Formal Proof

Yannis Kassios

February 20, 2009

A *formal proof* is not a natural language argument. It is a calculation that follows *precise rules*. This is the whole point of Formal Methods. Instead of using natural (informal) language to reason about program correctness, we use formal notation and proof. Formal notation and proof are rigorous, unambiguous and can be checked mechanically.

In this course, *all proofs will be formal*, unless otherwise specified. Therefore, it is essential to learn the basics of formal proofs. We start by proving small boring theorems of binary theory. Later on, we will move to more interesting theories, but it is essential that we get this formal proof thing right first.

After discussing the basics (Sect. 1), we will move on to the proofs of special forms of expressions (Sect. 2). In Sect. 3 we discuss the advanced techniques of *monotonicity* and *context*.

1 Basics

So how do we prove a binary expression, say A? A common way to go is to write a big series of equations

$$A = B = C = ... = \top$$

such that, each individual equation (i.e. $A = B$, $B = C$ and so forth) is an "obvious" theorem (more on what "obvious" means, later on in Sect. 1.1). By the transitivity of equality, this proves that $A = \top$ which classifies A as a theorem. Of course, we could also go the other way, beginning with $\top$ and working our way to A.

More generally, we can prove that $\top \Rightarrow A$ (or $A \Leftarrow \top$). This proves that A is *at least as true as* $\top$. Since $\top$ is always true, this makes A always true, which classifies it again as a theorem.

We are thus allowed to use not only equations, but also implications in our big proving formula, e.g.:

$$A = B \Leftarrow C = D \Leftarrow ... \Leftarrow \top$$

The transitivity properties of both equality and implication mean that the above proves $A \Leftarrow \top$, which proves A. Again, each individual step must be an "obvious" theorem.

To prove A, we also have the (dual) option of disproving $\neg A$. This can be done by proving $\neg A \Rightarrow \bot$ in a similar manner.

1.1 What is "obvious"

So what is an "obvious" step in a proof? It basically means that it is a direct instantiation of one of the binary laws at the back of the book (see the examples in the lectures and the book). You may also prove your own lemmas and then use them in a proof as "obvious" steps.

Later on, when we start tackling real programming problems, this will turn out to be an overkill. In those later exercises, several very obvious steps may be skipped, as long as you convince the marker that you know what you are doing. But right now we are talking basic stuff, so you must be very detailed in your proofs.

You are allowed to use the transparency property of equality and substitute a sub-expression with an equal sub-expression, citing the relevant law. In that case, it helps if you underline the substituted expression, as in the following example:

$\underline{a \vee a} \Rightarrow b$ idempotence
$=$ $a \Rightarrow b$

Of course, you don't have to do that, if the sub-expression is obvious. But in complicated expressions, it helps the marker understand what you are doing without frustrating him or her; we don't want that, do we?

1.2 Common Pitfalls

It is noteworthy to remember not to fall into any of the following traps:

- Starting from $\top$ and strengthening instead of weakening, i.e.
  $\top \Leftarrow Z \Leftarrow Y ... \Leftarrow A$ WRONG!
  or starting from A and weakening instead of strengthening, i.e.
  $A \Rightarrow Z \Rightarrow Y ... \Rightarrow \top$ WRONG!
  This proves that A implies $\top$, which we already know is a theorem, but does not prove anything about A. Remember, if you want to prove A, you should weaken $\top$ to A or strengthen A to $\top$.

- Mixing $\Rightarrow$ and $\Leftarrow$. Don't do that. All implications must go the same way. Otherwise, you don't prove much:
  $A \Leftarrow B \Rightarrow C$
  means that B implies A and that B implies C, but it says nothing about the relation between A and C.

- A lot of people think of proofs as a sequence of expressions which are not connected. This is not correct. A proof is one long binary expression. The connectors between the various parts of the proof must be there. They are very important. If you forget the connectors, not only you are making a seemingly insignificant formal mistake in your syntax, you also run the more serious risk of producing a wrong proof. This is because the connectors actually tell you which direction your implications are going. As I've noted above, the direction is very important. Here is an example of a "proof" gone bad:

$(a \wedge \top) \vee b$	specialization	WRONG!
$\top \vee b$	base	WRONG!
$\top$		WRONG!

  The connectors are missing, which means that the proof is wrong anyway. But it is not a wrong proof of a theorem. It is a wrong proof of a non-theorem, which is a serious mistake. What went wrong? Here is what we see when we use connectors:

	$(a \wedge \top) \vee b$	specialization
$\Rightarrow$	$\top \vee b$	base
$=$	$\top$	

  We see that this is not a proof for $(a \wedge \top) \vee b$, because it uses the wrong direction in the implication (see the point above).

2 Proving special forms

If A is of some special form, we don't have to reduce it to $\top$.

If A is an equality, we can prove it by starting from the left side of the equality and working our way to the right side (or we can start from the right and move to the left). Remember, in this case, we are only permitted to use equality in each step; implication won't do.

If A is an implication, we can start from one of the operands and move to the other. We can use equality and implication in each step, but the direction of the implication must be the same as the direction of the implication in A. For example, the solution of 6(m) can be done as follows:

	$a \Rightarrow \neg a$	material implication
$=$	$\neg a \vee \neg a$	idempotence
$=$	$\neg a$	reflexivity
$\Rightarrow$	$\neg a$	

If A is a conjunction, we can prove each conjunct separately.

3 Monotonicity and Context

And, finally, we will talk about the advanced techniques of *(anti-)monotonicity* and *context*. Please learn these techniques very well. Not only they are very helpful in proving theorems, they are also a source of common errors if not properly applied.

3.1 (Anti-)monotonicity

If a sub-expression is in a *monotonic* context, you are allowed to substitute it for a weaker (stronger) sub-expression and weaken (strengthen) the whole expression. If a sub-expression is in an *anti-monotonic* context, you are allowed to substitute it for a weaker (stronger) sub-expression and strengthen (weaken) the whole expression. Again, it helps if you underline the relevant expression.

For example, let us prove $\neg(a \Rightarrow b) \Rightarrow a$ using anti-monotonicity.

	$\neg(a \Rightarrow \underline{b})$	antimonoticity, base
$\Rightarrow$	$\neg(\underline{a \Rightarrow \bot})$	indirect proof
$=$	$\neg\neg a$	double negation
$=$	a	

What we did in the first step is substitute b with $\bot$. By Base Law, we know that b is weaker than $\bot$ (i.e. $b \Leftarrow \bot$ is a theorem) and because b is in an anti-monotonic position, this substitution strengthens the whole expression (the direction of the implication is now $\Rightarrow$).

3.1.1 Common Pitfalls

The commonest problems with the use of monotonicity are as follows:

• Ignoring whether a sub-expression is in a monotonic or anti-monotonic context. Monotonic contexts preserve the direction of the implication, while anti-monotonic contexts reverse it. If you forget that, then you might end up making a mistake such as this:

	$\underline{a} \Rightarrow b$	base: $a \Rightarrow \top$-WRONG!
$\Rightarrow$	$\top \Rightarrow b$	identity
$=$	b	

Here, we used $a \Rightarrow \top$ to substitute a with $\top$. But we disregarded the context of a and the overall direction of the implication is wrong. Since a is in an anti-monotonic context, the direction of the overall implication should have been reversed, like this:

	$\underline{a} \Rightarrow b$	base: $a \Rightarrow \top$
$\Leftarrow$	$\top \Rightarrow b$	identity
$=$	b	

• Disregarding the fact that some expressions are in neither monotonic nor anti-monotonic context. This means that no substitution can be made based on an implication law. For example, the following doesn't work:

	$a = b$	base: $a \Rightarrow \top$-WRONG!
$\Rightarrow$	$\top = b$	identity
$=$	b	

Here, the sub-expression a is neither in a monotonic context nor in an antimonotonic one. So we are not allowed to use an implication law such as $a \Rightarrow \top$ to substitute it in the expression.

Both problems mentioned here are very likely to occur if one forgets the connectors between the lines of a proof, as mentioned in Sect. 1.2.

3.2 The Context Rule

The second advanced rule that you can use is the context rule. There are many context rules listed on p.11 of the book. Here let us use a context rule for implication and redo 6(m):

	$a \Rightarrow \underline{\neg a}$	context (assume a is a theorem in the underlined expression)
$=$	$a \Rightarrow \neg\top$	binary axiom
$=$	$a \Rightarrow \bot$	indirect proof
$=$	$\neg a$	

In the first step, we assumed that a is a theorem while transforming the underlined sub-expression. The context of the underlined sub-expression allows us to do that. The context rule is very powerful and therefore very useful later on in proofs of program correctness.

3.2.1 Common Pitfalls

Don't forget one basic thing about the context rule: you are allowed to transform *one of the operands* at a time. Not both of them. Transforming both operands using the context rule is *wrong*. For example, in $a \wedge a$, you are allowed to use the context rule to transform the expression to $a \wedge \top$ or to $\top \wedge a$. But you *cannot* use it simultaneously to both operands, transforming the expression to $\top \wedge \top$.

Some people use the context rule totally inappropriately in situations in which no context rule can apply. There are context rules for conjunctions, disjunctions and implications but there is no context rule for equality. Thus, the following is wrong:

	$a = \underline{a}$	context rule (assume a in the underlined expression) - WRONG!
$=$	$a = \top$	identity
$=$	a	

There is no context rule for $a = b$ that justifies changing b using a as a theorem, so we cannot do that.

4 Key Points

A Formal Proof is not a natural language argument but a mathematical calculation. This calculation takes the form of a (usually big) binary expression.

To prove an expression A, we weaken $\top$ to A or strengthen A to $\top$. This means that the proof is of the form $A \Leftarrow ... \Leftarrow \top \text{ or } \top \Rightarrow ... \Rightarrow A$. The direction of the implication is very important and cannot be reversed.

A proof is a single binary expression and not a sequence of expressions. Connectors between the separate lines of the proof are important and should never be forgotten.

Special Forms of Expressions can be proved in different ways. To prove an equality $A = B$ you may start from A and end in B (or from B to A) following only equation steps. To prove an implication $A \Rightarrow B$, you may start from A and end in B following implication steps (or from B to A following reverse implication steps). To prove a conjunction $A \wedge B$, you may prove A and B separately.

Monotonicity is important when substituting sub-expressions. Substituting a sub-expression a for a weaker one makes the whole expression weaker if a is in a monotonic context and stronger if a is in an anti-monotonic context.

Some sub-expressions are in neither a monotonic nor an anti-monotonic context. A typical example is the operands of an equality. We are not allowed to substitute such sub-expressions with weaker or stronger ones in a proof.

The most powerful proof technique is the context rule. It involves transforming part of an expression assuming another part as a theorem. Various context rules are included in the textbook. Care must be taken to ensure that a valid context rule applies when we are substituting a sub-expression using context.