CSC236 winter 2020, week 8: Proving termination

Recommended supplementary reading: Chapter 2 Vassos course notes

Colin Morris
colin@cs.toronto.edu
http://www.cs.toronto.edu/~colin/236/W20/

March 2, 2020

Proving termination

```python
def imax(A):
  """Pre: A is non-empty and contains comparable items.
  Post: return the maximum element in A
  """
  curr = A[0]
  i = 1
  while i < len(A):
    if A[i] > curr:
      curr = A[i]
    i += 1
  return curr
```

Eventually i must reach `len(A)`...

A corollary of principle of well-ordering

All decreasing sequences of natural numbers are finite.

Spot the decreasing sequence

3-5

```
1  def imax(A):
2    """Pre: A is non-empty and contains comparable items.
3    Post: return the maximum element in A
4    """
5    curr = A[0]
6    i = 1
7    while i < len(A):
8      if A[i] > curr:
9        curr = A[i]
10     i += 1
11   return curr
```

len(A) - i

decreasing? ✓ by l10

in $\mathbb{N}$? yes, but need a loop inv

- $i_j \leq len(A)$ (can prove using while condition)
- $i_j \in \mathbb{N}$

Another corollary of PWO

Any increasing sequence of natural numbers with an upper bound is finite.

Recipe: proving termination

Define some quantity m_j associated with each iteration j of the loop.

- will be defined in terms of one or more variables that change inside the loop
- e.g. $m_j = \mathrm{len}(A) - i_j$

Show that

- Every $m_j \in \mathbb{N}$
- the sequence $\langle m_0, m_1, m_2, \ldots \rangle$ is decreasing.

Example: A2Q3 appendix

```
1  def R(A):
2      B = []
3      i = 0
4      while i < len(A):
5          a = A[i]
6          b = A[(i+1) % len(A)]
7          if a == b:
8              B.append(a)
9          i += 1
10     return B
```

Loop inv: $i_j \leq len(A)$

Lemma (R termination)

R terminates on any $A \in \mathbb{N}^*$

Proof.

Let $m_j = \text{len}(A) - i_j$ be a quantity associated with each loop iteration j. By Lemma 1.3 (a), $m_j \in \mathbb{N}$. By line 9, $m_{j+1} = m_j - 1$. Thus $m_0, m_1, m_2, \ldots$ is a decreasing sequence of natural numbers, and therefore finite. Therefore, R terminates. ☐

(optional conclusion)

merge

```
1  def merge(A, B):
2    """Pre: A and B are sorted lists of numbers.
3    Post: return a sorted permutation of A+B
4    """
5    i = j = 0
6    C = []
7    while i < len(A) and j < len(B):
8      if A[i] <= B[j]:
9        C.append(A[i])
10       i += 1
11     else:
12       C.append(B[j])
13       j += 1
14   return C + A[i:] + B[j:]
```

$m_k = (\text{len}(A) + \text{len}(B)) - (i_k + j_k)$

$m_{k+1} < m_k$? ✓ by lines 10+13

$m_k \in \mathbb{N}$? Use loop invs:

- $i_k \leq \text{len}(A)$
- $j_k \leq \text{len}(B)$
- $i_k, j_k \in \mathbb{N}$

bitcount (week 7 tutorial exercise)

```python
1  def bitcount(n):
2      """Pre: n is a positive int.
3      Post: return the number of digits in the binary representation of n
4      """
5      i = 1
6      while n > 1:
7          n = n//2
8          i += 1
9      return i
```

m_j = # of bits in n

($= n_j$) ← Alternative

decreasing? $n_{j+1} = n_j // 2$ by l7

WTS: $n//2$ has fewer bits than n (by lemma in sample sol'ns)

This follows from $n > 1$ (by while cond)

$\in \mathbb{N}$? yes, by def'n

Tricky example

Let b_j denote # of digits in n_j

```python
def mystery(n):
  """Pre: n is a positive int
  """
  i = 0
  while n > 1:
    if n % 10 == 0:
      n = n // 10
    else:
      n += 1
    i += 1
  return i
```

$m_j = \underbrace{11 \times b_0} - i_j$

$m_{j_2} = 10 + n_0 - i_j$

Decreasing?

WTP $m_{j+1} < m_j$

$$m_{j+1} = 10 + n_0 - i_{j+1}$$
$$= 10 + n_0 - i_j - 1$$
$$= m_j - 1$$

$m_j \in \mathbb{N}$?

$n = 11$

i	
1	11
2	
3	

See Piazza for full write-up