CSC236H

Introduction to the Theory of Computation

Program Correctness – Review

- **Program Correctness:** the program produces a **correct output** on every **acceptable input**.

- **Program Specification:**

  - **Precondition**: an assertion which states what *must* be true before the program starts execution (acceptable inputs).

  - **Postcondition**: an assertion which states what must be true when the program terminate (correct output for the given input).

- A program is **correct with respect to a specification**, if whenever the precondition holds before the program starts execution, then
  1. the program terminates;
  2. after the termination, the postcondition holds.

Correctness of Recursive Programs – Review

- We do not prove termination and partial correctness separately, we encapsulate both ideas in a single predicate.

- The predicate must be defined over the **size** of the input.

- General form for the predicate:

  - $P(n)$: If precondition holds, and the program runs, and its input size is n, then the program halts and the postcondition holds after it halts.

  - Sanity check for identifying the correct predicate variable: The input for any recursive calls of the code is smaller than the input for the initial call of the code. So the input size must decrease in each recursive call.

- Must prove termination and the postcondition for all possible **paths**.

Correctness of Recursive Programs – Review

- **Precondition:** A is a list of numbers. $0 \leq s \leq e \leq len(A) - 1$ and $A[s..e]$ is sorted in non-decreasing order.
- **Postcondition:** Return t such that $s \leq t \leq e$ and $A[t] = x$, if such a t exists; otherwise return -1.

```
def RecBinSearch(A, s, e, x):
1.    if (s == e) then:
2.       if (A[s] == x) then:
3.          return s
4.       else:
5.          return -1
6.    else:
7.       m = (s + e) // 2            # integer division
8.       if (A[m] >= x) then:
9.          return RecBinSearch(A, s, m, x)
10.      else:
11.         return RecBinSearch(A, m + 1, e, x)
```

$P(n)$: If s, e are integers such that $0 \leq s \leq e \leq len(A) - 1$, and $A[s..e]$ is sorted in non-decreasing order, and $len(A[s..e]) = n$, then $RecBinSearch(A, s, e, x)$ terminates and returns t such that $s \leq t \leq e$ and $A[t] = x$, if such a t exists; otherwise it returns -1.

Exercises

- Prove the correctness of the following programs:

- **Precondition:** S is a string.
- **Postcondition:** Returns the reverse of S.

```
def rev(S):
1.    if (len(S) == 0):
2.       return S
3.    else:
4.       return [S[-1]] + rev(S[:-1])
```

- **Precondition:** $n \in \mathbb{N}$.
- **Postcondition:** Returns the total sum of every digit that n contains.

```
def digitsSum(n):
1.    if n < 10:
2.       return n
3.    else:
4.       return n % 10 + digitsSum(n//10)
```

Correctness of Iterative Programs

- Correctness of Iterative Programs:
  - **Termination:** $Precondition \Rightarrow Termination$.
  - **Partial Correctness** $Precondition \wedge Termination \Rightarrow Postcondition$.

Correctness of Iterative Programs – Termination

- **Termination:** $Pre \Rightarrow Term.$
- **Partial Correctness** $Pre \wedge Term \Rightarrow Post.$

- **Precondition:** $Pre\_Cond.$
- **Postcondition:** $Post\_Cond.$

```
def iter_prog(r_1,...,r_n):
#Pre_Cond
          .
          .
          .

     while loop_cond: {
               .
               .
               .
     }

     #Some instructions after the loop
     return res
  #Post_Cond
```

Loop Termination

- Associate with the loop a **loop measure** m:

    1. m decreases with each iteration of the loop;
    2. m is always a natural number at the beginning of each loop iteration

- If such an m exists, then the loop terminates: eventually m reaches 0, which is the smallest natural number, and therefore the loop cannot have any more iterations.

Loop Termination – Example

- **Precondition:** A is a non-empty list of numbers.
- **Postcondition:** Returns the average of the numbers in A.

```
def avg(A):
1.    sum = 0
2.    i = 0
3.    while i < len(A):
4.          sum += A[i]
5.          i += 1
6.    return sum / len(A)
```

Correctness of Iterative Programs – Loop Invariant

- **Precondition:** $Pre\_Cond.$
- **Postcondition:** $Post\_Cond.$

```
def iter_prog(r_1,...,r_n):
#Pre_Cond
        .
        .
        .

    while loop_cond: {
            .
            .
            .
    }

    #Some instructions after the loop
    return res
 #Post_Cond
```

- **Loop Invariant:** a statement that is true on entering the loop, and after every iteration.

Loop Invariant – Example

- **Precondition:** A is a non-empty list of numbers.
- **Postcondition:** Returns the average of the numbers in A.

```
def avg(A):
1.    sum = 0
2.    i = 0
3.    while i < len(A):
4.          sum += A[i]
5.          i += 1
6.    return sum / len(A)
```

Steps in proving Correctness of Iterative Programs

1. Formulate a loop invariant (**LI**).
2. Prove the LI using **Induction**:
    a) Prove that assuming the precondition holds, then the LI holds on entering the loop; **(Base Case)**
    b) Prove that if the LI holds before an iteration, then it also holds after that iteration. **(Induction Step)**
3. Use the LI to prove **partial correctness**:
    a) Proving that if the loop halts, then the postcondition follows:
    The **loop exit condition** (negation of the condition in the while loop) and the **LI** implies **postcondition**.
4. Use LI to find a **loop measure** m such that
    a) the value of m is a natural number on entering the loop, and after every iteration.
    b) the value of m decreases with every iteration.
5. Prove that the **loop measure** m actually satisfies the above conditions.

Steps in proving Correctness of Iterative Programs

- **Precondition:** $n \in \mathbb{N}$
- **Postcondition:** Returns n^2.

```
def Sq(n):
1.    s = 0; d = 1; i = 0
2.    while i < n:
3.        s = s + d
4.        d = d + 2
5.        i = i + 1
6.    return s
```

Exercises

- Prove the correctness of the following programs.

- **Precondition:** x and y are natural numbers, and $x \geq y$.
- **Postcondition:** Returns $x * y$.

```
def M(x,y):
1.     i = 0;
2.     r = 0
3.     while i < y:
4.           r += x
5.           i += 1
6.     return r
```

Exercises

- **Precondition:** $n \in \mathbb{N}$
- **Postcondition:** Returns $n!$.

```
def G(n):
1.    i = 0; f = 1;
2.    while i < n:
3.          i = i + 1
4.          f = f * i
5.    return f
```

- **Precondition:** A is a list of natural numbers and the length of A is greater than 0.
- **Postcondition:** Returns largest number in A.

```
def H(A):
1.    result = 0
2.    i = 0
3.    while(i < len(A)):
4.        if(A[i] > result):
5.             result = A[i]
6.        i += 1
7.    return result
```