Discrete Mathematics

PDF generated using the open source mwlib toolkit. See http://code.pediapress.com/ for more information.
PDF generated at: Wed, 12 Jun 2013 14:13:51 UTC

Contents

Articles

Introduction	1
Set theory	1
Functions and relations	12
Logic	22
Graph theory	34
Recursion	38

References

Article Sources and Contributors	42
Image Sources, Licenses and Contributors	43

Article Licenses

License	44

Introduction

Mathematics can help you solve many problems by training you to think well. This book will help you think well about **discrete** problems: problems like chess, in which the moves you make are exact, problems where tools like calculus fail because there's no continuity, problems that appear all the time in games, puzzles, and computer science.

We hope you'll enjoy discovering discrete mathematics here, and we hope you'll find this a good reference for quickly picking up the details you'll forget with time.

../Set theory/ >

Notice to contributors: If you wish to contribute images of any kind that contains labels, please use the Arial font, italic, at 16pts for consistency.

Set theory

Set Theory starts very simply: it examines whether an object *belongs*, or does *not belong*, to a *set* of objects which has been described in some non-ambiguous way. From this simple beginning, an increasingly complex (and useful!) series of ideas can be developed, which lead to notations and techniques with many varied applications.

Definition of a Set

The definition of a set sounds very vague at first. A ***set*** can be defined as a collection of ***things*** that are brought together because they obey a certain ***rule***.

These 'things' may be anything you like: numbers, people, shapes, cities, bits of text ..., literally anything.

The key fact about the 'rule' they all obey is that it must be *well-defined*. In other words, it enables us to say for sure whether or not a given 'thing' belongs to the collection. If the 'things' we're talking about are English words, for example, a well-defined rule might be:

'... has 5 or more letters'

A rule which is not well-defined (and therefore couldn't be used to define a set) might be:

'... is hard to spell'

Elements

A 'thing' that belongs to a given set is called an ***element*** of that set. For example:

Henry VIII is an element of the set of Kings of England

Notation

Curly brackets $\{\ldots\}$ are used to stand for the phrase 'the set of ...'. These braces can be used in various ways. For example:

We may *list* the elements of a set:

$$\{-3, -2, -1, 0, 1, 2, 3\}$$

We may *describe* the elements of a set:

$$\{\text{ integers between } -3 \text{ and } 3 \text{ inclusive}\}$$

We may use an *identifier* (the letter x for example) to represent a *typical element*, a $|$ symbol to stand for the phrase 'such that', and then the rule or rules that the identifier must obey:

$$\{x|x \text{ is an integer and } |x| < 4\}$$

or

$$\{x|x \in \mathbb{Z}, |x| < 4\}$$

The last way of writing a set - called *set comprehension* notation - can be generalized as:

$x|P(x)$, where $P(x)$ is a statement (technically a *propositional function*) about x and the set is the collection of all elements x for which P is true.

The symbol $\in$ is used as follows:

$\in$ means 'is an element of ...'. For example: $\text{dog} \in \{\text{quadrupeds}\}$

$\notin$ means 'is not an element of ...'. For example: $\text{Washington DC} \notin \{\text{European capital cities}\}$

A set can be *finite*: $\{\text{British citizens}\}$

... or *infinite*: $\{7, 14, 21, 28, 35, \ldots\}$

(Note the use of the *ellipsis* $\cdots$ to indicate that the sequence of numbers continues indefinitely.)

Sets will usually be denoted using *upper case* letters: A , B , ...

Elements will usually be denoted using *lower case* letters: x , y , ...

Some Special Sets

Universal Set

The set of all the 'things' currently under discussion is called the ***universal set*** (or sometimes, simply the ***universe***). It is denoted by **U**.

The universal set doesn't contain everything in the whole universe. On the contrary, it restricts us to just those things that are relevant at a particular time. For example, if in a given situation we're talking about numeric values – quantities, sizes, times, weights, or whatever – the universal set will be a suitable set of numbers (see below). In another context, the universal set may be {alphabetic characters} or {all living people}, etc.

Empty set

The set containing no elements at all is called the ***null set***, or ***empty set***. It is denoted by a pair of empty braces: $\{\}$ or by the symbol $\varnothing$.

It may seem odd to define a set that contains no elements. Bear in mind, however, that one may be looking for solutions to a problem where it isn't clear at the outset whether or not such solutions even exist. If it turns out that there isn't a solution, then the set of solutions is empty.

For example:

If $U = \{\text{words in the English language}\}$ then $\{\text{words with more than 50 letters}\} = \varnothing$.

If $U = \{\text{whole numbers}\}$ then $\{x|x^2 = 10\} = \varnothing$.

Operations on the empty set

Operations performed on the empty set (as a set of things to be operated upon) can also be confusing. (Such operations are nullary operations.) For example, the sum of the elements of the empty set is zero, but the product of the elements of the empty set is one (see empty product). This may seem odd, since there are no elements of the empty set, so how could it matter whether they are added or multiplied (since "they" do not exist)? Ultimately, the results of these operations say more about the operation in question than about the empty set. For instance, notice that zero is the identity element for addition, and one is the identity element for multiplication.

Some special sets of numbers

Several sets are used so often, they are given special symbols.

The natural numbers

The 'counting' numbers (or whole numbers) starting at 1, are called the ***natural numbers***. This set is sometimes denoted by **N**. So **N** = {1, 2, 3, ...}

Note that, when we write this set by hand, we can't write in **bold** type so we write an N in blackboard bold font: $\mathbb{N}$

Integers

All whole numbers, positive, negative and zero form the set of ***integers***. It is sometimes denoted by **Z**. So **Z** = {..., -3, -2, -1, 0, 1, 2, 3, ...}

In blackboard bold, it looks like this: $\mathbb{Z}$

Real numbers

If we expand the set of integers to include all decimal numbers, we form the set of ***real numbers***. The set of reals is sometimes denoted by **R**.

A real number may have a *finite* number of digits after the decimal point (e.g. 3.625), or an *infinite* number of decimal digits. In the case of an infinite number of digits, these digits may:

recur; e.g. 8.127127127...

... or they may not recur; e.g. 3.141592653...

In blackboard bold: $\mathbb{R}$

Rational numbers

Those real numbers whose decimal digits are finite in number, or which recur, are called ***rational numbers***. The set of rationals is sometimes denoted by the letter **Q**.

A rational number can always be written as exact fraction *p*/*q*; where *p* and *q* are integers. If *q* equals 1, the fraction is just the integer *p*. Note that *q* may NOT equal zero as the value is then undefined.

For example: 0.5, -17, 2/17, 82.01, 3.282828... are all rational numbers.

In blackboard bold: $\mathbb{Q}$

Irrational numbers

If a number *can't* be represented exactly by a fraction *p*/*q*, it is said to be ***irrational***.

Examples include: √2, √3, π.

Set Theory Exercise 1

Click the link for Set Theory Excercise 1

Relationships between Sets

We’ll now look at various ways in which sets may be related to one another.

Equality

Two sets *A* and *B* are said to be ***equal*** if and only if they have exactly the same elements. In this case, we simply write:

$A = B$

Note two further facts about equal sets:

The *order* in which elements are listed does not matter.

If an element is listed *more than once*, any repeat occurrences are ignored.

So, for example, the following sets are all equal:

$\{1, 2, 3\} = \{3, 2, 1\} = \{1, 1, 2, 3, 2, 2\}$

(You may wonder why one would ever come to write a set like $\{1, 1, 2, 3, 2, 2\}$. You may recall that when we defined the *empty set* we noted that there may be no solutions to a particular problem - hence the need for an empty set. Well, here we may be trying several different approaches to solving a problem, some of which in fact lead us to the same solution. When we come to consider the *distinct* solutions, however, any such repetitions would be ignored.)

Subsets

If all the elements of a set A are also elements of a set B, then we say that A is a ***subset*** of B, and we write:

$A \subseteq B$

For example:

If $T = \{2, 4, 6, 8, 10\}$ and $E = \{\text{even integers}\}$, then $T \subseteq E$

If $A = \{\text{alphanumeric characters}\}$ and $P = \{\text{printable characters}\}$, then $A \subseteq P$

If $Q = \{\text{quadrilaterals}\}$ and $F = \{\text{plane figures bounded by four straight lines}\}$, then $Q \subseteq F$

Notice that $A \subseteq B$ does not imply that B must necessarily contain extra elements that are not in A; the two sets could be equal – as indeed Q and F are above. However, if, in addition, B does contain at least one element that isn't in A, then we say that A is a ***proper subset*** of B. In such a case we would write:

$A \subset B$

In the examples above:

E contains 12, 14, ... , so $T \subset E$

P contains $, ;, &, ..., so $A \subset P$

But Q and F are just different ways of saying the same thing, so $Q = F$

The use of $\subset$ and $\subseteq$ is clearly analogous to the use of $<$ and $\leq$ when comparing two numbers.

Notice also that *every* set is a subset of the *universal set*, and the *empty set* is a subset of *every* set.

(You might be curious about this last statement: how can the empty set be a subset of *anything*, when it doesn't contain any elements? The point here is that for every set A, the empty set **doesn't** contain any elements that **aren't** in A. So $\emptyset \subseteq A$ for all sets A.)

Finally, note that if $A \subseteq B$ and $B \subseteq A$ then A and B must contain exactly the same elements, and are therefore equal. In other words:

If $A \subseteq B$ and $B \subseteq A$ then $A = B$

Disjoint

Two sets are said to be ***disjoint*** if they have no elements in common. For example:

If A = {even numbers} and B = {1, 3, 5, 11, 19}, then A and B are disjoint.

Venn Diagrams

A *Venn diagram* can be a useful way of illustrating relationships between sets.

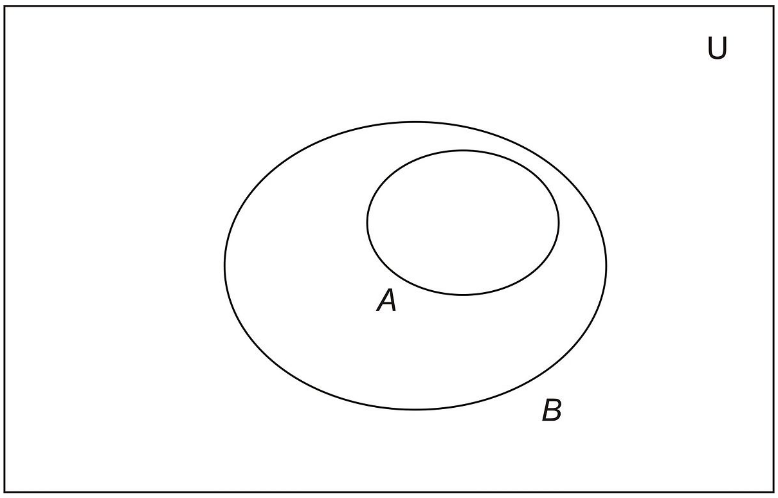
In a Venn diagram:

The *universal set* is represented by a *rectangle*. Points inside the rectangle represent elements that are in the universal set; points outside represent things not in the universal set. You can think of this rectangle, then, as a 'fence' keeping unwanted things out - and concentrating our attention on the things we're talking about.

Other sets are represented by *loops*, usually oval or circular in shape, drawn inside the rectangle. Again, points inside a given loop represent elements in the set it represents; points outside represent things *not* in the set.

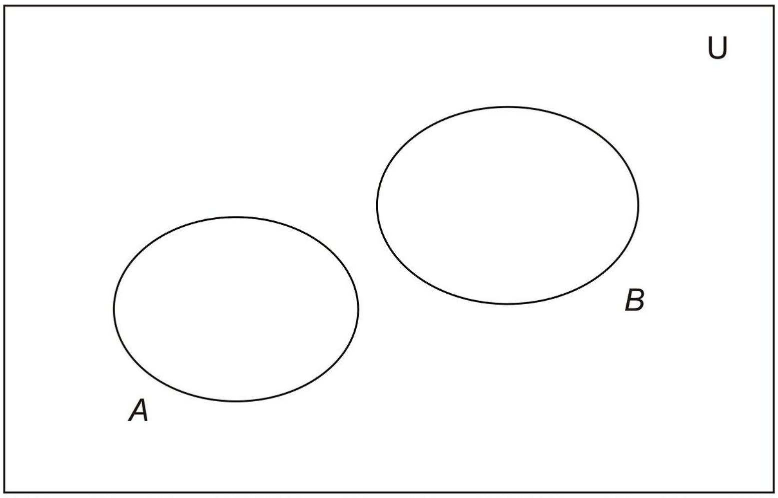
On the left, the sets A and B are disjoint, because the loops don't overlap.

On the right A is a subset of B, because the loop representing set A is entirely enclosed by loop B.



$A \subset B$

Venn diagrams: Fig. 2



A and B are *disjoint*

Venn diagrams: Fig. 1

Venn diagrams: Worked Examples

Example 1

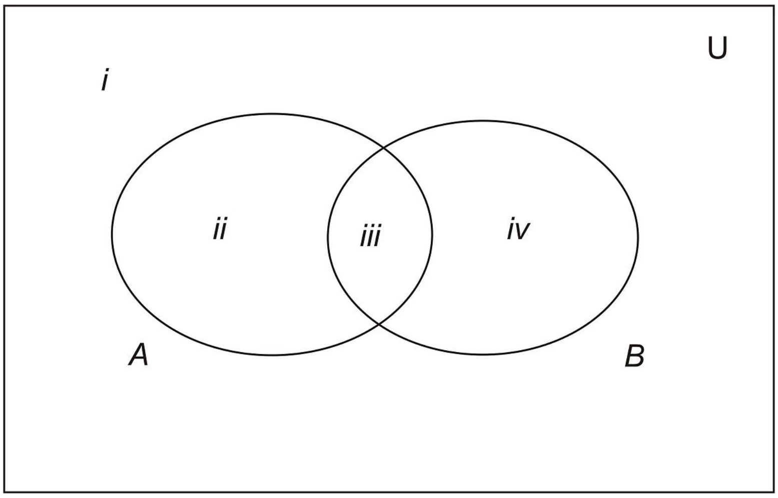
Fig. 3 represents a Venn diagram showing two sets *A* and *B*, in the general case where nothing is known about any relationships between the sets.

Note that the rectangle representing the universal set is divided into four regions, labelled *i*, *ii*, *iii* and *iv*.

What can be said about the sets *A* and *B* if it turns out that:

(a) region *ii* is empty?

(b) region *iii* is empty?



Venn diagrams: Fig. 3

(a) If region *ii* is empty, then *A* contains no elements that are not in *B*. So *A* is a subset of *B*, and the diagram should be re-drawn like *Fig 2* above.

(b) If region *iii* is empty, then *A* and *B* have no elements in common and are therefore disjoint. The diagram should then be re-drawn like *Fig 1* above.

Example 2

(a) Draw a Venn diagram to represent three sets *A*, *B* and *C*, in the general case where nothing is known about possible relationships between the sets.

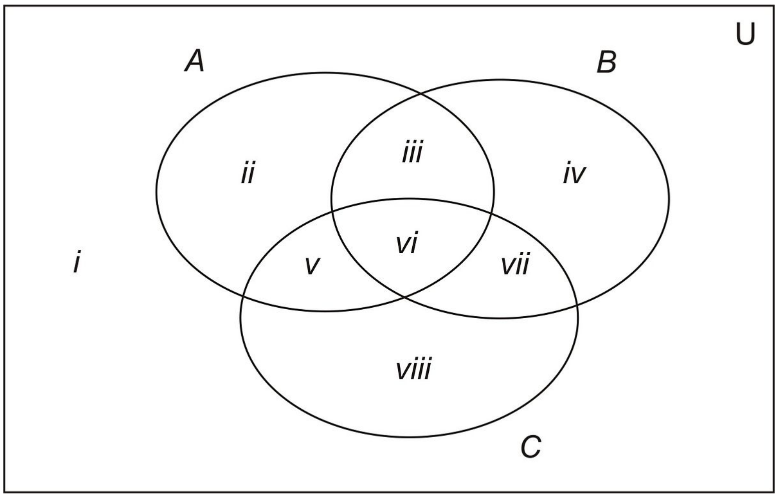
(b) Into how many regions is the rectangle representing **U** divided now?

(c) Discuss the relationships between the sets *A*, *B* and *C*, when various combinations of these regions are empty.

(a) The diagram in *Fig. 4* shows the general case of three sets where nothing is known about any possible relationships between them.

(b) The rectangle representing **U** is now divided into 8 regions, indicated by the Roman numerals *i* to *viii*.

(c) Various combinations of empty regions are possible. In each case, the Venn diagram can be re-drawn so that empty regions are no longer included. For example:



Venn diagrams: Fig. 4

If region *ii* is empty, the loop representing *A* should be made smaller, and moved inside *B* and *C* to eliminate region *ii*.

If regions *ii*, *iii* and *iv* are empty, make *A* and *B* smaller, and move them so that they are both inside *C* (thus eliminating all three of these regions), but do so in such a way that they still overlap each other (thus retaining region *vi*).

If regions *iii* and *vi* are empty, 'pull apart' loops *A* and *B* to eliminate these regions, but keep each loop overlapping loop *C*.

...and so on. Drawing Venn diagrams for each of the above examples is left as an exercise for the reader.

Example 3

The following sets are defined:

U = {1, 2, 3, …, 10}

A = {2, 3, 7, 8, 9}

B = {2, 8}

C = {4, 6, 7, 10}

Using the two-stage technique described below, draw a Venn diagram to represent these sets, marking all the elements in the appropriate regions.

The technique is as follows:

Draw a 'general' 3-set Venn diagram, like the one in *Example 2*.

Go through the elements of the universal set one at a time, once only, entering each one into the appropriate region of the diagram.

Re-draw the diagram, if necessary, moving loops inside one another or apart to eliminate any empty regions.

Don't begin by entering the elements of set *A*, then set *B*, then *C* – you'll risk missing elements out or including them twice!

Solution

After drawing the three empty loops in a diagram looking like *Fig. 4* (but without the Roman numerals!), go through each of the ten elements in **U** - the numbers 1 to 10 - asking each one three questions; like this:

First element: 1

Are you in *A*? No

Are you in *B*? No

Are you in *C*? No

A 'no' to all three questions means that the number 1 is outside all three loops. So write it in the appropriate region (region number *i* in *Fig. 4*).

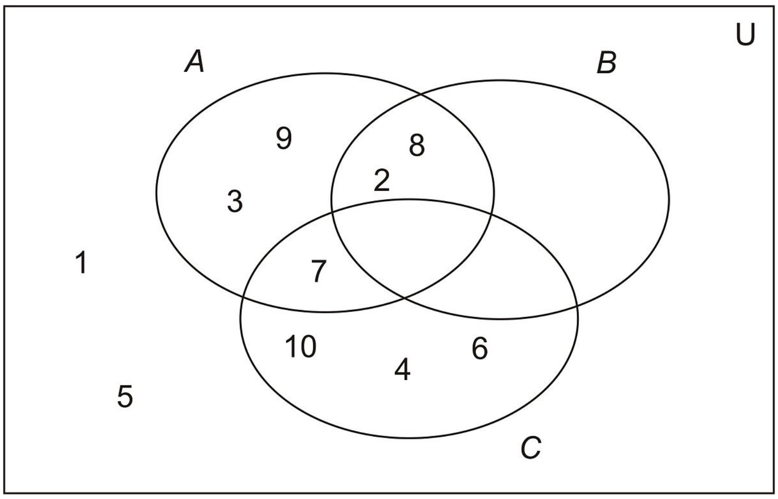
Second element: 2

Are you in *A*? Yes

Are you in *B*? Yes

Are you in *C*? No

Yes, yes, no: so the number 2 is inside *A* and *B* but outside *C*. Goes in region *iii* then.



Venn diagrams: Fig. 5

...and so on, with elements 3 to 10.

The resulting diagram looks like *Fig. 5*.

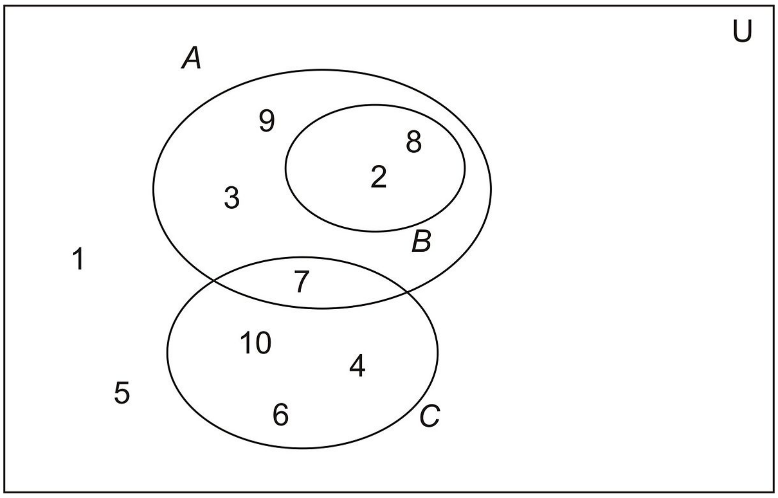
The final stage is to examine the diagram for empty regions - in this case the regions we called *iv*, *vi* and *vii* in *Fig. 4* - and then re-draw the diagram to eliminate these regions. When we've done so, we shall clearly see the relationships between the three sets.

So we need to:

pull *B* and *C* apart, since they don't have any elements in common.

push *B* inside *A* since it doesn't have any elements outside *A*.

The finished result is shown in *Fig. 6*.



Venn diagrams: Fig. 6

The regions in a Venn Diagram and Truth Tables

Perhaps you've realized that adding an additional set to a Venn diagram *doubles* the number of regions into which the rectangle representing the universal set is divided. This gives us a very simple pattern, as follows:

With one set loop, there will be just two regions: the inside of the loop and its outside.

With two set loops, there'll be four regions.

With three loops, there'll be eight regions.

...and so on.

It's not hard to see why this should be so. Each new loop we add to the diagram divides each existing region into two, thus doubling the number of regions altogether.

In *A*?	**In *B*?**	**In *C*?**
Y	Y	Y
Y	Y	N
Y	N	Y
Y	N	N
N	Y	Y
N	Y	N
N	N	Y
N	N	N

But there's another way of looking at this, and it's this. In the solution to *Example 3* above, we asked three questions of each element: *Are you in A? Are you in B?* and *Are you in C?* Now there are obviously two possible answers to each of these questions: *yes* and *no*. When we *combine* the answers to three questions like this, one after the other, there are then $2^3 = 8$ possible sets of answers altogether. Each of these eight possible combinations of answers corresponds to a different region on the Venn diagram.

The complete set of answers resembles very closely a *Truth Table* - an important concept in *Logic*, which deals with statements which may be *true* or *false*. The table on the right shows the eight possible combinations of answers for 3 sets *A*, *B* and *C*.

You'll find it helpful to study the patterns of Y's and N's in each column.

As you read down column *C*, the letter changes on every row: Y, N, Y, N, Y, N, Y, N

Reading down column *B*, the letters change on every other row: Y, Y, N, N, Y, Y, N, N

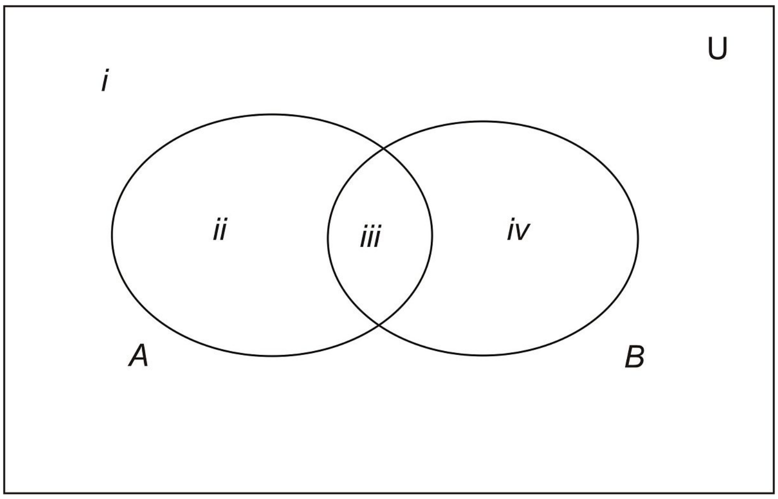
Reading down column *A*, the letters change every four rows: Y, Y, Y, Y, N, N, N, N

Set Theory Exercise 2

Click link for Set Theory Exercise 2.

Operations on Sets

Just as we can combine two numbers to form a third number, with operations like 'add', 'subtract', 'multiply' and 'divide', so we can combine two sets to form a third set in various ways. We'll begin by looking again at the Venn diagram which shows two sets *A* and *B* in a general position, where we don't have any information about how they may be related.



Venn diagrams: Fig. 7

In *A*?	In *B*?	Region
Y	Y	*iii*
Y	N	*ii*
N	Y	*iv*
N	N	*i*

The first two columns in the table on the right show the four sets of possible answers to the questions *Are you in A?* and *Are you in B?* for two sets *A* and *B*; the Roman numerals in the third column show the corresponding region in the Venn diagram in *Fig. 7*.

Intersection

Region *iii*, where the two loops overlap (the region corresponding to 'Y' followed by 'Y'), is called the *intersection* of the sets A and B. It is denoted by $A \cap B$. So we can define intersection as follows:

> The ***intersection*** of two sets A and B, written $A \cap B$, is the set of elements that are in A **and** in B.

(Note that in symbolic logic, a similar symbol, $\wedge$, is used to connect two logical propositions with the **AND** operator.)

For example, if $A = \{1, 2, 3, 4\}$ and $B = \{2, 4, 6, 8\}$, then $A \cap B = \{2, 4\}$.

We can say, then, that we have combined two sets to form a third set using the *operation of intersection*.

Union

In a similar way we can define the *union* of two sets as follows:

> The ***union*** of two sets A and B, written $A \cup B$, is the set of elements that are in A **or** in B (or both).

The union, then, is represented by regions *ii*, *iii* and *iv* in *Fig. 7*.

(Again, in logic a similar symbol, $\vee$, is used to connect two propositions with the **OR** operator.)

> So, for example, $\{1, 2, 3, 4\} \cup \{2, 4, 6, 8\} = \{1, 2, 3, 4, 6, 8\}$.

You'll see, then, that in order to get into the intersection, an element must answer 'Yes' to *both* questions, whereas to get into the union, *either* answer may be 'Yes'.

The $\cup$ symbol looks like the first letter of 'Union' and like a cup that will hold a lot of items. The $\cap$ symbol looks like a spilled cup that won't hold a lot of items, or possibly the letter 'n', for i'n'tersection. Take care not to confuse the two.

Difference

> The ***difference*** of two sets A and B (also known as the ***set-theoretic difference*** of A and B, or the ***relative complement*** of B in A) is the set of elements that are **in** A **but not in** B.

This is written $A - B$, or sometimes $A \setminus \mathrm{B}$.

The elements in the difference, then, are the ones that answer 'Yes' to the first question *Are you in A?*, but 'No' to the second *Are you in B?*. This combination of answers is on row 2 of the above table, and corresponds to region *ii* in *Fig.7*.

> For example, if $A = \{1, 2, 3, 4\}$ and $B = \{2, 4, 6, 8\}$, then $A - B = \{1, 3\}$.

Complement

So far, we have considered operations in which *two* sets combine to form a third: *binary* operations. Now we look at a *unary* operation - one that involves just *one* set.

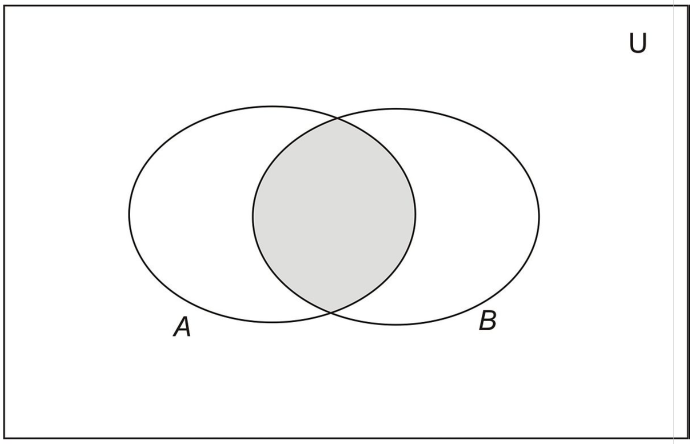
> The set of elements that are **not** in a set A is called the ***complement*** of A. It is written A' (or sometimes A^{C}, or $\bar{A}$).

Clearly, this is the set of elements that answer 'No' to the question *Are you in A?*.

> For example, if $\mathbf{U} = \mathbf{N}$ and $A = \{\text{odd numbers}\}$, then $A' = \{\text{even numbers}\}$.

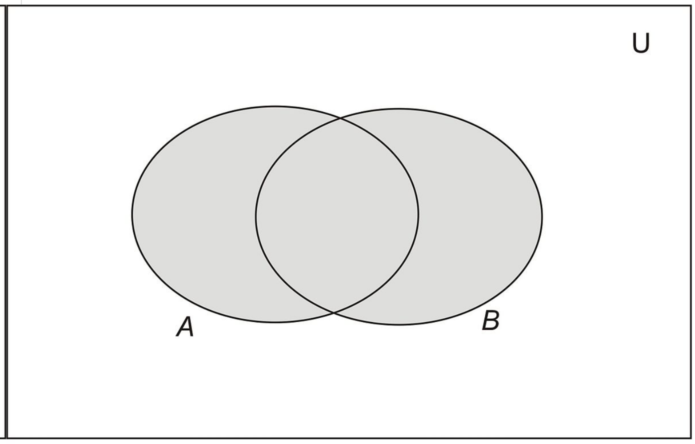
Notice the spelling of the word *complement*: its literal meaning is 'a complementary item or items'; in other words, 'that which completes'. So if we already have the elements of A, the complement of A is the set that *completes* the universal set.

Summary



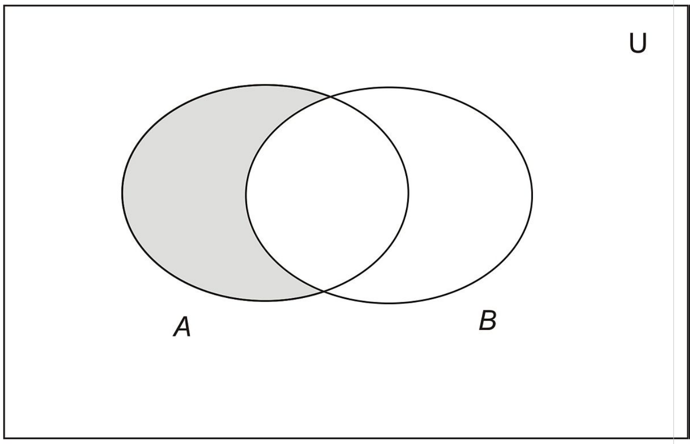
$A \cap B$

Intersection: things that are in *A* **and** in *B*



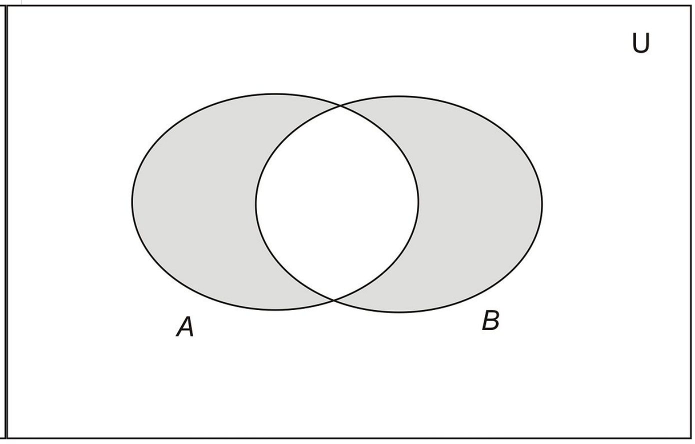
$A \cup B$

Union: things that are in *A* **or** in *B* (or both)



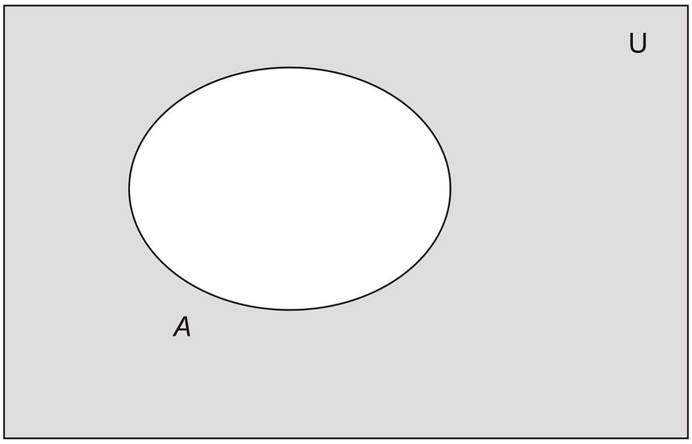
$A - B$

Difference: things that are in *A* **and not** in *B*



$A \Delta B$

Symmetric Difference: things that are in *A* **or** in *B* **but not** both



A'

Complement: things that are **not** in *A*

Cardinality

Finally, in this section on *Set Operations* we look at an operation on a set that yields not another set, but an integer.

The ***cardinality*** of a finite set A, written $|A|$ (sometimes $\#(A)$ or $n(A)$), is the number of (distinct) elements in A. So, for example:

If A = {lower case letters of the alphabet}, $|A| = 26$.

Generalized set operations

If we want to denote the intersection or union of n sets, $A_1, A_2, ..., A_n$ (where we may not know the value of n) then the following *generalized set notation* may be useful:

$A_1 \cap A_2 \cap ... \cap A_n = \bigcap_{i=1}^{n} A_i$
$A_1 \cup A_2 \cup ... \cup A_n = \bigcup_{i=1}^{n} A_i$

In the symbol $\bigcap_{i=1}^{n} A_i$, then, i is a variable that takes values from 1 to n, to indicate the repeated intersection of all the sets A_1 to A_n.

Set Theory Exercise 3

Click link for Set Theory Exercise 3

Set Theory Page 2

Set Theory continues on Page 2.

Previous topic: ../Introduction/ | Contents:Discrete Mathematics | Next topic: ../Functions and relations/

Functions and relations

This article examines the concepts of the *function* and the *relation*.

A ***relation*** is any association between elements of one set, called the ***domain*** or (less formally) the *set of inputs*, and another set, called the ***range*** or *set of outputs*. Some people mistakenly refer to the range as the *codomain*, but as we will see, that really means the *set of all possible outputs*—even values that the relation does not actually use.

For example, if the *domain* is a set Fruits = {apples, oranges, bananas} and the *codomain* is a set Flavors = {sweetness, tartness, bitterness}, the flavors of these fruits form a relation: we might say that apples are related to (or associated with) **both** sweetness and tartness, while oranges are related to tartness only and bananas to sweetness only. (We might disagree somewhat, but that is irrelevant to the topic of this book.) Notice that "bitterness", although it is one of the possible Flavors (codomain), is not really used for any of these relationships; so it is not part of the *range* {sweetness, tartness}.

Another way of looking at this is to say that a relation is a *subset of ordered pairs* drawn from the *set of all possible ordered pairs* (of elements of two other sets, which we normally refer to as the *Cartesian product* of those sets). Formally, R is a relation if

$R \subseteq \{(x, y) \mid x \in X, y \in Y\}$

for the domain X and codomain Y.

Using the example above, we can write the relation in set notation: {(apples, sweetness), (apples, tartness), (oranges, tartness), (bananas, sweetness)}.

One important kind of relation is the *function*. A ***function*** is a relation that has **exactly one output** for every possible input **in the domain**. (Unlike the codomain, the domain does not necessarily have to include all possible objects of a given type. In fact, we sometimes intentionally use a *restricted domain* in order to satisfy some desirable property.) For example, the relation that we discussed above (flavors of fruits) is **not** a function, because it has two possible outputs for the input "apples": sweetness and tartness.

The main reason for not allowing multiple outputs with the same input is that it lets us apply the same function to different forms of the same thing without changing their equivalence. That is, if x = y, and f is a function with x (or y) in its domain, then f(x) = f(y). For example, z - 3 = 5 implies that z = 8 because f(x) = x + 3 is a function defined for all numbers x.

The converse, that f(x) = f(y) implies x = y, is not always true. When it is, f is called a **one-to-one** or **invertible function**.

Relations

In the above section dealing with functions and their properties, we noted the important property that all functions must have, namely that if a function does map a value from its domain to its co-domain, it must map this value to only one value in the co-domain.

Writing in set notation, if *a* is some fixed value:

$$|\{f(x)|x=a\}| \in \{0, 1\}$$

The literal reading of this statement is: the *cardinality* (number of elements) of the set of all values f(x), such that x=a for some fixed value a, is an element of the set {0, 1}. In other words, the number of *outputs* that a function f may have at any fixed *input* a is either zero (in which case it is *undefined* at that input) or one (in which case the output is unique).

However, when we consider the *relation*, we relax this constriction, and so a relation may map one value to more than one other value. In general, a relation is **any** subset of the Cartesian product of its domain and co-domain.

All functions, then, can be considered as relations also.

Notations

When we have the property that one value is related to another, we call this relation a *binary relation* and we write it as

$$x \; R \; y$$

where R is the relation.

For arrow diagrams and set notations, remember for relations we do not have the restriction that functions do and we can draw an arrow to represent the mappings, and for a set diagram, we need only write all the ordered pairs that the relation does take: again, by example

$$f = \{(0,0),(1,1),(1,-1),(2,2),(2,-2)\}$$

is a relation and not a function, since both 1 and 2 are mapped to two values, 1 and -1, and 2 and -2 respectively) example let A=2,3,5;B=4,6,9 then A*B=(2,4),(2,6),(2,9),(3,4),(3,6),(3,9),(5,4),(5,6),(5,9) Define a relation R=(2,4),(2,6),(3,6),(3,9) add functions and problems to one another

Some simple examples

Let us examine some simple relations.

Say f is defined by

{(0,0),(1,1),(2,2),(3,3),(1,2),(2,3),(3,1),(2,1),(3,2),(1,3)}

This is a relation (not a function) since we can observe that 1 maps to 2 and 3, for instance.

Less-than, "<", is a relation also. Many numbers can be less than some other fixed number, so it cannot be a function.

Properties

When we are looking at relations, we can observe some special properties different relations can have.

Reflexive

A relation is *reflexive* if, we observe that for all values a:

$a \mathrm{R} a$

In other words, all values are related to themselves.

The relation of equality, "=" is reflexive. Observe that for, say, all numbers a (the domain is **R**):

$a = a$

so "=" is reflexive.

In a reflexive relation, we have arrows for all values in the domain pointing back to themselves:

Note that ≤ is also reflexive (a ≤ a for any a in **R**). On the other hand, the relation < is not (a < a is false for any a in **R**).

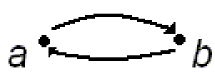
Symmetric

A relation is *symmetric* if, we observe that for all values a and b:

$a \mathrm{R} b$ implies $b \mathrm{R} a$

The relation of equality again is symmetric. If $x=y$, we can also write that $y=x$ also.

In a symmetric relation, for each arrow we have also an opposite arrow, i.e. there is either no arrow between x and y, or an arrow points from x to y and an arrow back from y to x:



Neither ≤ nor < is symmetric (2 ≤ 3 and 2 < 3 but not 3 ≤ 2 nor 3 < 2 is true).

Transitive

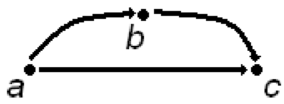
A relation is *transitive* if for all values *a*, *b*, *c*:

a R *b* and *b* R *c* implies *a* R *c*

The relation *greater-than* ">" is transitive. If $x > y$, and $y > z$, then it is true that $x > z$. This becomes clearer when we write down what is happening into words. *x* is greater than *y* and *y* is greater than *z*. So *x* is greater than both *y* and *z*.

The relation *is-not-equal* "≠" is not transitive. If $x \neq y$ and $y \neq z$ then we might have $x = z$ or $x \neq z$ (for example $1 \neq 2$ and $2 \neq 3$ and $1 \neq 3$ but $0 \neq 1$ and $1 \neq 0$ and $0 = 0$).

In the arrow diagram, every arrow between two values *a* and *b*, and *b* and *c*, has an arrow going straight from *a* to *c*.



Antisymmetric

A relation is *antisymmetric* if we observe that for all values *a* and *b*:

a R *b* and *b* R *a* implies that $a=b$

Notice that antisymmetric is not the same as "not symmetric."

Take the relation *greater than or equal to*, "≥" If $x \geq y$, and $y \geq$ x, then *y* must be equal to *x*. a relation is anti-symmetric if and only if a∈A, (a,a)∈R

Trichotomy

A relation satisfies *trichotomy* if we observe that for all values *a* and *b* it holds true that: *a*R*b or b*R*a*

The relation *is-greater-or-equal* satisfies since, given 2 real numbers *a* and *b*, it is true that whether $a \geq b$ or $b \geq a$ (both if $a = b$).

Problem set

Given the above information, determine which relations are reflexive, transitive, symmetric, or antisymmetric on the following - there may be more than one characteristic. (Answers follow.) *x* R *y* if

1. $x = y$
2. $x < y$
3. $x^2 = y^2$
4. $x \leq y$

Answers

1. Symmetric, Reflexive, Transitive and Antisymmetric
2. Transitive, Antisymmetric
3. Symmetric, Reflexive, Transitive and Antisymmetric ($x^2 = y^2$ is just a special case of equality, so all properties that apply to $x = y$ also apply to this case)
4. Reflexive, Transitive and Antisymmetric (and satisfying Trichotomy)

Equivalence relations

We have seen that certain common relations such as "=", and congruence (which we will deal with in the next section) obey some of these rules above. The relations we will deal with are very important in discrete mathematics, and are known as *equivalence relations*. They essentially assert some kind of equality notion, or *equivalence*, hence the name.

Characteristics of equivalence relations

For a relation R to be an *equivalence relation*, it must have the following properties, viz. R must be:

- symmetric
- transitive
- reflexive

(A helpful mnemonic, S-T-R)

In the previous problem set you have shown equality, "=", to be reflexive, symmetric, and transitive. So "=" is an equivalence relation.

We denote an equivalence relation, in general, by $x \sim y$.

Example proof

Say we are asked to prove that "=" is an equivalence relation. We then proceed to prove each property above in turn (Often, the proof of transitivity is the hardest).

- **Reflexive**: Clearly, it is true that $a = a$ for all values a. Therefore, = is reflexive.
- **Symmetric**: If $a = b$, it is also true that $b = a$. Therefore, = is symmetric
- **Transitive**: If $a = b$ and $b = c$, this says that a is the same as b which in turn is the same as c. So a is then the same as c, so $a = c$, and thus = is transitive.

Thus = is an equivalence relation.

Partitions and equivalence classes

It is true that when we are dealing with relations, we may find that many values are related to one fixed value.

For example, when we look at the quality of *congruence*, which is that given some number a, a number congruent to a is one that has the same remainder or *modulus* when divided by some number n, as a, which we write

$a \equiv b \pmod{n}$

and is the same as writing

$b = a+kn$ for some integer k.

(We will look into congruences in further detail later, but a simple examination or understanding of this idea will be interesting in its application to equivalence relations)

For example, $2 \equiv 0 \pmod{2}$, since the remainder on dividing 2 by 2 is in fact 0, as is the remainder on dividing 0 by 2.

We can show that congruence is an equivalence relation (This is left as an exercise, below **Hint** use the equivalent form of congruence as described above).

However, what is more interesting is that we can group all numbers that are equivalent to each other.

With the relation congruence *modulo* 2 (which is using n=2, as above), or more formally:

$x \sim y$ if and only if $x \equiv y \pmod{2}$

we can group all numbers that are equivalent to each other. Observe:

$$0 \equiv 2 \equiv 4 \equiv \ldots \pmod{2}$$

$$1 \equiv 3 \equiv 5 \equiv \dots \pmod 2$$

This first equation above tells us all the *even* numbers are equivalent to each other under ~, and all the *odd* numbers under ~.

We can write this in set notation. However, we have a special notation. We write:

[0]={0,2,4,...}

[1]={1,3,5,...}

and we call these two sets *equivalence classes*.

All elements in an equivalence class by definition are equivalent to each other, and thus note that we do not need to include [2], since 2 ~ 0.

We call the act of doing this 'grouping' with respect to some equivalence relation *partitioning* (or further and explicitly *partitioning a set S into equivalence classes under a relation* ~). Above, we have partitioned $\mathbf{Z}$ into equivalence classes [0] and [1], under the relation of congruence modulo 2.

Problem set

Given the above, answer the following questions on equivalence relations (Answers follow to even numbered questions)

1. Prove that congruence is an equivalence relation as before (See hint above).
2. Partition {x | 1 ≤ x ≤ 9} into equivalence classes under the equivalence relation

$$x \sim y \text{ iff } x \equiv y \pmod 6$$

Answers

2. [0]={6}, [1]={1,7}, [2]={2,8}, [3]={3,9}, [4]={4}, [5]={5}

Partial orders

We also see that "≥" and "≤" obey some of the rules above. Are these special kinds of relations too, like equivalence relations? Yes, in fact, these relations are specific examples of another special kind of relation which we will describe in this section: the *partial order*.

As the name suggests, this relation gives some kind of ordering to numbers.

Characteristics of partial orders

For a relation R to be a partial order, it must have the following three properties, viz R must be:

- reflexive
- antisymmetric
- transitive

(A helpful mnemonic, R-A-T)

We denote a partial order, in general, by $x \preceq y$.

Example proof

Say we are asked to prove that "≤" is a partial order. We then proceed to prove each property above in turn (Often, the proof of transitivity is the hardest).

Reflexive

Clearly, it is true that $a \leq a$ for all values a. So ≤ is reflexive.

Antisymmetric

If $a \leq b$, and $b \leq a$, then a *must* be equal to b. So ≤ is antisymmetric

Transitive

If $a \leq b$ and $b \leq c$, this says that a is less than b and c. So a is less than c, so $a \leq c$, and thus ≤ is transitive.

Thus ≤ is a partial order.

Problem set

Given the above on partial orders, answer the following questions

1. Prove that divisibility, |, is a partial order (a | b means that a is a factor of b, i.e., on dividing b by a, no remainder results).
2. Prove the following set is a partial order: $(a, b) \preceq (c, d)$ implies $ab \leq cd$ for a,b,c,d integers ranging from 0 to 5.

Answers

2. Simple proof; Formalization of the proof is an optional exercise.

  Reflexivity: $(a, b) \preceq (a, b)$ since $ab=ab$.

  Antisymmetric: $(a, b) \preceq (c, d)$ and $(c, d) \preceq (a, b)$ since $ab \leq cd$ and $cd \leq ab$ imply $ab=cd$.

  Transitive: $(a, b) \preceq (c, d)$ and $(c, d) \preceq (e, f)$ implies $(a, b) \preceq (e, f)$ since $ab \leq cd \leq ef$ and thus $ab \leq ef$

Posets

A partial order imparts some kind of "ordering" amongst elements of a set. For example, we only know that 2 ≥ 1 because of the partial ordering ≥.

We call a set A, ordered under a general partial ordering $\preceq$, a *partially ordered set*, or simply just *poset*, and write it (A, $\preceq$).

Terminology

There is some specific terminology that will help us understand and visualize the partial orders.

When we have a partial order $\preceq$, such that $a \preceq b$, we write $\prec$ to say that a $\preceq$ but $a \neq b$. We say in this instance that a *precedes* b, or a is a predecessor of b.

If (A, $\preceq$) is a poset, we say that a is an immediate predecessor of b (or a immediately precedes b) if there is no x in A such that $a \prec x \prec b$.

If we have the same poset, and we also have a and b in A, then we say a and b are *comparable* if $a \preceq b$ or $b \preceq a$. Otherwise they are *incomparable*.

Hasse diagrams

Hasse diagrams are special diagrams that enable us to visualize the structure of a partial ordering. They use some of the concepts in the previous section to draw the diagram.

A Hasse diagram of the poset (A, $\preceq$) is constructed by

- placing elements of A as points
- if *a* and *b* ∈ A, and *a* is an immediate predecessor of b, we draw a line from *a* to *b*
- if $a \prec b$, put the point for *a* lower than the point for *b*
- not drawing loops from *a* to *a* (this is assumed in a partial order because of reflexivity)

Operations on Relations

There are some useful operations one can perform on relations, which allow to express some of the above mentioned properties more briefly.

Inversion

Let R be a relation, then its inversion, R^{-1} is defined by

R^{-1} := {(a,b) | (b,a) in R}.

Concatenation

Let R be a relation between the sets A and B, S be a relation between B and C. We can concatenate these relations by defining

R • S := {(a,c) | (a,b) in R and (b,c) in S for some b out of B}

Diagonal of a Set

Let A be a set, then we define the diagonal (D) of A by

D(A) := {(a,a) | a in A}

Shorter Notations

Using above definitions, one can say (lets assume R is a relation between A and B):

R is *transitive* if and only if R • R is a subset of R.

R is *reflexive* if and only if D(A) is a subset of R.

R is *symmetric* if R^{-1} is a subset of R.

R is *antisymmetric* if and only if the intersection of R and R^{-1} is D(A).

R is *asymmetric* if and only if the intersection of D(A) and R is empty.

R is a *function* if and only if R^{-1} • R is a subset of D(B).

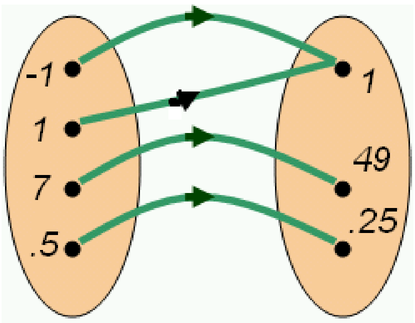
In this case it is a function A → B. Let's assume R meets the condition of being a function, then

R is *injective* if R • R^{-1} if a subset of D(A).

R is *surjective* if {b | (a,b) in R} = B.

Functions

A function is a relationship between two sets of numbers. We may think of this as a *mapping*; a function *maps* a number in one set to a number in another set. Notice that a function maps values to **one and only one** value. Two values in one set could map to one value, but one value **must never** map to two values: that would be a relation, *not* a function.



For example, if we write (define) a function as:

$$f(x) = x^2$$

then we say:

'f of x equals x squared'

and we have

$$f(-1) = 1$$
$$f(1) = 1$$
$$f(7) = 49$$
$$f(1/2) = 1/4$$
$$f(4) = 16$$

and so on.

This function f maps numbers to their squares.

Range, image, codomain

If D is a set, we can say

$$f(D) = \{f(x) |\ x \in D\}$$

which forms a new set, called the *range* of f. D is called the *domain* of f, and represents all values that f takes.

In general, the range of f is usually a subset of a larger set. This set is known as the *codomain* of a function. For example, with the function f(x)=cos x, the range of f is [-1,1], **but** the codomain is the set of real numbers.

Notations

When we have a function f, with domain D and range R, we write:

$$f : D \longrightarrow R$$

If we say that, for instance, x is mapped to x^2, we also can add

$$f : D \longrightarrow R;\ x \longmapsto x^2$$

Notice that we can have a function that maps a point (x,y) to a real number, or some other function of two variables -- we have a set of ordered pairs as the domain. Recall from set theory that this is defined by the *Cartesian product* - if we wish to represent a set of all real-valued ordered pairs we can take the Cartesian product of the real numbers with itself to obtain

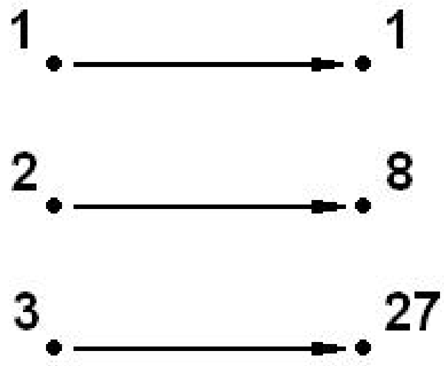
$$\mathbb{R} \times \mathbb{R} = \mathbb{R}^2 = \{(x, y)|\ x \text{ and } y \in \mathbb{R}\}\,.$$

When we have a set of *n*-tuples as part of the domain, we say that the function is *n*-ary (for numbers *n*=1,2 we say unary, and binary respectively).

Other function notation

Functions can be written as above, but we can also write them in two other ways. One way is to use an arrow diagram to represent the mappings between each element. We write the elements from the domain on one side, and the elements from the range on the other, and we draw arrows to show that an element from the domain is mapped to the range.

For example, for the function f(x)=x^3, the arrow diagram for the domain {1,2,3} would be:



Another way is to use set notation. If f(x)=y, we can write the function in terms of its mappings. This idea is best to show in an example.

Let us take the domain D={1,2,3}, and f(x)=x^2. Then, the range of f will be R={f(1),f(2),f(3)}={1,4,9}. Taking the Cartesian product of D and R we obtain F={(1,1),(2,4),(3,9)}.

So using set notation, a function can be expressed as the Cartesian product of its domain and range.

```
f(x)
```

This function is called *f*, and it takes a *variable x*. We substitute some value for *x* to get the second value, which is what the function maps x to.

Previous topic: ../Functions and relations/ | Contents:Discrete Mathematics | Next topic: ../Number theory/

Logic

Overheard on the bus…

"I’ve heard it said that wearing a hat leads to baldness."

"Oh really? I’ve heard that bald men are generally good tempered."

"In that case, I’m glad to see that Jones has started wearing a hat. His temper has been rather short lately!"

In conventional algebra, we use letters and symbols to represent numbers and the operations associated with them: +, -, ×, ÷, etc. By so doing we can simplify and solve complex problems. In *Logic*, we seek to express statements, and the connections between them in algebraic symbols - again with the object of simplifying complicated ideas. Unfortunately, like ordinary algebra, the opposite seems true initially. This is probably because simple examples always seem easier to solve by common-sense methods!

Propositions

A ***proposition*** is a statement which has *truth value*: it is either *true* (T) or *false* (F).

Example 1

Which of the following are propositions?

(a) 17 + 25 = 42

(b) July 4 occurs in the winter in the Northern Hemisphere.

(c) The population of the United States is less than 250 million.

(d) Is the moon round?

(e) 7 is greater than 12.

(f) x is greater than y.

Answers

(a) is a proposition; and of course it has the 'truth value' *true*.

(b) is a proposition. Of course, it's *false*, but it's still a proposition.

(c) is a proposition, but we may not actually *know* whether it's *true* or *false*. Nevertheless, the fact is that the statement itself *is* a proposition, because it is definitely either *true* or *false*.

(d) is not a proposition. It's a question.

(e) is a proposition. It's *false* again, of course.

(f) is a bit more debatable! It's certainly a *potential* proposition, but until we know the values of x and y, we can't actually say whether it is *true* or *false*. Note that this isn't quite the same as (c), where we may not know the truth value because we aren't well-enough informed. See the next paragraph.

Propositional Functions

A *function* is, loosely defined, an operation that takes as input one or more *parameter values*, and produces a single, well-defined output.

You're probably familiar with the sine and cosine functions in trigonometry, for example. These are examples of functions that take a single number (the size of an angle) as an input and produce a decimal number (which in fact will lie between +1 and -1) as output.

If we want to, we can define a function of our own, say *RectangleArea*, which could take *two* numbers (the length and width of a rectangle) as input, and produce a single number as output (formed by multiplying the two input numbers together).

In (f) above, we have an example of a ***Propositional Function***. This is a function that produces as output not a number like sine, cosine or *RectangleArea*, but a *truth value*. It's a statement, then, that becomes a proposition when it is supplied with one or more parameter values. In (f), the parameters are x and y. So if $x = 2$ and $y = 7$, its output is *False*; if $x = 4$ and $y = -10$, its output is *True*.

More about propositional functions later.

Notation

We shall often represent propositions by lower-case letters p, q, ...

Compound Propositions

Propositions may be modified by means of one or more *logical operators* to form what are called ***compound propositions***.

There are three logical operators:

conjunction: $\wedge$ meaning AND

disjunction: $\vee$ meaning OR

negation: $\neg$ meaning NOT

Example 2

p represents the proposition "Henry VIII had six wives".

q represents the proposition "The English Civil War took place in the nineteenth century".

(a) Connect these two propositions with OR. Is the resulting compound proposition true or false?

(b) Now connect them with AND. Is this compound proposition true or false?

(c) Is the 'opposite' of p true or false?

Answers

(a) $p \vee q$ is "Henry VIII had six wives or the English Civil War took place in the nineteenth century"

This is *true*. The first part of the compound proposition is true, and this is sufficient to make the whole statement true – if a little odd-sounding!

If you think that this example seems very artificial, imagine that you're taking part in a History Quiz; there are two questions left, and you need to get at least one right to win the quiz. You make the two statements above about Henry VIII and the Civil War. Do you win the quiz? Yes, because it is true that either Henry VIII had six wives or the English Civil War took place in the nineteenth century.

Note that this is an *inclusive* OR: in other words, we don't rule out *both* parts being true. So $p \vee q$ means "Either p is true, or q is true, or both".

(b) $p \wedge q$ is "Henry VIII had six wives and the English Civil War took place in the nineteenth century"

This is *false*.

To be true, both parts would have to be true. This is equivalent to your needing *both* questions right to win the quiz. You fail, because you got the second one wrong.

(c) The opposite of p, which we write as $\neg p$, is "Henry VIII did not have six wives". This is clearly false. And in general, if p is true, then $\neg p$ is false, and vice versa.

Example 3

p is "The printer is off-line"

q is "The printer is out of paper"

"r" is "The document has finished printing"

Write as English sentences, in as natural a way as you can:

(a) $p \vee q$

(b) $r \wedge q$

(c) $q \wedge \neg r$

(d) $\neg(p \vee q)$

Answers

(a) The printer is off-line or out of paper.

Notice how we often leave words out when we're writing or speaking English. This sounds much more natural than "The printer is off-line or the printer is out of paper".

(b) The document has finished printing and the printer is out of paper.

The subject of each part of the sentence is different now, so no words are missing this time.

(c) The printer is out of paper and the document has not finished printing.

But and *And*

The statement in (c) could be someone reporting a problem, and they might equally well say:

(c) The printer is out of paper *but* the document has not finished printing.

So note that, in logic, *but* and *and* mean the *same thing*. It's just that we use *but* to introduce a note of contrast or surprise. For example, we might well say:

The sun is shining brightly, *but* it's freezing cold outside.

Logically, we could use *and* to connect both parts of this sentence, but(!) it's more natural to use *but*.

In (d) what does $\neg(p \vee q)$ mean? Well, $p \vee q$ means either p is true or q is true (or both). When we place $\neg$ in front, we negate this. So it means (literally):

It is not true that either the printer is off-line or the printer is out of paper.

In other words:

(d) The printer is neither off-line nor out of paper.

Notice that it's often convenient to translate $\neg$ using the phrase 'It is not true that …'.

Logic Exercise 1

Click link for Logic Exercise 1.

Truth Tables

Consider the possible values of the compound proposition $p \wedge q$ for various combinations of values of p and q. The only combination of values that makes $p \wedge q$ true is where p and q are *both true*; any other combination will include a *false* and this will render the whole compound proposition false. On the other hand, the compound proposition $p \vee q$ will be true if *either* p or q (or both) is true; the only time $p \vee q$ is false is when both p and q are false.

We summarise conclusions like these in what is called a ***Truth Table***, the truth table for AND being:

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

The truth table for OR is:

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

The order of the Rows in a Truth Table

Notice the pattern of T's and F's in the first two columns of each of the truth tables above. In the first column (the truth values of *p*), there are 2 T's followed by 2 F's; in the second (the values of *q*), the T's and F's change on *each row*. We shall adopt this order of the rows throughout this text. Adopting a convention for the order of the rows in a Truth Table has two advantages:

It ensures that all combinations of T and F are included. (That may be easy enough with just two propositions and only four rows in the Truth Table; it's not so easy with, say, 4 propositions where there will be 16 rows in the table.)

It produces a standard, recognisable output pattern of T's and F's. So, for example, T, F, F, F is the output pattern (or 'footprint' if you like) of AND ($\wedge$), and T, T, T, F is the footprint of OR ($\vee$).

The truth table for NOT

Each of the two truth tables above had two 'input' columns (one for the values of *p* and one for *q*), and one 'output' column. They each needed four rows, of course, because there are four possible combinations of T's and F's when two propositions are combined. The truth table for NOT ($\neg$) will be simpler, since $\neg$ is a *unary* operation – one that requires a single proposition as input. So it just two columns – an input and an output – and two rows.

p	$\neg p$
T	F
F	T

Drawing up Truth Tables

The method for drawing up a truth table for any compound expression is described below, and four examples then follow. It is important to adopt a rigorous approach and to keep your work neat: there are plenty of opportunities for mistakes to creep in, but with care this is a very straightforward process, no matter how complicated the expression is. So:

Step 1: Rows

Decide how many rows the table will require. One input requires only two rows; two inputs require 4 rows; three require 8, and so on. If there are n propositions in the expression, 2^n rows will be needed.

Step 2: Blank Table

Draw up a blank table, with the appropriate number of input columns and rows, and a single output column wide enough to accommodate comfortably the output expression. If 8 or more rows are needed, you'll probably find it helps to rule a horizontal line across the table every *four* rows, in order to keep your rows straight.

Step 3: Input Values

Fill in all the input values, using the convention above for the Order of Rows in a Truth Table; that is to say, begin with the *right-most* input column and fill in the values T and F, alternating on *every* row. Then move to the next column to the left, and fill in T's and F's changing on *every second* row. And so on for all the remaining columns. The left-most column will then contain T's in the first half of all the rows in the table, and F's in the second half.

Step 4: Plan your strategy

Study carefully the *order* in which the operations involved in evaluating the expression are carried out, taking note of any brackets there may be. As in conventional algebra, you don't necessarily work from left to right. For example, the expression $p \vee \neg q$ will involve working out $\neg q$ first, then combining the result with p using $\vee$. When you've worked out the order in which you need to carry out each of the operations, rule additional columns under the *output* expression - one for each stage in the evaluation process. Then number each of the columns (at its foot) in the order in which it will be evaluated. The column representing the **final output** will be the last stage in the evaluation process, and will therefore have the highest number at its foot.

Step 5: Work down the columns

The final stage is to work **down each column** in the order that you've written down in *Step 4*. To do this, you'll use the truth tables for AND, OR and NOT above using values from the input columns and any other columns that have already been completed. Remember: work *down the columns*, not across the rows. That way, you'll only need to think about one operation at a time.

You're probably thinking that this all sounds incredibly complicated, but a few examples should make the method clear.

Worked examples

Example 4

Produce truth tables for:

(a) $\neg(\neg p)$

(b) $p \wedge (\neg q)$

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

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

Solutions

(a) $\neg(\neg p)$ is pretty obviously the same as p itself, but we'll still use the above method in this simple case, to show how it works, before moving on to more complicated examples. So:

Step 1: Rows

There's just one input variable here, so we shall need two rows.

Step 2: Blank Table

So the table is:

p	$\neg(\neg p)$
.	
.	

Step 3: Input Values

Next, we fill in the input values: just one T and one F in this case:

p	$\neg(\neg p)$
T	
F	

Step 4: Plan your strategy

As in 'ordinary' algebra we evaluate whatever's in brackets first, so we shall first (1) complete the $(\neg p)$ values, followed (2) by the left-hand $\neg$ symbol, which gives us the final output values of the whole expression. We rule an extra column to separate these two processes, and write the (1) and (2) at the foot of these two columns. Thus:

p	$\neg$	$(\neg p)$
T		
F		
	(2) Output	(1)

Step 5: Work down the columns

Finally we insert the values in column (1) – F followed by T – and then use *these* values to insert the values in column (2). So the completed truth table is:

p	$\neg$	$(\neg p)$
T	T	F
F	F	T
	(2) Output	(1)

As we said at the beginning of this example, $\neg(\neg p)$ is clearly the same as p, so the pattern of the output values, T followed by F, is identical to the pattern of the input values. Although this may seem trivial, the same technique works in much more complex examples, where the results are far from obvious!

(b) $p \wedge (\neg q)$

Step 1

There are two input variables, p and q, so we shall need four rows in the table.

Steps 2 & 3

In the q column write T's and F's alternating on every row; in the p column alternate every two rows. At this stage, the table looks like this:

p	q	$p \wedge (\neg q)$
T	T	
T	F	
F	T	
F	F	

Steps 4 & 5

As *Example (a)*, we begin (1) by evaluating the expression in brackets, $(\neg q)$, and then (2) we combine these results with p using the $\wedge$ operator. So we divide the output section of the table into two columns; then work down column (1) and finally column (2). The completed table is:

p	q	$p \wedge$	$(\neg q)$
T	T	F	F
T	F	T	T
F	T	F	F
F	F	F	T
		(2) output	(1)

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

Steps 1 to 3

As in *Example (b)*.

Steps 4 & 5

There will be 5 stages in evaluating the expression $(p \wedge q) \vee (\neg p \vee \neg q)$. In order, they are:

(1) The first bracket: $(p \wedge q)$

(2) The $\neg p$ in the second bracket

(3) The $\neg q$ in the second bracket

(4) The $\vee$ in the second bracket

(5) The $\vee$ between the two brackets. This final stage, then, represents the output of the complete expression.

Reminder: Don't work **across** the rows; work **down** the columns in order (1) to (5). That way, you'll only have to deal with a single operation at a time.

The completed table is:

p	q	$(p \wedge q)$	$\vee$	$(\neg p$	$\vee$	$\neg q)$
T	T	T	T	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	T	F
F	F	F	T	T	T	T
		(1)	(5) output	(2)	(4)	(3)

Notice that the output consist solely of T's. This means that $(p \wedge q) \vee (\neg p \vee \neg q)$ is always *true* whatever the values of p and q. It is therefore a *tautology* (see below).

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

This simple expression involves 3 input variables, and therefore requires $2^3 = 8$ rows in its truth table. When drawing this truth table by hand, rule a line below row 4 as an aid to keeping your working neat. It is shown as a double line in this table. The completed table is shown below.

p	q	r	$q \wedge$	$(p \vee r)$
T	T	T	T	T
T	T	F	T	T
T	F	T	F	T
T	F	F	F	T
F	T	T	T	T
F	T	F	F	F
F	F	T	F	T
F	F	F	F	F
			(2) output	(1)

Tautology

An expression which always has the value *true* is called a ***tautology***.

In addition, any statement which is redundant, or idempotent, is also referred to as a tautology, and for the same reason previously mentioned. If **P** is True then we can be sure that **P** □ **P** is true, and **P** □ **P** is also true.

Logic Exercise 2

Click link for Logic Exercise 2.

Order of Precedence

In 'ordinary' algebra, the order of precedence in carrying out operations is:

1 brackets

2 exponents (powers)

3 × and ÷

4 + and -

In the algebra of logic, brackets will often be inserted to make clear the order in which operations are to be carried out. To avoid possible ambiguity, the agreed rules of precedence are:

1 brackets

2 NOT (¬)

3 AND (∧)

4 OR (∨)

So, for example, $p \vee q \wedge r$ means:

Evaluate $q \wedge r$ first.

Then combine the result with $p \vee$.

Since it would be easy to misinterpret this, it is recommended that brackets are included, even if they are not strictly necessary. So $p \vee q \wedge r$ will often be written $p \vee (q \wedge r)$.

Logically Equivalent Propositions

Look back to your answers to questions 2 and 3 in *Exercise 2*. In each question, you should have found that the last columns of the truth tables for each pair of propositions were the same.

Whenever the final columns of the truth tables for two propositions p and q are the same, we say that p and q are ***logically equivalent***, and we write:

$p \equiv q$

Example 5

Construct truth tables for

(i) $p \vee (q \wedge r)$, and

(ii) $(p \vee q) \wedge (p \vee r)$,

and hence show that these propositions are logically equivalent.

Solution

(i)

p	q	r	$p \vee$	$(q \wedge r)$
T	T	T	T	T
T	T	F	T	F
T	F	T	T	F
T	F	F	T	F
F	T	T	T	T
F	T	F	F	F
F	F	T	F	F
F	F	F	F	F
			(2) output	(1)

(ii)

p	q	r	$(p \lor q)$	$\land$	$(p \lor r)$
T	T	T	T	T	T
T	T	F	T	T	T
T	F	T	T	T	T
T	F	F	T	T	T
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	F	F	T
F	F	F	F	F	F
			(1)	(3) output	(2)

The outputs in each case are T, T, T, T, T, F, F, F. The propositions are therefore logically equivalent.

Example 6

Construct the truth table for $\neg(\neg p \lor \neg q)$, and hence find a simpler logically equivalent proposition.

Solution

p	q	$\neg$	$(\neg p$	$\lor$	$\neg q)$
T	T	T	F	F	F
T	F	F	F	T	T
F	T	F	T	T	F
F	F	F	T	T	T
		(4) output	(1)	(3)	(2)

We recognise the output: T, F, F, F as the 'footprint' of the AND operation. So we can simplify $\neg(\neg p \lor \neg q)$ to

$$p \land q$$

Laws of Logic

Like sets, logical propositions form what is called a Boolean Algebra [1]: the laws that apply to sets have corresponding laws that apply to propositions also. Namely:

Commutative Laws

$$p \land q \equiv q \land p$$

$$p \lor q \equiv q \lor p$$

Associative Laws

$$(p \land q) \land r \equiv p \land (q \land r)$$

$$(p \lor q) \lor r \equiv p \lor (q \lor r)$$

Distributive Laws

$$p \land (q \lor r) \equiv (p \land q) \lor (p \land r)$$

$$p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$$

Idempotent Laws

$$p \land p \equiv p$$

$$p \lor p \equiv p$$

Identity Laws

$p \land \mathrm{F} \equiv \mathrm{F}$

$p \lor \mathrm{F} \equiv p$

$p \land \mathrm{T} \equiv p$

$p \lor \mathrm{T} \equiv \mathrm{T}$

Involution Law

$\neg(\neg p) \equiv p$

De Morgan's Laws

$\neg(p \lor q) \equiv (\neg p) \land (\neg q)$

(sometimes written p NOR q)

$\neg(p \land q) \equiv (\neg p) \lor (\neg q)$

(sometimes written p NAND q)

Complement Laws

$p \land \neg p \equiv \mathrm{F}$

$p \lor \neg p \equiv \mathrm{T}$

$\neg\mathrm{T} \equiv \mathrm{F}$

$\neg\mathrm{F} \equiv \mathrm{T}$

Worked Examples

Example 7

Propositional functions p, q and r are defined as follows:

p is "$n = 7$"

q is "$a > 5$"

r is "$x = 0$"

Write the following expressions in terms of p, q and r, and show that each pair of expressions is logically equivalent. State carefully which of the above laws are used at each stage.

(a)

$((n = 7) \lor (a > 5)) \land (x = 0)$

$((n = 7) \land (x = 0)) \lor ((a > 5) \land (x = 0))$

(b)

$\neg((n = 7) \land (a \le 5))$

$(n \ne 7) \lor (a > 5)$

(c)

$(n = 7) \lor (\neg((a \le 5) \land (x = 0)))$

$((n = 7) \lor (a > 5)) \lor (x \ne 0)$

Solutions

(a)

$(p \lor q) \land r$

$(p \land r) \lor (q \land r)$

$$\begin{aligned}(p \vee q) \wedge r &= r \wedge (p \vee q) && \textit{Commutative Law}\\ &= (r \wedge p) \vee r \wedge q) && \textit{Distributive Law}\\ &= (p \wedge r) \vee (q \wedge r) && \textit{Commutative Law} \text{ (twice)}\end{aligned}$$

(b)

First, we note that

$\neg q$ is "$a \leq 5$"; and

$\neg p$ is "$n \neq 7$".

So the expressions are:

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

$\neg p \vee q$

$$\begin{aligned}\neg(p \wedge \neg q) &= \neg p \vee \neg(\neg q) && \textit{De Morgan's Law}\\ &= \neg p \vee q && \textit{Involution Law}\end{aligned}$$

(c)

First, we note that

$\neg r$ is "$x \neq 0$".

So the expressions are:

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

$(p \vee q) \vee \neg r$

$$\begin{aligned}p \vee (\neg(\neg q \wedge r)) &= p \vee (\neg(\neg q) \vee \neg r) && \textit{De Morgan's Law}\\ &= p \vee (q \vee \neg r) && \textit{Involution Law}\\ &= (p \vee q) \vee \neg r && \textit{Associative Law}\end{aligned}$$

Logic Exercise 3

Click link for Logic Exercise 3.

Logic Page 2

Logic continues on Page 2.

Previous topic:../Number theory/|Contents:Discrete Mathematics|Next topic:../Enumeration/

References

[1] http://en.wikipedia.org/wiki/Boolean_algebra

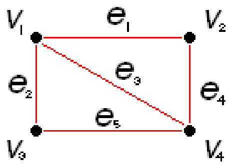
Graph theory

A *graph* is a mathematical way of representing the concept of a "network".

Introduction

A network has points, connected by lines. In a graph, we have special names for these. We call these points *vertices* (sometimes also called nodes), and the lines, *edges*.

Here is an example graph. The edges are red, the vertices, black.



In the graph, v_1, v_2, v_3, v_4 are vertices, and e_1, e_2, e_3, e_4, e_5 are edges.

Definitions of graph

There are several roughly equivalent definitions of a **graph**. Most commonly, a graph G is defined as an ordered pair $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is called the graph's **vertex-set** and $E = \{e_1, \ldots, e_m\} \subset \{\{x, y\} | x, y \in V\}$ is called the graph's **edge-set**. Given a graph G , we often denote the vertex—set by $V(G)$ and the edge—set by $E(G)$. To visualize a graph as described above, we draw n dots corresponding to vertices $v_1, \cdots, v_n$. Then, for all $i, j \in \{1, \ldots, n\}$ we draw a line between the dots corresponding to vertices v_i, v_j if and only if there exists an edge $\{v_i, v_j\} \in E$. Note that the placement of the dots is generally unimportant; many different pictures can represent the same graph.

Alternately, using the graph above as a guide, we can define a graph as an ordered triple $G = (V, E, f)$:

- a set of vertices, commonly called V
- a set of edges, commonly called E
- a relation $f : E \to \{\{x, y\} | x, y \in V\}$ that maps to each edge a set of *endpoints*, known as the *edge-endpoint relation*. We say an edge $e \in E$ is **incident** to a vertex $v \in V$ iff $v \in f(e)$.

In the above example,

- V={v_1, v_2, v_3, v_4}
- E={e_1, e_2, e_3, e_4, e_5}
- f such that e_1 maps to {v_1, v_2}, e_2 maps to {v_1, v_3}, e_3 maps to {v_1, v_4}, e_4 maps to {v_2, v_4}, and e_5 maps to {v_3, v_4}.

If f is not injective — that is, if $\exists e, e' \in E$ such that $e \neq e', f(e) = f(e')$ — then we say that G is a **multigraph** and we call any such edges $e, e' \in E$ *multiple edges*. Further, we call edges $e \in E$ such that $|f(e)| = 1$ **loops**. Graphs without multiple edges or loops are known as **simple graphs**.

Graphs can, conceivably, be infinite as well, and thus we place no bounds on the sets V and E. We will not look at infinite graphs here.

Directions, Weights, and Flows

We define a **directed graph** as a graph such that f maps into the set of ordered pairs $\{(x,y)|x,y \in V\}$ rather than into the family of two-element sets $\{\{x,y\}|x,y \in V\}$. We can think of an edge $e \in E$ such that $f(e) = (x,y)$ as 'pointing' from x to y. As such we would say that x is the *tail* of edge e and that y is the *head*. This is one of the vagaries of graph theory notation, though. We could just as easily think of x as the head and y as the tail. To represent a directed graph, we can draw a picture as described and shown above, but place arrows on every edge corresponding to its direction.
In general, a **weight** on a graph G is some function $c : E(G) \to \mathbb{R}$.

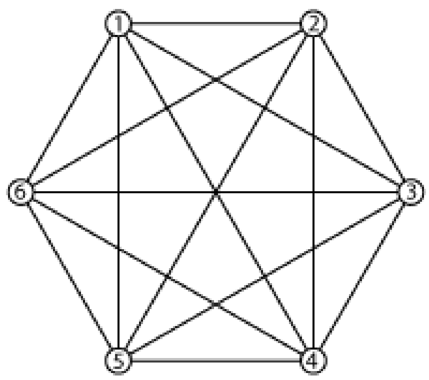
A **flow** (G,c) is a directed graph $G = (V,E,f)$ paired with a weight function such that the weight "going into" any vertex is the same amount as the weight "going out" of that vertex. To make this more formal, define sets

- $f^+(v) = \{e \in E(G)|f(e) = (v,x), x \in V(G)\},\ \forall v \in V(G)$
- $f^-(v) = \{e \in E(G)|f(e) = (x,v), x \in V(G)\},\ \forall v \in V(G)$

Then, formally stated, our requirement on the weight function is $\sum_{e \in f^+(v)} c(e) = \sum_{e \in f^-(v)} c(e),\ \forall v \in V(G).$

Special Graphs

Some graphs occur frequently enough in graph theory that they deserve special mention. One such graphs is the *complete graph* on n vertices, often denoted by K_n. This graph consists of n vertices, with each vertex connected to every other vertex, and every pair of vertices joined by exactly one edge. Another such graph is the *cycle graph* on *n* vertices, for *n* at least 3. This graph is denoted C_n and defined by V := {1,2,..,n} and E := Template:1,2,{2,3}, ..., {n-1,n},Template:N,1. Even easier is the *null graph* on *n* vertices, denoted N_n; it has *n* vertices and no edges! Note that $N_1 = K_1$ and $C_3 = K_3$.



The complete graph on 6 vertices

Some Terms

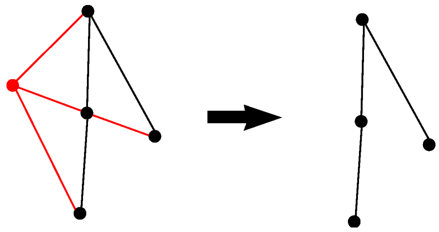
Two vertices are said to be *adjacent* if there is an edge joining them. The word *incident* has two meanings:

- An edge *e* is said to be incident to a vertex *v* if *v* is an endpoint of *e*.
- Two edges are also incident to each other if both are incident to the same vertex.

Two graphs *G* and *H* are said to be *isomorphic* if there is a one-to-one function from (or, if you prefer, one-to-one correspondence between) the vertex set of *G* to the vertex set of *H* such that two vertices in *G* are adjacent if and only if their images in *H* are adjacent. (Technically, the multiplicity of the edges must also be preserved, but our definition suffices for simple graphs.)

Subgraphs

A *subgraph* is a concept akin to the subset. A subgraph has a subset of the vertex set V, a subset of the edge set E, and each edge's endpoints in the larger graph has the same edges in the subgraph. A



Generated Subgraphs

A subgraph H of G is *generated* by the vertices $\{\ a, b, c, \ldots\}$ $\in H$ if the edge set of H consists of all edges in the edge set of G that joins the vertices in $H = \{\ a, b, c, \ldots\}$.
A *path* is a sequence of edges $< e_1, \ldots, e_N >$ such that e_i is adjacent to e_{i+1} for all i from 1 to N-1. Two vertices are said to be connected if there is a path connecting them.

Trees and Bipartite Graphs

A *tree* is a graph that is (i) connected, and (ii) has no cycles. Equivalently, a tree is a connected graph with exactly $n - 1$ edges, where there are n nodes in the tree.

A *Bipartite graph* is a graph whose nodes can be partitioned into two disjoint sets U and W such that every edge in the graph is incident to one node in U and one node in W. A tree is a bipartite graph.

A *complete bipartite graph* is a bipartite graph in which each node in U is connected to every node in W; a complete bipartite graph in which U has n vertices and V has m vertices is denoted $K_{n,m}$.

Adjacent,Incident,End Vertices

Self loops,Parallel edges,Degree of Vertex

Pendant Vertex : Vertex Degree one "Pendant Vertex" Isolated Vertex : Vertex Degree zero "Isolated Vertex"

Hamiltonian and Eulerian Paths

Hamiltonian Cycles: A Hamiltonian Cycle received its name from Sir William Hamilton who first studied the travelling salesman problem. A Hamiltonian cycle is a path that visits every vertex once and only once i.e. it is a walk, in which no edge is repeated (a trail) and therefore a trail in which no vertex is repeated (a path). Note also it is a cycle, the last vertex is joined to the first.

A graph is said to be Eulerian if it is possible to traverse each edge once and only once, i.e. it has no odd vertices or it has an even number of odd vertices (semi-Eulerian). This has implications for the Königsberg problem. It may be easier to imagine this as if it is possible to trace the edges of a graph with a pencil without lifting the pencil off the paper or going over any lines.

Planar Graphs

A *planar graph* is an undirected graph that can be drawn on the plane or on a sphere in such a way that no two edges cross, where an edge $e = (u, v)$ is drawn as a continuous curve (it need not be a straight line) from u to v.

Kuratowski proved a remarkable fact about planar graphs: A graph is planar if and only if it does not contain a subgraph homeomorphic to K_5 or to $K_{3,3}$. (Two graphs are said to be homeomorphic if we can shrink some components of each into single nodes and end up with identical graphs. Informally, this means that non-planar-ness is caused by only two things—namely, having the structure of K_5 or $K_{3,3}$ within the graph).

Coloring Graphs

A graph is said to be planner if it can be drawn on a plane in such way that no edges cross one anather except of course of vertices

Each term, the Schedules Office in some university must assign a time slot for each final exam. This is not easy, because some students are taking several classes with finals, and a student can take only one test during a particular time slot. The Schedules Office wants to avoid all conflicts, but to make the exam period as short as possible.

We can recast this scheduling problem as a question about coloring the vertices of a graph. Create a vertex for each course with a final exam. Put an edge between two vertices if some student is taking both courses. For example, the scheduling graph might look like this: Next, identify each time slot with a color. For example, Monday morning is red, Monday afternoon is blue, Tuesday morning is green, etc.

Assigning an exam to a time slot is now equivalent to coloring the corresponding vertex. The main constraint is that adjacent vertices must get different colors; otherwise, some student has two exams at the same time. Furthermore, in order to keep the exam period short, we should try to color all the vertices using as few different colors as possible. For our example graph, three colors suffice: red, green, blue.

The coloring corresponds to giving one final on Monday morning (red), two Monday afternoon (blue), and two Tuesday morning (green)...

K Coloring

Many other resource allocation problems boil down to coloring some graph. In general, a graph G is kcolorable if each vertex can be assigned one of k colors so that adjacent vertices get different colors. The smallest sufficient number of colors is called the chromatic number of G. The chromatic number of a graph is generally difficult to compute, but the following theorem provides an upper bound:

Theorem 1. A graph with maximum degree at most k is (k + 1)colorable.

Proof. We use induction on the number of vertices in the graph, which we denote by n. Let P(n) be the proposition that an nvertex graph with maximum degree at most k is (k + 1)colorable. A 1 vertex graph has maximum degree 0 and is 1colorable, so P(1) is true.

Now assume that P(n) is true, and let G be an (n + 1)vertex graph with maximum degree at most k. Remove a vertex v, leaving an nvertex graph G . The maximum degree of G is at most k, and so G is (k + 1)colorable by our assumption P(n). Now add back vertex v. We can assign v a color different from all adjacent vertices, since v has degree at most k and k + 1 colors are available. Therefore, G is (k + 1)colorable. The theorem follows by induction.

Weighted Graphs

A **weighted graph** associates a label (weight) with every edge in the graph. Weights are usually real numbers, and often represent a "cost" associated with the edge, either in terms of the entity that is being modeled, or an optimization problem that is being solved.

Recursion

In this section we will look at certain mathematical processes which deal with the fundamental property of ***recursion*** at its core.

What is recursion?

Recursion, simply put, is the process of describing an action in terms of itself. This may seem a bit strange to understand, but once it "clicks" it can be an extremely powerful way of expressing certain ideas.

Let's look at some examples to make things clearer.

Exponents

When we calculate an exponent, say x^3, we multiply x by itself three times. If we have x^5, we multiply x by itself five times.

However, if we want a recursive definition of exponents, we need to define the action of taking exponents in terms of itself. So we note that x^4 for example, is the same as $x^3 \times x$. But what is x^3? x^3 is the same as $x^2 \times x$. We can continue in this fashion up to $x^0=1$. What can we say in general then? Recursively,

$$x^n = x \times (x^{n-1})$$

with the fact that

$$x^0=1$$

We need the second fact because the definitions fail to make sense if we continue with negative exponents, and we would continue indefinitely!

Recursive definitions

Reducing the problem into same problem by smaller inputs. for example

```
          a power n
          2 power 4
  the recursion(smaller inputs) of this function is = 2.2.2.2.1
  for this we declare some recursive definitions
 a=2
 n=4
 f(0)=1
 f(1)=2
 f(2)=2
 f(3)=2
 f(4)=2
 for this recursion we form a formula f(x)= a.f(n-1)
 by putting these samaller values we get the same answer.
```

Recurrence relations

In mathematics, we can create recursive *functions*, which depend on its previous values to create new ones. We often call these *recurrence relations*.

For example, we can have the function :$f(x)=2f(x-1)$, with $f(1)=1$ If we calculate some of f's values, we get

1, 2, 4, 8, 16, ...

However, this sequence of numbers *should* look familiar to you! These values are the same as the function 2^x, with x = 0, 1, and so on.

What we have done is found a *non-recursive* function with the same values as the *recursive* function. We call this *solving* the recurrence relation.

Linear recurrence relations

We will look especially at a certain kind of recurrence relation, known as *linear*.

Here is an example of a linear recurrence relation:

$f(x)=3f(x-1)+12f(x-2)$, with f(0)=1 and f(1)=1

Instead of writing $f(x)$, we often use the notation a_n to represent $a(n)$, but these notations are completely interchangeable.

Note that a linear recurrence relation should always have stopping cases, otherwise we would not be able to calculate $f(2)$, for example, since what would $f(1)$ be if we did not define it? These stopping cases when we talk about linear recurrence relations are known as *initial conditions*.

In general, a linear recurrence relation is in the form

$a_n=A_1a_{n-1} + A_2a_{n-2} + ... + A_ja_{n-j}$
with $f(t_1)=u_1, f(t_2)=u_2, ..., f(t_j)=u_j$ as initial conditions.

The number j is important, and it is known as the *order* of the linear recurrence relation. Note we always need at least j initial conditions for the recurrence relation to make sense.

Recall in the previous section we saw that we can find a nonrecursive function (a *solution*) that will take on the same values as the recurrence relation itself. Let's see how we can solve some linear recurrence relations - we can do so in a very systematic way, but we need to establish a few theorems first.

Solving linear recurrence relations

Sum of solutions

This theorem says that:

If $f(n)$ and $g(n)$ are both solutions to a linear recurrence relation $a_n=Aa_{n-1}+Ba_{n-2}$, their sum is a solution also.

This is true, since if we rearrange the recurrence to have $a_n-Aa_{n-1}-Ba_{n-2}=0$ And we know that $f(n)$ and $g(n)$ are solutions, so we have, on substituting into the recurrence

$f(n)-Af(n-1)-Bf(\mathrm{n}-2)=0$

$g(n)-Ag(n-1)-Bg(\mathrm{n}-2)=0$

If we substitute the sum $f(n)+g(n)$ into the recurrence, we obtain

$(f(n)+g(n))-A(f(n-1)+g(n-1))-B((f(n-2)+g(n-2))=0$

On expanding out, we have

$f(n)-Af(n-1)-Bf(\mathrm{n}-2)+g(n)-Ag(n-1)-Bg(\mathrm{n}-2)$

But using the two facts we established first, this is the same as

$0+0=0$

So $f(n)+g(n)$ is indeed a solution to the recurrence.

General solution

The next theorem states that:

Say we have a second-order linear recurrence relation, $a_n - Aa_{n-1} - Ba_{n-2} = 0$, with supplied initial conditions.
Then αr^n is a solution to the recurrence, where r is a solution of the quadratic equation

$x^2 - \mathrm{A}x - \mathrm{B} = 0$

which we call the *characteristic equation*.
We guess (it doesn't matter why, accept it for now) that αr^n may be a solution. We can prove that this is a solution IF (and only if) it solves the characteristic equation ;

We substitute αr^n (r not zero) into the recurrence to get

$\alpha r^n - A\alpha r^{n-1} - B\alpha r^{n-2} = 0$

then factor out by αr^{n-2}, the term farthest on the right

$\alpha r^{n-2}(r^2 - Ar - B) = 0$

and we know that r isn't zero, so r^{n-2} can never be zero. So $r^2 - Ar - \mathrm{B}$ must be zero, and so αr^n, with r a solution of $r^2 - Ar - B = 0$, will indeed be a solution of the linear recurrence. Please note that we can easily generalize this fact to higher order linear recurrence relations.

Where did this come from? Why does it work (beyond a rote proof)? Here's a more intuitive (but less mathematically rigorous) explanation.

Solving the *characteristic equation* finds a function that satisfies the linear recurrence relation, and conveniently doesn't require the summation of all n terms to find the *n*th one.
We want : a function F(n) such that F(n) = A * F(n-1) + B * F(n-2)
We solve : $x^2 = A * x + B$, and call the solution(s) *r*. There can be more than one value of r, like in the example below!
We get : a function $F(n) = r^n$ that satisfies F(n) = A * F(n-1) + B * F(n-2)
Let's check: Does $r^n = A*r^{n-1} + B*r^{n-2}$? Divide both sides by r^{n-2} and you get $r^2 = A*r + B$, which must be true because r is a solution to $x^2 = A* x + B$
Why does $a*r^n$ also satisfy the recurrence relation? If F(n)is a solution to the recurrence relation, so is F(n)+ F(n), based on the "Sum of Solutions" theorem above. One can then take that sum, 2*F(n), and add another F(n) to get 3*F(n), and it will still satisfy the recurrence (and so on...). Thus any whole number multiple of F(n), such as *a**F(n) will satisfy the recurrence relation (*a* can also be any fraction and probably any real number at all, but I'm too lazy to adapt the current explanation). Because r^n satisfies the recurrence, so does $a*r^n$.

Because we have a second order recurrence, the general solution is the sum of two solutions, corresponding to the two roots of the characteristic equation. Say these are r and s. The the general solution is $C(r^n)+D(s^n)$ where C,D are some constants. We find them using the two (there must be two so that we can find C and D) starting values of the relation. Substituting these into the general solution will give two equtions which we can (hopefully) solve.

Example

Let's work through an example to see how we can use the above theorems to solve linear recurrence relations. Examine the function $a(n)$ given here

$a(n)=a(n\text{-}1)+2a(n\text{-}2)$

The characteristic equation of this recurrence relation is

$r^2\text{-}r\text{-}2 = 0$ from above, as A=1 and B=2

i.e. $(r\text{-}2)(r+1)=0$ which has roots 2, -1.

So the general solution is $C(2^n)+D(-1)^n$.

To find C and D for this specific case, we need two starting values, let's say $a(1) = 0$ and $a(2) = 2$. These give a system of two equations

$0 = C(2^1)+D(-1)^1$

$2 = C(2^2)+D(-1)^2$

Solving these two equations yields: C = 1/3 , D = 2/3, so the solution is $1/3*(2^n)+2/3*(-1)^n$.

Article Sources and Contributors

Introduction *Source*: http://en.wikibooks.org/w/index.php?oldid=2379605 *Contributors*: Adrignola, Alsocal, Dysprosia, Jack the Leprechaun, Jguk, Recent Runes, The enemies of god, 11 anonymous edits

Set theory *Source*: http://en.wikibooks.org/w/index.php?oldid=2533809 *Contributors*: Adrignola, Arthurvogel, B9 hummingbird hovering, CloudNine, Cybermaster, Dysprosia, Fishpi, FlyingPenguins, Grendelkhan, Gruntmaster6000, Jguk, Jonathan48, MarkHudson, Matthew Low, Matthewglenadams, Nigeltn35, Panic2k4, Paul, Pribis, Pzone, Recent Runes, Sakiv classic, Savh, Shamilj, Sigma 7, Skimpysking, Spal, The enemies of god, 138 anonymous edits

Functions and relations *Source*: http://en.wikibooks.org/w/index.php?oldid=2529832 *Contributors*: Adrignola, AndersJansson, Avanderbergh, Avicennasis, BCW, Calberty, Cybermaster, Dysprosia, Greenbreen, Grendelkhan, JenVan, Jguk, Jomegat, Leonidas, Panic2k4, Patrick, Peter.shank, QuiteUnusual, Recent Runes, Satya Mounika, Sigma 7, Soeb, Theresa knott, Webaware, 115 anonymous edits

Logic *Source*: http://en.wikibooks.org/w/index.php?oldid=2413077 *Contributors*: Adrignola, Jgv9a, Nigeltn35, Psyced, Xania, 4 anonymous edits

Graph theory *Source*: http://en.wikibooks.org/w/index.php?oldid=2506973 *Contributors*: Adrignola, DavidCary, Dysprosia, Fermatprime, Iamunknown, Jguk, Jomegat, Kuldeep, Laleena, MikeBorkowski, QuiteUnusual, Ransomw, Recent Runes, Rich Farmbrough, Sigma 7, SpaceMoose, Tannersf, Valentine.w, 33 anonymous edits

Recursion *Source*: http://en.wikibooks.org/w/index.php?oldid=2508904 *Contributors*: Adrignola, Avicennasis, Dysprosia, Eka.a.kurniawan, GoldenGun, Jguk, QuiteUnusual, Vogone, Xania, 27 anonymous edits

Image Sources, Licenses and Contributors

License