

## C241 Homework 7: Solutions

1) Using only the Algebraic Laws of Logic and Induction prove the following:

a) DeMorgans for  $n$  variables:  $\neg(p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_n) = (\neg p_1 \vee \neg p_2 \vee \neg p_3 \vee \dots \vee \neg p_n)$ .

Base Case (n = 2):  $\neg(p_1 \wedge p_2) \equiv (\neg p_1 \vee \neg p_2)$

This is just the normal DeMorgan's Law, we know it's true because it's one of the Laws of Logic.

Induction Hypothesis: Assume that DeMorgan's Laws work for  $n$  variables, so assume:  $\neg(p_1 \wedge p_2 \wedge p_3 \dots \wedge p_n) \equiv (\neg p_1 \vee \neg p_2 \vee \neg p_3 \dots \vee \neg p_n)$

Induction Step (n + 1): If DeMorgan's Laws hold for  $n$  variables, do they hold for  $n + 1$  variables?

$$\begin{aligned} & \neg(p_1 \wedge p_2 \wedge p_3 \dots \wedge p_n \wedge p_{n+1}) \\ & \equiv \neg((p_1 \wedge p_2 \wedge p_3 \dots \wedge p_n) \wedge (p_{n+1})) \text{ Associative Law} \\ & \equiv \neg(p_1 \wedge p_2 \wedge p_3 \dots \wedge p_n) \vee \neg(p_{n+1}) \text{ DeMorgan's (the two variable version)} \\ & \equiv (\neg p_1 \vee \neg p_2 \vee \neg p_3 \dots \vee \neg p_n) \vee \neg(p_{n+1}) \text{ This was our Induction Hypothesis.} \\ & \equiv (\neg p_1 \vee \neg p_2 \vee \neg p_3 \dots \vee \neg p_n \vee \neg p_{n+1}) \text{ Associative Law} \end{aligned}$$

b) Distributive property for  $n$  variables:  $p \wedge (q_1 \vee q_2 \vee q_3 \vee \dots \vee q_n) = (p \wedge q_1) \vee (p \wedge q_2) \vee (p \wedge q_3) \vee \dots \vee (p \wedge q_n)$

Base Case (n = 2):  $p \wedge (q_1 \vee q_2) \equiv (p \wedge q_1) \vee (p \wedge q_2)$

This is just the normal two variable Distributive Law.

Induction Hypothesis: Assume that the Distributive Law works for  $n$  variables, so assume:  $p \wedge (q_1 \vee q_2 \vee q_3 \vee \dots \vee q_n) = (p \wedge q_1) \vee (p \wedge q_2) \vee (p \wedge q_3) \vee \dots \vee (p \wedge q_n)$

Induction Step ( $n + 1$ ): If the Distributive Law works for  $n$  variables, does it work for  $n + 1$ ?

$$\begin{aligned}
 & p \wedge (q_1 \vee q_2 \dots \vee q_n \vee q_{n+1}) \\
 \equiv & p \wedge ((q_1 \vee q_2 \dots \vee q_n) \vee q_{n+1}) \text{ Associative Law} \\
 \equiv & (p \wedge (q_1 \vee q_2 \dots \vee q_n)) \vee (p \wedge q_{n+1}) \text{ Distributive Law (the two variable version)} \\
 \equiv & ((p \wedge q_1) \vee (p \wedge q_2) \dots \vee (p \wedge q_n)) \vee (p \wedge q_{n+1}) \text{ This was our Induction Hypothesis.} \\
 \equiv & (p \wedge q_1) \vee (p \wedge q_2) \dots \vee (p \wedge q_n) \vee (p \wedge q_{n+1}) \text{ Associative Law}
 \end{aligned}$$

2) Use the language **P** defined below to answer the following questions.

$$\frac{P \subset V^+ \text{ with } V = \{(\,,)\}}{\quad}$$

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**.

a) Which of these strings are in the language **P**?

- |                               |                          |
|-------------------------------|--------------------------|
| ok (i) $(( ( ) ) )$           | ok (iv) $(( ) ( ) )$     |
| no (ii) $( ) ( )$             | no (v) $(( ) ( ) ( ) )$  |
| ok (iii) $(( ( ) ( ) ) ( ) )$ | ok (vi) $(( ( ) ( ) ) )$ |

b) Using structural induction, prove that every element in **P** has the same number of (‘s as it has )’s. It may help to use variables like  $L_u$  and  $R_u$  to represent how many left and right parentheses there are in string  $u$ .

Let  $L_s$  be the number of left parentheses in string  $s$ , and  $R_s$  be the number of right parentheses in string  $s$ . (You must explicitly state what variables like this represent before you can use them).

**Base Case  $u = ()$ :**

$$L_{()} = 1 = R_{()} \quad \text{The base case holds.}$$

**Induction Hypothesis:**  $L_u = R_u$  and  $L_v = R_v$

(we need both since both  $u$  and  $v$  are used as inputs to constructors)

**First Induction Step: (u)**

(you need an induction step for each constructor)

$$L_{(u)} \stackrel{?}{=} R_{(u)}$$

$$L_u + 1 \stackrel{?}{=} R_u + 1$$

$$R_u + 1 = R_u + 1$$

common sense

induction hypothesis

**Second Induction Step: (u v)**

$$L_{(uv)} \stackrel{?}{=} R_{(uv)}$$

$$L_u + L_v + 1 \stackrel{?}{=} R_u + R_v + 1$$

$$R_u + R_v + 1 = R_u + R_v + 1$$

common sense

induction hypotheses

**3) Below is the definition for the language  $L$  and two recursive functions on it. Use these definitions to answer the following questions.**

$$\frac{L \subset V^+ \text{ with } V = \{a, b, \bullet\}}{}$$

1.  $\bullet \in L$
- 2a.  $u \in L \Rightarrow au \in L$
- 2b.  $u \in L \Rightarrow bu \in L$
3. There is nothing else in  $L$ .

$$\frac{I : L \rightarrow L}{}$$

1.  $I(\bullet) = \bullet$
- 2a.  $I(au) = bI(u)$
- 2b.  $I(bu) = aI(u)$

$$\frac{D : L \rightarrow L}{}$$

1.  $D(\bullet) = \bullet$
- 2a.  $D(au) = aaD(u)$
- 2b.  $D(bu) = bbD(u)$

a) Find the strings output by the following functions. *Write out each step of the recursion.*

(i)  $I(aabb\bullet)$

$bI(abb\bullet)$

$bbI(bb\bullet)$

$bbaI(b\bullet)$

$bbaaI(\bullet)$

$bbaa\bullet$

ii)  $D(bba\bullet)$

$bbD(ba\bullet)$

$bbbbD(a\bullet)$

$bbbbaaD(\bullet)$

$bbbbaa\bullet$

b) Using structural induction prove that for every string  $u \in L$ ,  $I(I(u)) = u$ .

**Base Case**  $u = \bullet$ :

$I(I(\bullet)) \stackrel{?}{=} \bullet$

$I(\bullet) \stackrel{?}{=} \bullet$                       rule I.1

$\bullet = \bullet$                                       rule I.1 The base case holds.

**Induction Hypothesis:**  $I(I(u)) = u$

(we only need one, since  $u$  is the only input to the constructors 2a and 2b of L)

**First Induction Step:** Show that  $I(I(au)) = au$

(we need two, since there's two constructors)

$I(I(au)) \stackrel{?}{=} au$

$I(bI(u)) \stackrel{?}{=} au$

rule I.2a

$aI(I(u)) \stackrel{?}{=} au$

rule I.2b

$au = au$

induction hypothesis

**Second Induction Step:** Show that  $I(I(bu)) = bu$

$I(I(bu)) \stackrel{?}{=} bu$

$I(aI(u)) \stackrel{?}{=} bu$

rule I.2a

$bI(I(u)) \stackrel{?}{=} bu$

rule I.2b

$bu = bu$

induction hypothesis

c) Using structural induction prove that for every string  $u \in L$ ,  $D(u)$  contains an even number of  $a$ 's (of course, 0 is considered an even number).

**Base Case  $u = \bullet$ :**

$D(\bullet) = \bullet$  rule D.1

$\bullet$  has 0  $a$ 's

The base case holds.

**Induction Hypothesis:** Assume  $D(u)$  has an even number of  $a$ 's

(we only need one, since  $u$  is the only input to the constructors 2a and 2b of L)

**First Induction Step:** Show that  $D(au)$  has an even number of  $a$ 's

(we need two, since there's two constructors)

$D(au) = aaD(u)$  by rule D.2a

$aaD(u)$  has 2  $a$ 's, plus the number of  $a$ 's in  $D(u)$ , which, by our induction hypothesis, is even. So, since an even number plus two is an even number,  $aaD(u)$  has an even number of  $a$ 's.

**Second Induction Step:** Show that  $D(bu)$  has an even number of  $a$ 's

$D(bu) = bbD(u)$  by rule D.2a

$bbD(u)$  has the same number of  $a$ 's as  $D(u)$ , which, by our induction hypothesis, is even. So,  $bbD(u)$  has an even number of  $a$ 's.

4) A binary tree is a tree where each node either has 0 children (a leaf) or exactly 2 children. Prove that all binary trees have an odd number of nodes.

**Base Case:** a single node (a root). Well, a single node *is* only one node, and 1 is an odd number. So our base case has an odd number of nodes.

**Induction Step:**

Induction hypothesis: let's assume that we have two binary trees,  $T_1$  and  $T_2$  that each has an an odd number of nodes. (We'll say  $T_1$  has  $n_1$  nodes and  $T_2$  has  $n_2$  nodes).

Then, can we show that there's an odd number of nodes in the binary tree made by joining  $T_1$  and  $T_2$  together with a new root node? Well, the new tree has all the nodes of  $T_1$ , and all the nodes of  $T_2$ , along with the new root node we added when we joined the trees together. So that's a total of  $n_1 + n_2 + 1$  nodes. The sum of two odd numbers is an even number, so  $n_1 + n_2$  is even (by our induction hypothesis  $n_1, n_2$  are both odd). And an even number plus one is an odd number, so  $n_1 + n_2 + 1$  is odd, and the new tree does have an odd number of nodes.

More formally: since  $n_1, n_2$  are odd, we can write them as  $n_1 = 2m_1 + 1$  and  $n_2 = 2m_2 + 1$ . Then  $n_1 + n_2 + 1 = 2m_1 + 1 + 2m_2 + 1 + 1 = 2(m_1 + m_2 + 1) + 1$  which is an odd number.