Lecture 8: correctness proof of iterative programs

Amir H. Chinaei, Fall 2016

Office Hours: W 2-4 BA4222

ahchinaei@cs.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/

Course page:
http://www.cdf.toronto.edu/~csc236h/fall/index.html

Section page:
http://www.cdf.toronto.edu/~csc236h/fall/amir_lectures.html
review

❖ Last lecture
  ▪ correctness proof of recursive programs
    • based on the program specification
      – i.e., pre- & post- conditions
    • normally using induction
      – proof is parallel to the code

❖ this week
  ▪ correctness proof of iterative programs
    • based on the program specification
      – i.e., pre- & post- conditions
Example 76: $x^n$

```python
def power(x, n):
    r = 1
    c = 0
    while c < n:
        r = r * x
        c = c + 1
    return r
```

- How to start the proof, formally?
  - recall, we need to show:
    
    preconditions $\Rightarrow$ postconditions
Example 76: pre- post- conditions

- power(x, n):
  - preconditions:
    - $0 \leq n \in \mathbb{N}$
    - $x \in \mathbb{R}$
  - postconditions:
    - power(x, n) terminates and returns $x^n$


**Example 76: correctness of power, \(x^n\):**

- We want to show: preconditions \(\implies\) postconditions
  
  \(P(n):\) if \(n \in \mathbb{N}, x \in \mathbb{R},\) then \(\text{power}(x, n)\) terminates and returns \(x^n.\)

- **Partial correctness:**
  
  \(P'(n):\) if \(n \in \mathbb{N}, x \in \mathbb{R},\) and \(\text{power}(x, n)\) terminates, then it returns \(x^n.\)

- To prove **partial correctness** of iterative algorithms, we use
  
  - **loop invariant:** an assertion (predicate) that must be true before and after each iteration of the loop
Example 76: loop invariant

```python
def power(x, n):
    r = 1
    c = 0
    while c < n:
        r = r * x
        c = c + 1
    return r

• loop invariant:
  • $LI(c, r): 0 \leq c \leq n$ and $r = x^c$
```
recipe

- show the \( LI \) holds before the loop starts
  - show \( LI(c_0, r_0) \) holds

- show the \( LI \) holds at each iteration of the loop
  - show \( LI(c_k, r_k) \rightarrow LI(c_{k+1}, r_{k+1}) \)

- show the \( LI \) holds after the loop exits
  - by showing \( LI(c_n, r_n) \) holds,
    - conclude the postcondition is met.
Example 76: *partial correctness*

\[ LI(c, r): 0 \leq c \leq n \text{ and } r = x^c \]

```
def power(x, n):
    r = 1
    c = 0
    while c < n:
        r = r * x
        c = c + 1
    return r
```
Example 76: partial correctness

\[ LI(c, r): 0 \leq c \leq n \text{ and } r = x^c \]
Example 76: \textit{partial correctness}

\[ LI(c, r): 0 \leq c \leq n \text{ and } r = x^c \]

```python
def power(x, n):
    r = 1
    c = 0

    while c < n:
        r = r * x
        c = c + 1

    return r
```

an important task remains:

- Termination of power
  - To prove termination of iterative algorithms, we use
    - loop variant: a number, $k_c$, associated with each iteration $c$ of the loop, and we show
      - $k_c \in \mathbb{N}$
      - $k_c$ is decreasing
Example 76: *loop variant*

```python
def power(x, n):
    r = 1
    c = 0
    while c < n:
        r = r * x
        c = c + 1
    return r
```

- **loop variant:**
  - $k_c = n - c$
Example 76: terminations

\[ k_c = n - c \]

```python
1  def power(x, n):
2      r = 1
3      c = 0
4
5      while c < n:
6          r = r * x
7          c = c + 1
8
9      return r
```
Example 77: iterative binSearch

```python
def iteBinSearch(x, A):
    # Precondition: A is a sorted array, and
    #               A is not empty.
    # Postcondition: Return an integer p such that 0<= p <= length(A)
    # and A[p] = x, if such a p exists; otherwise return -1.

    b = 0
    e = len(A)-1

    while b != e:
        m = (b + e) // 2  # midpoint
        if x <= A[m]:
            e = m
        else:
            b = m + 1

        if A[b] == x:
            return b
        else:
            return -1
```
Example 77: pre- post- conditions

- \text{iteBinSearch}(x, A):
  - \textbf{preconditions}:
    - \(A\) is not empty
    - elements of \(A\) are sorted non-decreasingly
    - elements of \(A\) and \(x\) are comparable
  - \textbf{postconditions}:
    - \text{iteBinSearch}(x, A) terminates and returns \(p\) such that \(0 \leq p \leq \text{Length}(A) - 1\) and \(x = A[p]\) if such a \(p\) exists;
    - otherwise it terminates and returns \(-1\).
Example 77: correctness of \textit{iteBinSearch}:

- We want to show: preconditions $\Rightarrow$ postconditions

  $P(n)$: \textbf{if} A is not empty and non-decreasing and $n = \text{Length}(A) > 1$, and x is comparable to elements of A, \textbf{then} \textit{iteBinSearch}(x,A) terminates and returns $p$ such that $0 \leq p \leq \text{Length}(A) - 1$ and $x = A[p]$ if such a $p$ exists; otherwise it terminates and returns $-1$.

- \textbf{Part A) Partial correctness:}

  $P'(n)$: \textbf{if} A is not empty and non-decreasing and $n = \text{Length}(A) > 1$, and x is comparable to elements of A and \textit{iteBinSearch}(x,A) terminates, \textbf{then} it returns $p$ such that $0 \leq p \leq \text{Length}(A) - 1$ and $x = A[p]$ if such a $p$ exists; otherwise it returns $-1$.

- \textbf{Part B) Termination:}

  $P''(n)$: \textbf{if} A is not empty and non-decreasing and $n = \text{Length}(A) > 1$, and x is comparable to elements of A, \textbf{then} \textit{iteBinSearch}(x,A) terminates.
Example 77: *loop invariant*

```python
def iteBinSearch(x, A):
    b = 0
    e = len(A)-1
    while b != e:
        m = (b + e) // 2  # midpoint
        if x <= A[m]: e = m
        else: b = m + 1
    if A[b] == x: return b
    else: return -1
```

❖ *loop invariant:*
  - ...

```
recipe

- show the LI holds before the loop starts
  - show $LI(b_0, e_0)$ holds

- show the LI holds at each iteration of the loop
  - show $LI(b_k, e_k) \rightarrow LI(b_{k+1}, e_{k+1})$

- show the LI holds after the loop exits
  - and from there, conclude the postcondition is met.

We use induction to show the LI holds for all $i$’s.
Example 77: *partial correctness*

\( P(i) \): if the loop iterates at least \( i \) times, \( LI(b_i, e_i) \) holds.

```python
def iterativeBinSearch(x, A):
    b = 0
    e = len(A) - 1

    while b != e:
        m = (b + e) // 2
        if x <= A[m]:
            e = m
        else:
            b = m + 1

    if A[b] == x:
        return b
    else:
        return -1
```

**correctness 8-20**
Example 77: partial correctness

\( P(i) \): if the loop iterates at least \( i \) times, \( LI(b_i, e_i) \) holds.

```python
1 def iteBinSearch(x, A):
2     b = 0
3     e = len(A) - 1
4
5     while b != e:
6         m = (b + e) // 2
7         if x <= A[m]:
8             e = m
9         else:
10             b = m + 1
11
12     if A[b] == x:
13         return b
14     else:
15         return -1
```

Example 77: *partial correctness*

\[ \text{LI}(b_i, e_i): 0 \leq b_i \leq e_i \leq n - 1 \text{ and if } x \text{ is in } A, \text{ it's in } A[b_i..e_i] \]

\[ P(i): \text{if the loop iterates at least } i \text{ times, } \text{LI}(b_i, e_i) \text{ holds.} \]

```
def iteBinSearch(x, A):
    b = 0
    e = len(A) - 1

    while b != e:
        m = (b + e) // 2
        if x <= A[m]:
            e = m
        else:
            b = m + 1

    if A[b] == x:
        return b
    else:
        return -1
```
an important task remains:

- Termination of `iteBinSearch`
  - To prove termination of iterative algorithms, we use
    - **loop variant**: a number, $k_i$, associated with each iteration $i$ of the loop, and we show
      - $k_i \in \mathbb{N}$
      - $k_i$ is decreasing

"correctness 8-23"
Example 77: loop variant

- loop variant:

```python
def iteBinSearch(x, A):
    b = 0
    e = len(A)-1

    while b != e:
        m = (b + e) // 2
        if x <= A[m]:
            e = m
        else:
            b = m + 1

    if A[b] == x:
        return b
    else:
        return -1
```
Example 77: terminations

```python
def iteBinSearch(x, A):
    b = 0
    e = len(A) - 1

    while b != e:
        m = (b + e) // 2
        if x <= A[m]:
            e = m
        else:
            b = m + 1

    if A[b] == x:
        return b
    else:
        return -1
```