ICCS208: Assignment 4

Theeradon Sarawek
theeradon.sar@student.mahidol.edu
26 October 2024

1: Task 1: Missing Tile

Any 2^n-by-2^n grid with one painted cell can be tiled using L-shaped triominoes such that the entire grid is covered by triominoes but no triominoes overlap with each other nor the painted cell.

(1) Subtask 1

To prove this by induction, we assume the predicate will be that some 2^i by 2^i grid with one painted cell anywhere (since we want to be more general/open) is tileable.

BC: A 2^1 by 2^1 grid is tileable as with one painted cell, we know that it can fit one L-shaped trimino into it. There are four possible configurations, noted by the missing tile in each corner.

IS: Assume a P(k) = 2^k by 2^k grid is tileable with one painted cell. In theory, this should mean a P(k+1) grid, or 2^{k+1} by 2^{k+1} grid is tileable. To prove this, we can write the following:

$$\begin{aligned} 2^{k+1} \cdot 2^{k+1} &= 2(2^k) \cdot 2(2^k) && (1) \\ &= (2 \cdot 2) \cdot (2^k \cdot 2^k) && (2) \\ &= 4(2^k \cdot 2^k) && (3) \end{aligned}$$

This implies that a grid of 2^{k+1} by 2^{k+1} is composed of four of our inductive hypothesis grids.

We assume that three out of four of these grids will have a corner piece painted, like in our base case. This means the painted corner pieces can be filled with an L-shape. The fourth grid will have the painted tile, which can be anywhere in the fourth grid.

2: Task 4: Tail Sum Of Squares

Consider a function *sumSqr* and its helper function *sumHelper*:

```
int sumHelper(int n , int a) {
    if (n==0) return a;
    else return sumHelper(n-1, a + n*n);
}
int sumSqr(int n) { return sumHelper(n, 0); }
```

We want to prove that for n $\geq$ 1, sumSqr(n),$\rightarrow$ $1^2+2^2+3^2+...+n^2$. First, let's use mathematical induction to prove that sumHelper does what is intended.

Consider our predicate: sumSqr(n) $\equiv$ $\forall$a, sumHelper(n, a) $\rightarrow a + \sum_{i=1}^{n} i^2$

B.C: sumSqr(0) $\equiv$ sumHelper(0, 0) $\rightarrow$ 0

I.S: Assume for $n >= 1$ that sumSqr(n - 1) $\equiv$ $\forall$a', sumHelper(n-1, a') $\rightarrow a' + \sum_{i=1}^{n-1} i^2$

We know that $a' = a + n^2$ since it derives from sumHelper(n, a) $\rightarrow$ (sumHelper(n-1, a + n^2);

This means that $a' + \sum_{i=1}^{n-1} i^2 = a + n^2 + \sum_{i=1}^{n-1} i^2$

Which, if we consider n^2 into the sum would lead to $a + \sum_{i=1}^{n} i^2$

Therefore, we know through mathematical induction that *sumHelper* does what it is intended.

3: Task 5: Mysterious Function

Consider the following Python code:

```
def foo(n):
    assert n>=1
    if n == 1:
        return (1, 2)
    else:
        p, q = foo(n-1)
        return (q + p*n*(n+1), q*n*(n+1))
```

We want to prove for $n \geq 1$ that foo(n) $\rightarrow$ (p, q) such that $\frac{p}{q} = 1 - \frac{1}{n+1}$

B.C: foo(1) $\rightarrow$ (1,2) such that $\frac{1}{2} = 1 - \frac{1}{1+1} = 1 - \frac{1}{2} = \frac{1}{2}$

I.S: Assume that for some $n > 1$ that foo(n-1) $\rightarrow$ (p',q') such that $\frac{p'}{q'} = 1 - \frac{1}{n-1+1}$, or $1 - \frac{1}{n}$

From the function, we can see that (p, q) = $(q' + p' * n(n+1), q' * n(n+1))$

Now let's begin proving.

$$\frac{q' + p'(n)(n+1)}{q'(n)(n+1)} = \frac{q'}{(q')(n)(n+1)} + \frac{p'(n)(n+1)}{q'(n)(n+1)} \tag{4}$$

$$\frac{1}{(n)(n+1)} + \frac{p'}{q'} \tag{5}$$

$$\frac{1}{(n)} \cdot \frac{1}{(n+1)} + \frac{p'}{q'} \tag{6}$$

$$I.H \rightarrow \frac{1}{n} \cdot \frac{1}{n+1} + 1 - \frac{1}{n} \tag{7}$$

$$= 1 + \frac{1}{n} \cdot \frac{1}{n+1} - \frac{1}{n} \tag{8}$$

$$= 1 + \frac{1}{n} \cdot (\frac{1}{n+1} - 1) \tag{9}$$

$$= 1 + \frac{1}{n} \cdot (\frac{1-(n+1)}{n+1}) \tag{10}$$

$$= 1 + \frac{1}{n} \cdot \frac{-n}{n+1} \tag{11}$$

$$= 1 - \frac{1}{n+1} \tag{12}$$

Therefore, we can conclude that per mathematical induction, $n \geq 1$ that foo(n) $\rightarrow$ (p, q) such that $\frac{p}{q} = 1 - \frac{1}{n+1}$

4: Task 6: Midway Tower Of Hanoi

---

(1) Subtask I

We want to prove that $solve\_hanoi(n, ..., ...)$ prints exactly $2^n - 1$ lines of instructions.

B.C: $n = 0$ would print $2^0 - 1 = 0$ lines. Likewise, $n = 1$ would print $2^1 - 1 = 1$ lines. solve_hanoi(n-1,...,...) would do nothing since $n - 1 = 0$

I.S: Assume $solve\_hanoi(n, ..., ...)$ returns $2^n - 1$ lines for some $n \geq 1$. In the function, we see $solve\_hanoi(n - 1)$ called twice. Per IH we assume that one call will print out $2^{n-1} - 1$ instructions. Putting all the calls into one equation would lead to:

$$
\begin{aligned}
2^{n-1} - 1 + 1 + 2^{n-1} - 1 \quad & (13)\\
= 2^{n-1} + 2^{n-1} - 1 \quad & (14)\\
= \frac{2^n}{2} + \frac{2^n}{2} - 1 \quad & (15)\\
= \frac{1}{2} \cdot 2^n + \frac{1}{2} \cdot 2^n - 1 \quad & (16)\\
= 2^n - 1 \quad & (17)
\end{aligned}
$$

Therefore, per mathematical induction $solve\_hanoi(n, ..., ...)$ prints exactly $2^n - 1$ lines of instructions.