0 Proof Techniques

28 January 2020

Sebastian Wild

version 2020-01-28 14:13

Outline

0 Proof Techniques

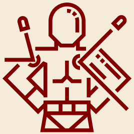
What is a *formal* proof?

A formal proof (in a logical system) is a **sequence of statements** such that each statement

1. is an *axiom* (of the logical system), or
2. follows from previous statements using the *inference rules* (of the logical system).

Among experts: Suffices to *convince a human* that a formal proof *exists*.

But: Use formal logic as guidance against faulty reasoning. ⇝ bulletproof



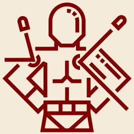
What is a *formal* proof?

A formal proof (in a logical system) is a **sequence of statements** such that each statement

1. is an *axiom* (of the logical system), or
2. follows from previous statements using the *inference rules* (of the logical system).

Among experts: Suffices to *convince a human* that a formal proof *exists*.

But: Use formal logic as guidance against faulty reasoning. $\rightsquigarrow$ bulletproof



Notation:

$A \Leftrightarrow B \equiv A \Rightarrow B \land B \Rightarrow A$

- Statements: $A \equiv$ "it rains", $B \equiv$ "the street is wet".
- Negation: $\neg A$ "Not A."
- And/Or: $A \land B$ "A and B"; $A \lor B$ "A or B or both."
- Implication: $A \Rightarrow B$ "If A, then B."
- Equivalence: $A \Leftrightarrow B$ "A holds true *if and only if ('iff')* B holds true."

Clicker Question

Is the following statement true?

"If the Earth is flat, then ships can fall over its rim."

A Yes

B No

C Neither

pingo.upb.de/622222

Clicker Question

Is the following statement true? $A \Rightarrow B \equiv \neg A \vee B$

"If the Earth is flat, then ships can fall over its rim."

A Yes ✓ **B** ~~No~~ **C** ~~Neither~~

`pingo.upb.de/622222`

0.1 Proof Templates

Implications

To prove $A \Rightarrow B$, we can

- directly derive B from A *direct proof*
- prove $(\neg B) \Rightarrow (\neg A)$ *indirect proof, proof by contraposition*
- assume $A \wedge \neg B$ and derive a contradiction *proof by contradiction, reductio ad absurdum*
- distinguish cases, i. e., separately prove $(A \wedge C) \Rightarrow B$ and $(A \wedge \neg C) \Rightarrow B$. *proof by exhaustive case distinction*

$B \vee \neg A \equiv \neg\neg B \vee \neg\neg(\neg A)$

$\equiv \neg(\neg B) \vee \neg(\neg\neg A)$

This should read:
A implies B == B or not A
== not not B or not A
== not A or not (not B)
== (not B) implies (not A)

Clicker Question

Suppose we want to prove:
"If n^2 is an even number, then n is also even."
For that we show that when n is odd, also n^2 is odd.
Which proof template do we follow?

- **A** direct proof: $A \Rightarrow B$
- **B** indirect proof: $(\neg B) \Rightarrow (\neg A)$
- **C** proof by contradiction: $A \land \neg B \Rightarrow \lightning$
- **D** proof by case distinction: $(A \land C) \Rightarrow B$ and $(A \land \neg C) \Rightarrow B$

pingo.upb.de/622222

Clicker Question

Suppose we want to prove:
"If n^2 is an even number, then n is also even."
For that we show that when n is odd, also n^2 is odd.
Which proof template do we follow?

- **A** ~~direct proof: $A \Rightarrow B$~~
- **B** indirect proof: $(\neg B) \Rightarrow (\neg A)$
- **C** ~~proof by contradiction: $A \wedge \neg B \Rightarrow \lightning$~~
- **D** ~~proof by case distinction: $(A \wedge C) \Rightarrow B$ and $(A \wedge \neg C) \Rightarrow B$~~

`pingo.upb.de/622222`

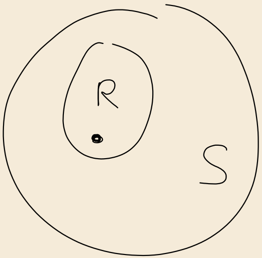
Equivalences

To prove $A \Leftrightarrow B$,
we prove both implications $A \Rightarrow B$ and $B \Rightarrow A$ separately.

(Often, one direction is much easier than the other.)

Set Inclusion and Equality

To prove that a set S contains a set R, i. e., $R \subseteq S$, we prove the implication $x \in R \Rightarrow x \in S$.



To prove that two sets S and R are equal, $S = R$, we prove both inclusions, $S \subseteq R$ and $R \subseteq S$ separately.

0.2 Mathematical Induction

Quantified Statements

Notation

- Statements with parameters: $A(x) \equiv$ "x is an even number."
- Existential quantifiers: $\exists x : A(x)$ "There exists some x, so that $A(x)$."
- Universal quantifiers: $\forall x : A(x)$ "For all x it holds that $A(x)$."

  Note: $\forall x : A(x)$ is equivalent to $\neg\exists x : \neg A(x)$

Quantifiers can be nested, e. g., *ε-δ-criterion for limits*:

$$\lim_{x \to \xi} f(x) = a \quad :\Leftrightarrow \quad \forall \varepsilon > 0\ \exists \delta > 0 : \big(|x - \xi| < \delta\big) \Rightarrow \big|f(x) - a\big| < \varepsilon.$$

To prove $\exists x : A(x)$, we simply list an example ξ such that $A(\xi)$ is true.

For-all statements

To prove $\forall x : A(x)$, we can

- derive $A(x)$ for an *"arbitrary but fixed value of* x*"*, or,
- for $x \in \mathbb{N}_0$, use *induction*, i. e.,
  - prove $A(0)$, *induction basis*, and
  - prove $\forall n \in \mathbb{N}_0 : A(n) \Rightarrow A(n+1)$ *inductive step*

More general variants of induction:

- complete/strong induction
  inductive step shows $\big(A(0) \wedge \cdots \wedge A(n)\big) \Rightarrow A(n+1)$
- structural/transfinite induction
  works on any *well-ordered* set, e. g., binary trees, graphs, Boolean formulas, strings, . . .
  no infinite strictly decreasing chains

0.3 Correctness Proofs

Formal verification

- verification: prove that a program computes the correct result

⇝ **not** our focus in COMP 526
but some techniques are useful for *reasoning* about algorithms

Here:

1. Prove that loop or recursive call eventually *terminates*.
2. Prove that a *loop* computes the *correct* result.

Proving termination

To prove that a recursive procedure $\texttt{proc}(x_1, \ldots, x_m)$ eventually terminates, we

- define a *potential* $\Phi(x_1, \ldots x_m) \in \mathbb{N}_0$ of the parameters
  (Note: $\Phi(x_1, \ldots x_m) \geq 0$ by definition!)
- prove that every recursive call decreases the potential, i. e.,
  any recursive call $\texttt{proc}(y_1, \ldots, y_m)$ inside $\texttt{proc}(x_1, \ldots, x_m)$ satisfies

$$\Phi(y_1, \ldots, y_m) \ < \ \Phi(x_1, \ldots, x_m)$$

⇝ $\texttt{proc}(x_1, \ldots, x_m)$ terminates because
we can only strictly *decrease* the (integral!) potential a *finite* number of times from its initial value

- Can use same idea for a loop: show that potential decreases in each iteration.
  - ⇝ see tutorials for an example.

Loop invariants

Goal: Prove that a *post condition* holds after execution of a (terminating) loop.

```
// (A) before loop
while cond do
    // (B) before body
    body
    // (C) after body
end while
// (D) after loop
```

For that, we

- find a *loop invariant* I (that's the tough part!)
- prove that I holds at (A)
- prove that $I \wedge cond$ at (B) imply I at (C)
- prove that $I \wedge \neg cond$ imply the desired post condition at (D)

Note: I holds before, during, and after the loop execution, hence the name.

Loop invariant – Example

- loop condition: $cond \equiv i < n$
- post condition (after line 9):
  $curMax = \max_{k\in[0..n-1]} A[k]$
- loop invariant:
  $I \equiv curMax = \max_{k\in[0..i-1]} A[k] \wedge i \leq n$

```
1  procedure arrayMax(A,n)
2      // input: array of n elements, n ≥ 1
3      // output: the maximum element in A[0..n − 1]
4      curMax := A[0];  i = 1
5      // (A)
6      while i < n do
7          // (B)
8          if A[i] > curMax
9              curMax := A[i]
10         i := i + 1
11         // (C)
12     end while
13     // (D)
14     return curMax
```

We have to proof:

(i) I holds at (A) $i = 1$ $I = curMax = A[0]$ ✓

(ii) $I \wedge cond$ at (B) $\Rightarrow$ I at (C)

(ii) ∘ $A[i] > curMax = \max_{k\in[0..i-1]} A[k]$
$\Rightarrow curMax = A[i]$ at 9.5
$\Rightarrow curMax = \max_{k\in[0..i-1]} A[k] = A[i]$

∘ $A[i] \leq curMax$

(iii) $I \wedge \neg cond$ $\Rightarrow$ post condition

I at (D) $\wedge$ $i \geq n$ $\Rightarrow$
$curMax = \max_{k\in[0..i-1]} A[k] \wedge i \leq n \wedge i \geq n$ $\Rightarrow i = n$ $\Rightarrow$ $curMax = \max_{k\in[0..n-1]} A[k]$