MUST

Wisdom & Virtue

MIRPUR UNIVERSITY OF SCIENCE AND TECHNOLOGY (MUST), MIRPUR

DEPARTMENT OF SOFTWARE ENGINEERING

Formal Methods in Software Engineering

Lecture [7]: Function Signs, Argument & Proofs, Algorithms Proof

Engr. Samiullah Khan

(Lecturer)

Topics discussed in Today's Lectures

- Predicate
- Function Signs
- Argument & Proofs
- Algorithm Design

Predicates

- Upper case Roman letters, plus square brackets:
- Examples:
  - H[a] : a is happy
  - R[a, b] : a respects b
  - S[a,b,g] : a sold b to g
  - H[a,b,g,d] : a is happy that b sold g to d

Function Signs

- In predicate logic, function symbols (or function signs) are used to represent functions
  - Mappings that take one or more arguments (objects) and return a single object
- A function sign represents a rule or operation that produces an object when applied to arguments

Form: $f(t1,t2,\ldots,tn)$

- where: f is a function sign $t1,t2,\ldots,tn$ are terms (constants, variables, or other function results)

Function Signs

- Examples:
  - m(a) : the mother of a
  - s(a,b) : the sum of a and b
  - s(a,b,g) : the sum of a, b, g

Common Function Sign Examples

Function Sign	Meaning	Example Expression	Explanation
father(x)	father of x	father(ali)	Returns the father of Ali
sum(x, y)	sum of x and y	sum(2, 3)	Returns 5 (object/value)
age(x)	age of x	age(ahmad)	Returns Ahmad's age (a number)
mother(x)	mother of x	mother(sara)	Returns Sara's mother
plus(x, y)	arithmetic addition	plus(4, 7)	Returns 11

Complete Predicate Logic Example Using Function Signs

Let's use them inside predicates (which give truth values):

Older(father(ali), ali)
→ "Ali's father is older than Ali."

Greater(sum(x, y), z)
→ "The sum of x and y is greater than z."

Female(mother(sara))
→ "Sara's mother is female."

HasCar(ownerOf(car1))
→ "The owner of car1 has a car."

Equal(plus(2,3), 5)
→ "The sum of 2 and 3 equals 5."

Argument & Proofs in Predicate Logic

- In predicate logic, an **argument** is a set of premises (statements assumed to be true) and a conclusion that supposedly follows from them.
- A **proof** is a sequence of logical steps showing that the conclusion logically follows from the premises using valid inference rules

- **Structure of an Argument**

$$\text{Premise 1, Premise 2, } \ldots, \text{ Premise n} \Longrightarrow \text{Conclusion}$$

If the conclusion follows logically from the premises, the argument is **valid**.

Argument & Proofs - Examples

- **Example 1:**

Premises:

1. ∀x (Human(x) → Mortal(x))   (Every human is mortal)
2. Human(Socrates)   (Socrates is a human)

Conclusion:

∴ Mortal(Socrates)   (Socrates is mortal)

✅ **Proof:**

From (1) and (2), by **Universal Instantiation** and **Modus Ponens**, we conclude **Mortal(Socrates)**.

- **Example 2:**

Premises:

1. ∀x (Cat(x) → Animal(x))
2. Cat(Kitty)

Conclusion:

∴ Animal(Kitty)

✅ **Proof:**

Using **Universal Instantiation** (1) → Cat(Kitty) → Animal(Kitty)

- **Example 3:**

Premises:

1. ∀x (Bird(x) → Fly(x))
2. Bird(Sparrow)

Conclusion:

∴ Fly(Sparrow)

✅ **Proof:**

Universal Instantiation + Modus Ponens

Argument & Proofs - Examples

- **Example 4:**

Premises:

1. $\forall x\,(Student(x) \rightarrow Studies(x))$
2. $Student(Ali)$

Conclusion:

$\therefore Studies(Ali)$

✅ **Proof:**

From (1) and (2), we infer **Studies(Ali)**.

- **Example 5:**

Premises:

1. $\forall x\,(Father(x) \rightarrow Male(x))$
2. $Father(Ahmed)$

Conclusion:

$\therefore Male(Ahmed)$

✅ **Proof:**

From (1) and (2), by **Modus Ponens**, Ahmed is male.

Argument & Proofs - Summary

- Summary

Term	Meaning
Argument	Set of premises and a conclusion
Proof	Step-by-step derivation showing validity
Valid Argument	Conclusion follows necessarily from premises
Invalid Argument	Conclusion does not logically follow

Algorithm Design

- An algorithm is a sequence of simple steps that can be followed to solve a problem
- These steps must be organized in a logical, and clear manner
- We design algorithms using three basic methods of control: sequence, selection, and repetition
  i. Sequential control
  ii. Selection Control
  iii. Repetition

Algorithm Design

i. Sequential control

- Sequential Control means that the steps of an algorithm are carried out in a sequential manner where each step is executed exactly once
- Let's look at the following problem:
  - We need to obtain the temperature expressed in Fahrenheit degrees and convert it to degrees Celsius
  - An algorithm to solve this problem would be:
    1. Read temperature in Fahrenheit
    2. Apply conversion formula
    3. Display result in degrees Celsius

Algorithm Design

i. Sequential control (Contd…)

- A pseudocode is a mixture of English, symbols, and selected features commonly used in programming languages
- Here is the above algorithm written in pseudocode:
  i. READ degrees_Farenheit
  ii. Degrees_Celcius=(5/9)*(degrees_Farenheit-32)
  iii. DISPLAY degrees_Celsius
- Another option for describing an algorithm is to use a graphical representation (Flow Chart) in addition to of pseudocode.

Algorithm Design

ii. Selection Control

- In Selection Control only one of a number of alternative steps is executed
- Let's see how this works in specific examples
- Example: Control access to a computer depending on whether the user has supplied a username and password that are contained in the system database.

i. IF (user name in database AND password corresponds to user)

ii. THEN Accept user

iii. ELSE Deny user

Algorithm Design

iii. Repetition

- In Repetition one or more steps are performed repeatedly.
- Example: Read numbers and add them up until their total value reaches (or exceeds) a set value represented by S.

  i. WHILE (total<S)

  ii. DO Read number

  iii. total=total+number

  iv. END DO

Problem Solving

- Problem solving is to *find set of actions to achieve the desired goals*
- Problem solving goes through several steps starting from defining the initial problem, generating tentative solutions, eliminate errors, then solve new problem till convergence or satisfaction

P1 -- the initial problem
TS -- generate tentative solutions
EE -- eliminate errors
P2 -- the new problem

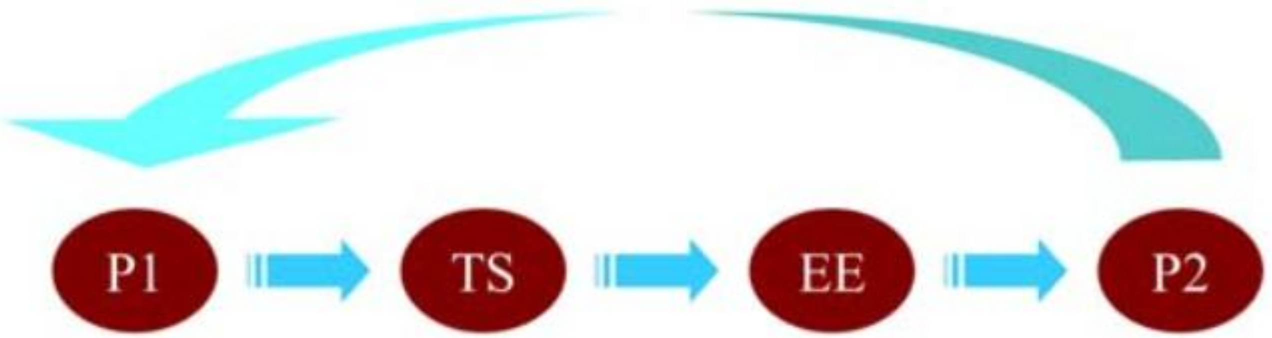

Figure 1-3. Problem Solving

Problem Solving

- Problem solving process is to search for optimum solution in the problem domain
- This can be achieved using the following steps:

i. Define initial state

ii. Describe set of all possible actions, with state space and transition caused by these actions

iii. Define goals and performance indicators

iv. Define path cost function that assigns and calculates the cost for each possible path from initial state to final goal

Problem Solving

- Key feature in the problem solving process is the formal representation of the problem elements, i.e.
  - Initial state
  - Actions
  - Goals
  - Path costs
- The search algorithm will greatly help to find the optimum solution with minimum cost

THANKS