *prove* that $S(k + 1)$ is true ie that

$$1^3 + 2^3 + 3^3 + \dots + k^3 + (k + 1)^3 = \frac{1}{4}(k + 1)^2((k + 1) + 1)^2.$$

To establish this, add $(k + 1)^3$ to both sides of $(*)$. Then

$$\begin{aligned}
& 1^3 + 2^3 + 3^3 + \dots + k^3 + (k + 1)^3 = \frac{1}{4}k^2(k + 1)^2 + (k + 1)^3 \\
&= (k + 1)^2\left(\frac{1}{4}k^2 + (k + 1)\right)
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{4}(k+1)^2(k^2 + 4(k+1)) \\
&= \frac{1}{4}(k+1)^2(k^2 + 4k + 4) \\
&= \frac{1}{4}(k+1)^2(k+2)^2 \\
&= \frac{1}{4}(k+1)^2((k+1)+1)^2.
\end{aligned}$$

Hence $S(k+1)$ is true. Therefore it is true that $1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{1}{4}n^2(n+1)^2$ for all n by the principle of induction.

Example 3

Show that $n^3 + 2n$ is divisible by 3 for every n .

Solution. Let $S(n)$ be the statement that “ $n^3 + 2n$ is divisible by 3”.

Step 1. Check $S(1)$ is true. With $n = 1$, $n^3 + 2n = 1 + 2 \cdot 1 = 3$ which is divisible by 3 so that $S(1)$ is true.

Step 2 Assume $S(k)$ true ie that $k^3 + 2k$ is divisible by 3.

We must *prove* that $S(k+1)$ is true ie that $(k+1)^3 + 2(k+1)$ is divisible by 3. Now

$$\begin{aligned}
(k+1)^3 + 2(k+1) &= k^3 + 3k^2 + 3k + 1 + 2k + 2 \\
&= k^3 + 3k^2 + 5k + 3 \\
&= (k^3 + 2k) + 3k^2 + 3k + 3 \\
&= (k^3 + 2k) + 3(k^2 + k + 1).
\end{aligned}$$

Since $S(k)$ is assumed true, it follows that $(k^3 + 2k)$ is divisible by 3. Clearly $3(k^2 + k + 1)$ is divisible by 3. Therefore $(k^3 + 2k) + 3(k^2 + k + 1)$ is divisible by 3. Thus, $S(k+1)$

is true. Hence the statement “ $n^3 + 2n$ is divisible by 3 for every n ” is true.

Exercise Induction can be used to show that if X is a set with n elements, then \mathcal{P} has 2^n elements. Read the proof on Page 65 of the Textbook.

FUNCTIONS (Total and Partial).

See pp 125-131 of Textbook.

1.12 Definition Let A and B be sets and let $U \subseteq A$. A *partial function* or *partial mapping* $f : A \rightarrow B$ with *domain* U and *codomain* or *range* B is a rule which assigns to each $a \in U$ a unique element $b \in B$. We say f *maps* A *into* B .

Diagrammatically, we can represent this by:-

So f is thought of as some mechanism (perhaps an algorithm or program) which takes elements of U to elements of B .

Terminology and notation.

- (i) We write $U = \text{dom}(f)$ for the domain of f .
- (ii) If $\text{dom}(f)$ is all of A ie if $U = A$, we call f a total function.

Convention

We often use the term “function” to mean total function, and use the term “partial function” when we do mean a partial function ie when $\text{dom}(f) \subset A$.

- (iii) We say f is undefined at values $a \in A \setminus U$.

Note Partial functions f frequently arise in computing because programs which compute them may fail to halt at some values. At such values, f is undefined.

- (iv) If $b \in B$ is the unique correspondent of $a \in U$, we write $b = f(a)$ and call b the image of a under f .

Examples.

1) Let $A = \{a, b, c\}$ and $B = \{1, 2\}$. Then the assignment $a \mapsto 1, b \mapsto 1, c \mapsto 2$ defines a total function $f : A \rightarrow B$.

So does the assignment $a \mapsto 1, b \mapsto 1, c \mapsto 1$ and this function is called the constant function with value 1. ($f(x) = 1$ for all $x \in A$. This can be generalized to arbitrary sets A, B : define $f(a) = b_0$ for all $a \in A$, where $b_0 \in B$ is fixed; then f is called the constant function with value b_0 and is total.)

The assignment $a \mapsto 1, a \mapsto 2, b \mapsto 1, c \mapsto 1$ does not de-

fine a function at all because a is being sent to two different images ie the “unique” part of the definition is violated.

2) The assignment $(m, n) \mapsto m + n$ defines a total function $N \times N \rightarrow N$; the assignment $(m, n) \mapsto \frac{m}{n}$ defines a partial function $N \times N \rightarrow Q$ – the function is undefined at $n = 0$ and so its domain is $N \times (N \setminus \{0\})$.

3) (i) The assignment $n \mapsto 2n$ defines a total function $f : N \rightarrow N$ where $f(n) = 2n$.

(ii) The assignment $n \mapsto n^2$ defines a total function $g : N \rightarrow N$ where $g(n) = n^2$.

4) Take a natural number $m, m \neq 0$, and define $n \bmod m$ (read “ n modulo m ”) to be the remainder when n is divided by m .

Eg. take $m = 11$. Then $0 \bmod 11 = 0, 1 \bmod 11 = 1, \dots, 10 \bmod 11 = 10, 11 \bmod 11 = 0, 12 \bmod 11 = 1, 13 \bmod 11 = 2, \dots, 21 \bmod 11 = 10, 22 \bmod 11 = 0, 23 \bmod 11 = 1, \dots$ etc.

The assignment $n \mapsto n \bmod 11$ defines a function $h : N \rightarrow \{0, 1, 2, \dots, 10\}$ called a hash function - it is total - where $h(n) = n \bmod 11$. See page 127 for a discussion of hash functions in storing data in memory cells in a computer.

5) Let L_A denote the set of lists over a set A .

(i) The assignment $L \mapsto$ sorted version of L defines a function $\text{Sort}: L_A \rightarrow L_A$

eg $\text{Sort}([1, 2, 1, 5, 4]) = [1, 1, 2, 4, 5]$

$\text{Sort}([\]) = [\]$

$\text{Sort}([1, 2, 3]) = [1, 2, 3]$.

The main problem of course is to devise efficient algorithms or programs to compute the function sort (sort is a total function).

(ii) The assignment

$[\] \mapsto [\]$

$[h \mid T] \mapsto T$

defines a total function $L_A \rightarrow L_A$.

(iii) The assignment $[h \mid T] \mapsto h$ defines a partial function $L_A \rightarrow A$ – it is undefined on the empty list, so its domain is $L_A \setminus \{[\]\}$.

6) Let $a, b \in R$. Then the assignment $x \mapsto ax + b$ determines a total function $R \rightarrow R$.

7) Floor and ceiling functions.

Let $x \in R$ be a real number. Define $\lfloor x \rfloor$ to be the greatest integer n s.t. $n \leq x$. We call $\lfloor x \rfloor$ the floor of x . Define $\lceil x \rceil$ to be the smallest integer n s.t. $x \leq n$. We call $\lceil x \rceil$ the ceiling of x . Both operations define total functions $R \rightarrow Z$.

Note

remember that numbers get smaller as we go to the left eg. $-4 < -3$ even though the magnitude 4 is greater than the magnitude 3.

$$\text{Eg. } \lfloor 8.3 \rfloor = 8, \quad \lceil 8.3 \rceil = 9$$

$$\lfloor -8.3 \rfloor = -9, \quad \lceil -8.3 \rceil = -8$$

$$\lfloor 6 \rfloor = 6, \quad \lceil 6 \rceil = 6$$

$$\lfloor -6 \rfloor = -6, \quad \lceil -6 \rceil = -6$$

etc. See p. 128 of Text.

FURTHER DEFINITIONS (for a function $f : A \rightarrow B$)

1.13 Definition Suppose $f : A \rightarrow B$ is a function (either total or partial). The image set of f , written $Im(f)$ or $f(U)$, is the set $\{b \in B; b = f(a) \text{ for some } a \in U\} \subseteq B$. If $Im(f) = B$, f is called onto or surjective.

Note that the definition of function, 1.12, does not say that every $b \in B$ must be the image of some $a \in A$. The condition that this holds is exactly the condition of surjectivity. Always $f : A \rightarrow f(U)$ is surjective of course.

Examples

1) $f : N \rightarrow N$ defined by $f(n) = 2n$ is not surjective eg there is no n s.t. $f(n) = 3$.

2) $f : N \rightarrow N$ defined by $f(n) = n^2$ is also not surjective eg 2 is never $f(n)$ for any n .

3) The hash function $h : N \rightarrow \{0, 1, \dots, 10\}$, $h(n) = n \bmod 11$, is surjective: let $b \in \{0, 1, 2, \dots, 10\}$ then $h(b) = b \bmod 11 = b$. So $h(N) = \{0, 1, 2, \dots, 10\}$.

4) $f : R \rightarrow R$ by $f(x) = ax + b$ is surjective provided $a \neq 0$ (otherwise it is not). Let $y \in R$ be any element of the codomain. Then $f\left(\frac{y-b}{a}\right) = a\left(\frac{y-b}{a}\right) + b = y - b + b = y$ ie there is $x \in R$ s.t. $f(x) = y$.

1.14 Definition Suppose $f : A \rightarrow B$ is a function (either total or partial again). We call f *1-1 (one-to-one)* or *injective* if, whenever $a_1, a_2 \in \text{dom}(f)$ and $f(a_1) = f(a_2)$, then $a_1 = a_2$. This condition can be stated equivalently in terms of negation as: whenever $a_1, a_2 \in \text{dom}(f)$ and $a_1 \neq a_2$, we have $f(a_1) \neq f(a_2)$.

Exercise. Show the equivalence of the formulations in this definition.

For a general function $f : A \rightarrow B$ we can have

ie distinct elements a_1, a_2 can be mapped to the same element $b \in B$ - the definition 1.12 does not prevent this happening. The condition that f be injective does prevent it. Thus, if f is injective, then we have

Examples

- 1) $f : N \rightarrow N$ defined by $f(n) = 2n$ is injective because if $f(n_1) = f(n_2)$, then $2n_1 = 2n_2$ and therefore $n_1 = n_2$.
- 2) $f : Z \rightarrow N$ defined by $f(n) = n^2$ is not injective because $1 \neq -1$ and yet $f(1) = 1^2 = 1 = (-1)^2 = f(-1)$.
- 3) The hash function $h(n) = n \bmod 11$ is not injective because $h(0) = 0 = h(11)$ yet $0 \neq 11$.
- 4) The function $f : R \rightarrow R$ defined by $f(x) = ax + b$ is injec-

tive if $a \neq 0$ because $f(x_1) = f(x_2) \Rightarrow ax_1 + b = ax_2 + b \Rightarrow ax_1 = ax_2 \Rightarrow x_1 = x_2$ if $a \neq 0$.

1.15 Definition Suppose $f : A \rightarrow B$ is a total function. We call f bijjective if it is both injective and surjective.

This definition can be extended to partial functions but we omit this.

Examples

1) $f : N \rightarrow N$ where $f(n) = 2n$ is not bijective (not surjective).

2) $f : N \rightarrow N$ where $f(n) = n^2$ is not bijective (not surjective).

3) $h : N \rightarrow \{0, 1, 2, \dots, 10\}$ where $h(n) = n \bmod 11$ is not bijective (not injective).

4) $f : \{a, b, c\} \rightarrow \{1, 2, 3\}$ where $f(a) = 1, f(b) = 2, f(c) = 3$ is bijective.

5) $f : R \rightarrow R$ where $f(x) = ax + b$ ($a \neq 0$) is bijective.

6) Let $R^{>0} = \{x \in R; x > 0\}$ = set of all positive real numbers. Then $f : R^{>0} \rightarrow R$ where $f(x) = \log_e x$ is bijective.

Composition of Functions. (see p 130 of Textbook).

Suppose $f : N \rightarrow N$ is defined by $f(n) = n^2$ for all $n \in N$ and $g : N \rightarrow N$ is defined by $g(n) = 2^n$ for all $n \in N$ – both are total functions.

Consider the following “process”:

For any $n \in N$, $f(n) = n^2 \in N = \text{dom}(g)$ and therefore $g(f(n)) = g(n^2) = 2^{n^2}$ is defined. Since f is a function, $f(n) = n^2$ is unique. Since g is a function $g(n^2) = 2^{n^2}$ is also unique. So the overall effect is a single assignment $n \mapsto 2^{n^2}$ for which the correspondent 2^{n^2} is *unique* for *each* n – it therefore defines a function $N \rightarrow N$. This function uses both f and g in its definition and is written $g \circ f$ (note the order). Thus, $g \circ f : N \rightarrow N$ is defined by $(g \circ f)(n) = g(f(n))$. We can generalize this:

1.16 Definition Let $f : A \rightarrow B$ and $g : B \rightarrow C$ be partial functions (which includes the possibility of their being total), and suppose $\text{Im}(f)$ and $\text{dom}(g)$ satisfy the relationship $\text{Im}(f) \subseteq \text{dom}(g)$. Then we can define a function $g \circ f : A \rightarrow C$, called the *composition* of f and g – note the order again – by $(g \circ f)(a) = g(f(a))$ for all $a \in \text{dom}(f)$.

Examples

1) If $f : N \rightarrow N$, $f(n) = n^2$, and $g : N \rightarrow N$, $g(n) = 2^n$, then $(g \circ f)(n) = 2^{n^2}$ and $g \circ f : N \rightarrow N$. Note that f and g are total functions and so the condition $Im(f) \subseteq dom(g)$ is satisfied.

2) Similarly, $Im(g) \subseteq dom(f)$ and we can therefore define $f \circ g : N \rightarrow N$ by $(f \circ g)(n) = f(2^n) = (2^n)^2 = 2^{2n}$ for all $n \in N$.

Since $2^{2n} \neq 2^{n^2}$ in general, we see that $g \circ f \neq f \circ g$ in general.

Note.

Our definition of equality of functions is:

1.17 Definition Two functions f and g are *equal*, written $f = g$, iff:

- (i) $dom(f) = dom(g)$,
- (ii) $codomain(f) = codomain(g)$, and
- (iii) $f(x) = g(x)$ for all $x \in dom(f)$ ($= dom(g)$).

Examples continued

3) Take f and g as in Examples 1 and 2 above.

Then $f \circ f$ can be defined since $Im(f) \subseteq N = dom(f)$ and we have $(f \circ f)(n) = f(f(n)) = f(n^2) = f(m)$ where we put $m = n^2$. But $f(m) = m^2 = (n^2)^2 = n^4$. Therefore $(f \circ f)(n) = f(f(n)) = (n^2)^2 = n^4$ for all $n \in N$.

Similarly $g \circ g$ can be defined since $Im(g) \subseteq N = dom(g)$, and we have $(g \circ g)(n) = g(g(n)) = g(2^n) = g(m)$ where we put $m = 2^n$. But $g(m) = 2^m = 2^{2^n}$. Therefore $(g \circ g)(n) = g(g(n)) = 2^{2^n}$ for all $n \in N$.

4) Define $f : R \rightarrow R$ by $f(x) = +\sqrt{x}$ and define $g : R \rightarrow R$ by $g(x) = -x$. Then g is a total function ie $g(x) = -x$ for all $x \in R$ ie $dom(g) = R$. But f is partial since $dom(f) = R^{\geq 0} = \{x \in R; x \geq 0\}$ (\sqrt{x} is a complex number if $x < 0$). In this case we have $Im(g) = R \not\subseteq dom(f)$ and so we cannot define $f \circ g$ here. Eg. $(f \circ g)(+1) = f(g(+1)) = f(-1) = \sqrt{-1}$ which is not defined as a real number. However, $Im(f) = R^{\geq 0} \subseteq R = dom(g)$ so $g \circ f$ can be defined and $(g \circ f)(x) = g(f(x)) = g(+\sqrt{x}) = -\sqrt{x}$ for all $x \in dom(f) = R^{\geq 0}$.

5) Suppose $f : N \rightarrow N$ is defined by $f(n) = n^2 - 2n - 3$. Find $(f \circ f)(n)$.

Solution First note that f is total and so $(f \circ f)(n)$ is defined for all n . Put $m = f(n)$. Then $(f \circ f)(n) = f(f(n)) = f(m) = m^2 - 2m - 3$
 $= (f(n))^2 - 2f(n) - 3$
 $= (n^2 - 2n - 3)^2 - 2(n^2 - 2n - 3) - 3$
 $= (n^2 - 2n - 3)(n^2 - 2n - 3) - 2n^2 + 4n + 6 - 3$
 $= n^4 - 4n^3 - 2n^2 + 12n + 9 - 2n^2 + 4n + 3$
 $= n^4 - 4n^3 - 4n^2 + 16n + 12.$

Invertible Functions. (See p 130 of Textbook).

Let A be any set. We define the identity function on A , written I_A or i_A , by $I_A : A \rightarrow A$ where $I_A(a) = a$ for all $a \in A$.

1.18 Definition A total function $f : A \rightarrow B$ is called *invertible* if there is a total function $g : B \rightarrow A$ s.t.

(i) $g \circ f = I_A$ ($A \xrightarrow{f} B \xrightarrow{g} A$ so $g \circ f : A \rightarrow A$).

(ii) $f \circ g = I_B$ ($B \xrightarrow{g} A \xrightarrow{f} B$ so $f \circ g : B \rightarrow B$)

Equivalently (i) and (ii) can be stated

(i)' $(g \circ f)(a) = a$ for all $a \in A$.

(ii)' $(f \circ g)(b) = b$ for all $b \in B$.

If such a function g exists, then it is called an *inverse* of f and written $g = f^{-1}$.

Note (1) We will see shortly that if an inverse g of f exists,

then it is unique and hence we will in future refer to the inverse $g = f^{-1}$ of f .

2) Suppose $g = f^{-1}$ is the inverse of f ie $g \circ f = I_A$ and $f \circ g = I_B$. Then $f \circ g = I_B$ and $g \circ f = I_A$. Therefore f is the inverse g^{-1} of g . So f and g are inverses of each other.

3) The Definition 1.18 can be extended to partial functions, but we omit this here.

1.19 Theorem A function $f : A \rightarrow B$ (a total function) is invertible iff it is bijective.

Proof

Suppose $f : A \rightarrow B$ is invertible and let $g : B \rightarrow A$ be an inverse of f . We show f is bijective.

To show f is 1-1

Let $a_1, a_2 \in A$ and suppose $f(a_1) = f(a_2)$. Then $g(f(a_1)) = (g \circ f)(a_1) = I_A(a_1) = a_1$ and $g(f(a_2)) = (g \circ f)(a_2) = I_A(a_2) = a_2$ using the fact that g is an inverse of f . Since $f(a_1) = f(a_2)$, we get immediately that $g(f(a_1)) = g(f(a_2))$ because g is a (well-defined) function. Therefore $a_1 = a_2$ and hence f is 1-1.

To show f is onto

Let $b \in B$ be any element of B , and let $a = g(b) \in A$. Then $f(a) = f(g(b)) = (f \circ g)(b) = I_B(b) = b$, since g is

an inverse of f . Thus, there exists $a \in A$ s.t. $f(a) = b$ and hence f is onto. Therefore f is bijective.

Conversely, suppose $f : A \rightarrow B$ is bijective. We must construct an inverse $g : B \rightarrow A$ of f . Let $b \in B$ be arbitrary, and let $a \in A$ be an element s.t. $f(a) = b$; such $a \in A$ exists because f is onto. Furthermore, a is unique with the property $f(a) = b$ since f is 1-1 ie if $f(a_1) = b$ and $f(a_2) = b$, then $f(a_1) = f(a_2)$. Therefore, $a_1 = a_2$ since f is 1-1. Given $b \in B$, denote the corresponding a by $g(b)$. Thus, for *each* $b \in B$ there is a *unique* correspondent $a = g(b) \in A$. Therefore we have a function $g : B \rightarrow A$ which is defined by $g(b) = \text{unique element } a \in A \text{ s.t. } f(a) (= f(g(b))) = b$.

Claim g is an inverse of f .

To see this: straight from the definition of g we have

$$(f \circ g)(b) = f(g(b)) = f(a) = b \text{ for all } b \in B \text{ ie } f \circ g = I_B.$$

Also

$$(g \circ f)(a) = g(f(a)) = g(b) = a \text{ for all } a \in A \text{ ie } g \circ f = I_A.$$

Therefore, g is an inverse of f . ■

Note. This theorem is important because it gives an “external” check for invertibility ie check whether or not the function is bijective.

1.20 Corollary If $f : A \rightarrow B$ has an inverse g , then g is unique.

Proof Suppose g_1 and g_2 are inverses of f , and let $b \in B$ be arbitrary. Since g_1 and g_2 are both inverses of f , we have $f(g_1(b)) = (f \circ g_1)(b) = I_B(b) = b$, and $f(g_2(b)) = (f \circ g_2)(b) = I_B(b) = b$.

Therefore, we have $f(g_1(b)) = b = f(g_2(b))$. But f is bijective (and hence 1-1) by Th^m 1.19. Hence $g_1(b) = g_2(b)$. Since $b \in B$ is arbitrary we have $g_1 = g_2$. ■

1.21 Corollary If $f : A \rightarrow B$ is invertible, then $(f^{-1})^{-1} = f$ ie if $g = f^{-1}$, then $g^{-1} = (f^{-1})^{-1} = f$.

Proof Let $g = f^{-1}$. Then we have

$$g \circ f = I_A \quad \text{and} \quad f \circ g = I_B \quad \dots (1).$$

But

$g \circ (f^{-1})^{-1} = g \circ g^{-1} = I_A$ since g^{-1} is the inverse of g and $g = f^{-1}$. So we have $g \circ (f^{-1})^{-1} = I_A$.

Also $(f^{-1})^{-1} \circ g = g^{-1} \circ g = I_B$, again because g^{-1} is the inverse of g and $g = f^{-1}$. So we have $(f^{-1})^{-1} \circ g = I_B$.

Putting these together we get the following pair of equations

$$g \circ (f^{-1})^{-1} = I_A \quad \text{and} \quad (f^{-1})^{-1} \circ g = I_B \quad \dots (2).$$

and

$$g \circ f = I_A \quad \text{and} \quad f \circ g = I_B \quad \dots (1).$$

These say that both $(f^{-1})^{-1}$ and f are inverses of g . Therefore by uniqueness of inverses we have $(f^{-1})^{-1} = f$. ■

Examples of Inverse Functions.

1) Suppose $A = \{0, 1, 2\}$ and $B = \{a, b, c\}$. Define $f : A \rightarrow B$ and $g : B \rightarrow A$ by

$$f(0) = a \quad g(a) = 0$$

$$f(1) = b \quad g(b) = 1$$

$$f(2) = c \quad g(c) = 2$$

Then $g \circ f : A \rightarrow A$ is defined by

$$\left. \begin{array}{l} (g \circ f)(0) = g(f(0)) = g(a) = 0 \\ (g \circ f)(1) = g(f(1)) = g(b) = 1 \\ (g \circ f)(2) = g(f(2)) = g(c) = 2 \end{array} \right\} \Rightarrow g \circ f = I_A$$

Also $f \circ g : B \rightarrow B$ is defined by

$$\left. \begin{array}{l} (f \circ g)(a) = f(g(a)) = f(0) = a \\ (f \circ g)(b) = f(g(b)) = f(1) = b \\ (f \circ g)(c) = f(g(c)) = f(2) = c \end{array} \right\} \Rightarrow f \circ g = I_B$$

Therefore $g = f^{-1}$ ie f and g are inverses of each other.

2) Take $A = B = R$ and define $f : A \rightarrow B$ ie $f : R \rightarrow R$ by $f(x) = ax + b$, where $a \neq 0$ and b are real constants, and define $g : B \rightarrow A$ ie $g : R \rightarrow R$ by $g(y) = \frac{y - b}{a}$. Since f and g are both total and $A = B$, both $g \circ f$ and $f \circ g$ are defined (Exercise: check this). Therefore, we have

$$(g \circ f)(x) = g(f(x)) = g(ax + b) = \frac{ax + b - b}{a} = \frac{ax}{a} = x.$$

Hence $(g \circ f)(x) = x$ for all x and therefore $g \circ f = I_A$.

Also $(f \circ g)(y) = f(g(y)) = f\left(\frac{y - b}{a}\right) = a\left(\frac{y - b}{a}\right) + b = y - b + b = y$. Thus, $(f \circ g)(y) = y$ for all y and so $f \circ g = I_B$.

Therefore, $g = f^{-1}$, and f and g are inverses of each other.

3) Take $A = B = R^{\geq 0}$ and define $f : A \rightarrow B$ ie $f : R^{\geq 0} \rightarrow R^{\geq 0}$ by $f(x) = +\sqrt{x}$ and define $g : B \rightarrow A$ ie $g : R^{\geq 0} \rightarrow R^{\geq 0}$ by $g(x) = x^2$. Then $g \circ f$ and $f \circ g$ are both defined (Exercise: check this) and we have

$(g \circ f)(x) = g(f(x)) = g(+\sqrt{x}) = (+\sqrt{x})^2 = x$ for all x and so $g \circ f = I_A$. Also we have

$(f \circ g)(x) = f(g(x)) = f(x^2) = +\sqrt{x^2} = x$ for all x and so $f \circ g = I_B$.

Therefore, $g = f^{-1}$, and f and g are inverses of each other.

Earlier we saw an example of functions $f : N \rightarrow N$ and $g : N \rightarrow N$ ($f(n) = n^2, g(n) = 2^n$) s.t. $g \circ f : N \rightarrow N$ and $f \circ g : N \rightarrow N$ are both defined but $f \circ g \neq g \circ f$. Thus, the operation “ \circ ” is not commutative. However, “ \circ ” is associative:

1.22 Proposition Suppose $f : A \rightarrow B, g : B \rightarrow C$ and $h : C \rightarrow D$ are three total functions. Then $(h \circ g) \circ f = h \circ (g \circ f)$.

Proof First note that $h \circ g : B \rightarrow D$ is defined and $(h \circ g) \circ f : A \rightarrow D$ is defined. Also $g \circ f : A \rightarrow C$ is defined and $h \circ (g \circ f) : A \rightarrow D$ is also defined (Exercise: check these claims). Thus we have to show that

$$((h \circ g) \circ f)(a) = (h \circ (g \circ f))(a) \text{ for all } a \in A.$$

Consider the LHS $((h \circ g) \circ f)(a)$. Put $\theta = h \circ g$. Then $((h \circ g) \circ f)(a) = (\theta \circ f)(a) = \theta(f(a)) = (h \circ g)(f(a)) = h(g(f(a)))$; so LHS = $h(g(f(a)))$.

Consider the RHS $(h \circ (g \circ f))(a)$. Put $\phi = g \circ f$. Then $(h \circ (g \circ f))(a) = (h \circ \phi)(a) = h(\phi(a)) = h((g \circ f)(a)) = h(g(f(a)))$; so RHS = $h(g(f(a)))$. Thus, LHS = RHS and the result is established. ■

Exercise. Show that this result holds for partial functions $f : A \rightarrow B$, $g : B \rightarrow C$ and $h : C \rightarrow D$ provided that $Im(f) \subseteq dom(g)$ and $Im(g) \subseteq dom(h)$.

2 ELEMENTARY COMPUTABILITY

There is no reference to the Text for this material.

2.1 THE UNLIMITED REGISTER MACHINE (URM)

We consider an abstract computer called the *unlimited register machine* (URM). It is one of many abstract models of computation which have been considered: the Turing machine (the most popular model), random access machines, re-write systems, λ -definability, logic programming systems etc. Abstract models are used to study theoretical questions such as:

- (1) What is the smallest set of operations which can be used to carry out any computation?
- (2) Can we characterize the class of all functions computable by any computer and (hence) can we characterize the class of all problems solvable by any computer?
- (3) What are the theoretical limits to computation; are there any problems not solvable by any computer? Etc.

Let $\mathbb{N} = \{0, 1, 2, \dots\}$ denote the set of natural numbers including zero. The URM consists of an infinite sequence of registers labelled $R_1, R_2, R_3 \dots$ each of which, at any moment in time, contains an element of \mathbb{N} . We denote the

number contained in R_n by r_n . (In practice, only finitely many of the r_n are non-zero):

R_1	R_2	R_3	R_4	R_5	R_6	\dots				
r_1	r_2	r_3	r_4	r_5	r_6					

The contents of the registers may be altered in response to any one of four instructions which the machine recognizes. A finite *list* of instructions constitutes a program (so each instruction in a program is numbered and the order matters).

THE INSTRUCTIONS

- *Zero the n th register, $Z(n)$.* The URM's response is to replace r_n by 0 i.e. $0 \rightarrow R_n$ (0 is placed in R_n) or $r_n := 0$ (“ r_n becomes 0” i.e. $:=$ indicates an assignment statement).
- *Successor of the n th register, $S(n)$.* The URM's response is to add 1 to r_n i.e. $r_n := r_n + 1$.
- *Transfer instruction, $T(m, n)$.* The URM's response is to replace r_n by r_m i.e. $r_n := r_m$. This instruction transfers the contents of R_m to R_n – note that the content of R_m does not change.
- *The jump instruction, $J(m, n, q)$.* The URM's response is as follows:

if $r_m = r_n$, then jump to q th instruction if there is one, otherwise stop;

if $r_m \neq r_n$, then carry on to arithmetically-next instruction if there is one, otherwise stop.

2.2 Example Suppose the URM is in the configuration:

R_1	R_2	R_3	R_4	R_5	R_6	\dots				
9	6	5	23	7	0					

(i) Apply the instruction $Z(3)$, then the configuration becomes:

R_1	R_2	R_3	R_4	R_5	R_6	\dots				
9	6	0	23	7	0					

(ii) Apply the instruction $S(4)$ to this configuration and it becomes:

R_1	R_2	R_3	R_4	R_5	R_6	\dots				
9	6	0	24	7	0					

(iii) Apply the instruction $T(4, 6)$ to this last configuration and it becomes:

R_1	R_2	R_3	R_4	R_5	R_6	\dots				
9	6	0	24	7	24					

■

2.3 Note (1) This machine is due to Shepherdson and Sturgis (“Computability of Recursive Functions”). It has exactly the same computational power as the Turing machine but, unlike the Turing machine, the URM computes different functions without having to re-specify the machine in any way. In this respect it operates in much the same way as modern day computers.

(2) Suppose a program P has s instructions and one of them, I_k say, is $I_k = J(m, n, q)$ where $q > s$. Then, whenever the URM is in a configuration with $r_m = r_n$ when I_k is encountered, it will jump to a non-existent instruction ($q > s$) and will therefore stop.

(3) An instruction of type $I_k = J(1, 1, q)$ always causes the machine to jump to instruction I_q . Therefore, if $q < k$ we can create a loop; if $q > s$ (= number of instructions in P), the machine stops.

The facts (2) and (3) above are important tools in programming the URM.

COMPUTATIONS

To perform a computation, the URM is provided with a program $P = I_1, I_2, \dots, I_s$ and an initial configuration

R_1	R_2	R_3	R_4	R_5	R_6	\dots					
a_1	a_2	a_3	a_4	a_5	a_6						

i.e. initial values a_n of r_n are given for each $n \in \mathbb{N}$, where each $a_n \in \mathbb{N}$.

MODE OF OPERATION

We define this inductively as follows: Start by obeying I_1 (think of this as the basis step). At any future stage in the computation, suppose that the URM is obeying instruction I_k . On completion of I_k , the machine proceeds to *next instruction* defined as follows:

if I_k not a jump instruction, then *next instruction* = I_{k+1} = arithmetically-next instruction;

if I_k is a jump instruction $J(m, n, q)$, then

$$\text{next instruction} = \begin{cases} I_q & \text{if } r_m = r_n, \\ I_{k+1} & \text{otherwise,} \end{cases}$$

where r_m and r_n are the current contents of R_m and R_n resp. on completion of instruction I_k .

The machine proceeds thus as long as possible. The computation stops or halts when and only when there is no next instruction i.e. when after performing an instruction I_k , next instruction is I_v with $v > s$. This happens in one of the fol-

lowing ways:

- (i) if $k = s$ i.e. last instruction in P has been obeyed and I_s is not a jump instruction;
- (ii) if $I_k = J(m, n, q)$, $r_m = r_n$ and $q > s$;
- (iii) if $I_k = J(m, n, q)$, $r_m \neq r_n$ and $k = s$.

Thus the computation stops or halts after the instruction I_k , and the final configuration is the configuration:

R_1	R_2	R_3	R_4	R_5	R_6	\dots				
r_1	r_2	r_3	r_4	r_5	r_6					

in which the value of r_n , for each $n \in \mathbb{N}$, is the content of register R_n at this stage.

2.4 Example Consider the following program P :

- $I_1 = J(1, 2, 6)$
- $I_2 = S(2)$
- $I_3 = S(3)$
- $I_4 = J(1, 2, 6)$
- $I_5 = J(1, 182)$
- $I_6 = T(3, 1)$

and the computation which results with the initial configuration

R_1	R_2	R_3	R_4	R_5	R_6	\dots				
9	7	0	0	0	0					

in which every register beyond the second contains 0.

The computation is displayed as follows:

Initial Configuration:

9	7	0	0	0	\dots					
---	---	---	---	---	---------	--	--	--	--	--

next instruction = I_1 (the machine starts):

9	7	0	0	0	\dots					
---	---	---	---	---	---------	--	--	--	--	--

next instruction = I_2 (since $r_1 \neq r_2$):

9	8	0	0	0	\dots					
---	---	---	---	---	---------	--	--	--	--	--

next instruction = I_3 :

9	8	1	0	0	\dots					
---	---	---	---	---	---------	--	--	--	--	--

next instruction = I_4 :

9	8	1	0	0	\dots					
---	---	---	---	---	---------	--	--	--	--	--

next instruction = I_5 (since $r_1 \neq r_2$):

9	8	1	0	0	...					
---	---	---	---	---	-----	--	--	--	--	--

next instruction = I_2 (since $r_1 = r_1$):

9	9	1	0	0	...					
---	---	---	---	---	-----	--	--	--	--	--

next instruction = I_3 :

9	9	2	0	0	...					
---	---	---	---	---	-----	--	--	--	--	--

next instruction = I_4 :

9	9	2	0	0	...					
---	---	---	---	---	-----	--	--	--	--	--

next instruction = I_6 (since $r_1 = r_2$).

Stop, there is no next instruction (I_6 is the last instruction in P and is not a jump instruction). Final configuration is:

2	9	2	0	0	...					
---	---	---	---	---	-----	--	--	--	--	--

Exercise

Carry out the computation under the program of Example 2.4 with initial configuration:

8	4	2	0	0	0	...				
---	---	---	---	---	---	-----	--	--	--	--

Flow charts

The essence of a program and the progress of a computation under it is often conveniently described informally by a flow chart. The following is a flow chart for the computation of the Example 2.4.

Typical configuration

x	y	z	0	0	0	...				
-----	-----	-----	---	---	---	-----	--	--	--	--

After k cycles round the loop

x	$y+k$	$z+k$	0	0	0	...
---	-------	-------	---	---	---	-----

If $x = y + k$, stop. Then

$z+k$	$y+k$	$z+k$	0	0	...
-------	-------	-------	---	---	-----

is the final configuration.

Note that “tests” e.g. $r_1 = r_2?$ are placed in diamond shaped boxes, actions are placed in rectangular boxes. So, when solving a problem (ie writing a program) it sometimes helps to:

- 1) Draw a flow chart;
- 2) translate the flowchart into a program.

Some computations never stop. For example the program

$$I_1 = S(1)$$

$$I_2 = J(1, 1, 1)$$

never stops on any initial configuration:

a_1	a_2	a_3	a_4	\dots						
-------	-------	-------	-------	---------	--	--	--	--	--	--

next instruction = I_1 (the machine starts):

$a_1 + 1$	a_2	a_3	a_4	\dots
-----------	-------	-------	-------	---------

next instruction = I_2 (since I_1 is not a jump instruction):

$a_1 + 1$	a_2	a_3	a_4	\dots
-----------	-------	-------	-------	---------

next instruction = I_1 ($J(1, 1, 1)$ always fires):

$a_1 + 2$	a_2	a_3	a_4	\dots
-----------	-------	-------	-------	---------

next instruction = I_2 :

$a_1 + 2$	a_2	a_3	a_4	\dots
-----------	-------	-------	-------	---------

next instruction = I_1 :

$a_1 + 3$	a_2	a_3	a_4	\dots
-----------	-------	-------	-------	---------

next instruction = I_2 , etc.

The flow chart is

Exercise

Show that the computation under the program in Example 2.4 never halts with initial configuration

2	3	0	0	0	\dots					
---	---	---	---	---	---------	--	--	--	--	--

Remark It is for this sort of reason that we consider partial functions in computer science – on some inputs a program halts and on others it does not. But we don't know in advance on which values it does halt.

2.5 A Famous Problem

The halting problem - Alan Turing. An obvious question is the following “Is there an algorithm/procedure/program A which accepts as input arbitrary programs P (in some language) together with their inputs \underline{x} and tells us whether or not $P(\underline{x})$ ie P running on \underline{x} halts?” Thus, we input P and \underline{x} to A ie $A(P, \underline{x})$. Then A says yes if it is the case that $P(\underline{x})$ will eventually halt, and says no if it is the case that $P(\underline{x})$ will never halt. The answer is that no such algorithm can exist (see later) ie this problem is undecidable.

2.6 NOTATION

Let a_1, a_2, a_3, \dots be an infinite sequence of elements of \mathbb{N} , and let P be a program. We write:

(i) $P(a_1, a_2, a_3, \dots)$ for the computation under P with initial configuration

a_1	a_2	a_3	\dots							
-------	-------	-------	---------	--	--	--	--	--	--	--

(ii) $P(a_1, a_2, a_3, \dots) \downarrow$ to mean that the computation $P(a_1, a_2, a_3, \dots)$ eventually halts (ie converges).

(iii) $P(a_1, a_2, a_3, \dots) \uparrow$ to mean that the computation $P(a_1, a_2, a_3, \dots)$ never halts (ie diverges).

In most initial configurations we will consider, all but finitely many of the a_i are zero ie for some n , $a_i = 0$ for all $i > n$.

So the usual initial configuration is

a_1	a_2	a_3	\dots	a_n	0	0	0	\dots		
-------	-------	-------	---------	-------	---	---	---	---------	--	--

and we write

(iv) $P(a_1, a_2, \dots, a_n)$ for the computation $P(a_1, a_2, a_3, \dots, a_n, 0, 0, 0, \dots)$.

2.7 URM COMPUTABLE FUNCTIONS

For a positive natural number $n \in \mathbb{N}$ (so $n = 1, 2, 3, \dots$ etc) we form the Cartesian product \mathbb{N}^n of \mathbb{N} with itself n times.

Thus $\mathbb{N}^n = \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \dots \times \mathbb{N} =$

$\{(x_1, x_2, \dots, x_n); x_1 \in \mathbb{N}, x_2 \in \mathbb{N}, \dots, x_n \in \mathbb{N}\}$.

We call $(x_1, x_2, \dots, x_n) \in \mathbb{N}^n$ an n-tuple and often write $\underline{x} = (x_1, x_2, \dots, x_n)$ or $\underline{a} = (a_1, a_2, \dots, a_n)$ for typical elements of \mathbb{N}^n .

Eg. $\mathbb{N}^1 = \mathbb{N} = \{x; x \in \mathbb{N}\}$

$$\mathbb{N}^2 = \mathbb{N} \times \mathbb{N} = \{(x_1, x_2); x_1 \in \mathbb{N}, x_2 \in \mathbb{N}\}$$

So $(0, 0), (0, 1), (2, 0), (33, 100) \dots$ etc. are some elements of \mathbb{N}^2 .

$$\mathbb{N}^3 = \mathbb{N} \times \mathbb{N} \times \mathbb{N} = \{(x_1, x_2, x_3); x_1 \in \mathbb{N}, x_2 \in \mathbb{N}, x_3 \in \mathbb{N}\}.$$

So $(0, 0, 0), (0, 1, 0), (2, 0, 20) \dots$ etc. are elements of \mathbb{N}^3 .

We consider n -ary partial functions f ie functions f of arity n ie functions $f : \mathbb{N}^n \rightarrow \mathbb{N}$ (taking values in \mathbb{N}) with $dom(f) \subseteq \mathbb{N}^n$. (“ n -ary” or “arity n ” mean that f takes n places of argument).

Eg. (i) $f : \mathbb{N} \rightarrow \mathbb{N}$ by $f(x) = 2x$ is a total *unary* function ie has arity 1

(ii) $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ by $f(x, y) = x + y$ is a total *binary* function ie has arity 2

(iii) $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ defined by

$$f(a_1, a_2) = \begin{cases} a_1 - a_2 & \text{if } a_1 \geq a_2, \\ \text{undefined} & \text{otherwise} \end{cases}$$

is a partial function of arity 2 – its domain is the set

$$dom(f) = \{(a_1, a_2) \in \mathbb{N}^2; a_1 \geq a_2\} \subset \mathbb{N}^2.$$

Note that we use $f(a_1, a_2)$ or $f(x, y)$ or $f(x_1, y_1)$ etc. to denote typical values of f .

(iv) $f : \mathbb{N}^3 \rightarrow \mathbb{N}$ defined by

$$f(a_1, a_2, a_3) = \begin{cases} \frac{a_1+a_2}{a_3} & \text{if } a_3 \text{ exactly divides } a_1 + a_2, \\ \text{undefined} & \text{otherwise} \end{cases}$$

is a ternary partial function ie has arity 3 and $dom(f) \subset \mathbb{N}^3$.

Exercise What is the domain of f ?

FURTHER NOTATION If f and g are n -ary partial functions, we write $f \simeq g$ or $f(\underline{x}) \simeq g(\underline{x})$ to mean that for any $\underline{x} = (x_1, x_2, \dots, x_n) \in \mathbb{N}^n$, either both $f(\underline{x})$ and $g(\underline{x})$ are undefined or $f(\underline{x})$ and $g(\underline{x})$ are both defined and are equal.

Thus, $f \simeq g$ iff

(i) $dom(f) = dom(g) = D$ – say, and

(ii) $f(\underline{x}) = g(\underline{x})$ for all $\underline{x} \in D$.

Suppose $f : \mathbb{N}^n \rightarrow \mathbb{N}$ is an n -ary partial function. What should it mean to say that f can be computed by a program P on the URM?

Consider the computation $P(a_1, a_2, \dots, a_n)$. If we have $(a_1, a_2, \dots, a_n) \in dom(f)$, we expect this computation $P(a_1, a_2, \dots, a_n)$ to stop with the value $b = f(a_1, a_2, \dots, a_n)$ in some distinguished register (which may as well be R_1). If $(a_1, a_2, \dots, a_n) \notin dom(f)$, we expect $P(a_1, a_2, \dots, a_n) \uparrow$.

2.8 Definition Let $f : \mathbb{N}^n \rightarrow \mathbb{N}$ be a partial function.

(a) Suppose P is a program and $a_1, a_2, \dots, a_n, b \in \mathbb{N}$.

(i) The computation $P(a_1, a_2, \dots, a_n)$ *converges* to b , written $P(a_1, a_2, \dots, a_n) \downarrow b$, if $P(a_1, a_2, \dots, a_n) \downarrow$ and in the final configuration b is in R_1 .

(ii) P *URM-computes* f if, for every $a_1, a_2, \dots, a_n, b \in \mathbb{N}$, $P(a_1, a_2, \dots, a_n) \downarrow b$ iff $(a_1, a_2, \dots, a_n) \in \text{dom}(f)$, and $f(a_1, a_2, \dots, a_n) = b$. (In particular, this means that $P(a_1, a_2, \dots, a_n) \downarrow$ iff $(a_1, a_2, \dots, a_n) \in \text{dom}(f)$).

(b) The function f is *URM-computable* if there is a program P that URM-computes f .

NOTATION The class of all URM-computable functions is denoted by \mathcal{C} . The subclass of all n -ary URM-computable functions is denoted by \mathcal{C}_n . We will often use the term “computable” to mean URM-computable.

2.9 Examples

(a) $f(x, y) = x + y$. Here f is of arity 2 i.e. is binary and is total ($\text{dom}(f) = \mathbb{N} \times \mathbb{N}$). We obtain $x + y$ by adding 1 to x (using successor instruction) y times. Initial configuration is:

R_1	R_2	R_3	R_4	R_5	R_6	...		
x	y	0	0	0	0			

and a typical later configuration will be:

R_1	R_2	R_3	R_4	R_5	R_6	\dots			
$x+k$	y	k	0	0	0				

where $k = 0$ initially. In fact, the content k of register R_3 is used as a counter to record the number of times that 1 has been added to x , and the program halts when $k = y$, leaving the value $x + y$ in R_1 . The flow chart is:

and a suitable program is:

$$I_1 = J(3, 2, 5)$$

$$I_2 = S(1)$$

$$I_3 = S(3)$$

$$I_4 = J(1, 1, 1)$$

running this we get

x	y	0	0	0	0	...		
-----	-----	---	---	---	---	-----	--	--

$x+1$	y	1	0	0	0	...		
-------	-----	---	---	---	---	-----	--	--

$x+2$	y	2	0	0	0	...		
-------	-----	---	---	---	---	-----	--	--

etc. until we finally stop with final configuration:

$x+y$	y	y	0	0	0	...		
-------	-----	-----	---	---	---	-----	--	--

Thus, $f(x, y) = x + y$ is computable.

(b) $f(x) = x \ominus 1$ – called cut-off subtraction so that

$$f(x) = \begin{cases} x - 1 & \text{if } x \geq 1, \\ 0 & \text{if } x = 0, \end{cases}$$

thus f is unary and total. Take initial configuration:

x	0	1	0	0	0	...
-----	---	---	---	---	---	-----

and typical configuration (with $k = 0$ initially):

x	k	$k+1$	0	0	0	...
-----	-----	-------	---	---	---	-----

Note the “1” in the initial configuration which is not in the standard form but has been pre-processed (for convenience) by running $S(3)$ on the standard form. The program will be designed to do the following: if $x = 0$, stop; otherwise run two counters containing k and $k + 1$, starting with $k = 0$. Check whether or not $x = k + 1$. If so, the required value is k ; if not, increase both counters by 1 and check again.

Exercise

Draw a flow chart.

A suitable program is:

$$I_1 = J(1, 4, 7)$$

$$I_2 = J(1, 3, 6)$$

$$I_3 = S(2)$$

$$I_4 = S(3)$$

$$I_5 = J(1, 1, 2)$$

$$I_6 = T(2, 1)$$

Tracing this we get:

With $x = 0$. $I_1 = J(1, 4, 7)$ jumps ($r_1 = r_4 = 0$). Next instruction does not exist, so the machine stops with $x = 0 \in R_1$. Thus we output 0 which is correct since $f(0) = 0$.

With $x = 1$. $I_1 = J(1, 4, 7)$ does not jump ($r_1 \neq r_4$ i.e. $x \neq 0$). Next instruction = $I_2 = J(1, 3, 6)$ jumps ($x = 1$).

Next instruction = $I_6 = T(2, 1)$ puts 0 in R_1 ; machine now stops (no next instruction). Thus we output 0 which is correct because $f(1) = 1 - 1 = 0$.

With $x = 2$. $I_1 = J(1, 4, 7)$ does not jump ($r_1 \neq r_4$ i.e. $x \neq 0$). Next instruction = $I_2 = J(1, 3, 6)$ does not jump ($x \neq 1$). Next instruction = $I_3 = S(2)$ puts 1 in R_2 . Next instruction = $I_4 = S(3)$ puts 2 in R_3 . Next instruction = $I_5 = J(1, 1, 2)$ jumps. Next instruction = $I_2 = J(1, 3, 6)$ jumps ($x = 2$). Next instruction = $I_6 = T(2, 1)$ puts 1 in R_1 and machine stops (no next instruction). So we output 1 which is correct because $f(2) = 2 - 1 = 1$.

Etc. Thus $f(x) = x \ominus 1$ is a computable function.

$$(c) f(x) = \begin{cases} \frac{1}{2}x & \text{if } x \text{ even,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Here f is unary and partial – its domain is the set of even integers. So the program we write must not stop on odd inputs.

Procedure to compute $f(x)$: run two counters containing k and $2k$ for $k = 0, 1, 2, \dots$. For successive values of k , check whether or not $x = 2k$. If so, the required answer is k ; otherwise increase k by 1 and $2k$ by 2 (because $2(k + 1) = 2k + 2$) and repeat. If x is odd, this procedure clearly will

never halt because x will never be $2k$.

Typical configuration is (with $k = 0$ initially):

x	2k	k	0	0	0	...
---	----	---	---	---	---	-----

A suitable program is:

$$I_1 = J(1, 2, 6)$$

$$I_2 = S(3)$$

$$I_3 = S(2)$$

$$I_4 = S(2)$$

$$I_5 = J(1, 1, 1)$$

$$I_6 = T(3, 1)$$

The first few configurations on running this program are:

x	0	0	0	0	0	...
---	---	---	---	---	---	-----

x	2	1	0	0	0	...
---	---	---	---	---	---	-----

x	4	2	0	0	0	...
---	---	---	---	---	---	-----

Etc. If x is even, this stops with $x = 2k \in R_2$, for some k , and it transfers k from R_3 to R_1 i.e. $k = \frac{1}{2}x$ is output.

Therefore, f is computable.

Exercise Trace this program (i) with $x = 12$, and (ii) with $x = 13$.

$$(d) f(x, y) = \begin{cases} 0 & \text{if } x \leq y \\ 1 & \text{if } x > y. \end{cases}$$

Then f is of arity 2 and is total. We consider two programs for this. First, take initial configuration as

x	y	0	0	0	...
---	---	---	---	---	-----

and program P as follows:

$$I_1 = J(1, 3, 6)$$

$$I_2 = J(2, 3, 5)$$

$$I_3 = S(3)$$

$$I_4 = J(1, 1, 1)$$

$$I_5 = S(4)$$

$$I_6 = T(4, 1)$$

Trace this with various values of x :

Case 1 $x = 0$

Here $x \leq y$ for any y so $f(x, y) = 0$. What value does P output?

$I_1 = J(1, 3, 6)$ jumps.

Next instruction = $I_6 = T(4, 1)$ puts 0 in R_1 and m/c stops.

So we output 0 which is correct.

Case 2 $x = 1$

Subcase a) $y < x$

Here y must be 0 and $f(x, y) = 1$. What value does P output?

$I_1 = J(1, 3, 6)$ does not jump.

Next instruction = $I_2 = J(2, 3, 5)$ jumps.

Next instruction = $I_5 = S(4)$ puts 1 in R_4 .

Next instruction = $I_6 = T(4, 1)$ puts 1 in R_1 and m/c stops.

So we output 1 which is correct.

Subcase b) $x \leq y$

Here $y \geq 1$, and $f(x, y) = 0$. What value does P output?

$I_1 = J(1, 3, 6)$ does not jump ($x > 0$).

Next instruction $I_2 = J(2, 3, 5)$ does not jump ($y > 0$).

Next instruction = $I_3 = S(3)$ puts 1 in R_3 .

Next instruction = $I_4 = J(1, 1, 1)$ jumps.

Next instruction = $I_1 = J(1, 3, 6)$ jumps ($r_1 = r_3 = 1$).

Next instruction = $I_6 = T(4, 1)$ puts 0 in R_1 and m/c stops.
So we output 0 which is correct.

Case 3 $x = 2$

Subcase a) $y < x$

Here y must be 0 or 1 and $f(x, y) = 1$. What value does P output?

$I_1 = J(1, 3, 6)$ does not jump ($2 > 0$).

(i) sub-subcase $y = 0$

$I_2 = J(2, 3, 5)$ jumps ($r_2 = r_3 = 0$).

Next instruction = $I_5 = S(4)$ which puts 1 in R_4 .

Next instruction = $I_6 = T(4, 1)$ puts 1 in R_1 and m/c stops.

So we output 1 which is correct.

(ii) sub-subcase $y = 1$

$I_2 = J(2, 3, 5)$ does not jump ($1 > 0$)

Next instruction = $I_3 = S(3)$ which puts 1 in R_3 .

Next instruction = $I_4 = J(1, 1, 1)$ jumps.

Next instruction = $I_1 = J(1, 3, 6)$ does not jump ($2 > 1$).

Next instruction = $I_2 = J(2, 3, 5)$ jumps ($r_2 = r_3 = 1$).

Next instruction = $I_5 = S(4)$ puts 1 in R_4 .

Next instruction = $I_6 = T(4, 1)$ puts 1 in R_1 and m/c stops.

So we output 1 which is correct.

Subcase b) $x \leq y$ (So $y \geq 2$).

$I_1 = J(1, 3, 6)$ does not jump ($2 > 0$).

Next instruction = $I_2 = J(2, 3, 5)$ does not jump ($r_2 > 0$).

Next instruction = $I_3 = S(3)$ which puts 1 in R_3 .

Next instruction = $I_4 = J(1, 1, 1)$ jumps.

Next instruction = $I_1 = J(1, 3, 6)$ does not jump ($2 > 1$).

Next instruction = $I_2 = J(2, 3, 5)$ does not jump ($r_2 > 1$).

Next instruction = $I_3 = S(3)$ puts 2 in R_3 .

Next instruction = $I_4 = J(1, 1, 1)$ jumps.

Next instruction = $I_1 = J(1, 3, 6)$ jumps ($2 = 2$).

Next instruction = $I_6 = T(4, 1)$ puts 0 in R_1 and m/c stops.

So we output 0 which is correct.

Etc.

For the second program, P' , initial configuration is:

x	y	0	1	0	0	0	...
---	---	---	---	---	---	---	-----

(again run preprocessor $S(4)$ on standard initial configuration) and the typical configuration (with $k = 0$ initially) is

x	y	k	1	0	0	0	...
---	---	---	---	---	---	---	-----

Program is

$$I_1 = J(1, 3, 10)$$

$$I_2 = J(2, 3, 7)$$

$$I_3 = S(3)$$

$$I_4 = J(1, 3, 9)$$

$$I_5 = J(2, 3, 7)$$

$$I_6 = J(1, 1, 3)$$

$$I_7 = T(4, 1)$$

$$I_8 = J(1, 1, 10)$$

$$I_9 = T(5, 1)$$

Exercise

(1) Analyse this program as was done for the previous program P .

(2) What is the length of the shortest program to compute f ?

Question How can we *prove* that P or P' compute f ? (the basic question in “formal methods”)

2.10 The function $f_P^{(n)}$ computed by P .

In 2.8 and 2.9, we considered the problem “given a function f , write a program (if possible) to compute f on the URM”. Notice that lots of different programs may compute the same function. Eg. for the function in Example 2.9 d), we saw two

different programs for computing f (one had 6 instructions and the other had 9). Conversely, suppose given a program P and $n \geq 1$. By considering initial configurations

a_1	a_2	\dots	a_n	0	0	0	\dots
-------	-------	---------	-------	---	---	---	---------

we see that there is a unique n -ary function $f_P^{(n)}$ that P computes defined by:

$$f_P^{(n)}(a_1, \dots, a_n) = \begin{cases} \text{unique } b \text{ s.t. } P(a_1, \dots, a_n) \downarrow b \\ \quad \quad \quad \text{if } P(a_1, \dots, a_n) \downarrow, \\ \text{undefined} \quad \text{if } P(a_1, \dots, a_n) \uparrow. \end{cases}$$

Example Take P to be the program

$$I_1 = S(1)$$

$$I_2 = J(1, 1, 1)$$

Find the n -ary function $f_P^{(n)}$.

Solution Consider initial configuration

a_1	a_2	\dots	a_n	0	0	0	\dots
-------	-------	---------	-------	---	---	---	---------

corresponding to $f_P^{(n)}(a_1, \dots, a_n)$.

Start program, to get:

$I_1 = S(1)$ puts $a_1 + 1$ in R_1 .

Next instruction = $I_2 = J(1, 1, 1)$ jumps.

Next instruction = $I_1 = S(1)$ puts $a_1 + 2$ in R_1 .

Next instruction = $I_2 = J(1, 1, 1)$ jumps.

Next instruction = $I_1 = S(1)$ puts $a_1 + 3$ in R_1 etc.

After m cycles round this loop the configuration is:

$a_1 + m$	a_2	\dots	a_n	0	0	\dots
-----------	-------	---------	-------	---	---	---------

and this computation never halts for any input. Therefore $f_P^{(n)}$ is the *totally undefined* function i.e. $f_P^{(n)}(a_1, a_2, \dots, a_n) = \text{undefined}$ for all a_1, a_2, \dots, a_n . So domain of this function = \emptyset .

Note that this shows that the totally undefined function is computable since the program just given computes it.

Example.

Take P to be the program

$$I_1 = Z(1)$$

$$I_2 = S(1)$$

$$I_3 = S(1)$$

\dots

...

$$I_{k+1} = S(1)$$

in which there are k instructions $S(1)$ for some fixed k . Find the n -ary function $f_P^{(n)}$ determined by P .

Solution Take initial configuration:

a_1	a_2	\dots	a_n	0	0	0	\dots
-------	-------	---------	-------	---	---	---	---------

Start the computation:

$I_1 = Z(1)$ puts 0 in R_1 .

Next instruction = $I_2 = S(1)$ puts 1 in R_1 .

Next instruction = $I_3 = S(1)$ puts 2 in R_1 .

:

Next instruction = $I_{k+1} = S(1)$ puts k in R_1 and m/c stops with final configuration:

k	a_2	a_3	\dots	a_n	0	0	0	\dots
-----	-------	-------	---------	-------	---	---	---	---------

So $f_P^{(n)}(a_1, \dots, a_n) = k$ for all a_1, \dots, a_n and is totally defined (ie domain = \mathbb{N}^n). Thus, $f_P^{(n)}$ is the constant function with value k – which we now see is a computable function. In particular, with $n = 1$, $f_P^{(1)}(a) = k$ for all $a \in \mathbb{N}$.

2.11 DECIDABLE PREDICATES AND PROBLEMS

Suppose we want to know whether or not a number x equals a number y . We input x, y and check whether or not $x = y$. The answer is either Yes ie true or No ie false. Thus “=” can be thought of as a function $= : \mathbb{N} \times \mathbb{N} \rightarrow \{\text{true}, \text{false}\}$ ie can be thought of as a Boolean valued function or predicate. Make the identification $1 \equiv \text{Yes} \equiv \text{true}$ and $0 \equiv \text{No} \equiv \text{false}$, then we obtain a function $C_= : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$ (and hence $C_= : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$) defined by:

$$C_=(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y \end{cases}$$

called the characteristic function of “=”. Therefore, checking whether or not the URM (or any other computer) can decide the question “ $x = y$?” is exactly the same thing as checking whether or not the function $C_=$ is computable.

In general, let $M(\underline{x})$ denote an n -ary predicate ie $M : \mathbb{N}^n \rightarrow \{\text{true}, \text{false}\}$. So for each $\underline{x} = (x_1, \dots, x_n) \in \mathbb{N}^n$, either $M(\underline{x})$ is true or $M(\underline{x})$ is false.

Examples

1) Take M as equality ie $M(x, y)$ means “ $x = y$ ”. So M is binary and for each pair $(x, y) \in \mathbb{N}^2$, either “ $x = y$ ” is true or it is false. So equality is a predicate, as already noted.

2) Take M as the unary predicate “ > 100 ” ie $M(x)$ means “ $x > 100$ ”. Then for each $x \in \mathbb{N}$, “ $x > 100$ ” is either true or false so M is a predicate.

3) Take $M(x, y, z)$ to mean “ $x + y$ is divisible (exactly) by $z + 1$ ”. For each triple $(x, y, z) \in \mathbb{N}^3$, $M(x, y, z)$ is either true or false and so M is a ternary predicate (ie of arity 3).

2.12 Definition Given an n -ary predicate M , we define the *characteristic function* C_M of M by

$$C_M(\underline{x}) = \begin{cases} 1 & \text{iff } M(\underline{x}) \text{ is true,} \\ 0 & \text{iff } M(\underline{x}) \text{ is false.} \end{cases}$$

We say that M is *decidable* if the function C_M is computable.

We say M is *undecidable* if C_M is not computable.

Notes

1) C_M is always a total function because $M(\underline{x})$ is either true or false for every $\underline{x} \in \mathbb{N}^n$.

2) A predicate $M(\underline{x})$ is sometimes called a problem. So $M(\underline{x})$ denotes the problem of determining when $M(\underline{x})$ is true.

3) To say that a predicate or problem is decidable is to say that it can be solved by a procedure, algorithm or program (all the same thing).

Example

1) Take M to be the predicate equality ie $M(x, y)$ means “ $x = y$ ”; then M is decidable.

Solution The characteristic function C_M is

$$C_M(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y \end{cases}$$

We show that this function is computable.

Take initial configuration

x	y	0	1	0	0	0	...
-----	-----	---	---	---	---	---	-----

and program P as follows:

$$I_1 = J(1, 2, 4)$$

$$I_2 = T(3, 1)$$

$$I_3 = J(1, 1, 5)$$

$$I_4 = T(4, 1)$$

This program computes C_M so M ie “ $=$ ” is decidable.

2) Let $M(x, y)$ be the binary predicate “ $x > y$ ”. Then M is decidable.

Solution.

Let $C_M(x, y)$ be the characteristic function of M . Thus,

$$C_M(x, y) = \begin{cases} 1 & \text{if } x > y, \\ 0 & \text{if } x \leq y. \end{cases}$$

This function is computable, see Example d) of §2.9. Therefore M is decidable.

Exercise

Show that the following predicates are decidable.

- a) $M(x, y)$ meaning “ $x \neq y$ ”.
- b) $M(x)$ meaning “ $x = 0$ ”.

Question.

Are there any predicates or problems which are undecidable ie not solvable by any computer? The answer is yes and one example is the famous halting problem (see §2.5).

The halting problem.

There is no algorithm which accepts programs P (in any language but in the URM’s language in particular) together with inputs \underline{x} to P and tells us whether $P(\underline{x})$ halts or does not halt. Thus, the problem “ $P(\underline{x})$ halts” is undecidable.

Sketch proof (A.M. Turing)

Consider the real numbers between 0 and 1 ie the elements of $[0, 1]$. Each such a has a decimal expansion

$$a = 0.a_1a_2a_3a_4 \dots a_n \dots$$

Sometimes such an expansion terminates finitely ie all a_n are zero beyond a certain point

eg. $\frac{1}{4} = 0.2500000 \dots$, $\frac{1}{2} = 0.50000 \dots$

Sometimes there are infinitely many non-zero a_n in the expansion

eg. $\frac{1}{4} = 0.249999 \dots$, $\frac{1}{2} = 0.49999 \dots$, $\frac{\pi}{4} = 0.785398163 \dots$

We call a number $a \in [0, 1]$, computable if there is an algorithm or program P_a which computes the digits a_n in the expansion of a ie.

$$P_a(1) = a_1, P_a(2) = a_2, P_a(3) = a_3, \dots, P_a(n) = a_n, \dots$$

Example

The number $\frac{1}{2} = 0.5000 \dots = 0.4999 \dots$ is computable.

Solution Take $\frac{1}{2} = 0.4999 \dots$

With initial configuration

n	1	4	9	0	0	0	...
---	---	---	---	---	---	---	-----

The program

$$I_1 = J(1, 2, 4)$$

$$I_2 = T(4, 1)$$

$$I_3 = J(1, 1, 5)$$

$$I_4 = T(3, 1)$$

computes the n^{th} digit in the expansion $0.4999\dots$

Exercise Show that any rational number is computable.

Hint the decimal expansion of a rational number either terminates finitely or has a repeating block. For example, $\frac{1}{2} = 0.5000\dots$ terminates finitely; $\frac{229}{990} = 0.231313131\dots$ has a repeating block (in this case 31). There are irrational computable numbers eg. $0.101001000100001000001\dots$ is irrational but computable (check this as an exercise). The numbers $\pi - 3$ and $e - 2$ are computable irrational numbers (hard).

Next, we list all our programs in an infinite sequence

$P_1, P_2, P_3, \dots, P_n, \dots$. For URM program we do this as follows. For such a program P define the *norm*, $|P|$, of P to be the sum of all the integers which occur in the instructions in P .

Example. Suppose P is the program:

$$I_1 = Z(1)$$

$$I_2 = T(4, 1)$$

$$I_3 = S(1)$$

$$I_4 = J(1, 2, 5)$$

Then $|P| = 1 + 4 + 1 + 1 + 1 + 2 + 5 = 15$.

It is clear that:

- 1) Each program P has a norm $|P|$.
- 2) Given any number n , there are only finitely many programs P with $|P| = n$.

There are only finitely many programs P with $|P| = 1$; list these in some order (it does not matter how).

There are only finitely many programs P with $|P| = 2$; list these in some order (again it does not matter how).

There are only finitely many programs P with $|P| = 3$; list these in some order.

Etc. – keep on doing this for each n . We get a listing $P_1, P_2, P_3, \dots, P_n, \dots$ of all programs.

Now for each program P_n list the successive outputs from running P_n on inputs $1, 2, 3, \dots$; if a particular computation (say $P_n(m)$) never halts we indicate this with the symbol \bigcirc (so we get no output in this case). Note that any computa-

tion which terminates, say $P_n(m)$, outputs an element a_{nm} of \mathbb{N} . By following this computation with the sub routine which removes from a_{nm} the highest multiple of 10 it contains, we can suppose that $a_{nm} \in \{0, 1, 2, \dots, 9\}$. (Eg. use the division algorithm – see later.)

Next, for each n , list the outputs $P_n(1), P_n(2), P_n(3), \dots, P_n(m), \dots$ in a row as a “decimal”. We get a doubly infinite array looking something like the following:

$$\begin{array}{rcccccccc}
 P_1 : & - & 0.a_{11} & a_{12} & a_{13} & \bigcirc & a_{15} & a_{16} & \bigcirc & a_{18} & \dots \\
 P_2 : & - & 0.a_{21} & \bigcirc & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} & \dots \\
 P_3 : & - & 0.a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} & \dots \\
 P_4 : & - & 0.a_{41} & a_{42} & \bigcirc & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} & \dots \\
 P_5 : & - & 0.a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} & \dots \\
 \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
 \end{array}$$

where a_{nm} (the entry in the n^{th} row and the m^{th} column) is an integer in $\{0, 1, 2, \dots, 9\}$ if $P_n(m)$ halts; otherwise it is \bigcirc .

Clearly every computable number is contained in the list formed from the rows, working downwards. In fact, every computable number occurs in this list infinitely often for two reasons:

1) Numbers such as $0.5000\dots$ and $0.4999\dots$ represent the same number.

2) If a program P computes a particular computable number, then infinitely many trivial modifications of P compute that number.

None of this matters – all that matters is that all forms of every computable number are somewhere in the list.

Now “go down the diagonal” of this array ie consider

$a_{11}, a_{22}, a_{33}, a_{44}, \dots$ and define the number b_n , as follows, for $n = 1, 2, 3, \dots$

$$b_n = \begin{cases} (a_{nn} + 1) \bmod 10, & \text{if } P_n(n) \text{ halts,} \\ 0, & \text{otherwise.} \end{cases}$$

Note. The essential point of $(a_{nn} + 1) \bmod 10$ is that it is a number in $\{0, 1, 2, \dots, 9\}$ different from a_{nn} and is obtained algorithmically or computably from a_{nn} – any other way of doing this is fine.

Now let b be the real number in $[0, 1]$ defined by

$$b = 0.b_1b_2b_3\dots b_n\dots$$

This number is not computable since it differs from every row in the above array and therefore differs from every computable number.

Suppose there were an algorithm A which on input (P, n) , where P is a program and n an input to P , tells us whether or not $P(n)$ halts; we show that this supposition leads to the contradiction that b is computable by giving a procedure to compute the numbers b_n :

Procedure

Input (P_n, n) to A ie run $A(P_n, n)$.

Case 1. A tells us that $P_n(n)$ halts. Run $P_n(n)$ to get output $a_{nn} \in \{0, 1, \dots, 9\}$ from which we compute $(a_{nn} + 1) \bmod 10$ ie b_n .

Case 2. A tells us that $P_n(n)$ does not halt.

Then immediately write down 0 ie b_n again.

This gives us an effective procedure for calculating the b_n showing that b is in fact a computable number – contradiction. Thus, the halting problem is undecidable. ■

Note.

This has nothing to do with memory size, processor speed etc., etc. The halting problem is undecidable because there is no algorithm A to decide it. Many interesting problems in Computer Science are undecidable: there is no algorithm to write shortest versions of programs; there is no algorithm to check correctness of software etc., etc. It is ab-

stract machines such as the URM which have to be used to study such problems.

Worked Example (Exercises 2, No 4)

Suppose that P is a program without any jump instructions. Show that there is a number m s.t either $f_P^{(1)}(x) = m$ for all x , or $f_P^{(1)}(x) = x + m$ for all x .

Solution

For a given program P , let the length of P be the number of instructions in P ; this number n , say, is a natural number bigger than zero. Thus, we can use the principle of induction as a method of proof applied to the number $n = \text{length of } P$. Recall the principle of induction:

Suppose $S(n)$ is a statement or proposition depending on the natural number n . Suppose:

- (i) Basis step $S(n)$ is true with $n = 1$ ie $S(1)$ is true.
- (ii) Inductive step $S(n)$ is true with $n = k + 1$ whenever it is true with $n = k$.

Then $S(n)$ is true for all n .

To use this method to solve the given problem, we take the standard initial configuration

x	0	0	0	...
-----	---	---	---	-----

and take our induction hypothesis $\bar{S}(n)$ to be:

$\bar{S}(n)$ is the proposition that for any program P of length n ($n = 1, 2, 3, \dots$) without jump instructions, the content of each register of the URM in the final configuration (given the above initial configuration) is of the form m or $x + m$, for all x , where m is some fixed number depending only on the register. Thus, for each register R_k , there is a fixed number $m_k \in \mathbb{N}$ s.t. the content of R_k in the final configuration is either m_k for all x or $x + m_k$ for all x .

We want to show that $\bar{S}(n)$ is true for all n , and we note:

1) It follows from this that either $f_P^{(1)}(x) = m_1 = m$ for all x or

$f_P^{(1)}(x) = x + m_1 = x + m$ for all x

since the value of $f_P^{(1)}(x)$ is in R_1 in the final configuration.

Thus, the problem is solved by showing $\bar{S}(n)$ is true for all n .

2) It is obvious that any computation under P , where P has no jump instructions, must terminate (in fact, it follows from this that the collection of programs without jump instructions cannot compute all computable functions).

Step 1. The basis step. We check $\bar{S}(1)$ is true:

Let $P = I_1$ be a program with one instruction (which is not

a jump instruction).

a) Suppose $I_1 = Z(l)$ for some l .

Then in the final configuration

$$r_1 = \begin{cases} 0 & \text{if } l = 1, \\ x & \text{if } l \neq 1 \end{cases}$$

for all x ie either $r_1 = 0$ for all x or is $x = x + 0$ for all x (so $m = 0$). Also $r_i = 0 = m$ for all x whenever $i > 1$.

b) Suppose $I_1 = S(l)$ for some l . Then in the final configuration

$$r_1 = \begin{cases} x + 1 & \text{if } l = 1, \\ x & \text{if } l \neq 1 \end{cases}$$

for all x ie either $r_1 = x + 1 = x + m$ (with $m = 1$) for all x or $r_1 = x = x + 0 = x + m$ (with $m = 0$) for all x . Also for all $i > 1$,

$$r_i = \begin{cases} 1 & \text{if } i = l \quad (= m \text{ with } m = 1), \\ 0 & \text{if } i \neq l \quad (= m \text{ with } m = 0) \end{cases}$$

this holding for all x .

c) Suppose $I_1 = T(i, j)$.

If $i, j > 1$, then $r_1 = x = x + m$ (with $m = 0$) for all x , and $r_k = 0 = m$ (with $m = 0$) for all x whenever $k > 1$.

If $i = j = 1$, then $r_1 = x = x + m$ (with $m = 0$) for all x , and $r_k = 0 = m$ (with $m = 0$) for all x whenever $k > 1$.

If $i = 1$ and $j > 1$, then $r_1 = x = x + 0$ for all x , and for all x and each $k > 1$

$$r_k = \begin{cases} x \text{ if } k = j & (= x + 0 = x + m), \\ 0 \text{ if } k \neq j & (= 0 = m). \end{cases}$$

If $i > 1$ and $j = 1$, then $r_k = 0$ for all x and for every k ie $r_k = m = 0$.

Thus, in each case the contents of the registers are in the required form m or $x + m$, for all x , and so $\bar{S}(1)$ is true.

Step 2 The inductive step. We show $\bar{S}(k) \Rightarrow \bar{S}(k + 1)$ ie $\bar{S}(k + 1)$ true whenever $\bar{S}(k)$ true.

So suppose $\bar{S}(k)$ true. Let $P = I_1, \dots, I_k, I_{k+1}$ be any program (without jump instructions) of length $k + 1$. Then $P' = I_1, \dots, I_k$ is a program (without jump instructions) of length k , and so $\bar{S}(k)$ is true for P' . By our induction hypothesis, after running P' every register contains either m or

$x + m$ (for all x) for some m varying with the register. Running P amounts to applying I_{k+1} to the final configuration of the URM under P' .

a) Suppose $I_{k+1} = Z(l)$ for some l . Then for all x

$$r_1 = \begin{cases} 0 & \text{if } l = 1 \quad (= m \text{ with } m = 0), \\ m \text{ or } x + m & \text{if } l > 1. \end{cases}$$

Also, for $i > 1$ we have for all x that

$$r_i = \begin{cases} m \text{ or } x + m & \text{if } i \neq l, \\ 0 & \text{if } i = l \quad (= m \text{ with } m = 0). \end{cases}$$

b) Suppose $I_{k+1} = S(l)$ for some l .

Then for all $i \geq 1$ we have for all x that

$$r_i = \begin{cases} m + 1 \text{ or } x + (m + 1) & \text{if } i = l \quad (= m' \text{ or } x + m'), \\ m \text{ or } x + m & \text{if } i \neq l. \end{cases}$$

c) Suppose, finally, that $I_{k+1} = T(i, j)$.

If $r_i = m$, then $r_j := m$.

If $r_i = x + m$, then $r_j := x + m$.

For all other registers - no change. So, again, in all cases the registers in the final configuration have the required form.

Therefore, $\bar{S}(k + 1)$ is true as required.

3 RELATIONS

See Textbook pp 91 - 103.

Recall that, for sets X and Y , the *Cartesian product* $X \times Y$ is defined to be the set $\{(x, y); x \in X, y \in Y\}$.

3.1 Definition Let X and Y be sets. A (*binary*) *relation* R from X to Y is a subset of the Cartesian product $X \times Y$; thus $R \subseteq X \times Y$. If $(x, y) \in R$, we write xRy . Conversely, if xRy we may write $(x, y) \in R$. Thus, $(x, y) \in R$ and xRy are synonymous (in the form xRy , R is said to be *written infix*). If xRy , we say that x is *related (by R) to y* . In case $X = Y$, we call R a (*binary*) *relation on X* .

The set $\{x \in X; (x, y) \in R \text{ for some } y \in Y\}$ is called the *domain* of R . The set $\{y \in Y; (x, y) \in R \text{ for some } x \in X\}$ is called the *range* or *image set* of R .

Example 1

Let X be any non-empty set and let $R = \{(x, x); x \in X\} = \{(x, y); x, y \in X \text{ and } x = y\}$. Then R is a relation on X called *equality*. Note that the symbolism “ $(x, y) \in =$ ” is cumbersome and it is more natural to write this infix i.e. to write “ $x = y$ ”.

Example 2 (Page 92, Example 2.4.3)

Take $X = \{2, 3, 4\}$, $Y = \{3, 4, 5, 6, 7\}$ and take $R = \{(2, 4), (2, 6), (3, 3), (3, 6), (4, 4)\}$. Then $R \subseteq X \times Y$ since each element of R belongs to $X \times Y$. Eg. $(2, 4) \in X \times Y$ since $2 \in X$ and $4 \in Y$, $(2, 6) \in X \times Y$ etc. Thus, R is a relation from X to Y . Next, the domain of R is the set $\{x \in X; (x, y) \in R \text{ for some } y \in Y\} = \{2, 3, 4\} = X$. Finally, the range of R is the set $\{y \in Y; (x, y) \in R \text{ for some } x \in X\} = \{3, 4, 6\} \subset Y$.

Example 3

Let X be any set. Then

- (i) \emptyset is a relation on X .
- (ii) $X \times X$ is a relation on X .

Solution. (i) $R = \emptyset \subseteq X \times X$ and hence is a relation.

Here domain of R is \emptyset ie is empty and range of R is also \emptyset .

(ii) $R = X \times X \subseteq X \times X$ and hence R is a relation.

Here domain of $R = X =$ range of R .

Exercise: check these claims.

3.2 Ways of Defining Relations

(1) One way to define a relation R from X to Y is to specify which ordered pairs belong to R , as in Examples 1, 2 and 3 above (in the case of Example 3(i) no pairs belong to R ; in case of (ii) all pairs belong to R).

(2) Another way is to give a rule for membership ie $R = \{(x, y) \in X \times Y; P(x, y) \text{ holds}\}$ where $P(x, y)$ is a property or predicate depending on x and y .

Example 4

The Example 2 can be given in this form as

$$R = \{(x, y) \in X \times Y; x \text{ divides } y \text{ exactly}\}.$$

So with $X = \{2, 3, 4\}$, $Y = \{3, 4, 5, 6, 7\}$ we see that $R = \{(2, 4), (2, 6), (3, 3), (3, 6), (4, 4)\}$ as before.

Example 5

Take $X = Y = N$ and define R by

$$R = \{(n, m) \in N \times N; n \leq m\}.$$

Then $(1, 1) \in R$, $(1, 2) \in R$, $(3, 7) \in R, \dots$ etc. but $(2, 1) \notin R$, $(4, 2) \notin R$, $(9, 5) \notin R$, etc.

Note that domain of $R = N$ and range of $R = N$ because $(n, n) \in R$ for all $n \in N$.

Exercise check this claim.

(3) A third way to specify binary relations is to use a table (cf. relational databases.)

Example 6

For Example 2 again, the relation R is given by

$$R = \{(2, 4), (2, 6), (3, 3), (3, 6), (4, 4)\}.$$

We can display this

relation as a table, as follows:

X	Y
2	4
2	6
3	3
3	6
4	4

Example 7 (Page 92, Example 2.4.2)

The table

Student	Course
Bill	CS
Mary	Math
Bill	Art
Beth	History
Beth	CS
Dave	Math

gives a relation from Students to Courses. Eg (Bill, CS) is in the relation, (Bill, Art) is in the relation but (Bill, Math) is not.

3.3 Diagrammatic Representation of Relations

1) Use of Venn diagrams.

Let X and Y be sets. The ordered pairs in a relation R can be shown in Venn diagram form as follows:

This indicates that $(x_1, y_1) \in R$, $(x_1, y_2) \in R$, $(x_2, y_2) \in R$ etc. This representation is reminiscent of functions, and raises the question “What is the connection between relations and functions?” We answer this next.

3.4 Definition Let $f : X \rightarrow Y$ be a partial function.

We define the graph of f , written $\text{graph}(f)$, by $\text{graph}(f) = \{(x, f(x)); x \in \text{dom}(f)\} = \{(x, y) \in X \times Y; x \in \text{dom}(f) \text{ and } y = f(x)\}$. Thus, $\text{graph}(f) \subseteq X \times Y$.

Example 8

In the case $X = Y = R$, $\text{graph}(f)$ is the familiar graph of f and is a “curve” in the plane. Eg. if $f : R \rightarrow R$

is the total function defined by $f(x) = x^2$ for all $x \in R$. Then $\text{graph}(f) = \{(x, f(x)); x \in R\} = \{(x, x^2); x \in R\} \subset R \times R = R^2$.

Example 9

Take $A = \{a, b, c\}$ and $B = \{1, 2\}$. Suppose $f : A \rightarrow B$ is the function defined by $f(a) = 1$; $f(b) = 1$; $f(c) = 2$. Then $\text{graph}(f) = \{(a, 1), (b, 1), (c, 2)\} \subset A \times B$. It is not a curve in the plane in this case.

Now suppose $f : X \rightarrow Y$ is a partial function and form its graph, $\text{graph}(f)$. Since $\text{graph}(f) \subseteq X \times Y$, it is a relation R from X to Y . It has the following properties:

- (i) given *any* $x \in U = \text{dom}(f)$, there is an element $y (= f(x))$ s.t. $(x, y) \in R$.
- (ii) if (x, y) and (x, y') both belong to R , then $y = y'$ (by the *uniqueness* part of the definition of a function).

So, a function determines a relation with properties (i) & (ii).

Conversely, suppose R is a relation from X to Y which satisfies:

(i) There is a subset $U \subseteq X$ s.t. for every $x \in U$, there is some $y \in Y$ with $(x, y) \in R$.

(ii) If (x, y) and (x, y') both belong to R , then $y = y'$.

Then R determines a partial function $f : X \rightarrow Y$, with domain U , defined by

$$f(x) = y \text{ iff } x \in U \text{ and } (x, y) \in R.$$

Thus, (i) ensures that $f(x) = y$ is defined for *each* $x \in U$.

And (ii) ensures that f is well-defined i.e. the image $f(x)$ is

unique for each $x \in U$. To see this, suppose that $f(x) = y$

and $f(x) = y'$. Then $x \in U$ and both $(x, y) \in R$ and

$(x, y') \in R$. But then (ii) says that $y = y'$ as required.

Therefore, f is a function and, in fact, its graph coincides

with R (exercise).

Thus, functions correspond exactly to relations R satisfying

conditions (i) and (ii).

2) Digraphs. The second diagrammatic representation of re-

lations we consider is for relations on X , ie from X to X for

some set X , and is by means of digraphs.

Example 10 (See Example 2.4.4 on page 93)

Let $X = \{1, 2, 3, 4\}$ and define the relation R on X by

$$R = \{(x, y) \in X \times X; x \leq y\} = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\}.$$

We can represent this relation on X by a digraph as follows:

Draw a vertex (a dot) to represent each element of X :

Next, if $(x, y) \in R$, we draw an arrow (a directed edge) from x to y . If $x = y$, ie $(x, x) \in R$, we draw a loop starting and finishing at x (still regarded as a directed edge). ■

Conversely, if given a diagram of this form, we can recover the relation R on X from which it came just by letting $(x, y) \in R$ iff there is a directed edge from x to y , as follows.

Example 11 (see Example 2.4.5 on Page 93) for a similar problem).

Let $X = \{a, b, c, d\}$. Find the relation on X corresponding to the following digraph.

The relation R is

$$R = \{(a, a), (b, c), (c, b), (d, d), (b, d)\}. \blacksquare$$

3.5 Properties of Relations

3.6 Definition A relation R on a set X is called *reflexive* if $(x, x) \in R$ for every $x \in X$ ie if xRx for every $x \in X$.

Example 12

(1) The relation R in Example 10 is reflexive since $(1, 1) \in R$, $(2, 2) \in R$, $(3, 3) \in R$ and $(4, 4) \in R$ ie $(x, x) \in R$ for every $x \in X$ (in other words $x \leq x$ for all x).

(2) Let X be any set. Then the relation of equality on X is

reflexive since $x = x$ for every $x \in X$.

(3) Let R be the relation of inequality $<$ on N . Then R is not reflexive because it is not the case that $n < n$ for all $n \in N$.

Exercise. Show that a relation R on X is reflexive iff its digraph has a loop at each vertex.

3.7 Definition A relation R on a set X is called *symmetric* if, for all $x, y \in X$, whenever $(x, y) \in R$ we have $(y, x) \in R$, ie xRy implies yRx .

Example 13

1) Take $X = \{a, b, c, d\}$. Then the relation R given by $R = \{(a, a), (b, c), (c, b), (d, d)\}$ is symmetric. However, the relation R' given by $R' = \{(a, a), (b, c), (c, c)\}$ is not symmetric - since $(b, c) \in R'$ but $(c, b) \notin R'$.

2) Let X be any set. Then the relation of equality on X is symmetric since whenever we have $x = y$ we also have $y = x$.

3) Let R be the relation of inequality $<$ on N . Then R is not symmetric because, for example, $2 < 3$ but $3 \not< 2$.

Exercise

Show that a relation R on a set X is symmetric iff whenever there is a directed edge (in the digraph of R) from x to

y , there is a directed edge from y to x , this holding for all $x, y \in X$.

3.8 Definition A relation R on a set X is called *antisymmetric* if, for all $x, y \in X$, whenever $(x, y) \in R$ and $x \neq y$, then $(y, x) \notin R$.

NB. This is not the same thing as “not symmetric”.

This definition can equivalently be stated: for all $x, y \in X$, if xRy and yRx , then $x = y$.

Proof of the equivalence of these definitions.

The first form is: for all $x, y \in X$, $(x, y) \in R$ and $x \neq y \Rightarrow (y, x) \notin R \dots (1)$.

The second form is: for all $x, y \in X$, $(x, y) \in R$ and $(y, x) \in R \Rightarrow x = y \dots (2)$.

Suppose R satisfies the first form ie R satisfies (1), and suppose that $(x, y) \in R$ and $(y, x) \in R$. Then it cannot be the case that $x \neq y$ otherwise from (1) we get $(y, x) \notin R$ contrary to our current assumptions. Therefore $x = y$ and so (2) is satisfied.

Conversely, suppose R satisfies (2), and suppose that $(x, y) \in$

R and $x \neq y$. Then it cannot be the case that $(y, x) \in R$ otherwise from (2) we would get $x = y$ contrary to our assumptions. Therefore, $(y, x) \notin R$ and (1) is satisfied. Thus (1) and (2) are equivalent, as stated.

Example 14

(1) Let X be the set $\{1, 2, 3, 4\}$ and let R be the relation on X defined by \leq as in Example 10, ie $R = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\}$. Then R is antisymmetric because we see that we do not have $(x, y) \in R$ and $(y, x) \in R$ with $x \neq y$. Eg. $(1, 2) \in R$ but $(2, 1) \notin R$; $(2, 3) \in R$ but $(3, 2) \notin R$ etc.

In general, if X is any set of real numbers and R is the relation \leq on X , then R is antisymmetric: this is simply the well-known fact that whenever $x \leq y$ and $y \leq x$, then $x = y$.

(2) Let $X = \{1, 2, 3, 4\}$ and let R_1 be $\{(1, 2), (3, 2), (4, 4)\}$. Then R_1 is antisymmetric but not symmetric. Take $R_2 = \{(1, 2), (2, 1)\}$. Then R_2 is symmetric but not antisymmetric. Take $R_3 = \{(1, 1), (2, 2)\}$. Then R_3 is antisymmetric and is also symmetric. Let $R_4 = \{(1, 2), (2, 1), (3, 2), (4, 4)\}$. Then R_4 is not antisymmetric because we have $(1, 2)$ and $(2, 1)$

both in R_4 and yet $1 \neq 2$. Also R_4 is not symmetric because $(3, 2) \in R_4$ but $(2, 3) \notin R_4$.

Note. The examples in (2) show that “antisymmetric” is not the same thing as “not symmetric”.

3) Let A be any non-empty set and let X denote the power set $\mathcal{P}(A)$ of A ie X is the set of all subsets of A . Define the relation R on X by CRD iff $C \subseteq D$ for arbitrary subsets C, D of A ; so R is subset inclusion \subseteq . Then R is antisymmetric. To see this, suppose CRD and DRC . Then we have $C \subseteq D$ and $D \subseteq C$ and therefore $C = D$ by definition of equality of sets.

Exercise

Show that a relation R on a set X is antisymmetric iff there is at most one directed edge between each pair of vertices in the digraph of R .

3.9 Definition A relation R on a set X is called *transitive* if, for all $x, y, z \in X$, whenever (x, y) and (y, z) are both in R , then $(x, z) \in R$, ie xRy and yRz imply xRz .

Example 15

(1) Let R be the relation \leq on $X = \{1, 2, 3, 4\}$ as already considered. Then R is transitive (see the discussion on Page 96 of Text). Eg. both $(1, 2)$ and $(2, 3)$ are in R and so is

(1, 3) etc.

This example is a special case of the familiar fact that \leq on any set X of numbers satisfies: if $x \leq y$ and $y \leq z$, then $x \leq z$, ie \leq is transitive in general.

(2) Take X to be the set $\{1, 2, 3, 4\}$. Then the relation $R = \{(1, 1), (1, 2), (2, 3)\}$ is not transitive because (1, 2) and (2, 3) are in R , but (1, 3) is not.

(3) Let X be any set and let R be the relation of equality, $=$, on X . Then, whenever $x = y$ and $y = z$, we have $x = z$ ie xRy and yRz imply xRz and so R is transitive.

(4) Let X be the power set $\mathcal{P}(A)$ of a non-empty set A . Then the relation R of subset inclusion, ie CRD iff $C \subseteq D$, is transitive because if $C \subseteq D$ and $D \subseteq E$, then $C \subseteq E$. (Suppose $x \in C$ is arbitrary. Then $x \in D$ because $C \subseteq D$. Therefore $x \in E$ because $D \subseteq E$ and thus $C \subseteq E$).

Exercise

Show that a relation R on a set X is transitive iff whenever there are directed edges (in the digraph of R) from x to y and from y to z , there is a directed edge from x to z .

3.10 Partial Orders

3.11 Definition A relation R on a set X is called a *partial order* if R is:

- (i) Reflexive (xRx for all $x \in X$).
- (ii) Antisymmetric (xRy and $yRx \Rightarrow x = y$ for all $x, y \in X$).
- (iii) Transitive (xRy and $yRz \Rightarrow xRz$ for all $x, y, z \in X$).

Example 16

(1) Let R be the relation defined on N by xRy iff $x \leq y$ ie R is “ \leq ”. Then R is a partial order. We check this next:

(i) xRx for all $x \in N$ because $x \leq x$ for all $x \in N$ ie R is reflexive.

(ii) Suppose xRy and yRx . Then $x \leq y$ and $y \leq x$. Therefore, $x = y$, and so R is antisymmetric.

(iii) Suppose xRy and yRz . Then $x \leq y$ and $y \leq z$. Therefore $x \leq z$. Thus xRz and so R is transitive. Hence R is a partial order.

(2) let R be the relation defined on $N^+ = \{n \in N; n > 0\}$ by xRy iff x divides y exactly. Then R is a partial order:

(i) x divides itself exactly and therefore xRx for all $x \in N^+$; so R is reflexive.

(ii) Suppose xRy and yRx . Then x divides y exactly and

hence $x \leq y$. Also y divides x exactly and hence $y \leq x$. Therefore, $x = y$ and so R is antisymmetric.

(iii) Suppose xRy and yRz . Then x divides y exactly ie $y = k_1x$ for some $k_1 \in N^+$. Also y divides z exactly ie $z = k_2y$ for some $k_2 \in N^+$. Therefore, $z = k_2(k_1x) = (k_2k_1)x = kx$ for some $k \in N^+$. Therefore, x divides z exactly. Thus, R is transitive. Hence R is a partial order.

(3) Let $X = \mathcal{P}(A)$ be the power set of a non-empty set A and let R denote the relation of subset inclusion on X ie CRD iff $C \subseteq D$. Then R is a partial order on X :

(i) R is reflexive because $C \subseteq C$ ie CRD for all $C \in X$.

(ii) R is antisymmetric because whenever $C \subseteq D$ and $D \subseteq C$, we have $C = D$ ie CRD and DRC imply $C = D$.

(iii) R is transitive because whenever $C \subseteq D$ and $D \subseteq E$, we have $C \subseteq E$ ie CRD and DRE imply CRE .

Note

Partial orders can be used to order or compare elements of a set X . This idea is of extreme importance in computing, for example in:

(a) Task scheduling (in operating systems). See discussion on Page 97 and 98 in Text.

(b) In ordering information, knowledge and truth.

(c) In programming language semantics, etc.

(4) (This example is fundamental in programming language semantics). Let X denote the set of all partial functions $f : N \rightarrow N$ ie $X = \{f : N \rightarrow N; f \text{ is a partial function}\}$. We can order X as follows. Given two functions f and g in X we write fRg (or $f \leq g$) iff $\text{graph}(f) \subseteq \text{graph}(g)$. (Recall that $\text{graph}(f) = \{(n, f(n)); n \in \text{dom}(f)\} \subseteq N \times N$).

This allows us to compare functions and indeed R is a partial order:

(i) For any $f \in X$, fRf ie $f \leq f$ because $\text{graph}(f) \subseteq \text{graph}(f)$.

(ii) Suppose $f, g \in X$ and fRg and gRf ie $f \leq g$ and $g \leq f$. Then $\text{graph}(f) \subseteq \text{graph}(g)$ and $\text{graph}(g) \subseteq \text{graph}(f)$. Therefore $\text{graph}(f) = \text{graph}(g)$. We show that this forces $f = g$.

Suppose $n \in \text{dom}(f)$ is arbitrary. Then $f(n)$ is defined, with value $y = f(n)$ say, and $(n, y) = (n, f(n)) \in \text{graph}(f) = \text{graph}(g)$. Therefore $(n, y) \in \text{graph}(g)$. Therefore, $g(n)$ is defined and equals y ie $g(n) = y$. Therefore we have $\text{dom}(f) \subseteq \text{dom}(g)$ and $f(n) = g(n)$ for all $n \in \text{dom}(f)$.

Conversely, suppose $n \in \text{dom}(g)$ is arbitrary. By the same argument we get $\text{dom}(g) \subseteq \text{dom}(f)$ and $g(n) = f(n)$ for all $n \in \text{dom}(g)$. Therefore $\text{dom}(f) = \text{dom}(g) = U$, say, and $f(n) = g(n)$ for all $n \in U$. Therefore $f = g$ and so R is antisymmetric.

(iii) For any $f, g, h \in X$, if $f R g$, and $g R h$, then $f R h$ if $f \leq g$ and $g \leq h$, then $f \leq h$. To see this, we have $\text{graph}(f) \subseteq \text{graph}(g)$ and $\text{graph}(g) \subseteq \text{graph}(h)$ ie $\text{graph}(f) \subseteq \text{graph}(g) \subseteq \text{graph}(h)$ and therefore $\text{graph}(f) \subseteq \text{graph}(h)$. Thus, $f R h$.

Hence R is a partial order on X .

Remark

(i) All this extends immediately to the class $X = \{f : N^n \rightarrow N; f \text{ is a partial function}\}$ of all partial functions of arity n .

(ii) This example can be used to give abstract models of computation. Starting with some given function f_1 (maybe the least or nowhere defined function \perp) and by successively applying some algorithm (iteration or recursion) we obtain a sequence of functions f_n . This sequence has the property:
 $f_1 \leq f_2 \leq f_3 \leq \dots \leq f_n \leq f_{n+1} \leq \dots$

where “ \leq ” is as defined above. It also has a “limit” or least upper bound f , say, ie

$$f_1 \leq f_2 \leq f_3 \leq \dots \leq f_n \leq f_{n+1} \leq \dots \leq f$$

and f is “smallest” with respect to \leq . We think of the f_n as giving better and better approximations to f (which is what we are trying to compute).

In fact, in general terms, each recursive definition D (or recursive programming scheme) has associated with it an operator or function $T_D : X \rightarrow X$, where $X = \{f : N^n \rightarrow N; f \text{ is a partial function}\}$, and it can be shown that f is the least fixed point of T_D ie $T_D(f) = f$ and f is the smallest function (wrt \leq) with this property. This least fixed point of T_D is often taken to be the “meaning” or “semantics” of the recursive definition D .

Example 17 (Greatest common divisor gcd of positive numbers a, b – this is a particular case of the previous example).

Let $gcd(a, b)$ denote the greatest common division of a and b (i.e. the largest number c s.t. c divides both a and b exactly). The usual definition of this (see later) depends on the division algorithm i.e. we write

$a = qt(a, b)b + r(a, b)$, where $qt(a, b)$ is the number of times b divides into a , and $r(a, b)$ is the remainder when a is divided by b ($0 \leq r(a, b) < b$). Then gcd is defined recursively by

$$gcd(a, b) = \begin{cases} b & \text{if } r(a, b) = 0, \\ gcd(b, r(a, b)) & \text{otherwise.} \end{cases}$$

(Code for this, say in Pascal, will contain a line of the form “ $gcd := gcd(y, z)$ ”).

Question.

“What does this ‘definition’ mean?” (the problem is that the quantity being defined, gcd , occurs on both sides of the equals sign).

Answer. The least fixed point of the operator T_D we associate with this recursive definition (the point is that this itself does not use recursion).

3.12 Definition (see Page 97 of Text)

Suppose R is a partial order on a set X . If $x, y \in X$ and either xRy or yRx , we say x and y are *comparable*. If $x, y \in X$ and both $(x, y) \notin R$ and $(y, x) \notin R$, then we say that x and y are *incomparable*. If every pair x, y of elements in X is comparable, we call R a *total order*.

Example 18

(1) Let X be any non-empty set and let $R = X \times X$ ie $R = \{(x, y); x \in X, y \in X\}$ so that all pairs $(x, y) \in R$. Thus, every pair x, y is comparable (R is not a total order because R is not antisymmetric).

(2) Let $X = N$ and let R be the partial order determined by \leq . Then R is a total order since for every pair $x, y \in N$ either $x \leq y$ or $y \leq x$ ie either xRy or yRx . Thus, every pair x, y is comparable and R is a total order.

(3) Let $X = N^+$ and let R be the partial order defined on X by xRy iff x divides y exactly. Take $x = 2$ and $y = 3$. Then $(2, 3) \notin R$ and $(3, 2) \notin R$ so 2 and 3 are incomparable. Take $x = 2$ and $y = 4$. Then 2 divides 4 and therefore 2 and 4 are comparable (note that $(4, 2) \notin R$). Thus, R is not a total order.

(4) Let $A = \{1, 2, 3, 4\}$ and let X be the power set $\mathcal{P}(A)$ of A . Let R be the partial order defined on X by subset inclusion ie CRD iff $C \subseteq D$. Take $C = \{1\}$ and $D = \{2\}$. Then $C \not\subseteq D$ and $D \not\subseteq C$ so C and D are incomparable. Take $C = \{1\}, D = \{1, 2\}$. Then $C \subseteq D$ and so C and D are comparable (note that it is not the case that (DRC)). Again, R is not a total order.

3.13 Further definitions and properties of relations.

3.14 Definition Let R be a relation from X to Y (ie $R \subseteq X \times Y$). The *inverse* of R , denoted R^{-1} , is the relation from Y to X defined by $R^{-1} = \{(y, x) \in Y \times X; (x, y) \in R\}$.

Example 19

(1) Take $X = N$ and let R be the relation from X to X determined by \leq ie $R = \{(x, y) \in N \times N; x \leq y\}$. Then R^{-1} is the relation from X to X determined by \geq ie. $R^{-1} = \{(y, x); xRy\} = \{(y, x); x \leq y\} = \{(y, x); y \geq x\}$.

Eg. $(1, 2) \in R$ so $(2, 1) \in R^{-1}$; $(2, 1) \notin R$ so $(1, 2) \notin R^{-1}$.

(2) Take $X = \{2, 3, 4, 5, 6\}$ and let R be the relation from X to X defined by xRy iff x divides y exactly. So

$$R = \{(2, 2), (2, 4), (2, 6), (3, 3), (3, 6), (4, 4), (5, 5), (6, 6)\}.$$

Therefore,

$$R^{-1} = \{(2, 2), (4, 2), (6, 2), (3, 3), (6, 3), (4, 4), (5, 5), (6, 6)\}.$$

Hence, $yR^{-1}x$ iff y is exactly divisible by x .

(3) Let A be a non-empty set and let X be its power set $\mathcal{P}(A)$. Let R be the relation defined on X by CRD iff $C \subseteq D$ ie R is subset inclusion. Then $R^{-1} = \{(D, C); D \supseteq C\}$ ie R^{-1} is defined by subset containment.

Example 20

Suppose $X = \{0, 1, 2, 3\}$, $Y = \{1, 2, 3, 4, 5\}$.

Let $R = \{(0, 1), (0, 2), (1, 1), (3, 5)\}$ and $S = \{(2, 1), (3, 5)\}$ be relations from X to Y as indicated. Evaluate each of the following:

- (1) R^{-1}
- (2) S^{-1}
- (3) $(R^{-1})^{-1}$
- (4) $(S^{-1})^{-1}$

Solution

- (1) $R^{-1} = \{(1, 0), (2, 0), (1, 1), (5, 3)\}$
- (2) $S^{-1} = \{(1, 2), (5, 3)\}$
- (3) $(R^{-1})^{-1} = \{(0, 1), (0, 2), (1, 1), (3, 5)\} = R$
- (4) $(S^{-1})^{-1} = \{(2, 1), (3, 5)\} = S$

Note that we have

$$\text{dom}(R) = \{0, 1, 3\} = \text{image set}(R^{-1})$$

$$\text{image set}(R) = \{1, 2, 5\} = \text{dom}(R^{-1})$$

Exercise. Let R be any relation from X to Y . Show that

- (1) $\text{dom}(R) = \text{image set}(R^{-1})$
- (2) $\text{image set}(R) = \text{dom}(R^{-1})$

$$(3) (R^{-1})^{-1} = R.$$

3.15 Composition of relations

3.16 Definition (See Page 99 of Text).

Let R_1 be a relation from X to Y and R_2 be a relation from Y to Z . The *composition* of R_1 and R_2 , denoted by $R_2 \circ R_1$ (note the order – cf function composition) is the relation from X to Z defined by $R_2 \circ R_1 = \{(x, z) \in X \times Z; (x, y) \in R_1 \text{ and } (y, z) \in R_2 \text{ for some } y \in Y\}$. Thus, $x(R_2 \circ R_1)z$ iff xR_1y and yR_2z for some $y \in Y$.

Example 21

Suppose that $X = \{1, 2, 3\}$, $Y = \{2, 3, 5, 7, 11\}$ and $Z = \{12, 16, 21\}$. Suppose R_1 is the relation from X to Y defined by

$$R_1 = \{(1, 2), (1, 3), (1, 11), (2, 3), (2, 5), (2, 7), (2, 11)\}$$

and R_2 is the relation from Y to Z defined by

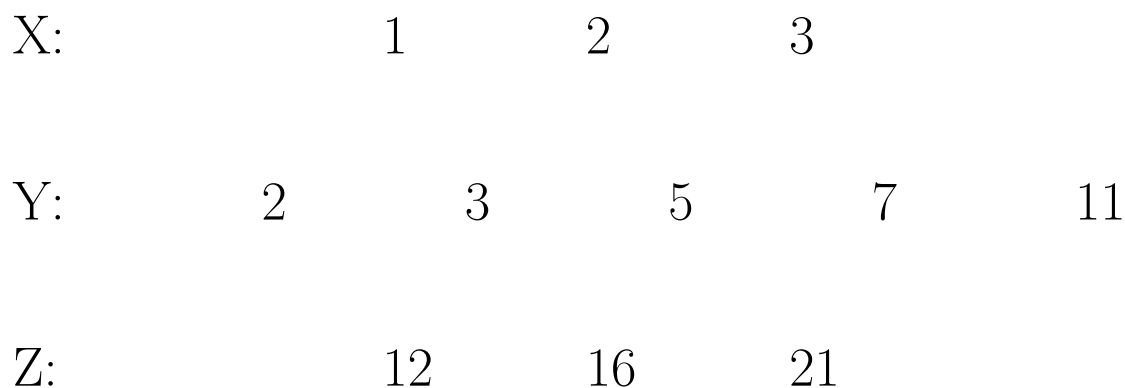
$$R_2 = \{(2, 12), (2, 16), (3, 12), (3, 21), (7, 21)\}.$$

Find $R_2 \circ R_1$.

Solution.

Applying the definition, we take each $x \in X$ and each $z \in Z$ and check whether or not there is a $y \in Y$ s.t. $(x, y) \in R_1$

and $(y, z) \in R_2$. Eg. $(1, 2) \in R_1$ and $(2, 12) \in R_2$ therefore $(1, 12) \in R_2 \circ R_1$. This is cumbersome and it is easier to use a slight generalization of the digraph:



We can read off $R_2 \circ R_1$ from this diagram:

$$R_2 \circ R_1 = \{(1, 12), (1, 16), (1, 21), (2, 12), (2, 21)\}$$

by looking for pairs x, z “connected by a path”.

Example 22 (Page 99 of Text)

Find the composition of the relations

$$R_1 = \{(1, 2), (1, 6), (2, 4), (3, 4), (3, 6), (3, 8)\}$$

and

$$R_2 = \{(2, u), (4, s), (4, t), (6, t), (8, u)\}$$

Solution.

Again we use the digraph:

X:	1	2	3	R_1
Y:	2	4	6	R_2
Z:	s	t	u	

So $R_2 \circ R_1 = \{(1, u), (1, t), (2, s), (2, t), (3, s), (3, t), (3, u)\}$.

Note that (as with functions) it does not necessarily make sense to talk about the composition in the opposite order. Thus, in the previous example $R_2 \circ R_1$ is defined because R_1 is from X to Y and R_2 from Y to Z so that $R_2 \circ R_1$ from X to Z is defined. But $R_1 \circ R_2$ is not defined because $X \neq Z$ and R_2 goes from Y to Z and R_1 from X to Y and we need $X = Z$ to “jump”.

Example 23

Suppose $X = \{0, 1, 2\}$ and R and S are the relations from X to X defined by

$$R = \{(0, 0), (0, 2), (1, 1)\}, \quad S = \{(1, 2), (2, 1)\}.$$
 Evaluate:

- (1) $(S \circ R)^{-1}$
- (2) $S^{-1} \circ R^{-1}$
- (3) $R^{-1} \circ S^{-1}$
- (4) $R^{-1} \circ R$
- (5) $R \circ R^{-1}$

Solution.

Note that because both R and S go from X to X , all composites involved are defined.

(1) $(S \circ R)^{-1}$

$$X: \quad \quad 0 \quad \quad 1 \quad \quad 2$$

$$X: \quad \quad 0 \quad \quad 1 \quad \quad 2$$

$$X: \quad \quad 0 \quad \quad 1 \quad \quad 2$$

So $S \circ R = \{(0, 1), (1, 2)\}$ hence $(S \circ R)^{-1} = \{(1, 0), (2, 1)\}$.

(2) $S^{-1} \circ R^{-1}$. We have $R^{-1} = \{(0, 0), (2, 0), (1, 1)\}$ and $S^{-1} = \{(2, 1), (1, 2)\}$, so:

$$X: \quad \quad 0 \quad \quad 1 \quad \quad 2$$

$$X: \quad \quad 0 \quad \quad 1 \quad \quad 2$$

$$X: \quad \quad 0 \quad \quad 1 \quad \quad 2$$

Therefore $S^{-1} \circ R^{-1} = \{(1, 2)\}$.

(3) $R^{-1} \circ S^{-1}$

X: 0 1 2

X: 0 1 2

X: 0 1 2

Therefore $R^{-1} \circ S^{-1} = \{(1, 0), (2, 1)\}$.

(4) $R^{-1} \circ R$

X: 0 1 2

X: 0 1 2

X: 0 1 2

So $R^{-1} \circ R = \{(0, 0), (1, 1)\}$.

(5) $R \circ R^{-1}$

X: 0 1 2

X: 0 1 2

X: 0 1 2

Therefore $R \circ R^{-1} = \{(0, 0), (0, 2), (1, 1), (2, 0), (2, 2)\}$.

Note. From this example we see that $R^{-1} \circ R \neq R \circ R^{-1}$ and that $(S \circ R)^{-1} \neq S^{-1} \circ R^{-1}$. In fact, $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$.

Exercise

Show that for any relations R and S for which $S \circ R$ is defined, we have $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$.

Relation Override

3.17 Definition Let R and S be relations. The *override of R by S* , denoted by $R \oplus S$, is the relation obtained by adding to S all those ordered pairs from R whose first coordinates are not in the domain of S .

Example 24

Suppose S and R are the relations $R = \{(0, 1), (0, 2), (2, 3)\}$

and $S = \{(0, 1), (1, 3), (3, 0)\}$. Find $R \oplus S$ and $S \oplus R$.

Solution

$$R \oplus S = \{(0, 1), (1, 3), (3, 0), (2, 3)\}$$

$$S \oplus R = \{(0, 1), (0, 2), (2, 3), (1, 3), (3, 0)\}$$

Note. Relation override can be used to model “update of records” in formal specification c.f. VDM.

3.18 Equivalence Relations (see Page 104 of Text).

3.19 Definition Let X be a non-empty set. A relation R on X is called an *equivalence relation* if R is reflexive, symmetric and transitive.

Notation

For a non-empty set X , let $I_X = \{(x, x); x \in X\} \subseteq X \times X$ denote the identity relation on X .

3.20 Proposition Let X be any non-empty set.

- (1) A relation R on X is reflexive iff $I_X \subseteq R$.
- (2) A relation R on X is symmetric iff $R^{-1} = R$.
- (3) A relation R on X is transitive iff $R \circ R \subseteq R$.

Proof

(1) Suppose R is reflexive. Thus, $(x, x) \in R$ for each $x \in X$. Therefore, $\{(x, x); x \in X\} \subseteq R$; ie $I_X \subseteq R$. Conversely,

suppose $I_X \subseteq R$. Let $x \in X$. Then $(x, x) \in I_X$ and hence $(x, x) \in R$ since $I_X \subseteq R$. Thus, $(x, x) \in R$ for each $x \in X$ and so R is reflexive.

(2) Suppose R is symmetric. Let $(x, y) \in R$ be arbitrary. Then $(y, x) \in R$ by symmetry. Therefore $(x, y) \in R^{-1}$ by definition of R^{-1} . Thus, $R \subseteq R^{-1}$. Now suppose $(y, x) \in R^{-1}$ is arbitrary. Then $(x, y) \in (R^{-1})^{-1} = R$ by definition of R^{-1} and the exercise showing $(R^{-1})^{-1} = R$. By symmetry we therefore get $(y, x) \in R$ and hence $R^{-1} \subseteq R$. Thus, $R = R^{-1}$.

Conversely, suppose $R = R^{-1}$. Let $(x, y) \in R$ be arbitrary. Then $(x, y) \in R^{-1}$ since $R = R^{-1}$. Therefore $(y, x) \in (R^{-1})^{-1} = R$ and so R is symmetric.

(3) Suppose R is transitive. Let $(x, z) \in R \circ R$ be arbitrary. Then for some $y \in X$ we have $(x, y) \in R$ and $(y, z) \in R$. By transitivity we then have $(x, z) \in R$. Therefore $R \circ R \subseteq R$. Conversely, suppose $R \circ R \subseteq R$. Let $(x, y) \in R$ and $(y, z) \in R$. Then $(x, z) \in R \circ R$ by definition of \circ . But $R \circ R \subseteq R$ and hence $(x, z) \in R$. Thus, R is transitive as required. ■

Example 25

Let X be the set $\{1, 2, 3\}$. Which of the following relations on X are equivalence relations?

(1) $\{(1, 1), (1, 2), (2, 1), (2, 2), (3, 3)\}$

(2) $\{(1, 1), (2, 2), (3, 3)\}$

(3) $\{(1, 2), (2, 3), (1, 3)\}$.

Solution

(1) This relation is clearly reflexive. Moreover its inverse is $\{(1, 1), (2, 1), (1, 2), (2, 2), (3, 3)\}$ which is itself. Therefore by (2) of Prop. 1 the relation is symmetric. Finally, to determine whether or not R is transitive we compute $R \circ R$ and use (3) of Prop. 3.20 for this relation R .

X:	1	2	3
----	---	---	---

X:	1	2	3
----	---	---	---

X:	1	2	3
----	---	---	---

$R \circ R = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 3)\} = R \subseteq R$ and so R is transitive. Thus, the relation in (1) is an equivalence relation.

(2) $\{(1, 1), (2, 2), (3, 3)\}$ is clearly reflexive, and is symmetric since $R^{-1} = R$. For transitivity, we calculate $R \circ R$:

X: 1 2 3

X: 1 2 3

X: 1 2 3

$R \circ R = \{(1, 1), (2, 2), (3, 3)\} = R$. Thus, R is transitive, since $R \circ R \subseteq R$, by (3) of Prop. 3.20. Thus, R is an equivalence relation.

(3) $\{(1, 2), (2, 3), (1, 3)\}$ is not reflexive eg $(1, 1) \notin R$; it is not symmetric eg $(1, 2) \in R$ but $(2, 1) \notin R$. However, this relation is transitive:

X: 1 2 3

X: 1 2 3

X: 1 2 3

$R \circ R = \{(1, 3)\} \subset R$ and thus R is transitive. Thus, R is not an equivalence relation.

3.21 Closures of Relations

3.22 Definition Let R be a relation on a set X . The *reflexive closure* of R is the smallest reflexive relation on X which contains R (smallest in the sense that if R' is any reflexive relation on X which contains R , then R' contains the reflexive closure).

3.23 Proposition The reflexive closure of R is $R \cup I_X$.

Proof

First we note that $R \cup I_X \subseteq X \times X$ and hence $R \cup I_X$ is a relation on X . Next, $R \cup I_X$ is reflexive by Prop. 3.20 (1) since $I_X \subseteq R \cup I_X$. Also, $R \cup I_X$ contains R since $R \subseteq R \cup I_X$. Thus $R \cup I_X$ is a reflexive relation on X containing R .

Finally, we show $R \cup I_X$ is smallest with these properties. Suppose R' is any relation on X which contains R and is reflexive. Then immediately $R \subseteq R'$. Since R' is reflexive, we have $I_X \subseteq R'$ by Prop. 3.20 (1) again. Therefore $R \cup I_X \subseteq R'$ and so $R \cup I_X$ is smallest with the required properties. ■

Example 26

Find the reflexive closure of the relation $R = \{(1, 1), (1, 2)\}$ on $X = \{1, 2\}$.

Solution The reflexive closure is $R \cup I_X = \{(1, 1), (1, 2)\} \cup$

$$\{(1, 1), (2, 2)\} = \{(1, 1), (1, 2), (2, 2)\}.$$

3.24 Definition Let R be a relation on X . The *symmetric closure* of R is the smallest symmetric relation on X which contains R .

3.25 Proposition The symmetric closure of R is $R \cup R^{-1}$.

Proof First note that $R \cup R^{-1} \subseteq X \times X$ and is therefore a relation on X . Also, we have $R \subseteq R \cup R^{-1}$ and so $R \cup R^{-1}$ contains R .

Next we show $R \cup R^{-1}$ is symmetric. Suppose $(x, y) \in R \cup R^{-1}$ is arbitrary. Then by definition of union, either $(x, y) \in R$ or $(x, y) \in R^{-1}$.

Case 1 Suppose $(x, y) \in R$. Then $(y, x) \in R^{-1}$, by definition of R^{-1} , and so $(y, x) \in R \cup R^{-1}$.

Case 2 Suppose $(x, y) \in R^{-1}$. Then $(y, x) \in (R^{-1})^{-1} = R$ (by earlier exercise) and so $(y, x) \in R \cup R^{-1}$. Therefore $R \cup R^{-1}$ is a symmetric relation on X containing R .

Finally, suppose R' is any symmetric relation on X which contains R . Then immediately we have $R \subseteq R'$. Therefore $R^{-1} \subseteq R'^{-1}$. But $R'^{-1} = R'$ since R' is symmetric, by Prop. 3.20 (2). Therefore $R^{-1} \subseteq R'$ and hence $R \cup R^{-1} \subseteq R'$ and so $R \cup R^{-1}$ is the smallest symmetric relation containing R ie the symmetric closure of R . ■

Example 27.

Find the symmetric closure of $R = \{(1, 1), (1, 2)\}$.

Solution. $R^{-1} = \{(1, 1), (2, 1)\}$. Thus, the symmetric closure of R is $R \cup R^{-1} = \{(1, 1), (1, 2)\} \cup \{(1, 1), (2, 1)\} = \{(1, 1), (1, 2), (2, 1)\}$.

3.26 Definition Let R be a relation on a set X . The *symmetric-reflexive closure* of R is the smallest symmetric and reflexive relation on X which contains R .

3.27 Proposition The symmetric-reflexive closure of R is $R \cup I_X \cup R^{-1}$.

Proof Exercise (could be an exam question.) ■

Example 28

Find the reflexive, symmetric and symmetric-reflexive closures of the following relations on $X = \{1, 2, 3\}$.

(1) $R = \{(1, 1), (1, 3), (2, 1)\}$

(2) $S = \{(1, 2), (1, 3), (2, 1)\}$

(3) $T = \{(1, 1)\}$

(4) \emptyset .

Solution. Here $I_X = \{(1, 1), (2, 2), (3, 3)\}$

(1) Reflexive closure of $R = \{(1, 1), (1, 3), (2, 1)\}$ is

$$\begin{aligned} R \cup I_X &= \{(1, 1), (1, 3), (2, 1)\} \cup \{(1, 1), (2, 2), (3, 3)\} \\ &= \{(1, 1), (1, 3), (2, 1), (2, 2), (3, 3)\}. \end{aligned}$$

$$\begin{aligned}
& \text{Symmetric closure of } R \text{ is } R \cup R^{-1} \\
& = \{(1, 1), (1, 3), (2, 1)\} \cup \{(1, 1), (3, 1), (1, 2)\} \\
& = \{(1, 1), (1, 3), (2, 1), (3, 1), (1, 2)\}.
\end{aligned}$$

Symmetric-reflexive closure of R is

$$\begin{aligned}
R \cup I_X \cup R^{-1} & = \{(1, 1), (1, 3), (2, 1), (2, 2), (3, 3)\} \\
& \quad \cup \{(1, 1), (3, 1), (1, 2)\} \\
& = \{(1, 1), (1, 3), (2, 1), (2, 2), (3, 3), (3, 1), (1, 2)\}.
\end{aligned}$$

(2) Reflexive closure of $S = \{(1, 2), (1, 3), (2, 1)\}$ is $S \cup I_X = \{(1, 2), (1, 3), (2, 1), (1, 1), (2, 2), (3, 3)\}$.

Symmetric closure of S is $S \cup S^{-1} = \{(1, 2), (1, 3), (2, 1)\} \cup \{(2, 1), (3, 1), (1, 2)\}$
 $= \{(1, 2), (1, 3), (2, 1), (3, 1)\}$.

The symmetric-reflexive closure is $S \cup I_X \cup S^{-1}$
 $= \{(1, 2), (1, 3), (2, 1), (1, 1), (2, 2), (3, 3)\}$
 $\cup \{(2, 1), (3, 1), (1, 2)\}$
 $= \{(1, 2), (1, 3), (2, 1), (1, 1), (2, 2), (3, 3), (3, 1)\}$.

(3) $T = \{(1, 1)\}$ is already symmetric ($T = T^{-1}$) and hence is equal to its symmetric closure. Therefore, its reflexive and symmetric-reflexive closures are both equal to $\{(1, 1)\} \cup I_X = \{(1, 1), (2, 2), (3, 3)\}$.

(4) \emptyset is already symmetric and hence equal to its symmetric closure. The reflexive and symmetric-reflexive closures are both equal to $\emptyset \cup I_X = \{(1, 1), (2, 2), (3, 3)\}$.

3.28 Transitive Closure

We first consider repeated composition.

Example 29.

Let X be the set $\{1, 2, 3\}$ and let R be the relation on X defined by

$R = \{(1, 2), (2, 3), (3, 2)\}$. Find:

- (1) $R \circ R$
- (2) $(R \circ R) \circ R$
- (3) $R \circ (R \circ R)$

Solution.

For (1):

X:	1	2	3
----	---	---	---

X:	1	2	3
----	---	---	---

X:	1	2	3
----	---	---	---

So $R \circ R = \{(1, 3), (2, 2), (3, 3)\}$.

For (2):

X: 1 2 3

X: 1 2 3

X: 1 2 3

So $(R \circ R) \circ R = \{(1, 2), (2, 3), (3, 2)\}$

For (3):

X: 1 2 3

X: 1 2 3

X: 1 2 3

So $R \circ (R \circ R) = \{(1, 2), (2, 3), (3, 2)\}$

Note that $(R \circ R) \circ R$ and $R \circ (R \circ R)$ are equal.

3.29 Proposition Composition of relations is associative.

Thus, if R_1, R_2, R_3 are relations on X , then $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$.

Proof. Exercise (c.f. case of functions). ■

In particular, if R is a relation on X , then $(R \circ R) \circ R = R \circ (R \circ R)$. Therefore, we can write $R \circ R \circ R$ without ambiguity since the two possible ways of putting brackets into $R \circ R \circ R$ give the same results.

Notation.

We write $R \circ R$ as R^2 , $R \circ R \circ R$ as R^3 , ..., in general $R \circ R \circ \dots \circ R$ (with n R s) as R^n .

The easiest way to interpret repeated composition of a relation is to think in terms of the digraph for the relation. For example suppose that we take $X = \{1, 2, 3\}$ and the relation $R = \{(1, 2), (2, 3), (3, 2)\}$ on X as above. The digraph for this relation is

X:	1	2	3
----	---	---	---

X:	1	2	3
----	---	---	---

The relation R itself corresponds to pairs of points the second of which can be reached in one “jump” from the first; for example we can move from 1 to 2 along the arc (ie directed edge) x , or from 2 to 3 along y , or from 3 to 2 along arc z .

The relation $R^2 = R \circ R$ is

X: 1 2 3

X: 1 2 3

X: 1 2 3

$R^2 = \{(1, 3), (2, 2), (3, 3)\}$ is the set of pairs of values which can be linked by exactly two jumps. Thus we can move from 1 to 3 by arcs x and y ; we can move from 2 to 2 by arcs y and z ; and from 3 to 3 by arcs z and y . Hence $(1, 3)$, $(2, 2)$ and $(3, 3)$ are elements of R^2 , as we see. Note that 1 and 2 cannot be linked in exactly two jumps, but they can in 3 jumps because $(1, 2) \in R^3 (= R \circ (R \circ R) = (R \circ R) \circ R)$ as we saw above.

Exercise.

Show that $(1, 2) \in R$, $(1, 2) \in R^3$, $(1, 2) \in R^5, \dots, (1, 2) \in R^n$ for any n odd (Note this shows that a pair of values may be linked in more than one way, in fact maybe in infinitely many ways); and that $(1, 2) \notin R^n$ for n even. Hint. use

induction.

In fact, R^3 is the set of pairs linked by 3 jumps, R^4 is the set of pairs linked by 4 jumps, and so on.

3.30 Definition Let X be a set and R a relation on X . We say that a point $b \in X$ is *reachable* (wrt R) from a point $a \in X$ (perhaps the same point b) if there is a value of k s.t. $(a, b) \in R^k$ ie if b can be reached from a in k steps. The concept of reachability itself defines a relation R^+ on X (where aR^+b iff b is reachable from a) called the *transitive closure* of R . Thus, $R^+ = R \cup R^2 \cup R^3 \cup \dots \cup R^n \cup \dots$

Note that if R is finite ie a finite set, then $R^+ = R \cup R^2 \cup \dots \cup R^n$, where n is the maximum number of jumps necessary between reachable points (n clearly exists if R is finite).

We have the following proposition.

3.31 Proposition Let R be a relation on a set X . Then R^+ is the smallest transitive relation on X which contains R (ie R^+ has the properties expected of the term “transitive closure”).

Proof

$$R^+ = R \cup R^2 \cup R^3 \cup \dots \cup R^n \cup \dots$$

Since $R^k \subseteq X \times X$ for each k , we have $R^+ \subseteq X \times X$ ie

R^+ is a relation on X . It is also clear that $R \subseteq R^+$ ie R^+ contains R .

To show R^+ is transitive: suppose $(x, y) \in R^+$ and $(y, z) \in R^+$. Then we can reach y from x in finitely many jumps and we can reach z from y in finitely many jumps:

$$x \qquad \qquad \qquad y \qquad \qquad \qquad z$$

Therefore we can reach z from x in finitely many jumps (the sum of the two numbers of jumps). Thus, $(x, z) \in R^+$ and R^+ is transitive.

To show R^+ is smallest with these properties: Suppose R' any transitive relation on X with $R \subseteq R'$. Since $R \subseteq R'$, we have $R \circ R \subseteq R' \circ R'$ (Exercise: check this). But R' is transitive and hence $R' \circ R' \subseteq R'$ by Prop. 3.20 (3) ie $R^2 \subseteq R'$. Similarly $R^3 = R \circ R^2 \subseteq R' \circ R' \subseteq R'$ so that $R^3 \subseteq R'$. In general (by induction) we get $R^k \subseteq R'$ for each k . Therefore $R^+ = R \cup R^2 \cup R^3 \cup \dots \cup R^n \cup \dots \subseteq R' \cup R' \cup R' \cup \dots \cup R' \cup \dots = R'$ ie $R^+ \subseteq R'$. So R^+ is the smallest transitive relation on X containing R . ■

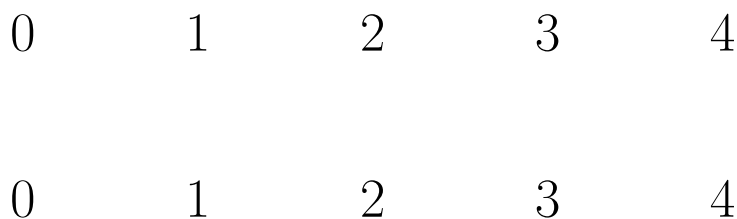
For “small” relations R , we can find the transitive closure by inspection of the digraph of R :

Example 30.

Find the transitive closure of the relation $R = \{(0, 1), (1, 3), (3, 4), (4, 2), (4, 0)\}$ on $X = \{0, 1, 2, 3, 4\}$.

Solution Note that the relation R is not transitive: we have $(0, 1) \in R$, $(1, 3) \in R$ but $(0, 3) \notin R$.

Digraph of R is:



From 0 we can “go” as follows:

0 to 1 to 3 to 4; from 4 go either to 2 (a “dead end”) or back to 0. Hence from 0 we can reach 1, 3, 4, 2, 0 (“cycle” now formed plus dead end - we get nothing more).

From 1 we can go: 1 to 3 to 4; from 4 to 2 (dead end) or to 0 to 1 (cycle now formed plus dead end). Hence from 1 we can reach 3, 4, 2, 0, 1.

From 2 we can reach nothing.

From 3 we can go: 3 to 4; from 4 to 2 (dead end) or to 0 to 1 to 3 (cycle now formed). Hence, from 3 we can reach 4, 2, 0, 1, 3.

From 4 we can go: 4 to 2 (dead end); or to 0 to 1 to 3 to 4

(cycle now formed). Hence from 4 we can reach 2, 0, 1, 3, 4.

Thus, R^+ is

$\{(0, 1), (0, 3), (0, 4), (0, 2), (0, 0)$
 $(1, 3), (1, 4), (1, 2), (1, 0), (1, 1)$
 $(3, 4), (3, 2), (3, 0), (3, 1), (3, 3)$
 $(4, 2), (4, 0), (4, 1), (4, 3), (4, 4)\}$

3.32 Transitive - reflexive closure

Transitive closure finds application in problems concerning networks, binary trees etc. Often we also want each point to be reachable from itself, which leads to:

3.33 Definition The *transitive-reflexive closure* of a relation R on X is the smallest transitive and reflexive relation on X containing R ; it is denoted by R^* .

3.34 Proposition For any relation R on X , we have $R^* = R^+ \cup I_X$.

Proof Exercise (again a good exam question). ■

Example 31 If $X = \{0, 1, 2, 3, 4\}$ and $R = \{(0, 1), (1, 3), (3, 4), (4, 2), (4, 0)\}$ find R^* .

Solution.

Here $I_X = \{(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)\}$

In the previous example we found R^+ . Therefore $R^* = R^+ \cup$

I_X is

$\{(0, 1), (0, 3), (0, 4), (0, 2), (0, 0)$

$(1, 3), (1, 4), (1, 2), (1, 0), (1, 1)$

$(2, 2)$

$(3, 4), (3, 2), (3, 0), (3, 1), (3, 3)$

$(4, 2), (4, 0), (4, 1), (4, 3), (4, 4)\}$.

Note We can regard the identity relation I_X on a set X as relating points reachable in zero steps ie we can regard

$I_X = R^0$, and we can regard R itself as relating points reachable in 1 step ie $R = R^1$. Then we can write

$$R^* = R^0 \cup R^1 \cup R^2 \cup R^3 \cup \dots$$

In particular, for finite relations R

$$R^* = R^0 \cup R^1 \cup R^2 \cup \dots \cup R^n \text{ for some } n.$$

Exercise Read pp 118 - 122 of Textbook, on relational databases.

4 INTRODUCTION TO LOGIC

Logic is used in computer science in at least three ways:

1. To study switching circuits in hardware (Boolean algebra).
2. In conjunction with sets, functions and relations to give formal specification of systems i.e. to describe precisely how the parts of a system interact (validation).
3. As a programming language (e.g. Prolog, Parlog): logic is used as a knowledge representation language (in Artificial Intelligence – AI) to represent “data”, and computation is logical deduction from the data. Logic is also used to build automated theorem provers in verification of software (and in mathematics).

PART 1 Propositional Logic See pp 1 - 18 of Textbook.

4.1 Propositions

4.2 Definition A proposition is a sentence which is either true or false, but not both.

This definition is informal – a formal treatment involves a formal description of syntax or grammar in a formal lan-

guage (and ultimately a formal description of semantics i.e. of meaning).

Propositions are the basic building blocks of logic.

Example 1

The following are propositions:

(1) The cat sat on the mat. This is either true or false, although we don't know which, and is therefore a proposition.

(2) All dogs like bones. This is either true or false – probably false – and is a proposition.

(3) The only positive integers which divide 7 are 1 and 7 itself. This is a true proposition.

(4) For every positive integer n , there is a prime number larger than n (a prime number is a number n whose only divisions are 1 and n). This is a true proposition.

(5) Earth is the only planet in the universe that has life. This is either true or false, but not both, although no one knows which at this point in time.

(6) There are only finitely many perfect numbers (n is perfect if n is equal to the sum of the divisors of n less than n – e.g. 6, 28, 496,...). This is true or false (not yet known) and therefore is a proposition.

Example 2

The following are not propositions.

- (1) Buy two tickets for the concert! This is a command, not a proposition.
- (2) Will I pass my summer exams? This is a question, not a proposition.
- (3) The number 8 smells awful. This is meaningless.

Note The problem with this informal approach is that it requires the “external” notion of truth to determine what is and what is not a proposition, which is something which should be decided by syntax – this is why a formal approach really is needed.

4.3 Notation

We often use lower case letters p, q, r etc. to represent propositions. We use the notation $p : \dots$ to mean that we define p to be the proposition \dots .

Example 3

- (1) $p : 1 + 1 = 3$ means that p is the proposition $1 + 1 = 3$ (which is a proposition which happens to be false).
- (2) $q : \text{all dogs like bones}$ means that q is the proposition “all dogs like bones”.

Combining propositions

We consider ways of combining propositions using what are called the connectives \wedge , \vee , \neg (and later \rightarrow and \leftrightarrow) to obtain compound propositions.

The connective \wedge (and).

Consider the proposition $p : 1 + 1 = 3$ and the proposition $q : \text{all dogs like bones}$. These two separate propositions can be combined into a single proposition

$1 + 1 = 3$ and all dogs like bones

(this is false, but it is a proposition) called their conjunction and, in symbols, written $p \wedge q$.

4.4 Definition Let p and q be propositions. We define the *conjunction* $p \wedge q$ of p and q to be the proposition “ p and q ” ie \wedge is the formal symbol for and.

The connective \vee (or).

The two propositions above can be combined into the single proposition

$1 + 1 = 3$ or all dogs like bones

called their disjunction and, in symbols, written $p \vee q$.

4.5 Definition Let p and q be propositions. We define the *disjunction* $p \vee q$ of p and q to be the proposition “ p or q ” ie \vee is the formal symbol for or.

Note We mean here the inclusive or which allows both propositions to be true, not the exclusive or (exor) which allows either one to be true but not both simultaneously.

The connective \neg (negation).

A third operation on propositions that we consider is negation. Consider again the proposition $p : 1 + 1 = 3$. If we write $\text{not}(1 + 1 = 3)$ to mean that “it is not the case that $1 + 1 = 3$ ”, then this statement is itself a proposition (it clearly is either true or false, and in fact is true). In symbols we write either $\neg p$ or \bar{p} .

4.6 Definition Let p be a proposition. We define the *negation* $\neg p$ or \bar{p} of p to be the proposition “not p ” ie \neg or $\bar{}$ is the formal symbol for not.

Example 4

Suppose $p : 2 + 3 = 6$, $q : 7$ is prime (so p is false and q is true). Write out

(a) $p \wedge q$

(b) $p \vee q$

(c) $\neg p$

(d) $\neg q$

Solution

- (a) $p \wedge q$ is the proposition “ $2 + 3 = 6$ and 7 is prime”.
- (b) $p \vee q$ is the proposition “ $2 + 3 = 6$ or 7 is prime”.
- (c) $\neg p$ is the proposition “not ($2 + 3 = 6$)” i.e. $2 + 3$ is not equal to 6.
- (d) $\neg q$ is the proposition “not (7 is prime)” i.e. 7 is not prime.

Truth Tables.

Propositions built up from other propositions by use of the connectives \wedge, \vee, \neg (and later $\rightarrow, \leftrightarrow$) are called *compound propositions*. It is clear that the truth value (t for true, f for false) of a compound proposition made up of propositions p and q depends on the truth values of p and q . The connection is described by means of a truth table.

4.7 Definition The *truth table for conjunction* (ie for the compound proposition $p \wedge q$) is given by

p	q	$p \wedge q$
t	t	t
t	f	f
f	t	f
f	f	f

where all possible combinations of truth values t, f for p and q are considered. So $p \wedge q$ is true only when p is true and q is true; otherwise it is false.

4.8 Definition The *truth table for disjunction* (ie for the compound proposition $p \vee q$) is given by

p	q	$p \vee q$
t	t	t
t	f	t
f	t	t
f	f	f

so $p \vee q$ is only false when p is false and q is false; otherwise it is true.

4.9 Definition The *truth table for negation* ie for $\neg p$ is given by

p	$\neg p$
t	f
f	t

so $\neg p$ is true when p is false and $\neg p$ is false when p is true.

The connectives \rightarrow and \leftrightarrow .

4.10 Definition If p and q are propositions, then the compound proposition $p \rightarrow q$, called a *conditional proposition*, is defined by “if p then q ” or “ p only if q ” or “ p implies q ”. The connective \rightarrow is called *material implication*, p is called the *antecedent* and q the *consequent*.

Example 5

(a) Suppose p is the proposition p : John works hard, and q is the proposition q : John passes his exams. Then $p \rightarrow q$ is the compound proposition if John works hard, then John passes his exams.

(b) Suppose p is the proposition p : $2 + 3 = 5$, and q is the proposition q : $3 + 3 = 7$. Then $p \rightarrow q$ is the proposition if $2 + 3 = 5$, then $3 + 3 = 7$.

4.11 Definition The *truth table* for \rightarrow (ie for $p \rightarrow q$) is

p	q	$p \rightarrow q$
t	t	t
t	f	f
f	t	t
f	f	t

so $p \rightarrow q$ is false only when p is true and q is false. In all other cases, $p \rightarrow q$ is true.

Note

This table can be taken as the definition of the connective \rightarrow . Indeed, for each of the five connectives we consider: $\wedge, \vee, \neg, \rightarrow$ and \leftrightarrow , the truth table can be taken as defining the connective.

Example 6

(a) Suppose $p : 2 + 3 = 5$ and $q : 3 + 3 = 6$. Then p is true, q is true and so $p \rightarrow q$ is true.

(b) Suppose $p : 2 + 3 = 5$ and $q : 3 + 3 = 7$. Then p is true, q is false and hence $p \rightarrow q$ is false.

(c) Suppose $p : 3 + 3 = 7$ and $q : 2 + 2 = 4$. Then p is false, q is true and so $p \rightarrow q$ is true.

(d) Suppose $p : 3 + 3 = 7$, $q : 2 + 8 = 12$. Then p is false, q is false and so $p \rightarrow q$ is true.

4.12 Definition Given the compound proposition $p \rightarrow q$, we call the compound proposition $q \rightarrow p$ the *converse* of $p \rightarrow q$.

Example 7

Suppose $p : 2 + 3 = 5$ and $q : 3 + 3 = 6$. Then $p \rightarrow q$

is true by Example 6 (a). Also, $q \rightarrow p$ is true by the truth table for \rightarrow (since both q and p are true). So sometimes a proposition $p \rightarrow q$ and its converse are both true. Now let $p : 3 + 3 = 7$ and $q : 2 + 4 = 6$. Then $p \rightarrow q$ is true since p is false. However, q is true and so $q \rightarrow p$ is false in this case. So sometimes a proposition $p \rightarrow q$ is true, but its converse $q \rightarrow p$ is false.

4.13 Definition Suppose p and q are propositions. Then the compound proposition $p \leftrightarrow q$, called a *bi conditional proposition*, is defined by “ p if and only if q ” ie if p then q and if q then p ie $(p \rightarrow q) \wedge (q \rightarrow p)$ so that its form is conditional proposition and its converse.

Example 8

Suppose p : the triangle PQR is a right triangle, and q : in the triangle PQR, $PQ^2 + QR^2 = PR^2$. Then $p \leftrightarrow q$ is the proposition “the triangle PQR is a right triangle iff $PQ^2 + QR^2 = PR^2$ ”.

4.14 Definition The *truth table* for \leftrightarrow (ie for $p \leftrightarrow q$) is given by

p	q	$p \leftrightarrow q$
t	t	t
t	f	f
f	t	f
f	f	t

Example 9

(a) Suppose $p : 2 + 3 = 5$, $q : 3 + 3 = 6$. Then p, q both true so $p \leftrightarrow q$ true.

(b) Suppose $p : 2 + 3 = 5$, $q : 3 + 4 = 8$. Then p true, q false so $p \leftrightarrow q$ false.

(c) Suppose $p : 3 + 4 = 8$, $q : 2 + 3 = 5$. Then p false, q true so $p \leftrightarrow q$ false.

(d) Suppose $p : 2 + 3 = 6$, $q : 3 + 4 = 8$. Then both p, q are false so $p \leftrightarrow q$ true.

Compound propositions and truth tables

The connectives $\wedge, \vee, \neg, \rightarrow$ and \leftrightarrow can be applied iteratively to form compound propositions or arbitrary complexity.

Example 10

The following are examples of compound propositions:

(1) $(p \vee q) \rightarrow (r \vee s)$.

(2) $((p \wedge \neg q) \leftrightarrow (p_1 \vee p_2 \vee p_3)) \wedge \neg(q \vee \neg r)$.

$$(3) ((\neg p_1) \vee p_2) \rightarrow ((p_3 \wedge p_4) \rightarrow (p_1 \vee p_4))$$

etc.

A compound proposition P is made up of n propositions p_1, p_2, \dots, p_n if it involves the propositions p_1, p_2, \dots, p_n , and no others, and the connectives. The truth value of P can be found by repeated application of the truth tables once the truth value of each p_i is given:

4.15 Definition Let $\mathcal{P} = \{p_1, \dots, p_n\}$ be a set of n propositions. Then a mapping $TA : \mathcal{P} \rightarrow \{t, f\}$ is called a *truth assignment*. So $TA(p_i) = t$ or $TA(p_i) = f$ for $i = 1, 2, \dots, n$.

Exercise

Show that there are 2^n truth assignments defined on \mathcal{P} .

Example 11

If $\mathcal{P} = \{p_1, p_2\}$ ie $n = 2$, then there are $2^2 = 4$ truth assignments on \mathcal{P} : $p_1 \rightarrow t, p_2 \rightarrow t$; $p_1 \rightarrow t, p_2 \rightarrow f$; $p_1 \rightarrow f, p_2 \rightarrow t$; $p_1 \rightarrow f, p_2 \rightarrow f$.

Example 12

Find truth tables for each of the following:

(1) $\neg(\neg p)$.

(2) $(\neg p) \vee (p \wedge q)$.

(3) $(\neg p) \wedge (p \wedge q)$.

(4) $(\neg p) \rightarrow (p \wedge q)$.

$$(5) (\neg p) \leftrightarrow (p \wedge q).$$

$$(6) (p \wedge q) \vee r.$$

Solution

(1)

p	$\neg p$	$\neg(\neg p)$
t	f	t
f	t	f

$2^1 = 2$ truth assignments.

(2)

p	q	$\neg p$	$p \wedge q$	$(\neg p) \vee (p \wedge q)$
t	t	f	t	t
t	f	f	f	f
f	t	t	f	t
f	f	t	f	t

$2^2 = 4$ truth assignments.

(3)

p	q	$\neg p$	$p \wedge q$	$(\neg p) \wedge (p \wedge q)$
t	t	f	t	f
t	f	f	f	f
f	t	t	f	f
f	f	t	f	f

(4)

p	q	$\neg p$	$p \wedge q$	$(\neg p) \rightarrow (p \wedge q)$
t	t	f	t	t
t	f	f	f	t
f	t	t	f	f
f	f	t	f	f

(5)

p	q	$\neg p$	$p \wedge q$	$(\neg p) \leftrightarrow (p \wedge q)$
t	t	f	t	f
t	f	f	f	t
f	t	t	f	f
f	f	t	f	f

(6)

p	q	r	$p \wedge q$	$(p \wedge q) \vee r$
t	t	t	t	t
t	t	f	t	t
t	f	t	f	t
t	f	f	f	f
f	t	t	f	t
f	t	f	f	f
f	f	t	f	t
f	f	f	f	f

$2^3 = 8$ truth assignments.

Logical Equivalence

4.16 Definition (see Page 13 of Text book)

Suppose P and Q are compound propositions both made up of the propositions in the set $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$. We say that P and Q are *logically equivalent*, written $P \equiv Q$, if for each truth assignment $TA : \mathcal{P} \rightarrow \{t, f\}$ either P and Q are both true or P and Q are both false.

Example 13

Show that:

$$(1) p \rightarrow q \equiv (\neg p) \vee q.$$

$$(2) p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p).$$

(3) $p \rightarrow q \equiv (\neg q) \rightarrow (\neg p)$ (the form $(\neg q) \rightarrow (\neg p)$ is called the contrapositive of $p \rightarrow q$ and is the basis of “proof by contradiction” ie we suppose $\neg q$ and derive $\neg p$ to obtain p and $\neg p$ – which is a contradiction).

$$(4) \neg(p \vee q) \equiv (\neg p) \wedge (\neg q).$$

$$(5) \neg(p \wedge q) \equiv (\neg p) \vee (\neg q).$$

$$(6) \neg(\neg p) \equiv p.$$

Note that (4) and (5) are known as De Morgan’s Laws.

Solution

(1) Truth table is

p	q	$\neg p$	$p \rightarrow q$	$(\neg p) \vee q$
t	t	f	t	t
t	f	f	f	f
f	t	t	t	t
f	f	t	t	t

$2^2 = 4$ truth assignments.

The last two columns coincide and therefore $p \rightarrow q \equiv \neg p \vee q$.

(2) Truth table is

p	q	$p \rightarrow q$	$q \rightarrow p$	$p \leftrightarrow q$	$(p \rightarrow q) \wedge (q \rightarrow p)$
t	t	t	t	t	t
t	f	f	t	f	f
f	t	t	f	f	f
f	f	t	t	t	t

The last two columns are equal and therefore $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$.

(3) The truth table is

p	q	$\neg p$	$\neg q$	$p \rightarrow q$	$(\neg q) \rightarrow (\neg p)$
t	t	f	f	t	t
t	f	f	t	f	f
f	t	t	f	t	t
f	f	t	t	t	t

Again, the last two columns are equal. Therefore $p \rightarrow q \equiv (\neg q) \rightarrow (\neg p)$.

Do (4), (5) and (6) as exercises (see Example 12 (1)). ■

Note

Because of these equivalences, we can always replace “ $p \rightarrow q$ ” by “ $(\neg p) \vee q$ ”. Therefore $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p) \equiv ((\neg p) \vee q) \wedge ((\neg q) \vee p)$. This shows that “ \rightarrow ” and “ \leftrightarrow ” are unnecessary (although useful) and can always be written in terms of \neg , \vee and \wedge .

4.17 Theorem We have the following logical equivalences.

1. Commutative laws:

$$p \wedge q \equiv q \wedge p; \quad p \vee q \equiv q \vee p$$

2. Associative laws:

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

3. Distributive laws:

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

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

4. Identity laws:

$$p \wedge t \equiv p; \quad p \vee f \equiv p$$

(where t stands for a proposition which is always true, a tautology, and f stands for a proposition which is always false, a contradiction, – see below).

5. Negation laws:

$$p \vee \neg p \equiv t; \quad p \wedge \neg p \equiv f$$

6. Double negation law:

$$\neg(\neg p) \equiv p$$

7. Idempotent laws:

$$p \wedge p \equiv p; \quad p \vee p \equiv p$$

8. De Morgan's laws:

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

9. Universal bound laws:

$$p \vee t \equiv t; \quad p \wedge f \equiv f$$

10. Absorption laws:

$$p \vee (p \wedge q) \equiv p; \quad p \wedge (p \vee q) \equiv p$$

11. Negations of t and f :

$$\neg t \equiv f; \quad \neg f \equiv t.$$

Proof Exercise - use truth tables. ■

The results in Theorem 4.17 can be used to algebraically manipulate propositions as an alternative to using truth tables. (In fact, (6) - (11) in this theorem can be deduced from (1) to (5) – which define a Boolean algebra).

Example 14

Show that:

$$(1) \neg(p \rightarrow q) \equiv p \wedge \neg q.$$

$$(2) (\neg((\neg p) \wedge q)) \wedge (p \vee q) \equiv p.$$

Solution

$$\begin{aligned} (1) \neg(p \rightarrow q) &\equiv \neg((\neg p) \vee q) \text{ by Example 13 (1)} \\ &\equiv (\neg(\neg p)) \wedge \neg q \text{ by De Morgan's laws} \\ &\equiv p \wedge \neg q \text{ by double negation law.} \end{aligned}$$

$$\begin{aligned} (2) &(\neg((\neg p) \wedge q)) \wedge (p \vee q) \\ &\equiv ((\neg(\neg p)) \vee (\neg q)) \wedge (p \vee q) \text{ by De Morgan's law} \\ &\equiv (p \vee \neg q) \wedge (p \vee q) \text{ double negation law} \\ &\equiv p \vee ((\neg q) \wedge q) \text{ distributive law used backwards} \\ &\equiv p \vee (q \wedge (\neg q)) \text{ by commutative law for } \wedge \end{aligned}$$

$\equiv p \vee f$ by negation laws

$\equiv p$ by identity laws. ■

See the exercises handed out for more examples like those in Example 14.

Tautologies

Consider the compound proposition $(p \wedge \neg q) \rightarrow p$. We have

$(p \wedge \neg q) \rightarrow p \equiv \neg(p \wedge \neg q) \vee p$ (Example 13)

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

$\equiv p \vee (\neg p \vee \neg(\neg q))$ (De Morgan's law)

$\equiv p \vee (\neg p \vee q)$ double negation law

$\equiv (p \vee \neg p) \vee q$ (associative law).

Now, *whatever truth values are assigned to p and q* , $p \vee \neg p \equiv t$ by negation laws and so by the universal bound laws $(p \vee \neg p) \vee q \equiv t \vee q \equiv q \vee t \equiv t$. Therefore, $(p \wedge \neg q) \rightarrow p \equiv t$ ie has truth value true for all assignments of truth values to p and q .

4.18 Definition A compound proposition P made up of propositions $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ is called a *tautology* if the truth value of P is true for all truth assignments $TA : \mathcal{P} \rightarrow \{t, f\}$.

Thus, $(p \wedge \neg q) \rightarrow p$ is a tautology. This means that its truth value is always true *no matter what propositions we substitute for the symbols p, q* . We write $\models (p \wedge \neg q) \rightarrow p$ to indicate that $(p \wedge \neg q) \rightarrow p$ is a tautology.

Notation

In general, if a compound proposition P is a tautology, we write $\models P$. The symbol “ \models ” is called the *semantic turnstile*.

4.19 Definition A compound proposition P made up of propositions $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ is called a *contradiction* if the truth value of P is false for all truth assignments $TA : \mathcal{P} \rightarrow \{t, f\}$.

Example 15

Show that $((\neg p) \vee (p \wedge q)) \wedge (p \wedge \neg q)$ is a contradiction.

Solution – using commutative laws in several places

$$\begin{aligned}
 & ((\neg p) \vee (p \wedge q)) \wedge (p \wedge \neg q) \\
 \equiv & ((\neg p \vee p) \wedge (\neg p \vee q)) \wedge (p \wedge \neg q) \text{ distributive laws} \\
 \equiv & (t \wedge (\neg p \vee q)) \wedge (p \wedge \neg q) \text{ negation laws} \\
 \equiv & (\neg p \vee q) \wedge (p \wedge \neg q) \text{ identity laws} \\
 \equiv & ((\neg p \vee q) \wedge p) \wedge \neg q \text{ by associativity} \\
 \equiv & (p \wedge (\neg p \vee q)) \wedge \neg q \text{ by commutative laws} \\
 \equiv & ((p \wedge \neg p) \vee (p \wedge q)) \wedge \neg q \text{ distributive laws}
 \end{aligned}$$

$\equiv (f \vee (p \wedge q)) \wedge \neg q$ negation laws
 $\equiv (p \wedge q) \wedge \neg q$ commutative law and identity law
 $\equiv p \wedge (q \wedge \neg q)$ by associative law
 $\equiv p \wedge f$ negation laws
 $\equiv f$ by universal bound laws. ■

Logical Consequence

4.20 Definition Suppose $Q, P_1, P_2, \dots, P_n, \dots$ are compound propositions. We say that Q is a *logical consequence* of $P_1, P_2, \dots, P_n, \dots$ if: whenever $P_1, P_2, \dots, P_n, \dots$ all have truth value true, then Q also has truth value true. We write $P_1, P_2, \dots, P_n, \dots \models Q$ to indicate this.

Example 16

Show that $((\neg p) \wedge q) \models \neg(p \wedge q)$ but that $((\neg p) \wedge q)$ and $\neg(p \wedge q)$ are not logically equivalent.

Solution

Consider the truth tables:

p	q	$\neg p$	$(\neg p) \wedge q$
t	t	f	f
t	f	f	f
f	t	t	t
f	f	t	f

p	q	$p \wedge q$	$\neg(p \wedge q)$
t	t	t	f
t	f	f	t
f	t	f	t
f	f	f	t

Clearly $((\neg p) \wedge q)$ and $\neg(p \wedge q)$ are not equivalent (two last columns differ). But whenever $((\neg p) \wedge q)$ is true (ie p false, q true) then also $\neg(p \wedge q)$ is true. Thus $((\neg p) \wedge q) \models \neg(p \wedge q)$.

Exercise

Show that the following hold for arbitrary compound propositions P, Q .

$$(1) P \models Q \quad \text{iff} \quad \models P \rightarrow Q.$$

$$(2) P \models Q \text{ and } Q \models P \quad \text{iff} \quad \models P \leftrightarrow Q \quad \text{iff} \quad P \equiv Q.$$

5 LOGIC PART 2 – Predicate Logic

See pp 18-31 of Textbook.

Consider the statement:

$$P : n \text{ is an odd integer.}$$

A proposition is a statement that is either true or false but not both. Statement P is not a proposition because its truth value depends on n i.e. changes with n . Eg. With $n = 1$, the statement made by P is true; with $n = 2$, the statement made by P is false. However, for each n , statement P is a proposition:

5.1 Definition Let $P(x)$ be a statement involving the variable x and let D be a set. We call P a *propositional function* with respect to D if, for each $x \in D$, $P(x)$ is a proposition. We call D the *domain of discourse* of P .

Example 1

The following are propositional functions

(1) $P(n)$ taken to be the statement “ n is an odd integer”, with D taken as N i.e. the set of natural numbers. For any given n , $P(n)$ is a proposition i.e. either true or false but not both. Eg. $P(1)$ is true, $P(2)$ is false etc. Thus, P is a propositional function.

(2) $P(n)$ taken to be the statement “ $n^2 + 2n$ is an even integer”, with D taken as N .

(3) $P(x)$ taken to be the statement “ $x^2 - x - 6 = 0$ ”, with D taken as R ie as the set of all real numbers.

(4) $P(x)$ taken to be the statement “ $x^2 \geq 0$ ”, with D taken as R .

In each of these cases, $P(n)$ or $P(x)$ is a proposition (true or false but not both) when given any value of n or x .

Thus, a propositional function generates a whole family of propositions $P(x)$, one for each $x \in D$, and so each $P(x)$ is either true or false (but not both) though P itself is not a proposition.

5.2 Definition (Quantifiers) Let P be a propositional function with domain of discourse D . The statement $\forall x P(x)$ is read “for all x , $P(x)$ ” or “for every x , $P(x)$ ” or “for each x , $P(x)$ ” and is called a *universally quantified statement*; the symbol \forall is called the *universal quantifier*. The statement $\forall x P(x)$ is true if $P(x)$ is true for every $x \in D$ and is false if $P(x)$ is false for at least one $x \in D$.

The statement $\exists x P(x)$ is read “for some x , $P(x)$ ” or “there exists x , $P(x)$ ” or “for at least one x , $P(x)$ ” and is called an *existentially quantified statement*; the symbol \exists is called

the *existential quantifier*. The statement $\exists x P(x)$ is true if $P(x)$ is true for at least one $x \in D$ and is false if $P(x)$ is false for every $x \in D$.

5.3 Note

(1) In the propositional function P , or $P(x)$, x is called a free or unquantified variable (it can vary freely over its domain of discourse). In the statements $\forall x P(x)$ and $\exists x P(x)$, x is called a bound or quantified variable (bound by \forall or \exists) c.f. analogy with $\int_a^b f(x, y)dx$ – x is bound and y is free. Changing x to t , say, makes no difference ie $\int_a^b f(x, y)dx = \int_a^b f(t, y)dt$. But changing y to some other value z , say, does matter.

(2) A propositional function P does not have a truth value (though each $P(x)$ does) but, given D , $\forall x P(x)$ and $\exists x P(x)$ do have truth values ie they are propositions. More generally, a statement with free variables is not a proposition, a statement with no free variables is a proposition.

(3) To specify D , we sometimes write $\forall x \in D, P(x)$ (read “for all x in D , $P(x)$ ”) instead of $\forall x P(x)$; and write $\exists x \in D, P(x)$ (read “there exists x in D , $P(x)$ ”) instead of writing $\exists x P(x)$.

Example 2

Find the truth value of the statement “ $\forall n \in N$, if n is even, then n^2 is even.”

Solution

Let Q be the propositional function given by $Q(n) : n$ is even, where $D = \text{domain of discourse} = N = \text{set of natural numbers}$. Let P be the propositional function given by $P(n) : Q(n) \rightarrow Q(n^2)$, and again $D = N$. So far each n , $P(n)$ is the proposition $Q(n) \rightarrow Q(n^2)$ ie “if n is even, then n^2 is even.” So the given problem is to find the truth value of $\forall n P(n)$.

Let $n \in N = D$ be arbitrary.

Case 1 n odd

In this case, $Q(n)$ is false. Therefore, $Q(n) \rightarrow Q(n^2)$ is true by the truth table for \rightarrow i.e. $P(n)$ is true.

Case 2 n even

If n is even, then $n = 2k$ for some k . Therefore, $n^2 = (2k)^2 = 4k^2 = 2(2k^2) = 2m$, where $m = 2k^2$. Therefore n^2 is also even. Therefore $Q(n) \rightarrow Q(n^2)$ has truth value true by truth table for \rightarrow (both antecedent and consequent are true) i.e. $P(n)$ is true.

So for every n , $Q(n) \rightarrow Q(n^2)$ is true ie $P(n)$ is true for ev-

ery n . Therefore $\forall n P(n)$ is true and so the given statement is true. ■

Example 3

Find the truth value of the statement $\exists x \in Q, P(x^2, 2)$, where Q denotes the set of all rational numbers, when:

- (i) P denotes equality ie $P(x, y)$ means “ $x = y$ ”;
- (ii) P denotes inequality \geq ie $P(x, y)$ means “ $x \geq y$ ”.

Solution (i) Let $D = Q \times Q$ and let P be the propositional function with domain of discourse $Q \times Q$ given by $P(x, y) : x = y$. Thus, $P(x^2, 2) : x^2 = 2$. The truth value of $\exists x \in Q, P(x^2, 2)$ is therefore the truth value of the statement “there exists a rational number x s.t. $x^2 = 2$ ”. We show that this is false.

Suppose $x = \frac{p}{q}$ is a rational number; thus $p, q \in Z$ ie are integers, $q \neq 0$ and p, q have no common factor. Suppose that $x^2 = 2$ ie $\frac{p^2}{q^2} = 2$. From this we get, by cross multiplying, that $p^2 = 2q^2$ and hence p^2 is even. Therefore p is even (exercise) ie $p = 2k$ for some k . But then $p^2 = 4k^2 = 2(2k^2)$ ie $4k^2 = 2q^2$ so $2k^2 = q^2$ and therefore we see that q^2 is even. Again, it follows that q itself must be even and thus both p and q are even. This means that they have a factor of 2 in common, contrary to the original assumption. Thus,

$\exists x \in Q, P(x^2, 2)$ is false.

(ii) Again $D = Q \times Q$ but this time P is the propositional function given by $P(x, y) : x \geq y$. So $\exists x \in Q, P(x^2, 2)$ is true if for at least one $x \in Q, P(x^2, 2)$ is true ie $x^2 \geq 2$ is true. For example, with $x = 3$ we have $3^2 > 2$ true and $3 \in Q$. Therefore, $\exists x \in Q, P(x^2, 2)$ is true. ■

5.4 Terminology

In Example 3 (i), we say that the symbol P (the propositional function) has been interpreted to be the relation of equality, $=$. In Example 3 (ii), P has been interpreted to be the relation of inequality \geq . In general, to give a truth value to a statement $\exists x P(x)$ or $\forall x P(x)$, we have to interpret the symbol P (the propositional function) in terms of a relation. The statement may be true or it may be false depending on the choice of relation. If the statement is true, we call the interpretation a model of or a model for the statement.

Example 4

Consider the statement

$$(\forall x \forall y \forall z (P(x, x) \wedge (P(x, y) \vee P(y, x)) \wedge (P(x, y) \wedge P(y, z) \rightarrow P(x, z))) \rightarrow \exists x \forall y P(x, y)).$$

- (i) Show that every interpretation over a finite set is a model.
- (ii) Find an interpretation which is not a model.

Solution Note that this example illustrates the use of several variables and mixing quantifiers.

(i) Suppose $A = \{a_1, a_2, \dots, a_n\}$ is any finite set, and suppose we interpret P as the relation R on A . Since P is of arity 2 (two places of variables) the relation R is binary (of arity 2) also.

Case 1

$\forall x \forall y \forall z (P(x, x) \wedge (P(x, y) \vee P(y, x)) \wedge (P(x, y) \wedge P(y, z) \rightarrow P(x, z))$ false. Then the given statement $(\forall x \forall y \forall z (\quad)) \rightarrow \exists x \forall y P(x, y)$ is true by truth table for \rightarrow (false \rightarrow true is true, false \rightarrow false is true).

Case 2

$\forall x \forall y \forall z (P(x, x) \wedge (P(x, y) \vee P(y, x)) \wedge (P(x, y) \wedge P(y, z) \rightarrow P(x, z))$ true. This tells us that R satisfies:

- (a) $R(a_i, a_i)$ for each i ie R is reflexive;
- (b) $R(a_i, a_j)$ or $R(a_j, a_i)$ for all i, j ie each pair a_i, a_j is comparable in R ;
- (c) if $R(a_i, a_j)$ and $R(a_j, a_k)$, then $R(a_i, a_k)$ ie R is transitive.

To verify (i), we must show $\exists x \forall y P(x, y)$ is true ie there is a k s.t $R(a_k, a_j)$ for all j ie A has a “least” element a_k for the relation R .

Consider A listed as $\{a_1, a_2, a_3, \dots, a_n\}$. Choose a_1 and compare it with each element of A - we can do this by (b). Certainly $R(a_1, a_1)$ by (a). If $R(a_1, a_j)$ for all j , then we have finished because we can take $a_k = a_1$. If not, there is a smallest natural number $n_1 > 1$ s.t. $(a_1, a_{n_1}) \notin R$ ie we have:

$R(a_1, a_1), R(a_1, a_2), R(a_1, a_3), \dots, R(a_1, a_{n_1-1})$, but
 $R(a_{n_1}, a_1)$.

By (c) we now have:

$R(a_{n_1}, a_1)$

$R(a_{n_1}, a_1)$ and $R(a_1, a_2)$ and therefore $R(a_{n_1}, a_2)$

$R(a_{n_1}, a_1)$ and $R(a_1, a_3)$ and therefore $R(a_{n_1}, a_3)$

⋮

$R(a_{n_1}, a_1)$ and $R(a_1, a_{n_1-1})$ and therefore $R(a_{n_1}, a_{n_1-1})$

and of course $R(a_{n_1}, a_{n_1})$ by (a). ie we have

$R(a_{n_1}, a_1), R(a_{n_1}, a_2), R(a_{n_1}, a_3), \dots, R(a_{n_1}, a_{n_1})$.

If $R(a_{n_1}, a_j)$ for all j , then we have finished - take $a_k = a_{n_1}$.

If not, then there is a smallest natural number $n_2 > n_1 > 1$

s.t. $(a_{n_1}, a_{n_2}), \notin R$. By the argument above we get

$R(a_{n_2}, a_1), R(a_{n_2}, a_2) \dots R(a_{n_2}, a_{n_1-1}), R(a_{n_2}, a_{n_1}), \dots,$

$R(a_{n_2}, a_{n_2})$. If $R(a_{n_2}, a_j)$ for all j , then we have finished -

we can take $a_k = a_{n_2}$. If not, then there is a smallest natural

number $n_3 > n_2 > n_1 > 1$ s.t. $(a_{n_2}, a_{n_3}) \notin R$ etc. By repeating this argument, we get a sequence $1 < n_1 < n_2 < n_3 < \dots$ of natural numbers. Since each of these $n_j \leq n$, eventually one of them equals n ie $n_m = n$ for some m . Choosing $a_k = a_{n_m}$ gives $R(a_{n_m}, a_j)$ for all j , i.e. $R(a_k, a_j)$ which proves (i).

(ii) Take the set A to be the set of negative integers together with zero: thus $A = \{\dots, -n, \dots, -3, -2, -1, 0\}$, and let R be the relation \leq on A ie $R(a_i, a_j)$ means $a_i \leq a_j$. Then (a) R is reflexive: $R(a_i, a_i)$ holds for all i (because $a_i \leq a_i$ for all i).

(b) Each pair of elements of A is comparable: either $a_i \leq a_j$ or $a_j \leq a_i$ for any i, j so either $R(a_i, a_j)$ or $R(a_j, a_i)$.

(c) R is transitive: given any a_i, a_j and $a_k \in A$, if $a_i \leq a_j$ and $a_j \leq a_k$, then $a_i \leq a_k$. Thus, $R(a_i, a_j)$ and $R(a_j, a_k)$ implies $R(a_i, a_k)$ and so R is transitive.

Therefore, if we interpret P to be the relation R , then the expression

$(\forall x \forall y \forall z (P(x, x) \wedge (P(x, y) \vee P(y, x)) \wedge (P(x, y) \wedge P(y, z) \rightarrow P(x, z)))$ is true.

But in this interpretation the expression $\exists x \forall y P(x, y)$ is not true. For this to be true, we would have to have an $a_k \in A$ s.t. $R(a_k, a_j)$ for all $a_j \in A$ ie an a_k s.t. $a_k \leq a_j$ for all $a_j \in A$ ie a smallest negative integer, and no such thing exists.

Therefore $(\forall x \forall y \forall z (P(x, x) \wedge (P(x, y) \vee P(y, x)) \wedge (P(x, y) \wedge P(y, z) \rightarrow P(x, z))) \rightarrow \exists x \forall y P(x, y))$ is not true in the present interpretation, which is therefore not a model for the given expression. ■

5.5 Terminology

An expression which is true in certain interpretations and false in others is called a *contingency*. Thus, the expression in Example 4 is a contingency. An expression which is true in every interpretation is said to be *universally valid* or *logically valid*. Finally, if E and $E_1, E_2, E_3, \dots, E_n, \dots$ are expressions, then we say that E is a *logical consequence* of the $E_1, E_2, \dots, E_n, \dots$, written $E_1, E_2, \dots, E_n, \dots \models E$, if, for all possible interpretations, whenever $E_1, E_2, \dots, E_n, \dots$ are all true in an interpretation, then E is also true in that interpretation. (This last definition captures the idea that E follows from the $E_1, E_2, \dots, E_n, \dots$ by the “laws of logic” alone and is absolutely fundamental).

Example 5

Show that

- (a) $(\exists x \forall y P(x, y)) \rightarrow (\forall y \exists x P(x, y))$ is logically valid.
- (b) $(\forall y \exists x P(x, y)) \rightarrow (\exists x \forall y P(x, y))$ is a contingency.
- (c) $\exists x \forall y P(x, y) \models \forall y \exists x P(x, y)$.

Solution

(a) Consider an arbitrary interpretation, call it I , for the expression in (a). Thus, we have an arbitrary set A and a binary relation R on A interpreting P .

Case 1

$\exists x \forall y P(x, y)$ is false in I . Then the expression

$(\exists x \forall y P(x, y)) \rightarrow (\forall x \exists y P(x, y))$ is true in I by truth table for \rightarrow .

Case 2

$(\exists x \forall y P(x, y))$ is true in I . Thus, there is an a_0 , say, belonging to A s.t. $R(a_0, a)$ for all $a \in A$. Therefore, given any $a \in A$, there exists an element $a_0 \in A$ s.t $R(a_0, a)$. Therefore $\forall y \exists x P(x, y)$ is true in I . Therefore, $(\exists x \forall y P(x, y)) \rightarrow (\forall y \exists x P(x, y))$ is true in I , by truth table for \rightarrow . Since I was arbitrary and the expression in (a) is true in I , the given expression is logically valid.

(b) First, consider the interpretation consisting of the set

$A = \{0\}$ and the relation R on A where $R = \{(0, 0)\}$. In this interpretation, $\forall y \exists x P(x, y)$ is true since the only possibility is $R(0, 0)$, which is true. Also, $\exists x \forall y P(x, y)$ is true since again $R(0, 0)$ is the only possibility and is true. Therefore the expression $(\forall y \exists x P(x, y)) \rightarrow (\exists x \forall y P(x, y))$ is true in I . Hence I is a model for the expression in (b).

Second, consider the interpretation I in which A is the set $A = \{\dots, -n, \dots, -3, -2, -1, 0\}$ of negative integers together with zero, and let R be the relation \leq on A . In this interpretation, $\forall y \exists x P(x, y)$ is true because, given any value a in A for y , we can choose x to be $a - 1$ (we can even choose x to be a itself) to get $a - 1 \leq a$ ie $R(a - 1, a)$. But, $\exists x \forall y P(x, y)$ is *false* in I . For this expression to be true, we need an $a_0 \in A$ (the value for x) s.t $a_0 \leq a$ for every a (a is the value for y). Thus, $(\forall y \exists x P(x, y)) \rightarrow (\exists x \forall y P(x, y))$ is false in I , and so this expression is a contingency.

(c) Suppose I any interpretation in which $\exists x \forall y P(x, y)$ is true. By the argument in Case 2 of (a), we get $\forall y \exists x P(x, y)$ also true. Therefore, $\exists x \forall y P(x, y) \models \forall y \exists x P(x, y)$. ■

5.6 Theorem (De Morgan's Law – Page 27 text).

Suppose P is a propositional function with domain of discourse D . Then:

(a) $\neg(\forall xP(x))$ and $\exists x\neg P(x)$ always have the same truth value, and therefore are equivalent.

(b) $\neg(\exists xP(x))$ and $\forall x\neg P(x)$ always have the same truth value, and therefore are equivalent.

Proof

The proof (a) is in the Textbook – read it as an exercise.

(b) Suppose $\neg(\exists xP(x))$ is true. Then $\exists xP(x)$ is false. The only way this can happen is that for no $x \in D$ is $P(x)$ true ie for every $x \in D$, $P(x)$ is false. Thus, for every $x \in D$, $\neg P(x)$ is true ie $\forall x\neg P(x)$ is true.

Now suppose $\neg(\exists xP(x))$ is false. Then $\exists xP(x)$ is true ie for some $x \in D$, $P(x)$ is true. Therefore for some $x \in D$, $\neg P(x)$ is false. Therefore, $\forall x\neg P(x)$ is false. (for $\forall x\neg P(x)$ to be true we would need $\neg P(x)$ to be true for every $x \in D$). ■

Note (See Page 28)

(1) A conjunction $P_1 \wedge P_2 \wedge \dots \wedge P_n$ of finitely many propositions P_1, P_2, \dots, P_n is true iff each proposition P_i is true. A universally quantified statement $\forall xP(x)$ can be thought of as an arbitrary family of propositions $\{P(x); x \in D\}$ and is true iff each $P(x)$ is true.

(2) A disjunction $P_1 \vee P_2 \vee \dots \vee P_n$ of finitely many propo-

sitions P_1, P_2, \dots, P_n is true iff at least one of them is true. An existentially quantified statement $\exists xP(x)$ again can be thought of as a family of propositions $\{P(x); x \in D\}$ and is true iff at least one of them is true.

(3) The Theorem 5.6 above generalizes the earlier version of De Morgan's laws in Theorem 4.17.

(4) This theorem shows that each of \forall and \exists can be defined in terms of the other. Therefore, only one of these quantifiers is needed, but it is convenient to retain both of them.

6 ELEMENTARY PROPERTIES OF THE INTEGERS, DIVISIBILITY

See Pages 151-161 of the Textbook for some of this material. Let Z denote the set of all integers ie $Z = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$. We have $N \subset Z$ where $N = \{0, 1, 2, 3, \dots\}$ is the set of all natural numbers. We assume familiarity with the binary operations $+$: $Z \times Z \rightarrow Z$ (addition) $-$: $Z \times Z \rightarrow Z$ (subtraction) and \times : $Z \times Z \rightarrow Z$ (multiplication). We write $n+m$ instead of $+(n, m)$, $n-m$ instead of $-(n, m)$ and $n \times m$ or $n \cdot m$ or just nm instead of $\times(n, m)$, for $n, m \in Z$. We assume familiarity with the identities satisfied by these operations:

6.1 Properties of $+$, \times , $-$

(1) $n + m = m + n$ and $n \times m = m \times n$ for all $m, n \in Z$ (*commutative laws*).

(2) $n + (m + k) = (n + m) + k$ and $n \times (m \times k) = (n \times m) \times k$ for all $n, m, k \in Z$ (*associative laws*).

(3) $n \times (m + k) = (n \times m) + (n \times k)$ and $(m + k) \times n = (m \times n) + (k \times n)$ for all $n, m, k \in Z$ (*distributive laws*).

(4) $0 + n = n + 0 = n$ for all $n \in Z$ (0 is the *identity for addition*). Also $1 \times n = n \times 1 = n$ for all $n \in Z$ (1 is the

identity for multiplication) (Existence of identity elements for $+$ and \times).

(5) For each $n \in \mathbb{Z}$, there is an integer denoted by $-n \in \mathbb{Z}$ s.t. $n + (-n) = (-n) + n = 0$ ($-n$ is called the *additive inverse* of n). There is no corresponding law for \times .

From these follow:

(6) If $n, m, k \in \mathbb{Z}$ and $n + m = n + k$, then $m = k$. If $n \times m = n \times k$ and $n \neq 0$, then $m = k$ (*cancellation laws*).

The first shows that the number 0 in (4) is unique with the properties stated there; the second shows that the number 1 in (4) is unique with the properties stated there (Exercise).

(7) Given $n, m \in \mathbb{Z}$, there exists exactly one $k \in \mathbb{Z}$ s.t. $n + k = m$ (*possibility of subtraction*). We denote this k by $m - n$. No corresponding law holds for multiplication.

(8) $m - n = m + (-n)$ for all $m, n \in \mathbb{Z}$.

(9) $-(-n) = n$ for all $n \in \mathbb{Z}$.

(10) $n \times (m - k) = (n \times m) - (n \times k)$ for all $n, m, k \in \mathbb{Z}$.

(11) $0 \times n = n \times 0 = 0$ for all $n \in \mathbb{Z}$.

(12) For all $m, n \in \mathbb{Z}$, if $m \times n = 0$, then either $m = 0$ or $n = 0$ (no *zero-divisors* - not the same as divisors of zero, see later).

(13) For all $m, n \in \mathbb{Z}$, we have $(-n) \times m = n \times (-m) =$

$-(n \times m)$.

(14) For all $m, n \in \mathbb{Z}$, we have $(-n) \times (-m) = n \times m$.

6.2 Definition A natural number $n \in \mathbb{N}$ is called *prime* if $n > 1$ and, whenever $n = rs$ with r and s in \mathbb{N} , we have either $r = 1$ or $s = 1$. An integer $n \in \mathbb{N}$ is called *composite* if n can be written as $n = r \times s$ with $r, s \in \mathbb{N}$ and $r > 1$ and $s > 1$.

Thus, every integer greater than 1 is either prime or composite because each part of this definition is the negation of the other (Exercise).

6.3 Definition If n and d are integers and $d \neq 0$, then we say that n is *divisible* by d iff $n = d \times k$ for some integer $k \in \mathbb{Z}$. We then say that n is a *multiple* of d or d is a *factor* of n or d is a *divisor* of n or d *divides* n . We write $d|n$ to indicate that d divides n . So $d|n$ iff there exists an integer k s.t. $n = dk$.

Recall that we define the modulus, or absolute value, $|x|$ of any real number x by

$$|x| = \begin{cases} x & \text{if } x \geq 0, \\ -x & \text{if } x < 0. \end{cases}$$

Note that $|x| \geq 0$ for all $x \in R$. In particular, this is defined for any $n \in Z$ ie

$$|n| = \begin{cases} n & \text{if } n \geq 0 \\ -n & \text{if } n < 0. \end{cases}$$

Recall also that

- (i) $|xy| = |x||y|$ for any real numbers x, y (and hence $|nm| = |n||m|$ for $n, m \in Z$), and
- (ii) $|x + y| \leq |x| + |y|$ for any real numbers x, y (and hence $|n + m| \leq |n| + |m|$ for $n, m \in Z$).

6.4 Proposition (1) Any non-zero integer k is a divisor of 0.

(2) If $a, b \in Z$ with a and b both positive and $a|b$, then $a \leq b$.

(3) The only divisors of 1 are 1 and -1 .

(4) “ $|$ ” is transitive ie if $a|b$ and $b|c$, then $a|c$ for any non-zero $a, b, c \in Z$.

Proof

(1) Let k be any non-zero integer. Then since $0 = k \cdot 0$, k is a divisor of the LHS ie of 0.

(2) Suppose $a, b \in Z$ with $a, b > 0$ and $a|b$. Thus, $a \geq 1$, $b \geq 1$ and $b = a \times k$ for some $k \in Z$. Since a and b are

both ≥ 1 , we must have $k \geq 1$. Therefore by elementary properties of inequalities we have $a \times k \geq a \times 1 = a$. Since $b = a \times k$ we have $b \geq a$ ie $a \leq b$ as required.

(3) By (2), any positive integer which divides 1 is less than or equal to 1. Also $1 = 1 \times 1$ and so 1 divides 1 ie 1 is a divisor of 1. Since there are no positive integers less than 1, 1 is the only positive divisor of 1. Now suppose d is a negative integer which divides 1 ie $1 = d \times k$ for some k . Then $1 = |1| = |d \times k| = |d||k|$. Hence $|d|$ is a positive integer which divides 1 and therefore $|d| = 1$, by first part of (3). Hence $d = -1$. Similarly, $k = -1$ so the only divisors of 1 are 1 and -1 .

(4) See Example 16 in §3 for this result with $a, b, c \in N$. Since $a|b$ we have $b = a \times k_1$ for some $k_1 \in Z$. Since $b|c$ we have $c = b \times k_2$ for some $k_2 \in Z$. Therefore $c = b \times k_2 = (a \times k_1) \times k_2 = a \times (k_1 \times k_2) = a \times k$, where $k = k_1 \times k_2 \in Z$. Thus, $a|c$ as required. ■

6.5 Theorem Any integer $n > 1$ is divisible by a prime number.

Proof Let $n \in N$ with $n > 1$. If n is prime, then n is divisible by a prime number (itself) since $n = n \times 1$, and we are done. If n is not prime, then n is composite ie $n = r_0 \times s_0$. By

definition of “composite”, $r_0 \neq 1$ and $s_0 \neq 1$ ie $r_0 > 1$ and $s_0 > 1$. Since r_0 and s_0 are divisors of n , we have $r_0 \leq n$ and $s_0 \leq n$ by 6.4 (2). But $r_0 > 1$ and $s_0 > 1$ and so we must have $r_0 < n$ and $s_0 < n$. To see this, suppose $r_0 = n$. Since $s_0 > 1$ we get $r_0 \times s_0 > r_0 \times 1 = r_0$ ie $n > r_0 = n$ – a contradiction; thus $r_0 < n$.

Similarly if we suppose $s_0 = n$ we get a contradiction. Thus, $n = r_0 \times s_0$ with $1 < r_0 < n$ and $1 < s_0 < n$.

Now r_0 and s_0 are both divisors of n . Consider r_0 . If r_0 is prime, we are done. If r_0 is not prime, it is composite and so $r_0 = r_1 \times s_1$ for $r_1, s_1 \in N$ with $r_1 \neq 1$ and $s_1 \neq 1$. By the argument above $1 < r_1 < r_0$ and $1 < s_1 < r_0$. By definition of divisibility we have $r_1 | r_0$. But $r_0 | n$ and therefore by transitivity of “|”, $r_1 | n$. So if r_1 is prime, we are done. If not, repeat this argument and keep on repeating it.

Claim This process must stop in finitely many steps by producing a prime number r_k s.t. $r_k | n$.

Proof If not, the process continues indefinitely and we produce a sequence $1 < \dots < r_k < r_{k-1} < r_{k-2} < \dots < r_1 < r_0 < n$. This is impossible since there are only finitely many integers (in fact, $n - 2$) between 1 and n . The condition for this process to stop is that r_k is prime. ■

The following result follows from the previous theorem.

6.6 Theorem (Fundamental Theorem of Arithmetic).

Let $n > 1$ be a natural number. Then there are prime numbers p_1, p_2, \dots, p_k and natural numbers e_1, e_2, \dots, e_k s.t. $n = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_k^{e_k}$. This representation is unique up to order.

Example 1

Take $n = 20$. The decomposition given by Theorem 6.6 is $20 = 2^2 \times 5^1$. Thus, $p_1 = 2$ and $e_1 = 2$, $p_2 = 5$ and $e_2 = 1$. Equally well we could write $20 = 5^1 \times 2^2$. Then $p_1 = 5$ and $e_1 = 1$, and $p_2 = 2$ and $e_2 = 2$. Both representations use the same primes and exponents, but the order is different.

6.7 Note

(1) If we order the p_1, p_2, \dots, p_k so that $p_1 < p_2 < p_3 < \dots < p_k$ we obtain the standard representation in Theorem 6.6 - which is unique.

(2) Theorem 6.6 can be used to find all the divisors of a natural number n : if $n = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_k^{e_k}$ is the decomposition of n given by Theorem 6.6, then any divisor of n takes the form $q_1^{f_1} \cdot q_2^{f_2} \cdot \dots \cdot q_l^{f_l}$ where the q_i are distinct and each is one

of the p_1, p_2, \dots, p_k (q_i is p_{j_i} , say), $l \leq k$ and $f_i \leq e_{j_i}$ for $i = 1, \dots, l$.

For example, take $n = 60 = 2^2 \times 3^1 \times 5^1$. The only divisors of n are

1, 2, 3, 5, 2×2 , 2×3 , 2×5 , 3×5 , $2 \times 3 \times 5$, $2^2 \times 3$, $2^2 \times 5$, $2^2 \times 3 \times 5$.

Some Simple Algorithms Related to Operations on Numbers

The following algorithms are written in (Pascal-like) pseudo-code (as in the Textbook) but can easily be implemented in any programming language; comments are placed in square brackets “[”, “]”.

6.8 The Division Algorithm

[Given a non-negative integer a and a positive integer d , the algorithm finds integers q and r s.t. $a = d \cdot q + r$, where $0 \leq r < d$ (q is called the quotient and r the remainder).

This is done by subtracting d repeatedly from a until the result is less than d but is still non-negative:

$$0 \leq a - d - d - d \dots - d = a - d \cdot q < d.$$

The total number of d s subtracted is q ; the quantity $a - dq$

is r .]

Input: a [a non-ve integer], d [a positive integer]

Algorithm body:

$r := a; q := 0;$

[Repeatedly subtract d from r until a number less than d is obtained. Add 1 to q each time d is subtracted]

while ($r \geq d$) do

$r := r - d;$

$q := q + 1$

end while;

[After execution of the while loop, $a = d \cdot q + r$]

Output: q, r [non-negative integers]

end Division Algorithm

Example 2

Trace the Division Algorithm when

(i) $a = 19, d = 4$

(ii) $a = 4, d = 19$

(iii) $a = 4, d = 4$

(iv) $a = 4, d = 2$.

Solution. “Trace” means display the values of the variables at each point in the execution of the algorithm.

(i) Iteration Number

	0	1	2	3	4
a	19				
Variable names d	4				
r	19	15	11	7	3
q	0	1	2	3	4

Do the remaining cases as exercises.

The Euclidean Algorithm

6.9 Definition Let a and b be integers that are not both zero. The *greatest common divisor* of a and b , denoted $\gcd(a, b)$, is that integer d with the following properties:

- (i) d is a common divisor of a and b ie $d|a$ and $d|b$.
- (ii) For all integers c , if $c|a$ and $c|b$ then $c \leq d$.

Note By Proposition 6.4 (1) any non-zero integer is a divisor of 0. Thus, if we take $a = b = 0$ in the above definition, then *any* integer is a common divisor of a and b . Therefore, there can be no greatest common divisor when $a = b = 0$.

Example 3 Find

- (i) $\gcd(72, 63)$.
- (ii) $\gcd(10^{20}, 6^{30})$.

Solution

(i) $72 = 8 \times 9 = 2^3 \times 3^2$ is the prime decomposition of 72 with $p_1 = 2, e_1 = 3; p_2 = 3, e_2 = 2$.

$63 = 7 \times 9 = 7^1 \times 3^2$ is the prime decomposition of 63 with $p_1 = 7, e_1 = 1; p_2 = 3, e_2 = 2$.

Therefore the divisors of 72 are

1; 2, 3; $2^2, 2 \times 3, \underline{3^2}$; $2^3, 2^2 \times 3, 2 \times 3^2$; $2^3 \times 3, 2^2 \times 3^2$; $2^3 \times 3^2$.

And the divisors of 63 are

1; 7, 3; $7 \times 3, \underline{3^2}$; 7×3^2 .

Common divisors (underlined) are 1, 3 and 3^2 . Clearly $3^2 = 9$ is greatest common divisor ie $\gcd(72, 63) = 9$.

(ii) $10^{20} = (2 \times 5)^{20} = 2^{20} \times 5^{20}$ is the prime decomposition of 10^{20} with $p_1 = 2, e_1 = 20; p_2 = 5, e_2 = 20$.

$6^{30} = (2 \times 3)^{30} = 2^{30} \times 3^{30}$ is the prime decomposition of 6^{30} with $p_1 = 2, e_1 = 30; p_2 = 3, e_2 = 30$.

Using these we can write down all divisors of 10^{20} and all divisors of 6^{30} , then we can look for common divisors, and then choose the largest of the common divisors (this is tedious).

Instead:

We note that $6^{30} = 2^{30} \times 3^{30} = 2^{20} \times 2^{10} \times 3^{30}$. Thus we

have:

$$10^{20} = \underline{2^{20}} \times 5^{20}, \text{ and}$$

$$6^{30} = 2^{30} \times 3^{30} = \underline{2^{20}} \times 2^{10} \times 3^{30}.$$

Therefore, 2^{20} is a common divisor of 10^{20} and 6^{30} .

Question Can we combine powers of 2 and 3 to get a divisor of 5^{20} , or is 5 a divisor of a combination of powers of 2 and powers of 3? Either way would get new common divisors of 10^{20} and of 6^{30} and therefore a higher common divisor of 10^{20} and 6^{30} than 2^{20} .

(1) Clearly no power of 2 higher than 20 is a common divisor. Eg. although $2^{21} | 2^{20} \times 2^{10} \times 3^{30}$ ie $2^{21} | 6^{30}$ we see that $2^{21} \nmid 2^{20} \times 5^{20}$ since $2 \nmid 5^{20}$ ($2 \nmid 5^n$ for any n – Exercise).

(2) Similarly, $3 \nmid 5^k$ for any k and in particular $3 \nmid 5^{20}$ (Exercise). Therefore, $2 \times 3 \nmid 5^{20}$ and indeed $2^n \times 3^m \nmid 5^k$ for any choice of $n \geq 1, m \geq 1$ and any k .

(3) On the other hand, $5 \nmid 2^n \times 3^m$ for any choice of n and m (Exercise) and in fact $5^k \nmid 2^n \times 3^m$ for any $k \geq 1$ and any choice of n and m . Therefore, $2^{20} \times 5^k \nmid 2^{20} \times 2^n \times 3^m$ for any choice of $k \geq 1$ and any choice of n and m .

From this we see that 2^{20} is the greatest common divisor ie $\gcd(10^{20}, 6^{30}) = 2^{20}$.

The idea used in these examples works in general to find $\gcd(a, b)$, but is not feasible for large a, b since it is not feasible to factor large numbers (the basis of secret coding of data). Instead, we use the Euclidean algorithm, which depends on the division algorithm, and is very fast and efficient.

6.10 Lemma If r is a positive integer, then $\gcd(r, 0) = r$.

Proof. Since $r|r$ and $r|0$, r is a common divisor of r and 0 . Suppose c is any common divisor of r and 0 , where $1 \leq c$. Then $c|r$ in particular. Therefore by 6.4(2), $c \leq r$. Therefore, $\gcd(r, 0) = r$. ■

6.11 Lemma Suppose a, b are integers with $b \neq 0$, and that q and r are non-negative integers s.t. $a = b.q + r$ (c.f. the division algorithm). Then $\gcd(a, b) = \gcd(b, r)$.

Proof Step 1. $\gcd(a, b) \leq \gcd(b, r)$.

We first show: any common divisor of a and b is a common divisor of b and r . Let $c|a$ and $c|b$. Then by definition, $a = n.c$ and $b = m.c$ for some integers n and m . Substitute into the equation $a = b.q + r$ to get:

$$n.c = m.c.q + r$$

so that $r = n.c - m.c.q = (n - m.q).c$, where $n - m.q$ is an integer. Therefore $c|r$, and hence we have $c|r$ and $c|b$ (by

supposition) so that c is a common divisor of b and r . It follows that in particular $\gcd(a, b)$ is a common divisor of b and r . Therefore $\gcd(a, b) \leq \gcd(b, r)$.

Step 2. $\gcd(b, r) \leq \gcd(a, b)$.

Proof. Similar to Step 1 - Exercise.

Hence, $\gcd(a, b) = \gcd(b, r)$. ■

Notes

(1) In the representation $a = b.q + r$ given by the division algorithm, the value q is sometimes called *DIV* (i.e. $q = aDIVb$) and r is sometimes called *MOD* (i.e. $r = aMODb$) for $b > 0$.

(2) Lemmas 6.10 and 6.11 gives us the following recursive definition for \gcd . For $0 \leq b < a$:

$$\gcd(a, b) = \begin{cases} a & \text{if } b = 0, \\ \gcd(b, r) & \text{if } b \neq 0, \end{cases}$$

where the “ r ” is as in the representation $a = b.q + r$. Or, using the notation *MOD* for r ,

$$\gcd(a, b) = \begin{cases} a & \text{if } b = 0, \\ \gcd(b, aMODb) & \text{if } b \neq 0. \end{cases}$$

Note that this formulation differs slightly from that in the definition given on Page 17 of Chapter 3 on relations because b is allowed to be zero here. Also note that this outputs b if

$a \text{ MOD } b = 0$ which is correct since $a \text{ MOD } b = 0$ iff $b|a$ and then $\text{gcd}(a, b) = b$. This gives us an algorithm to compute gcd , called the Euclidean algorithm:

Euclidean Algorithm

Input: a, b integers with $a > b \geq 0$.

Algorithm body:

If $b = 0$ then $\text{gcd} := a$;

[If $b > 0$, compute $r = a \text{ MOD } b$ (use a function or procedure call for the division algorithm). Then $\text{gcd}(a, b) := \text{gcd}(b, r)$.]

While $b \neq 0$ do

$r := a \text{ MOD } b$;

$a := b$;

$b := r$

end while;

[After execution of the while loop, $\text{gcd} :=$ final value of a .]

$\text{gcd} := a$;

Output: gcd [a positive integer]

end Euclidean Algorithm.

Exercise Write programs to implement

(i) the division algorithm. (ii) the Euclidean algorithm.

If $0 \leq b < a$, then by the division algorithm $r = a \text{MOD} b < b$. Hence, $0 \leq r < b < a$. Therefore the largest number in the pair $(b, a \text{Mod} b) =$ largest number in the pair $(b, r) = b <$ largest number in the pair (a, b) ; i.e. the pair (b, r) is “smaller” than the pair (a, b) . Therefore $r := 0$ eventually and the Euclidean algorithm *must* terminate. In fact, it is known that it terminates in not more steps than $5 \times$ number of digits in the smaller number b .

Example 4.

Use the Euclidean algorithm to find $\text{gcd}(8880, 3513)$.

Solution.

$$8880 = 3513 \times 2 + 1854$$

$$\therefore \text{gcd}(8880, 3513) = \text{gcd}(3513, 1854)$$

$$3513 = 1854 \times 1 + 1659$$

$$\therefore \text{gcd}(3513, 1854) = \text{gcd}(1854, 1659)$$

$$1854 = 1659 \times 1 + 195$$

$$\therefore \text{gcd}(1854, 1659) = \text{gcd}(1659, 195)$$

$$1659 = 195 \times 8 + 99$$

$$\therefore \text{gcd}(1659, 195) = \text{gcd}(195, 99)$$

$$195 = 99 \times 1 + 96$$

$$\therefore \text{gcd}(195, 99) = \text{gcd}(99, 96)$$

$$99 = 96 \times 1 + 3$$

$$\therefore \gcd(99, 96) = \gcd(96, 3)$$

$$96 = 3 \times 32 + 0$$

$$\therefore \gcd(96, 3) = \gcd(3, 0) = 3.$$

$$\text{Thus, } \gcd(8880, 3513) = 3.$$

Example 5.

Use the Euclidean algorithm to find $\gcd(631, 101)$.

Solution

$$631 = 101 \times 6 + 25$$

$$\therefore \gcd(631, 101) = \gcd(101, 25)$$

$$101 = 25 \times 4 + 1$$

$$\therefore \gcd(101, 25) = \gcd(25, 1)$$

$$25 = 1 \times 25 + 0$$

$$\therefore \gcd(25, 1) = \gcd(1, 0) = 1. \text{ Thus, } \gcd(631, 101) = 1.$$

We say two numbers a, b , both positive, are *relatively prime* or *coprime* if $\gcd(a, b) = 1$. Thus, 631 and 101 are relatively prime.

Example 6.

Express $\frac{355253}{379231}$ in lowest terms.

Solution. To express a ratio $\frac{a}{b}$ in lowest terms, where a and $b > 0$, is to write $\frac{a}{b} = \frac{d.e}{d.f}$ with d as large as possible, i.e. to write $\frac{a}{b} = \frac{d.e}{d.f}$ where $d = \gcd(a, b)$.
 So we want to find $\gcd(a, b) = \gcd(379231, 355253)$.

$$379231 = 355253 \times 1 + 23978$$

$$355253 = 23978 \times 14 + 19561$$

$$23978 = 19561 \times 1 + 4417$$

$$19561 = 4417 \times 4 + 1893$$

$$4417 = 1893 \times 2 + 631$$

$$1893 = 631 \times 3 + 0$$

$$\therefore \gcd(379231, 355253) = \gcd(631, 0) = 631.$$

Now

$$355253 = 631 \times 563$$

$$379231 = 631 \times 601$$

Therefore

$$\frac{355235}{379231} = \frac{631 \times 563}{631 \times 601} = \frac{563}{601}$$

in lowest terms.

Some Issues in the Theory of Algorithms

The following are the three main issues in the theory of algorithms, and therefore are central to theoretical computer science.

(1) Complexity and Efficiency

The main question in computational complexity is: “How complex and how efficient is a given algorithm. In particular, does it, in the large, behave like a polynomial function of time or of its input?” If so, it is regarded as a feasible or tractable algorithm, otherwise, i.e. if it behaves in the large like an exponential function of time or of its input, it is not regarded as feasible i.e. is intractable. This involves comparing the rate of growth of the number of steps made in the algorithm with standard functions such as powers n^k of n (polynomial growth), or $\log(n)$, or $n \log(n)$ etc. (feasible algorithms). On the other hand, if the rate of growth of the number of steps made in the algorithm compares with e^n , then the algorithm is not feasible.

One of the main outstanding problems in this part of the theory is the $P = NP$ problem. This is the problem of whether or not every function computable in polynomial time on a non-deterministic machine (in which there is a certain

amount of freedom, or guidance, in choosing the next step in a computation – non-deterministic Turing machines) is computable in polynomial time on ordinary (deterministic) machines.

(2) **Termination of an Algorithm**

The issue here is proving termination of an algorithm. If the algorithm is essentially a for...loop, then termination is automatic since a for...loop is executed a predetermined number of times and then stops. If the algorithm is essentially a while...loop, then the algorithm will keep going until the condition in the while...loop is violated, and this may never be the case. Thus, there is the need to determine whether or not the algorithm loops for ever. Often, the condition given in the while...loop can be made to depend on a positive integer which can be shown to diminish with each execution of the loop. Since a positive integer cannot be decremented indefinitely, this will force violation of the loop condition and prove termination. This method of proof is an important one.

Example 7

Consider again the division algorithm. We input a (a non-negative integer), and d (a positive integer). This algorithm

uses a while...loop. Initially, r is set equal to a . Then $r := a - d$ (assuming $a \geq d$), then $a - d - d$, then $a - d - d - d$, etc. In general, we subtract q d s and $r := a - d \cdot q$, and thus keeps diminishing while it remains $\geq d$. Clearly, $a - d \cdot q$ will eventually become $< d$. At this point, the algorithm terminates and outputs the values at the previous step.

Example 8

The Euclidean algorithm was set up using a while...loop (while $b \neq 0$ do...) and used the recursive call $\text{gcd}(a, b) := \text{gcd}(b, a \text{ MOD } b)$. It was noted that $(b, a \text{ MOD } b) < (a, b)$ in the sense that the largest number in the first pair $<$ largest number in second pair. Thus, a positive number keeps on reducing with each execution of the loop, and this will force termination.

Example 9 (Summer Exam, 1998)

Let a, b and c be integers with $c \neq 0$. Say what is meant by the statement “ c divides a ”, written $c|a$.

Suppose $c|a$ and $c|b$. Show that

- (i) $c|(a + b)$,
- (ii) $c|(a - b)$.

Deduce from (ii) that, if $a > b \geq 0$, we have

$$\text{gcd}(a, b) = \text{gcd}(a - b, b),$$

where $\gcd(a, b)$ denotes the greatest common divisor of a and b . Show how this fact may be used as the basis of an algorithm to compute $\gcd(a, b)$. Write the algorithm in pseudocode and explain why it must terminate. Illustrate its use by finding $\gcd(72, 60)$.

Solution For the first part: the statement $c|d$ means that there is an integer n such that $d = n.c$.

For (i) and (ii). If $c|a$ and $c|b$, then $a = n_1.c$ and $b = n_2.c$ for integers n_1 and n_2 . Therefore, $a + b = (n_1 + n_2).c$ and $a - b = (n_1 - n_2).c$. From this it follows that $c|(a + b)$ and $c|(a - b)$.

For the next part: Since any divisor c of a and b is a divisor of b and $a - b$ by the first part, c is a common divisor of b and $a - b$. In particular, this is true with $c = \gcd(a, b)$. Therefore, we get the inequality

$$\gcd(a, b) \leq \gcd(a - b, b).$$

Conversely, suppose $c|(a - b)$ and $c|b$. Then $a - b = n_1.c$ and $b = n_2.c$ for integers n_1 and n_2 . Thus, $a = b + n_1.c = n_2.c + n_1.c = (n_1 + n_2).c$ and hence $c|a$. Therefore, any common divisor c of $a - b$ and b is a common divisor of a and b . Again, this is true with the particular choice of

$c = \gcd(a - b, b)$ and we get the inequality

$$\gcd(a - b, b) \leq \gcd(a, b).$$

These two inequalities now give $\gcd(a, b) = \gcd(a - b, b)$ as required.

The Algorithm

[Given two positive integers A and B , variables a and b are set equal to A and B . Then the following procedure begins. If $a \neq 0$ and $b \neq 0$, then the larger of a and b is set equal to $a - b$ if $a \geq b$ or to $b - a$ if $a < b$, and the smaller of a and b is left unchanged. The procedure is repeated until eventually a or b becomes 0. By the fact that $\gcd(a, b) = \gcd(a - b, b)$, we get $\gcd(A, B) = \gcd(a, b)$ after each repetition of the procedure. After the last repetition, $\gcd(A, B) = \gcd(a, 0)$ or $\gcd(A, B) = \gcd(0, b)$ depending on whether a or b is 0. By Lemma 6.10, $\gcd(a, 0) = a$ and $\gcd(0, b) = b$. Therefore, after last repetition, $\gcd(A, B) = a$ if $a \neq 0$ or $\gcd(A, B) = b$ if $b \neq 0$.]

Input: A, B [positive integers]

Algorithm body:

$a := A, b := B;$

while ($a \neq 0$ and $b \neq 0$)

if $a \geq b$ then $a := a - b$

else $b := b - a$

end while;

if $a = 0$ then $gcd := b$

else $gcd := a$;

[After execution of the if-then-else statement, $gcd = gcd(A, B)$.]

output: gcd [a positive integer]

end the algorithm

This algorithm terminates because the largest number in the pair $(a - b, b)$ resp. in the pair $(a, b - a)$ is $<$ the largest number in the pair (a, b) , so again a positive number keeps on diminishing.

To trace this algorithm for $gcd(72, 60)$. We have:

$$\begin{aligned} gcd(72, 60) &= gcd(72 - 60, 60) = gcd(12, 60) = \\ gcd(12, 60 - 12) &= gcd(12, 48) = gcd(12, 48 - 12) = \\ gcd(12, 36) &= gcd(12, 36 - 12) = gcd(12, 24) = \\ gcd(12, 24 - 12) &= gcd(12, 12) = gcd(12 - 12, 12) = \\ gcd(0, 12) &= 12. \blacksquare \end{aligned}$$

(3) **Correctness of Algorithms**

The problem here is “Can we prove that an algorithm (or program, in general) correctly computes a function or correctly solves a problem?” It is the central question in so-called Formal Methods. Thus, the main issues concern proof of the correct specification of the system (validation), and proof of correct implementation (verification of code) i.e. “Have we written the right program, and have we written it right?”

6.12 Pre-conditions & Post-conditions, Loop Invariants (Dijkstra (Amsterdam), Tony Hoare (Oxford Programming Research Group))

Consider an algorithm; it produces a certain final state from a certain initial state. Both initial and final states can be expressed as *predicates* i.e. propositional functions of the input and output variables – called the pre-condition and the post-condition resp.

Example 10

(a) Algorithm to compute product of two non-negative integers

pre-condition: input variables m, n are non-negative integers.

post-condition: output variable p equals $m.n$.

(b) Algorithm to find the quotient and remainder on division of one positive integer by another

pre-condition: input variables a, b are positive integers

post-condition: output variables q and r are integers s.t. $a = b.q + r$ and $0 \leq r < b$.

(c) Algorithm to sort a one-dimensional array of real numbers

pre-condition: the input variable $A[1], A[2], \dots, A[n]$ is a one-dim. array of real numbers

post-condition: the output variable $B[1], B[2], \dots, B[n]$ is a one-dim. array of real numbers with same elements as $A[1], A[2], \dots, A[n]$ with property $B[i] \leq B[j]$ whenever $i \leq j$.

Note that in each case, the pre-condition and post-condition is either true or false depending on the values of the variables i.e. they are propositional functions or predicates.

The following is a key definition:

6.13 Definition A *proof* of algorithm correctness consists of showing that if the pre-condition of the algorithm is true

for a collection of values for the input variables and if the statements of the algorithm are executed, then the post-condition is also true.

Divide and Conquer Principle (applied to algorithms): steps of the algorithm are divided into sections with assertions about current state of algorithm variables inserted at suitable points:

[Assertion 1: pre-condition of algorithm]

{Algorithm statements}

[Assertion 2]

{Algorithm statements}

⋮

[Assertion $k - 1$]

{Algorithm statements}

[Assertion k : post-condition of algorithm]

Successive pairs of assertions are treated as pre- and post-conditions for the algorithm statements between them. For each $i = 1, 2, \dots, k - 1$ we prove that, if Assertion i is true and all the algorithm statements between Assertion i and Assertion $(i + 1)$ are executed, then Assertion $(i + 1)$ is true. Once all these individual proofs have been completed, one

knows that Assertion k is true. Since Assertion 1 and Assertion k are the pre- and post- conditions for the algorithm resp., we conclude that the entire algorithm is correct wrt its pre- and post- conditions.

Correctness of While Loops – Loop Invariants

This proves correctness of a loop wrt pre- and post- conditions, and is based on induction.

Suppose an algorithm contains a while loop, and that entry to this loop is restricted by a condition G called the guard. Suppose also that assertions describing the current states of algorithm variables have been placed immediately preceding and immediately following the loop – called the pre-conditions and the post-conditions of the loop. Thus the loop looks like:

[precondition for loop]

while (G)

[Statements in body of loop. None contain branching statements that lead outside the loop.]

end while

[postcondition for loop]

The following is another key definition.

6.14 Definition A loop is defined to be *correct with respect to its pre- and post- conditions* iff whenever the algorithm variables satisfy the pre-condition for the loop and the loop is executed, then the algorithm variables satisfy the post-conditions for the loop.

6.15 Theorem (Loop Invariant Theorem) Let a while loop with guard G be given, together with pre- and post- conditions that are predicates in the algorithm variables. Also let a predicate $I(n)$, called the *loop invariant*, be given. If the following four properties are true, then the loop is correct wrt its pre- and post- conditions.

- (I) Basis property: the pre-condition for the loop implies that $I(0)$ is true before the first iteration of the loop.
- (II) Inductive property: if the guard G and the loop invariant $I(k)$ are both true for an integer $k \geq 0$ before an iteration of the loop, then $I(k + 1)$ is true after iteration of the loop.
- (III) Eventual falsity of the guard: after a finite number of iterations of the loop, the guard G becomes false.
- (IV) Correctness of the post-condition: if N is the smallest number of iterations of the loop after which G is false and $I(N)$ is true, then the values of the algorithm variables will

be as specified in the post-condition of the loop.

Proof Assume that $I(n)$ is a predicate which satisfies (I) to (IV) above; we prove that the loop is correct wrt its pre- and post- conditions.

Consider the statement:

“For all integers $n \geq 0$, if the while loop iterates n times, then $I(n)$ is true”. . . . (A)

We will show that this statement is true by induction. For the basis step in the induction argument, suppose that the loop satisfies the pre-condition for the loop. Then, by Property (I) in the hypothesis of the theorem i.e. the Basis property, we have that $I(0)$ is true before the first iteration of the loop i.e. if the loop iterates 0 times, then $I(0)$ is true. Thus, Statement (A) holds with $n = 0$. Next, for the inductive step in the induction argument, suppose that Statement (A) holds for $n = k$ i.e. that the loop iterates k times and $I(k)$ is true. Then the guard G must be true for k because the loop will not have been entered for the k th time if the guard is false for k . Therefore, we have the guard G and $I(k)$ true before the k th iteration of the loop. Therefore, by Property (II) in the hypothesis of the theorem i.e. the Inductive property, we have that $I(k + 1)$ is true after the k th iteration of the loop.

Thus, if the loop iterates $k + 1$ times (so that it iterates k times), then $I(k + 1)$ is true. This completes the inductive step. Therefore, (A) is true by the principle of induction.

By Property (III) in the hypothesis i.e. the eventual falsity of the guard, the guard G will eventually become false. At this point the loop will have been iterated some number N of times but does not iterate again because G now becomes false (i.e. G is false for $N + 1$), and N is smallest with this property. By Statement (A), $I(N)$ is true because the loop was iterated for the N th time. So now we are in the situation that after the N th iteration, G is false and $I(N)$ is true and N is smallest with these properties. Therefore, by Property (IV) i.e. the correctness of the post-condition we have that the values of the algorithm variables will be as specified in the post-condition for the loop. Thus, the loop is correct wrt its pre- and post- conditions. ■

Note In using this theorem, the loop invariant $I(n)$ has to be chosen and it may not be obvious where it comes from.

Some Examples of Correctness Proofs

(1) Correctness of a loop to compute a product

The following loop is designed to compute the product $m.x$ for non-negative integer m and real number x , without using a built-in multiplication operation. Prior to the loop, variables i and $product$ have been introduced and given initial values $i = 0$ and $product = 0$.

[precondition: m is a non-negative integer, x is a real number, $i = 0$, and $product = 0$]

while ($i \neq m$)

1. $product := product + x$

2. $i := i + 1$

end while

[postcondition: $product = m.x$]

We take the loop invariant to be $I(n) : i = n$ and $product = n.x$

The guard condition G of the while loop is $G : i \neq m$

We use the loop invariant theorem to prove that the while loop is correct wrt the given pre- and post- conditions.

Solution We have to verify the conditions (I) to (IV) in the hypothesis of Theorem 6.15 relative to the guard G and the choice made of the loop invariant $I(n)$.

(I) Basis property [$I(0)$ is true before the first iteration of the loop].

Now $I(0)$ is “ $i = 0$ and $product = 0.x$ ”, which is true because we put $i = 0$ initially and $0.x = 0 =$ initial value of $product$.

(II) Inductive property [If G and $I(k)$ are true before a loop iteration, where $k \geq 0$, then $I(k + 1)$ is true after the loop iteration].

Suppose that $k \geq 0$ is a non-negative integer s.t. G and $I(k)$ are both true for k before an iteration of the loop. Then, as execution reaches the top of the loop, we have $i \neq m$ (because G is true) and also we have $i = k$ and $product = k.x$ (because $I(k)$ is true). Since $i \neq m$, the guard is passed and Statement 1. in the loop is executed. Before execution of Statement 1., we have $product_{old} = k.x$. Thus, execution of Statement 1. has the following effect:

$$product_{new} = product_{old} + x = k.x + x = (k + 1).x$$

Similarly, before Statement 2. is executed, we have $i_{old} = k$.

So after execution of Statement 2., we have

$$i_{new} = i_{old} + 1 = k + 1.$$

Therefore, after the loop iteration we have the statement: “ $i = k + 1$ and $product = (k + 1).x$ ” being true. But this statement is precisely $I(k + 1)$, and so $I(k + 1)$ is true which is what we needed to show.

(III) Eventual falsity of the guard [After a finite number of iterations of the loop, G becomes false].

The guard G is the condition $i \neq m$, where m is a non-negative integer (by the pre-condition). By (I) and (II), we know that:

“for all integers $n \geq 0$, if the loop is iterated n times, then $I(n)$ is true i.e. $i = n$ and $product = n.x$ ”.

(This is exactly Statement (A) in the proof of the theorem with $I(n)$ as currently chosen). So, after m iterations of the loop, $i = m$. Thus, G becomes false after m iterations of the loop.

(IV) Correctness of the post-condition [If N is the smallest number of iterations after which G is false and $I(N)$ is true, then the values of the algorithm variables will be as specified in the post-condition of the loop].

According to the post-condition, the value of $product$ after execution of the loop should be $m.x$. But if G becomes

false after N iterations, $i = m$ (because G is: $i \neq m$, so G being false is $i = m$). And if $I(N)$ is true, then “ $i = N$ and $product = N.x$ ” (this is $I(n)$ with n replaced by N). Since both conditions G is false and $I(N)$ is true are satisfied, we have $m = i = N$ and so $product = N.x = m.x$ which is the value according to the post-condition.

Thus, (I) to (IV) are verified and therefore the loop is correct wrt its pre- and post- conditions. ■

(2) Correctness of the Euclidean Algorithm

This algorithm accepts integers A and B with $A > B \geq 0$ and computes their greatest common divisor $gcd(A, B)$. Just before the loop in the algorithm, variables a, b , and r have been introduced with $a = A, b = B, r = B$. So, with pre- and post- conditions included the loop looks like:

[precondition: A and B are integers with $A > B \geq 0, a = A, b = B, r = B$]

while ($b \neq 0$)

1. $r := a \text{ MOD } b$

2. $a := b$

3. $b := r$

end while

[postcondition: $a = \gcd(A, B)$]

We take the loop invariant to be

$I(n) : \gcd(a, b) = \gcd(A, B)$ and $0 \leq b < a$.

The guard condition G of the while loop is $G : b \neq 0$

We use the loop invariant theorem to prove that the while loop is correct wrt the given pre- and post- conditions.

Solution We have to verify the conditions (I) to (IV) in the Theorem 6.15 relative to the guard G and the choice made of the loop invariant $I(n)$.

(I) Basis property [$I(0)$ is true before the first iteration of the loop].

$I(0)$ is: $\gcd(a, b) = \gcd(A, B)$ and $0 \leq b < a$.

According to the pre-condition, $a = A, b = B$, and $0 \leq B < A$. Therefore $\gcd(a, b) = \gcd(A, B)$ and $0 \leq b < a$.

So $I(0)$ is true.

(II) Inductive property [If G and $I(k)$ are true before a loop iteration, where $k \geq 0$, then $I(k + 1)$ is true after the loop iteration].

Suppose k a non-negative integer such that G and $I(k)$ are both true before an iteration of the loop. Since G true, $b_{old} \neq$

0 and loop is entered. Next, since $I(k)$ is true, immediately before statement 1. is executed we have

$$\gcd(a_{old}, b_{old}) = \gcd(A, B) \text{ and } 0 \leq b_{old} < a_{old}. \quad (1)$$

After execution of statement 1.,

$$r_{new} = a_{old} \text{ MOD } b_{old}.$$

Therefore, by the definition of MOD (i.e. the division algorithm),

$$a_{old} = b_{old} \cdot q + r_{new} \quad \text{for some integer } q$$

and r_{new} has the property that

$$0 \leq r_{new} < b_{old}. \quad (2)$$

By Lemma 6.11,

$$\gcd(a_{old}, b_{old}) = \gcd(b_{old}, r_{new}),$$

and by Equation (1) we have

$$\gcd(a_{old}, b_{old}) = \gcd(A, B).$$

Hence,

$$\gcd(b_{old}, r_{new}) = \gcd(A, B). \quad (3)$$

When statements 2. and 3. are executed,

$$a_{new} = b_{old} \quad \text{and} \quad b_{new} = r_{new}. \quad (4)$$

Substitute equations (4) into equations (3) to get

$$\gcd(a_{new}, b_{new}) = \gcd(A, B). \quad (5)$$

By inequality (2), $0 \leq r_{new} < b_{old}$. So substitute into this the values from equation (4) to get

$$0 \leq b_{new} < a_{new}. \quad (6)$$

Hence after the iteration of the loop, by equation (5) and inequality (6) we get

$$\gcd(a, b) = \gcd(A, B) \quad \text{and} \quad 0 \leq b < a,$$

which is $I(k + 1)$, as required.

(III) Eventual falsity of the guard [After a finite number of iterations of the loop, G becomes false].

We already know this, because b (the smaller of the two numbers a, b) eventually becomes 0 i.e. G eventually becomes false.

(IV) Correctness of the post-condition [If N is the smallest number of iterations after which G is false and $I(N)$ is true, then the values of the algorithm variables will be as specified in the post-condition of the loop].

Suppose G false and $I(N)$ true. Since G is false, $b = 0$. Since $I(N)$ is true

$$\gcd(a, b) = \gcd(A, B). \quad (7)$$

Finally, substitute $b = 0$ into equation (7) to get $\gcd(a, 0) = \gcd(A, B)$. But by Lemma 6.10, $\gcd(a, 0) = a$. Hence $a = \gcd(A, B)$, which is the post-condition as required. ■

7 COUNTING ARGUMENTS

See Chapter 4 of Textbook.

Many situations in computing (eg. analysis of algorithms) and in mathematics involve analyzing the number of ways in which things can be arranged or the number of possibilities that can occur or in counting a total in two different ways and comparing the results. We consider some of these questions next.

7.1 Basic Principles

Multiplication Principle (Page 198)

If an activity can be constructed in t successive steps and step 1 can be done in n_1 ways; step 2 can then be done in n_2 ways; \dots ; step t can then be done in n_t ways, then the total number of different possible activities is $n_1 \times n_2 \times \dots \times n_t$.

Example 1 (Page 199).

- (a) How many strings of length 4 can be formed using the letters $ABCDE$ if repetition is not allowed?
- (b) How many strings of Part (a) begin with the letter B ?
- (c) How many strings of Part (a) do not begin with the letter B ?

Solution

(a) A string of length 4 can be constructed in four steps: choose 1st letter; choose 2nd letter; choose 3rd letter; choose 4th letter. The first letter can be chosen in five ways. The second letter can then be chosen in four ways (repetition is not allowed). The third letter can be chosen in three ways, and the fourth in two ways. By the Multiplication Principle, total number = $5 \times 4 \times 3 \times 2 = 120$ strings.

(b) Strings that begin with B can be constructed in four steps: choose the first letter (B); choose the second letter; choose the third letter; choose the fourth letter. There is one way to choose the first letter (B); 4 ways to choose 2nd; 3 ways to choose 3rd; 2 ways to choose 4th. By the Multiplication Principle, there are $1 \times 4 \times 3 \times 2 = 24$ strings that start with the letter B .

(c) Hence there are $120 - 24 = 96$ strings that don't start with the letter B . ■

Exercise Read Examples 4.1.3 and 4.1.4 in Text.

The following fact can be very useful when used in conjunction with the counting principles we consider.

7.2 Exercise Suppose that X and Y are finite sets. Show that X and Y have the same number of elements iff there is a bijective function $\phi : X \rightarrow Y$ of X onto Y . Show more generally that if $\phi : X \rightarrow Y$ is an injective function, then X and the image set $\phi(X)$ of X have the same number of elements.

Example 2 (Page 200)

Let $X = \{x_1, x_2, \dots, x_n\}$ be an n -element set. How many ordered pairs (A, B) of subsets of X satisfy $A \subseteq B \subseteq X$?

Solution (Modification of solution given on page 200).

Let $A, B \subseteq X$ be arbitrary subsets of X , and form the ordered pair (A, B) and the ordered triple $(A, B \setminus A, X \setminus B)$.

Claim 1

The mapping $\phi : \mathcal{P}(X) \times \mathcal{P}(X) \rightarrow \mathcal{P}(X) \times \mathcal{P}(X) \times \mathcal{P}(X)$ defined by $\phi(A, B) = (A, B \setminus A, X \setminus B)$ is an injective mapping.

Proof

Suppose $\phi(A, B) = \phi(A', B')$. Then $(A, B \setminus A, X \setminus B) = (A', B' \setminus A', X \setminus B')$. Therefore, by definition of equality of ordered triples, we have in particular that $A = A'$ and $X \setminus B = X \setminus B'$. But $X \setminus B = X \setminus B'$ implies $B = B'$. Thus, $A = A'$ and $B = B'$ and therefore, by definition of equality of or-

dered pairs, we have $(A, B) = (A', B')$ and so ϕ is injective as required.

Since $\mathcal{P}(X)$ is finite (it has 2^n elements), $\mathcal{P}(X) \times \mathcal{P}(X)$ and $\mathcal{P}(X) \times \mathcal{P}(X) \times \mathcal{P}(X)$ are both finite sets. Therefore, by Exercise 7.2, the number of pairs $(A, B) =$ the number of triples of the form $(A, B \setminus A, X \setminus B)$.

Claim 2

$A \subseteq B$ iff $(A, B \setminus A, X \setminus B)$ forms a partition of X ie iff the sets $A, B \setminus A$ and $X \setminus B$ are pairwise disjoint and their union is X . In fact, the union of the sets $A, B \setminus A, X \setminus B = (A \cup (B \setminus A)) \cup (X \setminus B) = B \cup (X \setminus B) = X$ anyway ie the union of the sets $A, B \setminus A, X \setminus B$ is always X .

(Exercise The sets $A, B \setminus A, X \setminus B$ are not necessarily pairwise disjoint eg. consider $X = \{x_1, x_2, x_3, x_4\}$, $A = \{x_1, x_3, x_4\}$, $B = \{x_1, x_2, x_3\}$).

So the claim is: $A \subseteq B$ iff the sets $A, B \setminus A, X \setminus B$ are pairwise disjoint.

Proof

Suppose $A \subseteq B$. Then obviously $A \cap (B \setminus A) = \emptyset$, $(B \setminus A) \cap (X \setminus B) = \emptyset$ and $A \cap (X \setminus B) = \emptyset$.

Conversely, suppose the sets $A, B \setminus A, X \setminus B$ are pairwise disjoint. We show that $A \subseteq B$. Suppose not ie suppose there is an $x \in A$ with $x \notin B$. Then $x \in A$ and $x \in X \setminus B$ which implies $x \in A \cap (X \setminus B)$ which is a contradiction since $A \cap (X \setminus B) = \emptyset$.

Thus, the pairs (A, B) with $A \subseteq B$ are in 1-1 correspondence with the triples $(A, B \setminus A, X \setminus B)$ which form a partition of X . Hence there is the same number of such pairs as there are these triples. But each such triple corresponds to an assignment of each element of X to one of the three coordinate positions $(\square, \square, \square)$ ie to $(A, B \setminus A, X \setminus B)$ ie we take each $x_i \in X$ and drop it into one of the coordinate boxes. How many ways can we do this? There are three ways we can drop x_1 into a coordinate position. Then there are 3 ways we can drop x_2 into a coordinate position etc. Thus, by the Multiplication Principle, there are $3 \times 3 \times 3 \times \dots \times 3$ (with n factors) $= 3^n$ ways, and this is the same as the number of pairs (A, B) with $A \subseteq B$ so the required number is 3^n . ■

Example 3

Consider the following nested loop:

for $i := 1$ to 4

for $j := 1$ to 3

[statements in body of inner loop. None contain
branching statements which lead out of the inner loop]
next j
next i

How many times will the inner loop be iterated when the algorithm is implemented and run?

Solution The outer loop is iterated 4 times and during each iteration of the outer loop, the inner loop is iterated 3 times. Hence by the Multiplication Principle the number required is $4 \times 3 = 12$.

The Addition Principle (Page 201)

Suppose that X_1, X_2, \dots, X_t are sets and that the i^{th} set X_i has n_i elements. If the sets X_1, X_2, \dots, X_t are pairwise disjoint (ie $X_i \cap X_j = \emptyset$ if $i \neq j$), then the number of possible elements that can be selected from X_1 or X_2 or $X_3 \dots$ or X_t is $n_1 + n_2 + \dots + n_t$. (Equivalently, the union $X_1 \cup X_2 \cup \dots \cup X_t$ contains $n_1 + n_2 + \dots + n_t$ elements).

Note How do we know which principle to use? If we are counting objects that are constructed in successive steps, we

use the Multiplication Principle. If we have pairwise disjoint sets of objects and we want to know the total number of them, we use the Addition Principle. Some problems involve both principles.

Example 4 (Page 201)

How many eight-bit strings begin either 101 or 111?

Solution

An eight-bit string beginning 101 can be constructed in five successive steps: select fourth bit, select fifth bit, \dots , select eighth bit. Each bit can be selected in two ways, so by Multiplication Principle there are $2 \times 2 \times 2 \times 2 \times 2 = 32$ strings starting 101. Similarly there are 32 starting 111. Since the sets of strings starting 101 and those starting 111 are disjoint, by the Addition Principle there are $32 + 32 = 64$ strings starting 101 or 111. ■

Exercise read Examples 4.1.7 and 4.1.8

Example 5

A computer access code word consists of from one to three letters chosen from the 26 in the alphabet with repetitions

allowed. How many different code words are possible?

Solution

The set of all code words can be partitioned into subsets consisting of those of length 1, those of length 2, and those of length 3:

Code words of length 1	Code words of length 2	Code words of length 3
------------------------	------------------------	------------------------

Number of length 1 = 26

Number of length 2 = $26 \times 26 = 26^2$ - by the Multiplication Principle (each word of length 2 is constructed in two steps: choose first letter (26 choices); choose second letter (26 choices)).

Number of length 3 = $26 \times 26 \times 26 = 26^3$ - by the Multiplication Principle again (check this as an exercise).

Therefore by the Addition Principle, the total number of code words is

$$26 + 26^2 + 26^3 = 18,278. \quad \blacksquare$$

7.3 Permutations and Combinations

7.4 Definition A *Permutation* of n distinct objects x_1, x_2, \dots, x_n is an ordering (or rearrangement) of the objects.

Example 6 Let $X = \{x_1, x_2, x_3\}$. Possible permutations of x_1, x_2, x_3 are: $x_1x_2x_3$; $x_1x_3x_2$; $x_2x_1x_3$; $x_2x_3x_1$; $x_3x_1x_2$; $x_3x_2x_1$. And these are all the possible permutations of x_1, x_2, x_3 .

Note

(1) The order *matters* ie $x_1x_2x_3$ and $x_1x_3x_2$ are *different* permutations of x_1, x_2, x_3 .

(2) We note that in each permutation, each element of X occurs exactly once. Therefore a permutation corresponds to a bijective function $X \rightarrow X$ determined by the ordering relative to the natural one: $x_1x_2x_3$.

eg. $x_2x_1x_3$ corresponds to $\phi : X \rightarrow X$ where $\phi(x_1) = x_2$, $\phi(x_2) = x_1$, $\phi(x_3) = x_3$. Eg. $x_3x_2x_1$ corresponds to $\theta : X \rightarrow X$ where $\theta(x_1) = x_3$, $\theta(x_2) = x_2$, $\theta(x_3) = x_1$.

Therefore, we could define a permutation simply to be a bijection $X \rightarrow X$, and this definition has the advantage that it applies equally well to infinite sets.

How many permutations are there on a set X of n elements?

7.5 Proposition There are $n! = n(n - 1)(n - 2) \dots 3.2.1$ permutations on a set X of n elements.

Proof A permutation on X can be constructed as follows:

Choose the 1st element - there are n ways.

Choose the 2nd element - there are $(n - 1)$ ways (repetition is not allowed).

Choose the 3rd element - there are $(n - 2)$ ways (repetition is not allowed).

⋮

Choose the $(n - 1)$ st element - there are 2 ways.

Choose the n th element - there is 1 way.

By the Multiplication Principle, there is a total of $n \times (n - 1) \times (n - 2) \times \dots \times 3 \times 2 \times 1 = n!$ ways of constructing a permutation. ■

Example 7

On a set X of three elements there are $3! = 3 \times 2 \times 1 = 6$ permutations. On a set X of four elements there are $4! = 4 \times 3 \times 2 \times 1 = 24$ permutations. On a set X of five elements there are $5! = 120$ permutations. ■

Example 8

How many permutations of the letters $ABCDEF$ contain the substring DEF ?

Solution

Think of a token T as representing DEF - it keeps DEF fixed. We are then really looking for permutations of $ABCT$ eg $ABTC$ ($= ABDEFC$), $ATCB$ ($= ADEFCB$). Hence we are looking for permutations of a four element set, and there are $4! = 24$ of these by Proposition 7.5. ■

Example 9 (Page 211)

How many permutations of the letters $ABCDEF$ contain the letters DEF together in any ordering?

Solution

Again let T be the token for DEF (which keeps them together). We can construct the required permutations in two steps:

First select an ordering of the letters DEF - there are $3! = 6$ ways.

Next, select an ordering of $ABCT$. By the previous example there are $4! = 24$ ways of doing this. Therefore in total there are $6 \times 24 = 144$ permutations of the type required. ■

7.6 Definition (Page 212) An r -permutation of n distinct elements x_1, x_2, \dots, x_n is an ordering of an r -element subset of $\{x_1, x_2, \dots, x_n\}$.

The number of r -permutations of a set of n distinct elements is denoted by $P(n, r)$. Notice that an n -permutation is just a permutation as defined earlier.

Example 10

Taking the three elements a, b, c , all possible 2-permutations are ab, ba, ac, ca, bc, cb .

What is the value of $P(n, r)$?

7.7 Proposition $P(n, r) = n(n-1)(n-2) \dots (n-r+1) = \frac{n!}{(n-r)!}$.

Proof An r -permutation (where $r \leq n$) is constructed as follows:

Choose 1st element - there are n ways.

Choose 2nd element - there are $(n-1)$ ways.

Choose 3rd element = there are $(n-2)$ ways.

⋮

Choose r th element - there are $(n-r+1)$ ways.

By the Multiplication Principle, there is a total of $n(n-1)(n-2) \dots (n-r+1)$ r -permutations.

Finally,

$$\frac{n!}{(n-r)!} = \frac{n(n-1)(n-2)\dots(n-r+1)(n-r)(n-r-1)\dots 3.2.1}{(n-r)(n-r-1)\dots 3.2.1}$$

$$= n(n-1)(n-2)\dots(n-r+1)$$

as required. ■

Example 11 (Page 213)

In how many ways can we select a chairperson, a vice chairperson, a secretary and a treasurer in that order from a group of ten persons?

Solution

We need to choose four people from 10 and the order matters. Therefore the number is

$$P(10, 4) = 10 \times 9 \times 8 \times 7 = 5040. \quad \blacksquare$$

Example 12

In how many ways can seven distinct Martians and five distinct Jovians wait in line if no two Jovians stand together.

Solution

We adopt a two step approach:

Step 1 Line up the Martians, the order matters. Therefore there are $7! = 5040$ ways to do this. A typical line up is

- M_1 - M_2 - M_3 - M_4 - M_5 - M_6 - M_7 -

Step 2

No two Jovians can stand together. Therefore, the five Jovians must be placed in the 8 blanks and the order matters. Thus, the number of ways to do Step 2 = $P(8, 5) = 8 \times 7 \times 6 \times 5 \times 4 = 6720$ ways.

Hence, by the Multiplication Principle, the total number of ways of lining up the Martians and the Jovians is $5040 \times 6720 = 33,868,800$.

Combinations

A selection of objects without regard to order is called a *combination*.

7.8 Definition Let $X = \{x_1, x_2, \dots, x_n\}$ be a set containing n distinct elements.

(a) An *r-combination* of X is an unordered selection of r elements ($r \leq n$) from X (ie an r -element subset of X).

(b) The number of r -combinations of a set of n distinct elements is denoted by $C(n, r)$ or $\binom{n}{r}$ (or ${}^n C_r$ – see later).

7.9 Proposition For $r \leq n$, we have $C(n, r)$

$$= \frac{P(n, r)}{r!} = \frac{n(n-1)(n-2)\dots(n-r+1)}{r!} = \frac{n!}{r!(n-r)!}.$$

Proof

We consider r -permutations in a two step process:

Step 1

First, select an r -combination ie an unordered subset of X containing r elements. There are $C(n, r)$ ways to do this (by definition of $C(n, r)$).

Step 2

Now order this chosen set of r -elements. By Theorem 7.5 (with $n = r$) there are $r!$ ways to do this.

By the Multiplication Principle, the number of ways of carrying out Steps 1 and 2 is $C(n, r) \times r!$

But the result of carrying out Steps 1 and 2 in all possible ways is precisely to generate all r -permutations and the number of these is $P(n, r)$

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

Therefore we get the equation $P(n, r) = C(n, r) \times r!$ so

$$C(n, r) = \frac{P(n, r)}{r!} \text{ as required. } \blacksquare$$

Example 13

In how many ways can we select a committee of three from a group of 10 distinct persons?

Solution A committee is an unordered group of people.

So required number = $C(10, 3) = \frac{10 \times 9 \times 8}{3 \times 2 \times 1} = 120$. ■

Example 14 (Page 216)

In how many ways can we select a committee of two women and three men from a group of five distinct women and six distinct men?

Solution The selections are unordered and therefore number of ways of choosing the women = $C(5, 2) = \frac{5 \times 4}{2!} = 10$.

Number of ways of choosing the men = $C(6, 3) = \frac{6 \times 5 \times 4}{3!} = 20$.

The committee can be constructed in two successive steps:

Step 1 Choose the women.

Step 2 Choose the men.

Therefore, by the Multiplication Principle the total number of committees = $10 \times 20 = 200$. ■

Exercise

Work through Examples 4.2.20 and 4.2.21 in Textbook.

Note on the Binomial Coefficients (See Proposition 1.8)

Consider $(a + b)^n$:

$$(a + b)^2 = a^2 + 2ab + b^2.$$

$$(a + b)^3 = a^3 + \frac{3 \times 1}{1}a^2b + \frac{3 \times 2 \times 1}{2 \times 1}ab^2 + b^3.$$

$$(a + b)^4 = a^4 + \frac{4 \times 1}{1}a^3b + \frac{4 \times 3}{2 \times 1}a^2b^2 + \frac{4 \times 3 \times 2}{3 \times 2 \times 1}ab^3 + b^4.$$

In general

$$(a + b)^n = a^n + {}^nC_1a^{n-1}b + {}^nC_2a^{n-2}b^2 + \dots + {}^nC_ra^{n-r}b^r + \dots + b^n.$$

The coefficients nC_r in this expansion are called the *binomial coefficients*. In fact, these coefficients are related to combinations by

$${}^nC_r = \frac{n!}{r!(n-r)!} = C(n, r).$$

Thus, the coefficient nC_r in the binomial expansion coincides exactly with the number $C(n, r)$ of r -combinations of n distinct elements. ■