Peter Smith: An Introduction to Formal Logic (2nd edition)

Appendix on Propositional Truth Trees

The formal system in the first edition of *An Introduction to Formal Logic* (CUP 2003) was a truth tree system akin to that in e.g. Richard Jeffrey's *Formal Logic: Its Scope and Limits*. The main text of the second edition, *IFL2* is, however, natural deduction based.

Ideally, say I, students should know about both natural deduction and trees early in their logical education. But an introductory text covering both would begin to look rather dauntingly long. So you have to choose, on a balance of considerations for and against.

I still think there are good reasons on the side of teaching truth trees. So, some might still much rather follow Chapters 1 to 19 of *IFL2* by three chapters on propositional truth trees (as opposed to five chapters on propositional natural deduction). And others might just want to know something about trees. So as an Appendix (or partial alternative) here are rewritten and improved versions of the relevant tree chapters from *IFL1*.

These chapters, however, should also work as a standalone introduction to propositional trees. If you have an introductory knowledge of the PL language(s) of propositional calculus with the standard four connectives, and know about brute-force truth-table testing for (tautological) validity, you should be able to dive straight in and follow the story.

A set of exercises adapted from the exercises from *IFL1* will follow.

Thanks for helpful comments to David Makinson and Rowsety Moid.

Contents

1 Introducing PL truth trees

Assume that we know what it is for PL inferences to be tautologically valid, and know how tautological validity relates to logical validity. *One* way of determining whether an inference is tautologically valid will be familiar, namely by a brute-force truth-table test. In this Appendix, we explore another way of testing for tautological validity – one that is more elegant, usually faster, and considerably less boring!

In this chapter, then, we introduce *truth trees* by talking slowly through a number of examples. We streamline this new method of testing in the next chapter, though still proceeding relatively informally. We give an official summary account in Chapter 3.

1.1 Three very quick preliminaries

(1) In this chapter and the following two, take wffs in all our examples to belong to an interpreted PL language with the built-in connectives $\neg$, $\wedge$, $\vee$ and $\to$ (but, for now, *not* $\bot$, see §3.5 below). Take 'valid'/'invalid' always to mean *tautologically* valid/invalid.

(2) It will be useful to have a word for those wffs which are either atoms or negations of atoms. The conventional but slightly odd term is 'literal'. Let's use instead the more memorable 'simple'. Other wffs are therefore 'complex'.

(3) Most significantly, in this chapter we will usually prefer to say of a wff, not that it is *false* on a given valuation of its atoms, but rather that its *negation is true*. For example, instead of saying that a (tautologically) invalid inference is one where there's a valuation which makes the premisses true and conclusion false, we will now prefer to say that *an inference is (tautologically) invalid just if there is a relevant valuation which makes the premisses and the negation of the conclusion all true*.

Obviously nothing can turn on this alternative way of putting things; in our two-valued logical framework, a wff is false if and only if its negation is true. But there is a reason for this apparently quirky preference for writing things of the form $\neg\alpha :=$ T rather than $\alpha :=$ F. We reveal all in the next chapter.

1.2 'Working backwards'

(a) Considered the following daft inference with fifty premisses – all negated atoms, all different – and an unrelated conclusion:

A $\neg\mathsf{P}, \neg\mathsf{P}', \neg\mathsf{P}'', \neg\mathsf{P}''', \ldots, \neg\mathsf{P}''^{\cdots}{}'' \therefore \mathsf{Q}$

It would be quite crazy to try to write down a full truth table to test the status of this argument (which would take millennia). It is enough to remark that **A** is invalid if and only if there is a valuation of the fifty-one atoms which makes the premisses and negated conclusion all true – i.e. a valuation such that

$$\neg\mathsf{P} := \mathrm{T},\ \neg\mathsf{P}' := \mathrm{T},\ \neg\mathsf{P}'' := \mathrm{T},\ \neg\mathsf{P}''' := \mathrm{T},\ \ldots,\ \neg\mathsf{P}''^{\cdots}{}'' := \mathrm{T},\ \neg\mathsf{Q} := \mathrm{T}.$$

Then we merely have to note that we can immediately read off just such a valuation, namely

$$\mathsf{P} := \mathrm{F},\ \mathsf{P}' := \mathrm{F},\ \mathsf{P}'' := \mathrm{F},\ \mathsf{P}''' := \mathrm{F},\ \ldots, \mathsf{P}''^{\cdots}{}'' := \mathrm{F},\ \mathsf{Q} := \mathrm{F}.$$

Which settles it. **A** is invalid.

(b) Take next the argument

B $(\mathsf{P} \land \neg\mathsf{Q}), (\mathsf{R} \land \neg\mathsf{S}) \therefore (\mathsf{Q} \lor \mathsf{S})$.

This too is obviously invalid, as we can confirm with a sixteen-line truth table. But we can also argue as follows, this time laying out our reasoning step by step.

The inference **B** is invalid if and only if there is a valuation of the four atoms which makes the premisses true and the negation of the conclusion true too, so

(1)	$(\mathsf{P} \land \neg\mathsf{Q}) := \mathrm{T}$
(2)	$(\mathsf{R} \land \neg\mathsf{S}) := \mathrm{T}$
(3)	$\neg(\mathsf{Q} \lor \mathsf{S}) := \mathrm{T}$

This time, we can't instantly read off a valuation which makes (1) to (3) hold. But note that by the truth table for conjunction, (1) holds if and only if these claims hold too:

(4)	$\mathsf{P} := \mathrm{T}$
(5)	$\neg\mathsf{Q} := \mathrm{T}$

Similarly, (2) holds if and only if these claims do too:

(6)	$\mathsf{R} := \mathrm{T}$
(7)	$\neg\mathsf{S} := \mathrm{T}$

Finally, (3) holds if and only if the plain disjunction '$(\mathsf{Q} \lor \mathsf{S})$' is false, which requires that both disjuncts are false. Avoiding talk of falsehood, this means that the negation of each disjunct has to be true:

(8)	$\neg\mathsf{Q} := \mathrm{T}$
(9)	$\neg\mathsf{S} := \mathrm{T}$

So, putting everything together, we can ensure that (1) to (3) hold – i.e. we can make the premisses and negated conclusion of **B** all true – by taking the valuation where $\mathsf{P} := \mathrm{T}$, $\neg\mathsf{Q} := \mathrm{T}$ (i.e. $\mathsf{Q} := \mathrm{F}$), $\mathsf{R} := \mathrm{T}$, and $\neg\mathsf{S} := \mathrm{T}$ (i.e. $\mathsf{S} := \mathrm{F}$). Hence **B** is indeed invalid.

(c) For a contrasting third example, consider

C $(\mathsf{P} \land \neg\mathsf{Q}) \therefore \neg(\mathsf{Q} \land \mathsf{R})$.

Again, the inference is invalid just if there is a valuation of the three atoms here which makes the premiss and negated conclusion both true, i.e. which satisfies

(1) $(\mathsf{P} \land \neg\mathsf{Q}) := \mathrm{T}$
(2) $\neg\neg(\mathsf{Q} \land \mathsf{R}) := \mathrm{T}$

For (1) to obtain, we need

(3) $\mathsf{P} := \mathrm{T}$
(4) $\neg\mathsf{Q} := \mathrm{T}$

By the truth table for negation applied twice, for (2) to obtain, we need

(5) $(\mathsf{Q} \land \mathsf{R}) := \mathrm{T}$

and hence we need both

(6) $\mathsf{Q} := \mathrm{T}$
(7) $\mathsf{R} := \mathrm{T}$

But hold on! We have now hit a contradiction. *No valuation can make* $\neg\mathsf{Q}$ *and* Q *simultaneously true together*. So there can not possibly be a valuation satisfying both (1) and (2) and making the premiss of **C** true and conclusion false.

Which shows that **C** is valid.

(d) In these three examples we have been – as we put it before – *working backwards*.

On the brute-force truth-table method, in order to decide whether a given argument is tautologically valid, we start by laying out all the combinatorially possible valuations for the relevant atoms. Then we plod forwards through each valuation, calculating the truth values of the premisses and conclusion as we go, looking for a 'bad' line. But now – as already sketched in the preceding Interlude – we are starting on the same decision task from the other end. We suppose that there *is* a bad valuation, i.e. one where the premisses are true and the negation of the conclusion is true too. Then we try to unravel the implications of this supposition, working backwards towards uncovering a valuation of the relevant atoms that is indeed bad.

If we *can* uncover a bad valuation, then obviously the argument being tested is *not* valid. On the other hand, if we get inextricably entangled in contradiction as we work backwards, then (by an informal reductio ad absurdum inference) that shows that there is no bad valuation after all, and the argument in question is *valid*.

This is, in headline terms, the appealing strategy that we are exploring in this chapter. Though as we will see in the next section, things don't in general go quite as simply as our first easy examples might suggest.

(e) Before moving on, one quick comment about our example **C**. Note that at line (2) we gave the negation of the conclusion '$\neg(\mathsf{Q} \land \mathsf{R})$' as '$\neg\neg(\mathsf{Q} \land \mathsf{R})$' with its extra prefixed negation sign. Then at line (5) we stripped away the double negation again by using the obvious rule that tells us that $\neg\neg\alpha := \mathrm{T}$ holds just when $\alpha := \mathrm{T}$. Why go through this little two-step dance? Why not immediately jump to treating the negation of something of the form $\neg\alpha$ as plain α?

Because, in more advanced work (beyond the scope of this book), you will eventually encounter important logical systems with a non-standard understanding of how negation works. In particular, you will meet so-called intuitionistic logic where $\neg\neg\alpha$ is in general not equivalent to α. It is therefore *not* a good idea to get into the habit of simply ignoring double negations, even if that is permitted in standard two-valued logic.

For us, remember, negating a PL wff is *always* a matter of *adding* a prefixed negation sign.

1.3 Why trees?

(a) Here is an obviously invalid little argument:

D $(\mathsf{P} \lor \mathsf{Q}) \mathrel{\therefore} \mathsf{P}.$

What happens if we try working backwards to expose a valuation which makes the premiss true and negated conclusion true too?

We start, as before, with the assumption that there *is* a bad valuation, one which makes the following hold:

(1) $(\mathsf{P} \lor \mathsf{Q}) := \mathrm{T}$

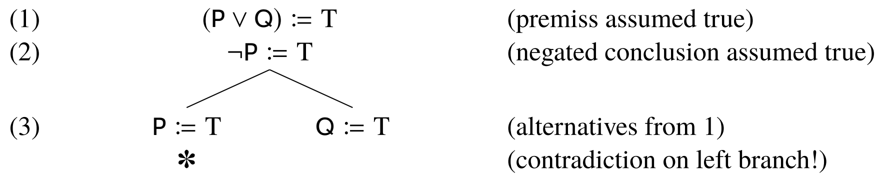
(2) $\neg\mathsf{P} := \mathrm{T}$

Given the truth table for disjunction, (1) tells us that either (3a) $\mathsf{P} := \mathrm{T}$ or (3b) $\mathsf{Q} := \mathrm{T}$ or perhaps both – and either alternative suffices to verify the disjunction. *But (1) doesn't tell us which alternative holds*. We will have to consider the two cases in turn.

In fact, the alternative (3a) where $\mathsf{P} := \mathrm{T}$ need not detail us long, for it is immediately inconsistent with (2), i.e. with $\neg\mathsf{P} := \mathrm{T}$.

The alternative (3b) where $\mathsf{Q} := \mathrm{T}$ fares better; there is no contradiction to be found this time. The resulting valuation on which $\neg\mathsf{P} := \mathrm{T}$ (i.e. $\mathsf{P} := \mathrm{F}$), $\mathsf{Q} := \mathrm{T}$ makes the premiss of **D** true and the negated conclusion true too. As we knew all along, then, the inference **D** is indeed invalid.

(b) We want a perspicuous way of setting out reasoning like this when our line of argument forks, and we have to explore alternatives. The obvious device is to draw forking branches in our chains of reasoning. We then get (upside down, trunk-at-the-top) *truth trees*, and can display our reasoning about **D** like this:


(1) $(\mathsf{P} \lor \mathsf{Q}) := \mathrm{T}$ (premiss assumed true)
(2) $\neg\mathsf{P} := \mathrm{T}$ (negated conclusion assumed true)
(3) $\mathsf{P} := \mathrm{T}$ $\mathsf{Q} := \mathrm{T}$ (alternatives from 1)
✻ (contradiction on left branch!)

This mode of presentation should be self-explanatory. The main novelty compared with our informal reasoning a moment ago is that, instead of considering options sequentially, we consider them side-by-side; we split our reasoning into left and right branches as we consider the alternative ways of making (1) come out true.

The other novelty is that we now begin to use '✻' as an *absurdity sign*. But NB, it is not a wff like '$\perp$'; it is just quasi-punctuation – like a bold exclamation mark (see §3.5) – signalling that we have hit a contradiction on a path down the truth tree. In other words, '✻' at the end of a path indicates that it features a pair of wffs α and $\neg\alpha$ which are both assigned T (note that the wff α here doesn't have to be an atom).

Our miniature downward branching tree for argument **D**, then, has two paths through it running from top to bottom, as we explore alternative potential ways of making true the initial claims at the top of its trunk. The left-hand path is, as we will say, *closed* with an absurdity marker; but the other path remains *open*. And from the open path we can read off a perfectly consistent assignment of values to the relevant atoms which satisfies (1) and (2), making **D** invalid.

(c) Take next the following argument:

E $\quad \mathsf{P}, \neg(\mathsf{P} \land \neg\mathsf{Q}) \therefore \mathsf{Q}.$

This is invalid if and only if we can consistently make the premisses and negated conclusion all true, as follows:

(1)	$\mathsf{P} := \mathrm{T}$	(premiss assumed true)
(2)	$\neg(\mathsf{P} \land \neg\mathsf{Q}) := \mathrm{T}$	(premiss assumed true)
(3)	$\neg\mathsf{Q} := \mathrm{T}$	(negated conclusion assumed true)

Now, a conjunction is false when at least one conjunct is false. Equivalently, a negated conjunction is true just when at least one negated conjunct is true. In other words, given $\neg(\alpha \land \beta) := \mathrm{T}$ we must have at least one of $\neg\alpha := \mathrm{T}$ or $\neg\beta := \mathrm{T}$. This is the principle we can apply in moving from (2) to the alternatives at (4):

(4)	$\neg\mathsf{P} := \mathrm{T}$	$\neg\neg\mathsf{Q} := \mathrm{T}$	(alternatives from 2)
	✻	✻	

This time *both* alternatives immediately produce contradictory assignments of truth values ($\mathsf{P} := \mathrm{T}$ and $\neg\mathsf{P} := \mathrm{T}$ on the left-branching path from top to bottom; $\neg\mathsf{Q} := \mathrm{T}$ and $\neg\neg\mathsf{Q} := \mathrm{T}$ on the right-branching path). So both paths get closed off with the absurdity sign. Which shows that neither of the potential ways of satisfying conditions (1) to (3) is consistent. Therefore the inference **E** has to be valid.

1.4 Two more examples

(a) For our next example, take the argument

F $\quad (\mathsf{P} \to \mathsf{Q}), (\mathsf{Q} \to \mathsf{R}) \therefore (\mathsf{P} \to \mathsf{R}).$

We can show this to be valid by a brute-force test. But we will now try to get the same result by 'working backwards': so we assume the argument *isn't* valid, and aim for absurdities. We start, then, by assuming that there is a valuation satisfying

(1)	$(\mathsf{P} \to \mathsf{Q}) := \mathrm{T}$	(premiss assumed true)
(2)	$(\mathsf{Q} \to \mathsf{R}) := \mathrm{T}$	(premiss assumed true)
(3)	$\neg(\mathsf{P} \to \mathsf{R}) := \mathrm{T}$	(negated conclusion assumed true)

Now, a material conditional is true, remember, if either its antecedent is false or consequent is true. So, avoiding talk of falsity, we have the following unpacking rule:

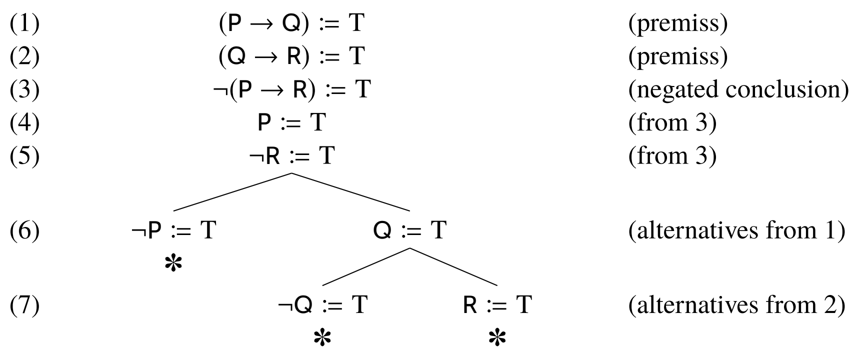
Given $(\alpha \to \beta) := \mathrm{T}$, we have to consider the alternatives $\neg\alpha := \mathrm{T}$ or $\beta := \mathrm{T}$.

And what are the implications of assuming a negated material conditional is true, i.e. that the conditional is false? This requires the conditional's antecedent to be true and conclusion false. Again avoiding talk of falsity, the corresponding unpacking rule is as follows:

Given $\neg(\alpha \to \beta) := \mathrm{T}$, then we must have both $\alpha := \mathrm{T}$ and $\neg\beta := \mathrm{T}$.

So which unpacking rule – to continue with that vivid metaphor – shall we apply first in constructing our tree? Let's start by applying the non-branching rule to (3); this delays having to consider branching possibilities and keeps our tree tidier. As we will confirm in §3.2(d), the order in which we tackle complex wffs can't make any difference to the eventual verdict on the argument which we are testing with a tree.

Following that plan, here's our completed truth tree:


(1) (P → Q) := T (premiss)
(2) (Q → R) := T (premiss)
(3) ¬(P → R) := T (negated conclusion)
(4) P := T (from 3)
(5) ¬R := T (from 3)
(6) ¬P := T
Q := T (alternatives from 1)
✻
(7) ¬Q := T
R := T (alternatives from 2)
✻
✻

First we draw out the implications of (3). Then at (6) we consider the implications of (1), exploring alternatives. The alternative on the left-hand path, though, doesn't give us a consistent way of ensuring (1) and (3) hold: so we can close off that path.

The right-hand path stays open at (6). But we haven't yet drawn out the implications of (2). This requires us to consider alternatives again, leading to our second fork in the tree. And either way, we immediately hit a contradiction with what's further up the path.

Which all goes to show that there is no consistent way of working backwards to make (1), (2) and (3) all correct. Hence, as we want, **F** is shown to be valid.

(b) An aside. Suppose we are given $(\alpha \to \beta) := \mathrm{T}$. It follows that one of these *three* options must hold: $\alpha := \mathrm{T}, \beta := \mathrm{T}$ or else $\neg\alpha := \mathrm{T}, \beta := \mathrm{T}$ or else $\neg\alpha := \mathrm{T}, \neg\beta := \mathrm{T}$ (corresponding to the three lines on the truth table where the material conditional is true). So why not adopt an unpacking rule for $(\alpha \to \beta) := \mathrm{T}$, with three-way branching?

Aesthetics! It is much neater to keep our trees binary if we can, with only two branches starting at any point where we have to explore alternatives. So this is what we will do.

(c) We take just one more example for now, in order to highlight two crucial points about constructing branching trees like these. So consider:

G $((\mathsf{P} \wedge \mathsf{Q}) \vee \mathsf{R})\ \therefore\ \neg(\neg\mathsf{P} \vee \neg\mathsf{R})$.

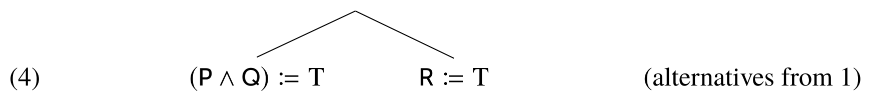
Is this valid? We look to see whether there is a valuation which makes the premiss and the negated conclusion both true, i.e. a valuation which gives us

(1)	$((\mathsf{P} \land \mathsf{Q}) \lor \mathsf{R}) := \mathrm{T}$	(premiss)
(2)	$\neg\neg(\neg\mathsf{P} \lor \neg\mathsf{R}) := \mathrm{T}$	(negated conclusion)

Trivially, (2) holds just when we have

(3)	$(\neg\mathsf{P} \lor \neg\mathsf{R}) := \mathrm{T}$	(from 2)

To unpack the implications of (1) we now need to consider alternative scenarios:


(4)
$(\mathsf{P} \land \mathsf{Q}) := \mathrm{T}$
$\mathsf{R} := \mathrm{T}$
(alternatives from 1)

For '$(\mathsf{P} \land \mathsf{Q})$' to be true, both the conjuncts have to be true. So we can continue the left branch as follows:

(5)	$\mathsf{P} := \mathrm{T}$	(from 4)
(6)	$\mathsf{Q} := \mathrm{T}$	(from 4)

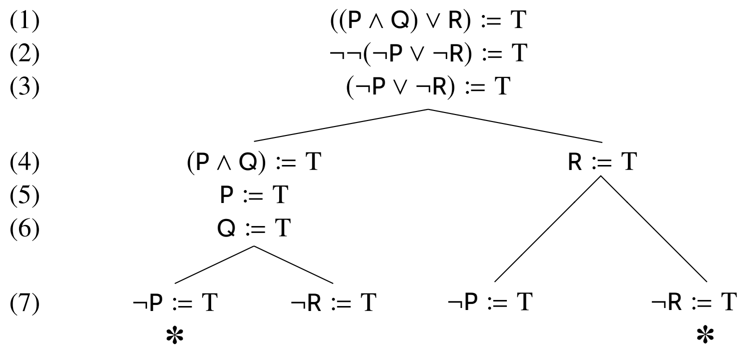
But note – and this illustrates the first obvious point we need to emphasize – these additional conditions '$\mathsf{P} := \mathrm{T}$' and '$\mathsf{Q} := \mathrm{T}$' *are added to the left branch only*. For it is only on the scenario described on the path going left at (4) that we are assuming '$(\mathsf{P} \land \mathsf{Q}) := \mathrm{T}$' holds; and so it is only while further elaborating *this* scenario that we are committed to the implications of that assumption.

Generalizing, we have this obvious but crucial rule:

> Suppose a statement $\alpha := \mathrm{T}$ appears on a tree, with α complex. When we unpack the (perhaps alternative) implications of this statement, we add the results *only* to the open paths through the tree which actually contain the statement!

(d) Continuing with our example, we haven't yet unpacked (3) '$(\neg\mathsf{P} \lor \neg\mathsf{R}) := \mathrm{T}$'. But note that this assumption sits both on the path going left from (1) to (6) and also on the path going right from (1) to (4). Since (3) is in play whichever route down the tree we follow, we need to go on to consider its implications either way. And to keep track of when we have fully unpacked the implications of assigning truth to some wff, the simplest policy is to add the relevant implications to all relevant paths at the same time. This is the policy we adopt.

So the next step in continuing our tree produces this:


(1)
(2)
(3)
(4)
(5)
(6)
(7)
$((\mathsf{P} \land \mathsf{Q}) \lor \mathsf{R}) := \mathrm{T}$
$\neg\neg(\neg\mathsf{P} \lor \neg\mathsf{R}) := \mathrm{T}$
$(\neg\mathsf{P} \lor \neg\mathsf{R}) := \mathrm{T}$
$(\mathsf{P} \land \mathsf{Q}) := \mathrm{T}$
$\mathsf{P} := \mathrm{T}$
$\mathsf{Q} := \mathrm{T}$
$\neg\mathsf{P} := \mathrm{T}$
✻
$\neg\mathsf{R} := \mathrm{T}$
$\mathsf{R} := \mathrm{T}$
$\neg\mathsf{P} := \mathrm{T}$
$\neg\mathsf{R} := \mathrm{T}$
✻

At line (7), then, we have added the alternatives generated by (3) to *both* open paths containing (3).

Generalizing gives us the second key point we need to emphasize:

> Suppose a statement $\alpha := \mathrm{T}$ appears on a tree, with α complex. When we unpack the (perhaps alternative) implications of this statement, we will add the results to *all* the open paths through the tree which actually contain the statement.

(e) Now to conclude work on our example (continuing to spell things out in detail). Two of the four extended paths generated by unpacking (3) to get to line (7) can be closed off. On the other two paths, there aren't any complex wffs still to unpack, so there are no more lurking contradictions to be exposed – therefore *those* two paths cannot be made to close. Let's take them in turn.

The left-hand open path contains the assignment of values $\mathsf{P} := \mathrm{T}$, $\mathsf{Q} := \mathrm{T}$, $\neg\mathsf{R} := \mathrm{T}$ (i.e. $\mathsf{R} := \mathrm{F}$). This valuation gives us one way of making (1) and (2) correct, i.e. one way of making **G**'s premiss true and its negated conclusion true too. That's enough already to show that **G** is *not* valid.

But to continue, the right-hand open path contains the assignment $\neg\mathsf{P} := \mathrm{T}$ (i.e. $\mathsf{P} := \mathrm{F}$), and $\mathsf{R} := \mathrm{T}$. Those values also make (1) and (2) correct, however we settle the value of 'Q' (why?). Hence, depending on the choice we then make for 'Q', this gives us two more ways of making **G**'s premiss true and its negated conclusion true too.

So, in total, our tree reveals *three* different bad valuations of the three relevant atoms making the argument's premiss true and conclusion false. Check that this agrees with the verdict of a brute-force truth table.

1.5 Trees and consistency

A final note in this chapter, just to make explicit a simple point about what we are doing here.

To say that the PL argument with premisses $\alpha_1, \alpha_2, \ldots \alpha_n$ and conclusion γ is invalid is just to say that there's a valuation which makes each of $\alpha_1, \alpha_2, \ldots \alpha_n, \neg\gamma$ simultaneously true, i.e. to say that those wffs are consistent (it should be a familiar point that we can trade in claims about validity for claims about consistency). And in fact our 'working backwards' test to decide the *validity* of PL arguments is, more generally, a test for the *consistency* of a bunch of PL wffs. We assume the wffs in question are consistent and aim to work backwards to uncover a valuation which shows that they really are consistent. If we can do so without hitting contradictions at every turn then, yes, those wffs *are* consistent. Otherwise they aren't.

1.6 Summary

Let's pause for a brisk half-time summary about the tree test for tautological validity. (We will be doing more work to elaborate and really nail down the following claims – but hopefully these headlines should already look plausible.)

We can test an inference for tautological validity by assuming that it is invalid, i.e. by assuming there is a 'bad' valuation which makes its premisses true and negated conclusion true too, and then working backwards, trying to uncover a consistent valuation which warrants that assumption.

In working backwards, we move at each stage from the assignment of truth to some complex wff to corresponding assignments of truth to some shorter subformula(s) – though we may have to consider alternatives. When we have to consider alternatives, our reasoning is naturally set out in the form of a downward branching tree.

If working backwards, perhaps through alternatives, always leads to contradictions, then we know that the original assumption that the inference is invalid has to be incorrect. In other words, if all paths through the tree setting out our working can be closed off with absurdity signs, then the argument is *valid*.

Conversely, if at least one alternative path through the tree stays open, i.e. involves no contradiction (even when we have unpacked all the implications of assignments of truth to complex wffs on the path), then the argument is indeed *invalid*.

2 Unsigned trees

In the last chapter, our 'working backwards' arguments to show tautological validity/ invalidity are regimented as *signed* trees (we will speak of 'trees' even when, as in our very first examples in §1.2, we actually have only a bare trunk without any branching). The trees are 'signed' in the sense that the wffs that appear on the trees are all *explicitly* assigned a truth value.

However, in our particular version of signed trees, the wffs are all assigned T – that's because, whenever we might have been tempted to say a wff was false, we instead said that its negation is true. The pay-off from this little bit of trickery is that we can now move to so-called *unsigned* truth trees, where wffs appear 'naked'. The move couldn't be simpler! *We just delete* ':= T' *wherever it appears on one of our signed trees*.

2.1 Rules for negation and conjunction

(a) The principles that we have been using for unpacking the implications of complex PL wffs assigned T on a signed tree can now become rules for dealing with 'naked' wffs on an unsigned tree. We have already met all the needed rules in their signed form. But in this chapter, we will (re-)present the rules as rules for unsigned trees, and in a more systematic order.

Sticking with the 'unpacking' metaphor, let's call the wff that we are unpacking at any stage the *parent* wff, and then call the shorter wffs we add its *children*. As before, we require our unpacking rules to ensure the following:

Parents-to-children When we apply a non-branching rule, if the parent wff is true (on some supposed valuation) then each new added child wff is true too. When we apply a branching rule, if the parent wff is true then, on at least one branch of each added fork, the new child wff is true too.

Let's start then with the rules for unpacking complex wffs of one of the following three shapes: $\neg\neg\alpha$, $(\alpha \wedge \beta)$, $\neg(\alpha \wedge \beta)$. In diagrammatic form, the rules are:

$$\begin{tikzcd}[column sep=small] (\neg\neg) & \neg\neg\alpha \arrow[d, dash] \\ & \alpha \end{tikzcd} \qquad \begin{tikzcd}[column sep=small, row sep=small] (\wedge) & (\alpha \wedge \beta) \arrow[d, dash] \\ & \alpha \\ & \beta \end{tikzcd} \qquad \begin{tikzcd}[column sep=small] (\neg\wedge) & & \neg(\alpha \wedge \beta) \arrow[dl, dash] \arrow[dr, dash] & \\ & \neg\alpha & & \neg\beta \end{tikzcd}$$

We read these diagrams as you would now expect. But to laboriously spell things out:

The rule (¬¬) tells us that, given a particular wff of the form $\neg\neg\alpha$ on the tree (invisibly assigned T), we can unpack the double negation by adding α (invisibly assigned T) to the tree. Add where? As explained in §1.4(c) and (d), we add a child α at the foot of every open path containing the parent wff we are unpacking.

The rule (∧) tells us that, given a parent wff of the form $(\alpha \wedge \beta)$ (invisibly assigned T), we unpack the conjunction by adding both children α and β (each invisibly assigned T), again at the foot of every open path containing the parent wff.

The rule (¬∧) tells us that given a parent wff of the form $\neg(\alpha \wedge \beta)$, then we unpack this by adding a fork with a branch to the child $\neg\alpha$ and a branch to the child $\neg\beta$, again at the foot of every open path containing the parent wff.

Each rule obviously satisfies Parents-to-children.

(b) Let's construct, then, an unsigned truth tree to test the argument

A ((P ∧ Q) ∧ R) ∴ (Q ∧ (R ∧ P)).

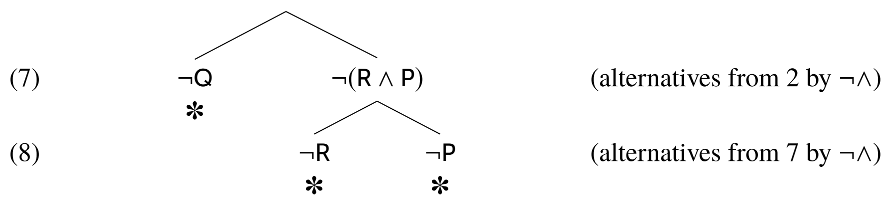
We start by assuming that there is valuation which makes the premiss true and negation of the conclusion true, so now write down those wffs unsigned at the top of the trunk:

(1)	((P ∧ Q) ∧ R)	(premiss)
(2)	¬(Q ∧ (R ∧ P))	(negated conclusion)

We follow the tactic of keeping our tree tidy by applying a non-branching rule before a branching rule when we have a choice. So we continue by adding more wffs which must also be true (on the same supposed valuation):

(3)	(P ∧ Q)	(from 1 by ∧)
(4)	R	(from 1 by ∧)
(5)	P	(from 3 by ∧)
(6)	Q	(from 3 by ∧)

Next, we apply our branching rule twice to extract alternatives from (2):

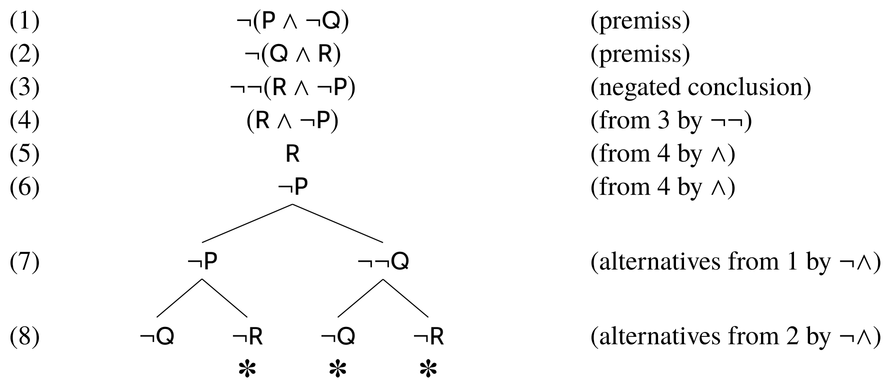


Each path leads to contradiction, since it contains a wff and its negation (both invisibly assigned T). Hence the assumption that **A**'s premiss and negated conclusion are both true leads at every turn to absurdity. Hence **A** is valid. Just as you will have hoped!

(c) Next, consider the argument

B ¬(P ∧ ¬Q), ¬(Q ∧ R) ∴ ¬(R ∧ ¬P).

You know now how a truth tree will start. And here's one finished tree:


(1) ¬(P ∧ ¬Q) (premiss)
(2) ¬(Q ∧ R) (premiss)
(3) ¬¬(R ∧ ¬P) (negated conclusion)
(4) (R ∧ ¬P) (from 3 by ¬¬)
(5) R (from 4 by ∧)
(6) ¬P (from 4 by ∧)
(7) ¬P ¬¬Q (alternatives from 1 by ¬∧)
(8) ¬Q ¬R ¬Q ¬R (alternatives from 2 by ¬∧)
✻ ✻ ✻

We have once more applied non-branching rules first, starting with (¬¬) to get line (4), and then applying (∧) to get the next two lines. Then we applied our branching rule (¬∧) to extract the implications of (1) and (2). (Experiment: what happens if we unpack (1), (2) and (3) in different orders: which order leads to the neatest tree, and why?)

This tree is now finished in the sense that no complex wff on the open path remains to be dealt with. – we have already fully drawn out the implications of each of (1) to (4). And we are left with an open path where there is no more unpacking to be done. So there can be no lurking contradictions yet to be exposed. This shows that **B** is invalid.

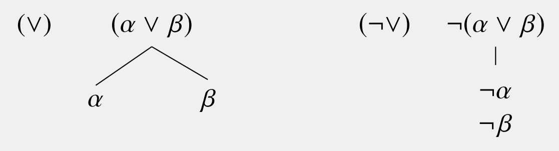
Why so? Well, by inspection you will see that our three rules so far not only satisfy the Parents-to-children principle, but also a kind of converse:

Children-to-parents Whenever we apply a rule, whether it is branching or non-branching, if a parent's one or two resulting children on a particular path through the tree are true (on some supposed valuation), so too is the parent wff.

Therefore, as we go up some particular path on a tree constructed using our rules so far, if all the children we encounter on that path are true, so too are their parents (and their parents' parents, etc.). So in our current example, if we take the valuation making the simple wffs '¬P', '¬Q', and 'R' on the open path all true, then this will make the parents of these wffs true, and those parents will make *their* parents (if any) true, thereby making the complex wffs (1), (2) and (3) all true together. Which confirms that **B** is invalid.

2.2 Rules for disjunction

(a) Next, here are the two rules for unpacking disjunctions, unnegated and negated:

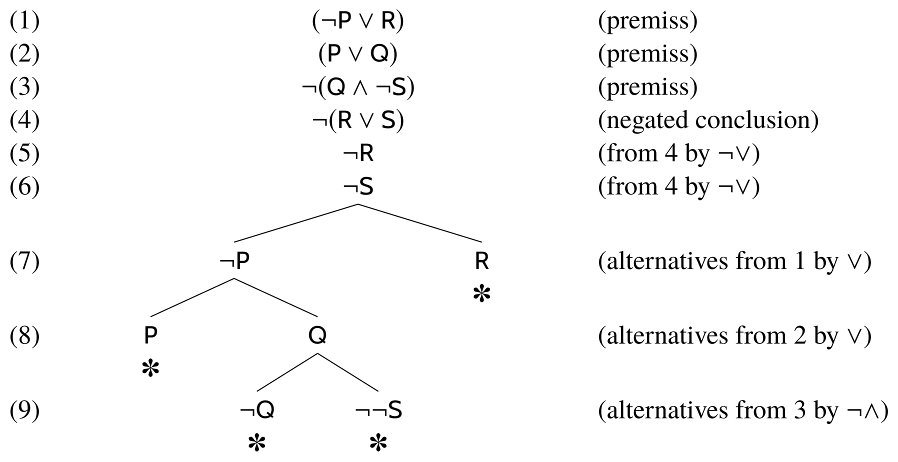


The branching rule (∨) is of course to be interpreted like the branching rule (¬∧); and the non-branching rule (¬∨) is to be interpreted like the non-branching rule (∧). And our rules obviously continue to satisfy Parents-to-children and Children-to-parents.

We can now apply these disjunction rules to construct an unsigned truth tree to test e.g. the following for validity:

C (¬P ∨ R), (P ∨ Q), ¬(Q ∧ ¬S) ∴ (R ∨ S).

And here is one possible result (if we follow the usual tactic of applying non-branching rules first when we can, to keep the tree tidy):


(1) (¬P ∨ R) (premiss)
(2) (P ∨ Q) (premiss)
(3) ¬(Q ∧ ¬S) (premiss)
(4) ¬(R ∨ S) (negated conclusion)
(5) ¬R (from 4 by ¬∨)
(6) ¬S (from 4 by ¬∨)
(7) ¬P R (alternatives from 1 by ∨)
✻
(8) P Q (alternatives from 2 by ∨)
✻
(9) ¬Q ¬¬S (alternatives from 3 by ¬∧)
✻ ✻

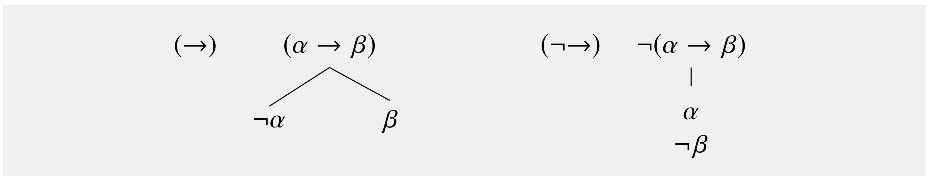
We have assumed that some valuation makes the premisses and negated conclusion of **C** all true, and worked backwards to try to uncover a valuation of simple wffs which makes this the case. But we hit contradiction every way we turn. So there is no such valuation. Hence the argument **C** is valid.

(b) We have continued to put line numbers on the left of our trees and put some minimal commentary on the right. But these are optional extras. Strip them away, and the uncommented tree which shows argument **C** is valid is very quick and neat.

We have in fact seen this very same argument before in *IFL2*, §15.5(b). It was there labelled **E** and we ran a truth-table test on it. So we are now in a position to compare the amount of work involved in the two tests for tautological validity applied to the same simple argument. And we see that in the time it takes to write down just the frame of the truth-table test (writing the premisses and conclusion across the top, and the catalogue of sixteen possible valuations of the atoms down the left side), i.e. before we have done *any* real work filling in the table by evaluating premisses and conclusions, the corresponding (uncommented) tree test for validity will be finished. That's not at all untypical.

2.3 Rules for the material conditional

We will now need two additional rules for dealing with material conditionals, unnegated and negated. We've seen the rules in their signed form before in §1.4(a), so we simply need to restate them in their unsigned form. Interpret the following diagrams for rules just like the previous cases, and note that the two Parent/Children principles hold again:



(a) Let's work through an example exploiting these rules. But now we speed things up by dropping the explicit running commentary on the right. So consider

D ((P → Q) → (R → ¬S)), (¬R → (Q ∧ P′)), (¬P → P′) ∴ (S → P′).

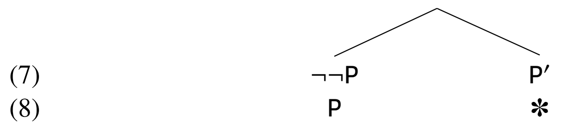
The inference is invalid, then, if there is a valuation which makes the following all true:

(1) ((P → Q) → (R → ¬S))
(2) (¬R → (Q ∧ P′))
(3) (¬P → P′)
(4) ¬(S → P′)

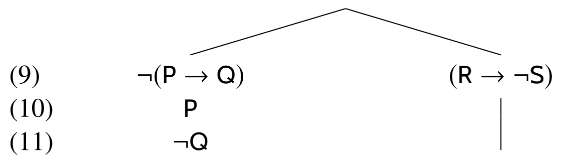
Applying the tactic of applying non-branching rules first, we start 'working backwards' by using rule (¬→) on (4). So these too must be true on the supposed valuation:

(5) S
(6) ¬P′

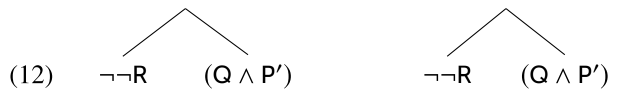
Let's deal with (3) next, because a moment's thought tells us that one resulting branch will immediately close. Then unpacking the wff on the other branch, we get:



We look next at the implications of assuming (1) is true on our supposed valuation, and then apply our non-branching rule (¬→) on the left:



Which leaves just the wff at (2) and the wff on the right-hand branch at (9) to unpack. Let's deal with the first, and add the results of applying the rule (→) to both open paths:



Which yields four open paths through the tree, and on each path there remain complex wffs still to be unpacked. Take the left-most path first. We can continue it thus:

(13) R

And now there are no complex wffs *on this left-most path* still to be unpacked; so there can be no lurking contradiction to be found, at least on *that* path. So let's make the simple wffs on that path all true, by putting $\mathsf{P} := \mathrm{T}$, $\mathsf{Q} := \mathrm{F}$, $\mathsf{R} := \mathrm{T}$, $\mathsf{S} := \mathrm{T}$, $\mathsf{P}' := \mathrm{F}$. By repeated applications of the Children-to-Parents principle, this valuation will make the more complex wffs found as we go up that path all true too. In particular, this valuation makes (1) to (4) at the top of the path true, showing that argument **D** is invalid.

If – as is usually the case – all we care about is a verdict on validity, we can now stop work on the tree. Finding one bad line on a truth table is enough to show the argument being tested in invalid. It is exactly similar here: finding one open branch on which every complex wff has been unpacked is enough to show that the argument being tested is invalid.

(b) We have now met seven unpacking rules for dealing with complex PL wffs on an unsigned tree. Do we need any more?

No. Any complex wff – i.e. any wff other than an atom or negated atom – falls into one of two classes. Either (i) it has one of the three binary connectives as its main connective. Or else (ii) it begins with a negation sign and then the main connective of the rest of the wff is either a binary connective or negation again. So (i) covers three kinds of wff, and (ii) covers another four. We need seven rules for dealing with complex wffs of these seven kinds. Which is exactly what we now have.

2.4 'Checking off'

(a) We should now highlight an obvious but crucial point.

Suppose that we have a currently open path as we work through a truth tree starting with the premisses and negation of the conclusion of an argument. This doesn't yet settle anything, if there is still a complex wff on the path that hasn't yet been unpacked.

Why? Because that wff's implications could – for all we yet know – generate a contradiction, closing the path. Hence, before we can conclude anything about an argument from a currently open path on its tree, we do have to check that *every* complex wff on the path really has been taken count of.

To make this check easier, it is very useful to have a device for signalling quite explicitly when a complex wff has been unpacked. The standard convention is to *check off* a wff at the point when we applying an unpacking rule to it.

(b) Now, suppose we have already applied the appropriate rule, adding children to every open path containing the parent wff we are unpacking. Then it would be pointlessly redundant to apply the rule to the very same wff again. It would be worse than redundant to get 'stuck in a loop', repeatedly re-applying the same rule to the same wff, time without end. Henceforth, then, we adopt the following rule:

When the appropriate unpacking rule has been applied to a complex wff, we *check off* the wff (with a '✓'). A wff, once checked off, is then treated as 'used up', i.e. as no longer available to have one of the unpacking rules (re)applied to it.

(c) Let's have a mini-example of checkmarks in use. So consider again

E $((\mathsf{P} \land \mathsf{Q}) \lor \mathsf{R}) \;\therefore\; \neg(\neg\mathsf{P} \lor \neg\mathsf{R})$.

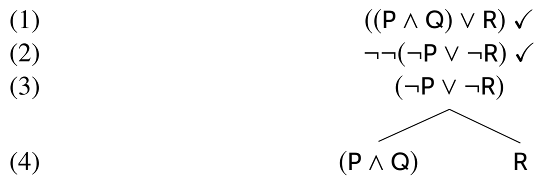
This was argument **G** in the last chapter, where we used a signed tree to test it. Now we use an unsigned tree, checking off complex wffs as we go. So start the tree with

(1) $((P \land Q) \lor R)$
(2) $\neg\neg(\neg P \lor \neg R)$

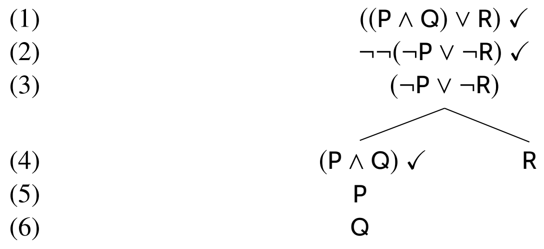
Applying the unpacking rule (¬¬), this becomes

(1) $((P \land Q) \lor R)$
(2) $\neg\neg(\neg P \lor \neg R)$ ✓
(3) $(\neg P \lor \neg R)$

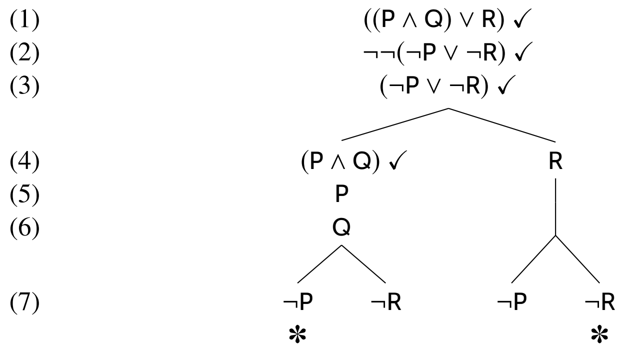
And then applying the unpacking rule (∨) to the first wff we get



Next, using the non-branching rule (∧) on the left branch, the tree becomes



Now, in this simple little example – even if we had not been explicitly checking wffs off as we unpack their implications – we would hardly be likely to overlook the fact that we haven't yet dealt with the third wff! But the use of check marks makes this quite unmissable. So we continue:



And *now* we are done. There are two open paths through the tree on which every complex wff is explicitly checked off. Therefore, as we already know, **E** is invalid.

2.5 Trees for tautologies

(a) We can test whether an argument $\alpha_1, \alpha_2, \ldots, \alpha_n \therefore \gamma$ is tautologically valid by putting all the αs and the negated conclusion $\neg\gamma$ at the top of a truth tree, and then using the tree-building rules. The argument is tautologically valid if and only if every path through a tree headed by the αs and $\neg\gamma$ closes.

A tautology, we remarked in *IFL2* §16.3, can be thought of as the conclusion of a tautologically valid argument with no premisses. This immediately tells us how to use a truth tree to determine whether a wff γ is a tautology. We put its negation $\neg\gamma$ at the top of a truth tree, and use the tree-building rules as before. Then γ is a tautology if and only if every path through a tree headed by $\neg\gamma$ closes.

(b) For a first example, take the wff

F $\quad ((\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})) \lor (\neg\mathsf{P} \land (\neg\mathsf{Q} \lor \neg\mathsf{R})))$.

Is this a tautology? To apply the tree test, we start with the wff's negation (quietly assigned the value T):

(1) $\quad \neg((\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})) \lor (\neg\mathsf{P} \land (\neg\mathsf{Q} \lor \neg\mathsf{R})))$

We then work backwards to determine whether this assumption in fact leads to contradiction. This wff has the form $\neg(\alpha \lor \beta)$, where α is '$(\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R}))$' and β is '$(\neg\mathsf{P} \land (\neg\mathsf{Q} \lor \neg\mathsf{R}))$'. So we apply the non-branching rule $(\neg\lor)$ to (1) to get two derived wffs $\neg\alpha$ and $\neg\beta$ at lines (2) and (3). We then continue in the obvious way by unpacking (2) to get as far as the following:

(1) $\quad \neg((\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})) \lor (\neg\mathsf{P} \land (\neg\mathsf{Q} \lor \neg\mathsf{R}))) \checkmark$

(2) $\quad \neg(\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})) \checkmark$

(3) $\quad \neg(\neg\mathsf{P} \land (\neg\mathsf{Q} \lor \neg\mathsf{R}))$

(4) $\quad \neg\mathsf{P} \qquad\qquad \neg(\mathsf{Q} \lor \mathsf{R}) \checkmark$

(5) $\quad \mid \qquad\qquad \neg\mathsf{Q}$

(6) $\quad \mid \qquad\qquad \neg\mathsf{R}$

At this stage, the only complex wff to unpack is (3); so let's apply the rule $(\neg\land)$, adding the results to both open branches:

(7) $\quad \neg\neg\mathsf{P} \quad \neg(\neg\mathsf{Q} \lor \neg\mathsf{R}) \qquad \neg\neg\mathsf{P} \quad \neg(\neg\mathsf{Q} \lor \neg\mathsf{R})$

$\quad ✻$

Which leaves us with three branches which so far remain open. Tackling the middle one first (hardly difficult!) we can add

(8) $\quad \mathsf{P}$

Remembering that (3) and now '$\neg\neg\mathsf{P}$' at (7) have been checked off, there are no remaining complex wffs on *this* path waiting to be unpacked. Hence there can be no lurking contradictions on this path – even if we continue work elsewhere on the tree, *this* path must stay open. Consider, then, the valuation which makes its simple wffs $\mathsf{P}, \neg\mathsf{Q}$ and and $\neg\mathsf{R}$ all true; this will make the more complex wffs further up this path true, hence will make (1) – the negation of **F** – true. Hence our valuation makes **F** false. Therefore it is not a tautology.

(c) One last example in this chapter. Consider the string of symbols

G $(\neg(\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})) \lor \neg(\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})))$.

Is this string even a wff? Well, that can be confirmed by the trick of changing the shape of some matching pairs of brackets, thus:

$$[\neg\{\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})\} \lor \neg\{\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})\}].$$

Then we see this is a wff of the form $[\alpha \lor \beta]$ with α the wff '$\neg\{\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})\}$', and β the wff '$\neg\{\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})\}$'. So is it a tautology?

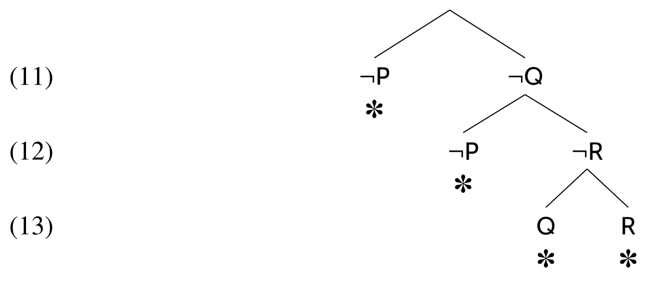
We start our tree test with the negation of **G**, which will therefore be of the form $\neg[\alpha \lor \beta]$ with the same α and β. So we can immediately apply the rule $(\neg\lor)$, to get

(1) $\neg[\neg\{\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})\} \lor \neg\{\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})\}]$ ✓
(2) $\neg\neg\{\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})\}$
(3) $\neg\neg\{\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})\}$

Continue in the obvious way, by applying the rule $(\neg\neg)$ to both (2) and (3), and check them off. Then apply the rule $(\land)$ twice to get

(1) $\neg[\neg\{\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})\} \lor \neg\{\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})\}]$ ✓
(2) $\neg\neg\{\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})\}$ ✓
(3) $\neg\neg\{\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})\}$ ✓
(4) $\{\neg(\mathsf{P} \land \mathsf{Q}) \land \neg(\mathsf{P} \land \mathsf{R})\}$ ✓
(5) $\{\mathsf{P} \land (\mathsf{Q} \lor \mathsf{R})\}$ ✓
(6) $\neg(\mathsf{P} \land \mathsf{Q})$
(7) $\neg(\mathsf{P} \land \mathsf{R})$
(8) P
(9) $(\mathsf{Q} \lor \mathsf{R})$

As this reminds us again, one lovely feature of working with trees is that, as we apply the unpacking rules, the wffs that we add to the tree get simpler and simpler. We are now just left with the last four very much more tractable wffs to deal with (as everything earlier has been checked off). It is a very simple matter to confirm that the tree now quickly closes. Unpacking (6), (7) and (9) in turn gives us:



Hence the assumption that we can make the negation of **G** true leads to contradiction; **G** is indeed a tautology.

2.6 Two sorts of trees: a reality check

Let's finish this chapter by very briefly emphasizing a key difference between syntactic parse trees (as in *IFL2* §9.2) and our semantic truth trees.

A parse tree disassembles a wff into its component subformulas as we go down the tree. So an expression above a branch point is a wff if and only if *both* expressions below the branch point are wffs. By contrast, on a truth tree, a wff above a branch point is true if and only if *at least one* of the wffs below the branch point is true.

So we can put it this way: there is a good sense in which branchings on parse trees are *conjunctive*, while branchings on truth trees are *disjunctive* and are exploring alternatives. You need to be alert to this important distinction whenever you encounter trees in logic.

2.7 Summary

In the previous chapter, we saw how to test an argument for tautological validity by 'working backwards'. We assume we can make the premiss and negated conclusion of the argument all true, and try to determine what the assumed bad valuation would look like. In the general case, this will necessitate considering alternative scenarios, and the resulting exploration can naturally be laid out in the form of a downward-branching tree.

By very simple tricks, we ensured that every wff on such a tree was assigned the value T. But then, if every wff is assigned the same value, we need not bother to explicitly write down the value – resulting in this chapter in our preferred *unsigned* trees.

Still somewhat informally, we have stated and explored the rules for constructing an unsigned tree using 'unpacking' rules taking us from more complex wffs on the tree to simpler wffs, and 'closing off' paths through the tree if they contain a contradictory pair of offs. (We give an official account in the next chapter.)

An argument is tautologically valid if every path through a properly constructed tree starting with the premisses and negated conclusion ends in contradiction and is closed. The argument is not valid if a corresponding tree has a branch that stays open even when every complex wff on it is unpacked.

We also noted that we can similarly use a tree to test a wff to determine whether it is a tautology. A wff is a tautology if and only if every path through a properly constructed tree starting with its negation ends in contradiction.

3 Truth trees further explored

The last two chapters introduced truth trees in a relatively relaxed way. This chapter starts by telling the same story again more crisply, in order to fix ideas and to provide a resource for future revision. We then say just a little more about the practice of using truth trees to test arguments for tautological validity. (We will not keep repeating the point made in §2.5, that trees can also be used to test wffs for being tautologies.)

3.1 Reminders about trees and our terminology

As is now familiar, truth trees will be downward-branching *binary* trees, meaning that a branch never forks more than two ways. We can take the general idea of a downward-branching binary tree to be intuitively clear. However, we should perhaps gather together some terminology, old and new. First, for binary trees in general:

(1) Positions on a tree are conventionally called *nodes*.

(2) Going down a binary tree, each node is followed by zero, one, or two *successor nodes*. A node is conventionally thought of as being joined to its successors by *edges*; but, for visual economy, in the case of truth trees, we have only drawn edges from a node to its successors when there is more than one successor – in other words, when the tree branches.

(3) Starting from the top node of the tree, the nodes and edges going downwards until the first branch point, if there is one, form the *trunk* of the tree.

(4) Start again from the top node and track down some route along edges through the tree to an *end-node* without successors, i.e. to the bottom tip of some branch. The nodes and edges form a (full) *path* through the tree.

(5) On a tree (as opposed to other kinds of 'graphs' made of nodes and edges) the paths downwards from the top node fan out without ever rejoining at a lower level. In an obvious sense, then, a tree is *loop-free*; this entails that there is only one path from the top node down to a given end-node.

Now for some initial terminology specific to our truth trees:

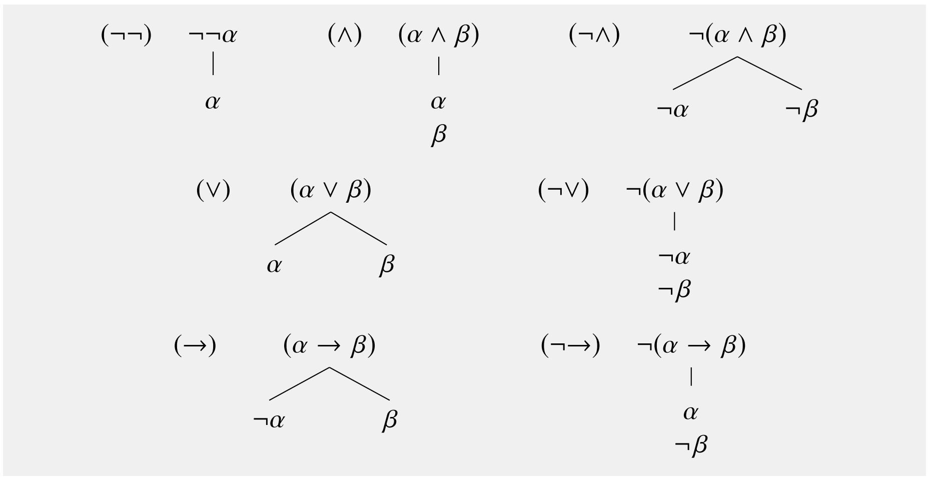
(6) On a truth tree, the nodes are occupied, labelled, or decorated by wffs from a PL language. If a wff α occupies a node on a path, we will say more briefly that the path *contains* α, or that α is *on* the path.

(7) If a path contains, for some wff α, both that wff and its negation $\neg\alpha$, then the path is *closed*. Otherwise it is *open*.

3.2 Testing arguments using truth trees, in summary

(a) Suppose, then, that we want to determine whether a given PL inference is tautologically valid. We are going to implement the idea of *working backwards* from the assumption the inference is invalid (i.e. the assumption that there's a valuation which makes the premisses and negated conclusion all true) by constructing a truth tree top downwards, with the premisses and negated conclusion of the inference at nodes forming the initial trunk of a tree (we can imagine each of them accompanied by an invisible ':= T'). We then aim to systematically 'unpack' the implications of those initial assignments – and then the implications of these implications, etc. – exploring alternatives when necessary.

As before, we call the wff that we are unpacking at any stage the *parent* wff, and call the new wffs we add when we unpack its implications its *children*. And we do the needed 'unpacking' of complex wffs using the following now-familiar rules:



How do we interpret these diagrammatically presented rules? To repeat, the non-branching rule (¬¬) tells us that, given a parent wff of the form $\neg\neg\alpha$ (invisibly assigned T) on a tree, we unpack its implications by adding α (invisibly assigned T) as as a child to new nodes at the foot of every open path containing that particular parent wff.

The non-branching rule (∧) tells us that, given a parent wff of the form $(\alpha \land \beta)$, we unpack its implications by adding both α and β to the foot of every open path containing that parent wff. Similarly for the other non-branching rules.

The branching rule (∨) tells us that, given a parent wff of the form $(\alpha \lor \beta)$, we unpack the alternatives by adding a binary fork with a branch to α and a branch to β to the foot of every open path containing that parent wff. Similarly for the other branching rules.

(b) And how do we deploy these unpacking rules in testing an inference for tautological validity? Here, in summary form, is the top-downwards procedure we have being using all along (adopting the policy of definitely stopping work on a tree if and when we find an open path on which we have unpacked every complex wff).

To construct a tree for testing whether the inference $\alpha_1, \alpha_2, \ldots, \alpha_n, \therefore \gamma$ is tautologically valid, proceed as follows:

(P1) Start the trunk of the tree by placing $\alpha_1, \alpha_2, \ldots, \alpha_n, \neg\gamma$ down the top nodes.

(P2) Inspect every path through the tree which doesn't yet finish with an absurdity sign to see whether it involves a contradiction, i.e. to see whether it contains both β and also the corresponding formula $\neg\beta$ (for some wff β). If it does, then we mark that the path is closed by adding the absurdity sign '✻'.

(P3) If every path through the tree is closed with an absurdity sign, then stop! Or if there is an open path on which every complex wff has been checked off, then also stop!

(P4) Otherwise, we choose some unchecked complex formula β that occurs on an open path. We then apply the appropriate unpacking rule and add the results to new nodes at the foot of every open path containing that occurrence of the wff β. We then check off β (using a checkmark '$\checkmark$').

(P5) Loop back to step (P2).

The 'checking off' rule ensures that the appropriate unpacking rule can only be applied once to any complex wff on the tree. Further, the new wffs added when we apply a rule are always shorter than the unpacked wff they result from. It follows that our unpacking-and-tree-extending procedure must eventually terminate as we unpack and check off shorter and shorter wffs. (We are assuming, as always, that an inference has only a finite number of premisses to start with!) Call any tree which results at the end of this process a *test tree* for the given inference under test.

(c) Two more bits of terminology applying to truth trees:

(1) A tree is *closed* if and only if every path through the tree is closed. Otherwise it stays *open*.

(2) A path through a tree is *completed open* if and only it is open and every complex wff on that path has been duly unpacked and checked off.

It is evident that, on any particular run of our tree building procedure, the test tree we end up with must either be closed, or else have at least one completed open path.

What do we learn from these alternative upshots?

A test tree for the inference $\alpha_1, \alpha_2, \ldots, \alpha_n, \therefore \gamma$ delivers verdicts as follows:

(VP) If the test tree is closed, the tested inference is tautologically valid.

(VN) If the test tree has a completed open path, the inference is tautologically invalid.

Why so? If the supposition that the αs plus $\neg\gamma$ are all true leads to contradiction whichever way we turn, then the supposition has to be false – we can't have the αs true and conclusion γ false. Or put it this way. Suppose everything on the initial trunk of our tree is true. Then our tree-building rules ensure that, as we extend the tree, we can choose a path where the wffs continue to be all true (see the 'Parents-to-children' principle in §2.1). And a path will all-true wffs cannot contain contradictory wffs, so will stay open. Hence, if there *is* a valuation which makes αs plus $\neg\gamma$ all true, the tree cannot close. Contraposing, we get (VP): if the tree does close, then there is no valuation making the αs true and γ false.

Suppose now that, starting with the αs plus $\neg\gamma$, we find a completed open path on the tree. Pick a valuation which makes the simple wffs on the path true – we can do this consistently as these simple wffs must be consistent with each other, or the path would be closed after all! Our tree-building rules ensure that, as we go up the path, this valuation makes all the more complex wffs we encounter true too, including the ones we started off with (see the 'Children-to-parents' principle in §2.1). So yes, there is a way of making the αs true and γ false, and the tested inference is indeed invalid. As (VN) says.

Hence our tree method does indeed work as advertised.

(d) Note that, as you would expect, our procedural rules allow us a free choice of which unchecked complex wff to unpack next each time we circle round to step (P4) – at least until we get to deal with the last unchecked complex wff. So there is usually no unique test tree. However, we do have the following key result:

Starting from the same initial wffs $\alpha_1, \alpha_2, \ldots, \alpha_n, \neg\gamma$ on the trunk of the tree, either any resulting test tree is closed, or alternatively any test tree has a completed open path. We can not get some closed trees and some with completed open paths.

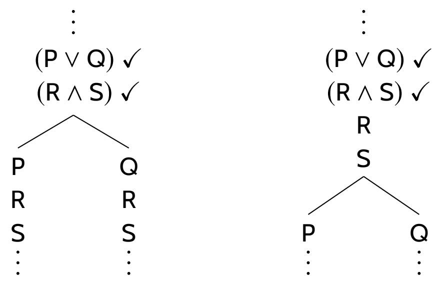
Why so? Because, given claims (VP) and (VN), if we could get both a closed tree and a (different!) tree with a completed open branch when testing a given argument, the argument would be both valid and invalid, which is impossible.

This confirms that it can't matter – at least as far as the ultimate verdict on validity is concerned – in which order we apply our tree-building rules.

3.3 Trees in practice

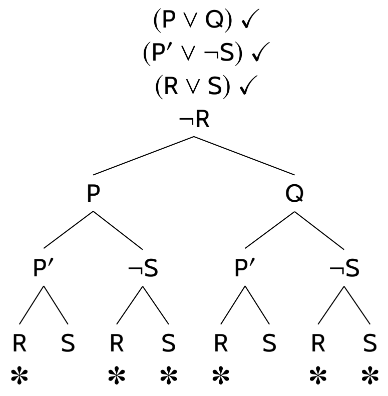
So much for a summary review of the theory. Now for three quick comments on using trees in practice. Again there is nothing new here; again we are just gathering together points we touched on in passing in the last two chapters.

(a) Compare first the following two fragments of truth trees:



The only difference between these two (parts of) trees is the order in which we have unpacked the pair of complex wffs: we end up, either way, with two paths with just the same wffs on them. But the second tree where we apply the non-branching rule first obviously avoids some duplication across branches and so needs a bit less time and ink to write down. That's a miniature illustration of our first maxim for getting neater trees: *in general, apply non-branching unpacking rules before branching rules, when we have the choice*.

(b) Compare next the following two completed truth trees. This time, the three complex wffs at the top of the tree all need to be unpacked using branching rules. But it still makes a difference in which order we do the unpacking:



On the left, we unpack the initial wffs in the order they appear on the trunk of the tree; and on the right, we unpack them in the reverse order. We end up in each case with two open paths, with the same simple wffs on them. But obviously the right-hand tree is a lot neater, because we can close off branches sooner. For instance, by choosing to unpack $(\mathsf{R} \lor \mathsf{S})$ first, we can immediately close off one alternative at the first fork, and don't have to explore it any further.

So that illustrates a second practical maxim: *wherever possible, choose to apply rules in an order which closes off branches as early as possible*.

(c) As we noted in §2.2(a), once we have found *one* completed open path on a tree for an argument, that is already enough to show that the argument in question is invalid. We have built this fact into our described official procedure for tree-building, but let's highlight it again: *assuming that all we care about is a yes/no verdict on validity, stop work on a tree once we have found one completed open path*.

(d) Let's work through a couple more annotated examples, keeping our practical maxims in mind. First, we will use a tree to test the following inference for validity:

A $(\mathsf{P} \land (\mathsf{R} \lor \mathsf{Q})),\ ((\mathsf{P} \land \neg\mathsf{Q}) \lor (\neg(\mathsf{P}' \lor \mathsf{R}) \lor \mathsf{S})),\ ((\mathsf{P}' \land \mathsf{S}) \lor \neg\mathsf{Q})$
$\therefore\ ((\mathsf{R} \lor \mathsf{Q}) \land \mathsf{S})$.

So we start in the familiar way with the premisses and negated conclusion at the top of the tree. We note that only one of these complex wffs calls for a non-branching rule to be applied, so – following our first maxim – let's start by unpacking that one:

(1)	$(P \land (R \lor Q))$ ✓	(premiss)
(2)	$((P \land \neg Q) \lor (\neg(P' \lor R) \lor S))$	(premiss)
(3)	$((P' \land S) \lor \neg Q)$	(premiss)
(4)	$\neg((R \lor Q) \land S)$	(negated conclusion)
(5)	P	(from 1 by $\land$)
(6)	$(R \lor Q)$	(from 1 by $\land$)

Now note that (6) and (4) each contains an occurrence of '$(R \lor Q)$', unnegated in the case of (6) and to-be-negated-on-one-fork when we unpack (4). So – following our second practical maxim – let's continue the tree

(7)	$\neg(R \lor Q)$	$\neg S$	(alternatives from 4 by $\lor$)
	✻		

With (4) now checked off, we have a choice of three complex wffs to unpack next, those at lines (2), (3) and (6). And although the last is the simplest, our second practical maxim suggests dealing with (3) first, because it quickly leads to another closed branch:

(8)	$(P' \land S)$ ✓	$\neg Q$	(alternatives from 3 by $\lor$)
(9)	P'	|	(from 8 by $\land$)
(10)	S	|	(from 8 by $\land$)
	✻		

The sensible thing to do next, obviously enough, is *now* to unpack (6). For this results in a branch which we can immediately close off:

(11)	R	Q	(alternatives from 6 by $\lor$)
		✻	

If you are keeping track, then you will see that on our open path every complex wff has now been checked off except (2), so let's now unpack that at last:

(12)	$(P \land \neg Q)$	$(\neg(P' \lor R) \lor S))$	(alternatives from 2 by $\lor$)

And now the end is in sight! Let's work on the left-hand open path first. Every complex wff on that path has now been unpacked except the last one, '$(P \land \neg Q)$'. So let's deal with that. Check off (12) and add:

(13)	P	(alternatives from 12 by $\land$)
(14)	$\neg Q$	(alternatives from 12 by $\land$)

There are no more complex wffs on *this* path to be unpacked; and there is no contradictory pair of wffs to be found along it either. It is a completed open path.

So no more work is required: we can read off from the simple wffs on that open path the valuation $P := T$, $Q := F$, $R := T$, $S := F$ which makes everything on that path true and hence shows that **A** is indeed invalid. We are done.

(e) For a final example, consider the following inference:

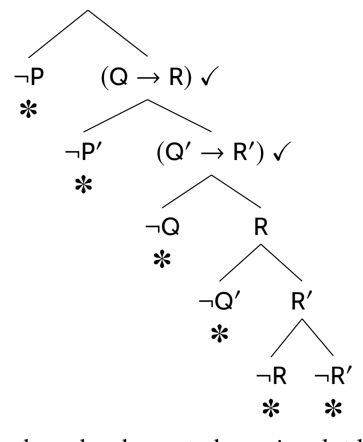
B $(P \to (Q \to R)),\ (P' \to (Q' \to R'))$
$\therefore\ ((P \wedge P') \to ((Q \wedge Q') \to (R \wedge R')).$

There are six atoms in play, and you are very welcome to write down a sixty-four line truth-table to confirm that this inference is indeed valid. But a truth tree is a *lot* quicker if you remember our practical maxims (especially if you omit the optional commentary).

Begin the tree with the premisses and negated conclusion and, in accord with the first maxim, keep applying any possible non-branching rule:

(1)	$(P \to (Q \to R))$	(premiss)
(2)	$(P' \to (Q' \to R'))$	(premiss)
(3)	$\neg((P \wedge P') \to ((Q \wedge Q') \to (R \wedge R'))\ \checkmark$	(negated conclusion)
(4)	$(P \wedge P')\ \checkmark$	(from 3 by $\neg\!\to$)
(5)	$\neg((Q \wedge Q') \to (R \wedge R'))\ \checkmark$	(from 3 by $\neg\!\to$)
(6)	P	(from 4 by $\wedge$)
(7)	P'	(from 4 by $\wedge$)
(8)	$(Q \wedge Q')\ \checkmark$	(from 5 by $\neg\!\to$)
(9)	$\neg(R \wedge R')$	(from 5 by $\neg\!\to$)
(10)	Q	(from 8 by $\wedge$)
(11)	Q'	(from 8 by $\wedge$)

That leaves (1), (2) and (9) as complex wffs still to check off. If we unpack (1) or (2) the tree forks with one branch immediately closing; not so if we unpack (9). So let's sensibly follow our second maxim and unpack (1) and (2) first. Then our tree neatly concludes:



(12)	(alternatives from 1 by $\to$)
(13)	(alternatives from 2 by $\to$)
(14)	(alternatives from 12 by $\to$)
(15)	(alternatives from 13 by $\to$)
(16)	(alternatives from 9 by $\neg\wedge$)

So yes, **B** is valid, and can be shown to be so in a lot less than 64 lines of working.

It is worth pausing to think, though, what would have happened if we had ignored our maxims and started by fully unpacking (1) and then (2) using our branching rules first. Fully unpacking (1) splits the tree, and then splits it again. So we get three open paths. Then fully unpacking (2) adds another three-way split at the foot of those paths, so we get nine open paths, before we go on to complete the tree. Which just reinforces the point that, if we are not careful, a tree for a given argument *can* sprawl horribly, even if there's another very tidy tree which gives the same result.

3.4 Comparative efficiency?

We have repeatedly seen that using a tree test on an argument can be significantly faster than plodding through the corresponding truth table test. Sometimes the gain in speed is extraordinary: remember, for example, Example **A** in §1.2.

Yet, as just remarked, things *can* become pretty messy too. As we saw in earlier chapters, the work needed on a truth table explodes exponentially with the number of atoms in play in the argument. But the work needed on a truth tree can *also* grow exponentially as the wffs in an argument get more numerous. Imagine dealing with a lot of unnegated disjunctions using the unpacking rule (∨). The tree will keep forking. If things go really badly, we get two branches, then four, then eight, then sixteen, then . . . . So the work needed to complete the tree can also explode. (To repeat, as we construct a tree, the added wffs get shorter and shorter; hence, even if things go really badly for us, the construction process must still eventually terminate. What we are worrying about now is *speed* of termination.)

We have seen that by following our practical maxims (stated again in §3.3), we can in good cases cut the work right down. But can we *always* tame trees and keep the needed amount of work under control? Indeed, is there *any* test for tautological validity which doesn't – in the worst cases – lead to exponential explosion as the arguments being tested get more involved?

No. Or so it is widely believed. But nobody really knows. The issue relates to a still unsolved deep problem in the foundations of the theory of computation, the so-called P vs NP problem. In fact, there is a million dollar prize on offer for the answer to *that*!

3.5 An afterword: '✻' vs '⊥'

In discussing truth-trees, we have been considering PL languages which *lack* an absurdity *constant* like '⊥' (i.e. lack an expression which can appear both as a stand-alone atomic PL wff, and as a component of complex wffs, and which always evaluates as false).

However our trees do involve an absurdity *sign* '✻' which serves as a glorified punctuation mark. It isn't a PL wff and can't appear as a constituent of one. If we think of our unsigned truth-trees as shorthand for signed trees, and think of our signed trees as encapsulating metalinguistic reasoning *about* a PL inference, then we can regard '✻' as belonging to our metalanguage.

This distinction between '⊥' as it appears e.g. in the main text of *IFL2* and '✻' as it appears in these chapters should be clear enough. But even so, it has probably been a good idea to follow convention and keep them apart by not having '⊥' in our object language.

Still, we *could* of course use truth-trees in the same style to deal with inferences in PL languages which *do* contain an absurdity constant. How? We would just need to add to our existing rules the obvious rule that any path of a tree containing '⊥' contains an absurdity and hence can be closed off with '✻'!

3.6 Summary

We have reviewed the procedures for constructing a truth tree for a PL inference, and the principles for extracting a verdict on whether the inference is tautologically valid – if the tree closes, the argument is valid; if the tree stays open, the argument is invalid.

We outlined arguments for those principles for extracting positive and negative verdicts from trees.

Test trees are not unique, but always give the same verdict.

Some practical maxims for working with trees: apply non-branching rules first; look for branchings that close quickly; stop once one completed open path is found.

Trees are typically faster than truth tables. But as we deal with trees for more and more complex inferences, in the worst cases the amount of work needed on a tree can exponentially explode.