Module #3 - Functions

University of Florida
Dept. of Computer & Information Science & Engineering

COT 3100
Applications of Discrete Structures
Dr. Michael P. Frank

Slides for a Course Based on the Text
Discrete Mathematics & Its Applications
(5th Edition)
by Kenneth H. Rosen

2004/3/2 (c)2001-2002, Michael P. Frank

Module #10:
Proof Strategies

Rosen 5th ed., §3.1
~28 slides, ~1 lecture

2004/3/2 (c)2001-2002, Michael P. Frank
Module #3 - Functions

Overview of Section 3.1

- Methods of mathematical argument (proof methods) can be formalized in terms of rules of logical inference.
- Mathematical proofs can themselves be represented formally as discrete structures.
- We will review both correct & fallacious inference rules, & several proof methods.

Module #3 - Functions

Applications of Proofs

- An exercise in clear communication of logical arguments in any area.
- The fundamental activity of mathematics is the discovery and elucidation of proofs of interesting new theorems.
- Theorem-proving has applications in program verification, computer security, automated reasoning systems, etc.
Module #3 - Functions

Proof Terminology

- **Theorem** - A statement that has been proven to be true.
- **Axioms, postulates, hypotheses, premises** - Assumptions (often unproven) defining the structures about which we are reasoning.
- **Rules of inference** - Patterns of logically valid deductions from hypotheses to conclusions.

More Proof Terminology

- **Lemma** - A minor theorem used as a stepping-stone to proving a major theorem.
- **Corollary** - A minor theorem proved as an easy consequence of a major theorem.
- **Conjecture** - A statement whose truth value has not been proven. (A conjecture may be widely believed to be true, regardless.)
Module #3 - Functions

Inference Rules - General Form

- *Inference Rule* - Pattern establishing that if we know that a set of *antecedent* statements of certain forms are all true, then a certain related *consequent* statement is true.

- *antecedent 1*  
  *antecedent 2*  ...  

  \[ \therefore \text{ consequent} \]

  "\therefore" means "therefore"

Inference Rules & Implications

- Each logical inference rule corresponds to an implication that is a tautology.

- *antecedent 1*  
  *antecedent 2*  ...  

  \[ \therefore \text{ consequent} \]

- Corresponding tautology: \(((\text{antecedent 1}) \land (\text{antecedent 2}) \land \ldots) \rightarrow \text{consequent}\)
Module #3 - Functions

Some Inference Rules

- \[ p \quad \text{Rule of Addition} \]
  \[ \therefore p \lor q \]

- \[ p \land q \quad \text{Rule of Simplification} \]
  \[ \therefore p \]

- \[ p \quad \text{Rule of Conjunction} \]
  \[ q \]
  \[ \therefore p \land q \]

Module #3 - Functions

Modus Ponens & Tollens

- \[ p \quad \text{Rule of } modus \ ponens \]
  \[ p \rightarrow q \]
  \[ \therefore q \]

- \[ \neg q \quad \text{Rule of } modus \ tollens \]
  \[ p \rightarrow q \]
  \[ \therefore \neg p \]
Module #3 - Functions

Syllogism Inference Rules

- \( p \rightarrow q \) \( \rightarrow \) Rule of hypothetical
  \( q \rightarrow r \) \( \rightarrow \) syllogism
  \( \therefore p \rightarrow r \)

- \( p \lor q \) \( \lor \) Rule of disjunctive
  \( \neg p \) \( \lor \) syllogism
  \( \therefore q \)

Module #3 - Functions

Formal Proofs

- A formal proof of a conclusion \( C \), given
  premises \( p_1, p_2, \ldots, p_n \) consists of a sequence
  of \( steps \), each of which applies some
  inference to premises or previously-proven
  statements (as antecedents) to yield a new
  true statement (the consequent).

- A proof demonstrates that if the premises
  are true, then the conclusion is true.
Module #3 - Functions

Formal Proof Example

• Premises:
  “It is not sunny and it is cold.”
  “We will swim only if it is sunny.”
  “If we do not swim, then we will canoe.”
  “If we canoe, then we will be home early.”

• Given these premises, prove “We will be home early” using inference rules.

Proof Example \textit{cont.}

• Let \(\text{sunny} = \text{“It is sunny”}\); \(\text{cold} = \text{“It is cold”}\);
  \(\text{swim} = \text{“We will swim”}\); \(\text{canoe} = \text{“We will canoe”}\);
  \(\text{early} = \text{“We will be home early”}\).

• Premises:
  (1) \(\neg \text{sunny} \land \text{cold}\)
  (2) \(\text{swim} \rightarrow \text{sunny}\)
  (3) \(\neg \text{swim} \rightarrow \text{canoe}\)
  (4) \(\text{canoe} \rightarrow \text{early}\)
Module #3 - Functions

Proof Example *cont.*

<table>
<thead>
<tr>
<th>Step</th>
<th>Proved by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ¬sunny ∧ cold</td>
<td>Premise #1.</td>
</tr>
<tr>
<td>2. ¬sunny</td>
<td>Simplification of 1.</td>
</tr>
<tr>
<td>3. swim→sunny</td>
<td>Premise #2.</td>
</tr>
<tr>
<td>4. ¬swim</td>
<td>Modus tollens on 2,3.</td>
</tr>
<tr>
<td>5. ¬swim→canoe</td>
<td>Premise #3.</td>
</tr>
<tr>
<td>6. canoe</td>
<td>Modus ponens on 4,5.</td>
</tr>
<tr>
<td>7. canoe→early</td>
<td>Premise #4.</td>
</tr>
<tr>
<td>8. early</td>
<td>Modus ponens on 6,7.</td>
</tr>
</tbody>
</table>

Common Fallacies

- A *fallacy* is an inference rule or other proof method that is not logically valid.
- Fallacy of **affirming the conclusion**: “p→q is true, and q is true, so p must be true.” (Consider F→T.)
- Fallacy of **denying the hypothesis**: “p→q is true, and p is false, so q must be false.” (Consider F→T.)
Module #3 - Functions

Circular Reasoning

- The fallacy of (explicitly or implicitly) assuming the very statement you are trying to prove in the course of its proof.
- Prove that an integer $n$ is even if $n^2$ is even.
- Attempted proof: “Assume $n^2$ is even. Then $n^2=2k$ for some integer $k$. Dividing both sides by $n$ gives $n=(2k)/n=2(k/n)$. So there is an integer $j$ (namely $k/n$) such that $n=2j$. Therefore $n$ is even.”

Begs the question: How do you show that $j=k/n=n/2$ is an integer, without assuming $n$ is even?

Module #3 - Functions

Removing the Circularity

Suppose $n^2$ is even $\therefore 2|n^2 \therefore n^2 \mod 2 = 0$. Of course $n \mod 2$ is either 0 or 1. If it’s 1, then $n \equiv 1 \pmod{2}$, so $n^2 \equiv 1 \pmod{2}$, using the theorem that if $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then $ac \equiv bd \pmod{m}$, with $a\equiv c \equiv n$ and $b\equiv d \equiv 1$. Now $n^2 \equiv 1 \pmod{2}$ implies that $n \mod 2 = 1$. So by the hypothetical syllogism rule, $(n \mod 2 = 1)$ implies $(n^2 \mod 2 = 1)$. Since we know $n^2 \mod 2 = 0 \neq 1$, by modus tollens we know that $n \mod 2 \neq 1$. So by disjunctive syllogism we have that $n \mod 2 = 0 \therefore 2|n \therefore n$ is even.
Module #3 - Functions

Inference Rules for Quantifiers

- **Universal instantiation**
  \[
  \forall x P(x) \quad \therefore P(o)
  \]
  (substitute any object \(o\))

- **Universal generalization**
  \[
  P(g) \quad \therefore \forall x P(x)
  \]
  (for \(g\) a general element of u.d.)

- **Existential instantiation**
  \[
  \exists x P(x) \quad \therefore P(c)
  \]
  (substitute a new constant \(c\))

- **Existential generalization**
  \[
  P(o) \quad \therefore \exists x P(x)
  \]
  (substitute any extant object \(o\))

Module #3 - Functions

Proof Methods

For proving implications \(p \rightarrow q\), we have:

- **Direct** proof: Assume \(p\) is true, and prove \(q\).
- **Indirect** proof: Assume \(\neg q\), and prove \(\neg p\).
- **Vacuous** proof: Prove \(\neg p\) by itself.
- **Trivial** proof: Prove \(q\) by itself.
- Proof by cases: Show \(p \rightarrow (a \lor b)\) and \((a \rightarrow q)\) and \((b \rightarrow q)\).
Proof by Contradiction

- Assume $\neg p$, and prove both $q$ and $\neg q$ for some proposition $q$.
- Thus $\neg p \rightarrow (q \land \neg q)$
- $(q \land \neg q)$ is a trivial contradiction, equal to $F$
- Thus $\neg p \rightarrow F$, which is only true if $\neg p = F$
- Thus $p$ is true.

Review: Proof Methods So Far

- Direct, indirect, vacuous, and trivial proofs of statements of the form $p \rightarrow q$.
- Proof by contradiction of any statements.
- Constructive and nonconstructive existence proofs.
Module #3 - Functions

Proving Existentials

- A proof of a statement of the form $\exists x P(x)$ is called an *existence proof*.
- If the proof demonstrates how to actually find or construct a specific element $a$ such that $P(a)$ is true, then it is a *constructive* proof.
- Otherwise, it is *nonconstructive*.

Module #3 - Functions

A Constructive Existence Proof

(Example 23, p.179)

- Show that for any $n > 0$ there exists a sequence of $n$ consecutive composite integers.
- Same statement in predicate logic:
  $\forall n > 0 \ \exists x \ \forall i \ (1 \leq i \leq n) \rightarrow (x+i \text{ is composite})$
Module #3 - Functions

The proof...

• Given \( n > 0 \), let \( x = (n + 1)! + 1 \).
• Let \( i \geq 1 \) and \( i \leq n \), and consider \( x + i \).
• Note \( x + i = (n + 1)! + (i + 1) \).
• Note \( (i+1)|(n+1)! \), since \( 2 \leq i+1 \leq n+1 \).
• Also \( (i+1)|(i+1) \). So, \( (i+1)|(x+i) \).
• \( \therefore x + i \) is composite.
• \( \therefore \forall n \exists x \forall 1 \leq i \leq n : x + i \) is composite. Q.E.D.

Module #3 - Functions

Nonconstructive Existence Proof

(Example 24, p. 180)
• Show that there are infinitely many primes.
• Show there is no largest prime.
• Show that for any prime number, there is a larger number that is also prime.
• Show that for any number, \( \exists \) a larger prime.
• Show that \( \forall n \exists p > n : p \) is prime.
Module #3 - Functions

Da proof...

- Given \( n > 0 \), prove there is a prime \( p > n \).
- Consider \( x = n! + 1 \). Since \( x > 1 \), we have \((x \text{ is prime}) \lor (x \text{ is composite})\).
- Case 1: \( x \) is prime. Obviously \( x > n \), so let \( p = x \) and we’re done.
- Case 2: \( x \) has a prime factor \( p \). But if \( p \leq n \), then \( p \mod x = 1 \). So \( p > n \), and we’re done.

Module #3 - Functions

The Halting Problem (Turing ‘36)

- Involves a \textit{non}-existence proof.
- The first mathematical function proven to have \textit{no} algorithm that computes it!
- The desired function is \( \text{Halts}(P, I) = \text{the truth value of the statement ‘Program } P \text{, given input } I \text{, eventually halts’} \).
- Implies general impossibility of predictive analysis of arbitrary computer programs.
Module #3 - Functions

The Proof

- Given any arbitrary program $H(P,I)$,
- Consider algorithm $Breaker$, defined as:
  
  ```
  procedure Breaker(P: a program)
  halts := H(P,P)
  if halts then while T begin end
  
  Note that $Breaker(Breaker)$ halts iff $H(Breaker,Breaker) = F$.
- So $H$ does not compute the function $Halts$!

Module #3 - Functions

Limits on Proofs

- Some very simple statements of number theory haven’t been proved or disproved!
  - E.g. Goldbach’s conjecture: Every integer $n>2$ is exactly the average of some two primes.
  - $\forall n>2 \exists$ primes $p,q : n=(p+q)/2$.
- There are true statements of number theory (or any sufficiently powerful system) that can never be proved (or disproved) (Gödel).