CSC236 *Intro. to* the Theory of Computation

Lecture 3: WOP, Structural *Induction*

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

recall

- use all resources available to you
  - before it becomes too late!
- what resources?
  - office Hours:
    - **M 2-3:30 in PT286C, W 2-4 BA4222, F 3:30-4:30 BA4270**
  - the course page and our section page
  - the CS Help Centre
  - the course forum
  - study groups and Peer Instruction
  - email ahchinaei @ cs.torotno.edu

review

- Week 01
  - Simple Induction
    - *AKA: Mathematical Induction or Principle of Mathematical Induction*
- Week 02
  - Strong Induction
    - *AKA: Complete Induction or Second Principle of Mathematical Induction*
- Over 30 examples
- Simple Ind and Strong Ind are equivalent
- This week
  - Well Ordering Principle
  - Structural Induction

review

- Simple Induction
  - it's a rule of inference:

$$\frac{\begin{array}{l} P(b) \\ P(k) \rightarrow P(k+1) \qquad \forall k \geq b \in \mathbb{N} \end{array}}{P(n) \qquad \forall n \geq b \in \mathbb{N}}$$

- Strong Induction
  - it's a rule of inference:

$$\frac{\begin{array}{l} P(b) \\ P(b) \wedge P(b+1) \wedge \ldots \wedge P(k) \rightarrow P(k+1) \qquad \forall k \geq b \in \mathbb{N} \end{array}}{P(n) \qquad \forall n \geq b \in \mathbb{N}}$$

review

- Simple Induction
  - *To show that all domino pieces fall over, we should show that*

    1) *there is a starting point,* *i.e.,* $P(b)$ *holds*

    *and **2)** all pieces are set in a well order such that*

    falling of piece k implies falling of piece k+1

    i.e., and $P(k) \rightarrow P(k+1)$ *holds too.*

- Strong Induction
  - *To show that all domino pieces fall over, we should show that*

    1) *there is a starting point,* *i.e.,* $P(b)$ *holds*

    *and **2)** all pieces are set in a well order such that*

    falling of all pieces to k implies falling of piece k+1

    i.e., and $(P(b) \wedge \ldots \wedge P(k)) \rightarrow P(k+1)$ *holds too.*

well-ordering principle: wop

- simple induction and strong induction are valid because of the well-ordering property:
- **WOP:** every nonempty subset of natural numbers has a minimum element.

Example 30: division algorithm

$P(n)$: if $n, d \neq 0 \in \mathbb{N}$, there are unique q and $r \in \mathbb{N}$ where $0 \leq r < d$, such that $n = dq + r$.

scratch work

n, d	*dq+r*	P(n)

Example 30: P(n): for any $d \neq 0 \in \mathbb{N}$, there are $q, r \in \mathbb{N}$, such that $n=dq+r$ and $0 \leq r<d$.

Proof by W.O.P.

- Let S be a subset of natural numbers of the form n-dq *where* $q \in \mathbb{N}$.
- S is nonempty because q can be as low as 0, *i.e.*, n is always in S.
- By the well-ordering property:
  - S has a least element, let's call it r, where $r = n-dq_0$
  - $r \geq 0$ because $r \in S \subseteq \mathbb{N}$.
  - and $r<d$; {otherwise, $r \geq d$

    $$n-dq_0 \geq d \Rightarrow n-d(q_0+1) \geq 0$$

    Let $r'=n-d(q_0+1)$, obviously $r'<r$ which contradicts r being the least element; hence, $r<d$. }
- Hence, there are q and $r \in \mathbb{N}$, such that $n=dq+r$ *and* $0 \leq r<d$.

□

Example 30: *uniqueness*

- so far, we proved q and $r \in \mathbb{N}$ exists such that $n=dq+r$ *and* $0 \leq r < d$
- proving q and r are unique does not require induction (or W.O.P).
- Proof by contradiction.
- Assume q and r are not unique, *i.e.*, there are $q`$ and $r` \in \mathbb{N}$ such that $n=dq`+r`$ and $0 \leq r` < d$

$$\Rightarrow dq`+r`=dq+r \qquad 1$$

$$\Rightarrow (q`-q)d=r-r` \qquad 2$$

- W.L.O.G, assume $q` \geq q$:
  - If $q`>q \Rightarrow q`-q>0 \Rightarrow q`-q \geq 1 \Rightarrow (q`-q)d \geq d \overset{by\ 2}{\Longrightarrow} r-r` \geq d \Rightarrow r \geq d+r`$ which is contradiction.
- Hence, $q`=q$ and $\overset{by\ 1}{\Longrightarrow} r=r`$ too. □

Example 31: cycles in round-robin tournaments

$P(n)$: if there is a cycle in a rrt, there is a cycle of 3.

scratch work

Example 31:

Example 31:

notes:

- Simple Ind, Strong Ind, and WOP are all equivalent.

inductive sets and structures

- If sets—and other structures—can be defined inductively (recursively), then
- their properties
  - can be implemented with recursive algorithms, and
  - can be proved with induction.

- ***Inductive definitions* of sets have two parts:**
  - The ***basis step*** specifies an initial collection of elements.
  - The ***recursive step*** specifies rules to form new elements in the set from those already known to be in the set.

define sets, inductively

- **Example 32:** the set of natural numbers, $\mathbb{N}$:
  Basis Step: $0 \in \mathbb{N}$;
  Recursive Step: If n is in $\mathbb{N}$, then $n + 1$ is in $\mathbb{N}$.

- **Example 33:** the set S of natural numbers of multiples of 3:
  Basis Step: $3 \in S$;
  Recursive Step: …

define sets, inductively

- **Example 34**, strings: the set Σ^* over the alphabet Σ:
  Basis Step: $\lambda \in \Sigma^*$ (λ is the empty string);
  Recursive Step: if w is in Σ^* and x is in Σ, then $wx \in \Sigma^*$.
- **Example 34'**, binary strings:
  if $\Sigma = \{0,1\}$, the strings in in Σ^* are the set of all binary strings, such as λ,0,1, 00,01,10, 11, etc.
- **Example 34''**, on trinary strings: if $\Sigma = \{a,b,c\}$, show that *aac* is in Σ^*.
  -
  -
  -

define properties, inductively

- **Example 35**, length of strings:

  Basis Step: $l(\lambda) = 0$;

  Recursive Step: $l(wx) = l(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$.

another example:

- **Example 36**, set of balanced strings, P:

  Basis Step: **()** $\in P$;

  Recursive Step:

  if $w \in P$, then **()*w*** $\in P$, **(*w*)** $\in P$ and ***w*()** $\in P$.

- show that (() ()) is in P.

- why is))(() not in P?

Example 37: FBT

- **Basis Step:**

  There is a full binary tree consisting of only a single vertex *r*;

- **Recursive Step:**

  If T_1 and T_2 are disjoint full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root *r* together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2.

forming FBTs

- Basis Step
- Step 1
- Step 2

Example 38: height of FBT

- The *height*, $h(T)$, of a full binary tree T can be defined as:
  - **Basis Step:** the height of a full binary tree T consisting of only a root r is $h(T) = 0$;
  - **Recursive Step:** if T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + \max(h(T_1), h(T_2))$.

Example 39: # of nodes

- The number of vertices, $n(T)$, of a full binary tree T can be defined as:
  - **Basis Step:** the number of vertices of a full binary tree T consisting of only a root r is $n(T) = 1$;
  - **Recursive Step:** if T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has the number of vertices $n(T) = 1 + n(T_1) + n(T_2)$.

proof by structural induction

- **Recipe:**
  - To prove a property, *P*, of the elements of a recursively defined structure holds, we should demonstrate these steps:
    - ***Proof Method:*** "structural induction"
    - **Basis Step:** show that *P* holds for all elements specified in the basis step of the structure definition.
    - **Inductive Step:** show that if *P* holds for each of the elements used to construct new elements, P holds for the new elements too.

The validity of structural induction can be shown to follow from simple induction

Example 40:

Theorem: if T is a FBT, then $n(T) \leq 2^{h(T)+1} - 1$.

scratch work

Example 40:

- **Proof Method**: structural induction.
- **Basis Step:**
- **Inductive Step:**

Example 40:

Example 41: set of simple expression, ε

Definition: ε

Basis Step: $x, y, z \in \varepsilon$

Inductive Step: $e_1, e_2 \in \varepsilon \Rightarrow (e_1+e_2)$ and $(e_1 x e_2) \in \varepsilon$

Prove $\forall e \in \varepsilon$, $vr(e)=op(e)+1$,

where $vr(e)$ denotes the # of variables and $op(e)$ denotes the # of operators in e.

Example 41:

Example 41:

notes: