

4.1 Formulas as Objects

Chapter 3 ended on a question that no derivation can answer. We show that a sequent holds by writing a derivation of it; but how could we show that one does **not** hold, that no derivation of $Q \rightarrow P$ from $P \rightarrow Q$ exists, however long and however clever? This chapter's answer is a theorem about **every** derivation of **every** formula, and such a theorem needs, first of all, an exact description of what a formula is. This section gives it.

4.1.1 Why formulas must become objects

Here is the theorem that Section 4.7 proves, stated loosely:

If ψ can be derived from premises Γ , then ψ is true in every situation in which all the premises are true.

Read it as a mathematician reads a statement about all integers. “For every integer n , $n^2 - n$ is even” can be proved only by someone who knows what an integer is: every integer is even or odd, and each kind can be handled. A proof needs a way to take hold of an arbitrary object of the kind the statement is about. The loose theorem above is about arbitrary derivations, and a derivation is a list of formulas, so the proof will need a way to take hold of an arbitrary formula.

Chapters 1–3 never needed one. There we read statements by eye: we found the main connective of $\neg(P \wedge Q) \rightarrow R$ (Definition 1.1.2) by looking, and we drew its tree by hand. That is fine for one statement at a time. It is useless for a claim about all statements, because “all the strings we would recognize as statements” is not a description anyone can reason from. So we now say exactly which objects we mean, in a form that later proofs can use.

The description has three parts. First, the **alphabet**, the symbols formulas are written with. Second, the **formation rules**, which say how formulas are built. Third, a closing clause saying that **nothing else** is a formula. The third is the one that is easy to forget and that makes everything else work.

4.1.2 The alphabet

Formulas are strings of symbols. A **string** is a finite list of symbols written one after another, and its **length** is the number of symbols in it. The symbols are these.

- **Atoms:** $p, q, r, p_1, p_2, p_3, \dots$, infinitely many of them. Each atom is **one** symbol: p_1 is a single symbol whose name happens to carry a subscript, not the two symbols p and 1.
- **The constant** $\perp$, read “contradiction”, as in Chapter 1.
- **The connectives:** $\neg$ (the only one that takes one part) and $\wedge, \vee, \rightarrow, \leftrightarrow$ (the **binary** connectives, which take two).
- **The parentheses** (and).

An atom plays the part that a letter played in Chapters 1–3: a statement whose inner structure we ignore. In those chapters the letters were capitals, P, Q, R , and they stood for **arbitrary** statements, atomic or compound. From now on we keep two kinds of letter apart, because the theorems of this chapter need the difference.

Remark. Who is who: the letters of this chapter.

Letter	Stands for	Inside a formula?	Example
$p, q, r, p_1, p_2, \dots$	an atom , one symbol of the language	yes	$(p \wedge \neg q)$
φ, ψ, χ (phi, psi, chi)	any formula: a metavariable , a name we use when talking about formulas	never	“if φ is a formula, so is $\neg\varphi$ ”
Γ, Δ (capital gamma, delta)	a finite list of formulas, as in the sequents of Chapter 1 (Definition 1.2.3)	never	$\Gamma \vdash \psi$
P, Q, R of Chapters 1–3	arbitrary statements; read from now on as atoms, or as any formulas (the last part of this section)	they are replaced	$P \rightarrow Q$ read as $p \rightarrow q$

Chapter 1 already used the three letters φ, ψ, χ in exactly this way, in the schemas of its rules (Definition 1.2.1).

Quick check. Which of these is a formula, and which is a name for formulas or for a list: $(p \rightarrow q)$; $(\varphi \rightarrow q)$; Γ ?

Solution. $(p \rightarrow q)$ is a formula. $(\varphi \rightarrow q)$ is not a formula, since φ is no symbol of the language; it is a pattern, and each formula put in for φ gives a formula. Γ names a list of formulas.

Two languages. Everything in this chapter is said in two languages at once. The **object language** is the language of formulas: its expressions are strings over the alphabet above, such as $(p \rightarrow q)$. The **metalanguage** is the language in which we talk about those strings: English with ordinary mathematics, together with names for formulas (φ, ψ) and for lists (Γ), and claims such as $\Gamma \vdash \psi$ and, from Section 4.5 on, $\Gamma \models \psi$. Neither $\vdash$ nor $\models$ is a symbol of the alphabet. The sentence “ $(p \rightarrow q)$ has length 5” belongs to the metalanguage and speaks about a string of the object language; $(p \rightarrow q)$ on its own is a string, not a claim. The theorems of this chapter, unique readability, soundness and completeness, are theorems **about** the object language, proved in the metalanguage by ordinary mathematics: induction, cases, counting. To speak about a symbol we name it, as we name a word. “ p is an atom” mentions the symbol p , in the way that “‘seven’ has five letters” mentions a word.

Quick check. Sort into strings of the object language, claims of the metalanguage, and neither:

- (a) $((p \wedge q) \rightarrow p)$

- (b) $p \wedge q \vdash p$
- (c) the main connective of $((p \wedge q) \rightarrow p)$ is $\rightarrow$
- (d) $(\varphi \wedge q)$
- (e) $\neg \perp$ is a formula

Solution. (a) A string of the object language: a formula.

(b) A claim of the metalanguage, since $\vdash$ is not a symbol of the alphabet. It is true, by $\wedge E$.

(c) A claim of the metalanguage, and true.

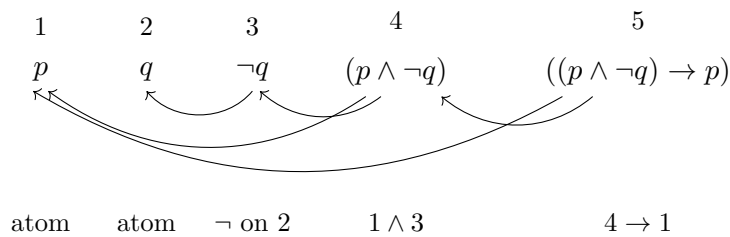
(d) Neither. The letter φ is a metavariable, not a symbol, so the expression is a pattern; each formula put in for φ gives a formula.

(e) A claim of the metalanguage, and true: $\perp$ is a formula, and $\neg$ in front of a formula gives a formula (clauses (F1) and (F2) of the definition below).

4.1.3 Formulas

Chapter 1 described a statement as something “built from smaller ones by a connective”. The definition below says exactly how the building goes. It is an **inductive definition**: it names the starting pieces, names the ways of building new formulas from old, and then closes the door on everything else.

Building one formula. Here is the formula $((p \wedge \neg q) \rightarrow p)$ built one step at a time.



Each entry is an atom, or is built from earlier entries by one connective. The atom p is used twice and listed once.

A list like this is called a formation sequence. The definition below has one more clause than the building steps. Its clause (F4) says that the formulas are exactly the last entries of such lists: the list you just saw, in general.

In clause (F4) below, $\circ$ stands for any one of the four binary connectives, as φ stands for any formula: $(\chi_j \circ \chi_l)$ means any one of $(\chi_j \wedge \chi_l)$, $(\chi_j \vee \chi_l)$, $(\chi_j \rightarrow \chi_l)$, $(\chi_j \leftrightarrow \chi_l)$.

A formula is a string built from atoms and $\perp$ by the connectives, one step at a time, with a pair of parentheses around every binary step.

Definition 4.1.1. Formula

The **formulas** are the strings given by the following clauses.

- (F1) Every atom is a formula, and $\perp$ is a formula.
- (F2) If φ is a formula, then $\neg\varphi$ is a formula.
- (F3) If φ and ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$ are formulas.
- (F4) **Nothing else** is a formula: a string is a formula **only if** it is the last entry of a **formation sequence**. A formation sequence is a finite list of strings $\chi_1, \dots, \chi_m$, with $m \geq 1$, in which **every** entry χ_i is an atom or $\perp$, or is $\neg\chi_j$ for some **earlier** entry χ_j (so $j < i$), or is $(\chi_j \circ \chi_l)$ for some earlier entries χ_j and χ_l (so $j, l < i$; $j = l$ is allowed) and some binary connective $\circ$.

In words, clause by clause.

- *(F1), the starting pieces.* A single atom is a formula, and so is $\perp$. These are the only formulas of length one.
- *(F2), negation.* Writing $\neg$ in front of a formula gives a formula. No parentheses are added: $\neg p$ and $\neg\neg p$ are formulas, and $\neg\neg p$ is $\neg$ in front of the formula $\neg p$.
- *(F3), the binary connectives.* Writing a binary connective between two formulas **and** wrapping the result in one pair of parentheses gives a formula. The two formulas may be the same: $(p \wedge p)$ is a formula.
- *(F4), nothing else.* Clauses (F1)–(F3) say which strings **are** formulas. They do not say that other strings are **not**. Clause (F4) does: a formula is exactly something we can reach from atoms and $\perp$ by finitely many uses of (F2) and (F3), and the formation sequence is the written record of such a reaching. Think of building blocks: (F1) lists the blocks, (F2) and (F3) the two ways of joining them, and (F4) says that only what is built in this way counts.

Two facts about formation sequences are used constantly, and both are immediate from the definition.

First, **every entry of a formation sequence is a formula**. Walk along the list. The first entry χ_1 is an atom or $\perp$, so it is a formula by (F1). Each later entry is an atom or $\perp$, a formula by (F1), or it is $\neg\chi_j$ or $(\chi_j \circ \chi_l)$ for earlier entries that are already formulas, and so it is a formula by (F2) or (F3). (Section 4.3 makes this walk exact with strong induction, in the fold “Sets built from a base”.)

Second, **every formula was made by one of the clauses from formulas**. Take a formula χ and a formation sequence ending in it. Its last entry χ is an atom or $\perp$, or it is $\neg\chi_j$, or it is $(\chi_j \circ \chi_l)$, where the earlier entries χ_j, χ_l are formulas by the first fact. So every formula is an atom, or $\perp$, or $\neg\varphi$ for a formula φ , or $(\varphi \circ \psi)$ for formulas φ, ψ . We will call this **looking at the last step** of a formula.

Looking at the last step already tells us something about the first symbol. A formula of the form $\neg\varphi$ begins with $\neg$, and one of the form $(\varphi \circ \psi)$ begins with (and ends with). So:

The first-symbol test. A formula of length one is an atom or $\perp$. A longer formula begins with $\neg$ or with (, and if it begins with (, it is $(\varphi \circ \psi)$ for formulas φ, ψ and ends with).

(A formula of the form $\neg\varphi$ has length at least 2, and one of the form $(\varphi \circ \psi)$ has length at least 5, since φ and ψ each have at least one symbol. So a formula of length one is made by (F1).)

Quick check. Write a formation sequence for $\neg(p \vee \perp)$, and say which earlier entries each entry uses.

Solution. p , $\perp$, $(p \vee \perp)$, $\neg(p \vee \perp)$. Entry 3 uses entries 1 and 2, and entry 4 uses entry 3; entries 1 and 2 use none.

Example. Formation sequences

Write down a formation sequence for each of the strings p , $\perp$, $\neg\neg q$ and $(\neg p \vee (q \rightarrow \perp))$, which shows that each is a formula.

Solution. For p , the one-entry list p is a formation sequence: its only entry is an atom. Likewise the list $\perp$. These are the degenerate cases, and they matter: a formula need not have been built at all. Every structural argument in Section 4.3 has to deal with them.

For $\neg\neg q$:

$$q, \quad \neg q, \quad \neg\neg q.$$

The first entry is an atom; the second is $\neg$ in front of the first; the third is $\neg$ in front of the second.

For $(\neg p \vee (q \rightarrow \perp))$:

$$p, \quad \neg p, \quad q, \quad \perp, \quad (q \rightarrow \perp), \quad (\neg p \vee (q \rightarrow \perp)).$$

Entries 1, 3 and 4 are atoms or $\perp$. Entry 2 is $\neg$ in front of entry 1. Entry 5 is $(\chi_3 \rightarrow \chi_4)$, and entry 6 is $(\chi_2 \vee \chi_5)$.

🔄 Second reading • Many formation sequences

A formula has many formation sequences. In [Example \(Formation sequences\)](#) we could have listed q before p in the last list, or added entries that are never used, such as r or $\neg r$. The definition asks only that **some** formation sequence ends in the string, and nothing in this chapter depends on which one we take.

To show that a string **is** a formula, we exhibit a formation sequence, just as we show that a sequent holds by exhibiting a derivation. To show that a string is **not** a formula, exhibiting is not enough; we need a property that every formula has and the string lacks. The first-symbol test supplies such properties.

Example. Strings that are not formulas

Show that none of the strings $p \wedge \rightarrow q$, $(p \wedge q, (\neg p)$ and $p \wedge q$ is a formula.

Solution. $p \wedge \rightarrow q$: it has length 4, so by the first-symbol test a formula would begin with $\neg$ or $($. It begins with p . Not a formula.

$(p \wedge q$: it begins with $($, so by the first-symbol test a formula would end with $)$. It ends with

q . Not a formula. (The missing parenthesis is the whole problem: $(p \wedge q)$ is a formula.)

$(\neg p)$: it begins with $($, so a formula would have to be $(\varphi \circ \psi)$, with the two symbols $\neg p$ between the parentheses equal to $\varphi \circ \psi$, and so containing a binary connective. The string $\neg p$ contains none. Not a formula. The official language puts no parentheses around a negation; (F2) adds none.

$p \wedge q$: it has length 3 and begins with p . Not a formula. This one is a surprise at first, since we have written $p \wedge q$ all along. It is **shorthand** for the formula $(p \wedge q)$, in the way explained later in this section.

The last two strings are ones Chapter 1 wrote freely, and that is no mistake in Chapter 1. Chapter 1 used a convenient shorthand; the official formulas carry every parenthesis that (F3) puts in and none that it does not. We come back to the shorthand below.

Quick check. Which of the following strings are formulas? For each, give a formation sequence or name the test it fails.

- (a) $\neg(p \rightarrow \perp)$
- (b) $(p \rightarrow \neg)$
- (c) $\perp \perp$

Solution. (a) A formula: p , $\perp$, $(p \rightarrow \perp)$, $\neg(p \rightarrow \perp)$ is a formation sequence.

(b) Not a formula. It begins with $($, so it would be $(\varphi \rightarrow \psi)$ with φ the string p and ψ the string $\neg$ (the only binary connective is the $\rightarrow$). But $\neg$ is a string of length one that is neither an atom nor $\perp$, so by the first-symbol test it is not a formula.

(c) Not a formula. It has length 2, so it would begin with $\neg$ or $($; it begins with $\perp$.

4.1.4 Why this definition

Each feature of the definition is there because something breaks without it.

Without (F4), clauses (F1)–(F3) would be satisfied by far too many collections of strings. The collection of **all** strings contains every atom and $\perp$, and is closed under writing $\neg$ in front and under wrapping two strings with a connective in parentheses; so (F1)–(F3) alone cannot tell us that $p \wedge \rightarrow q$ is excluded. Clause (F4) says the formulas are only those strings that the building steps reach. In other words, the formulas are the **smallest** collection of strings closed under (F1)–(F3).

Without the parentheses in (F3), a string could be read in two ways. Take $p \wedge q \vee r$. Built one way, it is “ $p \wedge q$ ” joined to r by $\vee$; built the other way, it is p joined to “ $q \vee r$ ” by $\wedge$. The two constructions have different parse trees (defined below), and they say different things (the everyday reading of Chapter 1 is enough here): if p is false and r is true, then “ p and q , or r ” is true, because r is, while “ p , and q or r ” is false, because p is. With the parentheses of (F3) the two constructions give two **different** strings,

$$((p \wedge q) \vee r) \quad \text{and} \quad (p \wedge (q \vee r)),$$

and the ambiguity disappears. Section 4.3 proves that it disappears for every formula: a formula can be built in only one way (the unique readability theorem).

🕒 **Second reading** · Two smaller points

Why not parentheses around a negation too? Nothing would break, but nothing would be gained. The symbol $\neg$ always stands in front of what it negates, so $\neg\varphi$ can be read in only one way without help. Keeping the official language as lean as possible makes the proofs about it shorter.

Why atoms, and not sentences like “ $x > 0$ ”? Because the theorems of this chapter are about **forms**, and the form of an argument is what remains when the content of its atomic statements is forgotten (Chapter 1’s slogan: logic studies the forms of arguments). An atom is a statement reduced to a name.

Warning. A formula is a string of symbols, and questions about formulas are questions about strings. “Is $(p \wedge p)$ the same formula as p ?” has the answer **no**: the first has length 5 and the second length 1, even though they always have the same truth value (Section 4.4 gives the means to check this). Sameness of formulas is letter-for-letter identity; sameness of meaning is a different relation, which Section 4.5 names.

4.1.5 Parse trees

A formation sequence records how a formula was built, but it is a list, and a list hides the shape of the building. Chapter 1 drew the shape as a tree, with the main connective at the top (the tree after Definition 1.1.2). We now make that drawing official.

A parse tree is a formula’s construction drawn upside down: the last step at the root, the atoms at the leaves.

Definition 4.1.2. Parse Tree

A **parse tree** for a formula is defined along the clauses of Definition 4.1.1.

- (T1) For an atom or $\perp$: a single node, labeled by that atom or by $\perp$.
- (T2) If T is a parse tree for φ , then the tree whose **root** is labeled $\neg$ and has T hanging below it as its only subtree is a parse tree for $\neg\varphi$.
- (T3) If T and T' are parse trees for φ and ψ , then the tree whose root is labeled $\circ$, with T hanging below it on the **left** and T' on the **right**, is a parse tree for $(\varphi \circ \psi)$.
- (T4) **Nothing else** is a parse tree: a tree is a parse tree for a formula φ **only if** that follows from finitely many uses of (T1)–(T3). In particular, a parse tree for $\neg\varphi$ is made by (T2), and a parse tree for $(\varphi \circ \psi)$ by (T3).

The **leaves** are the nodes with nothing below them; a **branch** is a path from the root down to a leaf.

In words: (T1) is the tree of a formula that was not built at all, a single node. (T2) puts a new

root $\neg$ on top of an existing tree. (T3) joins two existing trees under a new root carrying the connective, keeping left and right apart, because $(p \rightarrow q)$ and $(q \rightarrow p)$ are different formulas. (T4), as (F4) did for formulas, says that only trees built this way count; Section 4.3 uses it when it asks how a given tree could have been made. Parentheses do not appear in a parse tree: their only job in the string was to show the grouping, and the tree shows the grouping by its shape.

Every formula has a parse tree. Take a formation sequence of the formula and build a tree for each entry in turn: an atom or $\perp$ gets (T1), and an entry $\neg\chi_j$ or $(\chi_j \circ \chi_l)$ gets (T2) or (T3) from the trees already built for the earlier entries. The tree built for the last entry is a parse tree for the formula.

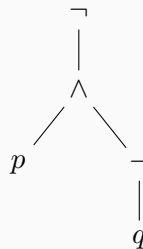
Could a formula have **two** different parse trees? That would happen if one string could be built in two different ways, as $p \wedge q \vee r$ could if parentheses were not written. Section 4.3 proves that it never happens (the unique readability theorem); until then we say “**a** parse tree”. After that section, every formula has exactly one, and we speak of **the** parse tree.

Example. Three parse trees

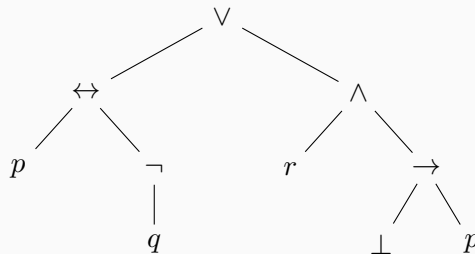
Draw a parse tree for each formula: (a) p ; (b) $\neg(p \wedge \neg q)$; (c) $((p \leftrightarrow \neg q) \vee (r \wedge (\perp \rightarrow p)))$, which uses every connective and $\perp$.

Solution. (a) A single node labeled p , by (T1). This is the degenerate tree: its root is also its only leaf, and its only branch has one node.

(b) By (T1) the atoms p and q get one-node trees; (T2) puts $\neg$ above q ; (T3) joins p and $\neg q$ under $\wedge$; (T2) puts $\neg$ on top.



(c) The main connective is the $\vee$: it is the last step, joining $(p \leftrightarrow \neg q)$ on the left to $(r \wedge (\perp \rightarrow p))$ on the right. Each side is taken apart in the same way.



The leaves are p , q , r , $\perp$ and p again: the atom p occurs twice in the formula, and each occurrence is its own leaf.

To draw a parse tree from a string, find the last step, draw it as the root, and repeat on the parts. For a formula $(\varphi \circ \psi)$, the last step is the binary connective that sits inside **exactly one**

pair of parentheses; in (c), it is the $\vee$, and every other binary connective sits inside at least two pairs. Section 4.3 proves that this recipe always finds the main connective, and only it.

4.1.6 Subformulas

Each node of a parse tree is the root of the part of the tree below it, and that part is itself a parse tree, built by the same clauses, for a formula. The formulas that arise in this way are the pieces a formula is built from.

Definition 4.1.3. Subformula

The **subformulas** of a formula φ are the formulas whose parse trees hang from the nodes of a parse tree for φ : for each node, the formula of which the part of the tree from that node downwards is a parse tree. The formulas at the children of the root are the **immediate subformulas** of φ .

Until Section 4.3 this depends, as far as we know, on the parse tree chosen; Section 4.3 shows there is only one, so the subformulas of φ are determined by φ .

So the subformulas of φ include φ itself (at the root) and each atom that occurs in φ (at the leaves). A formula of the form $\neg\varphi$ has one immediate subformula, φ ; a formula $(\varphi \circ \psi)$ has the two immediate subformulas φ and ψ (one, if φ and ψ are the same formula); an atom or $\perp$ has none. When we list the subformulas, a formula that sits at several nodes is listed once.

For the formula (c) of [Example \(Three parse trees\)](#), the subformulas are

$$\begin{aligned} &((p \leftrightarrow \neg q) \vee (r \wedge (\perp \rightarrow p))), \quad (p \leftrightarrow \neg q), \quad (r \wedge (\perp \rightarrow p)), \\ &\neg q, \quad (\perp \rightarrow p), \quad p, \quad q, \quad r, \quad \perp, \end{aligned}$$

nine in all. The tree has ten nodes, $\vee$, $\leftrightarrow$, p , $\neg$, q , $\wedge$, r , $\rightarrow$, $\perp$ and p again, and only p sits at two of them; so ten nodes give nine distinct subformulas. Counting by eye is easy to get wrong, and the safe method is to walk the tree node by node, as we just did.

A string that merely **appears inside** a formula need not be a subformula. The formula $((p \wedge q) \vee r)$ contains the string $q) \vee r$, which is not a formula at all, and it contains the string $p \wedge q$, which is only the shorthand of a formula. Its subformulas are the formulas at the nodes: $((p \wedge q) \vee r)$, $(p \wedge q)$, p , q and r .

Quick check. Draw the parse tree of $(\neg p \vee q) \leftrightarrow \neg(r \wedge p)$ (in official form, $((\neg p \vee q) \leftrightarrow \neg(r \wedge p))$) and list its subformulas.

Solution. The root is $\leftrightarrow$. Its left subtree is $\vee$ over $\neg p$ and q , where $\neg p$ is $\neg$ over the leaf p . Its right subtree is $\neg$ over $\wedge$, and $\wedge$ has the leaves r and p .

The subformulas are the formula itself, $(\neg p \vee q)$, $\neg(r \wedge p)$, $\neg p$, q , $(r \wedge p)$, p and r : **eight**, with p listed once although it sits at two leaves.

Warning. A parse tree is not a truth table. The tree records how a formula is **built**, and it is the same whatever the atoms stand for; nothing in it is true or false. Values come in Section 4.4, where they are computed **along** the tree, from the leaves up. Mixing the two,

for instance writing “T” beside a node while drawing the tree, is a habit worth not forming: the same tree carries different values under different assignments.

4.1.7 Shorthand and official formulas

Nobody wants to write $((p \wedge q) \rightarrow r) \leftrightarrow (p \rightarrow (q \rightarrow r))$. Chapter 1 used conventions that let us drop parentheses (Section 1.1, the conventions before Definition 1.1.2), and we keep using them. What changes is their status: they are **not** part of the definition of a formula. A string written with the conventions is **shorthand**, and it stands for exactly one official formula, the one obtained by putting the dropped parentheses back.

The translation, from shorthand to official form, goes as follows.

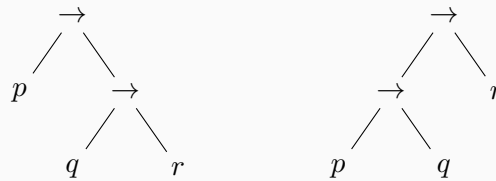
1. Put back the outer pair of parentheses of a formula whose main connective is binary: $p \wedge q$ stands for $(p \wedge q)$.
2. $\neg$ binds most tightly: it applies to the smallest formula that follows it. So $\neg p \wedge q$ stands for $(\neg p \wedge q)$, not for $\neg(p \wedge q)$.
3. $\wedge$ and $\vee$ bind more tightly than $\rightarrow$ and $\leftrightarrow$: $p \wedge q \rightarrow r$ stands for $((p \wedge q) \rightarrow r)$.
4. $\rightarrow$ groups to the right: $p \rightarrow q \rightarrow r$ stands for $(p \rightarrow (q \rightarrow r))$.
5. Chapter 1 sometimes wrote a negation in parentheses for emphasis, as in $(\neg P) \rightarrow Q$; such parentheses are simply removed.

In a mixture of $\wedge$ with $\vee$, or of $\rightarrow$ with $\leftrightarrow$, the shorthand keeps the parentheses (Chapter 1’s rule for such mixtures), so no rule is needed for those.

Example. Two groupings of three implications

Write the official formula for the shorthand $p \rightarrow q \rightarrow r$, draw its parse tree beside that of $((p \rightarrow q) \rightarrow r)$, and find a situation in which the two formulas say different things.

Solution. By rule 4, $p \rightarrow q \rightarrow r$ stands for $(p \rightarrow (q \rightarrow r))$: the first $\rightarrow$ is the main connective, and its conclusion is $(q \rightarrow r)$. The other grouping, $((p \rightarrow q) \rightarrow r)$, has the second $\rightarrow$ as main connective, and its hypothesis is $(p \rightarrow q)$. Left, $(p \rightarrow (q \rightarrow r))$; right, $((p \rightarrow q) \rightarrow r)$.



The two trees are mirror images in shape, and they say different things. Suppose p , q and r are all false. The left formula is an implication whose hypothesis p is false, so it is true (vacuous truth, Section 2.1). The right formula has the true hypothesis $p \rightarrow q$ (again vacuous, since p is false) and the false conclusion r , so it is false. The everyday reading of Chapters 1 and 2 is enough to see this; Section 4.4 makes it a computation.

This is why the grouping convention matters. Grouping to the right is the useful choice, because statements of the form “if P , then if Q , then R ” are common and $(P \rightarrow Q) \rightarrow R$ is rare.

Going the other way, from official to shorthand, we drop every pair of parentheses the rules would put back. $(\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q))$ becomes $\neg(p \vee q) \leftrightarrow \neg p \wedge \neg q$: the outer pair goes by rule 1, the pair around $\neg p \wedge \neg q$ by rule 3, and the pair after the first $\neg$ must stay, since without it rule 2 would read $\neg p \vee q$.

From now on we write formulas in shorthand whenever that is clearer, and every statement about “the formula $p \wedge q$ ” is about the official formula $(p \wedge q)$. Where the official form matters, as in counting parentheses or symbols, we write it out.

4.1.8 Letters, metavariables and substitution

We can now say exactly what the letters of Chapters 1–3 were.

Every rule was stated as a schema (Definition 1.2.1), with metavariables: $\wedge I$ writes $\varphi \wedge \psi$ from φ and ψ , **for any** formulas φ and ψ . The derivations of those chapters were written with capital letters, as in the derivation of Proposition 1.4.1:

1	$P \rightarrow Q$	
2	$Q \rightarrow R$	
3	P	asm, for $\rightarrow I$
4	Q	$\rightarrow E$ 1, 3
5	R	$\rightarrow E$ 2, 4
6	$P \rightarrow R$	$\rightarrow I$ 3–5

In the language of this chapter, read the letters P, Q, R as three atoms. Then this is a derivation of $p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$ in the official sense, written in shorthand. But Chapter 1 meant more: the derivation was a **pattern**, good for any statements put in for the letters. Here is that promise made exact.

A derivation written with letters is a template.

Substitution. Replace each atom of a derivation by a formula, **the same formula at every occurrence of the same atom**, and all such replacements at once. The result is again a derivation.

The fold below checks that every line stays a formula and stays justified.

🔄 **Second reading** · *Why the result is again a derivation*

Here is why. First, each line becomes a formula. Replacing atoms by formulas in a formula χ gives a formula: take a formation sequence of χ , write in front of it a formation sequence of each formula put in for an atom, and replace the atoms inside the old entries. Each old entry that was an atom has become the last entry of one of the sequences written in front; each old entry built by (F2) or (F3) is now built by the same clause from the replaced versions of the same earlier entries. So the new list is a formation sequence, and its last entry is the replaced χ .

Second, each line is still justified. Each line of the old derivation was a premise, an assumption, a repetition, or an instance of a rule’s schema (Definition 1.4.2, clauses (D1)–(D4)). Being an instance means: the line and the lines it cites are the schema with its metavariables replaced by certain formulas. The schema’s own symbols are connectives, $\perp$ and parentheses, never atoms,

so replacing atoms in the lines is the same as replacing atoms inside those certain formulas. The new lines are therefore the same schema with its metavariables replaced by the new formulas: an instance of the same rule.

Third, nothing about citations changes. The line numbers, the boxes, and which line cites which are untouched, and whether a line or a box is available depends on nothing else (Definition 1.4.2, the scope rule). So the result is a derivation, with the replaced premises and the replaced last line.

For instance, put $(p_1 \wedge p_2)$ for P , $\neg p_1$ for Q , and $(p_2 \vee p_1)$ for R in the derivation above:

1	$(p_1 \wedge p_2) \rightarrow \neg p_1$	
2	$\neg p_1 \rightarrow (p_2 \vee p_1)$	
3	$p_1 \wedge p_2$	asm, for $\rightarrow$ I
4	$\neg p_1$	$\rightarrow$ E 1, 3
5	$p_2 \vee p_1$	$\rightarrow$ E 2, 4
6	$(p_1 \wedge p_2) \rightarrow (p_2 \vee p_1)$	$\rightarrow$ I 3–5

Line 4 is still $\rightarrow$ E: lines 1 and 3 are $\varphi \rightarrow \psi$ and φ with φ the formula $p_1 \wedge p_2$ and ψ the formula $\neg p_1$. Line 6 is still $\rightarrow$ I with the same box.

The fact is immediate rule by rule, because every rule was stated for arbitrary formulas from the start. It is what the phrase “the letters stand for arbitrary statements” (Chapter 1, Section 1.2) and the clause “replace each letter ... the same statement for every occurrence” of Definition 2.3.1 were relying on. We use it twice later in the chapter, in the soundness proof of Section 4.7 and in the completeness proof of Section 4.8.

Remark. The replacement must be **uniform**. Replacing only the first P of the derivation above by $p_1 \wedge p_2$, and leaving the P on line 3 alone, would break line 4, which needs the hypothesis of line 1 to be literally the formula on line 3.

Remark (Looking ahead). Section 4.3 proves that every formula is built in only one way. Then “the parse tree” of a formula makes sense, and with it the height of a formula.

4.1.9 Exercises

4.1.9.1 A. Check your understanding

Exercise. A1

- (a) Decide which of the following strings are formulas. Give a formation sequence for each formula, and for each nonformula name the test or clause it fails: (i) $(p \vee \perp)$; (ii) $\neg\neg p$; (iii) $(p \neg q)$; (iv) $((p \rightarrow q))$; (v) $(p \wedge q \wedge r)$.
- (b) List the subformulas of $(\neg(p \rightarrow q) \vee q)$.

- (c) Write a formation sequence of $((p \vee q) \wedge \neg(p \vee q))$ in which $(p \vee q)$ is listed once.
- (d) Why does clause (F3) of Definition 4.1.1 put parentheses around its result, while (F2) does not?
- (e) Explain why clauses (F1)–(F3) alone do not show that $\rightarrow p$ is not a formula, and which clause does.

Solution. (a) (i) A formula: $p, \perp, (p \vee \perp)$.

(ii) A formula: $p, \neg p, \neg\neg p$.

(iii) Not a formula. It begins with $($, so by the first-symbol test it would be $(\varphi \circ \psi)$, and the string $p \neg q$ between the parentheses would contain a binary connective. It contains none.

(iv) Not a formula. It begins with $($, so it would be $(\varphi \circ \psi)$ with $\varphi \circ \psi$ the string $(p \rightarrow q)$. The only binary connective in that string is the $\rightarrow$, so φ would be the string $(p$, of length 2. A formula of length 2 begins with $\neg$, by the first-symbol test (one beginning with $($ has length at least 5). So $(p$ is not a formula, and neither is the string. So here the doubled outer pair is not official.

(v) Not a formula. It would be $(\varphi \wedge \psi)$ with the $\wedge$ one of the two. If it is the first, ψ is the string $q \wedge r$; if the second, φ is $p \wedge q$. Each has length 3 and begins with an atom, so by the first-symbol test neither is a formula. (Chapter 1's conventions give no reading of the shorthand $p \wedge q \wedge r$; Section 4.5 introduces it for the right-grouped $(p \wedge (q \wedge r))$ and explains why either grouping would do.)

(b) The formula is $(\neg(p \rightarrow q) \vee q)$. Its subformulas are $(\neg(p \rightarrow q) \vee q)$, $\neg(p \rightarrow q)$, $(p \rightarrow q)$, p and q : five, with q listed once although it sits at two leaves.

(c) $p, q, (p \vee q), \neg(p \vee q), ((p \vee q) \wedge \neg(p \vee q))$. Entry 4 uses entry 3, and entry 5 uses entries 3 and 4: $(p \vee q)$ is built once and used twice.

(d) A binary connective joins two formulas, and without parentheses a string such as $p \wedge q \vee r$ could be built in two ways, with different meanings (see “Why this definition”). A negation stands in front of the one formula it negates, so there is nothing to group.

(e) Clauses (F1)–(F3) say which strings are formulas and never which are not: the collection of all strings satisfies them. Clause (F4) says a formula is the last entry of a formation sequence, and from it follows the first-symbol test: $\rightarrow p$ has length 2 and does not begin with $\neg$ or $($.

Exercise. A2: Drill

Answer each part in a line or two.

- (a) Is $\neg\neg\neg p$ a formula?
- (b) Is $(p \wedge (q))$ a formula?
- (c) How many symbols has $((p \rightarrow q) \vee \neg r)$?
- (d) Is $p \wedge q \vdash p$ a formula?
- (e) What is the main connective of $\neg(p \wedge q) \rightarrow r$?
- (f) Which of these are subformulas of $((p \wedge q) \rightarrow r)$: $(p \wedge q)$; the string $q) \rightarrow r$; $(q \rightarrow r)$; r ?

- (g) Add one symbol to $(p \wedge \neg q)$ to make a formula.
- (h) Delete as few symbols as possible from $((\neg p))$ to make a formula.
- (i) Insert one pair of parentheses into $(p \vee q \vee r)$ to make a formula, in two different ways.
- (j) Write the official formula for $\neg p \vee q \rightarrow r$.
- (k) Draw the parse tree of $(p \rightarrow (q \rightarrow p))$ and list its leaves.
- (l) In the derivation of Proposition 1.4.1 printed in this section, put q for P , q for Q and p for R . Is the result a derivation?

Solution.

- (a) Yes: $p, \neg p, \neg\neg p, \neg\neg\neg p$ is a formation sequence.
- (b) No. It would be $(\varphi \wedge \psi)$ with ψ the string (q) . That string begins with $($ and has length 3, and a formula beginning with $($ has length at least 5.
- (c) Ten: $((p \rightarrow q) \vee \neg r)$.
- (d) No. It is a claim of the metalanguage about two formulas; $\vdash$ is not a symbol of the alphabet.
- (e) $\rightarrow$; officially the formula is $(\neg(p \wedge q) \rightarrow r)$.
- (f) $(p \wedge q)$ and r . The string $q) \rightarrow r$ is not a formula. $(q \rightarrow r)$ is a formula, but no node of the tree carries it.
- (g) $(p \wedge \neg q)$: the string begins with $($, so by the first-symbol test it must end with $)$.
- (h) Delete all four parentheses, leaving $\neg p$. The string has no binary connective, and a pair of parentheses is official only around a binary step; so the only formulas made of its symbols are $\neg p$ and p .
- (i) $((p \vee q) \vee r)$ and $(p \vee (q \vee r))$.
- (j) $((\neg p \vee q) \rightarrow r)$.
- (k) Root $\rightarrow$; on the left the leaf p , on the right $\rightarrow$ over q and p . The leaves are p, q, p .
- (l) Yes. Lines 1–6 become $q \rightarrow q, q \rightarrow p, q, q, p, q \rightarrow p$, and lines 4, 5, 6 are still instances of $\rightarrow E, \rightarrow E, \rightarrow I$. A substitution may put the same formula for two letters.

4.1.9.2 B. Practice

Exercise. B1: Trees from Chapter 1

Each formula below appears, with capital letters, in a result of Chapter 1 (Proposition 1.4.2 and Proposition 1.6.1); here the letters are atoms. For each, write the official formula, draw its parse tree, and list its subformulas.

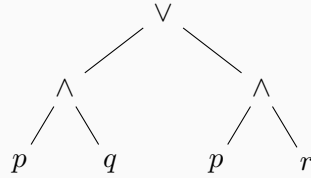
- (a) $(p \wedge q) \rightarrow r$
- (b) $p \rightarrow (q \rightarrow r)$
- (c) $(p \wedge q) \vee (p \wedge r)$

Solution. (a) Official form $((p \wedge q) \rightarrow r)$. Root $\rightarrow$; left subtree $\wedge$ with leaves p, q ; right subtree the leaf r . Subformulas: $((p \wedge q) \rightarrow r), (p \wedge q), p, q, r$.

(b) Official form $(p \rightarrow (q \rightarrow r))$. Root $\rightarrow$; left the leaf p ; right subtree $\rightarrow$ with leaves

q, r . Subformulas: $(p \rightarrow (q \rightarrow r))$, $(q \rightarrow r)$, p, q, r . (This is the left tree of [Example \(Two groupings of three implications\)](#). The formulas of (a) and (b) have different trees; Proposition 1.4.2 says that each can be derived from the other.)

(c) Official form $((p \wedge q) \vee (p \wedge r))$. Root $\vee$; left subtree $\wedge$ with leaves p, q ; right subtree $\wedge$ with leaves p, r .



Subformulas: $((p \wedge q) \vee (p \wedge r))$, $(p \wedge q)$, $(p \wedge r)$, p, q, r : six, with p listed once although it sits at two leaves.

Exercise. B2: Shorthand and back

- Write in shorthand, dropping every pair of parentheses the conventions allow: $((p \wedge q) \rightarrow (r \rightarrow p))$ and $(\neg(p \vee q) \rightarrow (\neg p \wedge \neg q))$.
- Write the official formula for: $p \rightarrow q \vee r \rightarrow \neg p$ and $\neg p \wedge q \leftrightarrow r$.

Solution. (a) $p \wedge q \rightarrow r \rightarrow p$. The outer pair goes by rule 1; $(p \wedge q)$ loses its pair because $\wedge$ binds more tightly than $\rightarrow$ (rule 3); $(r \rightarrow p)$ loses its pair because $\rightarrow$ groups to the right (rule 4).

$\neg(p \vee q) \rightarrow \neg p \wedge \neg q$. The outer pair goes; the pair around $\neg p \wedge \neg q$ goes by rule 3. The pair after the first $\neg$ stays: without it, rule 2 would make the $\neg$ apply to p alone.

(b) $(p \rightarrow ((q \vee r) \rightarrow \neg p))$. Rule 3 groups $q \vee r$ first; rule 4 then makes the first $\rightarrow$ the main connective, with conclusion $((q \vee r) \rightarrow \neg p)$.

$((\neg p \wedge q) \leftrightarrow r)$. Rule 2 makes $\neg$ apply to p alone; rule 3 groups $\neg p \wedge q$ before $\leftrightarrow$.

4.1.9.3 C. Going deeper

Exercise. C1: Forgetting the grouping convention

Suppose we forgot rule 4 of the shorthand, so that $p \rightarrow q \rightarrow r \rightarrow s$ could stand for any official formula obtained by inserting parentheses into it.

- List every official formula it could stand for, and draw the shape of each parse tree.
- How many are there? Say how the count is found, by splitting at the main connective.
- Which of them is the one rule 4 chooses?

Solution (Hint 1). The main connective is one of the three arrows. Choose it first.

Solution (Hint 2). Write c_n for the number of groupings of a chain of n atoms, and find c_1 , c_2 and c_3 first.

Solution. (a) The main connective is one of the three arrows; what lies on each side is then grouped in every possible way.

- Main connective the **first** arrow: p on the left, and $q \rightarrow r \rightarrow s$ on the right, grouped in one of two ways. This gives $(p \rightarrow (q \rightarrow (r \rightarrow s)))$ and $(p \rightarrow ((q \rightarrow r) \rightarrow s))$.
- Main connective the **second** arrow: $(p \rightarrow q)$ on the left and $(r \rightarrow s)$ on the right, each grouped in one way. This gives $((p \rightarrow q) \rightarrow (r \rightarrow s))$.
- Main connective the **third** arrow: $p \rightarrow q \rightarrow r$ on the left, grouped in one of two ways, and s on the right. This gives $((p \rightarrow (q \rightarrow r)) \rightarrow s)$ and $((p \rightarrow q) \rightarrow r) \rightarrow s$.

Their parse trees have the root $\rightarrow$ in each case: in the first two, the leaf p on the left and a chain of two arrows on the right (leaning right, or leaning left); in the third, two one-arrow trees side by side; in the last two, a chain of two arrows on the left and the leaf s on the right.

(b) There are 5. Write c_n for the number of groupings of a chain of n atoms joined by one binary connective. A chain of one or two atoms has one grouping: $c_1 = c_2 = 1$. For three atoms the main connective is the first or the second arrow, and the side with two atoms has one grouping: $c_3 = c_1c_2 + c_2c_1 = 2$. For four atoms, choosing the main connective splits the chain into a left part of k atoms and a right part of $4 - k$ atoms, for $k = 1, 2, 3$, and the two parts are grouped independently:

$$c_4 = c_1c_3 + c_2c_2 + c_3c_1 = 2 + 1 + 2 = 5.$$

The three terms are the three bullets of (a).

(c) Rule 4 groups to the right, so the main connective is the first arrow and its right side is grouped to the right again: $(p \rightarrow (q \rightarrow (r \rightarrow s)))$, the first formula of the list.

The numbers 1, 1, 2, 5, 14, 42, ... produced by this splitting are the **Catalan numbers**; they grow quickly: a chain of eleven atoms joined by ten connectives already has 16,796 readings. Parentheses, or a fixed convention, are not pedantry.