2.3 Formulas

Having come this far, we can now capture the concepts we introduced above in precise definitions.

A *language L for propositional logic* has its own reservoir of propositional letters. We shall not specify these; we shall just agree to refer to them by means of the metavariables p, q, and r, if necessary with subscripts appended. Then there are the brackets and connectives ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) which are common to all languages for propositional logic. Together these form the vocabulary of L. In the syntax we define what is meant by the *well-formed expressions* (*formulas, sentences*) in L. The definition is the same for all propositional languages.

Definition 1

(i) Propositional letters in the vocabulary of L are formulas in L.
(ii) If ψ is a formula in L, then $\neg\psi$ is too.
(iii) If ϕ and ψ are formulas in L, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, and $(\phi \leftrightarrow \psi)$ are too.
(iv) Only that which can be generated by the clauses (i)–(iii) in a finite number of steps is a formula in L.

The first three clauses of the definition give a recipe for preparing formulas; (iv) adds that only that which has been prepared according to the recipe is a formula.

We illustrate the definition by examining a few examples of strings of symbols which this definition declares are well-formed, and a few examples of strings which cannot be considered well-formed. According to definition 1, p,

$\neg\neg\neg\neg p$, $((\neg p \wedge q) \wedge r)$, and $((\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r)$ are examples of formulas, while pq, $\neg(\neg\neg p)$, $\wedge p \neg q$, and $\neg((p \rightarrow q \vee r))$ are not.

That p is a formula follows from clause (i), which states that all propositional letters of L are formulas of L. And $\neg\neg\neg\neg p$ is a formula on the basis of (i) and (ii): according to (i), p is a formula, and (ii) allows us to form a new formula from an existing one by prefixing the negation symbol, an operation which has been applied here four times in a row. In $((\neg p \wedge q) \wedge r)$, clause (iii) has been applied twice: it forms a new formula from two existing ones by first introducing an opening, or left, bracket, then the first formula, followed by the conjunction sign and the second formula, and ending with a closing, or right, bracket. In forming $((\neg p \wedge q) \wedge r)$, the operation has been applied first to $\neg p$ and q, which results in $(\neg p \wedge q)$, and then to this result and r. Forming disjunctions, implications, and equivalences also involves the introduction of brackets. This is evident from the fourth example, $((\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r)$, in which the outermost brackets are the result of forming the equivalence of $(\neg(p \vee q) \rightarrow \neg\neg\neg q)$ and r; the innermost are introduced by the construction of the disjunction of p and q; the middle ones result from the introduction of the implication sign. Note that forming the negation does not involve the introduction of brackets. It is not necessary, since no confusion can arise as to what part of a formula a negation sign applies to: either it is prefixed to a propositional letter, or to a formula that begins with a negation sign, or it stands in front of a formula, in whose construction clause (iii) was the last to be applied. In that case the brackets introduced by (iii) make it unambiguously clear what the negation sign applies to.

That pq, i.e., the proposition letter p immediately followed by the proposition letter q, is not a formula is clear: the only way to have two propositional letters together make up a formula is by forming their conjunction, disjunction, implication, or equivalence. The string $\neg(\neg\neg p)$ does not qualify, because brackets occur in it, but no conjunction, disjunction, implication, or equivalence sign, and these are the only ones that introduce brackets. Of course, $\neg\neg\neg p$ *is* well-formed. In $\wedge p \neg q$, the conjunction sign appears before the conjuncts, and not, as clause (iii) prescribes, between them. Also, the brackets are missing. In $\neg((p \rightarrow q \vee r))$, finally, the brackets are misplaced, the result being ambiguous between $\neg(p \rightarrow (q \vee r))$ and $\neg((p \rightarrow q) \vee r)$.

Exercise 1

For each of the following expressions, determine whether it is a formula of propositional logic.

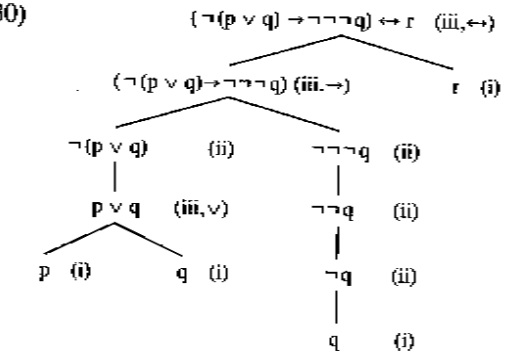
(i) $\neg(\neg p \vee q)$
(ii) $p \vee (q)$
(iii) $\neg(q)$
(iv) $(p_2 \rightarrow (p_2 \rightarrow (p_2 \rightarrow p_2)))$
(v) $(p \rightarrow ((p \rightarrow q)))$
(vi) $((p \rightarrow p) \rightarrow (q \rightarrow q))$

(vii) $((p_{28} \rightarrow p_3) \rightarrow p_4)$
(viii) $(p \rightarrow (p \rightarrow q) \rightarrow q)$
(ix) $(p \vee (q \vee r))$
(x) $(p \vee q \vee r)$
(xi) $(\neg p \vee \neg\neg p)$
(xii) $(p \vee p)$

Leaving off the outer brackets of formulas makes them easier to read and does not carry any danger of ambiguity. So in most of what follows, we prefer to abbreviate $((\neg p \wedge q) \wedge r)$ as $(\neg p \wedge q) \wedge r$, $((\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r)$ as $(\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r$, $((p \wedge q) \wedge (q \wedge p))$ as $(p \wedge q) \wedge (q \wedge p)$, $(p \rightarrow q)$ as $p \rightarrow q$, and $(\neg p \rightarrow q)$ as $\neg p \rightarrow q$. Analogously, we shall write $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$, $\phi \wedge (\phi \vee \chi)$, etc.

Definition 1 enables us to associate a unique *construction tree* with each formula. $(\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r$, for example, must have been constructed according to the tree given in figure (30).

(30)



That each formula has a unique construction tree is due to the fact that, because of the brackets, logical formulas are unambiguous. Beside each *node* in the tree we see the number of the clause from definition 1 according to which the formula at that node is a formula. A formula obtained by applying clause (ii) is said to be a negation, and $\neg$ is said to be its *main sign;* similarly, the main sign of a formula obtained by clause (iii) is the connective thereby introduced (in the example, it is written next to the formula). The main sign of the formula at the top of the tree is, for example, $\leftrightarrow$, and the formula is an equivalence. Note that the formulas at the lowest nodes are all atomic.

A formula ϕ appearing in the construction tree of ψ is said to be a *subformula* of ψ. The subformulas of $\neg(p \vee q)$ are thus: p, q, $p \vee q$, and $\neg(p \vee q)$, while the subformulas of $(\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r$ are: p, q, r, $\neg q$, $\neg\neg q$, $\neg\neg\neg q$, $p \vee q$, $\neg(p \vee q)$, $\neg(p \vee q) \rightarrow \neg\neg\neg q$, and $(\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r$. Any subformula ϕ of ψ is a string of consecutive symbols occurring in the string of symbols ψ, which is itself a formula. And conversely, it can be shown that any string of consecutive symbols taken from ψ which is itself a formula is a subformula of ψ. The proof will be omitted here.

Exercise 2

(a) Draw the construction trees of $(p_1 \leftrightarrow p_2) \vee \neg p_2$ and $p_1 \leftrightarrow (p_2 \vee \neg p_2)$ and of $((p \vee q) \vee \neg r) \leftrightarrow (p \vee (q \vee \neg r))$. In each of the three cases give the subformulas of the formula under consideration.

(b) Give all formulas that can be made out of the following sequence of symbols by supplying brackets: $p \wedge \neg q \rightarrow r$. Also supply their construction trees.

(c) Classify each of the following sentences as an atomic formula, a negation, a conjunction, a disjunction, an implication, or an equivalence.

(i)	$p \rightarrow q$	(vi)	$(p \rightarrow q) \vee (q \rightarrow \neg\neg p)$
(ii)	$\neg p$	(vii)	p_4
(iii)	p	(viii)	$(p_1 \leftrightarrow p_2) \vee \neg p_2$
(iv)	$(p \wedge q) \wedge (q \wedge p)$	(ix)	$\neg(p_1 \wedge p_2) \wedge \neg p_2$
(v)	$\neg(p \rightarrow q)$	(x)	$(p \wedge (q \wedge r)) \vee p$

We now discuss the nature of the last clause of definition 1, which reads:

> Only that which can be generated by the clauses (i)–(iii) in a finite number of steps is a formula in L.

A clause like this is sometimes called the *induction clause* of a definition. It plays a special and important role. If someone were to define a sheep as that which is the offspring of two sheep, we would not find this very satisfactory. It doesn't seem to say very much, since if you don't know what a sheep is, then you are not going to be much wiser from hearing the definition. The definition of a sheep as the offspring of two sheep is *circular*. Now it might seem that definition 1 is circular too: clause (ii), for example, states that a $\neg$ followed by a formula is a formula. But there is really no problem here, since the formula ϕ occurring after the $\neg$ is simpler than the formula $\neg\phi$, in the sense that it contains fewer connectives, or equivalently, that it can be generated by clauses (i)–(iii) in fewer steps. Given that this ϕ is a formula, it must be a formula according to one of the clauses (i)–(iii). This means that either ϕ is a propositional letter (and we know what these are), or else it is a composite formula built up of simpler formulas. So ultimately everything reduces to propositional letters.

In a definition such as definition 1, objects are said to have a given property (in this case that of being a formula) if they can be constructed from other, 'simpler' objects with that property, and ultimately from some group of objects which are simply said to have that property. Such definitions are said to be *inductive* or *recursive*.

The circular definition of a sheep as the offspring of two sheep can be turned into an inductive definition (i) by stipulating two ancestral sheep, let us call them Adam and Eve; and (ii) by ruling that precisely those things are sheep which are required to be sheep by (i) and the clause saying that the offspring of two sheep is a sheep. The construction tree of any given sheep, according to this inductive definition, would be a complete family tree going back to the first ancestral sheep Adam and Eve (though contrary to usual practice with family trees, Adam and Eve will appear at the bottom).

Most of what follows applies equally to all propositional languages, so instead of referring to the formulas of any particular propositional language, we shall refer to *the formulas of propositional logic*.

Because the concept of a formula is defined inductively, we have at our disposal a simple method by which we can prove that all formulas have some particular property which we may be interested in. It is this. In order to prove that all formulas have a property A, it is sufficient to show that:

(i) The propositional letters all have property A;
(ii) if a formula ϕ has A, then $\neg\phi$ must too;
(iii) if ϕ and ψ have property A, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, and $(\phi \leftrightarrow \psi)$ must too.

This is sufficient because of induction clause (iv), which ensures that every composite formula must be composed of some simpler formula(s) from which it inherits property A. A proof of this sort is called a *proof by induction on the complexity of the formula* (or a *proof by induction on the length of the formula*). As an example of a proof by induction on the complexity of a formula, we have the following simple, rigorous proof of the fact that all formulas of propositional logic have just as many right brackets as left brackets:

(i) Propositional letters have no brackets at all.
(ii) If ϕ has the same number of right brackets as left brackets, then $\neg\phi$ must too, since no brackets have been added or taken away.
(iii) If ϕ and ψ each have as many right brackets as left brackets, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, and $(\phi \leftrightarrow \psi)$ must too, since in all of these exactly one left and one right bracket have been added.

Quite generally, for every inductive definition there is a corresponding kind of proof by induction.

There are various points in this book where if complete mathematical rigor had been the aim, inductive proofs would have been given. Instead we choose merely to note that strictly speaking, a proof is required.

The fact that the concept of a formula has been strictly defined by definition 1 enables us to give strict inductive definitions of notions about formulas. For example, let us define the function $(\phi)^0$ from formulas to natural numbers by:

$(p)^0 = 0$,
$(\neg\phi)^0 = (\phi)^0$
$((\phi * \psi))^0 = (\phi)^0 + (\psi)^0 + 2$, for each two-place connective $*$.

Then, for each formula ϕ, $(\phi)^0$ gives the number of brackets in the formula ϕ.

Exercise 3 ◇

(a) The *operator depth* of a formula of propositional logic is the maximal length of a 'nest' of operators occurring in it. E.g., $((\neg p \wedge q) \wedge \neg r)$ has

operator depth 3. Give a precise definition of this notion, using the inductive definition of formulas.

(b) Think of the construction trees of formulas. What concepts are defined by means of the following ('simultaneous') induction?

$A(p) = 1$	$B(p) = 1$
$A(\neg\psi) = A(\psi) + 1$	$A(\neg) = \max(B(\psi), A(\psi) + 1)$
$A(\psi \circ \chi) = \max(A(\psi), (A\chi)) + 1$	$B(\psi \circ \chi) = \max(B(\psi), B(\chi),$
for the two-place connectives $\circ$	$A(\psi) + A(\chi) + 1),$

Exercise 4 ◇

(a) What notions are described by the following definition by induction on formulas?

$p^* = 0$	for propositional letters p
$(\neg\phi)^* = \phi^*$	
$(\phi \circ \psi)^* = \phi^* + \psi^* + 1$	for two-place connectives $\circ$
$p^+ = 1$	
$(\neg\phi)^+ = \phi^+$	
$(\phi \circ \psi)^+ = \phi^+ + \psi^+$	for two-place connectives $\circ$

(b) Prove by induction that for all formulas ϕ, $\phi^+ = \phi^* + 1$.