Semantic Tableaux for Propositional Logic

Contents

Chapter 1

Review: Discrete Mathematics

1.1 Reasoning

We want to be able to reason about real systems. For example, consider the following piece of code:

```
repeat
   sum := sum + x;
   x := x + 1;
until x = 0;
```

This says that we should continue to add the value of the variable x to the value of the variable sum, and then add 1 to the value of x, until x has the value 0. So, if x starts off with the value -4, the program would add -4, -3, -2, -1 in turn to the value of sum, and then move on to the next statement.

One property one wants any such program to have is "termination", i.e. the program will eventually halt and deliver an answer. In the case where x has the value -4 coming into this statement, we have seen that there is no problem. However, if x had the value 4, then we would add 4, 5, 6, 7, ... successively to sum without termination (we might well get an overflow, but we won't get a proper answer). It might well be that, for any valid set of data, x would necessarily have a negative value coming into this statement; we might even put a comment in the code to this effect:

```
{we must have x < 0 here}
repeat
   sum := sum + x;
   x := x + 1;
until x = 0;
```

We could then argue that, since x has a negative value coming into this statement, then the program (or, at least, this part of it) must terminate. We express this as follows:

$$(\texttt{x} < 0) \;\Rightarrow\; \text{(the program terminates)}.$$

The idea is that, if certain assumptions are satisfied (such as the data lying in a certain range) then we get the desired conclusion (such as the program terminating). The argument used to get from the assumptions to the conclusion will involve reasoning; notice that our argument has the form "if ... then ...".

In general, when reasoning, we consider statements of the form $P \Rightarrow Q$, where P and Q are **propositions** and $\Rightarrow$ denotes "implies"; we say "P implies Q" or "if P then Q". A proposition is a statement that can be true or false. For example:

> England won the Rugby World Cup.
> Auckland is in New Zealand.
> Hippopotami are purple.

are all propositions, whereas

Are you wet?
Don't step on the grass!
Round up the usual suspects.

are not.

For another example of an "if ... then ..." situation, we might have

$$x > 3 \Rightarrow x > 1, \tag{1.1}$$

which says that, if x is greater than three, then x is greater than one. Note that this is not the same as

$$x > 1 \Rightarrow x > 3. \tag{1.2}$$

We say that the statement (1.1) is the **converse** of (1.2); in general, $Q \Rightarrow P$ is the converse of $P \Rightarrow Q$. Note that $P \Rightarrow Q$ and $Q \Rightarrow P$ are not the same; in this particular example, (1.1) is true and (1.2) is false.

To put this another way, $P \Rightarrow Q$ means that "whenever P is true, then Q must be true as well". So we are allowing the possibility that P can be false (in which case $P \Rightarrow Q$ tells us nothing about whether or not Q is true). For example, in (1.1), we could put $x = 2$ (making $x > 3$ false and $x > 1$ true) or $x = 0$ (making both $x > 3$ and $x > 1$ false).

To prove an assertion of the form $P \Rightarrow Q$, we can therefore assume that P is true and then deduce that Q is true. Let's look at an example.

EXAMPLE. Prove that: $x^2 - 5x + 6 < 0 \Rightarrow x > 0$.

Assume that $x^2 - 5x + 6 < 0$, so that $5x > x^2 + 6$. Since $x^2 \geqslant 0$ for any x, we have that $5x > 6$, i.e. that $x > \frac{6}{5}$, and so $x > 0$. □

We could have arranged our proof in the form:

$$\begin{aligned} x^2 - 5x + 6 < 0 &\Rightarrow 5x > x^2 + 6 \\ &\Rightarrow 5x > 6 \qquad (\text{since } x^2 \geqslant 0) \\ &\Rightarrow x > \tfrac{6}{5} \\ &\Rightarrow x > 0. \end{aligned}$$

One can, in fact, formalize things even more; we will come to formal logic later in the course.

In this example, P is the statement $x^2 - 5x + 6 < 0$ and Q the statement $x > 0$. We have started by assuming that P is true and then deduced that Q is true. In general, reasoning starts with assumptions and, by means of logical arguments, arrives at conclusions.

One can sum up the interpretation of $P \Rightarrow Q$ by means of a **truth-table**:

P	Q	$P \Rightarrow Q$
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`true`
`false`	`false`	`true`

We see that, whenever P is true, then Q must be true; if P is false, then Q can either be true or false. This is why, when trying to prove $P \Rightarrow Q$, we could assume that P is true and then attempt to deduce that Q is true.

We see that $P \Rightarrow Q$ is false precisely when P is true and Q is false; so, in order that $P \Rightarrow Q$ be true, we need that, if Q is false, then P must be false also. In fact, we see that this is an equivalent statement. Therefore an alternative method of proving $P \Rightarrow Q$ is to start by assuming that Q is false and then deduce that P must also be false. This method of proof is known as **Proof by Contraposition**.

EXAMPLE. To prove that $x^2 - 5x + 6 < 0 \Rightarrow x > 0$, we shall prove the equivalent statement:

$$x \leqslant 0 \Rightarrow x^2 - 5x + 6 \geqslant 0.$$

Assume that $x \leqslant 0$, so that $-x \geqslant 0$. Then we have $-5x \geqslant 0$, so that $-5x + 6 \geqslant 0$, and then $x^2 - 5x + 6 \geqslant 0$ (since $x^2 \geqslant 0$). □

For another example of contraposition, consider the following statements:

> If the switch is off then the process is stopped.
> If the process is not stopped then the switch is not off.

These are equivalent statements. Another way of looking at this is to say that there is no possibility of the swich being off and the process not being stopped. This is not the same as saying

> If the process is stopped then the switch is off

however; there could well be other reasons why the process is stopped (such as a power cut).

Another method of proving $P \Rightarrow Q$ is to start by assuming that P is true and Q is false, and deduce a statement that we know cannot possibly be true; again, this shows that, if P is true, then Q must be true as well. This method of proof is known as **Proof by Contradiction**.

EXAMPLE. Prove that: $x^2 - 5x + 6 < 0 \Rightarrow x > 0$.

Assume that $x^2 - 5x + 6 < 0$ and that $x \leqslant 0$. Since $x^2 - 5x + 6 < 0$ we have that $x^2 < 5x - 6$. Now, since $x \leqslant 0$, we have $5x \leqslant 0$, and so $x^2 < 5x - 6 \leqslant -6$, a contradiction. □

If we have both $P \Rightarrow Q$ and $Q \Rightarrow P$, then we denote this by $P \Leftrightarrow Q$; in this case P is true if and only if Q is true. So, to prove $P \Leftrightarrow Q$, we need to prove both $P \Rightarrow Q$ and $Q \Rightarrow P$. For example, consider the following:

EXAMPLE. Prove that, for any whole number x,

$$x \text{ is even} \;\Leftrightarrow\; x^2 \text{ is even.}$$

If we let P denote the statement "x is even" and Q denote the statement "x^2 is even", then we are being asked to prove $P \Leftrightarrow Q$.

$\Rightarrow$) We want to show that, if x is even, then x^2 is even. So assume that x is even, i.e. that $x = 2m$ for some whole number m. Then

$$x^2 = (2m)^2 = 4m^2 = 2(2m^2)$$

is even.

$\Leftarrow$) We want to show that, if x^2 is even, then x is even. We prove this by contraposition, in that we will show that, if x is not even, then x^2 is not even.

Assume that x is not even, so that x is odd; so $x = 2m + 1$ for some whole number m. Then we have that

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

is odd, and so x^2 is not even. □

There is a point which we have glossed over in these examples. For example, when we proved that

$$x^2 - 5x + 6 < 0 \Rightarrow x > 0,$$

we were proving that this was true for all x, i.e. that, whatever the value of x, if $x^2 - 5x + 6 < 0$, then we must have that $x > 0$; another way of putting this is to say that there can be no value of x with $x^2 - 5x + 6 < 0$ and $x \leqslant 0$. So what we were really proving was:

$$\text{for all } x,\; x^2 - 5x + 6 < 0 \Rightarrow x > 0.$$

Terms such as "for all x" are sometimes taken as understood in this context. We come back to notions such as "for all x" later in the course when we study formal logic. On the other hand, consider:

$$x^2+5x-6<0 \Rightarrow x>0.$$

With the same convention as above (i.e. we are asserting that this statement holds for all x), we see that this is false; we can have a value of x (for example, $x=0$) where we do have $x^2+5x-6<0$ but not $x>0$. We say that $x=0$ is a **counter-example** to the claim.

In general, a counter-example is sufficient to disprove a claim; for example, claiming that all British prime ministers have been male is seen to be false by citing the case of Margaret Thatcher. Note, however, that an example is not enough to prove a claim; we can't prove $x^2-5x+6<0 \Rightarrow x>0$ just by substituting in some particular value for x. Similarly we can't establish that all British Labour prime ministers have been male just by citing the example of Tony Blair.

This distinction between example and proof is very important. An analogy might help here. Suppose I had a program $\mathcal{P}$ which I claimed calculated the sum

$$1+2+3+\ldots+n$$

for any input n and I wanted to convince you that my program is correct. I run $\mathcal{P}$ with an input of 1 and $\mathcal{P}$ outputs 1 as expected; I run $\mathcal{P}$ with an input of 2 and $\mathcal{P}$ outputs 3 which was what was expected (since $1+2=3$); I then claim $\mathcal{P}$ is correct. However, you are (rightly) sceptical and you then run $\mathcal{P}$ with an input of 3; this time $\mathcal{P}$ outputs 5, which is not 1 + 2 + 3, and so you've proved that my program doesn't do what it claimed.

The point is that, if $\mathcal{P}$ genuinely calculated $1+2+3+\ldots+n$ for any input n, then it would work no matter what value of n we put into $\mathcal{P}$; the fact that it just happens to work for some values of n is not the point. It might be that $\mathcal{P}$ was the program

```
input (n);
output (2*n-1)
```

that, on an input n, outputs $2n-1$. It just so happens that $2n-1$ is the same as $1+2+3+\ldots+n$ if $n=1$ or $n=2$, but they aren't the same in general. A program such as

```
input (n);
sum := 0;
for counter := 1 to n do
   sum := sum + counter;
output (sum)
```

would output $1+2+3+\ldots+n$.

In general proving that programs do what they claim is not an easy task; we have to formulate some precise model of a program and then prove that the program does what we want. While this may seem easy in these examples, it is somewhat different when we have millions of lines of code!

Another common mistake is to try and prove a statement P by deducing a true statement from P. This doesn't work; for example, consider the following:

$$\begin{aligned} 1=0 &\Rightarrow 1=0 \text{ and } 0=1 \\ &\Rightarrow 1+0=0+1 \\ &\Rightarrow 1=1. \end{aligned}$$

The statement $1=1$ is obviously true, and so we have deduced a true statement from $1=0$; however this doesn't make the statement $1=0$ true!

When developing a theory, there are some general terms we use, and we'll try to give some idea of their meaning here. We will be discussing statements; such a statement may or may not be true, but it must be precise. In some cases, we will be able to prove that a statement is true (i.e. a rigorous argument that establishes the validity of the statement) and we will then have a **proof** of that statement. A statement that has been proved to be true is often referred to as a **theorem**.

1.2 Sets

When studying structures, we often consider collections of objects; such a collection is referred to as a **set**. Intuitively, a set is an unordered collection of **elements**. When representing a set, we often just enclose the elements of the set in curly brackets, such as

$$\{Sleepy, Happy, Grumpy, Sneezy, Bashful, Dopey, Doc\},$$
$$\{0, 1, 2, 3, 4, ...\},$$
$$\{blue, red, yellow, orange, violet, indigo, green\},$$
$$\{Pascal, C, Miranda, Prolog, Algol, Fortran, Basic, Java\}.$$

We write $x \in S$ if x is an element of the set S, and $x \notin S$ otherwise. For example, if

$$A = \{Pascal, C, Miranda, Prolog, Algol, Fortran, Basic, Java\}.$$

then $Pascal \in A$ but $Ada \notin A$.

We use $\emptyset$ to denote the **empty set**, i.e. the set with no elements (for example, the set of all living dodos). For any element x, we have that $x \notin \emptyset$. We let

- $\mathbb{N}$ denote the set of **natural numbers**, i.e. the set $\{0, 1, 2, 3, \ldots\}$ of all non-negative whole numbers,
- $\mathbb{Z}$ denote the set of **integers**, i.e. the set $\{\ldots, -2, -1, 0, 1, 2, 3, \ldots\}$ of all positive and negative whole numbers,
- $\mathbb{R}$ denote the set of **real numbers**, i.e. the set of all numbers given by decimal expansions,
- $\mathbb{B}$ denote the set $\{\texttt{true}, \texttt{false}\}$ of truth values; this is sometimes referred to as the set of **Boolean values** or **truth-values**.

The set of all elements x in a set S that satisfy a property $P(x)$ is denoted by $\{x \in S : P(x)\}$; for example, $\{x \in \mathbb{Z} : 2 \leqslant x \leqslant 5\} = \{2, 3, 4, 5\}$, and the set of even integers could be written as

$$\{x \in \mathbb{Z} : x = 2m \text{ for some } m \in \mathbb{Z}\}.$$

We sometimes just write $\{x : P(x)\}$ if the set S is understood from the context; for example, we could denote the set of even integers by

$$\{x : x = 2m \text{ for some } m \in \mathbb{Z}\}.$$

If sets A and B have exactly the same elements (i.e. if $x \in A \Leftrightarrow x \in B$), we say that A and B are **equal** and we write $A = B$; otherwise, we write $A \neq B$. For example, $\{1, 2\} = \{1, 2\} = \{2, 1\}$ but $\{1, 2\} \neq \{1, 3\}$ and $\{1, 2\} \neq \{1, 2, 3\}$.

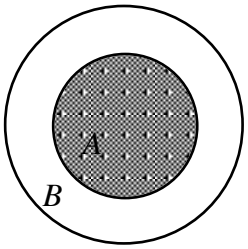


Figure 1.1: Subset

If all the elements of A are also elements of B (i.e. if we have $x \in A \Rightarrow x \in B$) then we say that A is a **subset** of B and write $A \subseteq B$; see Figure 1.1.

For example, $\{1\} \subseteq \{1,2\}$ and $\{1,2\} \subseteq \{1,2\}$. Another (equivalent) way of looking at this is to say that there is no element of B that is not in A; remember that the two statements

$$x \in A \Rightarrow x \in B \quad \text{and} \quad x \notin B \Rightarrow x \notin A$$

are equivalent by contraposition. Note also that $\emptyset \subseteq A$ for any set A.

We say that A is a **proper subset** of B if $A \subseteq B$ and $A \neq B$, and write $A \subset B$; for example, $\{1\} \subset \{1,2\}$.

One can prove various facts about sets; the following is just an example.

THEOREM. Let A and B be sets. Then $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.

PROOF. $\Rightarrow$) Suppose that $A = B$, i.e. $x \in A \Leftrightarrow x \in B$. Then we certainly have $x \in A \Rightarrow x \in B$ and $x \in B \Rightarrow x \in A$, so that $A \subseteq B$ and $B \subseteq A$.

$\Leftarrow$) Suppose that $A \subseteq B$ and $B \subseteq A$, i.e. $x \in A \Rightarrow x \in B$ and $x \in B \Rightarrow x \in A$. Then we certainly have that $x \in A \Leftrightarrow x \in B$, i.e. $A = B$. □

The following point is very important: distinguish between $\in$ and $\subseteq$; for example,

$$1 \in \mathbb{N}; \quad 1 \not\subseteq \mathbb{N}; \quad \{1\} \not\in \mathbb{N}; \quad \{1\} \subseteq \mathbb{N}.$$

We now introduce some operations on sets. If A and B are sets, then the **union** $A \cup B$ of A and B is the set whose elements are elements of A or of B (or both), and the **intersection** $A \cap B$ of A and B the set whose elements are elements of both A and B; more formally,

$$A \cup B = \{x : x \in A \text{ or } x \in B\}, \quad A \cap B = \{x : x \in A \text{ and } x \in B\};$$

see Figure 1.2. For example, $\{1,2\} \cup \{1,3\} = \{1,2,3\}$ and $\{1,2\} \cap \{1,3\} = \{1\}$.

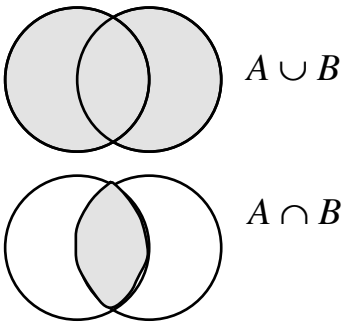


Figure 1.2: Union and intersection of two sets

Note that $A \cup B$ contains those elements in both A and B. If $A \cap B = \emptyset$, we say that A and B are **disjoint**.

We have various identities expressed in terms of $\cup$ and $\cap$; for example:

THEOREM. For any sets A, B and C, we have that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

PROOF. We can show that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ by showing that

$$x \in A \cap (B \cup C) \Leftrightarrow x \in (A \cap B) \cup (A \cap C).$$

Now:

$$\begin{aligned} x \in A \cap (B \cup C) &\Leftrightarrow x \in A \text{ and } x \in B \cup C \\ &\Leftrightarrow x \in A \text{ and } (x \in B \text{ or } x \in C) \\ &\Leftrightarrow (x \in A \text{ and } x \in B) \text{ or } (x \in A \text{ and } x \in C) \\ &\Leftrightarrow x \in A \cap B \text{ or } x \in A \cap C \\ &\Leftrightarrow x \in (A \cap B) \cup (A \cap C). \end{aligned}$$

This completes the proof □

In a similar way, we have the following:

THEOREM. For any sets A, B and C, we have that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

We now introduce another operation on sets: $A - B$ denotes the **difference** (or **complement** of B in A), i.e. the set $\{x \in A : x \notin B\}$ of elements that are in A but not in B; see Figure 1.3.

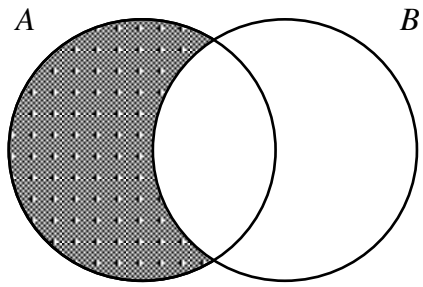


Figure 1.3: Difference of two sets

For example,

$$\{1,2\} - \{1,3\} = \{2\}; \quad \{1,2\} - \{1,2\} = \emptyset;$$
$$\{1,2\} - \{1,2,3\} = \emptyset; \quad \{1,2\} - \{3,4\} = \{1,2\}.$$

As another example, if A denoted the set of students doing module CO1011 and B the set of students doing module CO1003, then $A - B$ would denote the set of students doing CO1011 but not CO1003.

Now that we have introduced the operation $-$, there are further identities one can prove, such as the following:

THEOREM. For any sets A, B and C, we have that $A - (B \cap C) = (A - B) \cup (A - C)$.

PROOF. We have:

$$\begin{aligned} x \in A - (B \cap C) &\Leftrightarrow x \in A \text{ and } x \notin B \cap C \\ &\Leftrightarrow x \in A \text{ and } (x \notin B \text{ or } x \notin C) \\ &\Leftrightarrow (x \in A \text{ and } x \notin B) \text{ or } (x \in A \text{ and } x \notin C) \\ &\Leftrightarrow x \in A - B \text{ or } x \in A - C \\ &\Leftrightarrow x \in (A - B) \cup (A - C). \end{aligned}$$

This completes the proof. □

The equivalence of the statements

$$(x \notin B \cap C) \quad \text{and} \quad (x \notin B \text{ or } x \notin C)$$

may seem rather strange. Note that $x \in B \cap C$ means that x is in both B and C. So, for $x \notin B \cap C$, we have that it is not the case that x is in both B and C, i.e. that there is at least one of B and C that does not contain x. Another way of establishing the equivalence of these statements is to use truth-tables; we have mentioned these briefly and will come back to them later in the course.

In a similar way we have the following:

THEOREM. For any sets A, B and C, we have that $A - (B \cup C) = (A - B) \cap (A - C)$.

The identities in these last two theorems are known as **De Morgan's laws**.

If A is a set, the **powerset** $\wp(A)$ of A is defined to be the set of all subsets of A, i.e.

$$X \in \wp(A) \Leftrightarrow X \subseteq A.$$

For example, if $A = \{a, b\}$, then

$$\wp(A) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}.$$

If $A = \{a, b, c\}$, then

$$\wp(A) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{b,c\}, \{a,c\}, \{a,b,c\}\}.$$

Let's look at some more examples.

EXAMPLE. Note that $\emptyset$ is not the same as $\{\emptyset\}$; the first set has no elements and the second set has one element (namely $\emptyset$). In the same way that the subsets of $\{a\}$ are $\emptyset$ and $\{a\}$, we see that the subsets of $\{\emptyset\}$ are $\emptyset$ and $\{\emptyset\}$, so that $\wp(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$. □

EXAMPLE. Let A be the set of students at some university, and let E be the set of all the enrolments on the various modules taught by the university. Then we can consider each element of E as a subset of A (each enrolment being essentially a set of students), so that we could consider E as a subset of $\wp(A)$. We would then have that two students x and y are enrolled on a module in common if and only if there is an element e of E with $\{x, y\} \subseteq e$. (In Section 1.4 we will see a better way of modelling this situation.) □

For any set A, we let $|A|$ (or $\mathtt{card}\, A$) denote the **order** (or **cardinality**) of A, i.e. the number of elements in A; for example, if

$$A = \{Sleepy, Happy, Grumpy, Sneezy, Bashful, Dopey, Doc\},$$

then $|A| = 7$. We also have that $|\emptyset| = 0$. If $|A| = n$ for some natural number n, then we say that A is a **finite set**; otherwise A is an **infinite set**. For example, $\{Sleepy, Happy, Grumpy, Sneezy, Bashful, Dopey, Doc\}$ and $\mathbb{B}$ are finite sets whereas $\mathbb{N}$ and $\mathbb{Z}$ are infinite sets.

If A and B are disjoint sets, then we would have $|A \cup B| = |A| + |B|$, but this won't happen in general. So how is $|A \cup B|$ related to $|A|$ and $|B|$ in general?

Let $A \cap B = \{x_1, x_2, \ldots, x_n\}$, so that $|A \cap B| = n$, and let

$$A = \{x_1, x_2, \ldots, x_n, a_1, a_2, \ldots, a_p\},\ B = \{x_1, x_2, \ldots, x_n, b_1, b_2, \ldots, b_q\},$$

so that $|A| = n + p$, $|B| = n + q$. Then

$$A \cup B = \{x_1, x_2, \ldots, x_n, a_1, a_2, \ldots, a_p, b_1, b_2, \ldots, b_q\};$$

see Figure 1.4. So $|A \cup B| = n + p + q = (n + p) + (n + q) - n$, and hence we have

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

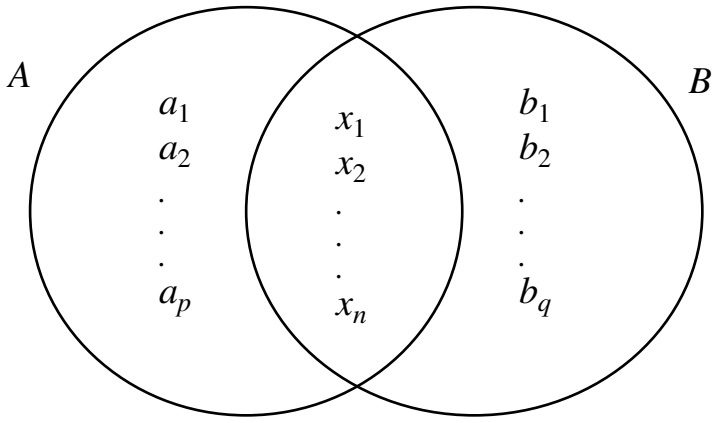


Figure 1.4: $|A \cup B| = n + p + q$

If A and B are disjoint, we have that $|A \cap B| = 0$, and hence that $|A \cup B| = |A| + |B|$ in that special case.

EXAMPLE. Suppose that we have 180 students taking module CO1003 and 150 students taking CO1011; in addition, we know that there are 100 students taking both modules. How many students are there taking at least one of the two modules?

Let A denote the set of students taking CO1003 and B denote the set of students taking CO1011, so that $|A| = 180$, $|B| = 150$ and $|A \cap B| = 100$. Then

$$\begin{aligned} \text{number students taking at least one module} &= |A \cup B| \\ &= |A| + |B| - |A \cap B| \\ &= 180 + 150 - 100 \\ &= 230. \end{aligned}$$

So there are 230 students taking at least one of the two modules. □

1.3 Relations

In many aspects of Computer Science, we are drawn to use the concept of a "relation". The idea of a relation is that it represents some property that may or may not hold between pairs of objects. For example, if we picked a person p and a dog d, then either p owns d or p does not own d; the relationship of ownership here holds between people and dogs.

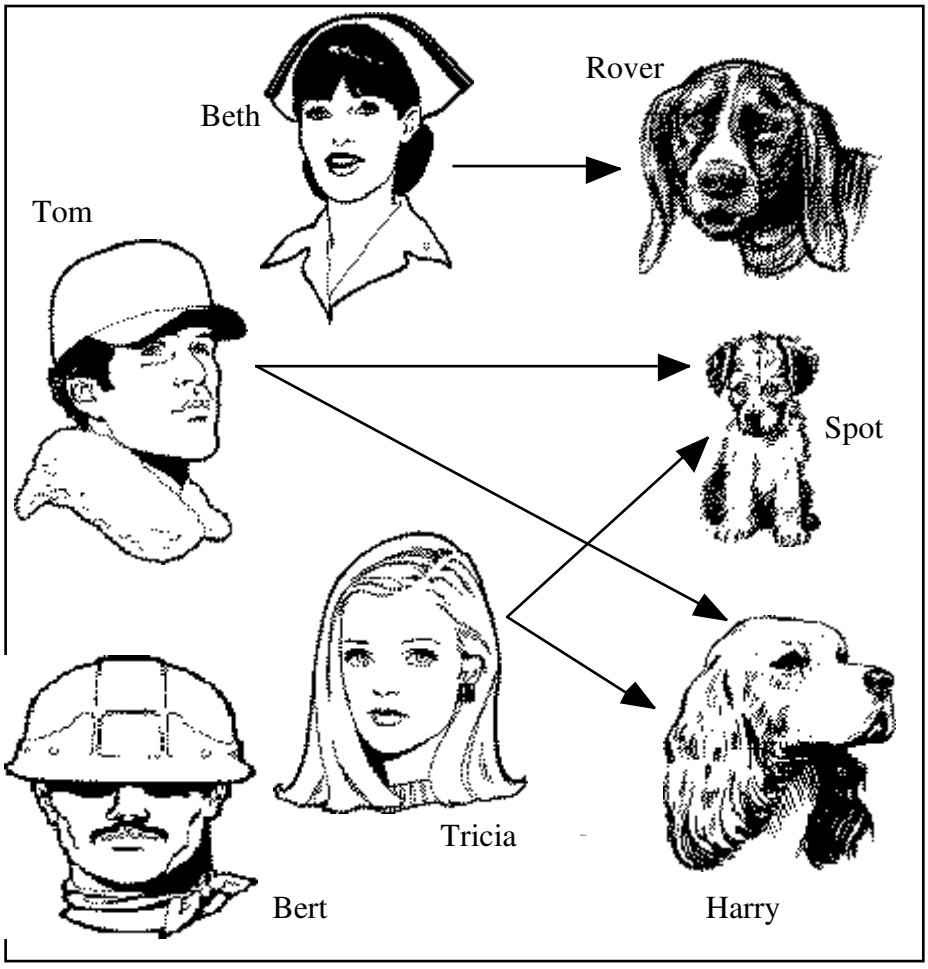


Figure 1.5: People owning dogs

In the example shown in Figure 1.5, we have a set $A = \{Beth, Tom, Tricia, Bert\}$ of people, a set $B = \{Rover, Spot, Harry\}$ of dogs, and we see that *Beth* owns *Rover*, that *Tom* owns *Spot* and *Harry*,

that *Tricia* also owns *Spot* and *Harry*, and that *Bert* does not own a dog. (We are allowing the possibility of joint ownership.) So we have described the relationship of ownership by saying who owns whom, i.e. by specifying a set of owner-dog pairs (p,d) such that p owns d:

$$(Beth,Rover),(Tom,Spot),(Tom,Harry),(Tricia,Spot),(Tricia,Harry).$$

We want to put this idea onto a more rigorous footing. We will first define the idea of "pairs of things" and then go on to describe relations.

Given two sets A and B, the **Cartesian product** $A\times B$ is the set

$$\{(a,b):a\in A,b\in B\},$$

where (a,b) is the **ordered pair** of the elements a and b, i.e. we are distinguishing between (a,b) and (b,a). For example, if $A=\{Tom,Jerry\}$ and B = $\{Cat,Mouse\}$, then

$$A\times B=\{(Tom,Cat),(Tom,Mouse),(Jerry,Cat),(Jerry,Mouse)\};$$

$$B\times A=\{(Cat,Tom),(Cat,Jerry),(Mouse,Tom),(Mouse,Jerry)\}.$$

For another example, if $A=\{1,2\}$ and $B=\{1,3\}$, then:

$$A\times B=\{(1,1),(1,3),(2,1),(2,3)\};\ \ B\times A=\{(1,1),(1,2),(3,1),(3,2)\}.$$

Note that we do not have $A\times B=B\times A$ in general. However, we do have that

$$|A\times B|=|A|\times|B|=|B|\times|A|=|B\times A|, \tag{1.3}$$

so that, if A and B are finite sets, then $A\times B$ and $B\times A$ have the same size. Note that, in (1.3), when we talk about $|A\times B|$, then "$\times$" is representing the Cartesian product of two sets, whereas, when we talk about $|A|\times|B|$, then "$\times$" is ordinary multiplication of numbers. As a particular instance of (1.3), we can take $B=\emptyset$, so that $|B|=0$ and

$$|A\times\emptyset|=|A|\times 0=0.$$

So $A\times\emptyset=\emptyset$. Another way of looking at this is to say that $A\times\emptyset$ is the set of all pairs (a,b) with $a\in A$ and $b\in\emptyset$ and, as there are no elements b with $b\in\emptyset$, then there are no such pairs (a,b). In a similar way, we have that $\emptyset\times A=\emptyset$.

When we move to more than two sets, we will let $A\times B\times C$ denote the set of all elements of the form

$$\{(a,b,c):a\in A,b\in B,c\in C\},$$

and then $A\times B\times C\times D$ the set of all elements of the form

$$\{(a,b,c,d):a\in A,b\in B,c\in C,d\in D\},$$

and so on. Note that, formally, $A\times B\times C$ is not the same as $(A\times B)\times C$, as the former has elements of the form (a,b,c), while the latter has elements of the form $((a,b),c)$.

EXAMPLE. Let $A=\{x\in\mathbb{N}:1\leqslant x\leqslant 31\}$, $B=\{January,February,March,\ldots,November,December\}$ and $C=\{x\in\mathbb{N}:2000\leqslant x\leqslant 2099\}$. Then every valid date this century is an element of $A\times B\times C$ (though not every element of $A\times B\times C$ is a valid date). □

EXAMPLE. In a relational database, information is stored in the form of tables; for example, in (a very simplified version of) a database on films, we might have a table like:

Film	**Star**	**Director**
MASH	Donald Sutherland	Robert Altmann
Raiders of the lost ark	Harrison Ford	Stephen Speilberg
The seventh seal	Max von Sydow	Ingmar Bergman
The good, the bad and the ugly	Clint Eastwood	Sergio Leone
Fargo	Frances McDormand	Joel Coen
Withnail and I	Richard E. Grant	Bruce Robinson
A beautiful mind	Russell Crowe	Ron Howard

If we let A denote the set of films, B the set of stars and C the set of directors, then every row of the database is an element of $A \times B \times C$. □

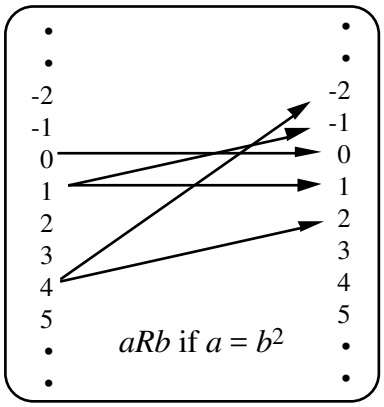


Figure 1.6: The relation $\{(b^2, b) : b \in \mathbb{Z}\}$

We now come onto the idea of a "relation". A **relation** R between sets A and B is a subset of $A \times B$; the idea here is that an element $a \in A$ may, or may not, be related to an element $b \in B$, so that R consists of just those pairs (a,b) such that a is related to b. We sometimes write aRb if $(a,b) \in R$. For example, if $R = \{(b^2, b) : b \in \mathbb{Z}\} \subseteq \mathbb{Z} \times \mathbb{Z}$, then $4R2$ and $4R-2$, but we do not have $2R4$ (see Figure 1.6).

For another example of a relation, see the people owning dogs example above, where

$$A = \{Beth, Tom, Tricia, Bert\}, \quad B = \{Rover, Spot, Harry\}, \quad \text{and}$$

$$R = \{(Beth, Rover), (Tom, Spot), (Tom, Harry), (Tricia, Spot), (Tricia, Harry)\} \subseteq A \times B.$$

For yet another example, let S be a set of students and M be the set of available modules. The relation R of "taking" can be described by the set of all pairs (s,m) where student s is taking module m. i.e.

$$R = \{(s,m) : \text{student } s \text{ is taking module } m\} \subseteq S \times M.$$

1.4 Functions

We now come to functions. Let A and B be sets; we will define a **function** f from A to B to be a relation between A and B such that, for each $a \in A$, there is exactly one $b \in B$ such that $(a,b) \in f$. The intuitive idea here is that, if the pair (a,b) is in f, then f assigns the value b to a and there is no ambiguity in this; we will write $f(a) = b$ or $f : a \mapsto b$ in this case. The pair (a,b) or $a \mapsto b$ is known as a **maplet**.

To be more specific, a **partial function** f from A to B is a subset of $A \times B$ such that, for each element $a \in A$, there is at most one $b \in B$ with $(a,b) \in f$, and a (**total**) **function** f from A to B is a subset of $A \times B$ such that, for each element $a \in A$, there is exactly one element $b \in B$ with $(a,b) \in f$; see Figure 1.7. We write $f(a) = b$ or $a \mapsto b$ if $(a,b) \in f$ in the case of a partial function as well.

If f is a (partial or total) function from A to B, then we write $f : A \to B$.

If f is a function from A to B, then we call A the **domain** $\texttt{dom}\ f$ of f and B the **codomain** of f. If f is not a total function, we let the domain of f be those elements in A for which f is actually defined.

If $X \subseteq A$, we often write $f(X)$ for the subset $\{f(x) : x \in X\}$ of B. The set $f(A) = \{f(a) : a \in A\}$ is called the **range** or **image** of f.

Let's look at some examples.

EXAMPLE. We define $f : \mathbb{R} \to \mathbb{R}$ by $f(x) = x^2$. Then f is a function, since $f(x)$ is uniquely determined for each $x \in \mathbb{R}$. The domain of f is $\mathbb{R}$, the codomain is $\mathbb{R}$ and the range is $\{x \in \mathbb{R} : x \geqslant 0\}$. □

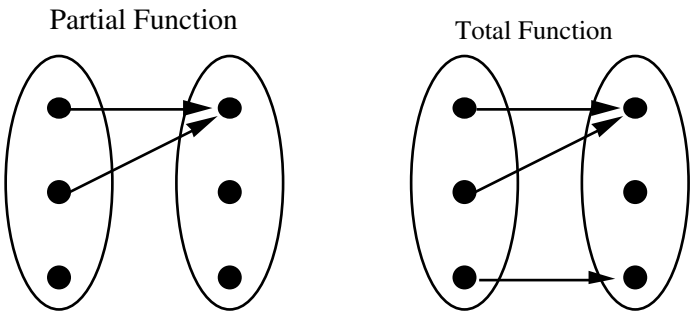


Figure 1.7: Partial and total functions

EXAMPLE. If we define $g : \mathbb{R} \to \mathbb{R}$ by $g(x) = \frac{1}{x}$, then g is a partial function from $\mathbb{R}$ to $\mathbb{R}$; if $x \neq 0$, then $g(x)$ exists and is uniquely defined, but, if $x = 0$, then $g(x)$ is not defined, so that g is not a total function.

The domain of g is $\mathbb{R} - \{0\}$ and the codomain is $\mathbb{R}$; the range is $\mathbb{R} - \{0\}$. □

EXAMPLE. If we define $h : \mathbb{R} \to \mathbb{R}$ by $h(x) = \sqrt{x}$, then h is not even a partial function from $\mathbb{R}$ to $\mathbb{R}$, as $h(x)$ is not uniquely defined for $x > 0$ (e.g. $\sqrt{4}$ could be 2 or -2).

If we take the convention that $\sqrt{x}$ is the positive square root of x, then h becomes a partial function. The domain of h is $\{x \in \mathbb{R} : x \geqslant 0\}$ and the range of h is also $\{x \in \mathbb{R} : x \geqslant 0\}$. □

EXAMPLE. Let $\mathcal{P}$ be a program with input from a set I and output from a set O. If $\mathcal{P}$ always terminates and gives an output that depends only on the input (i.e., if $\mathcal{P}$ is run twice with the same input, then it will necessarily compute the same output each time), then $\mathcal{P}$ computes a function from I to O. If we relax the condition that $\mathcal{P}$ always terminates (but keep the condition that the output depends only on the input), then $\mathcal{P}$ computes a partial function from I to O.

On the other hand, if the output is not uniquely determined by the input (i.e. if running $\mathcal{P}$ twice with the same input can produce different outputs), then $\mathcal{P}$ computes a relation between I and O. □

EXAMPLE. When we considered sets, we saw the following example. We let A be the set of students at some university, and we then let E be the set of enrolments on the various modules taught by the university. Each element of E is a subset of A, and we considered E as a subset of $\wp(A)$.

In some senses this is not a good model of the situation. For example, given an element of E, we do not know which module it represents the enrolment of; also, if two modules have precisely the same enrolment, then this will not be apparent in E, since we do not duplicate elements in a set.

A better model would be to consider a function f from M (the set of modules) to $\wp(A)$; in that case, we can retrieve the enrolment for a particular module m as it is just $f(m)$. Note that we can still tell if two students x and y are enrolled on a module in common; this will happen if and only if $\{x, y\} \subseteq f(m)$ for some module m. The set E of enrolments is now just the range of the function f. □

1.5 The Greek Alphabet

As a convenient source of extra symbols, we will ofter use Greek leters. For reference purposes, we list the Greek alphabet here.

alpha	A	α
beta	B	β
gamma	Γ	γ
delta	Δ	δ
epsilon	E	ε
zeta	Z	ζ
eta	H	η
theta	Θ	θ
iota	I	ι
kappa	K	κ
lambda	Λ	λ
mu	M	μ
nu	N	ν
xi	Ξ	ξ
omicron	O	o
pi	Π	π
rho	P	ρ
sigma	Σ	σ
tau	T	τ
upsilon	ϒ	υ
phi	Φ	φ
chi	X	χ
psi	Ψ	ψ
omega	Ω	ω

Chapter 2

Review: Propositional Logic

2.1 Introduction

When producing an argument, we wanted to assert that, whenever the assumptions were true, then the conclusion should be true also. This would be desirable in any valid argument; for example, consider:

All men are mortal.
Socrates is a man.
Therefore
Socrates is mortal.

Is this a valid argument? Yes it is, because, if the first two sentences are true, then the third sentence must also be true as well. However, the sentences

All men are mortal.
Socrates is mortal.

could be true, yet the sentence

Socrates is a man.

could be false; see Figure 2.1. If we are going to reason about computer systems, we clearly must start from sensible assumptions and then produce valid arguments; how can we tell if an argument is valid?

Figure 2.1: Socrates invalidates the argument!

For example, consider the following argument:

Either my program is correct or I have made a mistake.
My program is not correct.
Therefore I have made a mistake.

This is very similar in form to:

Either Michael is the lecturer or Rick is the lecturer.
Michael is not the lecturer.
Therefore Rick is the lecturer.

These two arguments have the same structure. If one of them is a valid argument, then surely the other one must be valid as well. Rather than examining every argument separately, we look at the form of the argument.

These two arguments both have the form

Either P or Q.
Not P.
Therefore Q.

We have presented the form of the argument by substituting letters such as P and Q for sentences; we call these letters **propositional letters**. We call the first sentences (Either P or Q, Not P) the **hypotheses** of the argument and the last sentence (Q) the **conclusion**.

Good arguments are those whose argument form is valid. By calling an argument form **valid** we mean that every instance of it which has true hypotheses has a true conclusion. For example,

Either P or Q.
Not P.
Therefore Q.

is a valid argument form. On the other hand,

Either not P or Q.
Not P.
Therefore Q.

is not a valid argument form; see the instance below!

Either my program does not work or unicorns exist.
My program does not work.
Therefore unicorns exist.

The system of logic produced by replacing sentences by propositional letters such as P and Q is called the **propositional calculus**.

We cannot capture all instances of valid argument forms in the propositional calculus. For example, consider our first example of a good argument:

All men are mortal.
Socrates is a man.
Therefore Socrates is mortal.

The propositional argument form would be "P, Q therefore R", which is invalid. A more powerful symbolism is the **predicate calculus**, which enables us to capture the validity of our argument about Socrates. We will come to the predicate calculus later in the course.

2.2 Connectives

In a natural language like English we use words like "and" to glue sentences together to form longer sentences:

The program compiles. The data is ready.

We can glue these sentences together with "and" to form

The program compiles and the data is ready.

We will introduce symbols for concepts such as this, and call such symbols **connectives**.

The symbol which we use to represent "and" in the propositional calculus is $\wedge$; "The program compiles and the data is ready" is symbolized by $P \wedge Q$. The proposition $P \wedge Q$ is known as the **conjunction** of P and Q.

The symbol $\vee$ is used to represent "or"; "The program compiles or the data is ready" has the form $P \vee Q$. The proposition $P \vee Q$ is known as the **disjunction** of P and Q.

In arguments we often use sentences whose form is conditional, such as "If the data is ready then we can run the program". To symbolize this we use the connective $\rightarrow$, which represents "implies". If P represents "The data is ready" and Q represents "We can run the program", then $P \rightarrow Q$ represents "If the data is ready then we can run the program".

The symbol $\rightarrow$ is very similar to $\Rightarrow$; we will use $\rightarrow$ when constructing sentences of formal logic and $\Rightarrow$ when constructing higher level arguments. Since every argument could, in principle, be reduced to a valid argument in formal logic, we can think of $\rightarrow$ as (among other things) being a representation in our formal logic of $\Rightarrow$.

Another connective is the **negation symbol** $\neg$; this takes a single sentence and turns it into another sentence, called the **negation** of the original sentence. For example, "The program does not compile" is the negation of "The program compiles". The sentence "The program does not compile" can therefore represented by $\neg P$.

If we use just the notation presented so far, some sentences are ambiguous. For example, what does the sentence $P \rightarrow Q \wedge R$ mean? We can use brackets to remove the ambiguity, writing

$$(P \rightarrow Q) \wedge R \quad \text{or} \quad P \rightarrow (Q \wedge R)$$

as appropriate. For example,

> if Leicester lose many matches then they will be relegated, and Leicester haven't got a strong side

is probably an example of $(P \rightarrow Q) \wedge R$, while

> if Leicester have a bad disciplinary record then they will be fined and some players will be suspended

is probably an example of $P \rightarrow (Q \wedge R)$.

Our last connective is $\leftrightarrow$ which has the same relationship to $\rightarrow$ that $\Leftrightarrow$ has to $\Rightarrow$; so $P \leftrightarrow Q$ is a shorthand for $(P \rightarrow Q) \wedge (Q \rightarrow P)$. So we can represent "I will win if and only if I score more points than my opponent" by $P \leftrightarrow Q$.

EXAMPLE. Let W stand for "I work hard", P for "I will pass the exam" and L for "I am lucky. Then we have the following:

English sentence	Representation
If I work hard then I will pass the exam	$W \rightarrow P$
If I don't work hard then I won't pass the exam	$\neg W \rightarrow \neg P$
Unless I work hard I won't pass the exam	$\neg W \rightarrow \neg P$
If I work hard and I am lucky then I will pass the exam	$W \wedge L \rightarrow P$
I am lucky and, if I work hard, then I will pass the exam	$L \wedge (W \rightarrow P)$
I will pass the exam only if I work hard	$P \rightarrow W$
I will pass the exam only if I work hard or I am lucky	$P \rightarrow W \vee L$
In order for me to pass the exam I must work hard	$P \rightarrow W$
If I fail the exam then I am unlucky	$\neg P \rightarrow \neg L$
I will pass the exam if and only if I work hard	$P \leftrightarrow W$

In some cases there are other representations than the ones we have given, but these are probably the most natural ones. □

We see that any sentence which is not a propositional letter is either of the form $\neg\varphi$ (where φ is some previously defined sentence) or else consists of two sentences joined by a connective ($\wedge$, $\vee$, $\rightarrow$ or $\leftrightarrow$). If the sentence is of the form $\neg\varphi$, we say that $\neg$ is the **main connective** of the sentence; if the sentence consists of two sentences joined by a connective ($\wedge$, $\vee$, $\rightarrow$ or $\leftrightarrow$), we say that this connective is the main connective of the sentence. For example, the main connective in $P \rightarrow (Q \wedge R)$ is $\rightarrow$.

Let's look at an example.

EXAMPLE. Let us consider the sentence

$$(P \wedge S) \rightarrow (Q \rightarrow (R \vee P)).$$

We can think of this sentence as being made up as follows. First, we take the propositional letters R and P and form the sentence $R \vee P$. We then take the propositional letter Q and the sentence $R \vee P$ and form $Q \rightarrow (R \vee P)$. Now we take the propositional letters P and S and form the sentence $P \wedge S$. Finally, we take the two sentences $P \wedge S$ and $Q \rightarrow (R \vee P)$ and form $(P \wedge S) \rightarrow (Q \rightarrow (R \vee P))$.

The main connective of this final sentence is the last connective we introduced, i.e. the first instance of $\rightarrow$ as we read the sentence $(P \wedge S) \rightarrow (Q \rightarrow (R \vee P))$ from left to right. □

Note that if we follow the above conventions strictly we would always have a lot of bracketting. While we must not introduce ambiguity, it would be nice to omit some brackets if we could. To cut down on the number of brackets we adopt a set of conventions to remove brackets where ambiguity can be avoided.

Our strategy is to rank the connectives in order of priority. The connective with the lowest ranking will be taken to be the main connective unless the brackets dictate otherwise.

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

We then have the following conventions:

- If $\varphi \rightarrow \psi$ is a sentence, and if the main connective in φ has higher ranking than $\rightarrow$, then we may omit the outermost brackets of φ, and similarly with ψ.
- We have a similar rule for $\neg$, $\leftrightarrow$, $\wedge$ and $\vee$.

Thus we read:

$$\neg P \rightarrow Q \vee R \quad \text{as} \quad (\neg P) \rightarrow (Q \vee R).$$

If we wanted the sentence $((\neg P) \rightarrow Q) \vee R$ we would have to leave in some brackets, and we would write $(\neg P \rightarrow Q) \vee R$. Some further examples:

$$\begin{array}{ccc} P \rightarrow Q \wedge (R \rightarrow S) & \text{means} & P \rightarrow (Q \wedge (R \rightarrow S)). \\ P \wedge \neg R \vee Q & \text{means} & (P \wedge (\neg R)) \vee Q. \\ R \rightarrow S \vee \neg Q & \text{means} & R \rightarrow (S \vee (\neg Q)). \\ Q \leftrightarrow R \vee S \rightarrow P & \text{means} & Q \leftrightarrow ((R \vee S) \rightarrow P). \end{array}$$

If there is any doubt, we can always leave the brackets in.

Another convention we will use here is that, if we have adjacent repeated instances of the same connective, then we will bracket from the left; so, for example,

$$\begin{array}{ccc} P \wedge Q \wedge R & \text{means} & (P \wedge Q) \wedge R; \\ P \rightarrow Q \rightarrow R & \text{means} & (P \rightarrow Q) \rightarrow R. \end{array}$$

Summary of the syntax of propositional logic

We call *syntax* the description of the set of allowed formulas. We can summarise the above as follows.

Syntax of propositional logic:

- $P, Q, \ldots$ are formulas (and they are called atomic propositions)
- If ϕ and ψ are formulas then the following are formulas:
  1. $\neg\phi$
  2. $\phi \wedge \psi$
  3. $\phi \vee \psi$
  4. $\phi \rightarrow \psi$
  5. $\phi \leftrightarrow \psi$

Question: Is this a precise definition of what a formula is? Is it precise enough?

Exercise: Take some formulas of your choice and transform them into trees (this is called *parsing*).

In computer science, we need to define new languages on a daily basis and it is common to abbreviate definitions of syntax like the one above as follows.

Syntax of propositional logic (short form):

$$Fma ::= P \mid Q \mid \neg Fma \mid Fma \wedge Fma \mid Fma \vee Fma \mid Fma \rightarrow Fma \mid Fma \leftrightarrow Fma$$

Both descriptions allow you to recognise whether a given expression is a formula, or also, to systematically generate all possible formulas.

Further reading for the curious: The above 'short form' is an instance of what is known as BNF, see the wikipedia article on Backus-Naur form.

2.3 Truth-tables

As we have said, we want to be able to distinguish valid arguments from invalid ones. While we can do this for small arguments by inspection, it becomes more difficult as the arguments become more complicated. To do this in general, we introduce the notion of **truth-values**.

The set $\mathbb{B}$ of truth-values has two members, true and false (usually represented as `true` and `false`). A **valuation** assigns a truth-value to each propositional letter (so we can think of a valuation as a function from propositional letters to truth-values).

Let's consider an example.

Propositional letters:	P, Q, R;
Valuation:	$(P, \texttt{true})$, $(Q, \texttt{true})$, $(R, \texttt{false})$.
	$P \mapsto \texttt{true}$;
	$Q \mapsto \texttt{true}$;
	$R \mapsto \texttt{false}$.

We see that the valuation simply assigns a truth-value to each propositional letter.

Simply mapping propositional letters onto truth-values is not sufficient; we want to extend this idea so

that every sentence is mapped onto a truth-value. To do this, we first define:

$$\begin{array}{llll} \texttt{true} \wedge \texttt{true} = \texttt{true}; & \texttt{true} \vee \texttt{true} = \texttt{true}; & \texttt{true} \to \texttt{true} = \texttt{true}; & \neg\texttt{true} = \texttt{false}; \\ \texttt{true} \wedge \texttt{false} = \texttt{false}; & \texttt{true} \vee \texttt{false} = \texttt{true}; & \texttt{true} \to \texttt{false} = \texttt{false}; & \neg\texttt{false} = \texttt{true}. \\ \texttt{false} \wedge \texttt{true} = \texttt{false}; & \texttt{false} \vee \texttt{true} = \texttt{true}; & \texttt{false} \to \texttt{true} = \texttt{true}; & \\ \texttt{false} \wedge \texttt{false} = \texttt{false}. & \texttt{false} \vee \texttt{false} = \texttt{false}. & \texttt{false} \to \texttt{false} = \texttt{true}. & \end{array}$$

The intuitive idea here is that, given a valuation v and a sentence φ, we know the truth-value of every propositional letter in φ under v, and we can then combine these values together, by the above rules, to get the truth-value $[\![\varphi]\!]_v$ of the sentence φ. When it is clear which valuation v is intended, we will just write $[\![\varphi]\!]$ as opposed to $[\![\varphi]\!]_v$.

We can sum up these rules in tabular form; for example, we have:

φ	ψ	$\varphi \wedge \psi$
true	true	true
true	false	false
false	true	false
false	false	false

This is known as the **truth-table** for $\wedge$. Provided that we know the truth-values assigned to φ and ψ under a valuation v, we can read off the truth-value of $\varphi \wedge \psi$ under this valuation. The truth-tables for the other connectives are:

φ	ψ	$\varphi \vee \psi$
true	true	true
true	false	true
false	true	true
false	false	false

φ	ψ	$\varphi \to \psi$
true	true	true
true	false	false
false	true	true
false	false	true

φ	$\neg\varphi$
true	false
false	true

These truth-tables enable us to determine the truth-values of sentences (under a given valuation) purely in terms of the truth-values of their constituents. We are interpreting the meaning of our connectives as functions from pairs of truth-values to truth-values (in the case of $\wedge$, $\vee$ and $\to$), and from truth-values to truth-values (in the case of $\neg$).

Note that our truth-table for $\vee$ gives that $\varphi \vee \psi$ is true if both φ is true and ψ is true. This is rather like the definition of union of sets; if we let P stand for $x \in A$ and Q stand for $x \in B$, then the condition $x \in A \cup B$ is equivalent to the sentence $P \vee Q$.

The truth-table for $\leftrightarrow$ can be derived from the truth-tables for $\wedge$ and $\to$ given that $\varphi \leftrightarrow \psi$ is a shorthand for $(\varphi \to \psi) \wedge (\psi \to \varphi)$:

φ	ψ	$\varphi \leftrightarrow \psi$
true	true	true
true	false	false
false	true	false
false	false	true

An assignment of truth-values to the propositional letters results in an assignment of truth-values to the sentence. For example, what is the truth-value of the sentence $P \to Q \vee R$ under the valuation $v(P) = \texttt{true}$, $v(Q) = \texttt{false}$, $v(R) = \texttt{true}$?

$$[\![Q \vee R]\!]_v = v(Q) \vee v(R) = \texttt{false} \vee \texttt{true} = \texttt{true};$$

$$[\![P \to Q \vee R]\!]_v = v(P) \to [\![Q \vee R]\!]_v = \texttt{true} \to \texttt{true} = \texttt{true}.$$

Using the idea of an assignment, we can give a definition of the notion of **validity**. If Γ is a set of sentences and ψ is a sentence, then we write $\Gamma \models \psi$ if every valuation that makes all the sentences in Γ true also makes ψ true. So, for example, we have

$$P \vee Q, \neg P \models Q.$$

Here Γ is the set of sentences $\{P \vee Q, \neg P\}$ and ψ is the sentence Q. In general, when we write $\Gamma \models \psi$, this indicates that the argument form with hypotheses Γ and conclusion ψ is valid.

Although Γ is a set of sentences, we'll remove the brackets $\{$ and $\}$ that normally enclose sets since there is no danger of confusion; in fact, we've already done this, since we wrote $P \vee Q, \neg P \models Q$ rather than $\{P \vee Q, \neg P\} \models Q$.

If $\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$, then $\Gamma \models \psi$ means that, whenever $\varphi_1, \varphi_2, \ldots, \varphi_n$ are all true, then ψ must be true, i.e., if $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$ is true, then ψ must be true as well. So another way of saying this is that $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n \rightarrow \psi$ must be true under all possible valuations.

If an argument form is not valid, we can show that it is not valid by producing a suitable valuation to serve as a counter-example.

EXAMPLE. For example, consider $P \rightarrow Q, \neg P \models \neg Q$. Let $v(P) = \texttt{false}$, $v(Q) = \texttt{true}$; then

$$[\![P \rightarrow Q]\!]_v = \texttt{true},\ [\![\neg P]\!]_v = \texttt{true},\ [\![\neg Q]\!]_v = \texttt{false}.$$

So, under this valuation v, the hypotheses are true but the conclusion is false, and thus we do not have $P \rightarrow Q, \neg P \models \neg Q$. □

We saw that we can determine the truth-value of any sentence φ under a valuation v by assigning truth-values to the propositional letters in φ via v and then using the truth-tables for the connectives in φ; we will now present a systematic technique for doing this.

EXAMPLE. For example, let's consider the sentence $\varphi = (P \rightarrow Q) \vee \neg R \rightarrow \neg\neg P$. Suppose that we are interested in its truth-value under the assignment v where $v(P) = \texttt{false}$, $v(Q) = \texttt{true}$ and $v(R) = \texttt{false}$. We first construct a one-row table as follows:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F										

Note that, for simplicity, we are just writing T for `true` and F for `false` in the table. We then enter the truth-values under the propositional letters in φ:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F	F		T			F				F

By the truth-table for $\neg$, $\neg R$ has the truth-value T, so we enter this in our table:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F	F		T		T	F				F

Likewise for $\neg\neg P$:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F	F		T		T	F		F	T	F

$P \rightarrow Q$ has truth-value F $\rightarrow$ T = T:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F	F	T	T		T	F		F	T	F

Since $\vee$ has precedence over $\rightarrow$, we now note that, as $P \rightarrow Q$ has truth-value T and $\neg R$ has truth-value T, we have that $(P \rightarrow Q) \vee \neg R$ has truth-value T $\vee$ T = T:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F	F	T	T	T	T	F		F	T	F

The last entry to fill in is that under the remaining $\rightarrow$ which is the main connective; now we have that $(P \rightarrow Q) \vee \neg R$ has truth-value T, $\neg\neg P$ has truth-value F, and so $\varphi = (P \rightarrow Q) \vee \neg R \rightarrow \neg\neg P$ has truth-value T $\rightarrow$ F = F:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F	F	T	T	T	T	F	F	F	T	F

We see that, by successive applications of the definitions of the connectives we finish up with a table where every propositional letter and connective in φ has a truth-value entered under it. The value under the main connective (which is always the last one to be entered) is then the truth-value of the sentence φ under the valuation v.

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
F	T	F	F	T	T	T	T	F	**F**	F	T	F
									$\uparrow$			

The sentence $(P \rightarrow Q) \vee \neg R \rightarrow \neg\neg P$ is therefore false under this particular valuation. □

If a sentence contains n propositional letters, then there are

$$\overbrace{2 \times 2 \times \ldots \times 2}^{n} = 2^n$$

possible assignments of truth-values to those letters. In the case of

$$(P \rightarrow Q) \vee \neg R \rightarrow \neg\neg P$$

for example, there are $2^3 = 8$ possible assignments. So we can construct a table with 2^n rows that will give the truth-value of the sentence under all possible assignments.

If a sentence comes out true under all valuations we call it a **tautology**; if a sentence comes out false under all valuations we call it a **contradiction** (recall the idea of "proof by contradiction"); otherwise (if the sentence is true under some valuation and false under another), we call it **contingent**.

For example, the sentence $(P \rightarrow Q) \vee \neg R \rightarrow \neg\neg P$ we considered above is contingent:

P	Q	R	$(P$	$\rightarrow$	$Q)$	$\vee$	$\neg$	R	$\rightarrow$	$\neg$	$\neg$	P
T	T	T	T	T	T	T	F	T	**T**	T	F	T
T	T	F	T	T	T	T	T	F	**T**	T	F	T
T	F	T	T	F	F	F	F	T	**T**	T	F	T
T	F	F	T	F	F	T	T	F	**T**	T	F	T
F	T	T	F	T	T	T	F	T	**F**	F	T	F
F	T	F	F	T	T	T	T	F	**F**	F	T	F
F	F	T	F	T	F	T	F	T	**F**	F	T	F
F	F	F	F	T	F	T	T	F	**F**	F	T	F

An easy example of a tautology is $P \vee \neg P$, and of a contradiction $P \wedge \neg P$:

P	P	$\vee$	$\neg$	P
T	T	**T**	F	T
F	F	**T**	T	F

P	P	$\wedge$	$\neg$	P
T	T	**F**	F	T
F	F	**F**	T	F

Note that asserting $\models \varphi$ is another way of saying that φ is a tautology and asserting that $\models \neg\varphi$ is another way of saying that φ is a contradiction.

2.4 Testing for validity

Suppose we want to test whether or not $\Gamma \models \psi$. We have seen that each row of a truth-table represents the value of a sentence under a certain valuation. By extending this idea, we can produce a table each of whose rows tells us the value of each sentence in Γ and the value of ψ under a certain valuation.

For example, consider $P \to Q, P \models Q$:

P	Q	P	$\to$	Q	P	$\models$	Q
T	T	T	**T**	T	**T**		**T**

This row tells us the value of each sentence under the valuation shown. Now consider the complete truth-table:

P	Q	P	$\to$	Q	P	$\models$	Q
T	T	T	**T**	T	**T**		**T**
T	F	T	**F**	F	**T**		**F**
F	T	F	**T**	T	**F**		**T**
F	F	F	**T**	F	**F**		**F**

For the argument to be valid, we need that, whenever the hypotheses $P \to Q$ and P are both true, then the conclusion Q is also true. So we look at all the rows in which all the hypotheses come out true; provided that the conclusion comes out true in every such row, we know that $P \to Q, P \models Q$.

In our case, the only row where the hypotheses are both true is the first one, and the conclusion is true there as well. Hence we do have $P \to Q, P \models Q$.

Let's look at some further examples. We do have $P \vee Q, \neg P \models Q$:

P	Q	P	$\vee$	Q	$\neg$	P	$\models$	Q
T	T	T	**T**	T	**F**	T		**T**
T	F	T	**T**	F	**F**	T		**F**
F	T	F	**T**	T	**T**	F		**T**
F	F	F	**F**	F	**T**	F		**F**

The only row where the hypotheses are both true is the third, and the conclusion is true there also.

However, we do not have $P \to Q, \neg P \models Q$:

P	Q	P	$\to$	Q	$\neg$	P	$\models$	Q
T	T	T	**T**	T	**F**	T		**T**
T	F	T	**F**	F	**F**	T		**F**
F	T	F	**T**	T	**T**	F		**T**
F	F	F	**T**	F	**T**	F		**F**

The last row represents a valuation where the hypotheses are both true but the conclusion is false.

There is an important point to notice here. The existence of a row (the third row) in this last example where the hypotheses are both true and the conclusion is also true is irrelevant; the point is that, whenever the hypotheses are true, then the conclusion must also be true; the argument is invalid if there is an instance (such as the fourth row here) where the hypotheses are true and the conclusion is false.

Let us look at one more example. Consider the following argument:

> If the data was accurate then the program has run and the answer is correct.
> If the program has run then we did not have a compilation error.
> We either had a compilation error or the answer is not correct.
> Therefore the data was not accurate.

Is this reasoning valid? Let

D represent "The data was accurate", R represent "The program has run",
A represent "The answer is correct" and C represent "We had a compilation error".

Then our argument becomes:

$$D \to R \wedge A,\ R \to \neg C,\ C \vee \neg A \models \neg D.$$

We test this by means of a (rather large) truth-table:

	D	R	A	C	D	$\to$	R	$\wedge$	A	R	$\to$	$\neg$	C	C	$\vee$	$\neg$	A	$\models$	$\neg$	D
1	T	T	T	T	T	**T**	T	T	T	T	**F**	F	T	T	**T**	F	T		**F**	T
2	T	T	T	F	T	**T**	T	T	T	T	**T**	T	F	F	**F**	F	T		**F**	T
3	T	T	F	T	T	**F**	T	F	F	T	**F**	F	T	T	**T**	T	F		**F**	T
4	T	T	F	F	T	**F**	T	F	F	T	**T**	T	F	F	**T**	T	F		**F**	T
5	T	F	T	T	T	**F**	F	F	T	F	**T**	F	T	T	**T**	F	T		**F**	T
6	T	F	T	F	T	**F**	F	F	T	F	**T**	T	F	F	**F**	F	T		**F**	T
7	T	F	F	T	T	**F**	F	F	F	F	**T**	F	T	T	**T**	T	F		**F**	T
8	T	F	F	F	T	**F**	F	F	F	F	**T**	T	F	F	**T**	T	F		**F**	T
9	F	T	T	T	F	**T**	T	T	T	T	**F**	F	T	T	**T**	F	T		**T**	F
10	F	T	T	F	F	**T**	T	T	T	T	**T**	T	F	F	**F**	F	T		**T**	F
11	F	T	F	T	F	**T**	T	F	F	T	**F**	F	T	T	**T**	T	F		**T**	F
12	F	T	F	F	F	**T**	T	F	F	T	**T**	T	F	F	**T**	T	F		**T**	F
13	F	F	T	T	F	**T**	F	F	T	F	**T**	F	T	T	**T**	F	T		**T**	F
14	F	F	T	F	F	**T**	F	F	T	F	**T**	T	F	F	**F**	F	T		**T**	F
15	F	F	F	T	F	**T**	F	F	F	F	**T**	F	T	T	**T**	T	F		**T**	F
16	F	F	F	F	F	**T**	F	F	F	F	**T**	T	F	F	**T**	T	F		**T**	F

We see that the only lines where the hypotheses are all true are 12, 13, 15 and 16, and the conclusion is true in all those cases; so the argument is valid.

2.5 Equivalence

Consider the truth-tables for $\neg(P \vee Q)$ and $\neg P \wedge \neg Q$ shown below:

P	Q	$\neg$	$(P$	$\vee$	$Q)$
T	T	**F**	T	T	T
T	F	**F**	T	T	F
F	T	**F**	F	T	T
F	F	**T**	F	F	F

P	Q	$\neg$	P	$\wedge$	$\neg$	Q
T	T	F	T	**F**	F	T
T	F	F	T	**F**	T	F
F	T	T	F	**F**	F	T
F	F	T	F	**T**	T	F

We see that the resulting truth-values are the same, i.e. that, whatever truth-values we assign to P and Q, the sentences $\neg(P \vee Q)$ and $\neg P \wedge \neg Q$ will have the same truth-value. We say that the sentences $\neg(P \vee Q)$ and $\neg P \wedge \neg Q$ are **equivalent**.

In general, we will write $\varphi \equiv \psi$ if the sentences φ and ψ are equivalent, i.e. if φ and ψ have the same truth-values under all possible valuations. So we have

$$\neg(P \vee Q) \ \equiv\ \neg P \wedge \neg Q.$$

In general, saying that φ and ψ are equivalent is the same as saying that $\varphi \leftrightarrow \psi$ is a tautology or, to put this another way, that we have $\models \varphi \leftrightarrow \psi$.

In a similar way, we can show that $\neg(P \wedge Q) \ \equiv\ \neg P \vee \neg Q$. So, for example, if P represented the assertion $x > 1$ and Q the assertion $x < 3$, then $\neg P$ represents $x \leqslant 1$ and $\neg Q$ represents $x \geqslant 3$. So, for example, the code

`IF NOT((x > 1) AND (x < 3)) THEN ...`

could also be written as

`IF (x <= 1) OR (x >= 3) THEN ...`

When proving the set theoretic equality

$$A-(B\cap C)=(A-B)\cup(A-C), \tag{2.1}$$

i.e. one of De Morgan's laws, we used the fact that the two statements

$$x\notin B\cap C \text{ and } (x\notin B \text{ or } x\notin C)$$

were equivalent. If we let P denote the statement $x\in B$ and Q denote $x\in C$, then $P\wedge Q$ denotes $x\in B\cap C$, so that $\neg(P\wedge Q)$ denotes $x\notin B\cap C$; similarly, $\neg P\vee\neg Q$ denotes $x\notin B$ or $x\notin C$. The equivalence of the statements $x\notin B\cap C$ and ($x\notin B$ or $x\notin C$) is another example of the logical equivalence of $\neg(P\wedge Q)$ and $\neg P\vee\neg Q$. So, in a sense, the set-theoretic equality (2.1) is essentially just an expression in set theory of the logical equivalence of $\neg(P\wedge Q)$ and $\neg P\vee\neg Q$. The facts that $\neg(P\wedge Q)$ and $\neg P\vee\neg Q$ are equivalent, as are $\neg(P\vee Q)$ and $\neg P\wedge\neg Q$, are also known as **De Morgan's laws**.

We also have that $P\to Q$ and $\neg Q\to\neg P$ are equivalent:

P	Q	P	$\to$	Q
T	T	T	**T**	T
T	F	T	**F**	F
F	T	F	**T**	T
F	F	F	**T**	F

P	Q	$\neg$	Q	$\to$	$\neg$	P
T	T	F	T	**T**	F	T
T	F	T	F	**F**	F	T
F	T	F	T	**T**	T	F
F	F	T	F	**T**	T	F

This is why "proof by contraposition" works; proving $P\to Q$ is equivalent to proving $\neg Q\to\neg P$.

As far as "proof by contradiction" is concerned, we said that proving $P\to Q$ was equivalent to proving that one could deduce some contradiction, φ say, from the assumptions P and $\neg Q$, i.e. that we have $P\wedge\neg Q\to\varphi$ for some contradiction φ. This equivalence can be seen in the following truth-tables:

P	Q	P	$\to$	Q
T	T	T	**T**	T
T	F	T	**F**	F
F	T	F	**T**	T
F	F	F	**T**	F

P	Q	P	$\wedge$	$\neg$	Q	$\to$	φ
T	T	T	F	F	T	**T**	F
T	F	T	T	T	F	**F**	F
F	T	F	F	F	T	**T**	F
F	F	F	F	T	F	**T**	F

Note that, as φ is a contradiction, the truth-value of φ is always false in the second table.

2.6 Interdefinability

We will now show that, as far as the truth-theoretic interpretation of the connectives is concerned, we do not, in fact, need all the connectives, as some can be defined (up to equivalence) in terms of others. For example, consider the following truth-tables:

φ	ψ	φ	$\wedge$	ψ
T	T	T	**T**	T
T	F	T	**F**	F
F	T	F	**F**	T
F	F	F	**F**	F

φ	ψ	$\neg$	$(\neg$	φ	$\vee$	$\neg$	$\psi)$
T	T	**T**	F	T	F	F	T
T	F	**F**	F	T	T	T	F
F	T	**F**	T	F	T	F	T
F	F	**F**	T	F	T	T	F

We see that $\varphi\wedge\psi$ is equivalent to $\neg(\neg\varphi\vee\neg\psi)$; so, as far as truth-values are concerned, one could define $\wedge$ in terms of $\vee$ and $\neg$, i.e. we have

- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$;

So we could omit $\wedge$ and define everything in terms of $\vee$, $\neg$ and $\rightarrow$. However, we also have:

- $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$;

so every sentence is equivalent to one that only uses the connectives $\vee$ and $\neg$.

In fact one can easily check that the following equivalences also hold:

- $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$;
- $\varphi \rightarrow \psi \equiv \neg(\varphi \wedge \neg\psi)$;

so every sentence is equivalent to one that only uses the connectives $\wedge$ and $\neg$. Lastly, we have:

- $\varphi \wedge \psi \equiv \neg(\varphi \rightarrow \neg\psi)$;
- $\varphi \vee \psi \equiv \neg\varphi \rightarrow \psi$;

so every sentence is equivalent to one that only uses the connectives $\rightarrow$ and $\neg$. (Note that we have not discussed defining $\leftrightarrow$ explicitly, but we already know that $\leftrightarrow$ can be defined in terms of the other connectives.)

In fact we can introduce a new two-place connective which will be sufficient on its own to express every sentence of the propositional calculus (up to equivalence). Let $\downarrow$ be the connective defined by the following truth-table:

φ	ψ	$\varphi \downarrow \psi$
T	T	F
T	F	F
F	T	F
F	F	T

Informally, a sentence involving $\downarrow$ is true when neither of the arguments are true, i.e. $\varphi \downarrow \psi$ is equivalent to $\neg(\varphi \vee \psi)$, which is equivalent to $\neg\varphi \wedge \neg\psi$:

$$\begin{aligned} \varphi \downarrow \psi &\equiv \neg(\varphi \vee \psi) && (2.2)\\ &\equiv \neg\varphi \wedge \neg\psi. && (2.3) \end{aligned}$$

First, we see that

$$\neg\varphi \equiv \neg(\varphi \vee \varphi) \equiv \varphi \downarrow \varphi \quad \text{by (2.2).} \tag{2.4}$$

Using the equivalences noted above we then have the following:

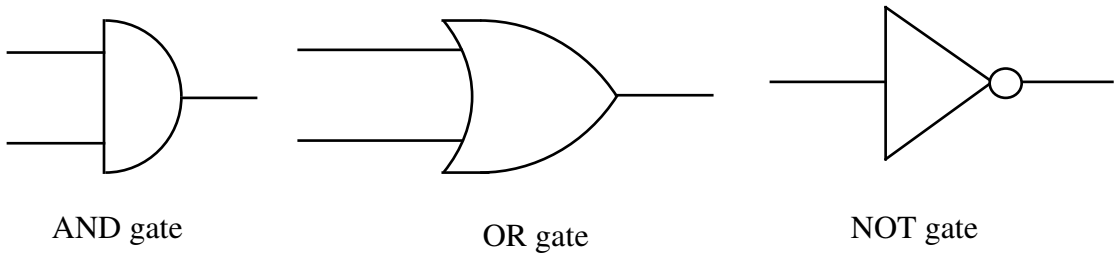
$$\begin{aligned} \varphi \vee \psi &\equiv \neg(\neg\varphi \wedge \neg\psi) \\ &\equiv \neg(\varphi \downarrow \psi) && \text{by (2.3)} \\ &\equiv (\varphi \downarrow \psi) \downarrow (\varphi \downarrow \psi) && \text{by (2.4),} \end{aligned}$$

$$\begin{aligned} \varphi \wedge \psi &\equiv \neg\neg\varphi \wedge \neg\neg\psi \\ &\equiv (\neg\varphi) \downarrow (\neg\psi) && \text{by (2.3)} \\ &\equiv (\varphi \downarrow \varphi) \downarrow (\psi \downarrow \psi) && \text{by (2.4),} \end{aligned}$$

$$\begin{aligned} \varphi \rightarrow \psi &\equiv \neg\varphi \vee \psi \\ &\equiv (\varphi \downarrow \varphi) \vee \psi && \text{by (2.4)} \\ &\equiv ((\varphi \downarrow \varphi) \downarrow \psi) \downarrow ((\varphi \downarrow \varphi) \downarrow \psi). \end{aligned}$$

While the above equivalences are clearly of theoretical interest, they are also of practical value in the design of electronic circuits; we will not go into a proper study of electronics here, but just touch on a few points to bring out the connection.

We show three basic circuit components in Figure 2.2; the AND gate, the OR gate and the NOT gate. These work essentially the same as the connectives $\wedge$, $\vee$ and $\neg$ in propositional logic; we have values 1 and 0 (for "on" and "off", corresponding to `true` and `false`), and the truth-tables for the three gates are then the same as the tables for the corresponding connectives.



	0	1
0	0	0
1	0	1

	0	1
0	0	1
1	1	1

0	1
1	0

Figure 2.2: Some basic gates

We saw that all propositional sentences are equivalent to one using just the single connective $\downarrow$ introduced above. Since (as we saw above) $\varphi \downarrow \psi$ is equivalent to $\neg(\varphi \vee \psi)$, we give the name "NOR" (for NOT-OR) to the corresponding gate; see Figure 2.3.

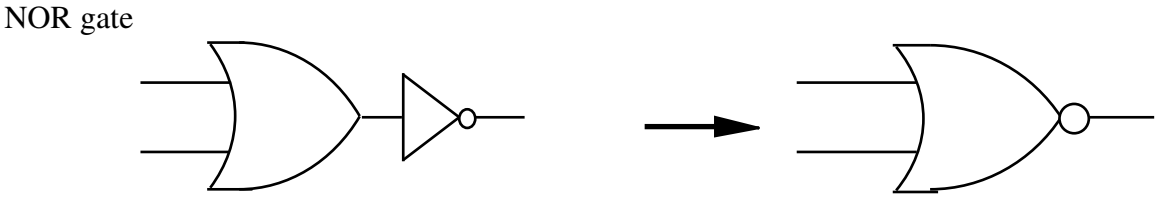


Figure 2.3: NOR gate

There is, in fact, just one more connective which is sufficient (by itself) to generate all sentences of propositional logic up to equivalence; if we call this connective $|$, then $\varphi|\psi$ is equivalent to $\neg(\varphi \wedge \psi)$. We call the corresponding gate the "NAND" gate; see Figure 2.4.

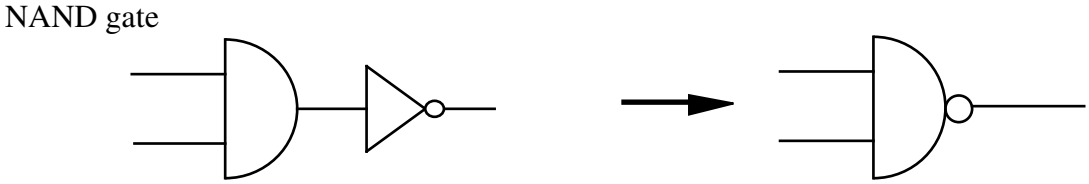


Figure 2.4: NAND gate

Chapter 3

Propositional Tableaux

We know that a truth-table has 2^n rows, where n is the number of propositional letters in the formula being tested. It follows that the truth-table method of testing for validity is extremely cumbersome if we have more than about three letters or so.

$$(P \to Q) \wedge (R \to S) \vee ((U \to \neg V) \to W) : \ 2^7 = 128 \text{ rows!}$$

Figure 3.1: Too many rows in the table!

We look at a potentially more efficient method of testing for validity, known as the **tableaux method**.

Suppose that we wish to test whether or not we have

$$\varphi_1, \varphi_2, \ldots, \varphi_n \models \psi$$

for some sentences $\varphi_1, \varphi_2, \ldots, \varphi_n$ and ψ. This will hold if, whenever $\varphi_1, \varphi_2, \ldots, \varphi_n$ are all true under a valuation, then ψ is also true under that valuation. So we do not have $\varphi_1, \varphi_2, \ldots, \varphi_n \models \psi$ if and only if there is a valuation v which makes $\varphi_1, \varphi_2, \ldots, \varphi_n$ all true but ψ false. Equivalently, we do not have $\varphi_1, \varphi_2, \ldots, \varphi_n \models \psi$ if and only if there is a valuation v which makes $\varphi_1, \varphi_2, \ldots, \varphi_n, \neg\psi$ all true.

Given this, we say that a set of sentences is **satisfiable** if and only if there is some valuation v such that, for every φ in the set, we have $[\![\varphi]\!]_v = \texttt{true}$. Put in this way, we do have that $\varphi_1, \varphi_2, \ldots, \varphi_n \models \psi$ if and only if the set $\{\varphi_1, \varphi_2, \ldots, \varphi_n, \neg\psi\}$ of sentences is not satisfiable.

Tableaux provide a procedure which enables us to determine whether or not a set of propositional sentences is satisfiable. When there is a suitable valuation, i.e. the set is satisfiable, then the procedure finds it; otherwise (when the set is not satisfiable) the procedure halts without success, which tells us that there is no such valuation.

Let us consider an example. To see whether or not

$$P, P \to Q \models Q,$$

we use the tableaux method to determine whether or not $\{P, P \to Q, \neg Q\}$ is satisfiable. If $\{P, P \to Q, \neg Q\}$ is satisfiable, then we do not have $P, P \to Q \models Q$; otherwise we do. (In fact it will turn that $P, P \to Q \models Q$ does hold).

Our first step in constructing a tableau for a set of sentences is to write down all the sentences, arranged in a vertical column; we call these the **initial sentences** of the tableau. In the example we have just considered, we have:

$$\begin{array}{c} P \\ P \to Q \\ \neg Q \end{array}$$

Now we know that any valuation under which $P \to Q$ is true must be either a valuation under which $\neg P$ is true or a valuation under which Q is true. We therefore use the picture below to indicate the two branching possibilities for a valuation for our set of formulae:

$$\begin{array}{ccc} & P & \\ & P \to Q & \\ & \neg Q & \\ / & & \backslash \\ \neg P & & Q \end{array}$$

What we mean by this picture is that any valuation which makes $\{P, P \to Q, \neg Q\}$ true must either make $\neg P$ true or Q true (or both true) as well.

When we have applied a tableau rule to a sentence on a vertex, we mark this fact by placing a tick against it; we say that the vertex has been **ticked**:

$$\begin{array}{ccc} & P & \\ & P \to Q \ \surd & \\ & \neg Q & \\ / & & \backslash \\ \neg P & & Q \end{array}$$

In this example, there are contradictory sentences on both branches; on the left-hand branch we have both P and $\neg P$, on the right-hand branch both Q and $\neg Q$. However, it is clear that no set of sentences containing contradictory sentences can be satisfied. We say that the resulting tableau is a **closed tableau** for $\{P, P \to Q, \neg Q\}$. We deduce that the initial set of sentences is not satisfiable, and hence that we do have $P, P \to Q \models Q$.

Let us consider another example: is it the case that $Q \models P \to Q$? To answer this question, we test the set of formulae $\{Q, \neg(P \to Q)\}$.

We know that every valuation v with $[\![\neg(P \to Q)]\!]_v = \texttt{true}$ is a valuation under which P is true and Q is false. We therefore complete the tableau as shown below:

$$\begin{array}{c} Q \\ \neg(P \to Q) \ \surd \\ P \\ \neg Q \end{array}$$

Since there is only one branch, and Q and $\neg Q$ are both on it, the initial set is unsatisfiable. So we do have that $Q \models P \to Q$.

Let us consider a slightly more complicated example: do we have that

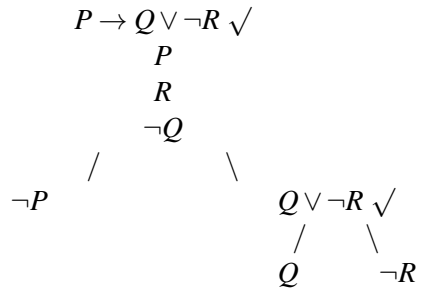
$$P \to Q \vee \neg R, P, R \models Q?$$

To determine this, we need to test the set of formulae $\{P \to Q \vee \neg R, P, R, \neg Q\}$. Let's prepare the rules we need in advance before constructing the tableau.

We already know how to handle $\to$. In the case of $\vee$, observe that any valuation v with $[\![\varphi \vee \psi]\!]_v = \texttt{true}$ must be such that either $[\![\varphi]\!]_v = \texttt{true}$ or else $[\![\psi]\!]_v = \texttt{true}$ (or both). We can represent this rule for constructing tableaux like this:

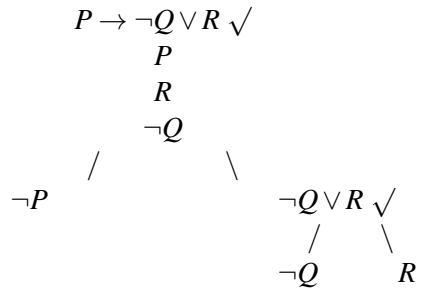
$$\begin{array}{ccc} & P \vee Q & \\ / & & \backslash \\ P & & Q \end{array}$$

We now have



Every branch contains a contradiction, so we know that the initial set of sentences is unsatisfiable, and hence that we do have $P \to Q \vee \neg R,\ P,\ R \models Q$.

Let's consider another example; what about $P \to \neg Q \vee R,\ P,\ R \models Q$? This time we have



We see that the two right hand branches do not contain contradictions. Since we cannot apply any more rules, the original set of sentences is satisfiable, and we do not have $P \to \neg Q \vee R,\ P,\ R \models Q$.

In fact, we can read off a valuation v from the tableau which makes the hypotheses true and the conclusion false; we see that we must have P, $\neg Q$ and R true, i.e. $v(P) = v(R) = \texttt{true}$ and $v(Q) = \texttt{false}$. With this valuation, $P \to \neg Q \vee R$, P and R are true, and Q is false, so that the argument form is invalid.

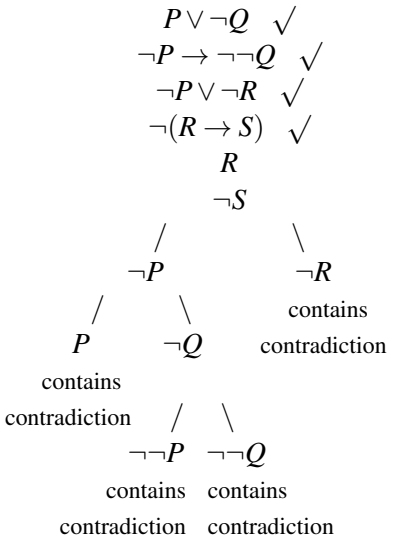
We now give a complete set of tableau rules for the connectives; you should confirm they are correct by checking them against the corresponding truth-tables.

$\wedge$	$\neg\wedge$	$\vee$	$\neg\vee$	$\to$	$\neg\to$	$\neg\neg$
$\varphi \wedge \psi$	$\neg(\varphi \wedge \psi)$	$\varphi \vee \psi$	$\neg(\varphi \vee \psi)$	$\varphi \to \psi$	$\neg(\varphi \to \psi)$	$\neg\neg\varphi$
φ	/ \	/ \	$\neg\varphi$	/ \	φ	φ
ψ	$\neg\varphi$ $\neg\psi$	φ ψ	$\neg\psi$	$\neg\varphi$ ψ	$\neg\psi$	

Let's look at one more example before continuing; how about

$$P \vee \neg Q, \neg P \to \neg\neg Q, \neg P \vee \neg R \models R \to S?$$

Here we have the tableau



We see that every branch contains a contradiction, so that we do have

$$P \vee \neg Q, \neg P \rightarrow \neg\neg Q, \neg P \vee \neg R \models R \rightarrow S.$$

We can simplify the presentation of the rules by observing that every rule either branches into two or does not branch. The formulae whose rule does not branch (except the ones which are double negated) we call **conjunctive** or **α-formulae**, and those which do branch **disjunctive** or **β-formulae**. Furthermore, in the case of each α-formula (or β-formula), we identify two components, called α_1 and α_2 (or β_1 and β_2 respectively):

Conjunctive			Disjunctive		
α	α_1	α_2	β	β_1	β_2
$\varphi \wedge \psi$	φ	ψ	$\varphi \vee \psi$	φ	ψ
$\neg(\varphi \vee \psi)$	$\neg\varphi$	$\neg\psi$	$\neg(\varphi \wedge \psi)$	$\neg\varphi$	$\neg\psi$
$\neg(\varphi \rightarrow \psi)$	φ	$\neg\psi$	$\varphi \rightarrow \psi$	$\neg\varphi$	ψ

Using this new notation, we can present tableaux in terms of just three rules:

$$\begin{array}{c} \alpha \\ | \\ \alpha_1 \\ \alpha_2 \end{array} \qquad \begin{array}{ccc} & \beta & \\ & / \quad \backslash & \\ \beta_1 & & \beta_2 \end{array} \qquad \begin{array}{c} \neg\neg\varphi \\ \varphi \end{array}$$

At any stage in the process we may well have a choice of rules we can apply. In general, it is a reasonable idea to apply an α-rule or the double negation rule in preference to a β-rule as this tends to cut down the number of sentences we have to consider at each point. However, it doesn't matter, in that the answer will be the same no matter in what order we apply the rules.

Let us summarize the tableau method; we first need some more definitions. A branch in a tableau is **closed** if it contains both φ and $\neg\varphi$ for some sentence φ; otherwise the branch is **open**. A tableau is **closed** if every branch is closed; otherwise the tableau is **open**. A sentence φ is a **literal** if φ is either a propositional letter or the negation of a propositional letter. A tableau is **complete** if every sentence in the tableau which is not a literal has been ticked.

To test whether or not we have $\Gamma \models \psi$, we carry out the above method with $\Gamma \cup \{\neg\psi\}$ as the set of initial sentences, applying the tableau rules until the tableau is complete. If the resulting tableau is closed, then we do have $\Gamma \models \psi$, i.e. we have a valid argument; if the resulting tableau is open, then it is not the case that $\Gamma \models \psi$.

One point is worth noting here. If we start with a set of sentences and, by applying the tableau rules, reach a closed tableau, then there is no point continuing, in that any tableau we can obtain from this closed tableau by applying further rules will also be closed. Accordingly, if we ever reach a closed tableau, then we know that the argument is valid. However, while the tableau is open, we must continue, in that we do not know whether the tableau will remain open as we apply further rules or will close up later. It is only when we reach a complete open tableau that we can be sure we have an invalid argument.

We should stress again that it doesn't matter in which order we apply the rules. If we start with $\Gamma \cup \{\neg\psi\}$ as the initial sentences, and if one sequence of choices leads to a closed tableau, then any sequence of choices will lead to a closed tableau. On the other hand, if one sequence of choices leads to a complete open tableau, then any sequence of choices will lead to a complete open tableau.