Assignment 2

Karan Sikka
ksikka@andrew.cmu.edu
E
February 2, 2012

Notice that we start at problem 0

---

a. Let a_n be a sequence where $a_0 = 1$, $a_1 = 4$, $a_n = 4a_{n-1} - 4a_{n-2}$.

Claim: $P(n) =$ "$a_n = (n+1)2^n \forall n \in \mathbb{N}, n \geq 2$"

Base Case ($n = 2$):
$a_2 = 4a_1 - 4a_0 = 4 * 4 - 4 = 12$
$a_2 = (2+1)2^2 = 3 * 4 = 12$

Inductive Hypothesis: $P(2) \vee ... \vee P(k)$ hold for some $k \in \mathbb{N}, k \geq 2$.

Inductive Step:
$a_{n+1} = 4a_n - 4a_{n-1}$
$a_{n+1} = 2^2(n+1)2^n - 2^2(n)2^{n-1}$
$a_{n+1} = (2n+2)2^{n+1} - (n)2^{n+1}$
$a_{n+1} = (2n+2-n)2^{n+1}$
$a_{n+1} = (n+2)2^{n+1}$

We have shown that $P(2)$ holds true, and that $P(2) \vee ... \vee P(k) \implies P(k+1)$. Therefore, by course of values induction, $P(n)$ holds $\forall n \in \mathbb{N}$.

b. Let S be the sentence in propostional logic "$(((x \wedge y) \implies z) \implies (\neg y \implies \neg z))$"
Let x be true, y be true, and z be false.
then S is logically equivalent to "$(((true) \implies false) \implies (false \implies true))$"
which is logically equivalent to "$(false \implies true)$"
which is logically equivalent to "$true$". Since there exists a truth assignment such that S is true, S has been proven to be satisfiable.
Say we let x be false, y be false, and z be true.
Now S is logically equivalent to "$(((false) \implies true) \implies (true \implies false))$"
which is logically equivalent to "$(true \implies false)$"
which is logically equivalent to "$false$". Since there exists a truth assignment such that S is false, S, is not a tautology (not valid).

Theorem? I hardly knew him!

---

a. |||$|$||$|||is a theorem. Proof:

1	|$	axiom	
2	||$\|$|	SURROUND on 1 and 1	
3	|$|	END on 1	b.
4	|$||	END on 3	
5	|||$\|$|$|||	SURROUND on 2 and 4	

Claim: No theorems have a $ in the first index.
We proceed by structural induction on the length of the theorems.
Base Case:
The axiom |$ does not have a $ in the first index.

Inductive Hypothesis:
Any theorems x and y with lengths less than k for some $k \in \mathbb{N}, k \geq 2$ do not have a $ character in the first index.

Inductive Step:
Case 1: Apply END to x. The result is x|. Since x does not lead with a $, x|does not lead with a $.

Case 2: Apply SURROUND to x and y. The result is |xy|. This clearly does not lead with a $.

Therefore, no theorem leads with a $.

Claim: No theorem contains 2 adjacent $ characters.
We proceed by structural induction on the theorems in this axiomatic system.
Base Case:
The axiom |$ does not contain two adjacent $ characters.

Inductive Hypothesis:
Any theorems x and y with lengths less than k for some $k \in \mathbb{N}, k \geq 2$ do not have 2 adjacent $ characters.

Inductive Step:
Case 1: Apply END on x. The resulting theorem would be x|. Since x did not contain adjacent $ characters (I.H.), x|does not because by appending a |we have not changed the arrangement of the $ characters in any way.

Case 2: Apply SURROUND on x and y. The result is |xy|. Again, since this operation does not rearrage the $ characters within x and y in any way, and since x and y do not contain adjacent $ characters (I.H.), the only way for two $ characters to be adjacent to each other is if x had a $ at the last index and y had a $ on the first index. However, we previously proved that no theorem has a $ in the first index. Therefore, the result will not have adjacent $ characters.

Now consider ||$|||$$|. This cannot be a valid theorem since it has adjacent $ characters.
■

Teaching Assistant Assistance

(a)

$x \wedge y \iff (x?y)?(x?y)$
$x \vee y \iff (x?x)?(y?y)$
$x \rightarrow y \iff x?(x?y)$
$x \leftrightarrow y \iff (x?y)?((x?x)?(y?y))$

(b)
Cheer up Tim! L is not valid.
Consider the interpretation where $x, y \in \mathbb{N}$
isStudent(x) $\Leftrightarrow$ true
isTA(y) $\Leftrightarrow$ false
Prefers(x,y,Tim) $\Leftrightarrow$ true

In this interpretation, the sentence evaluates to
true $\rightarrow$ (false $\rightarrow$ true)
which evaluates to
true $\rightarrow$ false
which evaluates to
false.

$\exists a, b, c \wedge$IsStudent(a)$\wedge$IsStudent(b)$\wedge$IsTA(c)$\wedge a \neq b \wedge Prefers(a, Tim, c) \wedge Prefers(b, Tim, c)$

Dragon Chess

Some notation:
Given an n by n square, we will denote the location of a dragon by a tuple (x,y), where x and y are integer indexes $\in [1, n]$. x represents the index of the row and y represents the index of the column.

A method for placing dragons:
Place a dragon in every location of the first row except for the location (1,1). That is $n-1$ dragons. We can also put a dragon in every location of the first column except for the location (1,1). That is another $n-1$ dragons. Thus we have placed $2(n-1)$ dragons on the board.

Claim: No dragons are threatened in any arrangement formed using this method.

Proof: Suppose there are 2(n-1) dragons on an n by n board arranged using this method. Let d be any dragon on the board, and let its location be (x,y). d is threatened if and only if there

exists a dragon with location (*,y) or (x,*), where * is an integer. Since we put dragons in locations of the form (1,*) and (*,1), d is only threatened if * = 1. However, this is not the case, since this arrangement specifically avoids putting a dragon in (1,1). Therefore no dragons are threatened.

Since we have found a method which places dragons such that no dragons are threatened, we can say that there is a lower bound of D_n. The lower bound is $2(n-1)$ or $2n-2$, as counted above.

1. all subgrids of a dragon arragement which satisfies the condition, also satisfy the condition
More notation:
We say a dragon board is "FULL" if it contains D_n dragons. We say a dragon board is "full" if there is no spot on the board when another dragon can fit such that all remain unthreatened afterwards. We call a dragon board A a "subgrid" of a dragon board B if and only if A a collection of consecutive squares of from board B, and A is a square board.

You can't have x dragons where $x \geq 2n-1$. Base case check Inductive hypothesis: For all dragon boards with side-length less than k for some k in the natural numbers, you can't have 2n-1 or more dragons. Inductive step: Assume for sake of contradiction that a dragon board can have 2n-1 dragons or more.

Let's try to fit them on a k+1 dimension board!

It makes sense to fit as many as you can in a k by k board, and then try to fit the rest in the remaining space. So we'll make the k by k board FULL, to the lower bound we previously established.

Now there are 2(k-1) or 2k-2 dragons in the k by k square. By lemma, there is a dragon in every row and column of that square. 2k+1 - (2k-2) = 3. We have to fit 3 dragons in the remaining space. Not possible. Therefore this isn't possible so IS proven.

What if instead, I put fewer dragons there so that all the rows and columns weren't taken up? 2k+1 / 2 is at least k.

Post Puzzle Pancake Party

Let T_n be the minimum number of flips required to sort the worst-case n-stack of pancakes.
Since the pancakes come in 2 sizes, we will donate the small size as 0, and the large size as 1. We can disply a vertical stack of pancakes by a vertical stack of their numbers. For examples, [small,large,large,large] could be represented as
0
1
1
1

We can find a lower bound for T_n by using the breaking apart method. Consider a stack of n pancakes which alternate size:

1 0 1 0 ... 1 0 At every point where a 1 is next to a 0, we know the spatula must perform a flip to break them apart, because in the final sorted pancake sequence, 1's are adjacent and 0's are adjacent. There are $n-1$ such points in the n-stack of pancakes that alternate. There is 1 additional point between the bottom small pancake and the plate, which in math terms brings the tally of number of flips from n-1 to n. Then, we realize that there is one point in the final stack where a small pancake is next to a large pancake (ie 000.111). We account for that by subracting 1 flip from the number of required flips, which brings the tally of flips from n to $n-1$. Hence, given an alternating stack of pancakes, it will take at least $n-1$ flips to get it sorted by the breaking-apart argument.This establishes a lower bound on T_n.
The following is a method of sorting any n-sized pancake stack.
First, in a pancake stack, a consecutive sequence of 1 size of pancakes will be called a "chunk".
In sorting a pancake stack, the goal is to get all the large pancakes to be consecutive and all the small pancakes to be consecutive on top of the small pancakes. In other terminology, the ultimate goal is to have a chunk of small pancakes on top of a chunk of large pancakes.
The method is as follows:

1. Scan from the top down, and place the spatula after the end of two chunks. Ie, in a stack like 110101010, you would scan past the first chunk (11) and then past the second chunk (0). Then flip, so that the first three pancakes are not 110 but 011. Then repeat.

The period represents the position of the spatula.
The first scan:
(11)(0).101010
then flip.
(0)(11).101010

The second scan:
(0)(111).01010
then flip.
(111)(0).01010

And so on.
Everytime a flip occurs, the number of chunks is decremented. This is because we stop the spatula at the point of discontinuity in the chunk sequence and perform a flip so that a chunk sequence such as AB.A becomes BA.A, which is a transition from 3 chunks to 2 chunks.

Therefore, the number of flips that this number requires is proportional to the initial number of chunks in the stack. In other words, it is inversely proportional to the number of consecutive sequences of 1's and 0's in the stack. Therefore, the worst case is one where each chunk is only 1 pancake long, or where the pancakes alternate (ie 1010101010). There are n chunks in the worst case n-sized pancake.

Since the goal is to reduce n chunks to 2 chunks (as initially stated), this will require n-2 flips.
Then, the last step will be to flip the stack so that the small pancakes are on top of the large pancakes. Ie. 11110000 $\mapsto$ 00001111. Therefore the tally of steps goes from n-2 to n-1 by adding 1

step. Since the worst case pancake stack of size n can be solved using this method in at most n-1 flips, n-1 is an upper bound on T_n

In conclusion, the worst-case n-stack must take at least n-1 flips to sort, by the breaking apart argument. Any n-stack takes at most n-1 flips using the method described above. Since $n-1 \leq T_n \leq n-1$, $T_n = n-1$. ■

The Ghost of Puzzle Hunts Past

The proof will be as follows:
1.. In this program, there is a point where x < b holds.
2. When x < base, n is 0.
3. If n becomes and stays 0, then x will decrement until $x \leq 0$, upon which the loop terminates.

First, we know that n is quotient when we do integer division of x by base, and m is the remainder for that division. Therefore, we can establish that x = n*base + m, by the properties of division.
Next, note that there is a line where base is doubled. Hence, for any expression after that line, we will replace the name "base" with "2*base" when considering it in a mathematical context.
Now let"s examine the line:
`n*base + m - 1`
Which, as discussed before, should be written as the following for mathmatical correctness:
`n*2*base + m - 1`
We can do some algebra: (b is the variable "base")
$x = n * 2b + m - 1 = 2nb + m - 1 = 2nb + 2m - m - 1$
$= 2(nb + m) - m - 1$
$x = 2x - m - 1$
$x = 2x - (m + 1)$
Now note that $0 \leq m < b$ due to the properties of the remainder in division.
Furthermore, we can add 1 to each of those terms to deduce:
$0 < m + 1 \leq b$
Now we know that $m + 1$ is strictly positive.

Going back to the line `base = 2 * base`, this tells us that an explicit form for the base at the k^{th} iteration of the loop is $b_k = 2^k$.
To get an explicit form for x at the k^{th} iteration of the loop, we have to solve the recurrance $x_{k+1} = 2x_k - c$ where $c = m + 1$ and $0 < c \leq b$.
$x_k = 2x_{k-1} - c$
$x_k = 2(x_{k-1} - \frac{c}{2})$
$x_k = x_0 2^{x_{k-1} - \frac{c}{2}}$

The solution of this recurrance is $x_k = x_0 2^{x_{k-1} - \frac{c}{2}}$, as shown above.

Now, it is clear that the n < base will become true as k becomes larger because:

$x_0 2^{x_{k-1}-\frac{c}{2}} < 2^k$
for very large k.

Therefore, x ¡ base $\rightarrow n = 0$. Therefore, we re-examine the last line of the loop to be `x = m-1`. Due to the properties of division, $n = 0 \rightarrow x = m$. Now that line is evaluated to be `x = x-1`. Now it is obvious that this will decrement x till x is 0, and the loop will terminate.