0 Proof Techniques

10 February 2021

Sebastian Wild

version `2021-02-10 01:11`

Outline

0 Proof Techniques

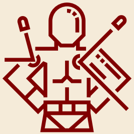
What is a *formal* proof?

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. $\rightsquigarrow$ bulletproof



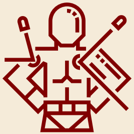
What is a *formal* proof?

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:

- Statements: $A \equiv$ "it rains", $B \equiv$ "the street is wet".
- Negation: $\neg A$ "Not A." "It does not rain".
- And/Or: $A \wedge B$ "A and B"; $A \vee B$ "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."

$\boxed{A \Leftrightarrow B \;:\Leftrightarrow\; A \Rightarrow B \wedge B \Rightarrow A}$

XOR

$A \oplus B$

either A or B

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

`sli.do/comp526`

Click on "Polls" tab

Clicker Question

$A \Rightarrow B \;:\Leftrightarrow\; \neg A \vee B$

Is the following statement true?

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

A $\quad\quad\quad\quad$ B

A Yes ✓

B ~~No~~

C ~~Neither~~

$A \Rightarrow B \qquad \neg A \vee \cancel{B} \qquad$ true

sli.do/comp526

Click on "Polls" tab

0.1 Proof Templates

Implications

To prove $A \Rightarrow B$, we can

A = input to program is valid (a list of numbers)

B = output of program is correct (the list is sorted)

- directly derive B from A *direct proof* (obvious)
- 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*

  e.g. $\sqrt{2}$ is irrational
- distinguish cases, i. e., separately prove $(A \wedge C) \Rightarrow B$ and $(A \wedge \neg C) \Rightarrow B$. *proof by exhaustive case distinction*

more than 2 cases possible

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

n odd $\leadsto$ write n as $n = 2k+1$ $\quad k \in \mathbb{N}$

$$\leadsto n^2 = (2k+1)^2 = \underbrace{4k^2+4k}_{\text{even}} + 1 = 2k'+1 \quad k' \in \mathbb{N}$$

`sli.do/comp526`

Click on "Polls" tab

Clicker Question

Suppose we want to prove:

"If n^2 is an even number, then n is also even."

A B

For that we show that when n is odd, also n^2 is odd.

Which proof template do we follow?

$\neg B$ $\neg A$ not even $\equiv$ odd

A ~~direct proof: $A \Rightarrow B$~~

B indirect proof: $(\neg B) \Rightarrow (\neg A)$ ✓

C ~~proof by contradiction: $A \wedge \neg B \Rightarrow ↯$~~

D ~~proof by case distinction: $(A \wedge C) \Rightarrow B$ and $(A \wedge \neg C) \Rightarrow B$~~

sli.do/comp526

Click on "Polls" tab

Equivalences

if and only if

To prove $A \Leftrightarrow B$, A iff 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 $\leq$

To prove that a set S contains a set R, i. e., $R \subseteq S$, R is subset of 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: $\underline{A(x)} \equiv$ "x is an even number."
- Existential quantifiers: $\underline{\exists x : A(x)}$ "There exists some x, so that $A(x)$."
- Universal quantifiers: $\underline{\forall x} : A(x)$ "For all x it holds that $A(x)$." $\forall x \in \mathbb{R}_{\geq 0} : A(x)$

  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*:

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

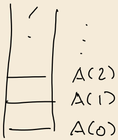
$\boxed{\delta \text{ can depend on } \varepsilon}$

To prove $\underline{\exists x : A(x)}$, 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., $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$
  - 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

— continued —

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

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

$$\leq \Phi(x_1, \ldots, x_m) - 1$$

⇝ $\texttt{proc}(x_1, \ldots, x_m)$ 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~~ 13):
  $$curMax = \max_{k \in [0..n-1]} A[k]$$
- loop invariant:
  $$I \equiv \underbrace{curMax = \max_{k \in [0..i-1]} A[k]}\ \wedge\ \underbrace{i \leq n}$$

```
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) ✓

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

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

(i) here (at (A)) have $i = 1$

$\rightsquigarrow I \equiv \underline{curMax = A[0]} \wedge \underline{1 \leq n}$ ✓

$[0..i-1] = \{0, \ldots i-1\} = \{0\}$

(ii) case distinction

(a) $A[i] > curMax \overset{I}{=} \max_{k \in [0..i-1]} A[k]$

then go to line 9. $curMax = A[i]$ after line 9

after line 10 $curMax = \max_{k \in [0 \ldots i-1]} A[k] = A[i-1]$ ✓

$$\left.\begin{array}{l} cond \equiv i < n \iff i \leq n-1 \\ \qquad\quad i+1 \leq n \end{array}\right\} \text{ at (B)}$$

$\rightsquigarrow$ at (C) have $i \leq n$ ✓

(iii) $\neg cond \equiv i \geq n$ $\quad i \leq n$ $\quad \rightsquigarrow$ $i = n$

$curMax = \max_{k \in [0..n-1]} A[k]$ $\longleftarrow$ post condition.

$\square$