

In this chapter, we explain mathematical notions, terminologies, and certain methods used in convincing logical arguments that we shall have need of throughout the book.

2.1 Set

A *set* is an assembly of things where “things” are objects such as natural numbers, real numbers, strings of symbols, and so on. A “thing” that constitutes a set is called an *element* of the set. In particular, when an element that constitutes a set is again a set, then the former set is called a *class*. In this case, instead of saying a set of sets, we say a *class of sets*. A set is described by enclosing the elements in braces, like $\{3, 7, 12\}$. A set composed of a finite number of elements is called a *finite set*, whereas a set of an infinite number of elements is called an *infinite set*. The set of natural numbers is expressed as $\{1, 2, 3, \dots\}$. The number of elements of a finite set A is called the *size* of set A and is denoted by $|A|$. A set whose size is zero is called an *empty set* and denoted by $\emptyset$. That is, $\emptyset$ denotes a set that has no elements. If a is an element of set A , then we say that a is contained in A , which is denoted by $a \in A$. On the other hand, $a \notin A$ means that a is not an element of set A . The set of all the natural numbers, that of all the integers, and that of all the real numbers are denoted by $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$, respectively.

Let A and B be sets. If any element of A is also an element of B , then A is a *subset* of B , which is denoted by $A \subseteq B$. Even if A equals B , $A \subseteq B$ holds by the definition. In particular, if $A \subseteq B$ and $A \neq B$, A is a *proper subset* of B , which is denoted by $A \subsetneq B$.

For sets A and B , the set that consists of all the elements A and those of B is the *union* of A and B , which is denoted by $A \cup B$. The set that consists of the elements that are contained in both A and B is the *intersection* of A and B , which is denoted by $A \cap B$.

Furthermore, the set that consists of the elements contained in A but not in B is the *difference* obtained by subtracting B from A , which is denoted by $A - B$. What is described above is illustrated in Fig. 2.1, in terms of figures which are called

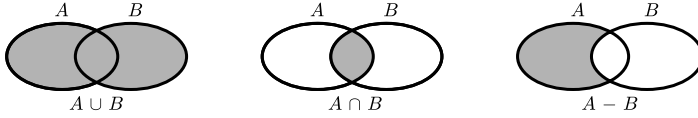
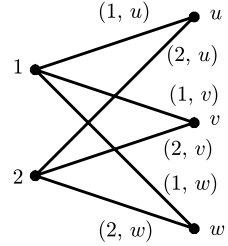


Fig. 2.1 Venn diagrams for union, intersection, and difference

Fig. 2.2 Cartesian product
 $\{1, 2\} \times \{u, v, w\}$



Venn diagrams. Let the universe, denoted by U , be the set of all elements under consideration. For a subset A of a set U , the difference set obtained by subtracting A from U is the *complement* of A in U , and when U is obvious, it is denoted by $\bar{A}$ and is simply called the complement of A .

The set that consists of all the subsets of a set A is called the *power set* of A and is denoted by $\mathcal{P}(A)$. For example, for $A = \{1, 2, 3\}$,

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

Let $P(x)$ denote a certain condition on x . The set that consists of elements x 's that satisfy the condition $P(x)$ is denoted by

$$\{x \mid P(x)\}.$$

For example, $\{n \mid n \text{ is a natural number that is divided by } 2\}$ denotes the set that consists of positive even numbers. The power set of A can be denoted by

$$\mathcal{P}(A) = \{B \mid B \subseteq A\},$$

in terms of this notation. What set does $\{(a, b) \mid a \in A, b \in B\}$ denote? For example, if $A = \{1, 2\}$ and $B = \{u, v, w\}$, the set denotes

$$\{(1, u), (1, v), (1, w), (2, u), (2, v), (2, w)\}.$$

If $P(a, b)$ in the notation above is taken to be the condition on (a, b) that both $a \in A$ and $b \in B$ hold, $\{(a, b) \mid a \in A, b \in B\}$ means the set of all the elements that satisfy the condition $P(a, b)$. As illustrated in Fig. 2.2, it consists of six pairs, each consisting of one from A and the other from B . In general, for sets A and B , $\{(a, b) \mid a \in A, b \in B\}$ is called the *Cartesian product* of A and B , and is denoted

by $A \times B$. Since the Cartesian product is frequently used in this book, the readers are expected to have a firm image of it in mind. The Cartesian product can be generalized for an arbitrary number of sets. For example, $A \times B \times C = \{(a, b, c) \mid a \in A, b \in B, c \in C\}$. In particular, the Cartesian product of k A s $A \times A \times \cdots \times A$ is denoted by A^k . In general, an element of A^k , denoted by $(a_1, a_2, \dots, a_k)$, is called a *k-tuple*. In particular, an element of the Cartesian product of two sets is called a 2-tuple or a *pair*.

There is a difference between a k -tuple and a set of size k , depending on whether or not we take into account the order of the elements associated with them. For example, the elements of $\mathbb{N}^3$ $(3, 7, 12)$, $(12, 7, 3)$, and $(7, 12, 3)$ are different 3-tuples, whereas $\{3, 7, 12\}$, $\{12, 7, 3\}$, $\{7, 12, 3\}$ as well as $\{3, 7, 7, 12\}$ are the same set since any of them consists of 3, 7, and 12. On the other hand, you may want to take into account how many times an element appears in a set. In such a case, a set is called a *multi-set*. For example, the multi-sets $\{3, 7, 12\}$ and $\{3, 7, 7, 12\}$ are different from each other, while $\{3, 7, 7, 12\}$ and, say, $\{3, 7, 12, 7\}$ are the same set.

2.2 Strings and Languages

A *string* is a sequence of symbols, which will be discussed frequently throughout this book. Suppose we are given a finite set of symbols. The strings we deal with are assumed to be composed of symbols taken from the set. Such a set is called an *alphabet* and will be denoted by Σ or Γ , etc. The number of symbols that appear in string w is called the *length* of w and is denoted by $|w|$. A string over Σ is one of a finite length that consists of symbols chosen from Σ . In particular, a string of length 0 is called an *empty string* and is denoted by ε . Though it is not easy to figure out what the empty string really is, it is indispensable, just as we cannot do without the empty set. Since the empty string is of length 0, and hence can be thought of as invisible, we denote it by the special symbol ε . The reason why we introduce the empty sequence will become clear later in the context of how it will be used. For example, concatenating the empty sequence ε and a sequence w does not change the original sequence, so both εw and $w\varepsilon$ turn out to be w .

Concatenation is just the connection of two strings to obtain a string. That is, connecting strings $a_1 \cdots a_m$ of length m and $b_1 \cdots b_n$ of length n makes the string $a_1 \cdots a_m b_1 \cdots b_n$ of length $m + n$. When it is necessary to indicate the operation of concatenation explicitly, the symbol “.” is used. So when we need to express the operation of concatenation explicitly, the string $a_1 \cdots a_m b_1 \cdots b_n$ obtained by connecting the strings $a_1 \cdots a_m$ and $b_1 \cdots b_n$ will be denoted by $a_1 \cdots a_m \cdot b_1 \cdots b_n$. The string $a_i a_{i+1} \cdots a_j$ which is a consecutive part of $a_1 \cdots a_n$ is called a *substring* of the original string. Note that as special cases both the string $a_1 \cdots a_n$ and the empty string are substrings of the string $a_1 \cdots a_n$.

A *language* is a set that consists of strings over an alphabet. The reason that we use the terminologies a language or an alphabet comes from the following. Taking English as an example, the alphabet turns out to be $\{a, b, c, \dots, y, z, \square\}$ consisting

of 26 symbols and the space symbol $\sqcup$, whereas the language turns out to be the set of correct sentences such as

the $\sqcup$ boy $\sqcup$ broke $\sqcup$ the $\sqcup$ window.

Since we pay no attention to aspects of English sentences such as correctness or meaning, we will use the terminology a string, rather than a sentence.

A set consisting of all the strings over an alphabet Σ is denoted by Σ^* . For example, if $\Sigma = \{0, 1\}$,

$$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}.$$

A language over Σ is a subset of Σ^* .

2.3 Functions and Problems

A *function* is what specifies the correspondence between elements in two sets. A function that defines the correspondence of elements of sets A to those of B is denoted by $f : A \rightarrow B$, and the element of B corresponding to an element a of A is denoted by $f(a)$, where a is said to be mapped to $f(a)$. A function is also called a *mapping*. The set A of a function $f : A \rightarrow B$ is called the *domain*, and B the *range*. If $f(a) \neq f(a')$ for any two different a and a' in A , the function $f : A \rightarrow B$ is called one-to-one. The condition can be interpreted as saying that different elements are differently named if element a of A is interpreted to be named $f(a)$. If for any element b of B , there exists an element a of A such as $f(a) = b$, then $f : A \rightarrow B$ is called a function *onto* B . The condition can be interpreted as saying that every name in B is used as a name of some element of A .

As examples of functions, consider the functions $f_{\text{add}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ and $f_{\text{mult}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ that represent addition and multiplication, respectively. These functions are defined by the equations $f_{\text{add}}(x, y) = x + y$ and $f_{\text{mult}}(x, y) = x \times y$, and are specified in Tables 2.1 and 2.2, respectively. In order to describe the correspondence completely in the form of tables as in Tables 2.1 and 2.2, the space for the tables must be infinite, which is impossible. Furthermore, notice that although when a in $f(a)$ is taken to be a pair (x, y) as in the case of addition we should write $f_{\text{add}}((x, y))$, but we simply denote it by $f_{\text{add}}(x, y)$.

Throughout this book, the term *problem* is used to mean function. Then, what is the difference between a function and a problem? These two notions differ only slightly in the sense that function is a concept in mathematics, while problem is used in the context where the correspondence associated with the problem might be expected to be computed. But it will turn out that there is a problem, such as the halting problem described in the preceding chapter, that can be defined but cannot be computed. The function f representing the halting problem is such that $f(\langle M \rangle) = 1$ if a Turing machine M eventually halts and $f(\langle M \rangle) = 0$ otherwise, where $\langle M \rangle$ denotes an appropriately coded sequence that represents a Turing machine M . So,

Table 2.1 $f_{\text{add}}(x, y)$

$x \backslash y$	1	2	3	4	...
1	2	3	4	5	...
2	3	4	5	6	...
3	4	5	6	7	...
4	5	6	7	8	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 2.2 $f_{\text{mult}}(x, y)$

$x \backslash y$	1	2	3	4	...
1	1	2	3	4	...
2	2	4	6	8	...
3	3	6	9	12	...
4	4	8	12	16	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

the halting problem can be represented as the function that decides whether a Turing machine halts eventually or continues to move around the states. The symbol H that denotes the halting problem represents the associated function as well, so that $f(\langle M \rangle) = H(\langle M \rangle)$ holds.

The membership problem for a language $L \subseteq \Sigma^*$ is a problem that, given a string w in Σ^* , asks whether $w \in L$ or $w \notin L$. So, the function that corresponds to the membership problem for a language L is defined as follows:

$$f(w) = \begin{cases} 1 & \text{for } w \in L, \\ 0 & \text{for } w \notin L. \end{cases}$$

A language $L \subseteq \Sigma^*$ and the function $f : \Sigma^* \rightarrow \{0, 1\}$ defined from L as above are substantially the same. In general, the set of elements of A that are mapped to b by a function $f : A \rightarrow B$ is denoted by $f^{-1}(b)$. That is, $f^{-1}(b) = \{a \mid f(a) = b\}$. Then, the equation $L = f^{-1}(1)$ holds between a language L and the function defined based on the language L . As mentioned above, since a language and the function associated with the language are substantially the same, solving the membership problem for a language L and computing the associated function $f : \Sigma^* \rightarrow \{0, 1\}$ are essentially the same task.

By the way, the answer for each question of either the halting problem or the membership problem is either affirmative or negative. Such a problem is called a decision problem or a *YES/NO problem*. In this book, affirmative and negative answers are expressed as 1 and 0, accepting and non-accepting, and YES and NO as well.

On the other hand, as in the case of addition and multiplication, there are problems that are not YES/NO problems. For example, for multiplication, (x, y) is an input and $x \times y$ is an output, and what defines the correspondence is the function $f_{\text{mult}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. In general, a value w or each instance w for a problem P that is substituted for x of a function $f(x)$ is called an *input* of the function or the problem, and the value returned, denoted by $f(w)$ or $P(w)$, is called its *output*.

2.4 Relations and Graphs

We can think of relations, such as the larger-than relation and the divided-by relation, in terms of a collection of pairs of elements that satisfy the corresponding relation. The collection of pairs, in turn, can be represented as a graph whose nodes represent the elements and whose edges represent the pairs.

To begin with, we take the larger-than relation, denoted by $>$, for the set $\{0, 1, \dots, 5\}$. The larger-than relation can be intuitively understood as the order on the number line. Such a relation can also be shown using arrows directing from a smaller number to a larger number, as shown in Fig. 2.3. If an arrow from m to n is denoted by (m, n) , the set of the arrows is the subset of the Cartesian product $\{0, 1, \dots, 5\} \times \{0, 1, \dots, 5\}$, that is, the set consisting of 15 pairs $\{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (0, 2), (1, 3), \dots, (0, 4), (1, 5), (0, 5)\}$.

Similarly, Fig. 2.4 shows the relation that indicates the difference of two numbers is even. That is, denoting the relation by R , mRn is defined to be such that $|m - n|$ is even. Rearranging the nodes in Fig. 2.4, we have Fig. 2.5, in which $\{0, 1, \dots, 5\}$ is partitioned into the two groups of even and odd numbers. Here, R is used to connect the numbers that are in the relation such as $0R4$, $4R2$, $1R3$, and it is also used to denote the set consisting of pairs (m, n) that are in the relation mRn . That is, in our case, R denotes the set consisting of 18 pairs $\{(0, 0), (2, 2), (4, 4), (0, 2), (2, 0), (2, 4), (4, 2), \dots, (5, 1), (1, 5)\}$ as well.

In general, a *relation* R on a set S is defined to be a subset of $S \times S$. Then, $(a, a') \in R$ is also denoted by aRa' . In this way, a subset of $S \times S$ is considered to be a relation on S . In particular, let us assume that a relation R on S satisfies the following three conditions.

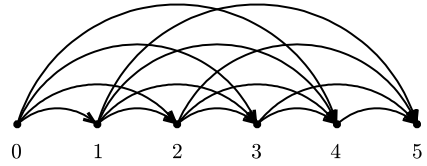
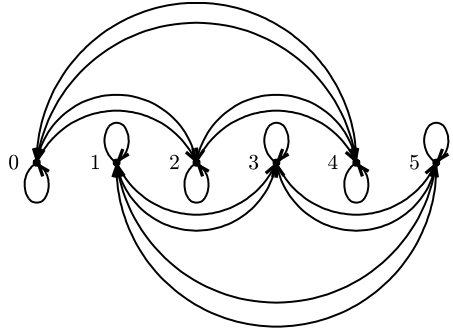
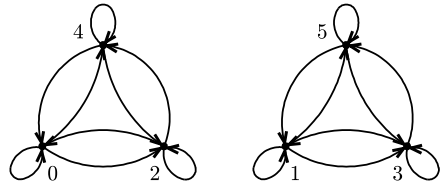
Reflexive law: xRx for all $x \in S$

Symmetric law: xRy implies yRx for all x and $y \in S$

Transitive law: xRy and yRz implies xRz for all x, y and $z \in S$

If a relation R satisfies the three conditions above, the relation R is called an *equivalence relation*. An equivalence relation R on a set S partitions S into blocks in such a way that any x and y with xRy are placed in the same block, and any different blocks do not overlap. Precisely speaking, a partition is defined as follows. Let us denote the class of subsets of a set S by $A = \{A_i \mid i \in I\}$, where $A_i \subseteq S$ and I denotes the set of subscripts that identify the subsets of S . A is called a *partition* if the following two conditions hold:

- (1) For any i and j with $i \neq j$, $A_i \cap A_j = \emptyset$, and
- (2) $S = \bigcup_{i \in I} A_i$

Fig. 2.3 Size relation**Fig. 2.4** Relation that difference is even**Fig. 2.5** Another graph drawing of the relation that the difference is even

where a subset A_i of S is called a *block*. The details are left to Exercise 2.3**, in which it is proved that the equivalence relation R and the partition A are equivalent notions to each other. Namely, an equivalence relation R induces the partition by placing elements in the relation R into the same block. Conversely, a partition A induces the equivalence relation R by relating elements in the same block.

For the example above, the larger-than relation $>$ satisfies the transitive law, but does not satisfy the reflexive and symmetric laws. On the other hand, the relation R indicating that the difference of two numbers is even satisfies the above three relations. Hence, R is an equivalence relation which induces the partition $\{\{0, 2, 4\}, \{1, 3, 5\}\}$ of the set $\{0, 1, \dots, 5\}$, as shown in Fig. 2.5.

A *directed graph* is a pair of a set of nodes and a set of edges connecting two nodes. A *node* is also called a *vertex*, and an *edge* is called an *arrow* or *arc*. Figure 2.6 shows an example of a directed graph in which the set of nodes is $\{1, 2, 3, 4\}$ and the set of edges is

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

Fig. 2.6 Example of a directed graph

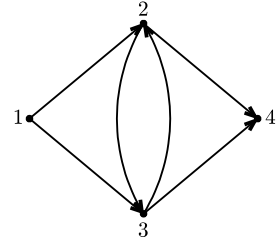
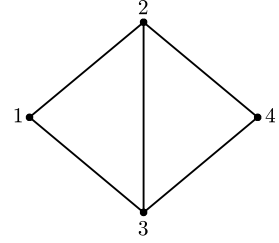


Fig. 2.7 Example of an undirected graph



A relation represented as a collection of pairs and a directed graph represented as a collection of nodes somehow connected with edges are slightly different in their expressions, but they express substantially the same thing. That is, looking at graphs we can easily figure out how nodes, representing elements, are related with each other.

If we disregard the direction of edges in a directed graph, as shown in Fig. 2.7 which corresponds to Fig. 2.6, we have the notion of an undirected graph. An undirected graph is defined to consist of a set of nodes and a set of undirected edges. In another words, in the case of an undirected graph, both (i, j) and (j, i) represent the same edge that connects nodes i and j . So to show that the direction is disregarded, edge (i, j) of an undirected graph is sometimes denoted by $\{i, j\}$ as well.

The **degree** of a node in an undirected graph is the number of edges that are connected to the node. For the example shown in Fig. 2.7, the degree of node 1 is 2 and that of node 2 is 3. On the other hand, in the case of directed graphs, we define notions of outdegree and indegree. The **outdegree** of a node in a directed graph is the number of edges that leave that node, while the **indegree** of a node is the number of edges that enter the node. For the example shown in Fig. 2.6, the indegree of node 1 is 0 and its outdegree is 2, and the indegree and the outdegree of node 2 are both 2.

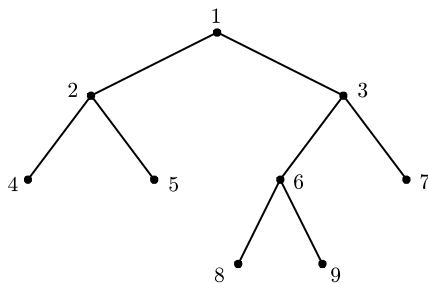
Formally, a **graph** is defined to be a pair (V, E) of a set V of nodes and a set E of edges. The directed graph shown in Fig. 2.6 is expressed as

$$(\{1, 2, 3, 4\}, \{(1, 2), (1, 3), (2, 3), (3, 2), (2, 4), (3, 4)\}),$$

and the undirected graph shown in Fig. 2.7 is expressed as

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

Fig. 2.8 Example of a tree where the root is 1 and the leaves are 4, 5, 8, 9, and 7



No matter whether it is a directed graph or an undirected graph, a **path** in a graph is a sequence of consecutive edges which is represented by $(v_0, v_1), (v_1, v_2), \dots, (v_{m-1}, v_m)$. A **simple path** is a path in which any node appears at most once. The **length of a path** is the number of edges in the path. If a path starts at a node and ends at the same node, the path is called a **closed path** or **cycle**.

If for any nodes v and v' in a graph there exists a path that connects from v to v' , the graph is called a connected graph. Note that in the case of a connected directed graph, it follows from the definition that for any node v and v' there exist a path from v to v' as well as a path from v' to v . If a connected undirected graph with one distinguished node, called a **root**, is such that there exists no closed path, then the graph is called a *rooted tree*, or simply a **tree**. Figure 2.8 shows an example of a tree. In the case of this example, node 1 is the root, while nodes 4, 5, 8, 9, and 7 are called leaves. As you can see from this example, a node in a tree is a **leaf** if a path from the root to that node cannot be extended any further.

For a graph $G = (V, E)$ and a subset $V' \subseteq V$, the subgraph G' of G induced from V' is one obtained by leaving only edges that connect nodes, both from V' . For example, let a graph G be the directed graph given in Fig. 2.6; then the subgraph G' induced from $\{1, 2, 3\}$ is $(\{1, 2, 3\}, \{(1, 2), (1, 3), (2, 3), (3, 2)\})$. Formally, the subgraph of $G = (V, E)$ induced from $V' \subseteq V$ is defined to be $(V', E \cap (V' \times V'))$.

In general, when we are given a relation that does not satisfy the transitive law, we can transform it into a transitive relation by adding edges appropriately. The relation obtained this way from relation R by adding as few edges as possible is the *transitive closure* of R .

By expressing a relation as the directed graph, we shall give relations and their corresponding transitive closures. The transitive closure of the relation given as Fig. 2.6 is illustrated in Fig. 2.9. Similarly, the transitive closure of the relation given as Fig. 2.10 is illustrated in Fig. 2.3. Clearly the transitive closures obtained above satisfy the transitive law.

Let a relation R be expressed as the directed graph G associated with the relation R . The *transitive closure* of R , denoted by $cl(R)$, is defined to be the graph obtained from G by adding every pair (v, v') of nodes as an edge as long as there exists a path in G from v to v' . It is easy to see that the transitive closure $cl(R)$ defined this way is the minimum in terms of the inclusion relation among the relations that include R and satisfy the transitive law.

Fig. 2.9 Transitive closure of the directed graph in Fig. 2.6

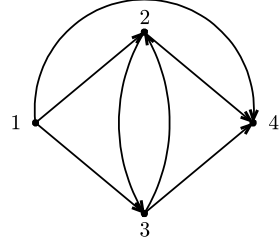


Fig. 2.10 Transitive closure of the relation given by this figure is given by Fig. 2.3



Table 2.3 Incidence matrix of the graph in Fig. 2.6

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 2.4 Incidence matrix of the graph in Fig. 2.7

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Before ending this section, we describe how to represent a graph. One way to represent a graph is to give a set V of nodes and a set E of edges, each set being expressed by listing its elements with appropriate punctuation marks between the elements. We also need to consider how to represent elements in these sets. For example, when $V = \{1, \dots, n\}$, node i can be represented as the corresponding binary number. Alternatively, a graph can also be represented as the $n \times n$ *adjacency matrix* defined as follows: the (i, j) element of the incidence matrix is 1 if (i, j) is an edge and 0 otherwise. Tables 2.3 and 2.4 give the incidence matrices for the graphs shown in Figs. 2.6 and 2.7, respectively. It is clear that an incidence matrix for an undirected graph is a *symmetric* one, that is, the (i, j) element is equal to the (j, i) element for any i and j .

2.5 Boolean Operations and Boolean Formulas

We shall explain *Boolean variables* that take the *Boolean values* of 0 or 1 and the *Boolean operations* applied to them. The mathematical system based on these concepts is called a *predicative logic* or *Boolean algebra*. If 1 is related to the truth and 0 to the false, the propositional logic becomes the basis of discussing the correctness

of inferences as well as the basis of exploring the relationship between inputs and outputs of the gates that constitute the hardware of computers.

A *Boolean formula* is one obtained by applying operations $\vee$, $\wedge$, and $\neg$ to *Boolean variables* that take the values 0 or 1. The operations $\vee$, $\wedge$, and $\neg$ are called *disjunction*, *conjunction*, and *negation*, respectively, and are defined as follows:

$$\begin{array}{lll} 0 \vee 0 = 0, & 0 \wedge 0 = 0, & \neg 0 = 1, \\ 0 \vee 1 = 1, & 0 \wedge 1 = 0, & \neg 1 = 0, \\ 1 \vee 0 = 1, & 1 \wedge 0 = 0, & \\ 1 \vee 1 = 1, & 1 \wedge 1 = 1. & \end{array}$$

We can interpret these operations by regarding Boolean values 1 and 0 as having a high or a low voltage, or indicating that a statement concerned holds or does not hold. That is, $x_1 \vee x_2$ means that “ x_1 is true” or “ x_2 is true,” whereas $x_1 \wedge x_2$ means that “ x_1 is true” and “ x_2 is true.” Furthermore, $\neg x$ means that the truth and falsity of x are inverted. $\neg x$ is also denoted by $\bar{x}$. As an example of a Boolean formula, let us consider

$$\overline{(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge \bar{x}_2)}.$$

According to the interpretations of $\vee$, $\wedge$, and $\neg$ mentioned above, it turns out that the Boolean formula can be interpreted as “ x_1 is 0 and x_2 is 1” or “ x_1 is 1 and x_2 is 0.” This is because the formula is interpreted as the negation of “both x_1 and x_2 are 1’s, or both x_1 and x_2 are 0’s.” In the following, this fact is derived by applying a transformation to the Boolean formula.

In general, for any Boolean formulas F , G , H , the *distributive law* expressed as

$$\begin{aligned} F \wedge (G \vee H) &= (F \wedge G) \vee (F \wedge H), \\ F \vee (G \wedge H) &= (F \vee G) \wedge (F \vee H) \end{aligned}$$

holds. It can easily be checked that these equations hold by substituting all the combinations of 0’s and 1’s for F , G , and H . Relating $\wedge$ to $\times$ and $\vee$ to $+$, the distributive law for the first equation is the same as that for the usual arithmetic equation. Furthermore, the second equation claims that the distributive law holds even if the $\vee$ ’s and $\wedge$ ’s in the first equation are interchanged. Similarly, the distributive law in which operations $\vee$ and $\wedge$ are applied from the right-hand side also holds as follows:

$$\begin{aligned} (G \vee H) \wedge F &= (G \wedge F) \vee (H \wedge F), \\ (G \wedge H) \vee F &= (G \vee F) \wedge (H \vee F). \end{aligned}$$

Furthermore, for any Boolean formulas F and G ,

$$\begin{aligned}\overline{F \vee G} &= \overline{F} \wedge \overline{G}, \\ \overline{F \wedge G} &= \overline{F} \vee \overline{G}\end{aligned}$$

hold, which is called *De Morgan's law*. The first equation holds because the negation of “ F or G are true” is equivalent to saying “both F and G are false,” while the second equation holds because the negation of “both F and G are true” is equivalent to saying “ F is false or G is false.” These relations are verified by substituting all the combinations of 0's and 1's for F and G in the formulas. Replacing Boolean variables F and G by variables A and B for sets, respectively, and replacing the disjunction $\vee$, the conjunction $\wedge$, and the negation $\neg$ by the set operations of the union $\cup$, the intersection $\cap$, and the complement $\bar{}$, respectively, we can obtain De Morgan's law for sets:

$$\begin{aligned}\overline{A \cup B} &= \overline{A} \cap \overline{B}, \\ \overline{A \cap B} &= \overline{A} \cup \overline{B}.\end{aligned}$$

By applying De Morgan's law and the distributive law to $\overline{(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge \bar{x}_2)}$, we can transform the formula to obtain the following:

$$\begin{aligned}\overline{(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge \bar{x}_2)} &= \overline{(x_1 \wedge x_2)} \wedge \overline{(\bar{x}_1 \wedge \bar{x}_2)} \\ &= (\bar{x}_1 \vee \bar{x}_2) \wedge (\bar{\bar{x}}_1 \vee \bar{\bar{x}}_2) \\ &= (\bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee x_2) \\ &= (\bar{x}_1 \wedge x_1) \vee (\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2) \vee (\bar{x}_2 \wedge x_2) \\ &= (\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2).\end{aligned}$$

The equation derived says that the original formula is equivalent to saying “ $x_1 = 0$ and $x_2 = 1$, or $x_1 = 1$ and $x_2 = 0$.” In transforming the formula, we used equations such as $\bar{\bar{x}} = x$ and $x \wedge \bar{x} = 0$ in addition to De Morgan's law and the distributive law.

In expressing Boolean formulas, the operation symbol $\wedge$ is often omitted throughout this book. For example, “just two of x_1, x_2, x_3 are 1's” is expressed as $x_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3$, and “at least two of x_1, x_2, x_3 are 1's” is expressed as $x_1x_2 \vee x_2x_3 \vee x_3x_1$.

2.6 Propositions and Proofs

A *proposition* is what claims an assertion whose validity is clearly determined. In particular, when a proposition includes variables, its validity is determined after substituting Boolean values for the variables. For propositions composed of various propositions, we shall explain the procedures to determine whether a composed proposition is true or false.

Table 2.5 Illustration explaining $P(n) \Rightarrow Q(n)$ and $P(n) \Rightarrow R(n)$

n	$P(n)$: n is divided by 6	$Q(n)$: n is divided by 3	$R(n)$: n is divided by 2
1	0	0	0
2	0	0	1
3	0	1	0
4	0	0	1
5	0	0	0
6	1	1	1
7	0	0	0
8	0	0	1
9	0	1	0
10	0	0	0
11	0	0	1
12	1	1	1
13	0	0	0
14	0	0	1
15	0	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Given propositions P and P' , we have a proposition “ P implies P' ” which is usually denoted by $P \Rightarrow P'$. Similarly, “ P implies P' , and P' implies P ” is denoted by $P \Leftrightarrow P'$. When $P \Leftrightarrow P'$, P and P' are said to be *equivalent* with each other. Furthermore, the proposition “both P and P' are true” is denoted by $P \wedge P'$, “either P or P' is true” as $P \vee P'$, and “ P is false” as $\overline{P}$. In this way, new propositions are obtained by applying $\Rightarrow$, $\Leftrightarrow$, $\vee$, $\wedge$, and $\overline{}$, etc., to variables.

Let n be a natural number. We consider the propositions $P(n)$, $Q(n)$, and $R(n)$ in terms of parameter n as follows:

$P(n)$: n is divided by 6,

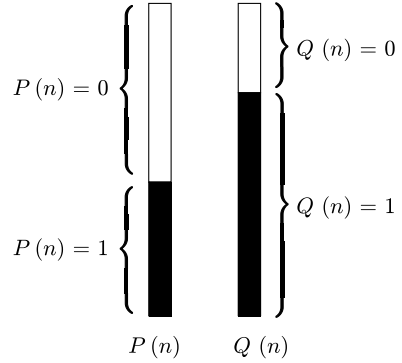
$Q(n)$: n is divided by 3,

$R(n)$: n is divided by 2.

If n is divided by 6, it is also divided by 3 and 2. Hence, clearly $P \Rightarrow Q$ and $P \Rightarrow R$ hold. This fact is also easily seen from Table 2.5, which shows whether $P(n)$, $Q(n)$, and $R(n)$ hold or not by means of 1 or 0. From Fig. 2.11, which schematically shows the relationship between $P(n)$ and $Q(n)$, we can see that not only $P \Rightarrow Q$ holds, but also $P \Rightarrow Q$ is equivalent to $\overline{Q} \Rightarrow \overline{P}$. This is because “for any n , $P(n) = 1$ implies $Q(n) = 1$ ” implies “for any n , $Q(n) = 0$ implies $P(n) = 0$.” Notice that the reverse of this implication, namely, “for any n , $Q(n) = 0$ implies $P(n) = 0$ ” implies “for any n , $P(n) = 1$ implies $Q(n) = 1$,” also holds.

$$\overline{Q} \Rightarrow \overline{P}$$

Fig. 2.11 Explanation of
 $\frac{P(n) \Rightarrow Q(n)}{Q(n) \Rightarrow P(n)}$ and
 $\frac{Q(n) \Rightarrow P(n)}{P(n) \Rightarrow Q(n)}$



is called the *contraposition* of $P \Rightarrow Q$. In general, a proposition is equivalent to its contraposition.

Let P and Q denote conditions on the same collection of variables, and let $D(P)$ and $D(Q)$ denote the sets of the values of the variables that satisfy P and Q , respectively. Since $P \Rightarrow Q$ means that values of the variables that satisfy P also satisfy Q , $P \Rightarrow Q$ is equivalent to $D(P) \subseteq D(Q)$. Furthermore, since $P \Leftrightarrow Q$ is equivalent to “ $P \Rightarrow Q$ and $Q \Rightarrow P$,” $P \Leftrightarrow Q$ is equivalent to $D(P) = D(Q)$. We can examine these relations by using the condition $P(n)$, $Q(n)$, and $R(n)$ described above. As illustrated in Fig. 2.5, we can see $P(n)$ is equivalent to $Q(n) \wedge R(n)$. Furthermore, we have the following equations:

$$D(P \wedge Q) = D(P) \cap D(Q), \quad D(P \vee Q) = D(P) \cup D(Q), \\ D(\overline{P}) = \overline{D(P)}.$$

Next, we describe how to make an argument to prove propositions. We begin with a proposition of the form $P \Rightarrow Q$ and explain how to prove that type of proposition. We discuss two types of arguments to prove such propositions: proof by contraposition and proof by contradiction.

Proof by contraposition is the method of proving $P \Rightarrow Q$ by deriving $\overline{Q} \Rightarrow \overline{P}$. The reason why we can argue this way is based on the fact that $P \Rightarrow Q$ is equivalent to $\overline{Q} \Rightarrow \overline{P}$. On the other hand, *proof by contradiction* is the method of proving $P \Rightarrow Q$ by showing that, if we assume P and $\overline{Q}$, then a contradiction follows. The fact that “ P and $\overline{Q}$ ” leads logically to a contradiction means that “ $P = 1$ and $Q = 0$ ” never happens. Therefore, the permissible value of (P, Q) is either one of $(1, 1)$, $(0, 1)$, or $(0, 0)$, thereby proving that $P \Rightarrow Q$ holds, as is easily seen in Fig. 2.11. Note that if we can show that the assumption “ P and $\overline{Q}$ ” leads to a contradiction, we can consequently conclude $\overline{Q} \Rightarrow \overline{P}$. Furthermore, when the proposition that we want to prove simply takes the form Q rather than $P \Rightarrow Q$, we can prove the proposition by verifying that the assumption $\overline{Q}$ leads to a contradiction, thereby proving Q . Comparing these two proof methods to prove $P \Rightarrow Q$, we can see that, in the case of proof by contraposition, we show that assuming $\overline{Q}$ leads to $\overline{P}$, whereas, in the case of proof by contradiction, we derive assuming that not only $\overline{Q}$ but also P

leads to a contradiction. But since the difference between these two proof methods is somewhat subtle, you may verify how they work in the following examples.

Example 2.1 Let x , y , and z be real numbers and let P and Q be conditions described as follows:

$$P: \quad x + y + z \geq 0,$$

$$Q: \quad \text{at least one of } x, y, z \text{ is greater than or equal to } 0.$$

We prove $P \Rightarrow Q$ by deriving $\overline{Q} \Rightarrow \overline{P}$ by *proof by contraposition*.

Proof Assume the negation of Q , namely, all x , y , and z are less than 0. Then clearly we have $x + y + z < 0$, thereby $\overline{P}$ is derived. Thus, $\overline{Q} \Rightarrow \overline{P}$ holds. $\square$

Note that, in the example above, the reason why proof by contraposition makes it easy to prove is because, as compared to the condition “at least one of x , y , and z is more than or equal to 0,” the condition “all of x , y , and z are less than 0” is easy to deal with, each of x , y , and z being referred to independently.

Example 2.2 Let P be as follows:

$$P: \quad \text{there exist an infinite number of prime numbers.}$$

In order to prove P based on *proof by contradiction*, we shall derive a contradiction by supposing $\overline{P}$.

Proof Assume the negation $\overline{P}$ of P , namely, “the number of prime numbers is finite.” So, let the primes be $p_1, p_2, \dots, p_m$. Then the natural number $p_1 p_2 \cdots p_m + 1$ is not divided by any of the prime numbers $p_1, p_2, \dots, p_m$ because the residue of the division is always 1. Therefore, that natural number is a prime number. This contradicts the fact that all of the prime numbers are listed as $p_1, p_2, \dots, p_m$. Since either P or $\overline{P}$ holds and $\overline{P}$ is negated by the contradiction, P holds. $\square$

Example 2.3 Let i , j , and k be natural numbers and let P and Q be as follows:

$$P: \quad i^2 + j^2 = k^2,$$

$$Q: \quad \text{at least one of } i, j, k \text{ is even.}$$

We shall show that assuming P and $\overline{Q}$ leads to a contradiction, thereby verifying $P \Rightarrow Q$ by proof by contradiction.

Proof Assume the negation of Q , namely, “any of i , j , and k is odd.” On the other hand, the square of an odd number, expressed as $2m + 1$, is given by

$$(2m + 1)^2 = 4(m^2 + m) + 1.$$

Hence the square of any odd number is odd. Since $\overline{Q}$ implies that any of i^2 , j^2 , and k^2 is odd, $i^2 + j^2$ is even, while k^2 is odd. Thus, this contradicts the assumption that $i^2 + j^2 = k^2$, which proves that $P \Rightarrow Q$. $\square$

As the next method of proof, we discuss mathematical induction. The proposition that we want to prove is denoted by $P(n)$, which is described in terms of a positive integer n . Mathematical induction is the method used to prove that $P(n)$ holds for all n . In order to grasp the method intuitively, we consider an infinite sequence of domino tiles. We can intuitively accept the following: if we can verify the two statements, namely, “the first domino tile falls down” and “for any positive integer n if the n th domino tile falls down, then the $(n + 1)$ th domino tile falls down,” then we can conclude that all of the domino tiles fall down. If we correspond the statement “the n th domino tile falls down” to the proposition “ $P(n)$ holds,” then what we conclude that the domino argument claims that $P(n)$ holds for all n . This is exactly what we want to derive. So all we need to do is to show the following statements (1) and (2), which correspond to the domino counterparts. Let

(1) $P(1)$ holds,

(2) For any positive integer n , if $P(n)$ holds, then $P(n + 1)$ also holds

Mathematical induction is the method for proving that $P(n)$ holds for all positive integer n 's by verifying that (1) and (2) above hold.

Example 2.4 Let $P(n)$ express “ $n^3 - n$ is divided by 3.” By mathematical induction we shall prove that $P(n)$ holds for any $n \geq 1$.

Proof First, $P(1)$ holds since $1^3 - 1 = 0$ is divided by 3. Next, we shall derive that if $P(n)$ holds, $P(n + 1)$ also holds. We have

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

Hence, if $n^3 - n$ is divided by 3 (that is, if $P(n)$ holds), then $(n + 1)^3 - (n + 1)$ is also divided by 3 (that is, $P(n + 1)$ holds). $\square$

2.7 Descriptions of Algorithms

An *algorithm* is a procedure that can be automatically executed to solve a problem. Employing an example, we shall explain how to describe algorithms throughout this book.

What we take as an example is the problem to ask whether or not, given a directed graph G and its two nodes s and t , there exists a path from s to t . It is called the *reachability problem*. First, an idea for solving it will be explained.

Fig. 2.12 Explanation of behavior of algorithm *REACH*

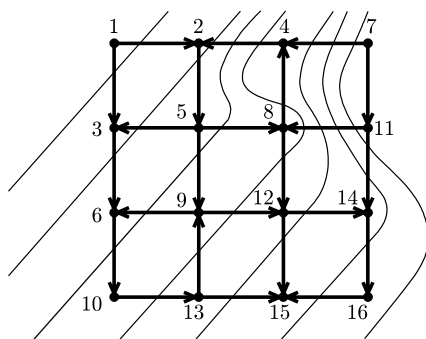


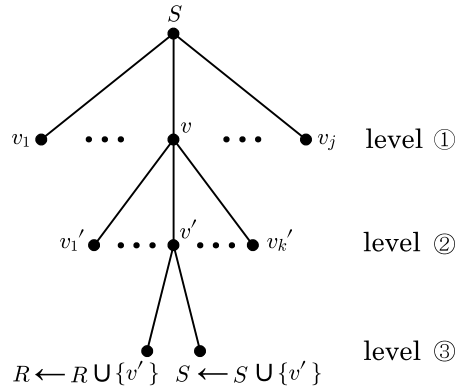
Figure 2.12 illustrates how an algorithm decides whether the node t is reachable from node $1 (= s)$ or not. The algorithm accumulates repeatedly the nodes that are reachable from node 1 to set R . It is shown in the figure that just as the wavefront travels, the algorithm repeatedly updates set R consisting of nodes reachable from node 1: starting with $R = \{1\}$; add all the nodes reachable from node 1, letting $R = \{1, 2, 3\}$; further add the reachable nodes from $\{1, 2, 3\}$, setting $R = \{1, 2, 3, 5, 6\}$; similarly let $R = \{1, 2, 3, 5, 6, 8, 9, 10\}$ and so on until no further nodes are added to R . After all the reachable nodes are added to R this way, the algorithm decides whether or not node t is reachable from node s depending on whether or not node t belongs to R .

In general, steps of an algorithm are divided into a number of groups called stages, each performing certain small intended tasks. In this case the algorithm, denoted *REACH*, is divided into three stages, which are numbered **1**, **2** and **3** as illustrated below. In the algorithm, R is used to denote the set of nodes that are found so far as the nodes reachable from node s , whereas S is used to denote the set of nodes that belong to R and are not yet checked to collect further nodes by going out from nodes in S along edges.

In stage **1**, R and S are set to be the sets consisting of only the node s . In stage **2**, to collect further nodes not yet found to be reachable, if there exists a node v in S , find all the new nodes v' that are connected from v by edges, and add them to both R and S ($R \leftarrow R \cup \{v'\}$, $S \leftarrow S \cup \{v'\}$), and finally the node v is removed from S ($S \leftarrow S - \{v\}$), where $\leftarrow$ means that what is expressed by its right-hand side is substituted into the left-hand side. So $S \leftarrow S \cup \{v'\}$, for example, simply means to add node v' to the collection of nodes in S . Repeat the update of sets R and S in this way until no further nodes are added. When S comes to be empty, hence no further nodes are being added, the algorithm proceeds to stage **3**. In stage **3**, the algorithm outputs YES if node t belongs to R and NO otherwise. This finishes the explanation of how the algorithm *REACH* works.

We present the algorithm *REACH* below. The description consists of the name of the algorithm, the input format, and finally the body of the algorithm consisting of the three stages.

Fig. 2.13 Hierarchical structure of indentation



Algorithm REACH

Input: $\langle G, s, t \rangle$, where s and t are the nodes of the graph G and the set of edges of G is denoted by E_G .

1. $R \leftarrow \{s\}, S \leftarrow \{s\}.$
2. While $S \neq \emptyset$, do the following for each v in S
 - for each (v, v') in E_G with $v' \notin R$, do the following
 - $R \leftarrow R \cup \{v'\},$
 - $S \leftarrow S \cup \{v'\}.$
 - $S \leftarrow S - \{v\}.$
3. If $t \in R$, output YES.
If $t \notin R$, output NO.

In the description of the algorithm, the first line shows that the name of the algorithm is *REACH*. The second line shows that the input is graph G together with its two nodes s and t , which are denoted as $\langle G, s, t \rangle$. In general, $\langle A \rangle$ denotes a description of entity A . For example, in the case that A is a graph, as mentioned in Sect. 2.4, $\langle G \rangle$ may be a list of the nodes and a list of the edges of the graph, or alternatively, a list of the elements of the adjacency matrix of the graph. In particular, when the entity is a string w , an input is described as w instead of $\langle w \rangle$ because in that case an algorithm can receive the string as input.

Taking *REACH* as an example of an algorithm, we explain how the *indentation* controls the execution of the algorithm. In the case of *REACH*, there are three levels ①, ②, and ③ of the indentation, as shown in the algorithm. In fact, the marks ①, ②, and ③ together with the associated vertical lines are just for explanation. When algorithms are written in practice, the levels of indentation are shown by just shifting the position of each line according to the corresponding level of the indentation.

Focusing exclusively on stage 2, Fig. 2.13 illustrates how the indentation controls the execution of *REACH*. Stage 2 is executed as follows: at level ①, for each node v in S , the one lower level ② (from the line beginning “for each (v, v') ” to the

line of $S \leftarrow S \cup \{v\}$ is executed repeatedly; at level ②, for each edge (v, v') , the one further lower level ③ (consisting of $R \leftarrow R \cup \{v'\}$, $S \leftarrow S \cup \{v'\}$) is executed repeatedly.

Figure 2.13 illustrates the moment when v is chosen from $S = \{v_1, \dots, v, \dots, v_j\}$ in level ① and v' is chosen from $\{v'_1, \dots, v', \dots, v'_k\}$ in level ②, where edges out of node v are denoted by $(v, v'_1), \dots, (v, v'), \dots, (v, v'_k)$. Note that algorithm *REACH* works well no matter in what order you choose node v and v' mentioned above.

2.8 Problems

2.1 Give the size of the power set of the set $\{1, 2, \dots, n\}$ in terms of n .

2.2 Let m and n be the sizes of sets A and B , respectively. Give the number of functions $f: A \rightarrow B$.

2.3** Show that a partition P is derived from a relation R that satisfies the reflexive, symmetric, and transitive laws. To do so, show how P is defined from R and that P defined so becomes a partition. Conversely, show that a relation R that satisfies the reflexive, symmetric, and transitive laws is derived from a partition P . To do so, show how R is defined from P and that R defined so satisfies the reflexive, symmetric, and transitive laws.

2.4** In the following, we give the proof of the statement that “a relation that satisfies the symmetric and transitive laws also satisfies the reflexive law.” Find a flaw in the proof.

Suppose xRy for any x and y . From the symmetric law, yRx holds. Therefore, since xRy and yRx , we have xRx from the transitive law. Thus, since x is arbitrary, the reflexive law holds.

2.5** Show that, given an undirected graph G with n nodes, if there exists a path from s to t whose length is equal to or more than n , then there exists a path from s to t whose length is equal to or less than $n - 1$. Similarly, show that the same statement holds for a directed graph.

2.6** From De Morgan’s law

$$\overline{F \vee G} = \overline{F} \wedge \overline{G},$$

$$\overline{F \wedge G} = \overline{F} \vee \overline{G},$$

derive the similar law

$$\overline{\overline{F \vee G \vee H}} = \overline{\overline{F}} \wedge \overline{\overline{G}} \wedge \overline{\overline{H}},$$

$$\overline{\overline{F \wedge G \wedge H}} = \overline{\overline{F}} \vee \overline{\overline{G}} \vee \overline{\overline{H}}.$$

Furthermore, generalize the above law to the case where the number of variables is arbitrary.

2.7** In the following, we give the proof of “for any club, the birthplaces of the members of the club are all the same.” Find a flaw in the proof.

Prove by induction on the number of members n of the club.

The base of induction: For $n = 1$, clearly the statement holds.

Induction step: Suppose that the statement holds for $n \geq 1$. We will show that the statement holds when the number of members is $n + 1$. Given a club consisting of $n + 1$ members, make a group consisting of n members, by removing one member in the club. From the hypothesis of induction, the birthplaces of all the members of the group are the same. Similarly, make another group consisting of n members by removing another member from the club. Then the birthplaces of all the members of the second group are also the same by the hypothesis of induction. Therefore, all the members of the club with $n + 1$ members which is the union of the two groups are the same.



<http://www.springer.com/978-0-85729-534-7>

Concise Guide to Computation Theory

Maruoka, A.

2011, XVII, 281 p., Hardcover

ISBN: 978-0-85729-534-7