Lecture 1

Notorious Bugs – BYTE, September 1995

http://www.byte.com/art/9509/sec7/art20.htm

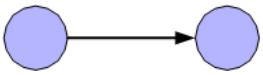
- 1987 : Therac-25 – The Bug that killed
- 1990: AT&T long distance break down
- 1991: Patriot Missile – Hitting own barracks, leaving 28 dead and 98 wounded

Program to test for two equal strings

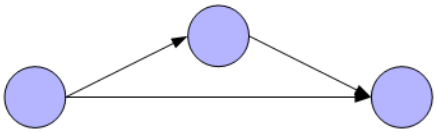
- Test cases:
- isEqual ("cat", "dog") - expected **false**
- isEqual ("Testing", "Testing") - expected **true**
- isEqual ("house", "home") - expected **false**

```
equal = strlen(string1) == strlen(string2);
if (equal)
    for (i = 0; i < strlen(string1); i++)
            equal = string1[i] == string2[i];
return equal;
```

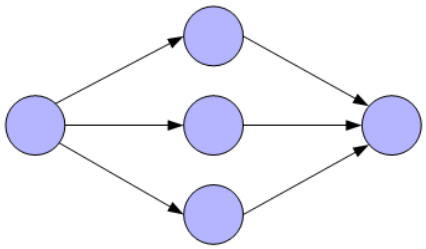
Some Notations



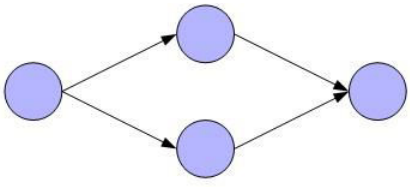
Sequence



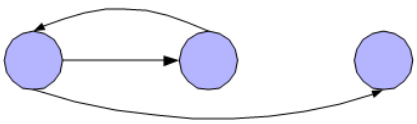
Selection – if statement



Selection – case statement



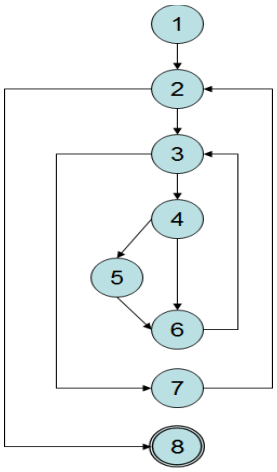
Selection – if-else statement



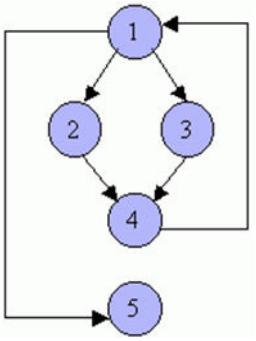
Loop

Flow graph for bubble sort

```
 sorted = false;                          // 1
 while (!sorted) {                        // 2
        sorted = true;
        for (int i = 0; i < SIZE-1; i++) {  // 3
              if (a[i] > a[i+1]) {          // 4
                swap(a[i], a[i+1]);         // 5
                sorted = false;
              }                             // 6
        }                                   // 7
 }                                          // 8
```

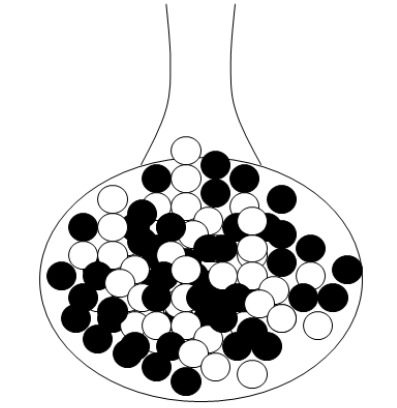



```
For (i=0; i<N; i++){                    // 1
If(condition1)
        // do something here            // 2
Else
        // do something here            // 3
//do something here                     // 4
}                                       // 5
```

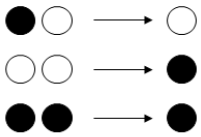


2^N Paths

Dijkstra's Game



Jar of Black and White Balls

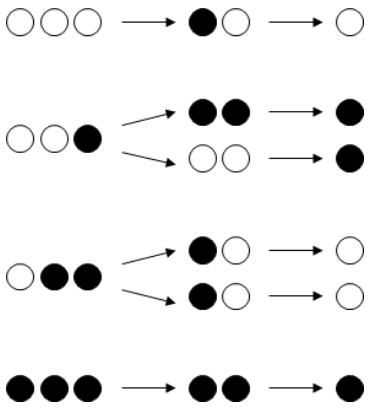


Rules of the Game

Heap of Black Balls

Lecture 2

A Three Ball Game



Mathematical Model

$$f(b,w) = \begin{cases} \text{(2 black out, 1 black in)} & b\text{-}2+1,\ w & \equiv\ b\text{-}1,\ w \\ \text{(2 white out, 1 black in)} & b+1,\ w\text{-}2 & \\ \text{(1 of each out, 1 white in)} & b\text{-}1,\ w\text{-}1+1 & \equiv\ b\text{-}1,\ w \end{cases}$$

- Total number of balls is reduced by exactly one in each move.
- Parity of the white ball does not change

Propositional Calculus

Implication

P	Q	P ⇒ Q
T	T	T
T	F	F
F	T	T
F	F	T

Bi-conditional – if and only if

P	Q	P ⇔ Q
T	T	T
T	F	F
F	F	T
F	T	F

P ⇔ Q means P ⇒ Q ∧ Q ⇒ P

- A compound proposition that is always true, irrespective of the truth values of the comprising propositions, is called a tautology.

  $p \vee \neg p$
- The propositions p and q are called logically equivalent if p ⇔ q is tautology.
- It is written as ,

  $p \equiv q$
- For example: $\neg (p \vee q) \equiv \neg p \wedge \neg q$

Some useful equivalence

- p or true ≡ true p or false ≡ p
- p and true ≡ p p and false ≡ false
- true ⇒ p ≡ p false ⇒ p ≡ true
- p ⇒ true ≡ true p ⇒ false ≡ not p
- p or p ≡ p p and p ≡ p
- not not p ≡ p
- p or not p ≡ true p and not p ≡ false

Distributivity of

- and over or
- or over and
- or over ⇒
- ⇒ over and
- ⇒ over or
- ⇒ over ⇒
- ⇒ over ⇔

Associativity of

- ∨, ∧, and ⇔

Commutativity of

- ∨, ∧, and ⇔

Demorgan's law

- Implication
- if and only if

TABLE 6 Logical Equivalences.

Equivalence	*Name*
$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity laws
$p \vee \mathbf{T} \equiv \mathbf{T}$ $p \wedge \mathbf{F} \equiv \mathbf{F}$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv \mathbf{T}$ $p \wedge \neg p \equiv \mathbf{F}$	Negation laws

TABLE 7 Logical Equivalences Involving Conditional Statements.

$p \to q \equiv \neg p \vee q$
$p \to q \equiv \neg q \to \neg p$
$p \vee q \equiv \neg p \to q$
$p \wedge q \equiv \neg(p \to \neg q)$
$\neg(p \to q) \equiv p \wedge \neg q$
$(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$
$(p \to r) \wedge (q \to r) \equiv (p \vee q) \to r$
$(p \to q) \vee (p \to r) \equiv p \to (q \vee r)$
$(p \to r) \vee (q \to r) \equiv (p \wedge q) \to r$

TABLE 8 Logical Equivalences Involving Biconditional Statements.

$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

Logic problem for the day

- Someone asks person A, "Are you a knight?" He replies, "If I am a knight then I'll eat my hat". Prove that A has to eat his hat.

Lecture 3

- A is a knight: A
- A eats his hat: H
- If I am a knight then I'll eat my hat: $A \Rightarrow H$
- We have seen that $(X \Leftrightarrow S)$ therefore $(A \Leftrightarrow A \Rightarrow H)$
- Objective is to logically deduce H

Truth Table Columns

A	H	$A \Rightarrow H$	$A \Leftrightarrow (A \Rightarrow H)$

Proof Using Truth Table

A	H	$A \Rightarrow H$
T	T	T
F	T	T
T	F	F
F	F	T

A	$A \Rightarrow H$	$A \Leftrightarrow (A \Rightarrow H)$
T	T	T
F	T	F
T	F	F
F	T	F

A	H	$A \Rightarrow H$	$A \Leftrightarrow (A \Rightarrow H)$
T	T	T	T
F	T	T	F
T	F	F	F
F	F	T	F

Proof using equivalences

A $\Leftrightarrow$ (A $\Rightarrow$ H)

≡ A $\Leftrightarrow$ (not A or H)

≡ (A and (not A or H)) or (not A and not (not A or H))

A and (not A or H)

≡ (A and not A) or (A and H)

≡ false or (A and H)

≡ A and H

not A and not (not A or H)

≡ not A and (A and not H)

≡ (not A and A) and not H

≡ false and not H

≡ false

Hence

A $\Leftrightarrow$ (not A or H)

≡ (A and H) or false

≡ A and H

Rules of Inference and Logical Deduction

- Introduction
- Elimination

Introduction

$$\frac{p \quad q}{p \text{ and } q}$$

Introduction

$$\frac{p}{p \text{ or } q} \qquad \frac{q}{p \text{ or } q}$$

Elimination

$$\frac{p \text{ and } q}{p} \qquad \frac{p \text{ and } q}{q}$$

Introduction

$$\frac{[p] \\ q}{p \Rightarrow q}$$

Elimination

$$\frac{p \quad p \Rightarrow q}{q}$$

Elimination

$$\frac{p \quad \text{not } p}{\text{false}} \qquad \frac{\text{false}}{p}$$

Does the Superman Exist?

If Superman were able and willing to prevent evil, he would do so. If Superman were unable to prevent evil, he would be incapable; if he were unwilling to prevent evil, he would be malevolent. Superman does not prevent evil. If Superman exists, he is neither incapable nor malevolent. Therefore Superman does not exist.

Inference and deduction

- Superman exists X
- Superman is willing to prevent evil W
- Superman is able to prevent evil A
- Superman is malevolent M
- Superman is incapable I
- Superman prevents evil E

- Our objective is to prove the proposition:

  ((W and A) $\Rightarrow$ E)
  and ((not A) $\Rightarrow$ I)
  and ((not W) $\Rightarrow$ M)
  and (not E)
  and (X $\Rightarrow$ not (I or M))
  $\Rightarrow$ not X

1. Assume

   ((W and A) $\Rightarrow$ E)
   and ((not A) $\Rightarrow$ I)
   and ((not W) $\Rightarrow$ M)
   and (not E)
   and (X $\Rightarrow$ not (I or M))

   The objective is now to prove not X

2. Assume X
   Use Elimination Rules to break No. 1 down into 5 premises

3. (W and A) $\Rightarrow$ E
4. (not A) $\Rightarrow$ I
5. (not W) $\Rightarrow$ M
6. not E

7. X $\Rightarrow$ not (I or M)

   Now application of elimination on 2 and 7 derives another simple proposition
   2. Assume X
   7. X $\Rightarrow$ not (I or M)

8. not (I or M)
   Now proving I or M will result in a contradiction
   We will now analyze W

9. Assume not W
10. M (from 5 and 9)
11. I or M (from 10 - introduction)

12. Assume W
13. Assume A
14. W and A (12 and 13)
15. E (3 and 14)
16. false (6 and 15)
17. I or M (16)

18. Assume not A
19. I (4 and 18)
20. I or M (19)
21. I or M (17 and 20)
22. I or M (11 and 21)
23. false (contradiction 8 and 22)
24. not X (2 and 23)

Logic problem for the day

On the island of knights and knaves, it is rumored that there is gold buried on the island. You ask one of the natives, A, whether there is gold on the island. He makes the following response: "There is gold on this island if and only if I am a knight."

The problem is as follows:

a) Can it be determined whether A is a knight or a knave?
b) Can it be determined whether there is gold on the island?

Lecture 4

- There is gold on the island G
- A is a knight A
- Therefore we have

A	G	A ↔ G	A ≡ (A ↔ G)
T	T	T	T
F	T	F	T
T	F	F	F
F	F	T	F

Associativity of Equivalence

- A ≡ B ≡ C

  can be evaluated as

  (A ≡ B) ≡ C

  or

  A ≡ (B ≡ C)

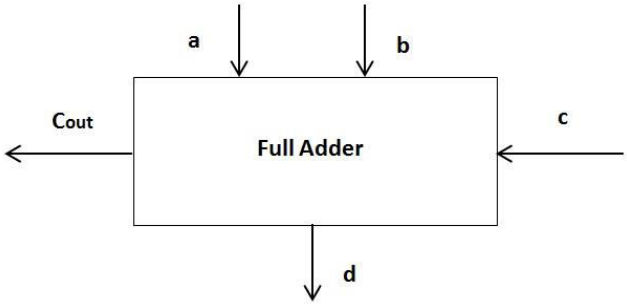
Even and odd numbers

- m+n is even ≡ m is even ≡ n is even
- m+n is even ≡ (m is even ≡ n is even)

A ≡ B ≡ C

A	B	C	A ≡ B	(A ≡ B) ≡ C
F	F	F	T	F
F	F	T	T	T
F	T	F	F	T
F	T	T	F	F
T	F	F	F	T
T	F	T	F	F
T	T	F	T	F
T	T	T	T	T

Full Adder



- a ≡ 1; A
- b ≡ 1; B
- c ≡ 1; C
- d ≡ 1; D
- D ≡ (A ≡ B ≡ C)

- There is gold on the island G
- A is a knight A

  A ≡ (A ≡ G)

  (A ≡ A) ≡ G

  true ≡ G

  G

Properties of equivalence

- constant true

  true ≡ p ≡ p

  true = (p ≡ p)

  (true ≡ p) = p

  p ≡ p ≡ q ≡ p ≡ r ≡ q

  p ≡ p ≡ p ≡ q ≡ q ≡ r

  true ≡ p ≡ true ≡ r

  p ≡ r

Replace the term which is repeated odd number of times by a single occurrence of the term and any terms which is repeated an even number of times by removing all occurrences.

- A is a knight A
- A says “I am a knight.” A

  A ≡ A
- Since this is always true, no meaningful conclusion can be made.

- A is a knight A
- B is a knight B
- A says “I am the same type as B” $A \equiv B$

$A \equiv A \equiv B$

B

Island of knights and knaves

Suppose A is the proposition “person A is a knight” and suppose A makes a statement S. Then A is true is the same as S is true. That is:

$A \equiv S$

Logic problem for the day

A tourist comes to a fork in the road, where one branch leads to a restaurant and one does not. A native of the island is standing at the fork. Formulate a single yes no question that the tourist can ask such that the answer will be yes if the left fork leads to the restaurant, and otherwise the answer will be no.

Lecture 5

- Let Q be the question
- Let A be "the native is a knight"
- Let L be "the left fork leads to the restaurant"
- The response to the question Q is yes is equivalent to $Q \equiv A$
- So we require that: $L \equiv Q \equiv A$ or $Q \equiv (L \equiv A)$
- Is the statement that the left fork leads to the restaurant equivalent to your being a knight?

Negation

- $\neg p \equiv p \equiv$ false
- $\neg p \equiv (p \equiv$ false$)$
- $(\neg p \equiv p) \equiv$ false

- $\neg p \equiv p \equiv q \equiv \neg p \equiv r \equiv \neg q$
- $\neg p \equiv \neg p \equiv p \equiv q \equiv \neg q \equiv r$
- true $\equiv p \equiv$ false $\equiv r$
- $p \equiv \neg\, r$

- There are two natives A and B. A says, "B is a knight is the same as I am a knave."
- What can you determine about A and B?
- A's statement is: $B \equiv \neg\, A$
- So, we have: $A \equiv B \equiv \neg\, A$

  $A \equiv \neg\, A \equiv B$

  false $\equiv B$

  $\neg\, B$

  A?

Golden Rule

- $p \wedge q \equiv p \equiv q \equiv p \vee q$

Implication

- $p \Rightarrow q \equiv p \equiv p \wedge q$
- $p \Rightarrow q \equiv q \equiv p \vee q$

- If I am a knight, B is a knight
- $A \Rightarrow B$
- $A \equiv A \Rightarrow B$
- $A \equiv A \equiv A \wedge B$
- $A \wedge B$

- Three of the inhabitants – A, B, and C – were standing together in a garden. A stranger passed by and asked A, “Are you a knight or a knave?” A answered but the stranger could not understand. The stranger then asked B, “What did A say?”. B replied, “A said that he is a knave”. At this point, the third C, said, “Don’t believe B; he is lying!” What are A, B, and C?

- B’s statement is: $A \equiv \neg A$
- C’s statement is: $\neg B$
- So, we have:

$$(B \equiv A \equiv \neg A) \wedge (C \equiv \neg B)$$
$$\neg B \wedge (C \equiv \neg B)$$
$$(\neg B \wedge C) \equiv (\neg B \wedge \neg B)$$
$$(\neg B \wedge C) \equiv \neg B$$
$$\neg B \wedge C$$

- A says, either I am a knave or B is a knight

$$A \equiv \neg A \vee B$$
$$A \equiv (A \equiv \text{false}) \vee B$$
$$A \equiv (A \vee B \equiv \text{false} \vee B)$$
$$A \equiv A \vee B \equiv B$$
$$A \wedge B$$

Lecture 6

Conditional Correctness:

- {P} S {Q}
  where
  P = initial state
  S = set of statements
  Q = final state
  Express the conditional correctness of S

Weakest Precondition:

- Wp(z := x, z ≥ y) ≡ x ≥ y
- Wp(t := x, t ≡ x_0) ≡ x = x_0
- Wp(i := i+1, i ≤ n) ≡ i < n
- Wp(i := 0, i = 0) ≡ true ~least restrictive, i.e. true for all values
- Wp(i := 0, i = 1) ≡ false ~impossible
- Wp(x := e, Q(x)) ≡ Q(e)
  - ✓ X will have a value which e had before executing statement
  - ✓ Q(e) denotes the predicate obtained by substituting e for all free occurrences of x in the predicate Q.

- Wp(i := i-1, i = 0) ≡ i – 1 = 0
- Wp(i := (l + u) div 2, l ≤ i ≤ u) ≡ l ≤ (l + u) div 2 ≤ u
- Wp(i := 1, i = 0) ≡ i = 0 ~ false
- Wp(x := y, x = z) ≡ z = y

Rule of Sequential Composition:

- wp(S_1;S_2,Q) ≡ wp(S_1, wp(S_2,Q))
  wp((x:= x+1; y := y+1), x = y)
  ≡ wp(x := x+1, wp(y := y+1, x = y))
  ≡ wp(x := x+1, x = y+1)
  ≡ x+1 = y+1
  ≡ x = y

- wp(S_1;S_2,Q) ≡ wp(S_1, wp(S_2,Q))
  wp((x:= 2*x+1; y := y-1), y = 3*x)
  ≡ wp(x := 2*x+1, wp(y := y-1, y = 3*x))
  ≡ wp(x := 2*x+1, y-1 = 3*x))
  ≡ y-1 = 3*(2*x+1)
  ≡ y = 6*x + 4

Lecture 7

Example:

- $\{x = x_0$ and $y = y_0\}$
  - ✓ t := x ; x := y ; y := t ;
- $\{y = x_0$ and $x = y_0\}$
- $\{ x = x_0$ and $y = y_0\}$ t := x $\{ t = x_0$ and $y = y_0\}$
- $\{ t = x_0$ and $y = y_0\}$
- x := y ; y :=t
- $\{y = x_0$ and $x = y_0\}$
- $\{ t = x_0$ and $y = y_0\}$ x := y $\{ t = x_0$ and $x = y_0\}$
- $\{ t = x_0$ and $x = y_0\}$ y := t $\{ y = x_0$ and $x = y_0\}$

Hoare's Consequence Rule:

- $P \rightarrow Q$ -- If P then Q

$$\frac{P \rightarrow Q \quad \{Q\}\ S\ \{R\}}{\{P\}\ S\ \{R\}} \qquad \frac{Q \rightarrow R \quad \{P\}\ S\ \{R\}}{\{P\}\ S\ \{R\}}$$

Rules for Conditions:

$$\frac{\{P \text{ and } C\}\ S\ \{Q\} \quad P \text{ and (not } C) \rightarrow Q}{\{P\} \text{ if } C \text{ then } S\ \{Q\}} \qquad \frac{\{P \text{ and } C\}\ S_1\ \{Q\} \quad \{P \text{ and (not } C)\}\ S_2\ \{Q\}}{\{P\} \text{ if } C \text{ then } S_1 \text{ else } S_2\ \{Q\}}$$

Dijkstra's Healthiness Condition:

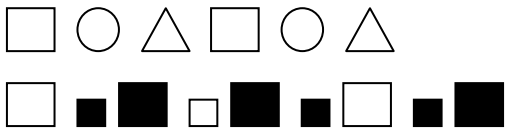
a. wp(S, false) // false empty set
   - ✓ False – law of excluded miracle

b. wp(S, true) // true universal set
   - ✓ Termination condition, all states that grunted termination of S.

c. wp(while $0 \neq n$ do n := n – 1, ture) $\equiv 0 \leq n$

Verification:

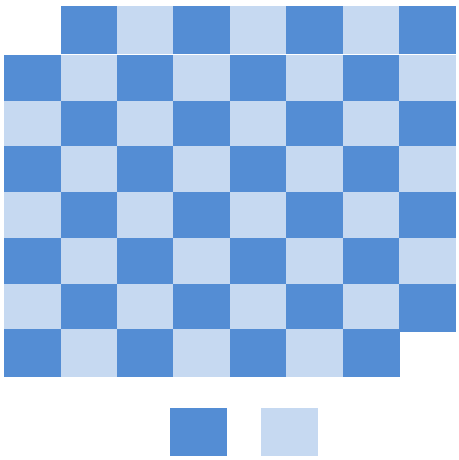
- Invariant
  - ✓ Something which is unchanging
  - ✓ Key to the proofs for programs containing loops (iteration or recursion)



  - ✓ Question seem to ask for next change
  - ✓ Answer lies in determining what does not change

Lecture 8

A Classical Problem:



Question: Can you completely cover the chessboard with these dominoes without partially using a domino? If so how, if not, prove that you cannot.

- if i <= j then
  m := i;
  else
  m := j;
- (m ≤ i and m ≤ j) and (m = i or m = j)

ELSE PART

- (i > j)
  m = j;
- (m = i or m = j) and (m ≤ j and m ≤ j)
- (i>j) and (j = i or j = j) and (j ≤ i and j ≤ j)
- (i>j) and (true) and (j ≤ i and true)
- (i>j) and (j ≤ i)
- (i>j)

IF PART

- (i <= j)
  m = i;
- (m = i or m = j) and (m ≤ i and m ≤ i)
- (i ≤ j) and (i = i or i = j) and (i ≤ i and i ≤ j)
- (i ≤ j) and (true) and (true and i ≤ j)
- (i ≤ j) and (i ≤ j)
- (i ≤ j)

Lecture 9

Loop Invariants:

- s = 0;
  for i := 1 to n do
  s = s + a[i];

- What is the 'loop invariant'?
- s is the sum of elements from a[1] to a[i] immediately before i is incremented!

Weakest Precondition for While Statement {P} while B do S {Q}

- Let W be while B do S
- Condition for termination of the loop
- $P_0 \equiv$ (not B)
- $P_1 \equiv$ B and wp(S,P_0) $\equiv$ wp(S, not B)
- $P_k \equiv$ B and wp(S, P_{k-1})
- wp(W, true) $\equiv$ wp(W, not B) $\equiv \exists(k: k \geq 0: P_k)$

- The invariant condition
- {I} while B do S {I and not B}
- a = 0;
- i = 0;
- while (i < N)
  a = a + i++;
- Loop Invariant: a = $\sum_{i=0}^{N-1} i$

Lecture 10

Verification of Functions

- Specification of a system as a set of functions where the internal state is hidden.
- Each function is specified as a set of pre and post conditions.
- Pre-condition must hold if the post-condition is to be true.

Example – minimum

- function min (X: in INTEGER_ARRAY) return INTEGER
  Pre: True
  Post: $\exists$ j in X'First .. X'Last : min(X) = X(j) and
  $\forall$ i in X'First .. X'Last : min(X) $\leq$ X(i) and X = X''

Specification of functions using pre- and post-conditions

- Procedure Search (X: in INTEGER_ARRAY;
  key: in INTEGER;
  found: in out Boolean;
  index: in out INTEGER);
  Pre: True
  Post: ((found and X(index) = key) or
  (NOT found and
  ($\forall$ j in X'First … X'Last : x(j) $\neq$ key)) and (X = X''))

- Procedure binarySearch (X: in INTEGER_ARRAY;
  key: in INTEGER;
  found: in out Boolean;
  index: in out INTEGER);
  Pre: $\forall$ j in X'First … X'Last-1 : x(j) <= x(j+1)
  Post: ((found and X(index) = key) or
  (NOT found and
  ($\forall$ j in X'First … X'Last : x(j) $\neq$ key)) and (X = X''))

- Procedure binary_Search (X: in INTEGER_ARRAY;
  key: in INTEGER;
  found: in out Boolean;
  L: in out INTEGER)
  begin
  bot: INTEGER := X'First;
  top: INTEGER := X'Last;
  mid: INTEGER;
  L := (bot + top) / 2;
  found := X(L) = key ;

```
        while (bot <= top AND NOT found) loop
        begin
          mid := (bot + top)/2;
          if X(mid) = key then
                found := TRUE;
                L := mid;
          elsif X(mid) < key then
                bot := mid + 1;
          else
                top := mid - 1;
          end if;
        end loop;
end binary_Search;
```