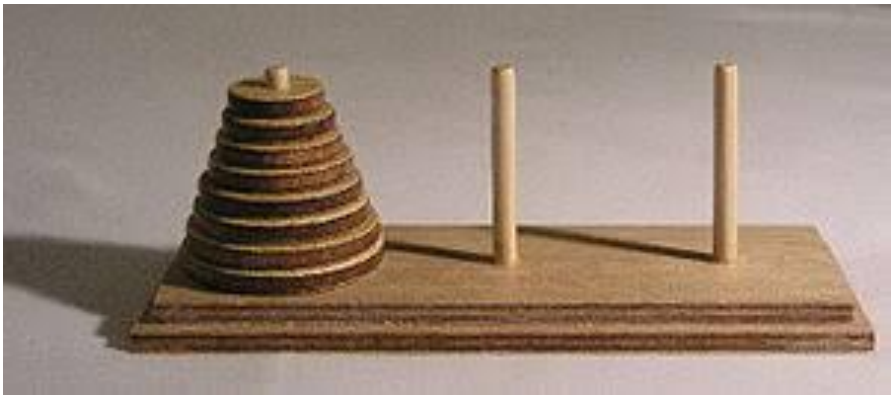
Hand-in Assignment 3



Abstract

This hand-in assignment concerns designing a linear model to describe the real system presented in the coming sections. The model has to be then implemented and solved using Z3. Results have to be presented in the form of a short written report (max 6 pages). Also, include the **code** as separate files.

Planning: Traditional *AI planning* tackles the problem of finding plans to achieve some goal, i.e., finding a sequence of activities that will transfer the initial world into one in which the goal description is true. This means that a description of the initial world, the (partial) specification of the desired world, and the list of available activities are input to the planner. A solution is a sequence of activities that leads from the initial world description to the goal world description and this is called a *plan*.

Task I: The Tower of Hanoi

The Tower of Hanoi is a mathematical puzzle invented by the French mathematician, Edouard Lucas, in 1883. It consists of three towers, and a number of disks of different sizes that can slide onto any tower. The final target of the game is to move the entire stack of disks onto another tower, without breaking the following rules:

- only one disk can be moved at each time step;
- no disk can be placed on the top of a smaller one;
- when moving a disk from a tower, it can only go on the top position on one of the other towers (which means on the top of a bigger disk if this is present already on this tower).

The original problem involved three towers and three disks and can be solved in seven time-steps. More generally it is possible to have *n* disks and *m* towers, but this will obviously change the number of steps required; for the particular case when we have only three towers, the total amount of steps depends on the number of disks, according to:

$$t_s = 2^n - 1 \quad (1)$$

Your Task: your job is to implement a model in Z3 to encode the Tower of Hanoi and check whether the puzzle is solvable or not, given a certain number of steps. Note that the code you write has to be parametrized, so that you can set different numbers of towers and disks. Then run your code for a number of disks that goes from 3 to at least 7, and 3 towers.

Now, let's assume we do not know (1), hence, we do not know beforehand how many steps we need to move the stack of disks from one tower to another; how can we turn the problem into an optimization problem so that we can have the solver automatically find the minimum number of step for us?

What you have: in the attached file a possible encoding for the problem is presented. You can use it to implement your solution in Z3 or try to find your own way. (**Hint**: there is a lot of room for improvement in the model presented in the attachment.)

Task II: A General Sorting Problem

The model you have developed in the previous task falls into the category of planning problems, where you have an initial configuration and you want to plan all the necessary moves to reach a final configuration. The task you have to accomplish now is similar, but the constraints you have to set are slightly different: imagine you have a robot crane operating on a platform where positions are uniquely defined. There is a certain number of objects (we will call them **bricks**) on the platform and you need to move them to other locations on the same platform; this time as well you want to minimize the number of moves since time (and the energy required to move the robot)…. is money!

The relevant constraints in this system are:

- only one brick can be moved at each time-step;
- two bricks cannot occupy the same position;
- each brick has to be considered unique.

Your Task: Based on the model you developed in the previous task, design a model to represent the robot sorting problem and then implement it in Z3. How scalable is your model? How much can you increase the size of the platform and the number of bricks before the state-space explodes? What would change in the model if it had classes of objects, i.e. objects of the same colour that can be considered identical?

Let's now imagine a more complex scenario, with multiple robots and platforms. Objects can be moved from a platform to another through shared in-between adjacent platforms, accessible by the robots working on such platform. Extend the model you implemented for the single platform scenario (this time you don't have to implement the model in Z3)

Hint: you might need to change your decision variables

So far the cost function has been the number of moves. What if we would like to have the overall distance covered by the robot while moving the objects around? Design the new cost function using both Cartesian and Manhattan coordinates for a square platform with arbitrary size $n \times n$.