Mathematical Reasoning (Part 1)

1

Mathematical Reasoning terms

- Theorem = statement that can be shown to be true
- proof = sequence of statements that show that the theorem is true
- axiom, postulates = assumptions, hypothesis
- rules of inference = rules used to draw conclusion for each step of a proof
- fallacies = incorrect reasoning
- lemma = simple theorem (used in the proof of a bigger theorem)

2

Mathematical Reasoning terms (cont.)

- Corollary = a proposition that can be established directly from a theorem
- conjecture = statement that has unknown truth value
- an argument is valid if whenever all hypothesis are true, the conclusion is also true (p->q for example)
  - but a hypothesis can be false, i.e. valid does not mean correct

3

Rules of inference

- These are tautology that we can use in a proof
- Example: tautology $(p \wedge (p \to q)) \to q$

Modus ponens (law of detachment)

$$\begin{array}{l} p \\ \underline{p \to q} \\ \therefore q \end{array}$$

4

addition	$p \to (p \vee q)$
simplification	$(p \wedge q) \to p$
conjunction	$[(p) \wedge (q)] \to (p \wedge q)$
Modus ponens	$[p \wedge (p \to q)] \to q$
Modus tollens	$[\neg q \wedge (p \to q)] \to \neg p$
Hypothetical syllogism	$[(p \to q) \wedge (q \to r)] \to (p \to r)$
Disjunctive syllogism	$[(p \vee q) \wedge \neg p] \to q$

5

Example of logical proof

- Show that $sendmail \to finishwork$

$$\neg sendmail \to sleepearly$$

$$sleepearly \to wakeupgood$$

leads to the conclusion that

$$\neg finishwork \to wakeupgood$$

See the board :)

6

- Show that

$$\neg sunny \wedge cold$$
$$swim \rightarrow sunny$$
$$\neg swim \rightarrow canoe$$
$$canoe \rightarrow backearly$$

leads to the conclusion that

$$backearly$$

See the board :)

7

Rules of inference for quantifier

U instantiation	$\forall x P(x) \rightarrow P(c)$ if $c \in U$
U generalization	$P(c)$ for an arbitrary $c \in U$ $\rightarrow \forall x P(x)$
E instantiation	$\exists x P(x) \rightarrow P(c)$ for some element $c \in U$
E generalization	$P(c)$ for some element $c \in U$ $\rightarrow \exists x P(x)$

8

Rules of inference for quantifier (example)

- Show that

$$x \in discretemathclass$$
$$\forall x[comstudent(x)]$$
$$Don \in discretemathclass$$

leads to the conclusion that

$$comstudent(Don)$$

See the board :)

9

Method of proving theorem in the form p->q

- Direct proof
  - assume p true
  - show q is true
- Indirect proof
  - is a direct prove of $\neg q \rightarrow \neg p$
- Vacuous proof
  - show that p is false

10

Method of proving theorem in the form p->q (cont.)

- Trivial proof
  - show q is true
- Contradiction proof
  - assume the negation of we are proving is true, then find any contradicting statements
  - e.g. to prove p, we must assume $\neg p$ to be true

11

Method of proving theorem in the form p->q (cont2.)

- Proof by cases
  - to prove $(p_1 \vee p_2 \vee \ldots \vee p_n) \rightarrow q$, we can, instead, prove that $(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \wedge \ldots (p_n \rightarrow q)$
  - to prove $p \leftrightarrow q$, we can prove $(p \rightarrow q) \wedge (q \rightarrow p)$
  - to prove $p_1 \leftrightarrow p_2 \leftrightarrow \ldots \leftrightarrow p_n$, we can prove

$$(p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_3) \wedge \ldots \wedge (p_n \rightarrow p_1)$$

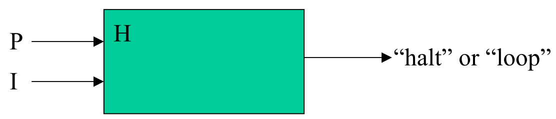
12

Proving theorem with quantifier

- To prove $\exists x P(x)$
  - just find an x that works = constructive proof
  - can prove by other means, such as contradiction
- To prove $\forall x P(x)$
  - usually it is proving that it is false
  - just show an x where P(x) is false = counter example

Important prove example: Halting problem

- There is no program that, given a program and an input, can determine if that program terminates
- Prove by contradiction:
- assume



- Let K be another machine such that
  - if H(P,P) halts, K(P) loops and vice versa
- Then we replace P by K for both machine H and K -> we see contradiction

Mathematical Reasoning (Part 2)

15

Mathematical Induction, proving $\forall x P(x)$

We do it by:

- show that the base case is true
- assume P(x) is true, and use this assumption to show that P(x+1) is true

Inductive hypothesis

16

Math. Induction, why does it work?!

- Assume we know the base case and P(x)->P(x+1) are true
- Let's assume that there is some x that makes P(x) false. That x must be a member of a set (let's say set S) which holds all x that P (x) is false.
  - Then there is a least element k of the set S
  - hence, k-1 is surely not in S and therefore P(k-1) must be true
  - by the first assumption, P(k) must also be true…. But this is contradiction.

17

Math induction example:

Prove: 1+3+5+… = n*n (increment for 2n-1 each time)

- Base case: n=1: yes, LHS =RHS
- Assume: …………..
- Show: …………..

See the board

18

Prove: $n < 2^n$

- Base case: $1 < 2^1$
- Assume: $n < 2^n$
- Show that: $(n+1) < 2^{(n+1)}$

See the board

Prove: $\sum_{j=0}^{n} ar^j = \frac{ar^{n+1} - a}{r - 1}$

- Base case: j=0
- Assume:………….
- Show that: ………….

See the board

Prove: $$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{2^n}\geq 1+\frac{n}{2}$$

- Base case: 1>=1+0/2
- Assume:………….
- Show that: $1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{2^{n+1}}\geq 1+\frac{n+1}{2}$

See the board

21

Prove: chessboard $2^n * 2^n$ with 1 square removed can be tiled using L-shape pieces

- Base case:
- Assume:
- Show that:

See the board

22

Second principle of Math Induction

- Base case
- Assume that P(next to base case) to P(n) are true
- show that P(n+1) is true
- Example: prove that any integer n (>1) is a product of prime
  - base case: P(2) yes, 2 is a product of prime
  - Assume: P(k) for all k<=n
  - show P(n+1)

See the board

23

Recursive (or inductive) definition

- Something that repeatedly call itself…but with smaller input, until the call ends at the base case
  - : $a_{n+1} = 2a_n$
  - : factorial f(n+1) = (n+1)f(n) , f(0) = 1
  - : fibonacci f(0) =0, f(1)=1, f(n) = f(n-1) + f(n-2)

24

Set with recursive definition

- Example:

$$3 \in S$$

$$x + y \in S$$ If x is in S and Y is also in S

- Prove: 3*n is in S

See the board

Well-formed formula (what's it got to do with recursion?)

- x is a well-formed formula if x is a numeral or a variable
- (f+g), (f-g), (f*g), (f/g), (f exp g) are well-formed formulae if f and g are.
  - X and 5 are well-formed, so is x+5
- T,F and p where p is a propositional variable, are well-formed
- if p and q are well-formed, so is

$$\neg p, p \wedge q, p \vee q, p \rightarrow q, p \leftrightarrow q$$

Well-known recursive algorithms

- Linear search: search(a,i,j,x) -
  - if a[i] = x then return 1
  - else if i=j then return 0
  - else search(a,i+1,j,x)
- Binary search: bisearch(a,i,j,x)
  - k = floor((i+j)/2)
  - if x=a[k] then return 1
  - else if (x<a[k] and i<k) then bisearch(a,i,k-1,x)
  - else if (x>a[k] and j>k) then bisearch(a,k+1,j,x)
  - else return 0

Recursion and iteration

- Fac(n) = n*fac(n-1) ->recursion
- x:=1; for(int k=1,k<=n,k++){x = x*k} -> iteration
- iteration needs less computation (see fibonacci)
  - recursion requires many additions
  - while iteration requires only few additions z:=x+y; x=y; y=z;

Mathematical Reasoning (Part 3)

29

Introduction to program verification

To prove that a program is correct

- show that the correct answer is obtained if the program terminates. (partial correctness) with respect to initial and final assertions
- show that the program always terminates

Initial assertion= properties of input

Final assertion = properties of output

$$p\{S\}q$$

To be correct-> whenever p is true, q must also be true. This notation represents the partial correctness

30

Rules about program correctness

$$[(p\{S_1\}q) \wedge (q\{S_2\}r)] \rightarrow p\{S_1;S_2\}r$$

$$[(p \wedge condition\{S\}q) \wedge (p \wedge \neg condition) \rightarrow q]$$
$$\rightarrow p\{if-condition, then-S\}q$$

$$[(p \wedge condition\{S_1\}q) \wedge (p \wedge \neg condition\{S_2\}q]$$
$$\rightarrow p\{if-condition, then-S_1, else-S_2\}q$$

31

Loop invariant

Loop invariant

$$(p \wedge condition\{S\}p) \rightarrow$$
$$p\{while-condition-S\}(\neg condition \wedge p)$$

32