CP Attachment Assignment 3

Sabino Roselli

July 2019

SAT Encoding of the ToH

Previous Encodings and Background Theory

As *Martins & Lynce* [1] suggest, this puzzle has been already solved in the past by using SAT encoding and there are different versions which might be more or less efficient, according to the number of variable and assertions declared while setting the model. A first solution to describe the real system would be to set the following variables:

$$\begin{cases} on(d, tw, t) \\ clear(tw, t) \\ move(d, tw, tw', t) \end{cases} \tag{1}$$

The variable *on* states where the disk d is at a certain time-step t; the variable *clear* tells which tower *tw* is available for a disk to move onto it at a time-step t; the variable *move* tells which disk *d*, from which tower *tw*, to which tower *tw'* at which time-step *t*. These Boolean variables will tell us what is happening in the system step after step by assuming a value True or False. For instance, if the variable *on(1,2,3)* is true it means that the disk number 1 is on tower number 2 at step number 3. We can now appreciate the difference between conventional programming and constraint programming. In the first case we would set some *for*, *while* or *if* cycle in order to update these three variables until we reach the final state; with CP we are actually defining all possible scenarios and then we will reduce them by setting constraints according to the real system limitation.

One negative aspect when programming this way is that the number of variables and (especially) assertions created grows exponentially with the problem's size. For instance for a ToH of three disks and three towers we would have 189 variables *move* because we know we will need seven steps to reach the final state; but if we add only two disks we will need 31 steps which means 1395 variables *move*. If we then take into account that assertions are a combination of these variables we get an idea of how big the model becomes.

[1] Martins, R., & Lynce, I. (2008). Effective CNF Encodings for the Towers of Hanoi. International Conference on Logic for Programming Artificial Intelligence and Reasoning.

This is why working with constraint programming also means trying to simplify the model as much as possible and contain the exponential growth. This is exactly what has been done in the ToH encoding, by substituting the variable *move* with the conjunction:

$$obj(d,t) \wedge from(tw,t) \wedge to(tw,t) \tag{2}$$

the variable *obj* will indicate which disk we are moving and the other two the starting and ending tower at that precise step. This way we don't need the variables *move* and *clear* anymore and so for a three disks problem we will only declare 132 variables instead of 273 and for a five-disks problem we would have only 806 variables instead of 1953.

Although this difference might seem relevant already, it is nothing if we think that all these variables have to be combined to form the constraint assertions and that by only adding one disk we could make the model ten times bigger.

Setting Constraints

We will now set the model for a generic ToH, where we have *ds* disks, *tws* towers and *ts* time-steps. Let us see though, one by one, all the constraints required to encode the ToH:

- **Precondition I**: this set of constraints states that if a disk is on a tower and a smaller disk is on the same tower the disk can not be moved (3).
- **Precondition II**: if we are to move a disk onto a certain tower and there is already a smaller one there, then is not possible to move our disk onto that tower (4).
- **Uniqueness of From variable** : if we are to move a disk, and it is on a certain tower, than the variable *from* will be true only for that tower (5).
- **Uniqueness of To variable**: we can only have one variable *to* being true for each time-step (6).
- **Uniqueness of Obj variable**: we are only allowed to move one disk at each time-step (7).
- **Non-moving Disks**: if a disk is not moving at a certain time-step it will be on the same tower at the following step and nowhere else (8).
- **Distinct From/To**: it is not allowed to move a disk from a tower to the same one (9).
- **Update**: if the we are to move a disk from a tower to another one, then the disk will be on the latter at the next time-step and only on that one (10).
- **Initial/Final State**: the disks are on the first tower at the first time-step and on the last one at the step *ts* (11).

$$\begin{cases} on(d,tw,t) \wedge (on(1,tw,t) \vee .....on(d-1,tw,t)) \Rightarrow \neg obj(d,t) \\ d \in [1,ds] \qquad tw \in [1,tws] \qquad t \in [1,ts] \end{cases} \tag{3}$$

$$\begin{cases} on(d,tw,t) \wedge (on(1,tw',t) \vee .....on(d-1,tw',t)) \Rightarrow \neg(obj(d,t) \wedge to(tw',t)) \\ d \in [1,ds] \qquad tw \in [1,tws] \qquad tw' \in [1,tws] \qquad t \in [1,ts] \end{cases} \tag{4}$$

$$\begin{cases} on(d,tw,t) \wedge obj(d,t) \Rightarrow from(tw,t) \wedge \neg from(1,t) \wedge ...\neg from(tw-1,t) \\ \qquad \wedge \neg from(tw+1,t) \wedge ....\neg from(tws,t) \\ d \in [1,ds] \qquad tw \in [1,tws] \qquad t \in [1,ts] \end{cases} \tag{5}$$

$$\begin{cases} to(tw,t) \Leftrightarrow \neg to(1,t) \wedge ...\neg to(tw-1,t) \wedge \neg to(tw+1,t) \wedge ....\neg to(tws,t) \\ tw \in [1,tws] \qquad t \in [1,ts] \end{cases} \tag{6}$$

$$\begin{cases} obj(d,t) \Leftrightarrow \neg obj(1,t) \wedge ...\neg obj(d-1,t) \wedge \neg obj(d+1,t) \wedge ....\neg obj(ds,t) \\ d \in [1,ds] \qquad t \in [1,ts] \end{cases} \tag{7}$$

$$\begin{cases} \neg obj(d,t) \wedge on(d,tw,t) \Rightarrow on(d,tw,t+1) \wedge \neg on(d,1,t+1) \wedge \\ ...\neg on(d,tw-1,t+1) \wedge \neg on(d,tw+1,t+1) \wedge ....\neg on(d,tws,t+1) \\ d \in [1,ds] \qquad tw \in [1,tws] \qquad t \in [1,ts] \end{cases} \tag{8}$$

$$\begin{cases} from(tw,t) \Rightarrow \neg to(tw,t) \\ tw \in [1,tws] \qquad t \in [1,ts] \end{cases} \tag{9}$$

$$\begin{cases} obj(d,t) \wedge from(tw,t) \wedge to(tw',t) \Rightarrow on(d,tw',t+1) \wedge \neg on(d,1,t+1) \wedge \\ ...\neg on(d,tw-1,t+1) \wedge \neg on(d,tw+1,t+1) \wedge ....\neg on(d,tws,t+1) \\ d \in [1,ds] \qquad tw \in [1,tws] \qquad t \in [1,ts] \end{cases} \tag{10}$$

$$\begin{cases} on(d,1,1) \wedge on(d,tws,ts) \\ d \in [1,ds] \end{cases} \tag{11}$$

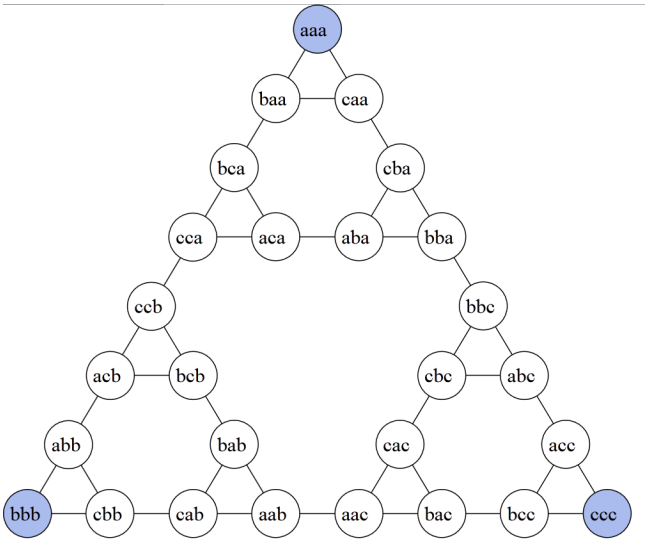

Figure 1: Space of Solutions for 3 disks and 3 towers.

The picture shows all the possible states that can occur when moving the disk among the towers. The aim of our model is to find the shortest path through this graph to connect the top of the pyramid to one of its base corners.