Spectra Example: Towers of Hanoi

Shahar Maoz

Tel Aviv University

Abstract. The Towers of Hanoi is a classic mathematical puzzle. This report presents a specification and simulation for the puzzle in Spectra [3]. In addition to assumptions and guarantees, it specifically demonstrates the use of Spectra defines and arrays with quantification.

1 Introduction

The Towers of Hanoi is a classic mathematical puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules: (1) only one disk can be moved at a time; (2) each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod, and (3) no larger disk may be placed on top of a smaller disk. With 3 disks, the puzzle can be solved in 7 moves. The minimal number of moves required to solve a Towers of Hanoi puzzle is $2^n - 1$, where n is the number of disks. A comprehensive discussion about the puzzle appears in [1].

We present a Spectra [3] specification for the puzzle and a simulation of its solution. The specification demonstrates the use of Spectra defines and arrays with quantification. All related files are available from `http://smlab.cs.tau.ac.il/syntech/`.

2 The Specification

2.1 Defines and variable declarations

The environment describes and controlls the state of the "world", specifically which disk is on which tower. We model that using the array `diskPosition` declared in line 10 of Listing 1.1.

The system controls the actions that should be taken at each step in order to solve the puzzle, specifically which disk should be moved and to which tower. We model it using two variables, `moveDiskNumber` and `moveDiskToTower`, declared in lines 13-14 of Listing 1.1.

Note that the use of defines for number of disks and number of towers makes the specification more readable and easier to change.

```
spec TowersOfHanoi

define NUMBEROFDISKS := 5;
define NUMBEROFDISKSMINUSONE := NUMBEROFDISKS-1;
define NUMBEROFDISKSMINUSTWO := NUMBEROFDISKS-2;

define NUMBEROFTOWERSMINUSONE := 2;

// position of the disks
env Int(0..NUMBEROFTOWERSMINUSONE)[NUMBEROFDISKS]
    diskPosition;

// current disk to move
sys Int(0..NUMBEROFDISKSMINUSONE) moveDiskNumber;
sys Int(0..NUMBEROFTOWERSMINUSONE) moveDiskToTower;
```

Listing 1.1. Spectra specification of the Towers of Hanoi puzzle: defines and variable declarations

```
asm startLeft:
forall i in Int(0..NUMBEROFDISKSMINUSONE) .
 diskPosition[i] = 0;

asm carryingADiskChangesItsPosition:
G forall i in Int(0..NUMBEROFDISKSMINUSONE) .
 forall j in Int(0..NUMBEROFTOWERSMINUSONE) .
  (moveDiskNumber=i & moveDiskToTower=j) ->
   (next(diskPosition[i]) = j);

asm notCarryingADiskKeepsDiskPosition:
G forall i in Int(0..NUMBEROFDISKSMINUSONE) .
 (!(moveDiskNumber=i))->
  (diskPosition[i]=next(diskPosition[i]));
```

Listing 1.2. Spectra specification of the Towers of Hanoi puzzle: assumptions

2.2 Assumptions and guarantees

Listing 1.2 presents the three assumptions that describe the expected behavior of the "world". First, initially, all disks are on the leftmost tower. Second, carrying a disk will indeed change its position. Finally, if a disk is not carried, it will not change its position.

Listing 1.3 presents the guarantees that the system must satisfy. First, the system must never put a larger disk on top of a smaller one. Second, when two disks are on the same tower, the larger disk cannot be moved. Third, all disks must eventually reach the rightmost tower. Finally, once all disks are at the rightmost tower, no disk should move anymore.

```
gar cantPutALargerDiskOnTopOfSmallerDisk:
G forall i in Int(1..NUMBEROFDISKSMINUSONE) .
 forall j in Int(0..NUMBEROFDISKSMINUSTWO) .
  forall k in Int(0..NUMBEROFTOWERSMINUSONE) .
   (j<i & moveDiskNumber=i & diskPosition[j]=k) ->
    moveDiskToTower != k;

gar mustMoveSmallestDiskFirst:
G forall i in Int(0..NUMBEROFDISKSMINUSONE) .
 forall j in Int(1..NUMBEROFDISKSMINUSONE) .
  (j>i & diskPosition[i]=diskPosition[j]) ->
   moveDiskNumber!=j;

gar allDisksMustReachRighmostTower:
GF forall i in Int(0..NUMBEROFDISKSMINUSONE) .
 diskPosition[i]=NUMBEROFTOWERSMINUSONE;

gar onceSolvedNoDiskMoves:
G (forall i in Int(0..NUMBEROFDISKSMINUSONE) .
 diskPosition[i]=NUMBEROFTOWERSMINUSONE) ->
  (forall i in Int(0..NUMBEROFDISKSMINUSONE) .
   diskPosition[i]=next(diskPosition[i]));
```

Listing 1.3. Spectra specification of the Towers of Hanoi puzzle: guarantees

Note that the last guarantee implicitly turns the specification's semantics into one that considers finite computation paths. In general, Spectra is defined over infinite computations. We have added the last guarantee in order to align the specification with the basic Towers of Hanoi puzzle, which considers finite solutions.

3 Some Example Analyses

In addition to synthesizing a controller, one may apply different analyses to study the specification.

As a first example, one may be interested in checking whether all assumptions are indeed required for realizability. To check this use Cores / Find an Assumptions Core on the Spectra Add-ons menu. The computed core includes two assumptions, the second and third assumptions, but not the first. This shows that the first assumption, about the initial state, is not necessary for realizability. Moreover, it shows that the second and third assumptions are both sufficient and necessary for realizability. Indeed, one can comment out one of the assumptions in the core and check that the specification becomes unrealizable.

As another example, one can check whether the specification is well-separated [2], i.e., intuitively, whether the system may be able to force the environment into violating the assumptions. Specifications that are not well-separated are considered problematic. To check this use Well-Separation / Diag-

nose Well-Separation of Environment. The output on the console will show that the specification is well-separated.

Acknowledgements

The authors thank Yarden Meshulam for the implementation of the simulation. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638049, SYNTECH).

References

1. Hinz, A.M., Klavzar, S., Milutinovic, U., Petr, C.: The Tower of Hanoi - Myths and Maths. Birkhäuser (2013). https://doi.org/10.1007/978-3-0348-0237-6, https://doi.org/10.1007/978-3-0348-0237-6
2. Maoz, S., Ringert, J.O.: On well-separation of GR(1) specifications. In: Zimmermann, T., Cleland-Huang, J., Su, Z. (eds.) Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016. pp. 362–372. ACM (2016), http://doi.acm.org/10.1145/2950290.2950300
3. Maoz, S., Ringert, J.O.: Spectra: A specification language for reactive systems. CoRR **abs/1904.06668** (2019), http://arxiv.org/abs/1904.06668