Applied Logic Weekly Assignment Report

Bohdan Tymofieienko B.S.
b.tymofieienko@student.fontys.nl
4132645

January 10, 2022

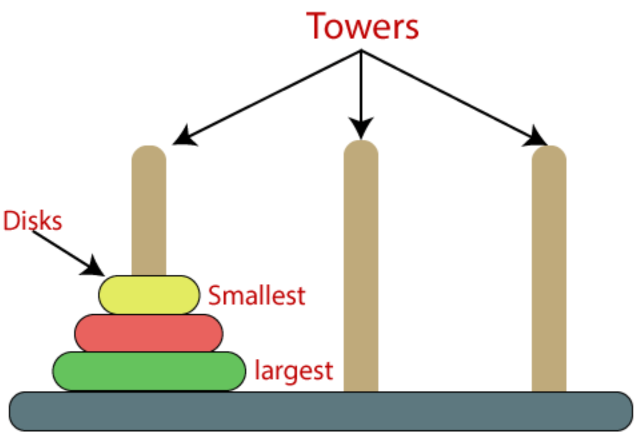
Contents

1 Extra assignment. Towers of Hanoi.

Towers of Hanoi is a famous puzzle. The target is to move all the disks from the left tower to the right tower following some rules.

Rules are next,

1. **Only one disk can be moved at a time.**
2. **No larger disks can be placed on top of smaller one.**
3. **Only top disk can be taken from the tower.**



For $Z3$ to solve this puzzle we define a set of constraint that needs to be satisfied.
Let B be a 3-dimensional array where dimension 1 is a step *(or position at time*, dimension 2 is a tower's number and dimension 3 is a level of the tower. For example in the first step the largest disk is on the third level and the smallest disk is on the first level.

There are three disks of different diameter: 1,2 and 3.

First of all we have to assure that amount of disks throughout the steps stays the same. We can do that by assuring that total diameter of all disks stays the same. In general this would've been incomplete assertion without saying that only one disk can be moved at a time. This restricts $Z3$ to say put 3 disk with diameter 1 instead of one with diameter 3, or any other permutations.

```
(forall ((step Int))
    (=> (<= 1 step (- MaxStep 1))
        (= (Total step) (Total (+ step 1)))
    )
)
```

Next we have to specify that no larger disk can be placed on top of small one. In the same assertion we say that each entry on a board is either 0 *(no disk)* or one of the disks. This is done to save space by not having 2 quantifiers. We specify explicitly that for all towers higher disk has to be less than lower disk.

```
(forall ((step Int) (tower Int) (level Int))
    (=> (<= 1 step MaxStep)
        (and
            (<= 0 (B step tower level) DiskCount)
            (<= (B step tower 1) (B step tower 2) (B step tower 3))
        )
    )
)
```

In the last assertion we have to specify that only one disk can be moved at a time. We do this declaring next : **For all two consecutive steps there exists one tower that stays unaltered and two other always change. In one of the changed towers one level becomes 0 and in another changed tower one level necessarily becomes non-zero.**

In $Z3$ syntax:

```
(forall ((step Int))
    (=> (<= 1 step MaxStep)
        (exists ((t1 Int) (t2 Int) (t3 Int))
            (and
                (distinct t1 t2 t3)
                (<= 1 t1 3)
                (<= 1 t2 3)
                (<= 1 t3 3)

                (CompareTwoTowers step (+ step 1) t1)
                (DiskMoved step (+ step 1) t2 true)
                (DiskMoved step (+ step 1) t3 false)
            )
        )
    )
)
```

The last parameter of function 'DiskMoved' is a boolean value and is exactly one time true and exactly one time false. As can be observed below, it is used to assure that one of the changed towers has one layer either zero **(Disk removed** or non-zero **(Disk added)** depending on input parameter. This helps us to avoid having two very similiar quantifier with just one statement different ('DiskRemoved' and 'DiskAdded').

```
(define-fun DiskMoved ((step1 Int)(step2 Int)(tower Int)(isZero Bool)) (Bool)
    (exists ((l1 Int)(l2 Int)(l3 Int))
        (and
            (distinct l1 l2 l3)
                        (<= 1 l1 3)
                        (<= 1 l2 3)
                        (<= 1 l3 3)
            (distinct (B step1 tower l1) (B step2 tower l1))
            (= (B step1 tower l2) (B step2 tower l2))
            (= (B step1 tower l3) (B step2 tower l3))
            (not (xor (= (B step2 tower l1) 0) isZero))
        )
    )
)
```

To make an output look clearer I apply a certain formula where I multiply the first level by 100, second level by 10 and added the sum to one value. In my opinion it makes the difference in terms of observability.

```
(+ (* 1000 (B 1 1 1)) (* 100 (B 1 1 2)) (* 10 (B 1 1 3)) (B 1 1 4))
```

1.1 Conclusion

Since we limit amount of steps to maximum 8 *(In fact 7, because step 1 is initial step)*, we can interpret the output as next:

Step/Tower	Tower 1	Tower 2	Tower 3
1	123	0	0
2	23	0	1
3	3	2	1
4	3	12	0
5	0	12	3
6	1	2	3
7	1	0	23
8	0	0	123

All the disks are moved from tower 1 to tower 3 following the initial rules. The most left digit represents the top level.

2 Solution for Four disks.

It is possible to extend the solution so that $Z3$ arranges four disks Hanoi towers. Logic stays completely the same, there is no need to add extra assertions. Some clauses need to be expanded to take fourth disk in account.

Additionally, we have to restrict minimal amount of steps to 16, since first step is an initial step. We can retrieve a next table from the $Z3$ output.

Please note: It might take really long time for $Z3$ to find the solution. In author's experience more than 60 minutes. Solution and output are attached.

Step/Tower	Tower 1	Tower 2	Tower 3
1	1234	0	0
2	234	1	0
3	34	1	2
4	34	0	12
5	4	3	12
6	14	3	2
7	14	23	0
8	4	123	0
9	0	123	4
10	0	23	14
11	2	3	14
12	12	3	4
13	12	0	34
14	2	1	34
15	0	1	234
16	0	0	1234