Metatheory

Tim Button
University of Cambridge

In 2005, P.D. Magnus released the first version of his open-source logic textbook, forall***x***. In 2012, Tim Button adapted this for a first-year course at the University of Cambridge. Metatheory takes up exactly where the Cambridge implementation of forall***x*** leaves off.

Once again, Tim would like to thank Magnus for making forall***x*** freely available. Tim would also like to thank Owen Griffiths, Nicholas Heitler, Brian King, Stella Rhode, and Mat Simpson for comments and corrections.

This copy of Metatheory was released on Saturday, 22 September 2018. The most recent version is available at nottub.com.

Typeset in XeLaTeX, using Libertinus Serif and Libertinus Sans. The style for typesetting proofs employs forallx.sty by P.D. Magnus; Magnus's file is based upn fitch.sty (v0.4) by Peter Selinger.

Contents

Introduction

This book is an introduction to the metatheory of truth-functional logic, also known as the propositional calculus.

This book does not itself contain an account of truth-functional logic, but instead assumes a prior understanding. More specifically, this book assumes a thorough understanding of the syntax, semantics and proof-system for TFL (truth-functional logic) that is presented in forall***x***:Cambridge. forall***x***:Cambridge can be obtained, free of charge, at www.nottub.com/forallx.shtml. There is nothing very surprising about the syntax or semantics of TFL, and the proof-system is a fairly standard Fitch-style system.

This book does not, though, presuppose any prior understanding of (meta)mathematics. Chapter 1 therefore begins by explaining induction from scratch, first with mathematical examples, and then with metatheoretical examples. Each of the remaining chapters presupposes an understanding of the material in Chapter 1.

One idiosyncrasy of this book is worth mentioning: it uses no set-theoretic notation. Where I need to talk *collectively*, I do just that, using plural-locutions. Although this is unusual, I think Metatheory benefits from being free of unnecessary epsilons and curly brackets. So, where some books say:

> $\{\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_n\}$ is inconsistent

to indicate that no valuation makes all of $\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_n$ true, I say:

> $\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_n$ are jointly inconsistent

I use upper-case Greek letters for plural terms, so I might equally say:

> Γ are jointly inconsistent

Inconsistency is therefore a (collective) property of sentences, rather than a property of sets (of sentences). Semantic entailment also comes out as a (collective) property of sentences. To indicate that every valuation that makes all of Γ true also makes $\boldsymbol{C}$ true, we write:

$$\Gamma \vDash \boldsymbol{C}$$

where, of course, Γ *are* some sentences. If there is a valuation that makes all of Γ true but $\boldsymbol{C}$ false, we write:

$$\Gamma \nvDash \boldsymbol{C}$$

The corresponding notions from proof-theory are also properties of sentences. Thus we write:

$$\Gamma \vdash \boldsymbol{C}$$

to indicate that there is a TFL-proof which starts with assumptions among Γ and ends with $\boldsymbol{C}$ on no further assumptions. If there is no such proof, we write:

$$\Gamma \nvdash \boldsymbol{C}$$

A special case of this notation is used when there is a TFL-proof which starts with assumptions among Γ and ends with '$\bot$' (i.e. with an absurdity). In this situation, we say that $\Gamma \vdash \bot$, or that Γ are *jointly contrary*. (I should note that '$\bot$' is not a primitive 0-place connective of TFL. Rather, '$\bot$' shows up only in TFL's proof-theory. For more details, see forall***x***:Cambridge§26.7)

Induction 1

This book proves a bunch of results *about* formal logical systems. Specifically, it proves results about truth-functional logic(s): that is, logics where all of the connectives are truth functional. More specifically still, it proves results about TFL, a particular truth-functional logic whose syntax, semantics and proof-system are detailed in forall*x*:Cambridge.

When we prove results *about* formal logical systems, we inevitably have to rely upon some background logical reasoning. This reasoning need not, itself, be completely formalised. Nonetheless, the logical principles that we shall invoke informally – like *modus ponens*, or *universal generalisation* – should all feel familiar from their formalised counterparts.

However, when proving results about formal logical systems, we also rely upon *mathematical* reasoning. And there is an important mathematical principle which we shall use frequently: *induction*.[1] Indeed, almost every result in this book will invoke induction, in some form or other. So, the task of this first chapter is to familiarise ourselves with induction.

1.1 Stating and justifying the Principle of Induction

I mentioned that induction is a mathematical principle. In its plain vanilla form, it governs the behaviour of the *natural numbers*. These are all the finite, whole, non-negative numbers: 0, 1, 2, 3, and so on. (In what follows, whenever I say 'number', I always mean 'natural number'.) Here is the principle:

Principle of Induction. Let φ be any property such that both:

- 0 has φ; and
- for every natural number n: if n has φ, then $n + 1$ has φ.

Then every natural number has φ.

Here is an intuitive way to think about the Principle of Induction. Imagine that we had infinitely many dominoes – one for every number – balanced on their ends. Whenever we prove 'n has φ', we get to knock down the corresponding domino. Our aim is to knock down all of them. But there are too many of them for us to hope

[1] NB: *mathematical* induction must be sharply distinguished from the kind of *empirical* induction that David Hume discussed.

to knock them down one at a time. Fortunately, dominoes have a nice property: *if* we can arrange them in a line , so that when a domino falls it knocks over the next in line, *then* we can knock all the dominoes down just by knocking the first one down. To knock down the first domino corresponds with proving that 0 has φ. And to arrange the dominoes in this way corresponds with proving the conditional: if n has φ then $n+1$ has φ.

Of course, this is just an analogy, and a spatial one at that. Why, then, should we *believe* the Principle of Induction? Here is an argument in its favour. Suppose you have proved both that 0 has φ and also that, for every n, if n has φ then $n+1$ has φ. Now, pick any number, m. Given what you have proved, you would be justified in writing down all of the following:

> 0 has φ
> if 0 has φ, then 1 has φ
> if 1 has φ, then 2 has φ
> ⋮
> if $m-1$ has φ, then m has φ

It would now follow, by m-many applications of *modus ponens*, that m has φ. And since m was arbitrary, this point generalises: every number has φ.

I doubt this line of argument will convince the determined sceptic.[2] And, in the end, the question 'Why believe in the Principle of Induction?' is a deep question in the philosophy of mathematics. That question is interesting, but it is a question for another day. For the purposes of this book, we shall take the Principle of Induction for granted.

1.2 Arithmetical proofs involving induction

To invoke induction in a proof, we must first identify some property, φ, that concerns us. We call this *the induction property*. We then do two things:

- Prove the *base case*: we need to show that 0 has φ. Often, this is a fairly straightforward task.
- Prove the *induction case*: we need to show that, for any n, if n has φ, so does $n+1$. This is normally the harder step. We typically proceed by 'choosing some arbitrary n', and assuming 'for induction' that n has φ. If, on just this basis, we can show that $n+1$ has φ, then we have proved the conditional: if n has φ, then $n+1$ has φ. And since n was arbitrary, this holds for *all* n.

To illustrate the strategy, I shall offer a few simple examples of the use of induction, in establishing some arithmetical results. (Just to be clear: I am not going to make any use of the arithmetical results *themselves*; I just want to show how induction is used in proofs.) So, for example, suppose we want to prove a claim that should be very obvious:

[2] It is worth spending some time asking yourself: Why won't this convince the sceptic?

Proposition 1.1. Every number is either even or odd. That is: for every number n, *either* there is a number k such that $n = 2k$ (i.e. n is even), *or* there is a number k such that $n = 2k + 1$ (i.e. n is odd).

I shall prove this Proposition by induction. The induction property will be *x is either odd or even.* It suffices to prove the base case and the induction case; then all we need to complete the proof is to appeal to induction. Here goes:

Proof. Evidently, $0 = 2 \times 0$. So 0 has the induction property.

Now pick an arbitrary number, n, and suppose (for induction) that n has the induction property. There are two cases to consider

- Suppose n is even, i.e. $n = 2k$, for some number k. Then $n + 1 = 2k + 1$, i.e. $n + 1$ is odd.
- Suppose n is odd, i.e. $n = 2k + 1$, for some number k. Then $n + 1 = 2k + 2 = 2(k + 1)$, i.e. $n + 1$ is even.

Either way, $n+1$ has the induction property. So if n has the induction property, then so does $n + 1$.

This completes the proof of Proposition 1.1 by induction. ■

In addition to the use of induction, a couple of things should strike you about this proof. In particular, you should note that it is not a *formal* proof in any particular deductive system. It is a bit of informal, *mathematical*, reasoning, written in mathematised English. That said, it is perfectly rigorous. (The symbol '■' at the end of the proof just indicates that the proof is complete; other texts use other symbols, e.g. 'QED' or '□'.)

To cement ideas, let me offer two more examples of proofs by induction:

Proposition 1.2. $2^0 + 2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$, for any number n.

Proof. Evidently $2^0 = 1 = 2^{0+1} - 1$. So 0 has the induction property.

Now, pick any arbitrary number, k, and suppose (for induction) that

$$2^0 + 2^1 + 2^2 + \ldots + 2^k = 2^{k+1} - 1$$

Then we have:

$$\begin{aligned} 2^0 + 2^1 + 2^2 + \ldots + 2^k + 2^{k+1} &= 2^{k+1} - 1 + 2^{k+1} \\ &= \left(2^{k+1} \times 2\right) - 1 \\ &= 2^{k+2} - 1 \end{aligned}$$

So the induction property holds for $k + 1$ as well. ■

Proposition 1.3. $0 + 1 + 2 + \ldots + n = \frac{n^2+n}{2}$, for any number n.

Proof. Evidently $0 = \frac{0^2+0}{2}$. So 0 has the induction property.

Now, pick any arbitrary number, k, and suppose (for induction) that

$$0 + 1 + 2 + \ldots + k = \frac{k^2 + k}{2}$$

Then we have:

$$\begin{aligned} 0 + 1 + 2 + \ldots + k + (k+1) &= \frac{k^2 + k}{2} + (k+1) \\ &= \frac{k^2 + k + 2(k+1)}{2} \\ &= \frac{(k^2 + 2k + 1) + (k+1)}{2} \\ &= \frac{(k+1)^2 + (k+1)}{2} \end{aligned}$$

So the induction property holds for $k+1$ as well. ■

We could keep going with examples from arithmetic, but these illustrate the use of induction, and give some sense of how powerful it is.

1.3 Strong Induction

Now let me introduce a second inductive principle:

Principle of Strong Induction. Let φ be any property. Suppose that, for every number n, if all the numbers less than n have φ, then n itself has φ. Then *every* number has φ.

Why should we believe *this*? In fact, it follows directly from the original Principle of Induction, as I shall now show:[3]

Proof. Suppose that, for every number n, if all the numbers less than n have φ, then n itself has φ. We aim to show that this entails that every number has φ (thereby vindicating the Principle of Strong Induction).

To do this, we shall define a new property, ψ, as follows:

> *every number less than x has φ*

We shall now prove by (ordinary) induction that every number has ψ.

> *Base case:* 0 has ψ, vacuously, since 0 is the smallest number.
> *Induction case:* Suppose, for (ordinary) induction, that n has ψ. That is to say that every number less than n has φ. So, by the supposition at the start of this proof, n has φ. It follows that every number less than $n+1$ has φ, i.e. that $n+1$ has ψ.

[3] And it entails the original Principle too, though I shall not show this.

It follows, by (ordinary) induction, that *every* number has ψ. And since every number has ψ, and for every number there is a larger number, every number has φ. Hence the use of Strong Induction is vindicated. ■

It is worth illustrating the power of the Principle of Strong Induction, with an arithmetical example:[4]

Proposition 1.4. Every number greater than 1 has a prime factorisation (i.e. it can be expressed as the product of one or more prime numbers).

Proof. Fix n, and suppose for (strong) induction that every number less than n has the induction property (i.e. the property that *if* $x > 1$ *then* x *has a prime factorisation*). There are now three (exhaustive) cases to consider:

> *Case 1:* $n \leq 1$. Then n obviously has the induction property.
> *Case 2:* n *is prime.* Then n obviously has the induction property.
> *Case 3:* $n > 1$ *and* n *is not prime.* Then $n = ab$ with both $n > a > 1$ and $n > b > 1$. By assumption, a and b have prime factorisations, i.e. $a = p_1 \times ... \times p_j$ and $b = q_1 \times ... \times q_k$, for some prime numbers $p_1, ..., p_j, q_1, ..., q_k$. So now $n = p_1 \times ... \times p_j \times q_1 \times ... \times q_k$, i.e. n has a prime factorisation.

So, in every case, n has the induction property on the assumption that every smaller number has the induction property. Hence, by strong induction, every number has the induction property. ■

Note that we needed to use *strong* induction here, because in the case where $n = ab$, we know nothing about a and b *except* that $n > a > 1$ and $n > b > 1$.

1.4 Induction on length

I now want to move on from arithmetical propositions to claims about TFL. By the end of this book, we will have proved some rather sophisticated results. But we need to start at the shallow end. In particular, we shall start by proving a syntactic result, which should already be familiar:

Proposition 1.5. In any TFL sentence, there are exactly as many instances of '(' as instances of ')'.

When I mention 'TFL sentences' here I have in mind the the objects that were defined recursively in forall*x*:Cambridge §6. In particular, then, I shall not allow us to rely upon the convenient bracketing conventions that we introduced later in forall*x*:Cambridge; so, we cannot, for example, allow for arbitrarily long conjunctions, and we cannot drop the outermost brackets of a sentence. To save space, in what follows I shall speak of 'sentences' rather than 'TFL sentences as given by the recursive definition of forall*x*:Cambridge §6'.

[4] This is one half of *the* Fundamental Theorem of Arithmetic; the other half is that prime factorisations are *unique*.

Now, we want to prove Proposition 1.5 by induction. But there is an immediate obstacle in our path. The property mentioned in Proposition 1.5 is *x has as many instances of '(' as of ')'*, and that is a property of *sentences*, rather than a property of *numbers*. So, if we are to prove this Proposition using induction, we need a way to associate sentences with numbers.

The easiest way to associate sentences with numbers is just by considering their length. More precisely, by the LENGTH of a sentence, we shall mean the number of instances truth-functional connectives (i.e. '$\neg$', '$\wedge$, '$\vee$', '$\rightarrow$' and '$\leftrightarrow$') that the sentence contains.[5] Thus, for example:

- 'B' has length 0
- 'A_{234}' has length 0
- '$\neg(B \vee C_4)$' has length 2
- '$\neg\neg B$' has length 2
- '$\neg\neg\neg B$' has length 3

Having defined the notion of the length of a sentence, we could now prove Proposition 1.5 by (strong) induction on numbers, considered as lengths of sentences. But things will go easier for us if we introduce *yet another* inductive principle that relates more directly to TFL:

Principle of Strong Induction on Length. Let φ be any property. Suppose that, for any sentence, $\boldsymbol{A}$, if every shorter sentence has φ, then $\boldsymbol{A}$ has φ. Then every sentence has φ.

This principle follows directly from (ordinary) strong induction:

Proof. Suppose that, for any sentence, $\boldsymbol{A}$, if every shorter sentence has φ, then $\boldsymbol{A}$ has φ. We aim to show that this entails that every sentence has φ (thereby vindicating the Principle of Strong Induction on Length).

To do this, we shall define a new property of the numbers, ψ, thus:

> *every sentence of length x has φ*

Pick any arbitrary number, n, and suppose for (strong) induction that every number less than n has ψ. That is, every sentence of length $< n$ has φ. So, by supposition, every sentence of length n has φ. So n has ψ. It follows, by strong induction, that every number has ψ.

Since every sentence has some number as its length, it now follows that every sentence has φ, as required. ∎

So, armed with the notion of the length of a sentence, and the Principle of Strong Induction on Length, we can now prove Proposition 1.5:

Proof of Proposition 1.5. Let $\boldsymbol{A}$ be any sentence, and suppose for (strong) induction (on length) that every shorter sentence has exactly as many instances of '(' as of ')'. We shall show that $\boldsymbol{A}$ has exactly as many instances of '(' as of ')', by considering

[5] Many books refer to this as the *complexity* of a sentence.

all the different forms that $\boldsymbol{A}$ might have. (These six cases are given to us by the *recursive definition* of the notion of a sentence; for a reminder of the definition, see forall$\boldsymbol{x}$:Cambridge §6.)

Case 1: $\boldsymbol{A}$ is atomic. Then $\boldsymbol{A}$ contains no brackets, so it has exactly as many instances of '(' as of ')'.

Case 2: $\boldsymbol{A}$ is $\neg\boldsymbol{B}$, for some sentence $\boldsymbol{B}$. Clearly, $\boldsymbol{B}$ is shorter than $\boldsymbol{A}$. By supposition, then, $\boldsymbol{B}$ contains exactly as many instances of '(' as of ')'. Hence $\boldsymbol{A}$ does too.

Case 3: $\boldsymbol{A}$ is $(\boldsymbol{B} \wedge \boldsymbol{C})$, for some sentences $\boldsymbol{B}$ and $\boldsymbol{C}$. Clearly, both $\boldsymbol{B}$ and $\boldsymbol{C}$ are shorter than $\boldsymbol{A}$. By supposition, then, $\boldsymbol{B}$ contains exactly as many instances of '(' as of ')', say i; and $\boldsymbol{C}$ contains exactly as many instances of '(' as of ')', say j. Hence $\boldsymbol{A}$ contains $i + j + 1$ instances of '(', and $i + j + 1$ instances of ')'.

Case 4: $\boldsymbol{A}$ is $(\boldsymbol{B} \vee \boldsymbol{C})$, for some sentences $\boldsymbol{B}$ and $\boldsymbol{C}$. Exactly similar to case 3.

Case 5: $\boldsymbol{A}$ is $(\boldsymbol{B} \rightarrow \boldsymbol{C})$, for some sentences $\boldsymbol{B}$ and $\boldsymbol{C}$. Exactly similar to case 3.

Case 6: $\boldsymbol{A}$ is $(\boldsymbol{B} \leftrightarrow \boldsymbol{C})$, for some sentences $\boldsymbol{B}$ and $\boldsymbol{C}$. Exactly similar to case 3.

There are no other forms that $\boldsymbol{A}$ could take. Hence, for any sentence, $\boldsymbol{A}$, if every shorter sentence has exactly as many instances of '(' as of ')', then so does $\boldsymbol{A}$. The result now follows by strong induction on length. ■

Now, I hope that it was obvious that the Proposition was true, before we set about proving it. The purpose of this proof was not, then, to *convince* you of its truth. Rather, I gave you this proof primarily in order to illustrate the Principle of Strong Induction on Length. Moreover, this proof exhibits three features that are worth mentioning explicitly.

First, even though the result was *obvious*, it took quite a lot of effort to prove it. This is fairly typical of metalogical results: showing them with sufficient rigour can be hard!

Second, we teamed an inductive principle with a recursive definition. This is fairly typical of metalogical results. Indeed, it is hardly surprising that induction and recursion should be best of friends. Recursive definitions tell you how to build objects up, stage-by-stage; proofs by (strong) induction involve showing that something holds at later stages if it has held at every stage so far.

Third, when we broke the proof down into cases, cases 3–6 were a bit repetitious. Indeed, all four of our two-place connectives have the same *syntactic* behaviour, and that is all we cared about during the course of the proof. So we could simply have rolled cases 3–6 together in something like the following:

Case 3: $\boldsymbol{A}$ is $(\boldsymbol{B} \triangleleft \boldsymbol{C})$, for some two-place connective $\triangleleft$ and sentences $\boldsymbol{B}$ and $\boldsymbol{C}$. Clearly, both $\boldsymbol{B}$ and $\boldsymbol{C}$ are shorter than $\boldsymbol{A}$. By supposition, then, $\boldsymbol{B}$ contains exactly as many instances of '(' as of ')', say i; and $\boldsymbol{C}$ contains exactly as many instances of '(' as of ')', say j. Hence $\boldsymbol{A}$ contains $i + j + 1$ instances of '(', and $i + j + 1$ instances of ')'.

saving ourselves some time and effort.[6] Generally, if we need to do a proof by cases – and we often shall – we should pause before ploughing in, to think about exactly what the relevant cases are.

1.5 Unique readability

I shall close this chapter by proving two more results about the syntax of TFL.

When $\boldsymbol{A}$ and $\boldsymbol{B}$ are expressions, say that $\boldsymbol{B}$ is an INITIAL SEGMENT of $\boldsymbol{A}$ *iff* $\boldsymbol{A}$ can be obtained by concatenating some (possibly no) symbols to the (right-hand) end of $\boldsymbol{B}$. For example:

- '$\neg$' is an initial segment of '$\neg\neg A$' and an initial segment of '$\neg(P \wedge Q)$', but not an initial segment of '$(\neg P \rightarrow Q)$'.
- '$(P \wedge \neg Q)$' is an initial segment of '$(P \wedge \neg Q)$', but not an initial segment of '$((P \wedge \neg Q) \vee R)$'

Here is a cute result about initial segments.

Lemma 1.6. For any sentence $\boldsymbol{A}$, the only sentence which is an initial segment of $\boldsymbol{A}$ is $\boldsymbol{A}$ itself.

Proof. Let $\boldsymbol{A}$ be any sentence, and suppose (for induction) that every shorter sentence, $\boldsymbol{B}$, has the property that the only sentence which is an initial segment of $\boldsymbol{B}$ is $\boldsymbol{B}$ itself (this is the induction property). There are three cases:

> *Case 1: $\boldsymbol{A}$ is atomic..* Then the only initial segment of $\boldsymbol{A}$ is $\boldsymbol{A}$ itself.
>
> *Case 2: $\boldsymbol{A}$ is $\neg\boldsymbol{B}$, for some sentence $\boldsymbol{B}$.* Let $\boldsymbol{A}^*$ be any sentence that is an initial segment of $\boldsymbol{A}$. So $\boldsymbol{A}^*$ is $\neg\boldsymbol{B}^*$, for some sentence $\boldsymbol{B}^*$. Evidently, $\boldsymbol{B}^*$ is an initial segment of $\boldsymbol{B}$. But $\boldsymbol{B}$ and $\boldsymbol{B}^*$ are both sentences that are shorter than $\boldsymbol{A}$; so by the induction hypothesis, $\boldsymbol{B}^*$ must just be $\boldsymbol{B}$ itself. Hence $\boldsymbol{A}^*$ is $\boldsymbol{A}$ itself, and so $\boldsymbol{A}$ has the induction property.
>
> *Case 3: $\boldsymbol{A}$ is $(\boldsymbol{B} \triangleleft \boldsymbol{C})$, for some sentences $\boldsymbol{B}$ and $\boldsymbol{C}$ and some two-place connective $\triangleleft$.* Let $\boldsymbol{A}*$ be any sentence that is an initial segment of $\boldsymbol{A}$. Evidently, $\boldsymbol{A}^*$ starts with a '('; so, by the recursion rules, $\boldsymbol{A}^*$ must be $(\boldsymbol{B}^* \blacktriangleleft \boldsymbol{C}^*)$, for some sentences $\boldsymbol{B}^*$ and $\boldsymbol{C}^*$ and some two-place connective $\blacktriangleleft$. Now, exactly as in Case 2, we can see that $\boldsymbol{B}^*$ must just be $\boldsymbol{B}^*$ itself (since $\boldsymbol{B}^*$ is an initial segment of $\boldsymbol{B}$). Consequently, $\triangleleft$ and $\blacktriangleleft$ must be the same connective. And so, by a similar argument, $\boldsymbol{C}^*$ must just be $\boldsymbol{C}$ itself (since $\boldsymbol{C}^*$ is an initial segment of $\boldsymbol{C}$). Hence $\boldsymbol{A}^*$ is $\boldsymbol{A}$ itself, and so $\boldsymbol{A}$ has the induction property.

This completes the strong induction on length. ■

Considered on its own, this result is not very thrilling. But I called it a 'lemma' because it will be used to prove a result that is sufficiently important to be called a 'theorem', and to have its own name.

[6] Note that '$\triangleleft$', like '$\boldsymbol{A}$', is a symbol of the *meta*language, not the *object* language. For a reminder of the distinction, forall*x*:Cambridge §7.

Before I state and prove the Theorem, I shall try to motivate it. When I introduced the idea of the MAIN LOGICAL OPERATOR of a sentence (in forall*x*:Cambridge §6), I described it as the symbol that 'was introduced *last*, when constructing the sentence' by the recursion rules. However, I gave you no guarantee that there was only *one* way to construct a sentence using those rules. Consequently, I gave you no guarantee that the main logical operator is *unique*. And, at that point in your lives, we were not in a position to guarantee this result. Now, though, we are; and here is that overdue guarantee.

Theorem 1.7. Unique Readability Theorem. For every sentence $\boldsymbol{A}$, exactly one of the following obtains:

1. $\boldsymbol{A}$ is an atomic sentence; or
2. there is a unique sentence, $\boldsymbol{B}$, such that $\boldsymbol{A}$ is $\neg\boldsymbol{B}$; or
3. there are unique sentences, $\boldsymbol{B}$ and $\boldsymbol{C}$, and a unique two-place connective, $\triangleleft$, such that $\boldsymbol{A}$ is $(\boldsymbol{B} \triangleleft \boldsymbol{C})$.

Proof of Theorem 1.7. Let $\boldsymbol{A}$ be any sentence. It is clear that *at most* one of (1), (2) or (3) obtains, since each requires that $\boldsymbol{A}$ start with a different symbol. It remains to show that *at least* one of them obtains. There are three cases to consider:

Case 1: $\boldsymbol{A}$ is atomic. i.e. case (1) obtains.

Case 2: $\boldsymbol{A}$ is $\neg\boldsymbol{B}$, for some sentence $\boldsymbol{B}$. Then it is clear that there is exactly *one* sentence that is formed by removing the negation, i.e. $\boldsymbol{B}$ is unique. So case (2) obtains.

Case 3: $\boldsymbol{A}$ is $(\boldsymbol{B} \triangleleft \boldsymbol{C})$, for some sentences $\boldsymbol{B}$ and $\boldsymbol{C}$ and some two-place connective $\triangleleft$. To establish that case (3) obtains, we need to show that this decomposition is *unique*. So suppose that there are also sentences, $\boldsymbol{B}^*$ and $\boldsymbol{C}^*$, and a two-place connective, $\blacktriangleleft$, such that $\boldsymbol{A}$ is $(\boldsymbol{B}^* \blacktriangleleft \boldsymbol{C}^*)$. We can use exactly the same technique as in the proof of Case 3 of Lemma 1.6 to show that $\boldsymbol{B}$ is $\boldsymbol{B}^*$, that $\triangleleft$ is $\blacktriangleleft$ and that $\boldsymbol{C}$ is $\boldsymbol{C}^*$; the only difference is that we now invoke Lemma 1.6 itself (rather than an induction hypothesis). Hence, case (3) obtains. ■

This uniqueness really matters. It shows that there can be no syntactic ambiguity in TFL, so that each sentence admits of one (and only one) reading. Contrast this with the famous newspaper headline: '8^{th} Army push bottles up German rear'.

Practice exercises

A. Fill in the details in Case 3 of the proof of Lemma 1.6, and in Case 3 of the proof of Theorem 1.7.

B. Prove the following, by induction on length: for any sentence $\boldsymbol{A}$, every valuation which assigns truth values to all of the atomic sentences in $\boldsymbol{A}$ assigns a *unique* truth value to $\boldsymbol{A}$.

C. For any sentence $\boldsymbol{A}$, let:

- $a_{\boldsymbol{A}}$ be the number of instances of atomic sentences in $\boldsymbol{A}$;
- $b_{\boldsymbol{A}}$ be the number of instances of '(' in $\boldsymbol{A}$;
- $c_{\boldsymbol{A}}$ be the number of instances of two-place connectives in $\boldsymbol{A}$.

As an example: if $\boldsymbol{A}$ is '$(\neg B \to (B \wedge C))$', then $a_{\boldsymbol{A}} = 3$, $b_{\boldsymbol{A}} = 2$ and $c_{\boldsymbol{A}} = 2$. Prove the following, by induction on length: for any sentence $\boldsymbol{A}$: $a_{\boldsymbol{A}} - 1 = b_{\boldsymbol{A}} = c_{\boldsymbol{A}}$.

D. Suppose a sentence is built up by conjoining together all of the sentences $\boldsymbol{A}_1, ..., \boldsymbol{A}_n$, in any order. So the sentence might be, e.g.:

$$(\boldsymbol{A}_5 \wedge ((\boldsymbol{A}_3 \wedge \boldsymbol{A}_2) \wedge (\boldsymbol{A}_4 \wedge \boldsymbol{A}_1)))$$

Prove, by induction on the number of conjuncts, that the truth value of the resulting sentence (on any valuation which assigns a truth value to all of $\boldsymbol{A}_1, ..., \boldsymbol{A}_n$) will be the same, no matter how, exactly, they are conjoined. Then prove that the same is true if we consider disjunction rather than conjunction. Which notational conventions that were introduced in forall$\boldsymbol{x}$:Cambridge are vindicated by these proofs?

Substitution 2

In this chapter, we shall put our new familiarity with induction to good work, and prove a cluster of results that connect with the idea of substituting one sentence in for another.

Before going further, we should state the idea of 'substitution' a bit more precisely. Where $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{S}$ are sentences, $\boldsymbol{A}[\boldsymbol{S}/\boldsymbol{C}]$ is the result of searching through $\boldsymbol{A}$ and finding and replacing every occurrence of $\boldsymbol{C}$ with $\boldsymbol{S}$. As an example, when $\boldsymbol{A}$ is '$(Q \wedge (\neg P \vee (Q \leftrightarrow R)))$', when $\boldsymbol{C}$ is 'Q' and when $\boldsymbol{S}$ is '$\neg(D \rightarrow B)$', then $\boldsymbol{A}[\boldsymbol{S}/\boldsymbol{C}]$ is '$(\neg(D \rightarrow B) \wedge (\neg P \vee (\neg(D \rightarrow B) \leftrightarrow R)))$'.

All of the results in this chapter employ this surprisingly powerful idea.

2.1 Two crucial lemmas about substitution

Given your familiarity with TFL, here is a line of thought which should strike you as plausible. Suppose you have a valid argument in TFL. Now, run through that argument, uniformly substituting every instance of some atomic sentence that occurs within that argument, with some *other* sentence. The resulting argument is still valid.

It is worth realising that this line of thought is *not* generally plausible in natural languages. For example, here is a valid argument in English:

- Isaac is an eft; so Isaac is a newt.

Its validity, plausibly, connects with the *meanings* of its component words. But recall that the semantics of TFL is purely extensional, and knows nothing about meanings. Its only dimension of freedom is over how we assign truth and falsity to components of sentences. And uniform substitution – of the sort mentioned in the previous paragraph – cannot, in a relevant sense, *increase* our options for assigning truth values to such components.

But to convince ourselves that this line of thought is correct, we of course require a rigorous proof. Here it is.

Lemma 2.1. For any atomic sentence $\boldsymbol{C}$ and any sentences $\boldsymbol{A}_1, ..., \boldsymbol{A}_{n+1}, \boldsymbol{S}$:
if $\boldsymbol{A}_1, ..., \boldsymbol{A}_n \vDash \boldsymbol{A}_{n+1}$, then $\boldsymbol{A}_1[\boldsymbol{S}/\boldsymbol{C}], ..., \boldsymbol{A}_n[\boldsymbol{S}/\boldsymbol{C}] \vDash \boldsymbol{A}_{n+1}[\boldsymbol{S}/\boldsymbol{C}]$.

Proof. Suppose $\boldsymbol{A}_1, ..., \boldsymbol{A}_n \vDash \boldsymbol{A}_{n+1}$. Suppose also that v is a valuation which makes all of $\boldsymbol{A}_1[\boldsymbol{S}/\boldsymbol{C}], ..., \boldsymbol{A}_n[\boldsymbol{S}/\boldsymbol{C}]$ true. Let w be a valuation which differs from v (if at all) only

by assigning a value to $\boldsymbol{C}$ as follows: w makes $\boldsymbol{C}$ true iff v makes $\boldsymbol{S}$ true. I now make a claim:

> **Claim**: for any sentence $\boldsymbol{A}$: w makes $\boldsymbol{A}$ true iff v makes $\boldsymbol{A}[\boldsymbol{S}/\boldsymbol{C}]$ true.

I shall prove the Claim by induction on the length of $\boldsymbol{A}$. Suppose, for induction, that the Claim is true of every shorter sentence. Now there are three cases:

> *Case 1: $\boldsymbol{A}$ is atomic.* Then either $\boldsymbol{A}$ is $\boldsymbol{C}$, in which case $\boldsymbol{A}[\boldsymbol{S}/\boldsymbol{C}]$ is just $\boldsymbol{S}$; or $\boldsymbol{A}[\boldsymbol{S}/\boldsymbol{C}]$ is just $\boldsymbol{A}$. Either way, the Claim holds of $\boldsymbol{A}$.
> *Case 2: $\boldsymbol{A}$ is $\neg\boldsymbol{B}$, for some sentence $\boldsymbol{B}$.* Then, by assumption, w makes $\boldsymbol{B}$ true iff v makes $\boldsymbol{B}[\boldsymbol{S}/\boldsymbol{C}]$ true. By considering the truth-table for '$\neg$', it follows that w makes $\neg\boldsymbol{B}$ true iff v makes $\neg\boldsymbol{B}[\boldsymbol{S}/\boldsymbol{C}]$ true.
> *Case 3: $\boldsymbol{A}$ is $(\boldsymbol{B} \vartriangleleft \boldsymbol{C})$, for some sentences $\boldsymbol{B}$ and $\boldsymbol{C}$ and some two-place connective $\vartriangleleft$.* This is much like Case 3; I leave the details as an exercise.

This proves the Claim, by induction. It follows from the claim that w makes all of $\boldsymbol{A}_1, ..., \boldsymbol{A}_n$ true. And, since $\boldsymbol{A}_1, ..., \boldsymbol{A}_n \vDash \boldsymbol{A}_{n+1}$, we see that w makes $\boldsymbol{A}_{n+1}$ true. So, appealing to the Claim again, v makes $\boldsymbol{A}_{n+1}[\boldsymbol{S}/\boldsymbol{C}]$ true. ■

Let's see this Lemma in action. Consider these three claims:

$$(\neg A \rightarrow (A \vee B)) \vDash (A \vee B)$$
$$(\neg\neg A \rightarrow (\neg A \vee \neg B)) \vDash (\neg A \vee \neg B)$$
$$(\neg(P \leftrightarrow \neg Q) \rightarrow ((P \leftrightarrow \neg Q) \vee \neg(B \wedge (C \leftrightarrow D)))) \vDash ((P \leftrightarrow \neg Q) \vee \neg(B \wedge (C \leftrightarrow D)))$$

The first of these is easy to check using a truth table; the second is marginally harder; and the third is a pain! But if we recognise that the second and third are the result of uniform substitution into the first, then we can use Lemma 2.1 to save ourselves a lot of work.

There is a deeper point here. Lemma 2.1 vindicates the practice of describing an *entire argument structure* as valid. For example, we can say that the following is a valid *pattern* of entailment:

$$(\neg\boldsymbol{A} \rightarrow (\boldsymbol{A} \vee \boldsymbol{B})) \vDash (\boldsymbol{A} \vee \boldsymbol{B})$$

We know already that at least one instance of this pattern of entailment is valid. Lemma 2.1 then tells us that the entailment will hold, *regardless* of what sentences we use for $\boldsymbol{A}$ and $\boldsymbol{B}$. Lemma 2.1 therefore underscores the purely structural nature of entailments, in our semantics for TFL.

Lemma 2.1 concerns tautological *entailment*. We shall also want to have a result to hand which concerns tautological *equivalence*. It will help us if we introduce the symbol '$\dashv\vDash$' to symbolise tautological equivalence. So we write:

$$\boldsymbol{A} \dashv\vDash \boldsymbol{B}$$

to symbolise that $\boldsymbol{A}$ and $\boldsymbol{B}$ are tautologically equivalent; or, in other words, that $\boldsymbol{A} \vDash \boldsymbol{B}$ and $\boldsymbol{B} \vDash \boldsymbol{A}$. Using this notation, we can prove a nifty little result:

Lemma 2.2. For any sentences $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{S}$: if $\boldsymbol{C} \dashv\vDash \boldsymbol{S}$, then $\boldsymbol{A} \dashv\vDash \boldsymbol{A}[\boldsymbol{S}/\boldsymbol{C}]$.

Proof. By an induction on the length of $\boldsymbol{A}$; I leave this as an exercise. ■

In a slogan: substitution of tautological equivalents preserves tautological equivalence. We shall rely upon this frequently, in what follows.

2.2 Interpolation

The next result uses the idea of substitution to prove something rather deep. When working with an argument, one can get the idea that there is an 'active core' to the argument, which should be distinguished from some of the irrelevant surrounding matter. The next result gives some formal content to this claim

I shall start with a little result which might be put as telling us the following: if I am to draw a conclusion *from* some claim, there had better be something in the original claim that my conclusion builds upon. More precisely:

Lemma 2.3. Suppose $\boldsymbol{A} \vDash \boldsymbol{B}$, that $\boldsymbol{A}$ is not a contradiction, and that $\boldsymbol{B}$ is not a tautology. Then at least one atomic sentence occurs in both $\boldsymbol{A}$ and $\boldsymbol{B}$.

Proof. Supposing $\boldsymbol{A}$ is not a contradiction, there is a valuation, v, that makes $\boldsymbol{A}$ true. Supposing $\boldsymbol{B}$ is not a tautology, there is a valuation, w, that makes $\boldsymbol{B}$ false. Supposing also $\boldsymbol{A}$ and $\boldsymbol{B}$ have no atomic sentences in common, then there is a valuation, x, which assigns the same values as v to all atomic sentence letters in $\boldsymbol{A}$, and the same values as w to all atomic sentence letters in $\boldsymbol{B}$. By hypothesis, x makes $\boldsymbol{A}$ true and $\boldsymbol{B}$ false. So $\boldsymbol{A} \nvDash \boldsymbol{B}$. ■

Our main result is going to employ a (fairly lengthy) induction to extend Lemma 2.3 rather radically. First, we need a definition. Suppose that $\boldsymbol{A} \vDash \boldsymbol{B}$. Then an INTERPOLANT linking $\boldsymbol{A}$ to $\boldsymbol{B}$ is any sentence $\boldsymbol{I}$ meeting all three of these conditions:

(INT1) $\boldsymbol{A} \vDash \boldsymbol{I}$
(INT2) $\boldsymbol{I} \vDash \boldsymbol{B}$
(INT3) every atomic sentence in $\boldsymbol{I}$ occurs both in $\boldsymbol{A}$ and in $\boldsymbol{B}$

Our result guarantees the existence of interpolants.

Theorem 2.4. Interpolation Theorem. Suppose $\boldsymbol{A} \vDash \boldsymbol{B}$, that $\boldsymbol{A}$ is not a contradiction, and that $\boldsymbol{B}$ is not a tautology. Then there is an interpolant linking $\boldsymbol{A}$ to $\boldsymbol{B}$.

Proof. Our proof consists in an (ordinary) induction on the number of atomic sentences that occur in $\boldsymbol{A}$ but not $\boldsymbol{B}$.

In the base case, there are no atomic sentences in $\boldsymbol{A}$ that are not also in $\boldsymbol{B}$. Hence we can let $\boldsymbol{A}$ itself be our interpolant, since $\boldsymbol{A} \vDash \boldsymbol{A}$ vacuously, and $\boldsymbol{A} \vDash \boldsymbol{B}$ by hypothesis.

Now fix n, and suppose (for induction) that we can find an interpolant whenever there are n atomic sentences that occur in the 'premise' but not the 'conclusion' of the entailment. Additionally, suppose that $\boldsymbol{A} \vDash \boldsymbol{B}$, that $\boldsymbol{A}$ is not a contradiction, that $\boldsymbol{B}$ is not a tautology, and that there are $n + 1$ atomic sentences that occur in $\boldsymbol{A}$ but

not in $\boldsymbol{B}$. Lemma 2.3 tells us that there is at least one atomic sentence, $\boldsymbol{C}$, common to both $\boldsymbol{A}$ and $\boldsymbol{B}$. Now, where:

- $\boldsymbol{D}$ is any sentence that occurs in $\boldsymbol{A}$ but not in $\boldsymbol{B}$
- $\top$ is a tautology which contains no atomic sentence apart from $\boldsymbol{C}$, e.g. $(\boldsymbol{C} \rightarrow \boldsymbol{C})$
- $\bot$ is a contradiction which contains no atomic sentence apart from $\boldsymbol{C}$, e.g. $(\boldsymbol{C} \wedge \neg \boldsymbol{C})$

I make the following claim:

Claim 1: $(\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}]) \vDash \boldsymbol{B}$

To prove Claim 1, let v be any valuation that makes $(\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}])$ true. If v makes $\boldsymbol{A}[\top/\boldsymbol{D}]$ true, let w differ from v just by making $\boldsymbol{D}$ true; otherwise, let w differ from v just by making $\boldsymbol{D}$ false. Either way, w makes $\boldsymbol{A}$ true, so w makes $\boldsymbol{B}$ true too (since $\boldsymbol{A} \vDash \boldsymbol{B}$). Finally, since $\boldsymbol{D}$ does not occur in $\boldsymbol{B}$, v makes $\boldsymbol{B}$ true too. This establishes the Claim.

Next, because there were $n + 1$ atomic sentences in $\boldsymbol{A}$ but not $\boldsymbol{B}$, and $\boldsymbol{D}$ was one of these atomic sentences, there must be exactly n atomic sentences in $\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}]$ but not $\boldsymbol{B}$. Hence, by Claim 1 and the induction hypothesis, we can find an interpolant, $\boldsymbol{I}$, linking $(\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}])$ to $\boldsymbol{B}$, i.e.:

1. $(\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}]) \vDash \boldsymbol{I}$
2. $\boldsymbol{I} \vDash \boldsymbol{B}$
3. every atomic sentence in $\boldsymbol{I}$ occurs in both $(\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}])$ and in $\boldsymbol{B}$

I shall now show that $\boldsymbol{I}$ is an interpolant linking $\boldsymbol{A}$ itself to $\boldsymbol{B}$. Point (2), above, tells us that $\boldsymbol{I} \vDash \boldsymbol{B}$. Moreover, given point (3), it is obvious that every atomic sentence in $\boldsymbol{I}$ occurs in both $\boldsymbol{A}$ and in $\boldsymbol{B}$. So it remains to show that $\boldsymbol{A} \vDash \boldsymbol{I}$. Given that $(\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}]) \vDash \boldsymbol{I}$, it suffices, by the transitivity of entailment, to prove the following claim:

Claim 2. $\boldsymbol{A} \vDash (\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}])$

To prove Claim 2, let v be any valuation that makes $\boldsymbol{A}$ true. If v makes $\boldsymbol{D}$ true, then since $\top$ is a tautology, v makes $\boldsymbol{A}[\top/\boldsymbol{D}]$ true. If v makes $\boldsymbol{D}$ false, then v makes $\boldsymbol{A}[\bot/\boldsymbol{D}]$ true. Either way, any valuation that makes $\boldsymbol{A}$ true makes $(\boldsymbol{A}[\top/\boldsymbol{D}] \vee \boldsymbol{A}[\bot/\boldsymbol{D}])$ true. This establishes Claim 2.

Hence $\boldsymbol{I}$ is an interpolant linking $\boldsymbol{A}$ to $\boldsymbol{B}$. This completes the proof of the induction case. The entire Theorem now follows by induction. ■

This is easily the longest proof that we have seen so far. So it is worth pausing, to make sure both that you understand every step in the proof, and also how the proof is put together.

In fact, this proof merits particular scrutiny since, implicitly, the proof does more than *just* prove the Theorem. The Interpolation Theorem states that, when $\boldsymbol{A} \vDash \boldsymbol{B}$, an interpolant *exists*. If this were all we knew, then we might have no idea about how to *find* an interpolant. But, viewed correctly, the proof yields an algorithm for

finding an interpolant. An ALGORITHM is a step-by-step process, followed 'thoughtlessly' (i.e. without any need for insight, just by implementing very simple little rules), that is guaranteed to terminate in finitely many steps. I shall describe the algorithm for finding an interpolant; and having done so, it should be clear how it relates to the proof.

Given that $\boldsymbol{A} \vDash \boldsymbol{B}$, linking $\boldsymbol{A}$ to $\boldsymbol{B}$ as follows:

Step 1: Designate $\boldsymbol{A}$ as the *input* sentence

Step 2: Let $\boldsymbol{D}$ be the first atomic sentence (alphabetically) that appears in the input sentence, but that does not appear in $\boldsymbol{B}$. Rewrite the input sentence twice, once uniformly substituting a tautology in place of $\boldsymbol{D}$ and once uniformly substituting a contradiction in place of $\boldsymbol{D}$. Let the disjunction of these two rewritten sentences be your new input sentence.

Step 3: Repeat step 2, until there are no atomic sentences that appear in the input sentence but not in $\boldsymbol{B}$. At that point, stop. The last sentence you wrote down is an interpolant linking $\boldsymbol{A}$ to $\boldsymbol{B}$.

It is worth seeing this in action. Suppose we start with the entailment:

$$((A \wedge (\neg B \vee C)) \wedge (\neg C \wedge D)) \vDash ((B \vee D) \wedge (\neg B \vee E))$$

Running the algorithm, our initial input sentence is:

$$((A \wedge (\neg B \vee C)) \wedge (\neg C \wedge D))$$

We would first eliminate A, obtaining (using '$\top$' to abbreviate some tautology containing 'B', and '$\bot$' to abbreviate some contradiction containing 'B'):

$$(((\top \wedge (\neg B \vee C)) \wedge (\neg C \wedge D)) \vee ((\bot \wedge (\neg B \vee C)) \wedge (\neg C \wedge D)))$$

This would become our new input sentence; from which we would eliminate C:

$$((((\top \wedge (\neg B \vee \top)) \wedge (\neg \top \wedge D)) \vee ((\bot \wedge (\neg B \vee \top)) \wedge (\neg \top \wedge D))) \vee$$
$$(((\top \wedge (\neg B \vee \bot)) \wedge (\neg \bot \wedge D)) \vee ((\bot \wedge (\neg B \vee \bot)) \wedge (\neg \bot \wedge D))))$$

And this is our interpolant! But, as the example illustrates, the output of this algorithm can be quite a mouthful. To simplify it, we can apply some 'clean-up' rules. Where '$\top$' abbreviates any tautology, and '$\bot$' abbreviates any contradiction, note, for example, that:

$$\top \vee \boldsymbol{A} \dashv\vDash \top$$
$$\bot \wedge \boldsymbol{A} \dashv\vDash \bot$$
$$\neg \top \dashv\vDash \bot$$
$$\neg \bot \dashv\vDash \top$$

Using these equivalences and Lemma 2.2, we can simplify the subsentences of our interpolant.[1] First, applying the rules for negation:

$$((((\top \wedge (\neg B \vee \top)) \wedge (\bot \wedge D)) \vee ((\bot \wedge (\neg B \vee \top)) \wedge (\bot \wedge D))) \vee$$
$$(((\top \wedge (\neg B \vee \bot)) \wedge (\top \wedge D)) \vee ((\bot \wedge (\neg B \vee \bot)) \wedge (\top \wedge D))))$$

[1] Recall that a sentence $\boldsymbol{A}$ is a SUBSENTENCE of $\boldsymbol{B}$ *iff* $\boldsymbol{A}$ is a part of $\boldsymbol{B}$. We here read 'part' in a generous sense, so that every sentence is a part of itself.

Next, applying the rules for conjunction and disjunction:

$$((((\top \wedge \top) \wedge \bot) \vee (\bot \wedge \bot)) \vee (((\top \wedge \neg B) \wedge D) \vee (\bot \wedge D)))$$

And, then, applying them again and again, we get, successively:

$$\begin{aligned} &((\bot \vee \bot) \vee ((\neg B \wedge D) \vee \bot)) \\ &(\bot \vee (\neg B \wedge D)) \\ &(\neg B \wedge D) \end{aligned}$$

Which is much simpler!

2.3 Duality

The final result of this chapter will use substitution to explore a very powerful idea. To explain the idea, recall the following equivalences, known as the De Morgan Laws:

$$\begin{aligned} (\boldsymbol{A} \wedge \boldsymbol{B}) &\dashv\vDash \neg(\neg \boldsymbol{A} \vee \neg \boldsymbol{B}) \\ (\boldsymbol{A} \vee \boldsymbol{B}) &\dashv\vDash \neg(\neg \boldsymbol{A} \wedge \neg \boldsymbol{B}) \end{aligned}$$

These indicate that there is a tight 'symmetry' between conjunction and disjunction. The notion of *duality* is a generalisation of this tight 'symmetry'. And in fact, it is worth introducing it with complete generality at the outset.

In TFL, our only primitive connectives are one-place (i.e. '$\neg$') and two-place (i.e. '$\wedge$', '$\vee$', '$\rightarrow$' and '$\leftrightarrow$'). But nothing stops us from introducing three-, four-, or five-place connectives; or, more generally, n-place connectives, for any number n we like. (We shall, however, always assume that our connectives are *truth-functional*.)

So, let $\triangleleft$ be an n-place connective. The DUAL of $\triangleleft$, which I shall generically denote by underlining it, i.e. $\underline{\triangleleft}$, is defined to have a characteristic truth table as follows:

$$\underline{\triangleleft}(\boldsymbol{A}_1, ..., \boldsymbol{A}_n) \dashv\vDash \neg\triangleleft(\neg \boldsymbol{A}_1, ..., \neg \boldsymbol{A}_n)$$

This is the obvious generalisation of the De Morgan Laws to many-place connectives.[2] And it is immediate, on this definition, that '$\wedge$' is the dual of '$\vee$', and that '$\vee$' is the dual of '$\wedge$'. I encourage you also to check that '$\uparrow$' is the dual of '$\downarrow$', and vice versa; and that '$\neg$' is its *own* dual.

Inspired by these examples, we might correctly conjecture the following:

Lemma 2.5. The dual of a dual is always the original connective; that is, $\underline{\underline{\triangleleft}}$ is the same connective as $\triangleleft$.

Proof. I start by noting:

$$\begin{aligned} \underline{\triangleleft}(\boldsymbol{A}_1, ..., \boldsymbol{A}_n) &\dashv\vDash \neg\triangleleft(\neg \boldsymbol{A}_1, ..., \neg \boldsymbol{A}_n) && \text{by definition} \\ \underline{\triangleleft}(\neg \boldsymbol{A}_1, ..., \neg \boldsymbol{A}_n) &\dashv\vDash \neg\triangleleft(\neg\neg \boldsymbol{A}_1, ..., \neg\neg \boldsymbol{A}_n) && \text{by Lemma 2.1} \\ \neg\underline{\triangleleft}(\neg \boldsymbol{A}_1, ..., \neg \boldsymbol{A}_n) &\dashv\vDash \neg\neg\triangleleft(\neg\neg \boldsymbol{A}_1, ..., \neg\neg \boldsymbol{A}_n) && \text{by the definition of '}\neg\text{'} \\ \neg\underline{\triangleleft}(\neg \boldsymbol{A}_1, ..., \neg \boldsymbol{A}_n) &\dashv\vDash \triangleleft(\boldsymbol{A}_1, ..., \boldsymbol{A}_n) && \text{by Lemma 2.2} \end{aligned}$$

[2] Hence some books talk of the 'De Morgan dual', rather than simply of the 'dual'.

Next, observe that, by definition:

$$\underline{\underline{\triangleleft}}(\boldsymbol{A}_1, ..., \boldsymbol{A}_n) \dashv\vDash \neg\underline{\triangleleft}(\neg\boldsymbol{A}_1, ..., \neg\boldsymbol{A}_n)$$

Hence, by the transitivity of entailment:

$$\underline{\underline{\triangleleft}}(\boldsymbol{A}_1, ..., \boldsymbol{A}_n) \dashv\vDash \triangleleft(\boldsymbol{A}_1, ..., \boldsymbol{A}_n)$$

as required. ■

We shall use the same kind of proof-technique in what follows, to great effect.

First, we need a little more notation, connecting duality with negation. Given any sentence, $\boldsymbol{A}$, we say $\underline{\boldsymbol{A}}$ is the result of replacing *every* connective in $\boldsymbol{A}$ with its dual, and $\overline{\boldsymbol{A}}$ is the result of replacing every atomic sentence in $\boldsymbol{A}$ with the negation of that atomic sentence. So, when $\boldsymbol{A}$ is the sentence '$(A \vee (\neg B \wedge C))$', $\underline{\boldsymbol{A}}$ is '$(A \wedge (\neg B \vee C))$' and $\overline{\boldsymbol{A}}$ is '$(\neg A \vee (\neg\neg B \wedge \neg C))$'.

Theorem 2.6. $\underline{\boldsymbol{A}} \dashv\vDash \neg\overline{\boldsymbol{A}}$, for any sentence $\boldsymbol{A}$.

Proof. Let $\boldsymbol{A}$ be any sentence; we suppose, for induction on length, that every shorter sentence $\boldsymbol{B}$ is such that $\underline{\boldsymbol{B}} \dashv\vDash \neg\overline{\boldsymbol{B}}$. There are two cases to consider:

Case 1: $\boldsymbol{A}$ is atomic. Then $\underline{\boldsymbol{A}}$ is just $\boldsymbol{A}$ itself; and $\neg\overline{\boldsymbol{A}}$ is just $\neg\neg\boldsymbol{A}$; and we know that $\boldsymbol{A} \dashv\vDash \neg\neg\boldsymbol{A}$.

Case 2: $\boldsymbol{A}$ is $\triangleleft(\boldsymbol{B}_1, ..., \boldsymbol{B}_n)$, for some n-place connective $\triangleleft$. So by hypothesis, $\underline{\boldsymbol{B}_i} \dashv\vDash \neg\overline{\boldsymbol{B}_i}$ for each $\boldsymbol{B}_i$. Now observe that:

$$\begin{aligned}
\underline{\triangleleft(\boldsymbol{B}_1, ..., \boldsymbol{B}_n)} &\dashv\vDash \underline{\triangleleft}(\underline{\boldsymbol{B}_1}, ..., \underline{\boldsymbol{B}_n}) && \text{by definition} \\
&\dashv\vDash \underline{\triangleleft}(\neg\overline{\boldsymbol{B}_1}, ..., \neg\overline{\boldsymbol{B}_n}) && \text{by Lemma 2.2} \\
&\dashv\vDash \neg\triangleleft(\neg\neg\overline{\boldsymbol{B}_1}, ..., \neg\neg\overline{\boldsymbol{B}_n}) && \text{by definition of } \underline{\triangleleft} \\
&\dashv\vDash \neg\triangleleft(\overline{\boldsymbol{B}_1}, ..., \overline{\boldsymbol{B}_n}) && \text{by Lemma 2.2} \\
&\dashv\vDash \neg\overline{\triangleleft(\boldsymbol{B}_1, ..., \boldsymbol{B}_n)} && \text{by definition}
\end{aligned}$$

and hence, given the transitivity of entailment, $\underline{\boldsymbol{A}} \dashv\vDash \neg\overline{\boldsymbol{A}}$.

This completes the induction on length. ■

And we can now obtain a very elegant result:

Theorem 2.7. $\boldsymbol{A} \vDash \boldsymbol{B}$ iff $\underline{\boldsymbol{B}} \vDash \underline{\boldsymbol{A}}$, for any sentences $\boldsymbol{A}$ and $\boldsymbol{B}$. Hence, in particular, $\boldsymbol{A} \dashv\vDash \boldsymbol{B}$ iff $\underline{\boldsymbol{A}} \dashv\vDash \underline{\boldsymbol{B}}$.

Proof. Left to right. Suppose $\boldsymbol{A} \vDash \boldsymbol{B}$. Then $\overline{\boldsymbol{A}} \vDash \overline{\boldsymbol{B}}$, by Lemma 2.1. So $\neg\overline{\boldsymbol{B}} \vDash \neg\overline{\boldsymbol{A}}$. So $\underline{\boldsymbol{B}} \vDash \underline{\boldsymbol{A}}$, by Theorem 2.6 and Lemma 2.2.

Right to left. Suppose $\underline{\boldsymbol{B}} \vDash \underline{\boldsymbol{A}}$. Then, by the preceding, $\underline{\underline{\boldsymbol{A}}} \vDash \underline{\underline{\boldsymbol{B}}}$. Hence $\boldsymbol{A} \vDash \boldsymbol{B}$, by Lemmas 2.5 and 2.2. ■

This ends my theorem-proving. But to close the chapter, I want to offer some sketch some reasons for thinking that duality is pretty great.

Illustration 1. You may have already noticed the following equivalences:

$$(\boldsymbol{A} \wedge (\boldsymbol{B} \vee \boldsymbol{C})) \dashv\vDash ((\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\boldsymbol{A} \wedge \boldsymbol{C}))$$
$$((\boldsymbol{A} \vee \boldsymbol{B}) \wedge \boldsymbol{C}) \dashv\vDash ((\boldsymbol{A} \wedge \boldsymbol{C}) \vee (\boldsymbol{B} \wedge \boldsymbol{C}))$$

If not, then you can test these using a truth table. But, having done so, Theorem 2.7 immediately yields two further equivalences for free:

$$(\boldsymbol{A} \vee (\boldsymbol{B} \wedge \boldsymbol{C})) \dashv\vDash ((\boldsymbol{A} \vee \boldsymbol{B}) \wedge (\boldsymbol{A} \vee \boldsymbol{C}))$$
$$((\boldsymbol{A} \wedge \boldsymbol{B}) \vee \boldsymbol{C}) \dashv\vDash ((\boldsymbol{A} \vee \boldsymbol{C}) \wedge (\boldsymbol{B} \vee \boldsymbol{C}))$$

These four equivalences are the Distribution Laws. We shall use them repeatedly in the next chapter.

Illustration 2. Here is a playful way to think about negation, due to Frank Plumpton Ramsey.[3] Suppose that we allow ourselves to represent a sentence's negation by flipping the entire sentence upside-down, rather than by prefixing the sentence with '¬'. With this notation, and supposing '♡' to be an *n*-place connective, we would have:

$$\begin{aligned}
\underline{\heartsuit}(\boldsymbol{A}_1, ..., \boldsymbol{A}_n) &\dashv\vDash \neg\heartsuit(\neg\boldsymbol{A}_1, ..., \neg\boldsymbol{A}_n) && \text{the original definition}\\
&\dashv\vDash \heartsuit(\boldsymbol{\forall}^{\mathrm{I}}, ..., \boldsymbol{\forall}^{\mathrm{u}}) && \text{replacing the interior negations}\\
&\dashv\vDash \diamondsuit(\boldsymbol{A}_1\text{'}...\text{'}\boldsymbol{A}_n) && \text{replacing the outer negation}
\end{aligned}$$

and with a little imagination concerning punctuation (commas and brackets), we see that the same effect could be achieved more simply by merely inverting the connective itself, i.e. we would represent the dual of '♡' by '♢'. A 'fully perspicuous' notation would, then, represent connectives and their duals by symbols which are mirror-images of each other. But that is exactly how we *do* represent conjunction and disjunction![4] Note, too, that if we adopted this notation, then Lemma 2.5 would be *visually* immediate. And this suggests (correctly) that the success of our Ramsey-inspired notation system depends only upon Lemma 2.2 and the equivalence: $\neg\neg\boldsymbol{A} \dashv\vDash \boldsymbol{A}$. Once we realise this, we can see that *all* of our results about duality are extremely general: they hold for any extensional language which respects the equivalence $\neg\neg\boldsymbol{A} \dashv\vDash \boldsymbol{A}$ and in which Lemmas 2.1 and 2.2 hold.

Illustration 3. An immediate consequence of our results so far is

$$\boldsymbol{A} \dashv\vDash \neg\overline{\underline{\boldsymbol{A}}}$$

I leave the proof of this consequence as an exercise. But the consequence prompts a subtle thought. Typically, when I hear you say 'London is the capital of the UK', or whatever, I take you to mean *that* London is the capital of the UK, and to be

[3] 'Facts and Propositions' (1927, *Proceedings of the Aristotelian Society Supplementary Volume* 7.1).

[4] There is something, then, to be said for representing negation with a horizontally symmetric '–'. But I have no great ideas for how to represent (bi)conditionals.

asserting this. What if, instead, I took you to mean that London is *not* the capital of the UK, and to be *rejecting* this? More generally, where before I took you to be *asserting* that $\boldsymbol{A}$, I now take you to be *rejecting* that $\overline{\underline{\boldsymbol{A}}}$. Would anything be wrong with my reinterpretation? If not: why should I assume that you mean *and*, rather than *or*, by the word 'and'?[5]

Practice exercises

A. Complete the inductions on length mentioned in the proofs of Lemmas 2.1 and 2.2.

B. Prove that $\boldsymbol{A} \dashv\models \neg\overline{\underline{\boldsymbol{A}}}$, for any sentence $\boldsymbol{A}$.

[5] For discussion of all this, see Gerald Massey, 'The Indeterminacy of Translation', pp.329ff. (1992, *Philosophical Topics* 20.1) and Hilary Putnam, 'Replies', pp.399–402 (1992, *Philosophical Topics* 20.1).

Normal forms 3

In this chapter, I prove two *normal form* theorems for TFL. These guarantee that, for any sentence, there is a tautologically equivalent sentence in some canonical normal form. Moreover, I shall give methods for finding these normal-form equivalents.

3.1 Disjunctive Normal Form

Say that a sentence is in DISJUNCTIVE NORMAL FORM *iff* it meets all of the following conditions:

(DNF1) No connectives occur in the sentence other than negations, conjunctions and disjunctions;
(DNF2) Every occurrence of negation has minimal scope (i.e. any '$\neg$' is immediately followed by an atomic sentence);
(DNF3) No disjunction occurs within the scope of any conjunction.

(For a reminder of the definition of the SCOPE of a connective, see forall***x***:Cambridge §6.) So, here are are some sentences in disjunctive normal form:

$$A$$
$$(A \wedge B) \vee (A \wedge \neg B)$$
$$(A \wedge B) \vee (A \wedge B \wedge C \wedge \neg D \wedge \neg E)$$
$$A \vee (C \wedge \neg P_{234} \wedge P_{233} \wedge Q) \vee \neg B$$

Note that I have here broken one of the maxims of this book (see §1.4) and *temporarily* allowed myself to employ the relaxed bracketing-conventions that allow conjunctions and disjunctions to be of arbitrary length (see forall***x***:Cambridge §10.3). These conventions make it easier to see when a sentence is in disjunctive normal form. I shall continue to help myself to these relaxed conventions, without further comment, until §3.3.

To further illustrate the idea of disjunctive normal form, I shall introduce some more notation. I write '$\pm\boldsymbol{A}$' to indicate that $\boldsymbol{A}$ is an atomic sentence which may or may not be prefaced with an occurrence of negation. Then a sentence in disjunctive normal form has the following shape:

$$(\pm\boldsymbol{A}_1 \wedge \ldots \wedge \pm\boldsymbol{A}_i) \vee (\pm\boldsymbol{A}_{i+1} \wedge \ldots \wedge \pm\boldsymbol{A}_j) \vee \ldots \vee (\pm\boldsymbol{A}_{m+1} \wedge \ldots \wedge \pm\boldsymbol{A}_n)$$

We now know what it is for a sentence to be in disjunctive normal form. The result that we are aiming at is:

Theorem 3.1. Disjunctive Normal Form Theorem. For any sentence, there is a tautologically equivalent sentence in disjunctive normal form.

Henceforth, I shall abbreviate 'Disjunctive Normal Form' by 'DNF'.

3.2 Proof of DNF Theorem via truth tables

My first proof of the DNF Theorem employs truth tables. I shall first illustrate the technique for finding an equivalent sentence in DNF, and then turn this illustration into a rigorous proof.

Let's suppose we have some sentence, $\boldsymbol{S}$, which contains three atomic sentences, 'A', 'B' and 'C'. The very first thing to do is fill out a complete truth table for $\boldsymbol{S}$. Maybe we end up with this:

A	B	C	$\boldsymbol{S}$
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	T

As it happens, $\boldsymbol{S}$ is true on four lines of its truth table, namely lines 1, 3, 7 and 8. Corresponding to each of those lines, I shall write down four sentences, whose only connectives are negations and conjunctions, where every negation has minimal scope:

- '$A \wedge B \wedge C$' which is true on line 1 (and only then)
- '$A \wedge \neg B \wedge C$' which is true on line 3 (and only then)
- '$\neg A \wedge \neg B \wedge C$' which is true on line 7 (and only then)
- '$\neg A \wedge \neg B \wedge \neg C$' which is true on line 8 (and only then)

But if I now disjoin all of these conjunctions, like so:

$$(A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C)$$

I have a sentence in DNF which is true on exactly those lines where one of the disjuncts is true, i.e. it is true on (and only on) lines 1, 3, 7, and 8. So this sentence has exactly the same truth table as $\boldsymbol{S}$. So I have a sentence in DNF that is tautologically equivalent to $\boldsymbol{S}$. Which is exactly what I required!

Now, the strategy that I just adopted did not depend on the specifics of $\boldsymbol{S}$; it is perfectly general. Consequently, we can use it to obtain a simple proof of the DNF Theorem.

Proof of DNF Theorem via truth tables. Pick any arbitrary sentence, $\boldsymbol{S}$, and let $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_n$ be the atomic sentences that occur in $\boldsymbol{S}$. To obtain a sentence in DNF

that is tautologically equivalent $\boldsymbol{S}$, we consider $\boldsymbol{S}$'s truth table. There are two cases to consider:

> *Case 1: $\boldsymbol{S}$ is false on every line of its truth table.* Then, $\boldsymbol{S}$ is a contradiction. In that case, the contradiction $(\boldsymbol{A}_1 \wedge \neg \boldsymbol{A}_1) \dashv\vDash \boldsymbol{S}$, and $(\boldsymbol{A}_1 \wedge \neg \boldsymbol{A}_1)$ is in DNF.
> *Case 2: $\boldsymbol{S}$ is true on at least one line of its truth table.* For each line i of the truth table, let $\boldsymbol{B}_i$ be a conjunction of the form
>
> $$(\pm \boldsymbol{A}_1 \wedge \ldots \wedge \pm \boldsymbol{A}_n)$$
>
> where the following rules determine whether or not to include a negation in front of each atomic sentence:
>
> $$\boldsymbol{A}_m \text{ is a conjunct of } \boldsymbol{B}_i \textit{ iff } \boldsymbol{A}_m \text{ is true on line } i$$
> $$\neg \boldsymbol{A}_m \text{ is a conjunct of } \boldsymbol{B}_i \textit{ iff } \boldsymbol{A}_m \text{ is false on line } i$$
>
> Given these rules, a trivial proof by induction shows that $\boldsymbol{B}_i$ is true on (and only on) line i of the truth table which considers all possible valuations of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_n$ (i.e. $\boldsymbol{S}$'s truth table).
> Next, let $i_1, i_2, \ldots, i_m$ be the numbers of the lines of the truth table where $\boldsymbol{S}$ is *true*. Now let $\boldsymbol{D}$ be the sentence:
>
> $$\boldsymbol{B}_{i_1} \vee \boldsymbol{B}_{i_2} \vee \ldots \vee \boldsymbol{B}_{i_m}$$
>
> Since $\boldsymbol{S}$ is true on at least one line of its truth table, $\boldsymbol{D}$ is indeed well-defined; and in the limiting case where $\boldsymbol{S}$ is true on exactly one line of its truth table, $\boldsymbol{D}$ is just $\boldsymbol{B}_{i_1}$, for some i_1.
> By construction, $\boldsymbol{D}$ is in DNF. Moreover, by construction, for each line i of the truth table: $\boldsymbol{S}$ is true on line i of the truth table *iff* one of $\boldsymbol{D}$'s disjuncts (namely, $\boldsymbol{B}_i$) is true on, and only on, line i. (Again, this is shown by a trivial proof by induction.) Hence $\boldsymbol{S}$ and $\boldsymbol{D}$ have the same truth table, and so are tautologically equivalent.

These two cases are exhaustive and, either way, we have a sentence in DNF that is tautologically equivalent to $\boldsymbol{S}$. ■

So we have proved the DNF Theorem. Before I say any more, though, I should immediately flag that I am hereby returning to the austere definition of a (TFL) sentence, according to which we can assume that any conjunction has exactly two conjuncts, and any disjunction has exactly two disjuncts.[1]

3.3 Proof of DNF Theorem via substitution

I now want to offer a second proof of the DNF Theorem.

At this point, you might reasonably ask: Why bother with *two* proofs? Certainly one proof is sufficient to establish a result. But there are several possible reasons for offering new proofs of known results. For example:

[1] NB: if you did the practice exercises for chapter 1, you will know how to justify these conventions rigorously; and you will also, essentially, have performed the two 'trivial proofs by induction' mentioned in the preceding proof.

- a new proof might offer more insight (or different insights) than the original proof;
- a new proof-technique might be useful in other ways: it might help you to prove other results (that you have not yet proved);
- sheer curiosity;

and more besides. All these reasons apply here.

Our first proof made use of truth tables. Our second proof will employ Lemma 2.2. The aim is to define an *algorithm* which will slowly replace subsentences with tautologically equivalent subsentences, pushing us in the direction of a DNF sentence, whilst preserving tautological equivalence, until eventually we end up with a tautologically equivalent sentence in DNF. (For a reminder about what an algorithm is, see §2.2.)

That is the big idea. But the details are conceptually trickier than those in the first proof. So we shall proceed slowly. I shall start with a relatively informal overview of the algorithm, which we shall call IntoDNF, and shall show how it applies to a particular input sentence. With the ideas fixed by the example, I shall offer a rigorous proof that the algorithm does what we require.

Our algorithm, IntoDNF, takes an input sentence, $\boldsymbol{S}$, and then brings us closer and closer to DNF. The initial step of the algorithm, then, is obvious:

Step 0. Write down your initial sentence, $\boldsymbol{S}$.

To make matters tractable, we shall work a particular application of the algorithm as we describe it. So, let us suppose that this is our input sentence:

$$((\neg A \to \neg(B \vee \neg C)) \to \neg(D \leftrightarrow \neg E))$$

The next step of the algorithm is designed to yield a tautologically equivalent sentence which satisfies (DNF1). Otherwise put, our aim is to eliminate all occurrences of conditionals and biconditionals from a sentence, whilst preserving tautological equivalence. So, here is what we should do:

Step 1a. Replace all subsentences of the form $(\boldsymbol{A} \to \boldsymbol{B})$ with $(\neg\boldsymbol{A} \vee \boldsymbol{B})$.
Step 1b. Replace all subsentences of the form $(\boldsymbol{A} \leftrightarrow \boldsymbol{B})$ with $((\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\neg\boldsymbol{A} \wedge \neg\boldsymbol{B}))$.

Here is the result of running through Step 1 on our example. (The markers on the left-side indicate the Step we have just finished.)

$$\begin{aligned}
0: &\quad ((\neg A \to \neg(B \vee \neg C)) \to \neg(D \leftrightarrow \neg E)) \\
1a: &\quad (\neg(\neg\neg A \vee \neg(B \vee \neg C)) \vee \neg(D \leftrightarrow \neg E)) \\
1b: &\quad (\neg(\neg\neg A \vee \neg(B \vee \neg C)) \vee \neg((D \wedge \neg E) \vee (\neg D \wedge \neg\neg E)))
\end{aligned}$$

We are a little closer to DNF, since this clearly satisfies (DNF1).

The next step of IntoDNF is designed to bring us to a tautologically equivalent sentence which satisfies (DNF2). Roughly, we need to 'push' the negations as deeply into the sentence as we possibly can. This is what the second step of IntoDNF does:

Step 2a: Replace all subsentences of the form $\neg\neg\boldsymbol{A}$ with $\boldsymbol{A}$.

Step 2b: Replace the first subsentence of the form $\neg(\boldsymbol{A} \wedge \boldsymbol{B})$ with $(\neg\boldsymbol{A} \vee \neg\boldsymbol{B})$.[2]
Step 2c: Replace the first subsentence of the form $\neg(\boldsymbol{A} \vee \boldsymbol{B})$ with $(\neg\boldsymbol{A} \wedge \neg\boldsymbol{B})$.
Step 2d: Repeat Steps 2a–2c, until no further change can occur.[3]

Applying Step 2 to to our example, we get the following sequence:

$$
\begin{aligned}
1:&\quad (\neg(\neg\neg A \vee \neg(B \vee \neg C)) \vee \neg((D \wedge \neg E) \vee (\neg D \wedge \neg\neg E)))\\
2a:&\quad (\neg(A \vee \neg(B \vee \neg C)) \vee \neg((D \wedge \neg E) \vee (\neg D \wedge E)))\\
2b:&\quad (\neg(A \vee \neg(B \vee \neg C)) \vee \neg((D \wedge \neg E) \vee (\neg D \wedge E)))\\
2c:&\quad ((\neg A \wedge \neg\neg(B \vee \neg C)) \vee \neg((D \wedge \neg E) \vee (\neg D \wedge E)))\\
2a:&\quad ((\neg A \wedge (B \vee \neg C)) \vee \neg((D \wedge \neg E) \vee (\neg D \wedge E)))\\
2b:&\quad ((\neg A \wedge (B \vee \neg C)) \vee \neg((D \wedge \neg E) \vee (\neg D \wedge E)))\\
2c:&\quad ((\neg A \wedge (B \vee \neg C)) \vee (\neg(D \wedge \neg E) \wedge \neg(\neg D \wedge E)))\\
2a:&\quad ((\neg A \wedge (B \vee \neg C)) \vee (\neg(D \wedge \neg E) \wedge \neg(\neg D \wedge E)))\\
2b:&\quad ((\neg A \wedge (B \vee \neg C)) \vee ((\neg D \wedge \neg\neg E) \wedge \neg(\neg D \wedge E)))\\
2c:&\quad ((\neg A \wedge (B \vee \neg C)) \vee ((\neg D \vee \neg\neg E) \wedge \neg(\neg D \wedge E)))\\
2a:&\quad ((\neg A \wedge (B \vee \neg C)) \vee ((\neg D \vee E) \wedge \neg(\neg D \wedge E)))\\
2b:&\quad ((\neg A \wedge (B \vee \neg C)) \vee ((\neg D \vee E) \wedge (\neg\neg D \vee \neg E)))\\
2c:&\quad ((\neg A \wedge (B \vee \neg C)) \vee ((\neg D \vee E) \wedge (\neg\neg D \vee \neg E)))\\
2a:&\quad ((\neg A \wedge (B \vee \neg C)) \vee ((\neg D \vee E) \wedge (D \vee \neg E)))
\end{aligned}
$$

Note that the sentence does not necessarily change at every sub-step. But after the last written step, no further changes *can* occur; so we are done with Step 2. And we are even closer to DNF, since this satisfies (DNF2).

All that remains is to meet condition (DNF3), i.e. to ensure that no disjunction occurs within the scope of any conjunction. Roughly, we need to 'push' the conjunctions into the scope of disjunctions. And that is the idea behind the final step of IntoDNF:

Step 3a: Replace the first subsentence of the form $(\boldsymbol{A} \wedge (\boldsymbol{B} \vee \boldsymbol{C}))$ with $((\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\boldsymbol{A} \wedge \boldsymbol{C}))$.
Step 3b: Replace the first subsentence of the form $((\boldsymbol{A} \vee \boldsymbol{B}) \wedge \boldsymbol{C})$ with $((\boldsymbol{A} \wedge \boldsymbol{C}) \vee (\boldsymbol{B} \wedge \boldsymbol{C}))$.
Step 3c: Repeat Steps 3a–3b, until no further changes can occur.

Applying Step 3 to our example, we get:

$$
\begin{aligned}
2:&\quad ((\neg A \wedge (B \vee \neg C)) \vee ((\neg D \vee E) \wedge (D \vee \neg E)))\\
3a:&\quad (((\neg A \wedge B) \vee (\neg A \wedge \neg C)) \vee ((\neg D \vee E) \wedge (D \vee \neg E)))\\
3b:&\quad (((\neg A \wedge B) \vee (\neg A \wedge \neg C)) \vee ((\neg D \wedge (D \vee \neg E)) \vee (E \wedge (D \vee \neg E))))\\
3a:&\quad (((\neg A \wedge B) \vee (\neg A \wedge \neg C)) \vee (((\neg D \wedge D) \vee (\neg D \wedge \neg E)) \vee (E \wedge (D \vee \neg E))))\\
3b:&\quad (((\neg A \wedge B) \vee (\neg A \wedge \neg C)) \vee (((\neg D \wedge D) \vee (\neg D \wedge \neg E)) \vee (E \wedge (D \vee \neg E))))\\
3a:&\quad (((\neg A \wedge B) \vee (\neg A \wedge \neg C)) \vee (((\neg D \wedge D) \vee (\neg D \wedge \neg E)) \vee ((E \wedge D) \vee (E \wedge \neg E))))
\end{aligned}
$$

[2] Here 'first' means 'first, working from left to right'. In fact, inspecting the proof of Lemma 3.3, it is clear that the *order* in which we perform these manipulations does not matter at all; we just need to do them one at a time.

[3] And we can check *that* no further change can occur, just by repeatedly running through Steps 2a–2c, until changes stop occurring.

And this is in DNF, as may become clearer if we *temporarily* allow ourselves to remove brackets, following the notational conventions of forall*x*:Cambridge §10.3:

$$(\neg A \wedge B) \vee (\neg A \wedge \neg C) \vee (\neg D \wedge D) \vee (\neg D \wedge \neg E) \vee (E \wedge D) \vee (E \wedge \neg E)$$

So we are done! Finally, it is worth noting that this algorithm – although lengthy – was *much* more efficient at generating a DNF sentence than the truth-table method would have been. In fact, the truth table for our example would have 32 lines, 22 of which are true.

So far, I have defined the algorithm, IntoDNF, via Steps 0 through to 3c. I have also provided a worked-application of IntoDNF. But it remains to *prove* that IntoDNF really is fit for purpose.

First, we need to prove that any outputs from IntoDNF are tautologically equivalent to the inputs. This is actually quite simple:

Lemma 3.2. For any input sentence: any outputs of IntoDNF are tautologically equivalent to the input sentence.

Proof. At Step 0, we simply write our input. Each instance of Steps 1a–3b then involves replacing a subsentence with a tautologically equivalent subsentence. In particular, we make use of the following equivalences:

in Step 1a:	$(\boldsymbol{A} \rightarrow \boldsymbol{B}) \dashv\vDash (\neg \boldsymbol{A} \vee \boldsymbol{B})$
in Step 1b:	$(\boldsymbol{A} \leftrightarrow \boldsymbol{B}) \dashv\vDash ((\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\neg \boldsymbol{A} \wedge \neg \boldsymbol{B}))$
in Step 2a:	$\neg\neg \boldsymbol{A} \dashv\vDash \boldsymbol{A}$
in Step 2b:	$\neg(\boldsymbol{A} \wedge \boldsymbol{B}) \dashv\vDash (\neg \boldsymbol{A} \vee \neg \boldsymbol{B})$
in Step 2c:	$\neg(\boldsymbol{A} \vee \boldsymbol{B}) \dashv\vDash (\neg \boldsymbol{A} \wedge \neg \boldsymbol{B})$
in Step 3a:	$(\boldsymbol{A} \wedge (\boldsymbol{B} \vee \boldsymbol{C})) \dashv\vDash ((\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\boldsymbol{A} \wedge \boldsymbol{C}))$
in Step 3b:	$((\boldsymbol{A} \vee \boldsymbol{B}) \wedge \boldsymbol{C}) \dashv\vDash ((\boldsymbol{A} \wedge \boldsymbol{C}) \vee (\boldsymbol{B} \wedge \boldsymbol{C}))$

Now, by Lemma 2.2, if we replace some subsentence of a sentence with a tautologically equivalent subsentences, we obtain a sentence that is tautologically equivalent to what we started with. So any instance of any Step of IntoDNF preserves tautological equivalence. So, provided the algorithm yields an output within finitely many steps, that output is indeed (by a trivial induction) tautologically equivalent to the input. ■

The next task is to show that IntoDNF *always* yields an output. We need to ensure, for example, that the algorithm never gets trapped on an infinite loop. At the same time as we prove this, we shall also show that the output is in DNF, as we would hope.

Unfortunately, the proof is surprisingly difficult. So I shall start with an intuitive overview of how the proof. We will aim, intuitively, to show each of the following:

- *The output of Step 1 is a sentence satisfying* (DNF1).
  This will be fairly trivial.

- *The output of Step 2 is a sentence also satisfying* (DNF2).
  Intuitively, this is because the negation-signs get pushed progressively deeper into the formula, until they precede only atomic sentences.
- *The output of Step 3 is a sentence also satisfying* (DNF3).
  Intuitively, this is because the conjunction-signs get pushed progressively deeper into the formula, until they have no disjunctions in their scope.

That is the main idea. As a student, your aim should be to grasp this main idea fully. Morally speaking, it just *must* be right. And, for just this reason, many books at this level leave matters here, and claim that they have proved the DNF Theorem. But, *just because we can*,[4] here is a full proof, which brings these intuitive points into sharp relief:

Lemma 3.3. On any input, IntoDNF is guaranteed to terminate, yielding an output in DNF.

Proof. Concerning Step 0: trivially, we will complete Step 0 (eventually), since this Step just involves writing down a single (finitely long) input sentence.

Concerning Step 1: The input sentence only contained a finite number of instances of '$\rightarrow$' and '$\leftrightarrow$'. These will all be eliminated (eventually) during Step 1. As such, the output at the end of Step 1 satisfies (DNF1).

Concerning Step 2: This is a bit harder. For each sentence $\boldsymbol{A}$, we define its not-height, $nh(\boldsymbol{A})$, recursively:[5]

$$\begin{aligned} nh(\boldsymbol{A}) &= 1, \text{if } \boldsymbol{A} \text{ is atomic} \\ nh(\neg\boldsymbol{A}) &= 3 \times nh(\boldsymbol{A}) - 1 \\ nh(\boldsymbol{A} \wedge \boldsymbol{B}) = nh(\boldsymbol{A} \vee \boldsymbol{B}) &= nh(\boldsymbol{A}) + nh(\boldsymbol{B}) \end{aligned}$$

I claim that, *if an instance of Step 2a, 2b or 2c has any effect on the sentence, then it reduces the sentence's not-height.* In the case of Step 2a, observe that:

$$\begin{aligned} nh(\neg\neg\boldsymbol{A}) &= 3 \times nh(\neg\boldsymbol{A}) - 1 \\ &= 3 \times \big(3 \times nh(\boldsymbol{A}) - 1\big) - 1 \\ &= 9 \times nh(\boldsymbol{A}) - 4 \end{aligned}$$

and this is greater than $nh(\boldsymbol{A})$ for any $\boldsymbol{A}$, since a sentence's not-height must always be at least 1. In the case of Step 2b, observe that:

$$\begin{aligned} nh(\neg(\boldsymbol{A} \wedge \boldsymbol{B})) &= 3 \times \big(nh(\boldsymbol{A} \wedge \boldsymbol{B})\big) - 1 \\ &= 3 \times \big(nh(\boldsymbol{A}) + nh(\boldsymbol{B})\big) - 1 \end{aligned}$$

[4] Can, and *therefore should*? There is a deep philosophical-cum-technical question: how much rigour does a proof require? But this is a book in formal logic, not philosophical logic; so I leave this as a cliffhanger.

[5] Why do we define not-height this way? The glib answer is: because *it works* in the proof. In fact, this glib answer really says it all. I have deliberately manufactured an arithmetical function, *nh*, with no other end in mind, than that successive applications of Steps 2a–2c of the algorithm are guaranteed to yield a decreasing sequence of *nh* values.

and this is always greater than:

$$\begin{aligned} nh(\neg \boldsymbol{A} \vee \neg \boldsymbol{B}) &= nh(\neg \boldsymbol{A}) + nh(\neg \boldsymbol{B}) \\ &= 3 \times nh(\boldsymbol{A}) - 1 + 3 \times nh(\boldsymbol{B}) - 1 \\ &= 3 \times \big(nh(\boldsymbol{A}) + nh(\boldsymbol{B})\big) - 2 \end{aligned}$$

The case of Step 2c is exactly similar, and this establishes the claim.

Consequently, repeatedly running through Steps 2a–2c yields a sequence of sentences with decreasing not-heights. Since the input sentence to Step 2 (i.e. the output from Step 1) has some finite not-height, after a finite number of applications of Steps 2a–2c, it will be impossible to reduce the not-height any further. This brings Step 2 to an end. Trivially, the resulting sentence still satisfies (DNF1). And it also satisfies (DNF2), since the not-height of any sentence satisfying (DNF1) but not satisfying (DNF2) clearly *can* be decreased by some application of Step 2a, 2b or 2c.

Concerning Step 3: This is like Step 2. For each sentence $\boldsymbol{A}$, we define its conjunction-height, $ch(\boldsymbol{A})$, recursively:

$$\begin{aligned} ch(\boldsymbol{A}) &= 2, \text{if } \boldsymbol{A} \text{ is atomic} \\ ch(\neg \boldsymbol{A}) &= ch(\boldsymbol{A}) \\ ch(\boldsymbol{A} \wedge \boldsymbol{B}) &= ch(\boldsymbol{A}) \times ch(\boldsymbol{B}) \\ ch(\boldsymbol{A} \vee \boldsymbol{B}) &= ch(\boldsymbol{A}) + ch(\boldsymbol{B}) + 1 \end{aligned}$$

I claim that, *if an instance of Step 3a or 3b has any effect on the sentence, then it reduces the sentence's conjunction-height.* In the case of Step 3a, observe that

$$\begin{aligned} ch(\boldsymbol{A} \wedge (\boldsymbol{B} \vee \boldsymbol{C})) &= ch(\boldsymbol{A}) \times \big(ch(\boldsymbol{B}) + ch(\boldsymbol{C}) + 1\big) \\ &= ch(\boldsymbol{A}) \times ch(\boldsymbol{B}) + ch(\boldsymbol{A}) \times ch(\boldsymbol{C}) + ch(\boldsymbol{A}) \end{aligned}$$

and, since $ch(\boldsymbol{A}) > 1$ for any $\boldsymbol{A}$, this must be larger than:

$$ch((\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\boldsymbol{A} \wedge \boldsymbol{C})) = ch(\boldsymbol{A}) \times ch(\boldsymbol{B}) + ch(\boldsymbol{A}) \times ch(\boldsymbol{C}) + 1$$

The case of Step 3b is exactly similar. This establishes the claim.

Now, as above: running through Steps 3a–3b repeatedly yields a sequence of sentences with decreasing conjunction-heights. So, after a finite number of applications of Steps 3a–3b, it will be impossible to reduce the conjunction-height any further. Trivially, the resulting sentence still satisfies (DNF1) and (DNF2). And it also satisfies (DNF3), since the conjunction-height of any sentence satisfying both (DNF1) and (DNF2) but not (DNF3) *can* be decreased by some application of Step 3a or 3b. ■

Now if we just combine combine Lemma 3.2 with Lemma 3.3, we have a new proof of the DNF Theorem, via substitution.

3.4 Cleaning up DNF sentences

I have offered two different proofs of the DNF Theorem. They also described two different algorithms for putting sentences into DNF. But these algorithms do not

always leave you with the most elegant output. Indeed, just as we can 'clean up' the output of the algorithm for calculating interpolants (see §2.2), we can 'clean up' the output of our algorithms for calculating DNF-equivalents. Here are the clean-up rules; I leave it as an exercise to *justify* them. First, we rewrite our DNF sentence using the relaxed notational conventions of forall$\boldsymbol{x}$:Cambridge §10.3. Now we apply the following:

Rule 1: remove repetitious conjuncts. If any disjunct contains the same conjunct more than once, remove all but one instance of that conjunct (respecting bracketing conventions as appropriate).
Example: if the output is:

$$(A \wedge B) \vee (A \wedge \neg C) \vee D \vee (A \wedge B)$$

then applying Rule 1 we get:

$$(A \wedge B) \vee (A \wedge \neg C) \vee D$$

Rule 2: remove overly-specific disjuncts. Delete any disjunct which entails any other disjunct. (If two or more disjuncts entail each other, delete all but the first of those disjuncts.)
Example: the output of our example in §3.3 was:

$$(A \wedge B) \vee (A \wedge \neg C) \vee (\neg D \wedge D) \vee (\neg E \wedge D) \vee (\neg D \wedge E) \vee (\neg E \wedge E)$$

Noting that a contradiction entails *everything*, we should remove both contradictory disjuncts, obtaining (still with relaxed conventions):

$$(A \wedge B) \vee (A \wedge \neg C) \vee (\neg E \wedge D) \vee (\neg D \wedge E)$$

Rule 3: invoke excluded middle. Suppose there is a disjunct whose conjuncts are just $\boldsymbol{A}_1, ..., \boldsymbol{A}_m$ and $\boldsymbol{B}$ (in any order), and another disjunct whose conjuncts are just $\boldsymbol{A}_1, ..., \boldsymbol{A}_m$ and $\neg\boldsymbol{B}$ (in any order), so that the two disjuncts disagree only on whether whether $\boldsymbol{B}$ should be prefixed with a negation. In that case, delete both disjuncts and replace them with the simpler disjunct $(\boldsymbol{A}_1 \wedge ... \wedge \boldsymbol{A}_m)$. (However, if we have *just* $\boldsymbol{B}$ and $\neg\boldsymbol{B}$ as disjuncts, then delete the entire sentence and leave the tautology $(\boldsymbol{B} \vee \neg\boldsymbol{B})$.)
Example: the output of the example in §3.2 was:

$$(A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C)$$

which can be simplified to:

$$(A \wedge C) \vee (\neg A \wedge \neg B)$$

3.5 Conjunctive Normal Form

So far in this chapter, I have discussed *disjunctive* normal form. Given the duality of disjunction and conjunction (see §2.3), it may not come as a surprise to hear that there is also such a thing as *conjunctive normal form* (CNF).

The definition of CNF is exactly analogous to the definition of DNF. So, a sentence is in CNF *iff* it meets all of the following conditions:

(CNF1) No connectives occur in the sentence other than negations, conjunctions and disjunctions;
(CNF2) Every occurrence of negation has minimal scope;
(CNF3) No conjunction occurs within the scope of any disjunction.

Generally, then, a sentence in CNF looks like this

$$(\pm \boldsymbol{A}_1 \vee \ldots \vee \pm \boldsymbol{A}_i) \wedge (\pm \boldsymbol{A}_{i+1} \vee \ldots \vee \pm \boldsymbol{A}_j) \wedge \ldots \wedge (\pm \boldsymbol{A}_{m+1} \vee \ldots \vee \pm \boldsymbol{A}_n)$$

where each $\boldsymbol{A}_k$ is an atomic sentence.

Since '$\neg$' is its own dual, and '$\vee$' and '$\wedge$' are the duals of each other, it is immediate clear that if a sentence is in DNF, then its dual is in CNF; and *vice versa*. Armed with this insight, we can immediately prove another normal form theorem:

Theorem 3.4. Conjunctive Normal Form Theorem. For any sentence, there is a tautologically equivalent sentence in conjunctive normal form.

Proof. Let $\boldsymbol{S}$ be any sentence. Applying the DNF Theorem to $\underline{\boldsymbol{S}}$, there is a DNF sentence, $\boldsymbol{D}$, such that $\underline{\boldsymbol{S}} \dashv\vDash \boldsymbol{D}$. So, by Theorem 2.7, $\underline{\underline{\boldsymbol{S}}} \dashv\vDash \underline{\boldsymbol{D}}$. Whence, by Lemma 2.5, $\boldsymbol{S} \dashv\vDash \underline{\boldsymbol{D}}$. But, since $\boldsymbol{D}$ is in DNF, $\underline{\boldsymbol{D}}$ is in CNF. ■

This slick proof is a further illustration of the power of duality. However, it might suggest that the DNF Theorem enjoys some kind of 'precedence' over the CNF Theorem. That would be misleading. We can easily prove the CNF Theorem directly, using either of the two proof techniques that we used to prove the DNF Theorem (whereupon the DNF Theorem could be proved as a consequence of the CNF Theorem and duality). I shall sketch the main ideas of the direct proofs of the CNF Theorem; I leave it as an exercise to make these ideas more precise.

Proof sketch of CNF Theorem via truth tables. Given a TFL sentence, $\boldsymbol{S}$, we begin by writing down the complete truth table for $\boldsymbol{S}$.

If $\boldsymbol{S}$ is *true* on every line of the truth table, then $\boldsymbol{S} \dashv\vDash (\boldsymbol{A}_1 \vee \neg \boldsymbol{A}_1)$.

If $\boldsymbol{S}$ is *false* on at least one line of the truth table then, for every line on the truth table where $\boldsymbol{S}$ is false, write down a disjunction $(\pm \boldsymbol{A}_1 \vee \ldots \vee \pm \boldsymbol{A}_n)$ which is *false* on (and only on) that line. Let $\boldsymbol{C}$ be the conjunction of all of these disjuncts; by construction, $\boldsymbol{C}$ is in CNF and $\boldsymbol{S} \dashv\vDash \boldsymbol{C}$. ■

Proof sketch of CNF Theorem via substitution. Using Steps 0–2 of IntoDNF, we obtain a sentence satisfying (CNF1) and (CNF2). To turn this into a sentence in CNF, we simply modify Step 3, using substitutions based on the duals (what else!) of the Distribution Laws used in Step 3 of IntoDNF; i.e.:

$$((\boldsymbol{A} \wedge \boldsymbol{B}) \vee \boldsymbol{C}) \dashv\vDash ((\boldsymbol{A} \vee \boldsymbol{C}) \wedge (\boldsymbol{B} \vee \boldsymbol{C}))$$
$$(\boldsymbol{A} \vee (\boldsymbol{B} \wedge \boldsymbol{C})) \dashv\vDash ((\boldsymbol{A} \vee \boldsymbol{B}) \wedge (\boldsymbol{A} \vee \boldsymbol{C}))$$

This allows us to pull conjunctions outside the scope of any disjunction. ■

Of course, after obtaining a sentence in CNF – however we chose to do it – we can clean-up the resulting sentence using rules similar to those of §3.4. I leave it as an exercise, to determine what the appropriate rules should be.

Practice exercises

A. Consider the following sentences:

1. $(A \to \neg B)$
2. $\neg(A \leftrightarrow B)$
3. $(\neg A \vee \neg(A \wedge B))$
4. $(\neg(A \to B) \wedge (A \to C))$
5. $(\neg(A \vee B) \leftrightarrow ((\neg C \wedge \neg A) \to \neg B))$
6. $((\neg(A \wedge \neg B) \to C) \wedge \neg(A \wedge D))$

For each sentence:

- use both algorithms (by truth table, and by substitution) to write down sentences in DNF that are tautologically equivalent to these sentences.
- use both algorithms to write down sentences in CNF that are tautologically equivalent to these sentences.

B. Offer proofs of the two 'trivial inductions' mentioned in the proof of Theorem 3.1 via truth tables.

C. Explain why the proof of Theorem 3.2 required 'a trivial induction' (in the last sentence). Prove that induction.

D. Justify each of the rules mentioned in §3.4, i.e. prove that applying each of them preserves tautological equivalence.

E. Fill out the details of the two proof sketches of the CNF Theorem. Determine the appropriate 'clean up' rules for CNF sentences.

Expressive adequacy 4

In chapter 3, we established two normal form theorems. In this chapter, we shall use them to demonstrate the expressive power of TFL.

4.1 The expressive adequacy of TFL

In discussing duality (§2.3), I introduced the general idea of an *n*-place connective. We might, for example, define a three-place connective, '♡', into existence, by stipulating that it is to have the following characteristic truth table:

$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\heartsuit(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$
T	T	T	F
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	F

Probably this new connective would not correspond with any natural English expression (in the way that '∧' corresponds with 'and'). But a question arises: if we wanted to employ a connective with this characteristic truth table, must we add a *new* connective to TFL? Or can we get by with the connectives we *already have*?

Let us make this question more precise. Say that some connectives are JOINTLY EXPRESSIVELY ADEQUATE *iff*, for any possible truth function, there is a scheme containing only those connectives which expresses that truth function. Since we can represent truth functions using characteristic truth tables, we could equivalently say the following: some connectives are jointly expressively adequate *iff*, for any possible truth table, there is a scheme containing only those connectives with that truth table.

I say 'scheme' rather than 'sentence', because we are not concerned with something as specific as a sentence (it might help to reread forall***x***:Cambridge §7, and this book's §2.1). To see why, consider the characteristic truth table for conjunction; this schematically encodes the information that a conjunction $(\boldsymbol{A} \wedge \boldsymbol{B})$ is true iff both $\boldsymbol{A}$ and $\boldsymbol{B}$ are true (whatever $\boldsymbol{A}$ and $\boldsymbol{B}$ might be). When we discuss expressive adequacy, we are considering something at the same level of generality.

The general point is, when we are armed with some jointly expressively adequate connectives, no truth function lies beyond our grasp. And in fact, we are in luck.[1]

Theorem 4.1. Expressive Adequacy Theorem. The connectives of TFL are jointly expressively adequate. Indeed, the following pairs of connectives are jointly expressively adequate:

1. '$\neg$' and '$\vee$'
2. '$\neg$' and '$\wedge$'
3. '$\neg$' and '$\rightarrow$'

Proof. Given any truth table, we can use the method of proving the DNF Theorem (or the CNF Theorem) via truth tables, to write down a scheme which has the same truth table. For example, employing the truth table method for proving the DNF Theorem, I can tell you that the following scheme has the same characteristic truth table as $\heartsuit(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$, above:

$$(\boldsymbol{A} \wedge \boldsymbol{B} \wedge \neg \boldsymbol{C}) \vee (\boldsymbol{A} \wedge \neg \boldsymbol{B} \wedge \boldsymbol{C}) \vee (\neg \boldsymbol{A} \wedge \boldsymbol{B} \wedge \neg \boldsymbol{C})$$

It follows that the connectives of TFL are jointly expressively adequate. I now prove each of the subsidiary results.

Subsidiary Result 1: expressive adequacy of '$\neg$' and '$\vee$'. Observe that the scheme that we generate, using the truth table method of proving the DNF Theorem, will only contain the connectives '$\neg$', '$\wedge$' and '$\vee$'. So it suffices to show that there is an equivalent scheme which contains only '$\neg$' and '$\vee$'. To show do this, we simply consider the following equivalence:

$$(\boldsymbol{A} \wedge \boldsymbol{B}) \dashv\vDash \neg(\neg \boldsymbol{A} \vee \neg \boldsymbol{B})$$

and reason as in §3.2 (I leave the details as an exercise).

Subsidiary Result 2: expressive adequacy of '$\neg$' and '$\wedge$'. Exactly as in Subsidiary Result 1, making use of this equivalence instead:

$$(\boldsymbol{A} \vee \boldsymbol{B}) \dashv\vDash \neg(\neg \boldsymbol{A} \wedge \neg \boldsymbol{B})$$

Subsidiary Result 3: expressive adequacy of '$\neg$' and '$\rightarrow$'. Exactly as in Subsidiary Result 1, making use of these equivalences instead:

$$(\boldsymbol{A} \vee \boldsymbol{B}) \dashv\vDash (\neg \boldsymbol{A} \rightarrow \boldsymbol{B})$$
$$(\boldsymbol{A} \wedge \boldsymbol{B}) \dashv\vDash \neg(\boldsymbol{A} \rightarrow \neg \boldsymbol{B})$$

Alternatively, we could simply rely upon one of the other two subsidiary results, and (repeatedly) invoke only one of these two equivalences. ■

In short, there is never any *need* to add new connectives to TFL. Indeed, there is already some redundancy among the connectives we have: we could have made do with just two connectives, if we had been feeling really austere.

[1] This, and Theorem 4.5, follow from a more general result, due to Emil Post in 1941. For a modern presentation of Post's result, see Pelletier and Martin 'Post's Functional Completeness Theorem' (1990, *Notre Dame Journal of Formal Logic* 31.2).

4.2 Individually expressively adequate connectives

In fact, some two-place connectives are *individually* expressively adequate. These connectives are not standardly included in TFL, since they are rather cumbersome to use. But their existence shows that, if we had wanted to, we could have defined a truth-functional language that was expressively adequate, which contained only a single primitive connective.

The first such connective we shall consider is '↑', which has the following characteristic truth table.

A	B	$A \uparrow B$
T	T	F
T	F	T
F	T	T
F	F	T

This is often called 'the Sheffer stroke', after Harry Sheffer, who used it to show how to reduce the number of logical connectives in Russell and Whitehead's *Principia Mathematica*.[2] (In fact, Charles Sanders Peirce had anticipated Sheffer by about 30 years, but never published his results.)[3] It is quite common, as well, to call it 'nand', since its characteristic truth table is the negation of the truth table for '∧'.

Proposition 4.2. '↑' is expressively adequate all by itself.

Proof. Theorem 4.1 tells us that '¬' and '∨' are jointly expressively adequate. So it suffices to show that, given any scheme which contains only those two connectives, we can rewrite it as a tautologically equivalent scheme which contains only '↑'. As in the proof of the subsidiary cases of Theorem 4.1, then, we simply apply the following equivalences:

$$\neg A \dashv\vDash (A \uparrow A)$$
$$(A \vee B) \dashv\vDash ((A \uparrow A) \uparrow (B \uparrow B))$$

to Subsidiary Result 1 of Theorem 4.1. ∎

Similarly, we can consider the connective '↓':

A	B	$A \downarrow B$
T	T	F
T	F	F
F	T	F
F	F	T

This is sometimes called the 'Peirce arrow' (Peirce himself called it 'ampheck'). More often, though, it is called 'nor', since its characteristic truth table is the negation of '∨'.

[2] Sheffer, 'A Set of Five Independent Postulates for Boolean Algebras, with application to logical constants,' (1913, *Transactions of the American Mathematical Society* 14.4)

[3] See Peirce, 'A Boolian Algebra with One Constant', which dates to c.1880; and Peirce's *Collected Papers*, 4.264–5.

Proposition 4.3. '↓' is expressively adequate all by itself.

Proof. As in Proposition 4.2, although invoking the dual equivalences:

$$\neg \boldsymbol{A} \dashv\vDash (\boldsymbol{A} \downarrow \boldsymbol{A})$$
$$(\boldsymbol{A} \wedge \boldsymbol{B}) \dashv\vDash ((\boldsymbol{A} \downarrow \boldsymbol{A}) \downarrow (\boldsymbol{B} \downarrow \boldsymbol{B}))$$

and Subsidiary Result 2 of Theorem 4.1. ■

4.3 Failures of expressive adequacy

In fact, the *only* two-place connectives which are individually expressively adequate are '↑' and '↓'. But how would we show this? More generally, how can we show that some connectives are *not* jointly expressively adequate?

The obvious thing to do is to try to find some truth table which we *cannot* express, using just the given connectives. But there is a bit of an art to this. Moreover, in the end, we shall have to rely upon induction; for we shall need to show that *no* scheme – no matter how *long* – is capable of expressing the target truth table.

To make this concrete, let's consider the question of whether '∨' is expressively adequate all by itself. After a little reflection, it should be clear that it is not. In particular, it should be clear that any scheme which only contains disjunctions cannot have the same truth table as negation, i.e.:

$\boldsymbol{A}$	$\neg \boldsymbol{A}$
T	F
F	T

The intuitive reason, why this should be so, is simple: the top line of the desired truth table needs to have the value False; but the top line of any truth table for a scheme which *only* contains disjunctions will always be True. But so far, this is just hand-waving. To make it rigorous, we need to reach for induction. Here, then, is our rigorous proof.

Proposition 4.4. '∨' is not expressively adequate by itself.

Proof. Let $\boldsymbol{A}$ by any scheme containing no connective other than disjunctions. Suppose, for induction on length, that every shorter scheme containing only disjunctions is true whenever all its atomic constituents are true. There are two cases to consider:

> *Case 1:* $\boldsymbol{A}$ *is atomic.* Then there is nothing to prove.
> *Case 2:* $\boldsymbol{A}$ *is* $(\boldsymbol{B} \vee \boldsymbol{C})$*, for some schemes* $\boldsymbol{B}$ *and* $\boldsymbol{C}$ *containing only disjunctions.* Then, since $\boldsymbol{B}$ and $\boldsymbol{C}$ are both shorter than $\boldsymbol{A}$, by the induction hypothesis they are both true when all their atomic constituents are true. Now the atomic constituents of $\boldsymbol{A}$ are just the constituents of both $\boldsymbol{B}$ and $\boldsymbol{C}$, and $\boldsymbol{A}$ is true whenever $\boldsymbol{B}$ and $\boldsymbol{C}$. So $\boldsymbol{A}$ is true when all of its atomic constituents are true.

It now follows, by induction on length, that any scheme containing no connective other than disjunctions is true whenever all of its atomic constituents are true. Consequently, no scheme containing only disjunctions has the same truth table as that of negation. Hence '$\vee$' is not expressively adequate by itself. ■

In fact, we can generalise Proposition 4.4:

Theorem 4.5. The *only* two-place connectives that are expressively adequate by themselves are '$\uparrow$' and '$\downarrow$'.

Proof. There are sixteen distinct two-place connectives. We shall run through them all, considering whether or not they are individually expressively adequate, in four groups.

Group 1: the top line of the truth table is True. Consider those connectives where the top line of the truth table is True. There are eight of these, including '$\wedge$', '$\vee$', '$\rightarrow$' and '$\leftrightarrow$', but also the following:

A	B	$A \circ_1 B$	$A \circ_2 B$	$A \circ_3 B$	$A \circ_4 B$
T	T	T	T	T	T
T	F	T	T	T	F
F	T	T	F	F	T
F	F	T	T	F	F

(obviously the names for these connectives were chosen arbitrarily). But, exactly as in Proposition 4.4, none of these connectives can express the truth table for negation. So there is a connective whose truth table they cannot express. So none of them is individually expressively adequate.

Group 2: the bottom line of the truth table is False. Having eliminated eight connectives, eight remain. Of these, four are false on the bottom line of their truth table, namely:

A	B	$A \circ_5 B$	$A \circ_6 B$	$A \circ_7 B$	$A \circ_8 B$
T	T	F	F	F	F
T	F	T	T	F	F
F	T	T	F	T	F
F	F	F	F	F	F

As above, though, none of these connectives can express the truth table for negation. To show this we prove that any scheme whose only connective is one of these (perhaps several times) is false whenever all of its atomic constituents are false. We can show this by induction, exactly as in Proposition 4.4 (I leave the details as an exercise).

Group 3: connectives with redundant positions. Consider two of the remaining four connectives:

$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{A} \circ_9 \boldsymbol{B}$	$\boldsymbol{A} \circ_{10} \boldsymbol{B}$
T	T	F	F
T	F	F	T
F	T	T	F
F	F	T	T

These connectives have redundant positions, in the sense that the truth value of the overarching scheme only depends upon the truth value of one of the atomic constituents. More precisely:

$$\boldsymbol{A} \circ_9 \boldsymbol{B} \dashv\vDash \neg \boldsymbol{A}$$
$$\boldsymbol{A} \circ_{10} \boldsymbol{B} \dashv\vDash \neg \boldsymbol{B}$$

Consequently, there are many truth functions that they cannot express. In particular, they cannot express either the tautologous truth function (given by '$\circ_1$'), or the contradictory truth function (given by '$\circ_8$'). To show this, it suffices to prove that any scheme whose only connective is either '$\circ_9$' or '$\circ_{10}$' (perhaps several times) is contingent, i.e. it is true on at least one line and false on at least one other line. I leave the details of this proof as an exercise.

Group 4. Only two connectives now remain, namely '↑' and '↓', and Propositions 4.2 and 4.3 show that both are individually expressively adequate. ■

The above inductions on complexity were fairly routine. To close this chapter, I shall offer some similar results about expressive limitation that are only slightly more involved.

Proposition 4.6. Exactly four 2-place truth functions can be expressed using schemes whose only connectives are biconditionals.

Proof. I claim that the four 2-place truth functions that can be so expressed are expressed by:

$$\boldsymbol{A} \leftrightarrow \boldsymbol{A}$$
$$\boldsymbol{A}$$
$$\boldsymbol{B}$$
$$\boldsymbol{A} \leftrightarrow \boldsymbol{B}$$

It is clear that we can express all four, and that they are distinct.

To see that these are the *only* functions which can be expressed, I shall perform an induction on the length of our scheme. Let $\boldsymbol{S}$ be any scheme containing only (arbitrarily many) biconditionals and the atomic constituents $\boldsymbol{A}$ and $\boldsymbol{B}$ (arbitrarily many times). Suppose that any such shorter scheme is equivalent to one of the four schemes listed above. There are two cases to consider:

> *Case 1:* $\boldsymbol{S}$ *is atomic.* Then certainly it is equivalent to either $\boldsymbol{A}$ or to $\boldsymbol{B}$.
> *Case 2:* $\boldsymbol{S}$ *is* $(\boldsymbol{C} \leftrightarrow \boldsymbol{D})$*, for some schemes* $\boldsymbol{C}$ *and* $\boldsymbol{D}$. Since $\boldsymbol{C}$ and $\boldsymbol{D}$ are both shorter than $\boldsymbol{S}$, by supposition they are equivalent to one of the four mentioned schemes. And it is now easy to check, case-by-case, that, whichever

of the four schemes each is equivalent to, $(\boldsymbol{C} \leftrightarrow \boldsymbol{D})$ is also equivalent to one of those four schemes. (I leave the details as an exercise.)

This completes the proof of my claim, by induction on length of scheme. ■

Proposition 4.7. Exactly eight 2-place truth functions can be expressed using schemes whose only connectives are negations and biconditionals.

Proof. Observe the following equivalences:

$$\neg\neg\boldsymbol{A} \dashv\vDash \boldsymbol{A}$$
$$(\neg\boldsymbol{A} \leftrightarrow \boldsymbol{B}) \dashv\vDash \neg(\boldsymbol{A} \leftrightarrow \boldsymbol{B})$$
$$(\boldsymbol{A} \leftrightarrow \neg\boldsymbol{B}) \dashv\vDash \neg(\boldsymbol{A} \leftrightarrow \boldsymbol{B})$$

We can use these tautological equivalences along the lines of Lemma 3.2 to show the following: for any scheme containing only biconditionals and negations, there is a tautologically equivalent scheme where no negation is inside the scope of any biconditional. (The details of this are left as an exercise.)

So, given any scheme containing only the atomic constituents $\boldsymbol{A}$ and $\boldsymbol{B}$ (arbitrarily many times), and whose only connectives are biconditionals and negations, we can produce a tautologically equivalent scheme which contains at most *one* negation, and that at the start of the scheme. By Proposition 4.6, the subscheme following the negation (if there is a negation) must be equivalent to one of only four schemes. Hence the original scheme must be equivalent to one of only eight schemes. ■

Practice exercises

A. Where '$\circ_7$' has the characteristic truth table defined in the proof of Theorem 4.5, show that the following are jointly expressively adequate:

1. '$\circ_7$' and '$\neg$'.
2. '$\circ_7$' and '$\rightarrow$'.
3. '$\circ_7$' and '$\leftrightarrow$'.

B. Show that the connectives '$\circ_7$', '$\wedge$' and '$\vee$' are not jointly expressively adequate.

C. Complete the exercises left in each of the proofs of this chapter.

Soundness 5

So far, I have discussed only the syntax and semantics of TFL. In this chapter, and the next, I shall relate TFL's semantics to its *proof system* (as defined in forall*x*:Cambridge). I shall prove that the semantics and the formal proof system mirror are perfectly suited to one another, in a sense to be made more precise.

5.1 Soundness defined, and setting up the proof

Intuitively, a formal proof system is sound iff it does not allow you to prove any invalid arguments. This is obviously a highly desirable property. It tells us that our proof system will never lead us astray. Indeed, if our proof system were not sound, then we would not be able to trust our proofs. The aim of this chapter is to prove that our proof system is sound.

Let me make the idea more precise. A formal proof system is SOUND (relative to a given semantics) *iff*, if there is a formal proof of $\boldsymbol{C}$ from assumptions among Γ, then Γ genuinely entails $\boldsymbol{C}$ (given that semantics). Otherwise put, to prove that TFL's proof system is sound, I need to prove the following

Theorem 5.1. Soundness. For any sentences Γ and $\boldsymbol{C}$: if $\Gamma \vdash \boldsymbol{C}$, then $\Gamma \vDash \boldsymbol{C}$

To prove this, we shall check each of the rules of TFL's proof system individually. We want to show that no application of those rules ever leads us astray. Since a proof just involves repeated application of those rules, this will show that no proof ever leads us astray. Or at least, that is the general idea.

To begin with, we must make the idea of 'leading us astray' more precise. Say that a line of a proof is SHINY iff the assumptions on which that line depends tautologically entail the sentence on that line.[1] To illustrate the idea, consider the following:

1	$F \to (G \wedge H)$	
2	$\quad F$	
3	$\quad G \wedge H$	$\to$E 1, 2
4	$\quad G$	$\wedge$E 3
5	$F \to G$	$\to$I 2–4

[1] The word 'shiny' is not standard among logicians.

Line 1 is shiny iff $F \to (G \wedge H) \vDash F \to (G \wedge H)$. You should be easily convinced that line 1 is, indeed, shiny! Similarly, line 4 is shiny iff $F \to (G \wedge H), F \vDash G$. Again, it is easy to check that line 4 is shiny. As is every line in this TFL-proof. I want to show that this is no coincidence. That is, I want to prove:

Lemma 5.2. Every line of every TFL-proof is shiny.

Then we will know that we have never gone astray, on any line of a proof. Indeed, given Lemma 5.2, it will be easy to prove the Soundness Theorem:

Proof of Soundness Theorem, given Lemma 5.2. Suppose $\Gamma \vdash \boldsymbol{C}$. Then there is a TFL-proof, with $\boldsymbol{C}$ appearing on its last line, whose only undischarged assumptions are among Γ. Lemma 5.2 tells us that every line on every TFL-proof is shiny. So this last line is shiny, i.e. $\Gamma \vDash \boldsymbol{C}$. ■

It remains to prove Lemma 5.2.

To do this, we observe that every line of any TFL-proof is obtained by applying some rule. So what I want to show is that no application of a rule of TFL's proof system will lead us astray. More precisely, say that a rule of inference is RULE-SOUND *iff* for all TFL-proofs, if we obtain a line on a TFL-proof by applying that rule, and every earlier line in the TFL-proof is shiny, then our new line is also shiny. What I need to show is that *every* rule in TFL's proof system is rule-sound.

I shall do this in the next section. But having demonstrated the rule-soundness of every rule, Lemma 5.2 will follow immediately:

Proof of Lemma 5.2, given that every rule is rule-sound. Fix any line, line n, on any TFL-proof. The sentence written on line n must be obtained using a formal inference rule which is rule-sound. This is to say that, if every earlier line is shiny, then line n itself is shiny. Hence, by strong induction on the length of TFL-proofs, every line of every TFL-proof is shiny. ■

Note that this proof appeals to a principle of strong induction on the length of TFL-proofs. This is the first time we have seen that principle, and you should pause to confirm that it is, indeed, justified. (Hint: compare the justification for the Principle of Strong Induction on Length *of sentences* from §1.4.)

5.2 Checking each rule

It remains to show that every rule is rule-sound. This is actually much easier than many of the proofs in earlier chapters. But it is also time-consuming, since we need to check each rule individually, and TFL's proof system has plenty of rules! To speed up the process marginally, I shall introduce a convenient abbreviation: 'Δ_i' will abbreviate the assumptions (if any) on which line i depends in our TFL-proof (context will indicate which TFL-proof I have in mind).

Lemma 5.3. Introducing an assumption is rule-sound.

Proof. If $\boldsymbol{A}$ is introduced as an assumption on line n, then $\boldsymbol{A}$ is among Δ_n, and so $\Delta_n \vDash \boldsymbol{A}$. ■

Lemma 5.4. $\wedge$I is rule-sound.

Proof. Consider any application of $\wedge$I in any TFL-proof, i.e. something like:

i	$\boldsymbol{A}$	
j	$\boldsymbol{B}$	
n	$\boldsymbol{A} \wedge \boldsymbol{B}$	$\wedge$I i, j

To show that $\wedge$I is rule-sound, we assume that every line before line n is shiny; and we aim to show that line n is shiny, i.e. that $\Delta_n \vDash \boldsymbol{A} \wedge \boldsymbol{B}$.

So, let v be any valuation that makes all of Δ_n true.

I first show that v makes $\boldsymbol{A}$ true. To prove this, note that all of Δ_i are among Δ_n. By hypothesis, line i is shiny. So any valuation that makes all of Δ_i true makes $\boldsymbol{A}$ true. Since v makes all of Δ_i true, it makes $\boldsymbol{A}$ true too.

We can similarly see that v makes $\boldsymbol{B}$ true.

So v makes $\boldsymbol{A}$ true and v makes $\boldsymbol{B}$ true. Consequently, v makes $\boldsymbol{A} \wedge \boldsymbol{B}$ true. So any valuation that makes all of the sentences among Δ_n true also makes $\boldsymbol{A} \wedge \boldsymbol{B}$ true. That is: line n is shiny. ■

All of the remaining lemmas establishing rule-soundness will have, essentially, the same structure as this one did.

Lemma 5.5. $\wedge$E is rule-sound.

Proof. Assume that every line before line n on some TFL-proof is shiny, and that $\wedge$E is used on line n. So the situation is:

i	$\boldsymbol{A} \wedge \boldsymbol{B}$	
n	$\boldsymbol{A}$	$\wedge$E i

(or perhaps with $\boldsymbol{B}$ on line n instead; but similar reasoning will apply in that case). Let v be any valuation that makes all of Δ_n true. Note that all of Δ_i are among Δ_n. By hypothesis, line i is shiny. So any valuation that makes all of Δ_i true makes $\boldsymbol{A} \wedge \boldsymbol{B}$ true. So v makes $\boldsymbol{A} \wedge \boldsymbol{B}$ true, and hence makes $\boldsymbol{A}$ true. So $\Delta_n \vDash \boldsymbol{A}$. ■

Lemma 5.6. $\vee$I is rule-sound.

Proof. I leave this as an exercise. ■

Lemma 5.7. $\vee$E is rule-sound.

Proof. Assume that every line before line n on some TFL-proof is shiny, and that $\wedge$E is used on line n. So the situation is:

m	$\boldsymbol{A} \vee \boldsymbol{B}$	
i	$\quad \boldsymbol{A}$	
j	$\quad \boldsymbol{C}$	
k	$\quad \boldsymbol{B}$	
l	$\quad \boldsymbol{C}$	
n	$\boldsymbol{C}$	$\vee$E m, i–j, k–l

Let v be any valuation that makes all of Δ_n true. Note that all of Δ_m are among Δ_n. By hypothesis, line m is shiny. So any valuation that makes Δ_n true makes $\boldsymbol{A} \vee \boldsymbol{B}$ true. So in particular, v makes $\boldsymbol{A} \vee \boldsymbol{B}$ true, and hence either v makes $\boldsymbol{A}$ true, or v makes $\boldsymbol{B}$ true. We now reason through these two cases:

Case 1: v makes $\boldsymbol{A}$ true. All of Δ_i are among Δ_n, with the possible exception of $\boldsymbol{A}$. Since v makes all of Δ_n true, and also makes $\boldsymbol{A}$ true, v makes all of Δ_i true. Now, by assumption, line j is shiny; so $\Delta_j \vDash \boldsymbol{C}$. But the sentences Δ_i are just the sentences Δ_j, so $\Delta_i \vDash \boldsymbol{C}$. So, any valuation that makes all of Δ_i true makes $\boldsymbol{C}$ true. But v is just such a valuation. So v makes $\boldsymbol{C}$ true.

Case 2: v makes $\boldsymbol{B}$ true. Reasoning in exactly the same way, considering lines k and l, v makes $\boldsymbol{C}$ true.

Either way, v makes $\boldsymbol{C}$ true. So $\Delta_n \vDash \boldsymbol{C}$. ■

Lemma 5.8. ¬E is rule-sound.

Proof. Assume that every line before line n on some TFL-proof is shiny, and that ¬E is used on line n. So the situation is:

i	$\boldsymbol{A}$	
j	$\neg\boldsymbol{A}$	
n	$\bot$	¬E i, j

Note that all of Δ_i and all of Δ_j are among Δ_n. By hypothesis, lines i and j are shiny. So any valuation which makes all of Δ_n true would have to make both $\boldsymbol{A}$ and $\neg\boldsymbol{A}$ true. But no valuation can do that. So no valuation makes all of Δ_n true. So $\Delta_n \vDash \bot$, vacuously. ■

Lemma 5.9. X is rule-sound.

Proof. I leave this as an exercise. ■

Lemma 5.10. ¬I is rule-sound.

Proof. Assume that every line before line n on some TFL-proof is shiny, and that ¬I is used on line n. So the situation is:

i		$\boldsymbol{A}$	
j		$\bot$	
n	$\neg\boldsymbol{A}$		$\neg$I i–j

Let v be any valuation that makes all of Δ_n true. Note that all of Δ_n are among Δ_i, with the possible exception of $\boldsymbol{A}$ itself. By hypothesis, line j is shiny. But no valuation can make '$\bot$' true, so no valuation can make all of Δ_j true. Since the sentences Δ_i are just the sentences Δ_j, no valuation can make all of Δ_i true. Since v makes all of Δ_n true, it must therefore make $\boldsymbol{A}$ false, and so make $\neg\boldsymbol{A}$ true. So $\Delta_n \vDash \neg\boldsymbol{A}$. ■

Lemma 5.11. TND is rule-sound.

Proof. Assume that every line before line n on some TFL-proof is shiny, and that TND is used on line n. So the situation is:

i		$\boldsymbol{A}$	
j		$\boldsymbol{B}$	
k		$\neg\boldsymbol{A}$	
l		$\boldsymbol{B}$	
n	$\boldsymbol{B}$		TND i–j, k–l

Let v be any valuation that makes all of Δ_n true. Either v makes $\boldsymbol{A}$ true, or it makes $\boldsymbol{A}$ false. We shall reason through these (sub)cases separately.

> *Case 1: v makes $\boldsymbol{A}$ true.* The sentences Δ_i are just the sentences Δ_j. By hypothesis, line j is shiny. So $\Delta_i \vDash \boldsymbol{B}$. Furthermore, all of Δ_i are among Δ_n, with the possible exception of $\boldsymbol{A}$. So v makes all of Δ_i true, and hence makes $\boldsymbol{B}$ true.
>
> *Case 2: v makes $\boldsymbol{A}$ false.* Then v makes $\neg\boldsymbol{A}$ true. Reasoning in exactly the same way as above, considering lines k and l, v makes $\boldsymbol{B}$ true.

Either way, v makes $\boldsymbol{B}$ true. So $\Delta_n \vDash \boldsymbol{B}$. ■

Lemma 5.12. →I, →E, ↔I, and ↔E are all rule-sound.

Proof. I leave these as exercises. ■

This establishes that all the basic rules of our proof system are rule-sound. Finally, we show:

Lemma 5.13. All of the derived rules of our proof system are rule-sound.

Proof. Suppose that we used a derived rule to obtain some sentence, $\boldsymbol{A}$, on line n of some TFL-proof, and that every earlier line is shiny. Every use of a derived rule can be replaced (at the cost of long-windedness) with multiple uses of basic rules.

(This was proved in forall***x***:Cambridge §30.) That is to say, we could have used basic rules to write $\boldsymbol{A}$ on some line $n + k$, without introducing any further assumptions. So, applying Lemmas 5.3–5.12 several times ($k + 1$ times, in fact), we can see that line $n + k$ is shiny. Hence the derived rule is rule-sound. ■

And that's that! We have shown that every rule – basic or otherwise – is rule-sound, which is all that we required to establish Lemma 5.2, and hence the Soundness Theorem (see §5.1).

But it might help to round off this chapter if I repeat my informal explanation of what we have done. A formal proof is just a sequence – of arbitrary length – of applications of rules. We have spent this section showing that any application of any rule will not lead you astray. It follows (by induction) that no formal proof will lead you astray. That is: our proof system is sound.

Practice exercises

A. Complete the Lemmas left as exercises in this chapter. That is, show that the following are rule-sound:

- ∨I. (*Hint*: this is similar to the case of ∧E.)
- X. (*Hint*: this is similar to the case of ¬E.)
- →I. (*Hint*: this is similar to ∨E.)
- →E.
- ↔I.
- ↔E.

Completeness 6

Our formal proof system is *sound*: it will never lead us astray. In this chapter, I shall show it is *complete*: every entailment is captured by some formal proof.

6.1 Completeness defined, and setting up the proof

We say that a formal proof system is COMPLETE (relative to a given semantics) *iff* whenever Γ entails $\boldsymbol{C}$, there is some formal proof of $\boldsymbol{C}$ whose assumptions are among Γ. To prove that TFL's proof system is complete, I need to show:

Theorem 6.1. Completeness. For any sentences Γ and $\boldsymbol{C}$: if $\Gamma \models \boldsymbol{C}$, then $\Gamma \vdash \boldsymbol{C}$.

This is the converse of the Soundness Theorem. Together, these two theorems vindicate our proof system entirely: it doesn't prove too much (it is sound); it doesn't prove too little (it is complete); it is just right.

As with the Soundness Theorem, the proof of the Completeness Theorem turns on a single important lemma. Here is that Lemma:

Lemma 6.2. For any sentences Δ: if $\Delta \nvdash \bot$, then there is a valuation that makes all of Δ true.

I shall explain how we might prove Lemma 6.2 in due course. But, once we have proved Lemma 6.2, the Completeness Theorem will follow almost immediately:

Proof of the Completeness Theorem, given Lemma 6.2. We shall prove the contrapositive, i.e. if $\Gamma \nvdash \boldsymbol{C}$, then $\Gamma \not\models \boldsymbol{C}$.

So, suppose that $\Gamma \nvdash \boldsymbol{C}$. Now, if there were a proof of '$\bot$' from Γ with $\neg\boldsymbol{C}$, we could manipulate this proof (using $\neg$I and DNE) to obtain a proof of $\boldsymbol{C}$ from Γ, contradicting our supposition. So it must be that:

$$\Gamma, \neg\boldsymbol{C} \nvdash \bot$$

Now, by Lemma 6.2, there is some valuation that makes all of Γ and $\neg\boldsymbol{C}$ true. Hence $\Gamma \not\models \boldsymbol{C}$. ■

All the hard work, then, will go into proving Lemma 6.2. And – fair warning – that *will* be hard work. In fact, I shall prove Lemma 6.2 in two different ways. One of these ways is more concrete, and gives us quite a lot of information; the other is more compressed and abstract, but also a bit more general.

6.2 An algorithmic approach

I start with the more concrete approach to Lemma 6.2. This works by specifying an algorithm, SimpleSearch, which determines, for any arbitrary sentences, whether or not those sentences are jointly contrary. (Iam assuming that we are given only *finitely many* sentences. I shall revisit this point in §6.4, but shall not mention it much until then.) Moreover, SimpleSearch two very desirable properties. For any sentences, Δ, that we feed in:

(A1) if Δ are jointly contrary, then the algorithm outputs a TFL-proof which starts with Δ as assumptions and terminates in '$\bot$' on just those assumptions; and
(A2) if Δ are not jointly contrary, then the algorithm yields a valuation which makes all of Δ true together.

Clearly, any algorithm with both of these properties is independently interesting. But the important thing, for our purposes, is that if we can prove that there is an algorithm with property (A2), then we will have proved Lemma 6.2 (pause for a moment to check this!) and hence the Completeness Theorem. So, my aim is to specify an algorithm, and then prove that it has these properties.

First steps towards an algorithm

To say that $\Delta \vdash \bot$ is to say that there is a TFL-proof, whose assumptions are all among Δ, and which terminates in '$\bot$' without any additional assumptions.

It obvious that, if '$\bot$' appears anywhere in a TFL-proof, then the TFL-proof involves an application of the rule $\neg$E. So our algorithm will have to look for situations in which we might be able to apply the rule $\neg$E.

Of course, to show that some sentences are jointly contrary, it is not enough simply to find a situation in which we can apply $\neg$E. For if we only obtain '$\bot$' within the scope of *additional* assumptions, we will not have shown that our *original* sentences are jointly contrary. Fortunately, there are rules that allow us to take instances of '$\bot$' that appear in the scope of additional assumptions, and derive an instance of '$\bot$' that appears within the scope of fewer assumptions, e.g.:

1	$A \vee B$	
2	$(\neg A \wedge \neg B)$	
3	$\neg A$	$\wedge$E 2
4	$\neg B$	$\wedge$E 2
5	$\quad A$	
6	$\quad \bot$	$\neg$E 5, 3
7	$\quad B$	
8	$\quad \bot$	$\neg$E 7, 4
9	$\bot$	$\vee$E 1, 5–6, 7–8

In fact, this TFL-proof is exactly the sort of proof that our algorithm will look for. In particular, the core of our algorithm involves only the techniques used in this proof: ∧E, introducing assumptions, using ¬E within the scope of those assumptions, and then using ∨E to manipulate the instances of '⊥'.

Illustrations of **SimpleSearch**, assuming CNF

In order to keep things simple, at this stage *I shall assume that the sentences that we want to test for joint contrariness are all in CNF*. (For a reminder of what this means, see §3.5.) I shall explain why this assumption is harmless towards the end of this section.

Given that we are testing some CNF sentences for joint contrariness, our first task is of course to start a TFL-proof with all of these sentences as assumptions.

Our second task is equally plain. Since the sentences are in CNF, we should eliminate all of the conjunctions that occur in them. More precisely: we should apply ∧E repeatedly, until no further application of ∧E would yield a sentence that we have not already obtained.

Given that our original sentences were in CNF, the output of all of these applications of ∧E will be a series of sentences containing only negations and disjunctions, where the scope of every occurrence of negation is minimal. Otherwise put: we will now be dealing with a bunch of sentences that are either atomic sentences, negations of atomic sentences, or disjunctions. Here is where things get interesting, and it will help us to have an example. So, suppose we start with the following sentences:

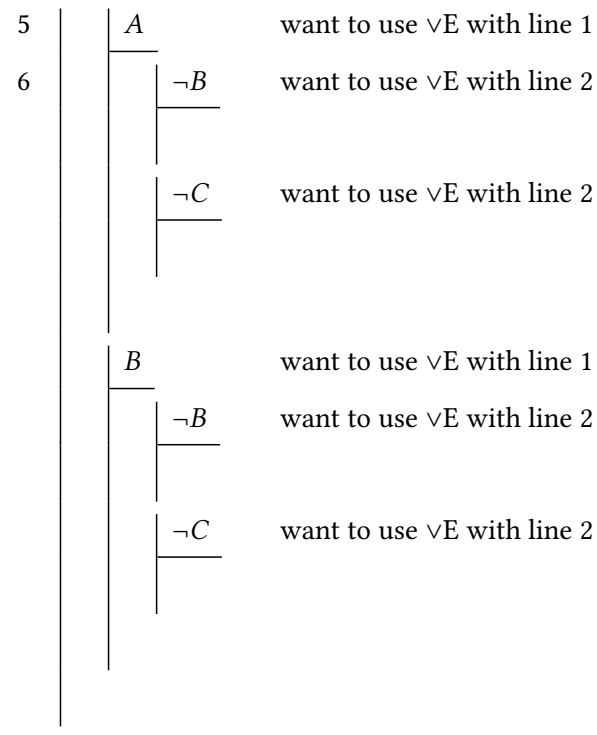
1	$(A \vee B)$
2	$(\neg B \vee \neg C)$
3	$\neg A$
4	C

There are no conjunctions to deal with, so our next task is to position ourselves so that we might be able to use ∨E further down the line. In particular, we must try to position ourselves to use ∨E with a citation that mentions line 1, and then two subproofs, one starting with the assumption 'A', the other starting with the assumption that 'B'. So we now continue the TFL-proof as follows (I have added some annotations to explain our motivations):

5		A	want to use ∨E with line 1
		B	want to use ∨E with line 1

When we set ourselves up for potential uses of ∨E in this way, we shall henceforth say that the disjunction in question (here, the disjunction on line 1) has been *expanded*. Note that, when we expand a disjunction by introducing new assumptions, we must leave gaps underneath these new assumptions. We will fill in these gaps as we continue to implement our algorithm. But, since we do not yet know how these gaps will be filled in, we cannot yet assign definite numbers to all the lines of our proof. So what we have written down cannot any longer be described as a TFL-proof; rather, it is a PROOF-SKELETON. This is how I shall refer to it, in what follows.

We have expanded our first disjunction, thereby positioning ourselves for a (potential) later use of ∨E with line 1. But there is another disjunction on line 2. We need to expand this disjunction too, positioning ourselves for a (potential) later use of ∨E with line 2. At this point, however, some care is needed. We have just made two new assumptions – 'A', and 'B', separately – and we might be able to use ∨E with line 2 inside the scope of *either* assumption. So we must continue our proof-skeleton as follows:



At this point, all the disjunctions in our proof-skeleton have been expanded. So we can move on to the next task.

Recall that we started with sentences in CNF. We first deal with all the conjunctions (by ∧E). We then unpacked all the disjunctions. So our proof-skeleton will now have lots of lines on it that contain simply an atomic sentence, or its negation. So the next task is to apply the rule ¬E wherever possible. In our example, we can apply it four times:

5	A	want to use ∨E with line 1
6	$\quad \neg B$	want to use ∨E with line 2
7	$\quad \bot$	¬E 5, 3
8	$\quad \neg C$	want to use ∨E with line 2
9	$\quad \bot$	¬E 5, 3
i	B	want to use ∨E with line 1
$i+1$	$\quad \neg B$	want to use ∨E with line 2
$i+2$	$\quad \bot$	¬E i, $i+1$
$i+3$	$\quad \neg C$	want to use ∨E with line 2
$i+4$	$\quad \bot$	¬E 4, $i+3$

Recall that our aim is to end up with a proof of '$\bot$' on no assumptions. So the most we can hope for, from each assumption, is that it generates '$\bot$'. When an assumption does generate '$\bot$', we will have no further use for it, except in a potential application of ∨E. So, at this point, we can close some of the gaps in our proof-skeleton, and put some flesh on its bones by adding in line numbers (and *dummy* line numbers, for ease of reference); I have done this, above. We now apply ∨E as often as we can:

5	A	want to use ∨E with line 1
6	$\quad \neg B$	want to use ∨E with line 2
7	$\quad \bot$	¬E 5, 3
8	$\quad \neg C$	want to use ∨E with line 2
9	$\quad \bot$	¬E 5, 3
10	$\bot$	∨E 2, 6–7, 8–9
11	B	want to use ∨E with line 1
12	$\quad \neg B$	want to use ∨E with line 2
13	$\quad \bot$	¬E 10, 11
14	$\quad \neg C$	want to use ∨E with line 2
15	$\quad \bot$	¬E 4, 13
16	$\bot$	∨E 2, 12–13, 14–15

We have managed to apply it twice here, citing line 2 on both occasions. Again, we can close some gaps and fill in some line numbers (removing the dummy line numbers where appropriate). And now we are able to apply ∨E again:

17	$\bot$	∨E 1, 5–10, 11–16

And now we have generated a complete TFL-proof of '$\bot$' from our initial assumptions. That is, *our initial assumptions were jointly contrary*.

Now, the preceding TFL-proof is not as slick as it could have been (in particular, steps 5–10 took longer than they needed to). But the important point is that we did, indeed, end up with a proof of '$\bot$' from our initial assumptions, without really having to *think*.

This is our first run-through of our algorithm, SimpleSearch. To get a better – but still informal – idea of how SimpleSearch works, I shall now consider an application of SimpleSearch to some sentences that are *not* jointly contrary. Suppose we start with the following sentences:

1	$(A \vee (\neg B \vee C))$
2	$(\neg A \vee \neg C)$
3	B

There are no conjunctions to deal with, so we immediately move to the task of expanding the disjunctions on line 1 and line 2. This yields:

4	A	want to use ∨E with line 1
5	$\neg A$	want to use ∨E with line 2
	$\neg C$	want to use ∨E with line 2
i	$(\neg B \vee C)$	want to use ∨E with line 1
	$\neg A$	want to use ∨E with line 2
	$\neg C$	want to use ∨E with line 2

Again, I have inserted a dummy line-number for ease of reference. And this line merits comment since, in expanding the disjunction on line 1, we have generated a *new* disjunction on line *i*. So, even though we have expanded all the disjunctions we *began* with, we still have an unexpanded disjunction. This, too, must be expanded. But at this point, we must take particular care. Our aim, in expanding a disjunction, is to set ourselves up for all possible future uses of ∨E. However, we plainly would not be able to use ∨E with *i* at any line *earlier* than line *i*. Consequently, we should expand that disjunction by continuing our proof-skeleton as follows:

4	A	want to use ∨E with line 1
5	$\neg A$	want to use ∨E with line 2
	$\neg C$	want to use ∨E with line 2
i	$(\neg B \vee C)$	want to use ∨E with line 1
	$\neg A$	want to use ∨E with line 2
	$\neg B$	want to use ∨E with line *i*
	C	want to use ∨E with line *i*
	$\neg C$	want to use ∨E with line 2
	$\neg B$	want to use ∨E with line *i*
	C	want to use ∨E with line *i*

And, at this point, we have expanded *all* the disjunctions that occur on our proof. So we move on to the next stage of our algorithm: we apply ¬E wherever we can; and then we apply ∨E wherever we can. The final result (adding in a few more dummy line-numbers) is:

1	$(A \vee (\neg B \vee C))$	
2	$(\neg A \vee \neg C)$	
3	B	
4	$\quad A$	want to use ∨E with line 1
5	$\quad \neg A$	want to use ∨E with line 2
6	$\quad \bot$	¬E 4, 5
7	$\quad \neg C$	want to use ∨E with line 2
i	$\quad (\neg B \vee C)$	want to use ∨E with line 1
$i+1$	$\quad \neg A$	want to use ∨E with line 2
$i+2$	$\quad \neg B$	want to use ∨E with line i
$i+3$	$\quad \bot$	¬E 3, $i+2$
$i+4$	$\quad C$	want to use ∨E with line i
k	$\quad \neg C$	want to use ∨E with line 2
$k+1$	$\quad \neg B$	want to use ∨E with line i
$k+2$	$\quad \bot$	¬E 3, $k+1$
$k+3$	$\quad C$	want to use ∨E with line i
$k+4$	$\quad \bot$	¬E $k+3$, k
$k+5$	$\quad \bot$	∨E 2, $k+1$–$k+2$, $k+3$–$k+4$

At this point, SimpleSearch has run its course. But the output is not a TFL-proof of '$\bot$' from the initial assumptions: it is a proof-skeleton with several gaps in it. I claim that I can infer from this that the initial assumptions are *not* jointly contrary. I further claim that there is a valuation that makes all of the initial assumptions true. Moreover, I claim that I can read off this valuation directly from the proof-skeleton!

I shall *justify* all of these claims in a moment. First, though, let me explain how to read off the valuation from the proof-skeleton. Let us say that a line in a proof-

skeleton is OPEN iff there is gap underneath it.[1] In my example, lines 7 and $i + 4$ are both open. I choose one of these open lines to work with; it might as well be the first, i.e. line 7. I then make a list of all the atomic sentences and negations of atomic sentences on lines that either occur on the open line or that occur on earlier lines (but not in a subproof that has been closed before the open line). So, working with line 7 of my example, my list is:[2]

$$\neg C, A, B$$

There is, indeed, a unique valuation which makes all of these sentences true. And this valuation will make all of my original assumptions true too. In the case of my example, this is easily verified.

In a moment, I shall prove that this was not just good fortune, but that SimpleSearch is guaranteed to behave in this way.

Relaxing the assumption of CNF

First, I need to explain why it was harmless to restrict our attention to sentences in CNF. The simple reason for this is that, given any sentences, Δ, we can always find some equivalent sentences, Δ^*, in CNF.

However, we need to be a little careful here. Given any sentences Δ, the CNF Theorem of §3.5 tells us that there are *tautologically* equivalent sentences, Δ^*, in CNF. But, given our aims, we cannot afford to rely upon this fact. After all, if TFL's proof-system cannot itself *prove* the equivalences between Δ and Δ^*, then manipulating Δ^* with TFL's proof-system will teach us *nothing* about the proof-theoretic consequences of Δ. And what we want is an algorithm which decides whether or not Δ are jointly contrary.

The upshot of this is that we need to prove a *proof-theoretic* analogue of the CNF Theorem.

Lemma 6.3. For any sentence, there is a *provably* equivalent sentence in CNF. Moreover, there is an algorithm for determining these equivalents.

Proof sketch. First, prove a proof-theoretic analogue of Lemma 2.2. That is, show the following: If $\boldsymbol{C} \dashv\vdash \boldsymbol{S}$, then $\boldsymbol{A} \dashv\vdash \boldsymbol{A}[\boldsymbol{S}/\boldsymbol{c}]$.

Then offer proof-theoretic analogues of Lemmas Lemmas 3.2–3.3, noting that each of the tautological equivalences appealed to (namely, Double Negation Elimination, the De Morgan Laws, and the Distribution Laws) are also *provable* equivalences. (Indeed, all but the Distribution Laws are *derived* rules of TFL's proof-system.) ■

[1] More precisely a line is open *iff* it meets all of the following four conditions. (i) The line is not '⊥'. (ii) No instance of ∧E occurs after the line. (iii) The next line in the proof-skeleton (if there is one) is not '⊥'. (iv) the next line in the proof-skeleton (if there is one) does not depend upon all of the assumptions of the former line.

[2] If I had worked instead with line $i + 4$, my list would have been: $C, \neg A, B$.

So, given any sentences, we can *first* put them into CNF, and then apply the steps that were sketched just earlier: apply ∧E, expand disjunctions, apply ¬E, and apply ∨E. Let's make this precise.

Statement of SimpleSearch

Given any sentences Δ:

Step 1. Start a TFL-proof, with Δ as your assumptions.

Step 2. For each sentence $\boldsymbol{D}$ among Δ, apply the algorithm which finds a CNF-sentence $\boldsymbol{D}^*$ such that $\boldsymbol{D}^* \dashv\vdash \boldsymbol{D}$. This will leave you with a single TFL-proof where each $\boldsymbol{D}^*$ appears on some line.

Step 3: Now apply ∧E as often as you can to lines of your TFL-proof that are in CNF, until any further application of ∧E would yield a sentence already on your proof.

Step 4: Expand the first disjunction in CNF which you have not yet expanded. More precisely: let $(\boldsymbol{A} \vee \boldsymbol{B})$ be the first disjunction in CNF that you have not yet expanded and let l be any line in your proof-skeleton that is both beneath the line on which the disjunction occurs; and which depends upon (at least) the assumptions on which the disjunction occurs. Then, beneath line l, introduce two new assumptions – $\boldsymbol{A}$ and $\boldsymbol{B}$ – separately, and with an empty line beneath both. At this point, we say that the disjunction '$(\boldsymbol{A} \vee \boldsymbol{B})$' has been *expanded*. Repeat this step until every disjunction in CNF has been expanded.

Step 5: Go through every *open* line and check to see whether ¬E can be applied. If so, apply it (filling in line numbers).

Step 6: Go through every *open* line and check to see whether ∨E can be applied (allowing you to generate a line with '⊥' on it). If so, apply it (filling in line numbers). Repeat this step until ∨E cannot be applied any further.

Step 7: Output the proof or proof-skeleton. This terminates SimpleSearch.

This is the algorithm SimpleSearch.

Proving that SimpleSearch behaves as required

I now want to prove that SimpleSearch has the properties (A1) and (A2), which I advertised for it.

The very first thing I need to do is to make sure that, when we run SimpleSearch, there is no chance of our getting trapped in an infinite loop. That is, we need a guarantee that SimpleSearch will output something *in the end*, regardless of the inputs we feed it.

Lemma 6.4. SimpleSearch is guaranteed to terminate, on any inputs.

Proof sketch. Step 1 is guaranteed to terminate eventually, since it just involves writing down finitely many sentences.

Step 2 is similarly guaranteed to terminate, since it only ever takes finitely many steps to generate a CNF equivalent from each of the finitely many assumptions.

Step 3 is guaranteed to terminate, since are only finitely many conjunction signs among the CNF sentences you have written down, and so you only need to apply ∧E finitely many times.

Step 4 is the tricky step. The worry is that, in expanding a disjunction (in CNF), you may have to introduce *many* new assumptions; and if one of the disjuncts is itself a disjunction, this can increase the total number of unexpanded disjunctions.[3] However, when we expand a disjunction, each of the disjuncts that we introduce as an assumption is *shorter* than the newly-expanded disjunction. So, even though expanding a disjunction may increase the number of unexpanded disjunctions, the unexpanded disjunctions keep getting shorter. Eventually, then, these will dwindle into atomic sentences or negations of atomic sentences, so that every disjunction has been expanded.[4]

Step 5 is guaranteed to terminate, since as Step 4 terminated, there will only be finitely many open lines, and inspecting whether or not ¬E can be applied on a given line is a bounded procedure (you only need to inspect all the earlier lines). Step 6 is similarly guaranteed to terminate. ■

So we now know that, given any inputs, SimpleSearch *will* yield an output. We next need to confirm that we can *trust* that output.

Lemma 6.5. If SimpleSearch outputs a proof or proof-skeleton whose last line is '$\bot$' on no assumptions other than Δ, then $\Delta \vdash \bot$

Proof sketch. By inspection, it is clear that SimpleSearch simply applies the rules of the TFL-proof system (in a very specific order) to a proof that starts with Δ. So if the proof-skeleton's last line is '$\bot$' on no assumptions other than Δ, then SimpleSearch has simply outputted a TFL-proof that $\Delta \vdash \bot$. ■

Lemma 6.6. If SimpleSearch outputs a proof or proof-skeleton whose last line is *not* '$\bot$' on no assumptions other than Δ, then:

1. there is at least one open line on the proof-skeleton; and
2. every open line yields (in the manner described) a valuation which makes all of Δ true; and
3. $\Delta \nvdash \bot$.

Proof. I shall prove each point separately, on the assumption that SimpleSearch terminates by outputting a proof-skeleton whose last line is *not* '$\bot$' on no assumptions other than Δ.

Proof of (1). There are two cases to consider:

> *Case 1: None of the CNF-sentences that we generated in Step 2 of SimpleSearch contained any occurrences of '$\vee$'.* Then, since the last line of our proof is not '$\bot$', the last line of our proof-skeleton is open.

[3] To see this in action, see what happens when your initial assumptions are '$(A \vee B)$' and '$((C \vee D) \vee (D \vee E))$'.

[4] One way to make this rigorous would be to prove, by induction, that if Δ contains exactly n instances of '$\vee$', Step 3 will require you to introduce at most $2^{n+1} - 2$ assumptions.

Case 2: At least one of the CNF-sentences that we generated in Step 2 of SimpleSearch contained an occurrence of '∨'. Then, if every assumption that we made when expanding disjunctions led to '⊥', we would have applied ∨E repeatedly until we obtained '⊥' on no assumptions. So at least one assumption did not lead to '⊥', and so there is an open line.

Proof of (2). Let Ω be the sentences that we read off (in the manner described earlier) from an open line on our proof (whose existence is guaranteed by point (1)). They are all atomic sentences and their negations. Now, since the line in question is open, this means that we cannot find both a sentence and its negation among Ω, since otherwise we could have applied ¬E. Hence there is a valuation, v, which makes all of them true.[5]

Now let $\boldsymbol{D}$ be any sentence among Δ; I claim that v makes $\boldsymbol{D}$ true. To prove this claim, I first consider the CNF sentence $\boldsymbol{D}^*$ generated in Step 1 of SimpleSearch, such that $\boldsymbol{D} \dashv\vdash \boldsymbol{D}^*$. It is easy to show that, for every (conjoined) disjunction in $\boldsymbol{D}^*$, at least one disjunct of that disjunction appears in Ω; it follows that each of the disjuncts is made true by v, so that $\boldsymbol{D}^*$ is made true by v (I leave the details of this as an exercise). Now, since $\boldsymbol{D}^* \vdash \boldsymbol{D}$, the Soundness Theorem tells us that $\boldsymbol{D}^* \models \boldsymbol{D}$; so v must make $\boldsymbol{D}$ true too.

Proof of (3). Suppose, for reductio, that $\Delta \vdash \bot$. Then, by the Soundness Theorem, there is no valuation that makes all of the Δ true. This contradicts what we have just shown in point (2). ■

And now – at long last – since we can trust the outputs of SimpleSearch, we can show that SimpleSearch has properties (A1) and (A2):

Theorem 6.7. Given any sentences Δ:

1. If $\Delta \vdash \bot$, then SimpleSearch yields a TFL-proof which starts with Δ as assumptions and terminates in '⊥'.
2. If $\Delta \nvdash \bot$, then SimpleSearch shows us as much and yields a valuation which makes all of Δ true.

Proof. Suppose $\Delta \vdash \bot$. Then by Lemma 6.6, SimpleSearch does *not* output a proof or proof-skeleton whose last line is not '⊥' on no assumptions other than Δ. But by Lemma 6.4, SimpleSearch does terminate. So the last line of the output must be '⊥', on no assumptions other than Δ, which, by Lemma 6.5, is an explicit TFL-proof that $\Delta \vdash \bot$.

Suppose $\Delta \nvdash \bot$. Then by Lemma 6.5, SimpleSearch does not output a proof-skeleton whose last line is '⊥' on no assumptions other than Δ. Now apply Lemma 6.6. ■

And with this result, we have achieved what we set out to achieve! But it might be worth reminding ourselves *why* we set out on this course. Our target for this chapter was to prove the Completeness Theorem. To prove that, we needed to prove Lemma 6.2. But Theorem 6.7 immediately entails Lemma 6.2 (just look at the

[5] For details of this last step, see Lemma 6.8, below.

second condition). So we have proved the Completeness Theorem, and acquired a useful little algorithm along the way!

6.3 A more abstract approach: polarising sentences

We could stop now. But I want to offer a second proof of the Completeness Theorem. It is more abstract, more general, and in a sense, more direct.

From the perspective of proving the Completeness Theorem, it is massive overkill to deploy SimpleSearch. For our purposes, all that matters is the following: when $\Delta \nvdash \bot$, there are some atomic sentences and negations of atomic sentences, Ω, from which we can read off a valuation that makes all of Δ true. From an abstract point of view, we needn't care how SimpleSearch *generates* Ω; only that Ω exists. The second proof of the Completeness Theorem builds upon this observation.

It will make our lives easier if we introduce a bit more terminology. Where Ω are some sentences, say that Ω are JOINTLY POLARISING *iff* both:[6]

(P1) every sentence among Ω is either an atomic sentence or a negation of an atomic sentence; and
(P2) if a sentence is among Ω, then its negation is not among Ω

Where Ω are jointly polarising, say that $\boldsymbol{A}$ is IN THE FIELD OF Ω iff $\boldsymbol{A}$ is an atomic sentence such that either $\boldsymbol{A}$ or $\neg\boldsymbol{A}$ is among Ω.

It is worth pausing for a moment to think about how this terminology, and this result, connects with SimpleSearch. When $\Delta \nvdash \bot$, SimpleSearch generates a proof-skeleton with at least one open line. From each open line, we can read off some jointly polarising sentences, Ω. Moreover, when $\boldsymbol{D}$ is any sentence among Δ, every atomic subsentence of $\boldsymbol{D}$ is in the field of Ω. At this point, we claimed (in the proof of Lemma 6.6) that there is bound to be some valuation that makes all of Ω true. That was right, and the more general result is simply this very easy observation.

Lemma 6.8. Suppose Ω are jointly polarising. Then there is exactly one valuation of the sentences in the field of Ω that makes every sentence among Ω true.

Proof. Define a valuation, v, as follows: for each sentence $\boldsymbol{A}$ in the field of Ω,

- v makes $\boldsymbol{A}$ true if $\boldsymbol{A}$ is among Ω
- v makes $\boldsymbol{A}$ false if $\neg\boldsymbol{A}$ is among Ω

Given (P2), v is well-defined. Now, if $\boldsymbol{B}$ is a sentence among Ω, then $\boldsymbol{B}$ is either $\boldsymbol{A}$ or $\neg\boldsymbol{A}$, for some sentence $\boldsymbol{A}$ in the field of Ω. Either way, v makes $\boldsymbol{B}$ true. Moreover, any valuation that assigned a different value to $\boldsymbol{A}$ would make $\boldsymbol{B}$ false. Hence v is unique. ■

Continuing, though, to think through how SimpleSearch relates to this terminology: the next stage in the proof of Lemma 6.6 was for us to show that the (unique) valuation that makes all of Ω true also makes all of Δ true too. This required a bit

[6] NB: this terminology is not standard among logicians.

of work (some of which I left as an exercise). But let me now part company with the proof of Lemma 6.6, and instead make a much more simple observation.

We are considering a situation where all of the atomic subsentences of some sentence (in this case, $\boldsymbol{D}$) are in the field of some jointly polarising sentences (in this case, Ω). We can obtain a very easy and general result about such situations.

Lemma 6.9. Suppose Ω are jointly polarising and that every atomic subsentence of $\boldsymbol{A}$ is in the field of Ω. Then either $\Omega \vDash \boldsymbol{A}$ or $\Omega \vDash \neg\boldsymbol{A}$.

Proof. Using Lemma 6.8, let v be the unique valuation of the sentences in the field of Ω that makes all of Ω true . Either v makes $\boldsymbol{A}$ true, or v makes $\boldsymbol{A}$ false. If v makes $\boldsymbol{A}$ true, then since v is the only valuation that makes all of Ω true, $\Omega \vDash \boldsymbol{A}$. If v makes $\boldsymbol{A}$ false, then it makes $\neg\boldsymbol{A}$ true, and similar reasoning shows that $\Omega \vDash \neg\boldsymbol{A}$. ■

Now, when we are in the business of proving the Completeness Theorem, our aim is to tie '$\vDash$' to '$\vdash$'. And so, in this regard, it is very striking that there is a perfect *proof-theoretic* analogue of Lemma 6.9:

Lemma 6.10. Suppose Ω are jointly polarising and that every atomic subsentence of $\boldsymbol{A}$ is in the field of Ω. Then either $\Omega \vdash \boldsymbol{A}$ or $\Omega \vdash \neg\boldsymbol{A}$.

Proof. By (strong) induction on the length of $\boldsymbol{A}$.

Fix any sentence $\boldsymbol{A}$, and assume that all of $\boldsymbol{A}$'s atomic subsentences are in the field of Ω. For induction, suppose that, where $\boldsymbol{B}$ is any sentence shorter than $\boldsymbol{A}$ and where all of $\boldsymbol{B}$'s atomic subsentences are among the sentences that jointly span Ω, either $\Omega \vdash \boldsymbol{B}$ or $\Omega \vdash \neg\boldsymbol{B}$. There are now six cases to consider:

> *Case 1: $\boldsymbol{A}$ is atomic.* Then either $\boldsymbol{A}$ is among Ω (in which case $\Omega \vdash \boldsymbol{A}$, by the rule R), or $\neg\boldsymbol{A}$ is among Ω (in which case $\Omega \vdash \neg\boldsymbol{A}$, by the rule R). So the atomic case is straightforward.
>
> *Case 2: $\boldsymbol{A}$ is $\neg\boldsymbol{C}$.* Suppose $\Omega \nvdash \boldsymbol{A}$, i.e. $\Omega \nvdash \neg\boldsymbol{C}$. Then $\Omega \vdash \boldsymbol{C}$, by the induction hypothesis. It follows that $\Omega \vdash \neg\neg\boldsymbol{C}$ (just expand the proof of $\boldsymbol{C}$ from Ω by supposing $\neg\boldsymbol{C}$ and then using $\neg$E and $\neg$I). So $\Omega \vdash \neg\boldsymbol{A}$. So either $\Omega \vdash \boldsymbol{A}$ or $\Omega \vdash \neg\boldsymbol{A}$.
>
> *Case 3: $\boldsymbol{A}$ is $(\boldsymbol{C} \wedge \boldsymbol{D})$.* Suppose $\Omega \nvdash \boldsymbol{A}$, i.e. $\Omega \nvdash \boldsymbol{C} \wedge \boldsymbol{D}$. Then either $\Omega \nvdash \boldsymbol{C}$, or $\Omega \nvdash \boldsymbol{D}$. For if $\Omega \vdash \boldsymbol{C}$ and $\Omega \vdash \boldsymbol{D}$, then $\Omega \vdash \boldsymbol{C} \wedge \boldsymbol{D}$ (just combine the two proofs and then use $\wedge$I). So, by the induction hypothesis, either $\Omega \vdash \neg\boldsymbol{C}$, or $\Omega \vdash \neg\boldsymbol{D}$. Either way, $\Omega \vdash \neg(\boldsymbol{C} \wedge \boldsymbol{D})$. (Showing this is left as an exercise.) That is, $\Omega \vdash \neg\boldsymbol{A}$. So either $\Omega \vdash \boldsymbol{A}$, or $\Omega \vdash \neg\boldsymbol{A}$.
>
> *Case 4: $\boldsymbol{A}$ is $(\boldsymbol{C} \vee \boldsymbol{D})$.* Suppose $\Omega \nvdash \boldsymbol{A}$, i.e. $\Omega \nvdash \boldsymbol{C} \vee \boldsymbol{D}$. Then $\Omega \nvdash \boldsymbol{C}$ and $\Omega \nvdash \boldsymbol{D}$. So, by hypothesis, $\Omega \vdash \neg\boldsymbol{C}$ and $\Omega \vdash \neg\boldsymbol{D}$. So $\Omega \vdash \neg(\boldsymbol{C} \vee \boldsymbol{D})$. (Showing this is left as an exercise). That is, $\Omega \vdash \neg\boldsymbol{A}$. So either $\Omega \vdash \boldsymbol{A}$, or $\Omega \vdash \neg\boldsymbol{A}$.
>
> *Case 5: $\boldsymbol{A}$ is $(\boldsymbol{C} \to \boldsymbol{D})$.* I leave this as an exercise.
>
> *Case 6: $\boldsymbol{A}$ is $(\boldsymbol{C} \leftrightarrow \boldsymbol{D})$.* I leave this as an exercise.

This completes the (strong) induction on length. ■

We here have a very tight connection between the behaviour of '$\models$' and '$\vdash$' in the case of jointly polarising sentences. In fact, if we like, we can prove a kind of mini-version of the Soundness and Completeness Theorems:

Lemma 6.11. Suppose Ω are jointly polarising and that every atomic subsentence of $\boldsymbol{A}$ is in the field of Ω. Then $\Omega \vdash \boldsymbol{A}$ iff $\Omega \models \boldsymbol{A}$.

Proof. Left-to-right. If $\Omega \vdash \boldsymbol{A}$, then $\Omega \models \boldsymbol{A}$ by the Soundness Theorem.

Right-to-left. I prove the contrapositive, i.e. that if $\Omega \nvdash \boldsymbol{A}$ then $\Omega \not\models \boldsymbol{A}$. So suppose $\Omega \nvdash \boldsymbol{A}$. Then $\Omega \vdash \neg\boldsymbol{A}$, by Lemma 6.10. So $\Omega \models \neg\boldsymbol{A}$, by the Soundness Theorem. But in that case, $\Omega \not\models \boldsymbol{A}$, since otherwise we would have that $\Omega \models (\boldsymbol{A} \wedge \neg\boldsymbol{A})$, contradicting Lemma 6.8. ■

What Lemma 6.11 tells us, in essence, is the following: if you pin down what is happening with atomic sentences and their negations, then you determine all the semantic and proof-theoretic consequences; moreover, these consequences *coincide*. (It is no wonder, then, that SimpleSearch worked precisely by breaking down sentences into pathways through TFL-proof-skeletons that consist of atomic sentences and their negations.) And my strategy for my second proof of the Completeness Theorem is to *lift* this result up into the more general case.

Here is the idea, in a bit more detail. Let Δ be any sentences such that $\Delta \nvdash \bot$. I will first generate some jointly polarising sentences, Ω, from Δ. Since Ω are jointly polarising, there is a unique valuation which makes them all true, and, in turn, this makes all of Δ true. This will prove Lemma 6.2, from which the Completeness Theorem follows (see §6.1).

At this level of generality, the strategy is very similar to our first proof of the Completeness Theorem. But there is a difference. To generate the jointly polarising sentences from Δ, I shall appeal to the simple fact that *either* Δ are jointly contrary *or* Δ are not jointly contrary. In saying this, I am not relying on the existence of algorithm (such as SimpleSearch) to determine which possibility obtains; I am simply appealing to an instance of excluded middle (in the metalanguage). And to show that the jointly polarising sentences tautologically entail every sentence among Δ, I shall lean upon our very general observations about jointly polarising sentences.

Without further ado, then, here is the proof.

Proof of Lemma 6.2 without an algorithm. Suppose $\Delta \nvdash \bot$; let $\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_m$ be all the atomic sentences that occur as subsentences in any of the sentences among Δ.[7] I generate my jointly polarising sentences by a recursive definition:

> *Base clause.* If $\Delta, \boldsymbol{A}_1 \nvdash \bot$, then let Ω_1 be $\boldsymbol{A}_1$. Otherwise, let Ω_1 be $\neg\boldsymbol{A}_1$
> *Recursion clause.* For each number n such that $1 < n < m$:
>
> - If $\Delta, \Omega_n, \boldsymbol{A}_{n+1} \nvdash \bot$, then let Ω_{n+1} be all of Ω_n and $\boldsymbol{A}_{n+1}$
> - Otherwise, let Ω_{n+1} be all of Ω_n and $\neg\boldsymbol{A}_{n+1}$

[7] The fact that there are only finitely many such atomic sentences is guaranteed by my assumption (flagged in §6.2, and still in play) that Δ are finite.

I now make an important claim about this recursive procedure:

> **Claim.** $\Delta, \Omega_n \nvdash \bot$, for all $1 \leq n \leq m$.

I shall prove this Claim by (ordinary) induction.

> *Base case.* If $\Delta, \boldsymbol{A}_1 \nvdash \bot$, then clearly $\Delta, \Omega_1 \nvdash \bot$. So suppose instead that $\Delta, \boldsymbol{A}_1 \vdash \bot$. In this case, Ω_1 is just $\neg\boldsymbol{A}_1$. Moreover, in this case there is a TFL-proof ending in '$\bot$' with assumptions among $\Delta, \boldsymbol{A}_1$. Manipulating this TFL-proof and employing the rule $\neg$I, we obtain $\Delta \vdash \neg\boldsymbol{A}_1$. And, since $\Delta \nvdash \bot$, we obtain $\Delta, \neg\boldsymbol{A}_1 \nvdash \bot$. So, again, $\Delta, \Omega_1 \nvdash \bot$.
> *Induction case.* Fix n and suppose, for induction, that $\Delta, \Omega_n \nvdash \bot$. Now suppose that $\Delta, \Omega_n, \boldsymbol{A}_{n+1} \vdash \bot$. Then, as before we can show that $\Delta, \Omega_n, \neg\boldsymbol{A}_{n+1} \nvdash \bot$.

It follows from the Claim that $\Omega_m \nvdash \bot$. Consequently, there is no sentence such that both it and its negation are among Ω_m. So Ω_m jointly satisfy both (P1) and (P2), and hence Ω_m are jointly polarising.

Now, observe that, for any sentence $\boldsymbol{B}$ among Δ, we have $\Delta, \Omega_m \vdash \boldsymbol{B}$ (just using the rule R). Since $\Delta, \Omega_m \nvdash \bot$, it must be that $\Delta, \Omega_m \nvdash \neg\boldsymbol{B}$. *A fortiori*, $\Omega_m \nvdash \neg\boldsymbol{B}$. So $\Omega_m \vdash \boldsymbol{B}$ by Lemma 6.10, and hence $\Omega_m \vDash \boldsymbol{B}$, by the Soundness Theorem. Generalising: $\Omega_m \vDash \boldsymbol{B}$, for all $\boldsymbol{B}$ among Δ.

Finally: by Lemma 6.8, there is a valuation that makes all of Ω_m true. And since $\Omega_m \vDash \boldsymbol{B}$, for every $\boldsymbol{B}$ among Δ, this valuation makes all of Δ true. ■

This proof of Lemma 6.2 is much more abstract than the proof I offered in §6.2. It provides very little information about the valuation that makes all of Δ true, only that one exists. Of course, at the point where we consider our definition of each Ω_{n+1}, we could use SimpleSearch to *test* for joint-contrariness. But there is no *need* for us to do this. As a result, the algorithm-free proof is crisper and brisker than the proof that proceeded via an algorithm. (Indeed, *despite* all the work in §6.2, I have left an *awful* lot for you to fill in!) So we have a nice and brisk proof of Lemma 6.2. And, of course, since we have a proof of that, we have a proof of the Completeness Theorem itself.

6.4 The superior generality of the abstract proof

The more abstract proof has a further advantage over its algorithmic rival: it is a bit more *general*. I shall explain why in this section, but I should caution that this material is for enthusiasts only.

In §§6.2–6.3, I assumed that we were only given *finitely* many sentences. But what happens when we relax this assumption? Before attempting to answer that question, we should take a little time to make sure we know what it would *mean* to say that $\Gamma \vDash \boldsymbol{C}$, or that $\Gamma \vdash \boldsymbol{C}$, when Γ are *infinite*.

To say that $\Gamma \vDash \boldsymbol{C}$ is to say that there is no valuation that makes all the sentences among Γ true whilst making $\boldsymbol{C}$ false. Thus far, I have encouraged you to think about valuations in terms of truth tables. If Γ are infinite, this will probably be inappropriate, since that would require you to believe in an *infinitely large* truth

table. Nonetheless, the use of truth tables is only an heuristic. And when Γ are infinite, it still makes sense to ask whether there is a valuation which makes all of them true (and also $\boldsymbol{C}$ false).

It might, however, be somewhat mind-boggling to consider a claim that $\Gamma \vdash \boldsymbol{C}$ when Γ are *infinite*. After all, every TFL-proof is finitely long: we number each of the lines (with natural numbers) and every proof has a final line. And we could hardly start a TFL-proof by writing down *infinitely many assumptions*, and then drawing inferences from them, since we would never even get to the end of the process of writing down the assumptions!

If we want to make sense of the idea that one might be able to prove something from *infinitely* many assumptions, then we need to conceive of $\Gamma \vdash \boldsymbol{C}$ as telling us that there is a TFL-proof, all of whose initial assumptions are *among* Γ, and which ends with $\boldsymbol{C}$ on no further assumptions. Otherwise put: to say that $\Gamma \vdash \boldsymbol{C}$ is to say that there is some a finitely long TFL-proof which starts with only finitely many of the assumptions among Γ, and which ends with $\boldsymbol{C}$ on no further assumptions.

Understanding '$\vdash$' in this way does not affect the proof of the Soundness Theorem at all (do spend a few minutes confirming this!). However, *both* of our proofs of the Completeness Theorem depended upon the assumption that we were working with only finitely many sentences.

This is clear in the case of the algorithmic proof offered in §6.2. When we are concerned with Completeness, we are asking whether it follows from the fact that $\Gamma \vDash \boldsymbol{C}$, that $\Gamma \vdash \boldsymbol{C}$. If we want to deploy SimpleSearch to address this, then we shall need to input Γ together with $\neg\boldsymbol{C}$ into SimpleSearch. But if Γ are infinite, SimpleSearch will not terminate. Indeed, we will not even get past Step 1 of SimpleSearch, since we will never finish writing *all* of Γ and $\neg\boldsymbol{C}$ as assumptions!

A similar problem arises for the more abstract proof of Lemma 6.2 offered in §6.3. In the course of that proof, I assumed that only finitely many atomic sentences occur as subsentences in the sentences among Δ, namely, $\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_m$, and I used this assumption to select my jointly polarising sentences, Ω_m. But if Δ are infinite, it might be that infinitely many atomic sentences occur as subsentences among Δ, and my proof will break down.

Fortunately, this last problem can be repaired. In making our repair, the crucial insight is that every TFL-proof is only ever finitely long. This insight will allow us to prove Lemma 6.2 in full generality.

Proof of Lemma 6.2, in the fully general case. Suppose $\Delta \nvdash \bot$; let $\boldsymbol{A}_1, \boldsymbol{A}_2, ...$, be all the atomic sentences that occur as subsentences in any of the sentences among Δ. We define:

> *Base clause.* If $\Delta, \boldsymbol{A}_1 \nvdash \bot$, then let Ω_1 be $\boldsymbol{A}_1$. Otherwise, let Ω_1 be $\neg\boldsymbol{A}_1$
> *Recursion clause.* For each number $n > 1$:
>
> - If $\Delta, \Omega_n, \boldsymbol{A}_{n+1} \nvdash \bot$, then let Ω_{n+1} be all of Ω_n and $\boldsymbol{A}_{n+1}$
> - Otherwise, let Ω_{n+1} be all of Ω_n and $\neg\boldsymbol{A}_{n+1}$

Just as before, we can prove by induction that:

> **Claim 1.** $\Delta, \Omega_n \nvdash \bot$, for all n.

Now let Ω be all and only the sentences that are to be found among some (any) Ω_n. I claim:

Claim 2. $\Delta, \Omega \nvdash \bot$

To establish this claim, suppose that $\Delta, \Omega \vdash \bot$. Since every TFL-proof is only finitely long, there must be finitely many sentences, $\pm\boldsymbol{A}_{i_1}, \pm\boldsymbol{A}_{i_2}, ..., \pm\boldsymbol{A}_{i_k}$, among Ω such that:

$$\Delta, \pm\boldsymbol{A}_{i_1}, \pm\boldsymbol{A}_{i_2}, ..., \pm\boldsymbol{A}_{i_k} \vdash \bot$$

But then all of *these* sentences, $\pm\boldsymbol{A}_{i_1}, \pm\boldsymbol{A}_{i_2}, ..., \pm\boldsymbol{A}_{i_k}$ are among some Ω_n. (Specifically, they are in Ω_n, when n is the largest number among $i_1, ..., i_k$.) So $\Delta, \Omega_n \vdash \bot$, which contradicts Claim 1.

It follows from Claim 2 that Ω are jointly polarising. The remainder of the proof now proceeds exactly as before. ■

In short, pursuing our more abstract proof-strategy allows us to offer a 'completely general' proof of the Completeness Theorem. What is more, we can obtain an important result as an immediate consequence.

Theorem 6.12. Compactness Theorem. Let Δ be any sentences. Suppose that, whenever you take only finitely many sentences of Δ, those finitely many sentences are jointly consistent. Then Δ themselves are jointly consistent.

Proof. We prove the contrapositive. Suppose that Δ are jointly inconsistent. By the (completely general) Completeness Theorem, $\Delta \vdash \bot$. Since every TFL-proof is only finitely long, there are finitely many sentences, Δ_0, among Δ such that $\Delta_0 \vdash \bot$. By the Soundness Theorem, Δ_0 are jointly inconsistent. ■

Practice exercises

A. Justify the following principles, which are appealed to in the proof of Lemma 6.10 and in the more abstract proof of Lemma 6.2.

1. If $\Gamma \vdash \boldsymbol{C}$ then $\Gamma \vdash \neg\neg\boldsymbol{C}$
2. If $\Gamma \nvdash \boldsymbol{C} \wedge \boldsymbol{D}$, then either $\Gamma \nvdash \boldsymbol{C}$ or $\Gamma \nvdash \boldsymbol{D}$
3. If $\Gamma \vdash \neg\boldsymbol{C}$ or $\Gamma \vdash \neg\boldsymbol{D}$, then $\Gamma \vdash \neg(\boldsymbol{C} \wedge \boldsymbol{D})$
4. If $\Gamma \vdash \neg\boldsymbol{C}$ and $\Gamma \vdash \neg\boldsymbol{D}$, then $\Gamma \vdash \neg(\boldsymbol{C} \vee \boldsymbol{D})$.
5. If $\Gamma \nvdash \bot$, then either $\Gamma, \boldsymbol{A} \nvdash \bot$ or $\Gamma, \neg\boldsymbol{A} \nvdash \bot$.
6. If $\Gamma \vdash \boldsymbol{A}$ and $\Gamma \nvdash \bot$ then $\Gamma, \boldsymbol{A} \nvdash \bot$.
7. If $\Gamma, \boldsymbol{A} \nvdash \bot$, then $\Gamma \nvdash \neg\boldsymbol{A}$
8. If $\Gamma, \neg\boldsymbol{A} \vdash \bot$, then $\Gamma \vdash \boldsymbol{A}$

B. Work through Cases 5 and 6 of the proof of Lemma 6.10.

Tim Button is a Senior Lecturer, and Fellow of St John's College, at the University of Cambridge. He mostly works on meta(meta)physics, logic, mathematics, and language. You can find out more at nottub.com.