More on Mathematical Induction

COMPSCI 230 — Discrete Math

March 8, 2016
More on Mathematical Induction

1. Induction Pitfalls

2. Mathematical Induction and Recursion

3. Strong Mathematical Induction
Mathematical Induction

- Used to prove predicates of the form

\[ \forall n \in \mathbb{Z} : n \geq a \rightarrow P(n) \]

- Inference rule: Let \( b \) be an integer with \( b \geq a \).

  Base case(s) : \( P(a) \land \ldots \land P(b) \)
  Inductive step : \( \forall k \in \mathbb{Z} : ((k \geq b) \land P(k)) \rightarrow P(k + 1) \)
  Conclusion : \( \forall n \in \mathbb{Z} : n \geq a \rightarrow P(n) \)
All Horses are the Same Color

- \( P(n) \): The horses in any group of \( n \) horses are all the same color
- Prove by induction that \( \forall n : n \geq 1 \rightarrow P(n) \)
- Base case \( P(1) \): A horse is the same color as itself, so \( P(1) \) is true
- Inductive step \( P(k) \rightarrow P(k + 1) \):
  - Make the inductive hypothesis that \( P(k) \) is true so all horses in any group of \( k \) are the same color
  - Number the horses in a new group of \( k + 1 \) horses from 1 to \( k + 1 \)
  - Because of \( P(k) \), horses 1 through \( k \) are the same color
  - For the same reason, horses 2 through \( k + 1 \) are the same color
  - The middle horses, between 2 and \( k \), do not change color if they are in different groups
  - By transitivity, horse 1 and horse \( k + 1 \) are the same color: Horse 1 color = any middle horse color = horse \( k + 1 \) color
- Are all horses the same color? If not, where is the flaw?
Observations

• The inductive step works for all $k > 1$ but not for $k = 1$
• Set of horses for $k + 1 = 2$ is $\{1, 2\}$
• There are no “middle horses” (between 2 and $k$, that is, between 2 and 1)
• Make sure that the inductive step proof is general
• It must hold for every $k \geq b$, not just for most of them
Mathematical Induction and Recursion

• There is an intimate connection between
  • recursively defined objects
  • and proving properties about them by induction
• Example: Prove that Russian Peasant Multiplication (RPM) computes the product of any integer $i$ with a nonnegative integer $j$
• FDM 3.7.1 does it based on iteration
• Iteration complicates the analysis
• Recursive thinking elucidates the connection best
• A recursive variant on FDM 3.7.1
• FDM 3.7.1 is not required reading. These slides are
### Sample Run of RPM

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>(i{2} \quad j{2} \quad p{2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{rpm}(12,5,0))</td>
<td></td>
<td></td>
<td>1100 101 0</td>
</tr>
<tr>
<td>(\text{rpm}(24,2,12))</td>
<td></td>
<td></td>
<td>11000 10 1100</td>
</tr>
<tr>
<td>(\text{rpm}(48,1,12))</td>
<td></td>
<td></td>
<td>110000 1 1100</td>
</tr>
<tr>
<td>(\text{rpm}(96,0,60))</td>
<td></td>
<td></td>
<td>1100000 0 111100</td>
</tr>
</tbody>
</table>

**Observation:** In each row, \(ij + p = 60\)
A Recursive Description of RPM

• We work with $i \in \mathbb{Z}$ and $j \in \mathbb{N}$ while keeping a running value for the product, initially zero
• $j$ is repeatedly divided by 2 to reveal its bits one at a time
• For each new bit in $j$, the value of $i$ is multiplied by 2
• Code structure:
  • If $j = 0$, we are done
  • Otherwise
    • If $j$ is odd, we add $i$ to the product, otherwise we leave the product as is
    • Either way, we apply RPM to twice $i$ and the integer division of $j$ by 2
Recursive Implementation of RPM

```python
# RPM multiplies integer i by nonnegative integer j
# by keeping a running product, initially 0
def rpm(i, j, p = 0):
    assert type(i) is int and type(j) is int and j >= 0
    # If j = 0, we are done and return the product.
    if j == 0: return p
    # Otherwise, we return the result of calling rpm on
    else: return rpm(
        # twice i,
        2 * i,
        # the integer division of j by 2,
        j // 2,
        # and the product plus i if j is odd
        p + i if j % 2 \
        # or the product itself otherwise
        else p)
```

# Mathematical Induction and Recursion

# More on Mathematical Induction

March 8, 2016 9 / 20
def rpm(i, j, p=0):
    assert type(i) is int and type(j) is int and j>=0
    if j==0: return p
    else: return rpm(2*i, j//2, p+i if j%2 else p)

Even more succinctly, and removing the assert statement just while we reason about the code:

def rpm(i, j, p=0):
    return p if j==0 else rpm(2*i, j//2, p+i if j%2 else p)
def rpm(i, j, p=0):
    return p if j==0 else rpm(2*i, j//2, p+i if j%2 else p)

• Suppose calling \( \text{rpm}(i_0, j_0) \).
• **Termination:** The argument \( j \) keeps shrinking and will hit 0
• **Correctness:** Prove that

\[
\forall n \geq 1 : I(n)
\]

where the predicate \( I(n) \) (an invariant) means

\[
ij + p = i_0j_0 \quad \text{in the } n\text{-th recursive call}
\]

• Since \( \text{rpm} \) returns when \( j = 0 \), it returns \( p = i_0j_0 \)
Base Case

```python
def rpm(i, j, p=0):
    return p if j==0 else rpm(2*i, j//2, p+i if j%2 else p)
```

- Suppose calling `rpm(i_{0}, j_{0})`. Prove that

\[ \forall n \geq 1 : I(n) \]

where \( I(n) \) means that \( ij + p = i_{0}j_{0} \) in the \( n \)-th recursive call

- Base case \( I(1) \): Initially, \( i = i_{0}, j = j_{0}, \) and \( p = 0, \) so

\[ ij + p = i_{0}j_{0} + 0 = i_{0}j_{0} \]

(notice that we do not know what \( i_{0}j_{0} \) is!)
def rpm(i, j, p=0):
    return p if j==0 else rpm(2*i, j//2, p+i if j%2 else p)

• \( I(n) \) means that \( ij + p = i_0j_0 \) in the \( n \)-th recursive call
• **Inductive step**: \( I(k) \rightarrow I(k+1) \)
• **Inductive assumption**: \( I(k) \) holds: \( ij + p = i_0j_0 \) in the \( k \)-th call
• What are the values \( i', j', p' \) of \( i, j, p \) in the \( k + 1 \)st call?

\[
i' = 2i \quad j' = \begin{cases} \frac{j-1}{2} & \text{if } j \text{ is odd} \\ \frac{j}{2} & \text{otherwise} \end{cases} \quad p' = \begin{cases} p + i & \text{if } j \text{ is odd} \\ p & \text{otherwise} \end{cases}
\]

• So

\[
i' j' + p' = \begin{cases} 2i \frac{j-1}{2} + p + i & \text{if } j \text{ is odd} \\ 2i \frac{j}{2} + p & \text{otherwise} \end{cases} = ij + p = i_0j_0
\]

(last equality by the inductive assumption)
• So \( I(k+1) \) holds: \( ij + p = i_0j_0 \) in the \( k + 1 \)-st call
• **Done!** \( rpm \) is correct
Strong Mathematical Induction

• Still prove $\forall n \geq a \ P(n)$
• Same base case: $P(a) \land \ldots \land P(b)$
• Inductive step of weak mathematical induction: $P(k) \rightarrow P(k + 1)$
• Inductive step of strong mathematical induction: $[P(a) \land \ldots \land P(k)] \rightarrow P(k + 1)$
• Used when the inductive assumption needs to be stronger in order to conclude $P(k + 1)$
• Proof of validity:
  • Let $Q(n)$ be the predicate $P(a) \land \ldots \land P(n)$
  • Use weak induction to prove $\forall n \geq a \ Q(n)$
  • (Weak) Inductive step $Q(k) \rightarrow Q(k + 1)$ now means $P(a) \land \ldots \land P(k) \rightarrow P(a) \land \ldots \land P(k + 1)$
  • ... and so in particular $P(a) \land \ldots \land P(k) \rightarrow P(k + 1)$
Strong Mathematical Induction

Multiple dominos contribute to toppling the next one
Football Example

• Assume that a football team can only score either 3 points (field goal) or 7 points (touchdown)
• Prove that (ignoring time constraints) it is mathematically possible for a team to score any number of points from 12 on up
Football Example

• Assume that a football team can only score either 3 points (field goal) or 7 points (touchdown)
• Prove that (ignoring time constraints) it is mathematically possible for a team to score any number of points from 12 on up
• $\forall n \geq 12 \exists f \in \mathbb{N} \exists t \in \mathbb{N} : 3f + 7t = n$

  \[ P(n) \]

• Observation: cannot score 1, 2, 4, 5, 8, or 11 points
• Base cases: $P(12) \land P(13) \land P(14)$
• $P(12)$: score four field goals (12)
• $P(13)$: score two field goals (6) and a touchdown (7)
• $P(14)$: score two touchdowns (14)
Strong Inductive Step

- Inductive assumption: \( k \geq 14 \) and \( P(12) \land ... \land P(k) \)
- Can the team then score \( k + 1 \) points with field goals and touchdowns?
- If we can score \( k + 1 - 3 \), then just add a field goal
  - \( k + 1 - 3 = k - 2 \) is between 12 (because \( k \geq 14 \)) and \( k \)
  - So \( P(k + 1 - 3) \) holds by the inductive assumption
- Add a field goal to score \( k + 1 \) points: \( P(k + 1) \) holds
- **Done!**
Observations

• We had to reach back *three* values of \( k \), not just one
• If \( k + 1 - 3 \) hadn’t worked, we could have tried \( k + 1 - 7 \)
• Needed three base cases because we reach three values back:
  \[
  P(12) \rightarrow P(15) \\
  P(13) \rightarrow P(16) \\
  P(14) \rightarrow P(17) \\
  P(15) \rightarrow P(18) \\
  \vdots
  \]
Number Theory Example

- Every integer greater than 1 has a prime divisor
- \( \forall n \geq 2 \ \exists p \in \mathbb{N} : p \text{ is prime and } p \mid n \)
- Base case \( P(2) \): 2 is prime and \( 2 \mid 2 \)
- **Strong** inductive assumption: \( k \geq 2 \) and every integer \( i \) with \( 2 \leq i \leq k \) has a prime divisor
- Inductive step: Does \( k + 1 \) then have a prime divisor?
  - Case 1: \( k + 1 \) is prime. \( P(k + 1) \) same reasoning as \( P(2) \)
  - Case 2: \( k + 1 = ab \) with \( a, b \in \mathbb{N} \) and \( 2 \leq a, b \leq k \)
  - So in particular \( P(a) \) holds: \( \exists \text{ prime } u \in \mathbb{N} : u \mid a \)
  - That is, \( a = uv \) for some prime \( u \) and integer \( v \)
  - So \( k + 1 = ab = uvb \) and \( u \) is a prime divisor of \( k + 1 \)
- **Done!**