CSE1007: Logical Fundamentals of Programming

Part III - Mathematical Induction

Scott Uk-Jin Lee

Department of Computer Science and Engineering
Hanyang University ERICA Campus

1^{st} Semester 2016

Keywords

- Inductive Definition
- Proof by Induction

Induction

- whenever P(x) is defined inductively,
  $\Rightarrow \forall x\ [P(x) \rightarrow Q(x)]$ can be proved using Proof by Induction
- Although with infinitely many instances, induction can justify a general conclusion on the basis of a finite number of proof steps
- proof by induction provides a much more powerful method of proof than ordinary general conditional proof

  e.g., mathematical induction on natural number

  $\forall x\ [\mathrm{NatNum}(x) \rightarrow Q(x)]$

Inductive Definition

Inductive Definition consists of :

1. **base clause**
   - specifies the basic elements of the defined set
2. **inductive clause** (one or more)
   - tells how to generate additional elements
3. **final clause**
   - tells that all elements are either basic or generated by the inductive clauses

Example: Inductive Definition of ‘ambig-wff’

Let primitive symbols A_1, A_2, ..., A_n be propositional letters

1. [base clause] each propositional letter is an ambig-wff
2. [inductive clause] if p is an ambig-wff, so is $\neg$p
3. [inductive clause] if p and q is ambig-wff,
   so are $p \wedge q$, $p \vee q$, $p \rightarrow q$, and $p \leftrightarrow q$
4. Nothing is ambig-wff unless it is generated by repeated application of 1, 2, and 3

- ambig-wffs defined above are ambiguous as there is no parenthesis

Inductive Proof: Case Study 1

Proposition: every ambig-wff contains at least one propositional letter

$$\forall p\ [(p \text{ is an ambig-wff}) \rightarrow Q(p)]$$

Proof: induction on the ambig-wff

basis all the propositional letters contains at least one propositional letter (consists of exactly one such letter)

induction (induction hypothesis) suppose p and q are ambig-wffs that each contains at least one propositional letter

- show that the new ambig-wff generated from clauses 2 and 3 will also contain at least one propositional letter
- $\neg$p contains all the propositional letters contained in p. So, contains at least one propositional letter
- p $\wedge$ q, p $\vee$ q, p $\rightarrow$ q, and p $\leftrightarrow$ q contain all the propositional letters contained in p and q. So, contains at least one propositional letter (actually two propositional letter)

- By induction, thus we conclude that all ambig-wffs contain at least one propositional letter
- now we can prove that $\neg\vee\rightarrow$ is not an ambig-wff

Inductive Proof: Case Study 2

Proposition: no ambig-wff has the symbol $\neg$ occurring immediately before one of the binary connectives: $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$

$$\forall p\ [(p \text{ is an ambig-wff}) \rightarrow Q(p)]$$

Proof: Let's prove this inductively! Any problem?

Stronger claim: no ambig-wff either begins with a binary connective, or end with $\neg$, or has $\neg$ immediately before a binary connective.
So let this stronger claim be Q' then $\forall p\ [Q'(p) \rightarrow Q(p)]$

New Proposition: $\forall p\ [(p \text{ is an ambig-wff}) \rightarrow Q'(p)]$

Proof: Let's prove this inductively! Any problem?

Note: when proofs by induction get stuck, stronger (more detailed) claim is required for the proof.

now we can prove that $A_1 \neg \rightarrow A_n$ is not an ambig-wff

Inductive Proof: Case Study 3

inductive definition: set *pal*

1. each letter in the alphabet (a, b, c, ..., z) is a *pal*
2. if a string α is a pal, so is the result of putting any letter of the alphabet both in front of and in back of α (e.g., $a\alpha a$, $b\alpha b$, $c\alpha c$, etc)
3. nothing is a *pal* unless it is generated by repeatedly applying 1 and 2

Inductive Proof: Case Study 3

Proposition: *pal* reads the same forwards and backwards,
in other words, every *pal* is a palindrome

Proof: induction on the set *pal*

basis any single letter reads the same forwards and backwards

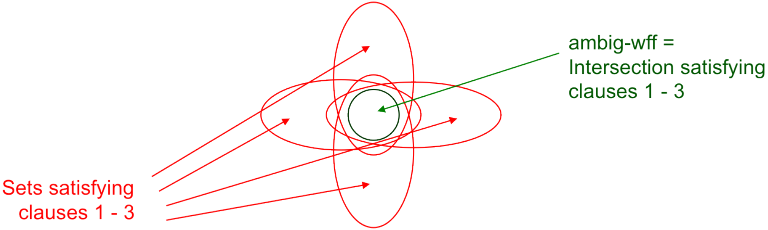
induction (induction hypothesis) suppose that the *pal* α reads the same forwards and backwards

- show that if you add a letter k to the beginning and end of α then the result $k\alpha k$ reads the same forwards and backwards
- when α' is the result of reversing α, reversing $k\alpha k$ gives $k\alpha$'k
  By inductive hypothesis $\alpha = \alpha$' and so the result of reversing $k\alpha k$ is $k\alpha k$ (reads the same forwards and backwards)

- By induction, thus we conclude that every *pal* is a palindrome

Inductive Definition in Set Theory

- The set S of ambig-wff is the smallest set satisfying the following clauses:
  1. each propositional letter is in S
  2. if p is in S, then so is $\neg$p
  3. if p and q are in S, then so are p $\wedge$ q, p $\vee$ q, p $\rightarrow$ q, p $\leftrightarrow$ q



Induction on the Natural Numbers

Inductive Definition: Natural Number (N^0)

1. 0 is a natural number
2. if n is a natural number, then n + 1 is a natural number
3. nothing is a natural number unless it is generated by repeatedly applying 1 and 2

Definition of natural number in set theory

- set N of natural number is the smallest set satisfying :

1. $0 \in N$
2. $n \in N$, then $x + 1 \in N$

Induction on N^0: Case Study

Proposition: for every natural number n, the sum of the first n natural number is n(n - 1)/2

$$\forall n \ (n \in p \rightarrow Q(n))$$

Proof: induction on the natural number N^0

basis sum of the first 0 natural number is 0

induction (induction hypothesis) suppose that there is a natural number k for which Q(k) holds

- show that Q(k+1) holds
- by inductive hypothesis, the sum of the first k natural number is $k(k - 1)/2$
- show that the sum of the first $k + 1$ natural number is $(k+1)(k + 1 - 1)/2 = (k+1)k/2$
- note that the sum of the first $k + 1$ natural number is k greater than the sum of the first k natural numbers

Case Study 1: Recursive Program

Requirement: write program that sums all natural numbers (from 0) up to the given natural number n

Program Specification

input : natural number n

output : $0 + 1 + 2 + ... + n$

Program

```
public natural sumToRec(natural n) {
   if (n == 0) return 0;
   else return n + sumToRec(n-1)
}
```

Does this program output the result as specified in the requirement?

Case Study 1: Recursive Program

Proof Goal: sumToRec(n) = 0 + 1 + 2 + ... + n

Proof:proof by induction

basis when the argument n = 0, program outputs the value 0 (sumToRec(0) = 0)

induction (induction hypothesis) suppose that the program outputs the correct value 0 + 1 + 2 + ... + k for the argument is k (sumToRec(k) = 0 + 1 + 2 + ... + k)

- show that the program outputs the correct value for the input k + 1
- since k + 1 is not 0, program is written to output (k + 1) + sumToRec(k)
- By induction hypothesis,
  sumToRec(k) + (k + 1) = (0 + 1 + 2 + ... + k) + (k + 1)

Case Study 2: Iterative Program

Program

```
public natural sumUpTo(natural n) {
    natural sum = 0;
    natural count = 0;
    while (count < n) {
        count += 1;
        sum += count;
    }
    return sum;
}
```

Proof Goal 1: when the input is n,
the body of while loop executes exactly n times

Proof Goal 2: when the input was n and program is terminated,
the value of the variable sum is the sum of
all natural numbers up to n

Case Study 2: Iterative Program

Proof Goal 1: when input is n, body of while loop executes exactly n times

Stronger Proof Goal 1: given any natural number k, once (k = n - count) and execution reaches the while loop condition, the loop executes exactly k times

Proof: induction on the value of n - count

basis Let n - count be 0 when execution reaches the loop condition. Then n = count and the condition becomes false. Since the body of loop no longer executes, when n - count is 0 the loop executes 0 times.

induction (induction hypothesis) When n - count is k, loop executes k times once the execution reaches the loop condition

- Show that the loop executes k + 1 times when n - count is k + 1. Let n - count = k + 1, then the loop condition becomes true (count < n) since n - count is positive number. Afterwards when the execution reaches the loop condition again, n - count = k. Then, by induction hypothesis, we know that the loop executes k times more. So, the body of loop executes k + 1 times in total.

Conclsion: when n is given as input, this program always terminates and executes the loop exactly n times

Proof Goal 2: when input is n and program terminates, the value of the variable sum is the sum of all natural numbers up to n

Stronger Proof Goal 2: after while loop has executed k times, the values of the variable sum and count is as below :

loop invariant : $sum = (0 + 1 + 2 + ... + k) \wedge count = k$

Proof: proof by induction

basis When $k = 0$ the while loop executes 0 times.
At this point $sum = 0$ and $count = 0$.

induction (induction hypothesis) After the body of the loop executes k times, the invariant holds.

- Show that the invariant still holds after the body of the loop executes k+1 times. Let sum be the value of sum after the loop executes k times, and let sum' be the value of sum after the loop executes k+1 times (count is represented similarly). The following formula also hold according to the hypothesis.

Case Study 2: Iterative Program

induction (continues)

sum = (0 + 1 + 2 + ... + count)

After the loop executes k+1 times, the value of count is incremented by 1 and the value of sum is incremented by count' (as the value of count is incremented first). Hence,

$$\begin{aligned} \text{sum}' &= \text{sum} + \text{count}' \\ &= \text{sum} + (\text{count} + 1) \\ &= 0 + 1 + 2 + \ldots + \text{count} + (\text{count} + 1) \\ &= 0 + 1 + 2 + \ldots + K + (K + 1) \end{aligned}$$

Conclusion: When the loop condition becomes false, we can see that count = n where sum = 0 + 1 + 2 + ... + n. This corresponds to the value that sum should output.

So. it satisfies our proof goal.