COSC 221 - Introduction to Discrete Structures

Lecture - Proof Techniques (I)

Readings

- ▷ Sections 4.1, 4.3, 5.1 - 5.3
- ▷ Computer Science Connections
  1. Computer Generated Proofs (Section 4.3)
  2. The Cost of Missing Proofs (Section 4.5)
  3. Loop Invariants (Section 5.2)
  4. Page 415 Regular Expressions (Section 8.3)

Importance of Proofs?

- Establish Truth
- Solve Problems

- Establish Truth
- Solve Problems

Proof Techniques

- Establish Truth
- Solve Problems

Programmers also Need Proofs

```
int[ ] magicSort(int[ ] array){
   return array;
}
```

Claim: magicSort() works!

Proof Techniques

- Establish Truth
- Solve Problems

Programmers also Need Proofs

```
int[ ] magicSort(int[ ] array){
   return array;
}
```

Claim: magicSort() works!

Not this kind !!!

- “It is not correct!”
- “No, try [1, 2, 3]”
- “It is so wrong!!”
- “How?”
- “It is definitely wrong!!!”
- “Because #?&@!!!!”

Proof Techniques

Importance of Proofs?

- Establish Truth
- Solve Problems

Pure Logic Proof

Prove EVEN(k^2) Given: EVEN(k)

(for a particular even number $k > 0$.)

$\forall n \in Z^+, \text{EVEN}(n) \Rightarrow (\exists\ m \in Z^+ \text{ such that } n = 2m).$
$\therefore\ \text{EVEN}(k) \Rightarrow (\exists\ m \in Z^+ \text{ such that } k = 2m).$

$\text{EVEN}(k) \rightarrow (\exists\ m \in Z^+ \text{ such that } k = 2m).$
$\text{EVEN}(k)$
$\therefore\ k = 2m \text{ for some } m \in Z^+.$

$\forall n \in N, (n = 2m \text{ for some } m \in N) \rightarrow \text{EVEN}(n).$
$k^2 = 4m^2 = 2 * (2m^2)$
$\therefore\ \text{EVEN}(k^2).$

A logically-sound argument

Programmers also Need Proofs

```
int[] magicSort(int[] array){
   return array;
}
```

Claim: magicSort() works!

Not this kind !!!

- "It is not correct!"
- "No, try [1, 2, 3]"
- "It is so wrong!!"
- "How?"
- "It is definitely wrong!!!"
- "Because #?&@!!!!"

Proof Techniques

Importance of Proofs?

- Establish Truth
- Solve Problems

Pure Logic Proof

Prove EVEN(k^2) Given: EVEN(k)

(for a particular even number $k > 0$.)

$$\forall n \in Z^+, \text{EVEN}(n) \;\Rightarrow\; (\exists\, m \in Z^+ \text{ such that } n = 2m).$$
$$\therefore\; \text{EVEN}(k) \;\Rightarrow\; (\exists\, m \in Z^+ \text{ such that } k = 2m).$$

$$\text{EVEN}(k) \;\rightarrow\; (\exists\, m \in Z^+ \text{ such that } k = 2m).$$
$$\text{EVEN(k)}$$
$$\therefore\; k = 2m \text{ for some } m \in Z^+.$$

Proofs and Good Proofs: Convincing argument that are

- Easy to Read
- Easy to Understand

▷ Establish Truth

▷ Solve Problems

Pure Logic Proof

Prove EVEN(k^2) Given: EVEN(k)

(for a particular even number $k > 0$.)

$\forall n \in Z^+, \text{EVEN}(n) \Rightarrow (\exists\, m \in Z^+ \text{ such that } n = 2m).$
$\therefore\ \text{EVEN}(k) \Rightarrow (\exists\, m \in Z^+ \text{ such that } k = 2m).$

$\text{EVEN}(k) \rightarrow (\exists\, m \in Z^+ \text{ such that } k = 2m).$
EVEN(k)
$\therefore\ k = 2m \text{ for some } m \in Z^+.$

Proofs and Good Proofs: Convincing argument that are

▷ Easy to Read

▷ Easy to Understand

Good Proof = Well-Written Essay

▷ Being Concise

▷ Enough Details

Logic statements + Statements in English

Proof Techniques

Importance of Proofs?

- ▷ Establish Truth
- ▷ Solve Problems

Problem-Solving Strategy

1. Conjecture/Assumption
2. Algorithms/Design/Models
3. Proofs. Yes — Happy!
4. No — Go to Step 1

Proofs and Good Proofs: Convincing argument that are

- ▷ Easy to Read
- ▷ Easy to Understand

Good Proof = Well-Written Essay

- ▷ Being Concise
- ▷ Enough Details

Logic statements + Statements in English

Proof Techniques

Importance of Proofs?

- ▷ Establish Truth
- ▷ Solve Problems

Proofs Help Understand Properties of

- Algorithmic Problems
- Computing Systems
- Mathematical Systems

Problem-Solving Strategy

1. Conjecture/Assumption
2. Algorithms/Design/Models
3. Proofs. Yes — Happy!
4. No — Go to Step 1

Proofs and Good Proofs: Convincing argument that are

- ▷ Easy to Read
- ▷ Easy to Understand

Good Proof = Well-Written Essay

- ▷ Being Concise
- ▷ Enough Details

Logic statements + Statements in English

Mathematics and the Axiomatic Method [a]

- From axioms to all possible true statements (Theorems).

E.g., Number Theory, Topology, Theory of Probability

[a] Invented by Euclid (300 BC):

Proof Techniques

- Regular Expressions
- Limitation of Computing

Proof Techniques

- Regular Expressions
- Limitation of Computing

Proof Techniques

Proof in Computing: From Proofs to Software

- Regular Expressions
- Limitation of Computing

(See Page 8-40, Textbook)

A pattern describing a collection of strings

E.g. $((10^*1) \cup 0)^*10^*$

Supported in most programming languages.

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

(zero or more 'a' followed by a single 'b')

- Regular Expressions
- Limitation of Computing

(See Page 8-40, Textbook)

A pattern describing a collection of strings

E.g. $((10^*1) \cup 0)^*10^*$

Supported in most programming languages.

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

(zero or more 'a' followed by a single 'b')

Parser/Engine

- How to design a parser?

Proof Techniques

- ▷ Regular Expressions
- ▷ Limitation of Computing

(See Page 8-40, Textbook)

A pattern describing a collection of strings

E.g. $((10^*1) \cup 0)^*10^*$

Parser/Engine

- ▷ How to design a parser?

Supported in most programming languages.

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

(zero or more ‘a’ followed by a single ‘b’)

Theorem in Theory of Computation

For any set S of strings, the following are equivalent

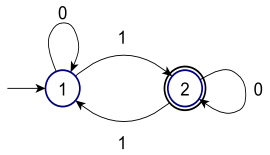
- ▷ S can be described by a regular expression
- ▷ S can be “recognized” by a **finite-state machine**

Applications

- Compiler/Programming Languages/
- Natural Language Processing
- Computational Biology

Supported in most programming languages.

- ▷ Regular Expressions
- ▷ Limitation of Computing



Model of Computing Devices with Limited Memory

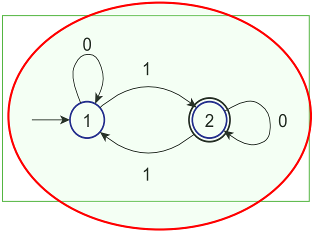
- ▷ Read input, one char at a time
- ▷ Change state based on transition rules
- ▷ "Yes" if finishing in "Accept"

Theorem in Theory of Computation

For any set S of strings, the following are equivalent

- ▷ S can be described by a regular expression
- ▷ S can be "recognized" by a **finite-state machine**

- Regular Expressions
- Limitation of Computing



Accepts:
"1", "10101", and even crazier

(Earth) 110100001 $\cdots$ 0100111 (Moon)

Memory Usage: one bit for two states!!!

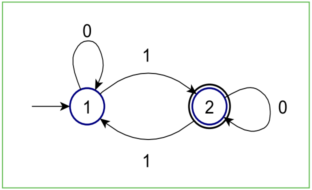
Theorem in Theory of Computation

For any set S of strings, the following are equivalent

- S can be described by a regular expression
- S can be "recognized" by a **finite-state machine**

Proof Techniques

- Regular Expressions
- Limitation of Computing



Accepts:
"1", "10101", and even crazier

(Earth) 110100001 $\cdots$ 0100111 (Moon)

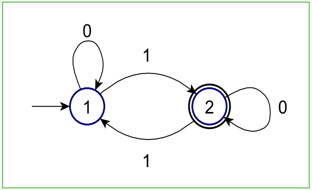
Memory Usage: one bit for two states!!!

Q.1) What does this cute machine accept?

A) Any binary string
B) Binary strings ending with "1"
C) Binary string containing odd number of 1's
D) Binary strings ending with "0"

Proof Techniques

- Regular Expressions
- Limitation of Computing



Accepts:
"1", "10101", and even crazier

(Earth) 110100001 $\cdots$ 0100111 (Moon)

Memory Usage: one bit for two states!!!

Q.2) Number of bits needed to encode a state?

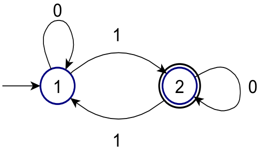
A) 1

B) 2

C) 4

D) $2^{\text{String Length}}$

- Regular Expressions
- Limitation of Computing



Accepts:
"1", "10101", and even crazier

(Earth) 110100001 $\cdots$ 0100111 (Moon)

Memory Usage: one bit for two states!!!

Q.2) Number of bits needed to encode a state?

A) 1

B) 2

C) 4

D) $2^{\text{String Length}}$

Dream algorithm for data streams

- Regular Expressions
- Limitation of Computing

Significance of the Proof:

- Limitation of devices with Fixed Amount of Memory
- Automatic way to design software parsers (back on this later on)

Theorem in Theory of Computation

For any set S of strings, the following are equivalent

- S can be described by a regular expression
- S can be "recognized" by a **finite-state machine**

Proof Techniques

- Regular Expressions
- Limitation of Computing

Existence of NP-Complete Problems

- S. Cook's Proof on Satisfiability Problem
- Turing Award (1982)

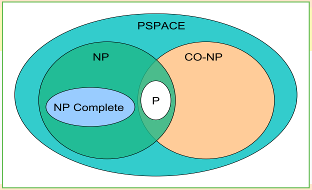
Proof Techniques

- ▷ Regular Expressions
- ▷ Limitation of Computing

(Roughly) Hardest among all problems whose solution can be verified efficiently on modern computers (Turing machines)

Existence of NP-Complete Problems

- ▷ S. Cook's Proof on Satisfiability Problem
- ▷ Turing Award (1982)



Proof Techniques

Proof in Computing: From Proofs to Software

- Regular Expressions
- Limitation of Computing

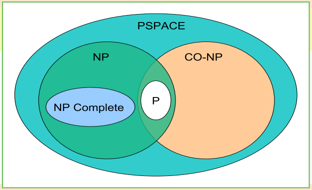
Theorem (S. Cook (1971), UoT)

SAT is NP-Complete

(Roughly) Hardest among all problems whose solution can be verified efficiently on modern computers (Turing machines)

Existence of NP-Complete Problems

- S. Cook's Proof on Satisfiability Problem
- Turing Award (1982)

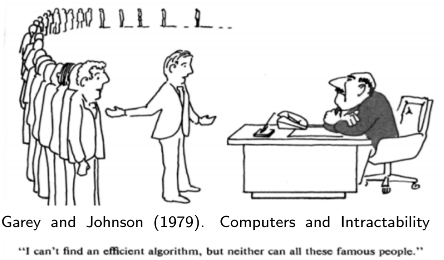


Proof Techniques

- ▷ Regular Expressions
- ▷ Limitation of Computing

Theorem (S. Cook (1971), UoT)

SAT is NP-Complete



Garey and Johnson (1979). Computers and Intractability

Existence of NP-Complete Problems

- ▷ S. Cook's Proof on Satisfiability Problem
- ▷ Turing Award (1982)

- Regular Expressions
- Limitation of Computing

Existence of NP-Complete Problems

- S. Cook's Proof on Satisfiability Problem
- Turing Award (1982)

- Proof of existence
- Proof by counterexample

Proof Techniques

- Proof of existence
- Proof by counterexample

$\exists x \in D$ such that $P(x)$

- Constructive
- Non-Constructive

Proof Techniques

- ▷ Proof of existence
- ▷ Proof by counterexample

$\exists x \in D$ such that $P(x)$

- ▷ Constructive
- ▷ Non-Constructive

Show/describe how to find such x

Example: Union of Regular Expressions

If two sets of strings, S_1 and S_2, can both be "recognized" by some finite-state machine, then $S_1 \cup S_2$ can be recognized by a finite-state machine.

Proof Techniques

- Proof of existence
- Proof by counterexample

$\exists x \in D$ such that $P(x)$
- Constructive
- Non-Constructive

Proof Idea:

Buy two machines
- M_1 for S_1
- M_2 for S_2

Assemble them!

Show/describe how to find such x

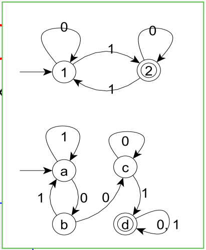
Example: Union of Regular Expressions

If two sets of strings, S_1 and S_2, can both be "recognized" by some finite-state machine, then $S_1 \cup S_2$ can be recognized by a finite-state machine.

Proof Techniques

Basic Proof Methods (I)

- Proof of existence
- Proof by counterex



$\exists x \in D$ such that $P(x)$

- Constructive
- Non-Constructive

Proof Idea:

Buy two machines

- M_1 for S_1
- M_2 for S_2

Assemble them!

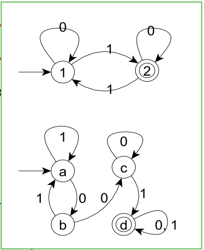
Example: Union of Regular Expressions

If two sets of strings, S_1 and S_2, can both be "recognized" by some finite-state machine, then $S_1 \cup S_2$ can be recognized by a finite-state machine.

Proof Techniques

Basic Proof Methods (I)

- Proof of existence
- Proof by counterex



$\exists x \in D$ such that $P(x)$

- Constructive
- Non-Constructive

Proof Idea:

Buy two machines

- M_1 for S_1
- M_2 for S_2

Assemble them!

Difficulty: Must have one input

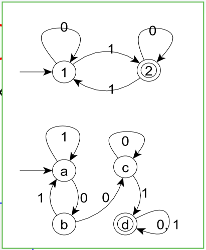
Example: Union of Regular Expressions

If two sets of strings, S_1 and S_2, can both be "recognized" by some finite-state machine, then $S_1 \cup S_2$ can be recognized by a finite-state machine.

Proof Techniques

Basic Proof Methods (I)

- Proof of existence
- Proof by counterex



$\exists x \in D$ such that $P(x)$

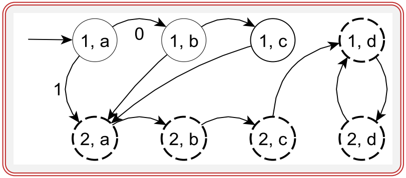
- Constructive
- Non-Constructive

Proof Idea:

Buy two machines

- M_1 for S_1
- M_2 for S_2

Assemble them!



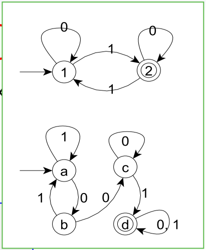
Example: Union of Regular Expressions

If two sets of strings, S_1 and S_2, can both be "recognized" by some finite-state machine, then $S_1 \cup S_2$ can be recognized by a finite-state machine.

Proof Techniques

Basic Proof Methods (I)

- ▷ Proof of existence
- ▷ Proof by counterex



$\exists x \in D$ such that $P(x)$

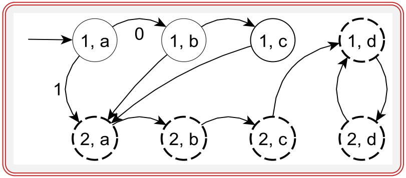
- ▷ Constructive
- ▷ Non-Constructive

Proof Idea:

Buy two machines

- ▷ M_1 for S_1
- ▷ M_2 for S_2

Assemble them!



Q.3) Do these two pictures make a Proof?

A) Yes

B) No

Example: Union of Regular Expression

If two sets of strings, S_1 and S_2, can b
machine, then $S_1 \cup S_2$ can be recogni

Proof Techniques

- Proof of existence
- Proof by counterexample

$\exists x \in D$ such that $P(x)$

- Constructive
- Non-Constructive

Show that the existence is logically guaranteed

Proof Techniques

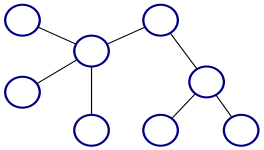
- Proof of existence
- Proof by counterexample

$\exists x \in D$ such that $P(x)$

- Constructive
- Non-Constructive

Show that the existence is logically guaranteed

Example: A tree has at least one leaf



- Tree — connected graph with no cycle
- Leaf — node with a single neighbor

Proof Techniques

- Proof of existence
- Proof by counterexample

$\exists x \in D$ such that $P(x)$

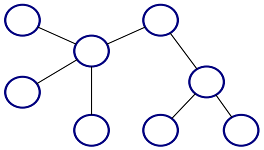
- Constructive
- Non-Constructive

Show that the existence is logically guaranteed

The two ends of the longest path!

There must be a path that is the longest!!!

Example: A tree has at least one leaf



- Tree — connected graph with no cycle
- Leaf — node with a single neighbor

- Proof of existence
- Proof by counterexample

Disproof of Universal Statements

- $\forall x \in D, P(x)$ or
- $\forall x \in D, P(x) \rightarrow Q(x)$

- Proof of existence
- Proof by counterexample

Disproof of Universal Statements

- $\forall x \in D, P(x)$ or
- $\forall x \in D, P(x) \rightarrow Q(x)$

Find an x^*, counterexample, such that $P(x^*)$ (or $P(x^*) \rightarrow Q(x^*)$) is false.

Proof Techniques

- ▷ Proof of existence
- ▷ Proof by counterexample

Yong' Claim: magicSort() works!

```
int[] magicSort(int[] array){
   return array;
}
```

Disproof of Universal Statements

- ▷ $\forall x \in D, P(x)$ or
- ▷ $\forall x \in D, P(x) \rightarrow Q(x)$

Find an x^*, counterexample, such that $P(x^*)$ (or $P(x^*) \rightarrow Q(x^*)$) is false.

- Proof of existence
- Proof by counterexample

Claim: If a graph has four or more nodes, then it is two-colorable

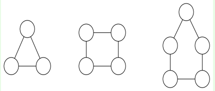
Q.4) Which one is a counterexample?

A) First

B) Second

C) Third

D) All of them



Disproof of Universal Statements

- $\forall x \in D, P(x)$ or
- $\forall x \in D, P(x) \to Q(x)$

Find an x^*, counterexample, such that $P(x^*)$ (or $P(x^*) \to Q(x^*)$) is false.

Linear Search

```
int lSearch(int[] values, int value){
  int numOfSteps = 0;
  boolean found = false;
  while( !found && numOfSteps < values.size){
    numOfSteps++;
    if(values[numOfSteps] == value) found = true;
  }
  return numOfSteps;
}
```

Disproof of Universal Statements

- ▷ $\forall x \in D, P(x)$ or
- ▷ $\forall x \in D, P(x) \to Q(x)$

Find an x^*, counterexample, such that $P(x^*)$ (or $P(x^*) \to Q(x^*)$) is false.

Linear Search

```
int lSearch(int[] values, int value){
  int numOfSteps = 0;
  boolean found = false;
  while( !found && numOfSteps < values.size){
    numOfSteps++;
    if(values[numO
  }
  return numOfStep
}
```

Q.5) Which one is a counterexample?

A) value $= 2$

B) value $= 4$

C) value $= 8$

D) value $= 9$

values: list of even integers

Claim: The while-loop terminates in less than $\text{values.size}/2$ iterations.

Disproof of Universal Statements

- ▷ $\forall x \in D, P(x)$ or
- ▷ $\forall x \in D, P(x) \to Q(x)$

Find an x^*, counterexample, such that $P(x^*)$ (or $P(x^*) \to Q(x^*)$) is false.

Additional Notes

Announcement

Assignment 2 Posted. Due on Fri, March 3rd

Additional Notes

Perfect Square: Every natural number is a perfect square

$\forall n \in Z^+$, n is a perfect square.

Counterexample: 5 is not a perfect square.

Euler's sum of powers conjecture of order 4

$\forall a, b, c, d \in Z^+, a^4 + b^4 + c^4 \neq d^4$.

Smallest counterexample found in 1988:

- a = 95800, b = 217519, c = 414560, d = 422481.

- Generalizing from the generic particular
- Proof by cases

- Generalizing from the generic particular
- Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \ \wedge \ \text{EVEN}(m) \ \rightarrow \ \text{EVEN}(m+n)$$

- Generalizing from the generic particular
- Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \wedge \text{EVEN}(m) \to \text{EVEN}(m+n)$$

Q.6) Is the following a proof?

- $2 + 4 = 6$
- $100 + 256 = 356$
- $\cdots$

Therefore, the statement true.

A) Yes

B) No

- Generalizing from the generic particular
- Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \land \text{EVEN}(m) \to \text{EVEN}(m+n)$$

Consider a particular, but **arbitrary**, x

Template for writing a good proof

- ▷ Generalizing from the generic particular
- ▷ Proof by cases

Example

$\forall n, m \in Z, \quad \text{EVEN}(n) \wedge \text{EVEN}(m) \to \text{EVEN}(m+n)$

Proof.

Let m and n be any even numbers. We have to show that $m+n$ is even.

Proof Techniques

Basic Proof Methods (II): Proof of $\forall x \in D, P(x) \to Q(x)$

- Generalizing from the generic particular
- Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \wedge \text{EVEN}(m) \to \text{EVEN}(m+n)$$

Proof.

Let m and n be any even numbers. We have to show that $m + n$ is even.

Prologue:

- Title, Setting, and
- Articulating Your Plan

- Generalizing from the generic particular
- Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \land \text{EVEN}(m) \rightarrow \text{EVEN}(m+n)$$

Proof.

Let m and n be any even numbers. We have to show that $m+n$ is even.

- By definition, $m = 2r$ and $n = 2s$ for some $r, s \in Z$.
- Then, $m + n = 2r + 2s = 2(r + s)$.
- Because $r + s \in Z$, $m + n$ is even by definition of an even number.

Prologue:

- Title, Setting, and
- Articulating Your Plan

- Generalizing from the generic particular
- Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \wedge \text{EVEN}(m) \to \text{EVEN}(m+n)$$

Proof.

Let m and n be any even numbers. We have to show that $m+n$ is even.

- By definition, $m = 2r$ and $n = 2s$ for some $r, s \in Z$.
- Then, $m + n = 2r + 2s = 2(r + s)$.
- Because $r + s \in Z$, $m + n$ is even by definition of an even number.

Prologue:

- Title, Setting, and
- Articulating Your Plan

Body: Logic Arguments

- Generalizing from the generic particular
- Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \wedge \text{EVEN}(m) \rightarrow \text{EVEN}(m+n)$$

Proof.

Let m and n be any even numbers. We have to show that $m+n$ is even.

- By definition, $m = 2r$ and $n = 2s$ for some $r, s \in Z$.
- Then, $m + n = 2r + 2s = 2(r + s)$.
- Because $r + s \in Z$, $m + n$ is even by definition of an even number.

Prologue:

- Title, Setting, and
- Articulating Your Plan

Body: Logic Arguments

Existential Instantiation:

If the existence of an object is guaranteed, we can give it a name

- ▷ Generalizing from the generic particular
- ▷ Proof by cases

Example

$$\forall n, m \in Z, \quad \text{EVEN}(n) \land \text{EVEN}(m) \rightarrow \text{EVEN}(m+n)$$

Proof.

Let m and n be any even numbers. We have to show that $m+n$ is even.

- By definition, $m = 2r$ and $n = 2s$ for some $r, s \in Z$.
- Then, $m + n = 2r + 2s = 2(r + s)$.
- Because $r + s \in Z$, $m + n$ is even by definition of an even number.

Q.E.D

Prologue:

- ▷ Title, Setting, and
- ▷ Articulating Your Plan

Body: Logic Arguments

Proof Techniques

- Generalizing from the generic particular
- Proof by cases

Example

$\forall n, m \in Z, \quad \text{EVEN}(n) \wedge \text{EVEN}(m) \rightarrow \text{EVEN}(m+n)$

Proof.

Let m and n be any even numbers. We have to show that $m+n$ is even.

- By definition, $m = 2r$ and $n = 2s$ for some $r, s \in Z$.
- Then, $m + n = 2r + 2s = 2(r + s)$.
- Because $r + s \in Z$, $m + n$ is even by definition of an even number.

Q.E.D

Prologue:

- Title, Setting, and
- Articulating Your Plan

Epilogue: Concluding

- Initials of Latin phrase meaning "this is what we need to show".
- Alternatively, we put a black square at the end of the last line.

- ▷ Generalizing from the generic particular
- ▷ Proof by cases

Example

$\forall n, m \in Z, \quad \text{EVEN}(n) \wedge \text{EVEN}(m) \rightarrow \text{EVEN}(m+n)$

Proof.

Let m and n be any even numbers. We have to show that $m+n$ is even.

- By definition, $m = 2r$ and $n = 2s$ f some $r, s \in Z$.
- Then, $m + n = 2r + 2s = 2(r+s)$
- Because $r + s \in Z$, $m + n$ is even definition of an even number.

Q.E.D

Three Parts of a Well-Constructed Proof

- ▷ Prologue: Setting and Strategy
- ▷ Body: Logic Arguments
- ▷ Epilogue: Concluding

- Generalizing from the generic particular
- Proof by cases

Triangle Inequality

$\forall$ x and $y \in R$, $|x+y| \leq |x| + |y|$

(Simpler Version of Example 4.15 (Page 427,Textbook))

The absolute value of the sum of two real numbers is no greater than the sum of their absolute values

Proof Techniques

Basic Proof Methods (II): Proof of $\forall x \in D, P(x) \rightarrow Q(x)$

- Generalizing from the generic particular
- Proof by cases

Triangle Inequality

$\forall$ x and $y \in R$, $|x+y| \leq |x| + |y|$

(Simpler Version of Example 4.15 (Page 427,Textbook)

Generalizing from generic particular not enough

Proof.

Let x and y be two real numbers.

Proof Techniques

- Generalizing from the generic particular
- Proof by cases

Triangle Inequality

$\forall$ x and $y \in R$, $|x+y| \leq |x|+|y|$

(Simpler Version of Example 4.15 (Page 427,Textbook)

Proof.

Let x and y be two real numbers.

Consider two cases in terms of the sign of $x+y$.

Case 1 $(x+y \geq 0)$ $|x+y| = x+y \leq |x|+|y|$.

Case 2 $(x+y < 0)$ $|x+y| = -(x+y) \leq |x|+|y|$.

Therefore, $|x+y| \leq |x|+|y|$. ■

- Generalizing from the generic particular
- Proof by cases

Triangle Inequality

$\forall$ x and $y \in R$, $|x+y| \leq |x| + |y|$

(Simpler Version of Example 4.15 (Page 427,Textbook))

Proof.

Let x and y be two real numbers.

Consider two cases in terms of the sign of $x+y$.

Case 1 $(x+y \geq 0)$ $|x+y| = x+y \leq |x|+|y|$.

Case 2 $(x+y < 0)$ $|x+y| = -(x+y) \leq |x|+|y|$.

Therefore, $|x+y| \leq |x|+|y|$. ■

Logic Foundation

$$\begin{array}{lccc} & p & \vee & q \\ & p & \rightarrow & r \\ & q & \rightarrow & r \\ \therefore & r & & \end{array}$$

- Generalizing from the generic particular
- Proof by cases

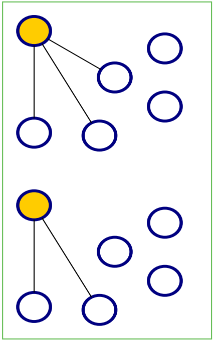
Strangers and Clubs

In a group of 6 people, there is either a club of 3 people **or** a group of 3 strangers

- Generalizing from the generic particular
- Proof by cases

Strangers and Clubs

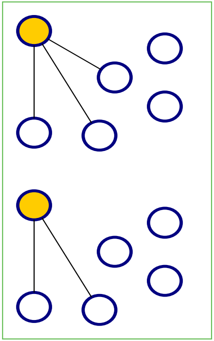
In a group of 6 people, there is either a club of 3 people **or** a group of 3 strangers



- Generalizing from the generic particular
- Proof by cases

Strangers and Clubs

In a group of 6 people, there is either a club of 3 people **or** a group of 3 strangers



Proof.

We use the method of proof by cases. Let x be one of the six people and consider two cases:

Case 1: x has at least 3 friends.

Case 2: x has at most 2 friends.

... ∎

References I