Isabelle for Philosophers

Ben Blumson

August 12, 2019

> It is unworthy of excellent men to lose hours like slaves in the labour of calculation which could safely be relegated to anyone else if machines were used.
>
> Liebniz

This is an introduction to the Isabelle proof assistant aimed at philosophers and students of philosophy.[1]

1 Propositional Logic

Imagine you are caught in an air raid in the Second World War. You might reason as follows:

> Either I will be killed in this raid or I will not be killed. Suppose that I will. The even if I take precautions, I will be killed, so any precautions I take will be ineffective. But suppose I am not going to be killed. Then I won't be killed even if I neglect all precautions; so on this assumption, no precautions are necessary to avoid being killed. Either way, any precautions I take will be either ineffective or unnecessary, and so pointless.[2]

The example is notable for two reasons.

First, if the argument were successful, then it would establish a version of *fatalism*, according to which we cannot influence future events. Any

[1] I found a very useful introduction to be Nipkow [7]. A person wishing to know more about how Isabelle works might first consult Paulson [8]. For the software itself and comprehensive documentation, see https://isabelle.in.tum.de/. Isabelle might not be the right too for your project – for a comparison of alternatives see Wiedijk [13].

[2] This example is from Dummett [5] p. 345, but the version quoted here is from Stalnaker [11] p. 280.

future event will either happen or not. If it happens then, according to the argument, it will happen even if I try to prevent it. On the other hand, if it doesn't happen, then it won't happen regardless of whether I try to prevent it. Either way, trying to prevent it is pointless. And the same goes for trying to prevent or bring about any other future event.

Second, and more importantly for our purposes, the argument is an example of a style emulated by the natural deduction rules for propositional logic.[3] In this system, each propositional connective is associated with two rules: an introduction rule, which allows you to introduce it into an argument, and an elimination rule, which allows you to derive something from it. Proofs in Isabelle are presented using natural deduction, so knowing the introduction and elimination rules is all you need to understand the proofs.[4] We will take the rules for each connective in turn.

1.1 Conditionals

Conditionals are translated using an arrow. So "if it's raining then it's cloudy", for example, is translated $A \longrightarrow B$, where A stands for "it's raining" and B stands for "it's cloudy". The right hand side, in this case "it's cloudy", is known as the "consequent" – since it's the consequence of the condition – whereas the left had side, in this case "it's raining" is known as the antecedent – since it's the condition upon which the consequent obtains. The next two subsections explain the introduction and elimination rules for conditionals.

1.1.1 Conditional Proof

According to the introduction rule for conditionals, sometimes known as "conditional proof", in order to prove a conditional one must assume the antecedent and show the consequent. Here is the very simplest example:

```
lemma A ⟶ A
proof (rule impI)
  assume A
  then show A.
qed
```

There are a few things to note about this example. The first line simply states the lemma to be proved – in this case $A \longrightarrow A$. The second line opens the proof, and says it will proceed by the rule of conditional proof, which is abbreviated as *impI*, for "implication introduction". The third line opens

[3] For a classic introduction to logic based on natural deduction, see Lemmon [6].

[4] This part of the Isabelle system is known as "Isar" for "Intelligible Semi-Automated Reasoning" See Wenzel [12].

the assumption A, the fourth line uses this assumption to show A, and the fifth line says that proof is finished.

There are two things about the proof that aren't quite so obvious. First, the word *then* at the beginning of the fourth line says that this step in the proof follows from the previous line. Second, the period at the end of the fourth line says, roughly, that this line reiterates, or is an instance of, something already assumed or proved – in this case the assumption in the previous line.

Here is another simple example:

```
lemma positive-paradox: A ⟶ B ⟶ A
proof (rule impI)
  assume A
  show B ⟶ A
  proof (rule impI)
    assume B
    from ‹A› show A.
  qed
qed
```

This proof is also very simple, but there's a few more things to note.

First, there are no brackets in the statement of the lemma. This is because there is convention that conditionals "associate to the right". In other words, the lemma translates back into English as "if A then if B then A", as opposed to "if if A then B then A".

Second, just as in the last proof, this proof assumes the antecedent A and shows the consequent $B \longrightarrow A$. But this time the consequent does not just reiterate something already given, but itself has to proved. So the fifth line opens a new subproof within the proof. This subproof is closed in the eighth line, and the proof as a whole closed in the ninth.

Third, on the seventh line A does not follow from the assumption just before on the sixth line, but from the much earlier assumption on the third line. So instead of using *then* to refer to the previous line, we use *from* $\langle A \rangle$ to refer all the way back to the assumption on the third line.

Fourth, if you look closely at the proof you will notice that the assumption B on the sixth line is not used to show anything at all – A is doing all the work. This is quite normal in classical natural deduction systems, but it's avoided in, for example, relevant logics, which take issue with the fact that B is "irrelevant" in this proof.[5] The logic automated by Isabelle is, of course, *classical* logic.

[5] For relevant logics, see especially Anderson and Belnap [1].

This point is important because it is not obvious that the lemma is true. Suppose, for example, that A translates “I will die young” and B translates “I will live healthily”. Then the lemma as a whole translates “If I will die young, then I will die young even if I live healthily” But if my unhealthy lifestyle is the cause of my death, this is intuitively false.

The example illustrates that classical logic is not philosophically neutral, even in some of its simplest manifestations. That means that not everything “proved” here – no matter how reliable the software or rigorous the proofs – is incontrovertibly true. Whether the reasoning below is right depends on whether classical logic is right. Nothing – person or machine – can guarantee that for certain.

The example also illustrates that results must be interpreted with care. The example sounds in English as if it supports a fatalistic conclusion – that my dying young is outside of my control. But classically, the sentence is equivalent to “if I will die young, then either I will not live healthily or I will die young” which, while true, has no fatalistic consequences.

Finally, note that the lemma has been given a name, viz. *positive-paradox*. This is helpful if we wish to refer back to the lemma in a later proof. It also reminds us about the significance of the lemma – in this case that it is one of the notorious “paradoxes of material implication”, for the reasons just mentioned.

Exercise 1 The Strict Positive Paradox

Practice using conditional proof by proving:

lemma *strict-positive-paradox*: $A \longrightarrow B \longrightarrow B$ **oops**

Where would a proponent of relevant logic find fault with this proof? Think of an example to show it’s not obvious that this lemma is true. This problem is known as the *strict* positive paradox of material implication, since it’s consequent $B \longrightarrow B$ is a necessary truth.

Note that the command *oops* allows you to state a lemma without proving it. Delete it before you start your proof. If you need to use a lemma that you haven’t proved in another proof, you can write *sorry* instead of *oops*. This command should obviously be used with care, since a lemma merely derived from an unproved lemma is itself unproved.

1.1.2 Modus Ponens

According to the *elimination* rule for conditionals, normally called modus ponens, from a conditional and its antecedent you can show its consequent. Here is a simple example:

```
lemma A ⟶ (A ⟶ B) ⟶ B
proof (rule impI)
  assume A
  show (A ⟶ B) ⟶ B
  proof (rule impI)
    assume A ⟶ B
    thus B using ‹A› by (rule mp)
  qed
qed
```

The important part of this proof is the step from the sixth to seventh line, which uses modus ponens to derive B from $A \longrightarrow B$ using B. The rest of the proof works by two applications of conditional proof, as just described in subsection 1.1.1. The only nuance is that *then show* is now abbreviated *thus*, which is purely for the sake of brevity.

Here is a slightly more complicated example:

```
lemma contraction: (A ⟶ A ⟶ B) ⟶ (A ⟶ B)
proof (rule impI)
  assume A ⟶ A ⟶ B
  show A ⟶ B
  proof (rule impI)
    assume A
    with ‹A ⟶ A ⟶ B› have A ⟶ B by (rule mp)
    thus B using ‹A› by (rule mp)
  qed
qed
```

Three things are notable about this proof. First, on line seven we said that we *have* $A \longrightarrow B$ instead of that we *show* it. This is just to reflect that $A \longrightarrow B$ is not our final goal of this subproof – it's just an intermediate step on our way to B which we reach only on the eight line. In general, *show* and *thus* will appear only on the last line of a proof or subproof.

Second, order matters – modus ponens works by deriving the consequent from the conditional followed by the antecedent, not from the antecedent followed by the conditional. To see what I mean, consider the following variation of the same proof:

```
lemma (A ⟶ A ⟶ B) ⟶ (A ⟶ B)
proof (rule impI)
  assume A ⟶ A ⟶ B
  show A ⟶ B
  proof (rule impI)
    assume A
    then have A ⟶ B using ‹A ⟶ A ⟶ B› by (rule rev-mp)
    thus B using ‹A› by (rule mp)
  qed
qed
```

Everything is almost the same, except on the seventh line, where the order is switched and *with* is replaced by *then* and *using* and *mp* is replaced by *rev-mp*. This is annoying, because we don't normally care which order we have the antecedent and conditional in when we apply modus ponens. But it doesn't matter, because you get used to it.

Exercise 2

Practice conditional proof and modus ponens by proving:

```
lemma prefixing: (A ⟶ B) ⟶ (C ⟶ A) ⟶ (C ⟶ B) oops
```

```
lemma suffixing: (A ⟶ B) ⟶ (B ⟶ C) ⟶ (A ⟶ C) oops
```

Would a relevant logician logician find these proofs acceptable? What about the proof of contraction above?

1.1.3 Biconditional Introduction

The introduction rule for a biconditional is just like the introduction rule for a conditional, except as well as assuming the left hand side and proving the right hand side, one must also assume the right hand side and prove the left hand side. Here is a very simple example:

```
lemma A ⟷ A
proof (rule iffI)
  assume A
  thus A.
next
  assume A
  thus A.
qed
```

Note that in order to prove a biconditional, one must prove two things: the right side from the left side, and the left side from the right side. The word *next* in the middle of the proof is just to signal the move from solving the first goal to solving the second.

1.1.4 Biconditional Elimination:

There are two elimination rules for biconditionals. The first is the same as modus ponens – from a biconditional and it's left hand side, one can infer it's right hand side. For example:

```
lemma (A ⟷ B) ⟶ A ⟶ B
proof
  assume A ⟷ B
  show A ⟶ B
```

```
 proof
   assume A
   with ‹A ⟷ B› show B by (rule iffD1)
 qed
qed
```

The second is the reverse – from a biconditional and it's right hand side, one can infer it's left hand side. For example:

```
lemma (A ⟷ B) ⟶ B ⟶ A
proof
 assume A ⟷ B
 show B ⟶ A
 proof
   assume B
   with ‹A ⟷ B› show A by (rule iffD2)
 qed
qed
```

Notice that both these proofs work by conditional proof, but I've omitted to say so. If you open a new subproof without specifying a rule, Isabelle will default to using the introduction rule for the main connective of what you are trying to prove. This helps to keep proofs tidy, and focus attention on the steps that matter most.

Exercise 3

Practice biconditional elimination and introduction by proving:

```
lemma (A ⟷ B) ⟷ (B ⟷ A) oops
```

1.2 Conjunction

Conjunctions are translated with a wedge. So "it's raining and it's cloudy", for example, is translated $A \wedge B$, where A stands for "it's raining" and B stands for "it's cloudy". The next two subsections explain the introduction and elimination rules for conjunctions.

1.2.1 Conjunction Elimination

According to the rule of conjunction elimination, from a conjunction, you can show each conjunct. For example, from $A \wedge B$ you can show A:

```
lemma A ∧ B ⟶ A
proof
 assume A ∧ B
 thus A by (rule conjE)
qed
```

And from $A \wedge B$ you can also show B:

```
lemma A ∧ B ⟶ B
proof
  assume A ∧ B
  thus B by (rule conjE)
qed
```

Here is a more interesting example:

```
lemma import: (A ⟶ B ⟶ C) ⟶ (A ∧ B ⟶ C)
proof
  assume A ⟶ B ⟶ C
  show A ∧ B ⟶ C
  proof
    assume A ∧ B
    then have A by (rule conjE)
    with ‹A ⟶ B ⟶ C› have B ⟶ C..
    from ‹A ∧ B› have B by (rule conjE)
    with ‹B ⟶ C› show C..
  qed
qed
```

Two things are notable about this proof. First, modus ponens or conditional elimination is used twice in this proof. But this has been abbreviated by two dots instead. This abbreviation can be used with all the basic introduction and elimination rules, for brevity.

Second, there are no brackets around $A \wedge B$ in the statement of the lemma. This is because there is a convention that conjunction has higher priority than implication. That means we do need brackets around $A \longrightarrow B$ in the following example:

```
lemma A ∧ (A ⟶ B) ⟶ B
proof
  assume A ∧ (A ⟶ B)
  then have A ⟶ B by (rule conjE)
  from ‹A ∧ (A ⟶ B)› have A by (rule conjE)
  with ‹A ⟶ B› show B..
qed
```

Notice that in both these proof we had to use conjunction elimination twice – once for each conjunct. This is of course a common pattern.

Exercise 4 Strengthening the Antecedent

Practice conjunction elimination by proving:

```
lemma strengthening-the-antecedent: (A ⟶ C) ⟶ (A ∧ B ⟶ C) oops
```

Thinks of an example to show it's not obvious that this lemma is true. Would a relevant logician find fault with this proof? This lemma is known as “strengthening the antecedent”, since $A \wedge B$ is stronger than, or in other words entails, A.

1.2.2 Conjunction Introduction

According to the rule for conjunction introduction, from the first and second conjuncts, you can show the conjunction. Here is a very simple example:

```
lemma conjunction-commutative: A ∧ B ⟶ B ∧ A
proof
  assume A ∧ B
  hence B..
  from ⟨A ∧ B⟩ have A..
  with ⟨B⟩ show B ∧ A by (rule conjI)
qed
```

Note that *hence* in this proof abbreviates *then have*, just as *thus* abbreviates *then show*, again for the sake of brevity.

Here is a more interesting example:

```
lemma export: (A ∧ B ⟶ C) ⟶ (A ⟶ B ⟶ C)
proof
  assume antecedent: A ∧ B ⟶ C
  show A ⟶ B ⟶ C
  proof
    assume A
    show B ⟶ C
    proof
      assume B
      with ⟨A⟩ have A ∧ B by (rule conjI)
      with antecedent show C..
    qed
  qed
qed
```

Note how in this proof we have named the opening assumption “antecedent” so we can refer back to it by name in the final step, instead of quoting the whole line. This becomes very useful in the presentation of more complex proofs. Not also that, as with modus pones, order matters for conjunction introduction.

Like conditionals, conjunctions “associate to the right”. This means that the associativity of conjunction has to be proved:

```
lemma conjunction-associative: A ∧ B ∧ C ⟷ (A ∧ B) ∧ C
proof
  assume left: A ∧ B ∧ C
  hence A..
  from left have B ∧ C..
  hence B..
  with ‹A› have A ∧ B..
  from ‹B ∧ C› have C..
  with ‹A ∧ B› show (A ∧ B) ∧ C..
next
  assume right: (A ∧ B) ∧ C
  hence A ∧ B..
  hence B..
  from right have C..
  with ‹B› have B ∧ C..
  from ‹A ∧ B› have A..
  thus A ∧ B ∧ C using ‹B ∧ C›..
qed
```

Notice that in the left to right direction of this proof, $A \wedge B$ couldn't be derived from $A \wedge B \wedge C$ in a single step – A and B had to be derived separately first. This is because the conjunction associates to the right, and so $A \wedge B$ is not a conjunct of $A \wedge B \wedge C$ at all – its conjuncts are just A and $B \wedge C$.

Exercise 5

Practice conjunction introduction by proving:

```
lemma (A ⟶ B) ⟶ (A ⟶ C) ⟶ A ⟶ B ∧ C oops
```

1.3 Disjunction

Disjunctions are translated with a vee. So “it's raining or it's cloudy” is translated $A \vee B$, where A stands for “it's raining” and B stands for “'it's cloudy”. Needless to say, disjunction is *inclusive*, so “it's raining or it's cloudy” is compatible with “it's raining and it's cloudy”. The next two subsections explain the introduction and elimination rules for disjunction.

1.3.1 Disjunction Introduction

There are two rules for disjunction introduction. According to the first, you can show a disjunction from its first disjunct. For example:

```
lemma A ⟶ A ∨ B
proof
  assume A
  thus A ∨ B by (rule disjI1)
qed
```

According to the second, you can show a disjunction from its second disjunct. For example:

```
lemma B ⟶ A ∨ B
```

```
proof
  assume B
  thus A ∨ B by (rule disjI2)
qed
```

Note that we can omit brackets around the disjunction, since it has higher priority than implication. However, conjunction has higher priority than disjunction.

Exercise 6

Practice disjunction introduction by proving:

```
lemma (A ⟶ B) ⟶ (A ⟶ B ∨ C) oops
```

1.3.2 Disjunction Elimination

Disjunction elimination is a bit more complicated. According to it, if you have a disjunction, and you can prove something from both its disjuncts, then you can prove that thing simpliciter. Here is a simple example:

```
lemma A ∨ A ⟶ A
proof
  assume A ∨ A
  thus A
  proof (rule disjE)
    assume A
    thus A.
  next
    assume A
    thus A.
  qed
qed
```

Note the use of “thus” on the fourth line of this proof. To use disjunction elimination, you need to show three things – the disjunction itself, that the conclusion follows from the first disjunct, and that the conclusion follows from the second disjunct. In this case, we already had the disjunction, so we wrote “thus” in order to use it. But we could equally well have written the proof in this slightly longer way:

```
lemma A ∨ A ⟶ A
proof
  assume A ∨ A
  show A
  proof (rule disjE)
    show A ∨ A using ‹A ∨ A›.
  next
    assume A
    thus A.
  next
    assume A
    thus A.
  qed
qed
```

Note also that disjunction elimination is the key rule in our motivating example from section 1 – I will be killed in the air raid or I will not, but *either way* taking precautions is pointless, so taking precautions is pointless.

Like conditionals and conjunctions, disjunctions associate to the right.

Exercise 7

Practice disjunction elimination by proving:

```
lemma (A ⟶ C) ∧ (B ⟶ C) ⟶ A ∨ B ⟶ C oops
```

Exercise 8 The Associativity of Disjunction

Prove the associativity of disjunction:

```
lemma A ∨ B ∨ C ⟷ (A ∨ B) ∨ C oops
```

Exercise 9

Practice disjunction elimination and introduction by proving:

```
lemma A ∨ B ∧ C ⟶ (A ∨ B) ∧ (A ∨ C) oops
```

Can you prove the converse from the rules covered so far? Why or why not?

1.4 Negation

Negation is translated by ¬, so "it's not raining" is translated ¬ A, where A stands for "it's raining". Like the other connectives, negation has an introduction rule and an elimination rule. We also discuss two other rules in this section – classical contradiction, which distinguishes classical from intuitionistic logic, and proof by cases.

1.4.1 Negation Elimination

According to the rule for negation elimination, if one has a negation, and also what it negates, then one can derive anything at all. Here is a simple example:

```
lemma negative-paradox: ¬ A ⟶ A ⟶ B
proof
  assume ¬ A
```

```
 show A ⟶ B
 proof
  assume A
  with ‹¬ A› show B by (rule notE)
 qed
qed
```

Notice that B in this proof is completely arbitrary, and could have been any proposition at all.

This point is philosophically important, because it is not obvious that the lemma is true. Suppose, for example, that A translates “I live healthily” and B translates “I will die young”. Then the lemma as a whole translates as “If I do not live healthily, then I will die young even if I live healthily”. But if my unhealthily lifestyle is the cause of my death, this is intuitively false.

The example is clearly closely related to the positive paradox of material implication from section 1.1.1, and for this reason it is known as the *negative* paradox of material implication. For this reason, both the lemma and the rule that supports it are rejected by relevant logicians (even though there is no unused assumption here).[6] This is worth remembering, since otherwise the negative paradox is often a source of surprise.

Exercise 10

Prove that a contradiction entails anything:

lemma *explosion*: $A \wedge \neg\ A \longrightarrow B$ **oops**

Exercise 11

Suppose the butler did it or the gardener did it. Then prove that if the butler didn't do it, the gardener did:

lemma $A \vee B \longrightarrow \neg\ A \longrightarrow B$ **oops**

How is the proof of this lemma related to the paradoxes of material implication. Would a relevant logician accept it?

1.4.2 Negation Introduction

According to the rule for negation introduction if you assume something, and then you show *False*, then you can show the negation of what you assumed. Here is an example, sometimes known as the law of non-contradiction:

lemma $\neg\ (A \wedge \neg\ A)$

[6] See Anderson and Belnap [1] pp. 163-7.

```
proof (rule notI)
  assume A ∧ ¬A
  hence  ¬ A..
  moreover from ‹A ∧ ¬ A› have A..
  ultimately show False by (rule notE)
qed
```

Two things are notable about this proof. First *False* doesn't have any introduction rule of its own – it's shown using by negation elimination, which as we emphasised in the previous subsection can be used to show *anything* from a contradiction.

Second, *False* was shown from two facts – $\neg A$ and A. So as to avoid having to refer back to the first of these by name, we used the command *moreover* followed by the command *ultimately.*

Exercise 12

Practice negation introduction by proving:

```
lemma A ⟶ ¬ ¬ A oops
```

Exercise 13

The next example is challenging, but instructive. Prove:

```
lemma ¬ ¬ (A ∨ ¬ A) oops
```

Hint: Assume $\neg (A \vee \neg A)$ and then prove $A \vee \neg A$ by disjunction introduction from $\neg A$. Can you prove simply $A \vee \neg A$ from the rules covered so far. Why or why not?

1.4.3 Classical Contradiction

The rules we have learnt so far constitute the propositional fragment of *intuitionistic* logic. To get the full strength of classical logic, we need the rule of classical contradiction, according to which if you can show *False* from a negation, then you can show what it negates. Here is the simplest example:

```
lemma (¬ A ⟶ False) ⟶ A
proof
  assume ¬ A ⟶ False
  show A
  proof (rule ccontr)
    assume ¬ A
    with ‹¬ A ⟶ False› show False..
  qed
qed
```

And here is a proof of double negation elimination

```
lemma double-negation-elimination: ¬¬A ⟶ A
proof
  assume ¬¬A
  show A
  proof (rule ccontr)
    assume ¬ A
    with ‹¬¬A› show False..
  qed
qed
```

Note that in many presentations of natural deduction, double negation elimination is the basic rule and it is classical contradiction which is derived.

Exercise 14 The Law of Excluded Middle

Prove the law of excluded middle:

```
lemma excluded-middle: A ∨ ¬ A oops
```

How is this proof related to the proof in exercise 13, and to double negation elimination?

1.4.4 Proof by Cases

Proof by cases is really the application of disjunction elimination using the law of excluded middle – but since this is such a common pattern, it helps to have an abbreviation. As a simple example, we use it to give another (circular) proof of the law of excluded middle itself:

```
lemma A ∨ ¬ A
proof cases
  assume A
  thus A ∨ ¬ A..
next
  assume ¬ A
  thus A ∨ ¬ A..
qed
```

Exercise 15 Conditional Excluded Middle

Prove the Law of Conditional Excluded Middle:

```
lemma (A ⟶ B) ∨ (A ⟶ ¬ B) oops
```

How is the proof related to the positive paradox? Can you think of an intuitive counterexample?

Exercise 16

Prove:

lemma $(A \longrightarrow B) \vee (B \longrightarrow A)$ **oops**

Think of an example to show it's not obvious that this lemma is true. How is the proof of the lemma related to the paradoxes of material implication?

Exercise 17

Prove the converse from Exercise 9:
lemma $(A \vee B) \wedge (A \vee C) \longrightarrow A \vee B \wedge C$ **oops**

Exercise 18 The Equivalence Thesis

The theory of conditionals encapsulated in the classical natural deduction rules can be summed up by the equivalence thesis, according to which a conditional is true if and only if its antecedent is false or its consequent is true. So prove:
lemma $(A \longrightarrow B) \longleftrightarrow (\neg A \vee B)$ **oops**

Equivalently, a conditional is true if and only if it's not the case that it's antecedent is true and it's consequent is false. So prove:
lemma $(A \longrightarrow B) \longleftrightarrow \neg (A \wedge \neg B)$ **oops**

Where would a proponent of relevant logic fault these proofs?

Exercise 19 The Air Raid

The motivating argument from section 1 could be formalised like this:

lemma
  assumes $A \vee \neg A$
  assumes $A \longrightarrow B \longrightarrow A$
  assumes $(B \longrightarrow A) \longrightarrow D$
  assumes $\neg A \longrightarrow C \longrightarrow \neg A$
  assumes $(C \longrightarrow \neg A) \longrightarrow D$
  shows D **oops**

Note that premises can be written with *assumes* and the conclusion with *shows*. Which of the premises can be proven in classical logic? Where could an intuitionist logician object to the argument. Where could a relevant logician object? And where must a classical logician, who accepts the equivalence thesis but rejects fatalism, object?

2 Predicate Logic

2.1 Universal Quantification

The universal quantifier is translated with an upside down "A". So "all men are mortal", for example, is translated as $\forall x.\ F\,x \longrightarrow G\,x$ where $F\,x$ stands for “is a man” and $F\ x$ for “is mortal”. The next two subsections explain the introduction and elimination rules for the universal quantifier.

2.1.1 Universal Elimination

If you have a universal statement, then you can replace the variable it binds with any term (of the same type). For example, if everything is an F then a is an F:

```
lemma (∀ x. F x) ⟶ F a
proof
  assume ∀ x. F x
  thus F a by (rule allE)
qed
```

Two things are notable about this example. The first is that the conventions for brackets are slightly different from usual – the scope of the quantifier is everything within the surrounding brackets. The second is that there has to be a space between the predicate and name or variable, to make sure they are different terms (the advantage of this is that terms don't have to be a single letter or character, and so you don't have to worry about running out).

Exercise 20

Practice universal elimination by proving:

```
lemma (∀ x. F x) ⟶ F a ∧ F b oops
```

Exercise 21 The Riddle of Dracula

Prove that if everyone is afraid of Dracula, then if Dracula is afraid only of me, then I am Dracula:

```
lemma (∀ x. R x d) ⟶ (∀ z. R d z ⟶ z = m) ⟶ d = m oops
```

Why is this lemma surprising?[7]

2.1.2 Universal Introduction

To introduce a universally quantified statement, once must first prove an instance for an arbitrary term. Here is a very simple example:

```
lemma ∀ x. F x ⟶ F x
proof (rule allI)
  fix a
  show F a ⟶ F a
```

[7] This example is from Richard Cartwright, reported by Smullyan [10] p. 212.

```
  proof
    assume F a
    thus F a.
  qed
qed
```

The role of *fix* in the third line is to introduce an arbitrary term. I've used the term *a*, as one might in an introductory logic textbook, but of course any new term would do – a popular choice in this case would just be *x*.

Exercise 22

Practice universal elimination and introduction by proving:

```
lemma (∀ x. F x ∧ G x) ⟶ (∀ x. F x) oops
```

Exercise 23

Prove that if everyone is at the party, then everyone in the world is at the party:

```
lemma (∀ x. F x) ⟶ (∀ x. F x ⟶ G x) oops
```

How is this lemma related to the positive paradox?

2.2 Existential Quantification

The existential quantifier is translated with a backward "E". So 'some man is mortal', for example, is translated $\exists x.\ F\ x \wedge G\ x$ where $F\ x$ stands for 'is a man' and $F\ x$ stands for 'is mortal'. The next two subsections explain the introduction and elimination rules for the existential quantifier.

2.2.1 Existential Introduction

According to the rule of existential introduction, from some term satisfying a sentence, one can show that something satisfies that sentence. For example:

```
lemma F a ⟶ (∃ x. F x)
proof
  assume F a
  thus ∃ x. F x by (rule exI)
qed
```

Here is a trickier example:

```
lemma ∃ x. ¬ F x ∨ F x
proof –
  from excluded-middle have ¬ F a ∨ F a.
  thus ∃ x. ¬ F x ∨ F x by (rule exI)
```

```
qed
```

Notice that there is a "–" just after *proof*. This is to stop Isabelle from defaulting to applying the existential introduction immediately, as she normally would. If she did, then she would expect you to show ¬ $F\ a \lor F\ a$ for some *old* name a. But you don't have any old name, and so you'd be stuck. Instead, you have to prove ¬ $F\ a \lor F\ a$ first, and then apply existential introduction afterwards – now to an old name.

Exercise 24 The Converse Drinkers Principle

Prove that there is someone such that if anyone drinks, then they do:

```
lemma ∃ x. (∃ y. F y) ⟶ F x oops
```

How is this proof related to the paradoxes of material implication?[8]

Exercise 25

Prove that if not everything is F, something is not F:

```
lemma not-all-implies-some-not: ¬ (∀ x. F x) ⟶ (∃ x. ¬ F x) oops
```

Would an intuitionist accept this proof?

Exercise 26

Prove that if everything is F, then something is F:

```
lemma (∀ x. F x) ⟶ (∃ x. F x) oops
```

2.2.2 Existential Elimination

According to the rule of existential elimination, if something satisfies a sentence, then you can obtain a name for that thing. For example:

```
lemma (∃ x. F x ∧ G x) ⟶ (∃ x. F x)
proof
  assume ∃ x. F x ∧ G x
  then obtain a where F a ∧ G a by (rule exE)
  hence F a..
  thus ∃ x. F x..
qed
```

Note that you can use any letter for the introduced term, but the computer can tell if you try to cheat. For example, you cannot prove:

```
lemma (∃ x. F x) ⟶ F a oops
```

[8] This problem is from Smullyan [10] p. 210-1. It is the converse of exercise 28.

Since although you can use existential elimination to obtain $F\ a$, your computer will not accept that as resolving your goal, since it knows that the "new" name you introduced is not the same as the "old" name you had in your goal (try it and you'll see what I mean).

Exercise 27

Practice existential introduction and elimination by proving:

lemma $(\exists\ x.\ F\ x) \longrightarrow (\exists\ x.\ F\ x \lor G\ x)$ **oops**

Exercise 28 The Drinker Principle

Prove that there is someone such that if they drink, then everybody drinks:

lemma $\exists\ x.\ F\ x \longrightarrow (\forall\ x.\ F\ x)$ **oops**

How is this theorem related to the paradoxes of material implication?[9]

2.3 Identity

The identity predicate is translated by the familiar sign =. So 'Hesperus is Phosphorus', for example, is translated as $a = b$.

2.3.1 Reflexivity

According to the introduction rule for identity, one may show at anytime that something is identical to itself. For example, we can prove that everything is self-identical:

lemma $\forall\ x.\ x = x$
proof
  fix a
  show $a = a$ **by** (*rule refl*)
qed

Exercise 29

Practice the reflexivity rule by proving:

lemma $F\ a \longrightarrow a = a$ **oops**

Exercise 30

Prove that everything is identical to something:

lemma $\forall\ x.\ \exists\ y.\ x = y$ **oops**

[9] This problem is from Smullyan [10], pp. 209-11. It's a common example in automated theorem proving. See, for example, Barendregt [2], pp. 54-55.

2.3.2 Substitution

According to the rule of substitution, if you have $x = y$ and you have A, then you can substitute y for occurrences of x in A. For example:

```
lemma a = b ⟶ F a ⟶ F b
proof
  assume a = b
  show F a ⟶ F b
  proof
    assume F a
    with ‹a = b› show F b by (rule subst)
  qed
qed
```

Notice that this rule only allows you to use $a = b$ to substitute a for b, and not vice versa. However, the following variation of the rule is available:

```
lemma a = b ⟶ F b ⟶ F a
proof
  assume a = b
  show F b ⟶ F a
  proof
    assume F b
    with ‹a = b› show F a by (rule ssubst)
  qed
qed
```

The difference is subtle – just one extra ‘s’ at the beginning of the rule.

Exercise 31

Prove the symmetry of identity:

```
lemma a = b ⟶ b = a oops
```

Exercise 32

Prove the transitivity of identity:

```
lemma a = b ⟶ b = c ⟶ a = c oops
```

Exercise 33 The Indiscernibility of Identity

Prove the indiscernibility of identicals:

```
lemma x = y ⟶ (F x ⟷ F y) oops
```

2.4 Definite Descriptions

According to the introduction rule for definite descriptions, to show that something is the F one may first show two things. First, that it is an F. Second that any arbitrary F is that thing. For example:

```
lemma (THE x. x = a) = a
proof (rule the-equality)
  show a = a..
next
  fix b
  assume b = a
  thus b = a.
qed
```

Note that one cannot eliminate definite descriptions in the way one might expect. For example, neither of the following can be proved:

```
lemma G (THE x. F x) ⟶ (∃ x. F x) oops
lemma F (THE x. F x) ⟶ (∃ x. ∀ y. F y ⟶ y = x) oops
```

The advantage of this is that definite descriptions function just like any other term. For example the following is valid:

```
lemma (∀ x. F x) ⟶ F (THE x. G x)
proof
  assume ∀ x. F x
  thus F (THE x. G x) by (rule allE)
qed
```

This is not in accordance with the traditional Russellian theory, so this is something that has to be kept in mind, especially since many philosophers do assume the Russellian analysis.[10]

Exercise 34

Practice introducing definite descriptions by proving:

```
lemma (∀ x. F x ⟷ x = a) ⟶ (THE x. F x) = a oops
```

3 Automation

By now you probably feel more like the slave from Liebniz' quotation than an excellent person. But happily, Isabelle contains many automated tools to make your work easier. I will describe three of the most useful.

3.1 Nitpick

Nitpick is a counterexample generator.[11] For example, to generate a counterexample to the fallacy of affirming the consequent, you could type:

```
lemma
  assumes p ⟶ q
```

[10] For Russell's theory of definite descriptions see [9].

[11] See Blanchette and Nipkow [3].

```
  assumes q
  shows p nitpick oops
```

In which case nitpick will inform you of a countermodel in which p is false and q is true.

3.2 Sledgehammer

Sledgehammer looks for a proof using various automated theorem provers.[12] Here is an example:

```
lemma (∀ x. F x ⟶ G x) ∨ (∃ x. F x ∧ ¬ G x) sledgehammer
  by auto
```

To produce an explicit natural deduction style proof, you can try:

```
lemma (∀ x. F x ⟶ G x) ∨ (∃ x. F x ∧ ¬ G x) sledgehammer [isar-proofs]
proof –
{ assume ¬ F v0-0 ∨ G v0-0
  have ?thesis
      by blast }
  then show ?thesis
    by blast
qed
```

Unsurprisingly, the result is not quite so legible as a hand written proof.

3.3 Try

What if you don't know whether the statement you're interested in is a theorem? Try try:

```
lemma (∀ x. ∃ y. R x y) ⟶ (∃ y. ∀ x. R x y) try oops
```

```
lemma (∃ x. ∀ y. R x y) ⟶ (∀ y. ∃ x. R x y) try
  by auto
```

References

[1] A. Anderson and N. Belnap. *Entailment, Vol. 1: The Logic of Relevance and Necessity.* Princeton University Press, Princeton, 1976.

[2] H. P. Barendregt. The Quest for Correctness. *Images of SMC Research*, 1996.

[12] See Blanchette and Paulson [4].

[3] J. C. Blanchette and T. Nipkow. Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder. In *International Conference on Interactive Theorem Proving*, pages 131–146. Springer, 2010.

[4] J. C. Blanchette and L. C. Paulson. Hammering Away-A User's Guide to Sledgehammer for Isabelle. Technical report, HOL, Tech. rep., 2016.

[5] M. Dummett. Bringing About the Past. *The Philosophical Review*, 73(3):338–359, 1964.

[6] E. J. Lemmon. *Beginning Logic*. Thomas Nelson and Sons, 1965.

[7] T. Nipkow. A Tutorial Introduction to Structured Isar Proofs. 2011.

[8] L. C. Paulson. *ML for the Working Programmer*. Cambridge University Press, 1996.

[9] B. Russell. On Denoting. *Mind*, 14(56):479–493, 1905.

[10] R. M. Smullyan. *What Is the Name of This Book?: The Riddle of Dracula and Other Logical Puzzles*. Prentice-Hall, Englewood Cliffs, 1978.

[11] R. Stalnaker. Indicative conditionals. *Philosophia*, 5(3):269–286, 1975.

[12] M. M. Wenzel. *Isabelle/Isar—a Versatile Environment for Human-Readable Formal Proof Documents*. PhD Thesis, Technische Universität München, 2002.

[13] F. Wiedijk. *The Seventeen Provers of the World: Foreword by Dana S. Scott (Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.