Student: Marco Antonio Manzo Ruiz

ID: 173127

Professor: Dr. Ingrid Kirschning Albers

Course: Artificial Intelligence

Project 1: Hanoi Towers in Prolog

Introduction

The Tower of Hanoi is a mathematical puzzle that consists of three different rods with “n” number of disks, each with different diameters. The first rod always has all “n” disk and the purpose of the puzzle is to shift the entire stack of disks from one rod to the last rod. However, there are rules that should be followed to correctly solve the puzzle. The first rule is that only one disk can be moved at a time. The second rule is that only the uppermost disk from a stack of disks can be moved into the top of another stack or into an empty stack, this means that the player cannot move the middle disks or the bottom disks of the stack. The final rule is that players cannot put a disk on top of a smaller disk. There is also a formula which tells us that the minimal number of moves required to solve the puzzle of “n” disks is $((2^n) – 1)$. This is considered the correct response in all cases and the one that this project aims to obtain while using prolog.

Representation of the elements needed to work in the Prolog code

The code that will be used to work in the Tower of Hanoi puzzle will be using different elements to make it work in Prolog. The first is the number of disks “N”, this is passed as an argument to the {hanoi(N)} predicate and it is used to control the recursive calls in the {move} predicate. The rods will be represented as “A”, “B”, and “C”, these will be used as arguments in the {move} predicate to indicate the beginning and destination of the disks, that is why the predicate is called {move}. In this code, there is also an added counter, which is used to keep track of the movements made, this was added to prove the previously mentioned formula of the minimal required movements $((2^n) – 1)$.

Explanation of the Rules of the problem

There will be three rules in the code, the first rule will deal with the situation in which there is only one disk to move, if this happens, the disk will be moved from the “From_rod” position into the “To_rod” position. It will also increment the counter by 1 and return its value, it will finally print the movement made and print the value of the counter.

The second rule will use the {move} predicate in a recursive way. It deals with the situation in which there is more than one disk. It calls the predicate with the argument “N1”, which is the number of disks minus one. Then, it moves the smaller disk to the auxiliary rod or the “Aux_rod”, which should be the one in the middle. It then increments the counter by one. It does this recursively until the puzzle is solved.

The third rule is the predicate hanoi(N), which starts the program when called. It takes the argument “N” and calls the {move} predicate with the number of disks and the three different rods, also it calls the counter.

Code

Project_1_173127.pl

File Edit Browse Compile Prolog Pce Help

Project_1_173127.pl

```
move(1, From_rod, To_rod, _, Counter, NewCounter) :-
NewCounter is Counter + 1,
write("Move disk 1 from rod "), write(From_rod),
write(" to rod "), write(To_rod),
write(", move #"), write(NewCounter), nl.

move(N, From_rod, To_rod, Aux_rod, Counter, NewCounter) :-
N > 1,
N1 is N - 1,
move(N1, From_rod, Aux_rod, To_rod, Counter, TempCounter),
NewCounter1 is TempCounter + 1,
write("Move disk "), write(N),
write(" from rod "), write(From_rod),
write(" to rod "), write(To_rod),
write(", move #"), write(NewCounter1), nl,
move(N1, Aux_rod, To_rod, From_rod, NewCounter1, NewCounter).

hanoi(N) :- move(N, 'A', 'C', 'B', 0, TotalMoves), write("Total moves: "), write(TotalMoves), nl.
```

Output

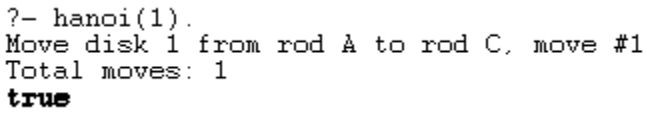

Figure 1: Case in which there is only 1 disk.

This first figure represents the output of when there is only 1 disk. To make the program work, the user must input “hanoi(1).” Without the quotation. Then, the program will return what was the movement and how many movements it took, in this case, 1. Finally, the program will print “true”. The formula is correct in this case because $((2^1) – 1) = 1$.

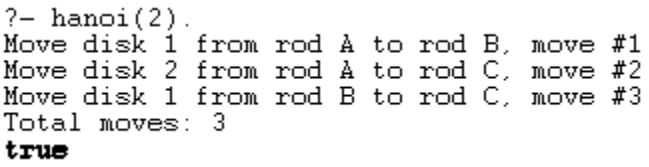

Figure 2: Case in which there are 2 disks.

This second figure represents the output of when there are 2 disks. To make the program work, the user must input “hanoi(2).” Without the quotation. Then, the program will return which movements were made and how many movements it took, in this case, 3. Finally, the program will print “true”. The formula is correct in this case because $((2^2) – 1) = 3$.

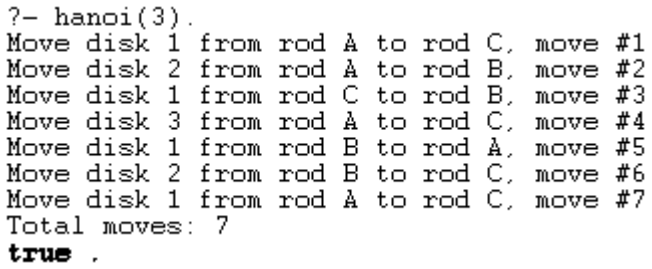

Figure 3: Case in which there are 3 disks.

This third figure represents the output of when there are 3 disks. To make the program work, the user must input “hanoi(3).” Without the quotation. Then, the program will return which movements were made and how many movements it took, in this case, 7. Finally, the program will print “true”. The formula is correct in this case because $((2^3) – 1) = 7$.

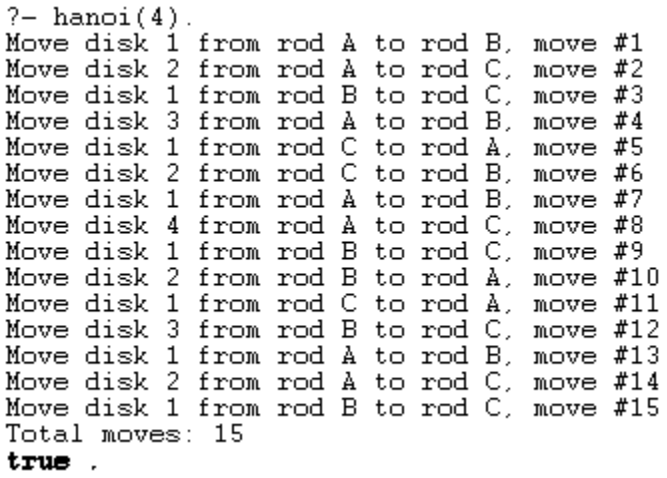

Figure 4: Case in which there are 4 disks.

This fourth figure represents the output of when there are 4 disks. To make the program work, the user must input “hanoi(4).” Without the quotation. Then, the program will return which movements were made and how many movements it took, in this case, 15. Finally, the program will print “true”. The formula is correct in this case because $((2^4) – 1) = 15$.

```
?- hanoi(5).
Move disk 1 from rod A to rod C, move #1
Move disk 2 from rod A to rod B, move #2
Move disk 1 from rod C to rod B, move #3
Move disk 3 from rod A to rod C, move #4
Move disk 1 from rod B to rod A, move #5
Move disk 2 from rod B to rod C, move #6
Move disk 1 from rod A to rod C, move #7
Move disk 4 from rod A to rod B, move #8
Move disk 1 from rod C to rod B, move #9
Move disk 2 from rod C to rod A, move #10
Move disk 1 from rod B to rod A, move #11
Move disk 3 from rod C to rod B, move #12
Move disk 1 from rod A to rod C, move #13
Move disk 2 from rod A to rod B, move #14
Move disk 1 from rod C to rod B, move #15
Move disk 5 from rod A to rod C, move #16
Move disk 1 from rod B to rod A, move #17
Move disk 2 from rod B to rod C, move #18
Move disk 1 from rod A to rod C, move #19
Move disk 3 from rod B to rod A, move #20
Move disk 1 from rod C to rod B, move #21
Move disk 2 from rod C to rod A, move #22
Move disk 1 from rod B to rod A, move #23
Move disk 4 from rod B to rod C, move #24
Move disk 1 from rod A to rod C, move #25
Move disk 2 from rod A to rod B, move #26
Move disk 1 from rod C to rod B, move #27
Move disk 3 from rod A to rod C, move #28
Move disk 1 from rod B to rod A, move #29
Move disk 2 from rod B to rod C, move #30
Move disk 1 from rod A to rod C, move #31
Total moves: 31
true .
```

Figure 5: Case in which there are 4 disks.

This fifth figure represents the output of when there are 5 disks. To make the program work, the user must input “hanoi(5).” Without the quotation. Then, the program will return which movements were made and how many movements it took, in this case, 31. Finally, the program will print “true”. The formula is correct in this case because $((2^5) – 1) = 31$.

Conclusion

In this project, the goal was to represent the Tower of Hanoi puzzle using the language Prolog. This represented several challenges. First, I needed to really understand the capabilities of Prolog to really understand what was happening when rules or predicates were made. As it is different from other programming languages that I've used, there were things such as the predicates that were hard to grasp at the beginning of the process. Secondly, I needed to play the puzzle to get used to it and see what I wanted the code to do, I also needed to understand the puzzle and seek the minimal number of movements that I had to do in each playthrough. Lastly, I needed to research different ways to implement the code to find the best way to make it work. This was mostly trial and error for there were many syntax issues at the beginning of the process, but I got better and even added a counter which represents the total number of moves. Finally, there is a functioning Prolog code which prints the movements made to solve the puzzle in the minimal number of movements and there is also a counter which knows how many movements were made, this helped me know if the formula $((2^n) - 1)$ was being fulfilled and at the end, it was.

GitHub link to the repository were the code is:

https://github.com/MarcoManzo13/ArtificialIntelligence.git

Reference

1- Tower of Hanoi—A Recursive approach. (2021). Retrieved 12 February 2023, from https://towardsdatascience.com/tower-of-hanoi-a-recursive-approach-12592d1a7b20#:~:text=The%20objective%20of%20the%20game,the%20top%20of%20smaller%20disks.