

## C241 Homework 9

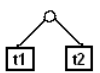
Due Wednesday, 11/10/09

1) Complete the following two structural induction proofs which do not involve recursive functions.

a) Prove that all binary trees have an odd number of nodes.

Base Case  $\circ$ : This has one node, and one's an odd number, so the base case is valid.

Induction Hypothesis: Let's say that  $t_1$  has  $n$  nodes, and  $t_2$  has  $m$  nodes. Assume  $n$  and  $m$  are both odd numbers.

Induction Step: Then show that  also has an odd number of nodes. Well, this tree will have all the nodes from  $t_1$  and  $t_2$ , along with the new root, so it'll have  $m + n + 1$  nodes. Since  $m$  and  $n$  are both odd,  $m + n$  will be even, and thus  $m + n + 1$  will be odd. So this tree will have an odd number of nodes.

b) Prove that all strings in the language  $P$  defined below have an equal number of left and right parentheses.

- $P$
- 
1.  $() \in P$
  - 2a.  $u \in P \Rightarrow (u) \in P$
  - 2b.  $u, v \in P \Rightarrow (uv) \in P$
  3. There is nothing else in  $P$ .

Let's use the variable  $L_s$  to mean the number of left parentheses in the string  $s$ , and  $R_s$  to mean the number of right parentheses in the string  $s$ .

Base Case  $()$ : Since  $L_{()} = 1$  (it has one left parenthesis) and  $R_{()} = 1$  (it has one right parenthesis), and  $1 = 1$ , the base case has an equal number of left and right parentheses.

Induction Steps:

Show that, if  $u$  has an equal number of left and right parentheses (in other words, if  $L_u = R_u$ ) (Induction Hypothesis), then  $(u)$  has an equal number of left and right parentheses. Well  $L_{(u)} = L_u + 1$  and

$R_{(u)} = R_u + 1$ . And, since  $L_u = R_u$  (IH), then it's true that  $L_u + 1 = R_u + 1$ . (In other words, if  $u$  has an equal number of left and right parentheses, and we add one right and one left parenthesis, the result still has an equal number of left and right parentheses)

Show that, if  $u$  and  $v$  both have an equal number of left and right parentheses (in other words, if  $L_u = R_u$  and  $L_v = R_v$ ) (Induction Hypothesis), then  $(uv)$  has an equal number of left and right parentheses. Well  $L_{(uv)} = L_u + L_v + 1$  and  $R_{(uv)} = R_u + R_v + 1$ . And, since  $L_u = R_u$  and  $L_v = R_v$  (IH), then it's true that  $L_u + L_v + 1 = R_u + R_v + 1$ .

**2) Use the definition of the language  $L$ , and the recursive functions  $C : L \rightarrow \mathbb{N}$ ,  $R : L \rightarrow L$ , and  $A : L \times L \rightarrow L$  to complete the following structural induction proofs**

|                                    |
|------------------------------------|
| $L(\{a, b\}^*)$                    |
| 1. $\epsilon \in L$                |
| 2a. $s \in L \Rightarrow as \in L$ |
| 2b. $s \in L \Rightarrow bs \in L$ |
| 3. There is nothing else in $L$ .  |

|                             |
|-----------------------------|
| $R : L \rightarrow L$       |
| 1. $R(\epsilon) = \epsilon$ |
| 2a. $R(as) = R(s)a$         |
| 2b. $R(bs) = R(s)b$         |

|                                |
|--------------------------------|
| $C : L \rightarrow \mathbb{N}$ |
| 1. $C(\epsilon) = 0$           |
| 2a. $C(as) = 1 + C(s)$         |
| 2b. $C(bs) = C(s)$             |

|                                |
|--------------------------------|
| $A : L \times L \rightarrow L$ |
| 1. $A(\epsilon, v) = v$        |
| 2a. $A(au, v) = aA(u, v)$      |
| 2b. $A(bu, v) = bA(u, v)$      |

**Extra Credit) Use structural induction to prove that  $R(R(s)) = s$ ,  $\forall s \in L$ . This involves first proving that  $\forall s \in L$ ,  $R(sa) = aR(s)$ , and similarly,  $R(sb) = bR(s)$ .**

First we'll show  $R(sa) = aR(s)$  (The proof for  $R(sb) = bR(s)$  is similar). We need to do this because the function definition for  $R$  only has a rule defined for  $R(as)$ ; we might guess what behavior it will have with  $R(sa)$ , but we can't put that guess in a proof unless we can prove it.

Prove that  $R(sa) = aR(s)$

Base Case: Show that  $R(\epsilon a) = aR(\epsilon)$

$$R(\epsilon a) \stackrel{?}{=} aR(\epsilon)$$

$$R(a) \stackrel{?}{=} a\epsilon$$

$$R(a\epsilon) \stackrel{?}{=} a$$

$$aR(\epsilon) \stackrel{?}{=} a$$

$$a\epsilon \stackrel{?}{=} a$$

$$a = a$$

Induction Hypothesis: Assume that for some  $s \in R$ ,  $R(sa) = aR(s)$

Induction Steps: Show that  $R(asa) = aR(as)$  and  $R(bsa) = aR(bs)$

We'll just show the  $R(asa) = aR(as)$  case, the  $R(bsa) = aR(bs)$  case is very similar.

$$R(asa) \stackrel{?}{=} aR(as)$$

$$R(sa)a \stackrel{?}{=} aR(as)$$

$$R(sa)a \stackrel{?}{=} aR(s)a$$

$$aR(s)a = aR(s)a \text{ Induction Hypothesis}$$

Ok, now we're ready for the  $R(R(s)) = s$  proof.

Base Case:  $R(R(\epsilon)) \stackrel{?}{=} \epsilon$

$$R(R(\epsilon)) \stackrel{?}{=} \epsilon$$

$$R(\epsilon) \stackrel{?}{=} \epsilon$$

$$\epsilon \stackrel{?}{=} \epsilon$$

Induction Hypothesis: Assume that for some  $s \in R$ ,  $R(R(s)) = s$

Induction Steps: Show that  $R(R(as)) = as$  and  $R(R(bs)) = bs$

We'll just show the  $R(R(as)) = as$  case, the  $R(R(bs)) = bs$  case is very similar.

$$R(R(as)) \stackrel{?}{=} as$$

$$R(R(s)a) \stackrel{?}{=} as$$

$aR(R(s)) \stackrel{?}{=} as$  Using our lemma,  $\forall s \in L, R(sa) = aR(s)$

$as = as$  Induction Hypothesis

**b) Write a recursive function  $D : L \rightarrow L$  which doubles the occurrences of  $a$  in a string (so  $D(abab) = aabaab$ ). Your function should be written in a similar style to the ones defined above.**

$$\frac{D : L \rightarrow \mathbb{N}}{\begin{array}{l} 1. \quad D(\epsilon) = \epsilon \\ 2a. \quad D(as) = aaD(s) \\ 2b. \quad D(bs) = bD(s) \end{array}}$$

**c) Use structural induction to prove that  $C(D(s)) = 2 \times C(s)$ ,  $\forall s \in L$**

Base Case:  $C(D(\epsilon)) \stackrel{?}{=} 2 \times C(\epsilon)$

$C(D(\epsilon)) \stackrel{?}{=} 2 \times C(\epsilon)$

$C(\epsilon) \stackrel{?}{=} 2 \times C(\epsilon)$

$0 \stackrel{?}{=} 2 \times 0$

$0 = 0$

Induction Hypothesis: Assume that for some  $s \in L$ ,  $C(D(s)) = 2 \times C(s)$

Induction Step: Then show that this means  $C(D(as)) = 2 \times C(as)$  and  $C(D(bs)) = 2 \times C(bs)$ . Here we must do both cases separately, because  $D$  treats  $a$  and  $b$  differently, ***the two cases are not similar.***

Show  $C(D(as)) = 2 \times C(as)$

$C(D(as)) \stackrel{?}{=} 2 \times C(as)$

$C(aaD(s)) \stackrel{?}{=} 2 \times (1 + C(s))$

$C(aaD(s)) \stackrel{?}{=} 2 + 2 \times C(s)$

$1 + C(aD(s)) \stackrel{?}{=} 2 + 2 \times C(s)$

$1 + 1 + C(D(s)) \stackrel{?}{=} 2 + 2 \times C(s)$

$2 + C(D(s)) \stackrel{?}{=} 2 + 2 \times C(s)$

$C(D(s)) = 2 \times C(s)$  True, by induction hypothesis.

Show  $C(D(bs)) = 2 \times C(bs)$

$C(D(bs)) \stackrel{?}{=} 2 \times C(bs)$

$C(bD(s)) \stackrel{?}{=} 2 \times C(s)$

$C(D(s)) = 2 \times C(s)$  True, by induction hypothesis.

d) Using structural induction prove that **A** is associative: ie, prove that for any three strings  $x, y, z \in L$   $A(x, A(y, z)) = A(A(x, y), z)$ . Do this by doing induction just on  $x$ , leaving  $y$  and  $z$  as variables which represent any arbitrary strings in  $L$ . This means that for your base case you'll use  $A(\epsilon, A(y, z)) \stackrel{?}{=} A(A(\epsilon, y), z)$ . For your induction hypothesis you'll assume that  $A(x, A(y, z)) = A(A(x, y), z)$  for some  $x \in L$  and for all  $y, z \in L$ . And, for your induction steps you'll show that  $A(ax, A(y, z)) = A(A(ax, y), z)$  and  $A(bx, A(y, z)) = A(A(bx, y), z)$

**Base Case**  $x = \epsilon, y, z \in L$ :

$$A(\epsilon, A(y, z)) \stackrel{?}{=} A(A(\epsilon, y), z)$$

$$A(y, z) \stackrel{?}{=} A(A(\epsilon, y), z)$$

**rule A.1**

$$A(y, z) = A(y, z)$$

**rule A.1, The base case holds.**

**Induction Hypothesis:** Assume that for some  $x$  and *all*  $y, z$  in  $L$ :  
 $A(x, A(y, z)) = A(A(x, y), z)$

**First Induction Step:**  $ax$

$$A(ax, A(y, z)) \stackrel{?}{=} A(A(ax, y), z)$$

$$aA(x, A(y, z)) \stackrel{?}{=} A(A(ax, y), z)$$

$$aA(x, A(y, z)) \stackrel{?}{=} A(aA(x, y), z)$$

$$aA(x, A(y, z)) \stackrel{?}{=} aA(A(x, y), z)$$

$$aA(A(x, y), z) = aA(A(x, y), z)$$

**induction hypothesis**

**Second Induction Step:**  $bx$

This is basically the same as the first induction step.

(e) Create your own recursive function over  $L$ , and use structural induction to prove some property your function has (in other words, write your own problem in the style of the problems above, and solve it). Please include a plain english description of what your function is intended to do.

$F$  defined below adds an extra  $a$  to the end of the string  $s$ . So  $F(bb) = bF(b) = bbF(\epsilon) = bba$ .

$$\frac{F : L \rightarrow \mathbb{N}}{\begin{array}{l} 1. \quad F(\epsilon) = a \\ 2a. \quad F(as) = aF(s) \\ 2b. \quad F(bs) = bF(s) \end{array}}$$

We'll prove that  $C(F(s)) = C(s) + 1$

Base Case:  $C(F(\epsilon)) \stackrel{?}{=} C(\epsilon) + 1$

$$C(F(\epsilon)) \stackrel{?}{=} C(\epsilon) + 1$$

$$C(a) \stackrel{?}{=} 0 + 1$$

$$1 + C(\epsilon) \stackrel{?}{=} 0 + 1$$

$$1 + 0 = 0 + 1$$

Induction Hypothesis: Assume that for some  $s \in L$ ,  $C(F(s)) = C(s) + 1$

Induction Steps: Then show that  $C(F(as)) = C(as) + 1$  and  $C(F(bs)) = C(bs) + 1$

First,  $C(F(as)) \stackrel{?}{=} C(as) + 1$

$$C(aF(s)) \stackrel{?}{=} 1 + C(s) + 1$$

$$1 + C(F(s)) \stackrel{?}{=} 1 + C(s) + 1$$

$C(F(s)) = C(s) + 1$  True by induction hypothesis.

The proof for  $C(F(bs)) = C(bs) + 1$  is very similar.

4) Below are a couple general problems, Shortest Job First and Interior Angles, which are solvable using *numerical* induction. To get full credit for each problem, all you need to do is correctly set up the proof, you don't need to complete it: For each problem first identify the property  $P(n)$  that needs to be proved, and then copy out the Numerical Induction Outline below, filling in the correct values for  $P(0), P(n)$ , and  $P(n + 1)$ . If you go on to correctly complete a proof, you will get extra credit. The first problem has been done as an example.

To prove that a property  $\mathbf{P}(n)$  is true  $\forall n \in \mathbb{N}$  such that  $n \geq b$  (in other words,  $b$  will be our base case), we need to prove the following:

**Numerical Induction Outline**

**Base Case:** Show that  $[\mathbf{P}(b)]$  is true

**Induction Hypothesis:** Assume that  $[\mathbf{P}(n)]$  is true for some  $n \geq b$

**Induction Step:** Show that  $[\mathbf{P}(n+1)]$  is also true

**Example Problem:** DeMorgan's Law states that  $\neg(q_1 \vee q_2) \equiv (\neg q_1 \wedge \neg q_2)$ . Prove that DeMorgan's law is also valid for  $n$  variables: in other words  $\neg(q_1 \vee q_2 \dots \vee q_n) \equiv (\neg q_1 \wedge \neg q_2 \dots \wedge \neg q_n)$  is true for any  $n \geq 2$ .

**Example Solution:**

$\mathbf{P}(n)$ : " $\neg(q_1 \vee q_2 \dots \vee q_n) \equiv (\neg q_1 \wedge \neg q_2 \dots \wedge \neg q_n)$ "

**Numerical Induction Outline**

Base Case: Show that  $\neg(q_1 \vee q_2) \equiv (\neg q_1 \wedge \neg q_2)$  is true

Induction Hypothesis: Assume that  $\neg(q_1 \vee q_2 \dots \vee q_n) \equiv (\neg q_1 \wedge \neg q_2 \dots \wedge \neg q_n)$  is true for some  $n \geq 2$

Induction Step: Show that  $\neg(q_1 \vee q_2 \dots \vee q_n \vee q_{n+1}) \equiv (\neg q_1 \wedge \neg q_2 \dots \wedge \neg q_n \wedge \neg q_{n+1})$  is also true.

**Shortest Job First:** Let's say I have a set of "jobs" (processes) to run on a processor (which can only run one job at a time), and each job takes a certain amount of time to run. We'll simplify things by assuming that the amount of time necessary is always an integer (so job  $j$  might take 3 time units to run, and job  $k$  might take 5 units). Then the "turn-around" time of a job is the time at which the job is finished running, given that the first job starts at time zero. So

if we run job  $j$  first, its turn-around time will be 3. If we run  $k$  first and then  $j$ , the turn around time for  $k$  will be 5, and the turn-around time for  $j$  will be 8. So the average turnaround time for  $j$  and  $k$  would be  $(5 + 8)/2 = 6.5$ . If I run  $j$  first and then  $k$ , the turn-around time for  $j$  will be 3,  $k$  will be 8, and their average will be  $(8 + 3)/2 = 5.5$ . Choosing the order to run the jobs in is called "scheduling", and we'd like to pick an order which minimizes the turn-around time of the jobs we're running. The "shortest job first" scheduling algorithm puts the jobs in increasing order by the amount of time it takes to run them (so it would run  $j$  before  $k$ ). Prove that for any set of  $n \geq 1$  jobs, the shortest-job-first algorithm will produce the smallest possible average turn-around time for the  $n$  jobs.

**Proof Outline:**

P(n): "For any set of  $n$  jobs, the shortest-job-first algorithm will produce the smallest possible average turn-around time for those jobs."

Numerical Induction Outline

Base Case: Show that For any single job, the shortest-job-first algorithm will produce the smallest possible average turn-around time for that job.

Induction Hypothesis: Assume for some value of  $n$  that, For any set of  $n$  jobs, the shortest-job-first algorithm will produce the smallest possible average turn-around time for those jobs.

Induction Step: Show that For any set of  $n + 1$  jobs, the shortest-job-first algorithm will produce the smallest possible average turn-around time for those jobs.

(the trick here is to show that shortest job first will include the slowest jobs in the fewest number of turn-around time summations)

**Interior Angles:**) The interior angles of a polygon are essentially the angles of the 'corners' of the polygon (the angles that point towards the interior of the polygon). For instance, in an equilateral triangle, each of the three corners (interior angles) has an angle of 60 degrees. So the three interior angles together sum to 180 degrees. In fact, this is true for all triangles, the sum of the interior angles of *any* triangle is always 180 degrees. And its not the only polygon we can make such a claim about. Prove that for all  $n \geq 3$ , the sum of the interior angles of a polygon with  $n$  sides will be  $180 \times (n - 2)$ .

(hint, for those attempting to complete the proof: you can divide a square into two triangles).

**Proof Outline:**

$P(n)$ : "The sum of the interior angles of a polygon with  $n$  sides will be  $180 \times (n - 2)$ ."

Numerical Induction Outline

Base Case: The sum of the interior angles of a polygon with 3 sides will be  $180 \times (3 - 2)$ . (In other words, the sum of the interior angles of a triangle is 180).

Induction Hypothesis: Assume for some value of  $n$  that, the sum of the interior angles of a polygon with  $n$  sides will be  $180 \times (n - 2)$ .

Induction Step: Show that the sum of the interior angles of a polygon with  $n + 1$  sides will be  $180 \times ((n + 1) - 2)$ .

(the trick here is to show that adding any point to a polygon with  $n$  sides is like replacing one of the sides with a triangle)