0 Proof Techniques

3 February 2022

Sebastian Wild

COMP526 (Spring 2022)
University of Liverpool

version `2022-02-03 12:05`

Learning Outcomes

1. Know logical *proof strategies* for proving implications, set inclusions, set equalities, and quantified statements.
2. Be able to use *mathematical induction* in simple proofs.
3. Know techniques for *proving termination* and *correctness* of procedures.

Unit 0: ***Proof Techniques***

Outline

0 Proof Techniques

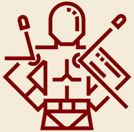
What is a *formal* proof?

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



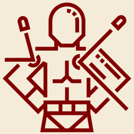
What is a *formal* proof?

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". either A or B A XOR B
- Negation: $\neg A$ "Not A"
- And/Or: $A \wedge B$ "A and B"; $A \vee B$ "A or B or both"
- Implication: $A \Rightarrow B$ "If A, then B"; $\neg A \vee B$
- Equivalence: $A \Leftrightarrow B$ "A holds true *if and only if ('iff')* B holds true."; $(A \Rightarrow B) \wedge (B \Rightarrow A)$

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`

Clicker Question

"$A \Rightarrow B$" $\equiv$ "$B \vee \neg A$"

Is the following statement true?

"If the $\underbrace{\text{Earth is flat}}_{A}$, then $\underbrace{\text{ships can fall over its rim.}}_{B}$"

A Yes

B ~~No~~

C ~~Neither~~

sli.do/comp526

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*

Clicker Question

Need the additional assumption here that n is an integer

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 $\Rightarrow$ $n = 2k+1 \quad k \in \mathbb{Z}$

$$n^2 = (2k+1)^2 = 4k^2+2k+1 = 2\underbrace{(2k^2+k)}_{=k' \in \mathbb{Z}}+1$$

$\Rightarrow n^2$ odd $\square$.

sli.do/comp526

Clicker Question

Suppose we want to prove: $A \Rightarrow B$

“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? $\neg B \Rightarrow \neg A$

A ~~direct proof:~~ $\cancel{A \Rightarrow B}$

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

C ~~proof by contradiction:~~ $\cancel{A \wedge \neg B \Rightarrow \lightning}$

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

sli.do/comp526

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*:

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

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

Clicker Question

Have you seen **proofs by *mathematical induction*** before?

- **A** Yes, could do it
- **B** Yes, but only vaguely remember
- **C** I’ve heard this term before, but ...
- **D** I have not heard “mathematical induction” before

`sli.do/comp526`

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*

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\begin{array}{c} A(2) \\ \Uparrow \\ A(1) \\ \Uparrow \\ A(0) \end{array}$$

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

continue 13:52

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.

correct

Proving termination

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

$\mathbb{N}_0 = \{0, 1, 2, \ldots\}$

- 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!)

$\texttt{proc}(x_1, \ldots, x_m):$
$\vdots$
$\texttt{proc}(y_1, \ldots, y_m)$
$\vdots$

- 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

$$\Phi(y_1, \ldots, y_m) < \Phi(x_1, \ldots, x_m) \qquad \text{which means}$$

$$\Phi(y_1, \ldots, y_m) \leq \Phi(x_1, \ldots, x_m) \mathbf{- 1}$$

$\rightsquigarrow$ $\texttt{proc}(x_1, \ldots, x_m)$ 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.
  - $\rightsquigarrow$ 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

precondition ≡ given A an array of $n \geq 1$ numbers

- loop condition: $cond \equiv i < n$
- post condition (in line 13):
  $curMax = \max_{k\in[0..n-1]} A[k]$
- 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]   ≡ postcondition
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 $i = 1$ and $curMax = A[0]$

$curMax = \max_{k\in[0..0]} A[k] = A[0]$ ✓

by precondition $n \geq 1$ $\quad i = 1 \leq n$ ✓

(ii) have I ∧ cond

Distinguish cases

(1) "$\max\limits_{k\in[0..i-1]} A[k] < A[i]$"

$I \Rightarrow curMax = \max\limits_{k\in[0..i-1]} A[k]$

$\Rightarrow curMax < A[i]$

$\Rightarrow$ enter line 9 and update

$curMax' := A[i]$

then update $i' := i+1$

$I \equiv \max\limits_{k\in[0..i'-1]} A[k] = curMax' \wedge i' \le n$ ✓

$\max\limits_{k\in[0..i'-1]} A[k] \equiv \max\limits_{k\in[0..i]} A[k] = A[i]$ ✓

$curMax' \equiv A[i]$

$i < n$

$\Rightarrow i' \le n$

(2) "$\max\limits_{k\in[0..i-1]} A[k] \ge A[i]$"

$\Rightarrow$ by I $curMax = \max\limits_{k\in[0..i-1]} A[k]$

$\Rightarrow curMax \ge A[i]$

$\Rightarrow curMax' = curMax$

then $i' := i+1$

$\max\limits_{k\in[0..i'-1]} A[k] = \max\limits_{k\in[0..i-1]} A[k]$

$= curMax'$ ✓

same as in case (1)

(iii) $\quad curMax = \max\limits_{k \in [0, \dots i-1]} A[k] \wedge \underline{i \le n \wedge i \geq n}$

$\Rightarrow i = n$

$i = n$

$$\Rightarrow curMax = \max_{k \in [0 \dots n-1]} A[k]$$

$\square$