



LIS3082-1 Inteligencia Artificial

Primavera 2023

Dr. Ingrid Kirschning

Project 1: Hanoi Towers in Prolog

12 de febrero de 2023

Alumna:

Frida Pérez Perfecto, ID: 166520

## Introduction

### Hanoi Towers

According to Educative (n.d.) the Towers of Hanoi is a mathematical game or puzzle consisting of three rods and a number of disks of various 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 this problem is to move the stack of disks from the initial rod to another rod, following these rules:

- Only one disk may be moved at a time.
- 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.
- No disk may be placed on top of a disk that is smaller than it.

The puzzle can be played with any number of disks, although many toy versions have around 7 to 9 of them. The minimal number of moves required to solve the Towers of Hanoi puzzle is  $2^n - 1$ , where  $n$  is the number of disks. We can solve this problem with an iterative or recursion solution.

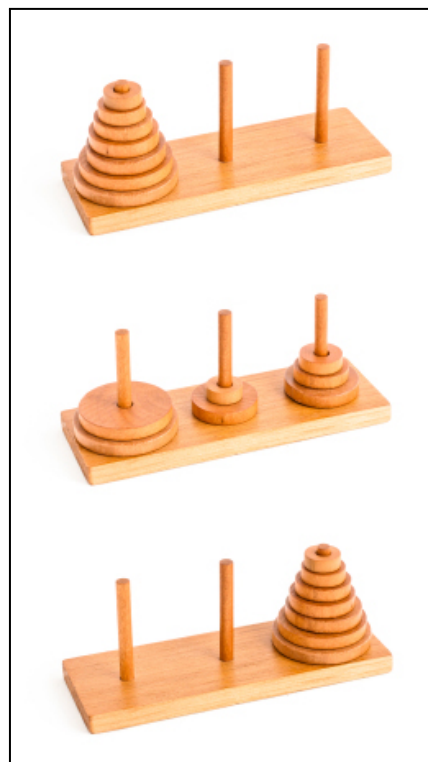


Figure 1. Hanoi Towers Game.

According to Scientific American (2017) the Towers of Hanoi was invented by a French mathematician, Édouard Lucas, in the 19th century. It is associated with a legend of a Hindu temple where the puzzle was supposedly used to increase the mental discipline of young priests. In the legend the young priests were given 64

gold disks stacked neatly on one of three posts. Each disk rested on a slightly larger disk. The priests' goal was to recreate the stack on a different post by moving disks, one at a time, to another post with the rule that a larger disk could never be placed on top of a smaller disk.

## Prolog

Prolog, from the French PROgrammation en LOGique, is a logic and declarative programming language. According to Calapini (2022) it is a programming language that is well-suited for developing logic-based artificial intelligence applications. This means that it allows the programmer to specify the rules and facts about a problem domain, and then the Prolog interpreter will use these to automatically infer solutions to problems. It was created by Alain Colmerauer and Robert Kowalski around 1972.

## Predicates

A relation between objects is called a predicate. In natural language a relation is symbolized by a sentence:

- Tom is a cat.
- Peter plays football.
- Susan's hair is black.
- Bill likes Cindy.

## Facts

In Prolog, a fact is a predicate expression that is written as:

`relation(object1,object2...).`

Figure 2. Facts syntax.

The programmer must define the interpretation of the objects and the relationships between them and they are unconditionally true. Some examples of facts can be:

- `man(john).`
- `cat(tom).`
- `mother(ana, maria).`
- `capital(paris, france).`

In these examples the first line states that john is a man, the second line states that tom is a cat, the third line states that ana is the mother of maria, and the fourth line states that paris is the capital of france.

As we can see, facts should always begin with a lowercase letter and end with a full stop. The relation is followed by the object or objects enclosed in parentheses, and as with a sentence, the fact ends with a period (.). The facts themselves can

consist of any letter or number combination, as well as the underscore \_ character. For example:

- loves\_to\_eat(sofia, pasta).
- teaches(andrea, course101).
- phone\_number(Mike, 11119997543).

However, names containing the characters -, +, \*, /, or other mathematical operators should be avoided.

## Variables

In Prolog, variables may be used to denote unknown but specific objects, some variables may denote an object the properties of which are specified without specifying the object itself.

According to variable (n.d.) a variable in Prolog is a string of letters, digits, and underscores (\_) beginning either with a capital letter or with an underscore. For example:

- X
- Sister
- \_
- \_thing
- -y21
- First\_name

We use them when we do not mind what value the variable has.

## Rules

A rule can be viewed as an extension of a fact with added conditions that also have to be satisfied for it to be true. It consists of two parts, a head and a body.

**Head :- Body.**

Figure 3. Rules syntax.

According to TutorialsPoint (n.d.) the symbol :- is known as *neck symbol*, and is pronounced as “if”. Rules describe a relationship among facts, and enable us to infer facts from other facts. For example:

- happy(lily) :- dances(lily).
- hungry(tom) :- search\_for\_food(tom).
- friends(jack, emily) :- lovesVolleyball(jack), lovesVolleyball(emily).
- goToPlay(ryan) :- isClosed(school), free(ryan).

In these examples the first line states that lily is happy if she dances, the second line states that tom is hungry if he is searching for food, the third line states that jack and emily are friends if both of them love to play volleyball, and the fourth line states that ryan will go to play if school is closed, and if he is free.

These are some rules that are conditionally true, so if the head is true, then the body is also true. If the body is false, then the head is false.

We can also define rules with variables. For example:

- likes(daniel, X) :- food(X).
- friends(X, Y) :- likes(X, Y), likes(Y, X).

In these examples the first line states that daniel likes X if X is food, and the second line states that X and Y are friends if X likes Y and Y likes X.

## Queries

According to Facts, Rules and Queries (n.d.) in Prolog, the query is the action of asking the program about the facts and rules represented in its database. The database is assumed to represent what is true about a particular problem domain.

In making a query you are asking Prolog whether it can prove that your query is true. If so, it answers "true" and displays any variable bindings that it made in coming up with the answer. If it fails to prove the query true, it answers "false". For example:

1 - tom(X).  
X = cat.

2- tom(dog).  
false.

3- mother(X, Y).  
X = ana.  
Y = maria.

4- capital(paris, france).  
true.

## Explanation

As we said previously, the objective of this problem is to move the stack of disks from the initial rod to another rod, following these rules:

- Only one disk may be moved at a time.
- 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.

- No disk may be placed on top of a disk that is smaller than it.

To solve this puzzle in Prolog we must code a procedure that accepts a  $N$  number of disks that we will have to move from the left rod to the right rod keeping the middle rod as intermediate. The challenge of this is to solve it with the fewest moves.

## Code

```
move(1, X, Y, _) :-
    write('Move top disk from '),
    write(X),
    write(' to '),
    write(Y),
    nl.
move(N, X, Y, Z) :-
    N>1,
    M is N-1,
    move(M, X, Z, Y),
    move(1, X, Y, _),
    move(M, Z, Y, X).
```

## Description

```
move(1, X, Y, _) :-
    write('Move top disk from '),
    write(X),
    write(' to '),
    write(Y),
    nl.
```

This part of the code means that we are just going to move one disk at a time, and from where we take it (X) is going to be the source rod, and where we place it (Y) is going to be the target rod, and any other rod is going to be the intermediate rod (\_). So it prints the moves we are making, from where we have to take the disk and where we have to place it, every move of a disk in a new line.

```
move(N, X, Y, Z) :-
    N>1,
    M is N-1,
    move(M, X, Z, Y),
    move(1, X, Y, _),
    move(M, Z, Y, X).
```

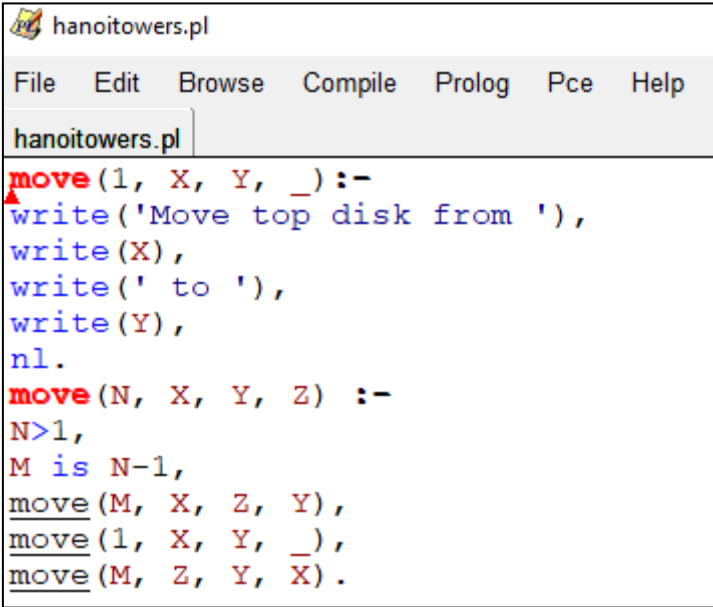
This part of the code means that if we have N number of disks to move, it lets us move M number of disks, and then the last one will be moved accordingly. So M number of disks will move from X to Z, keeping Y as intermediate, and then the largest disk will move from X to Y, keeping the other rod (Z) as intermediate, and those disks that moved last to Z. Then the M number of disks will move from Z to Y, which was the end goal, keeping X as intermediate.

## GitHub repository

The link to access this report and the program in Prolog is:  
<https://github.com/fridapp09/Prolog-HanoiTowers>

## Results

After writing the code in the Prolog editor, we compiled and executed it so we could give to the program some queries about how many disks there will be so the puzzle can be solved.



```
hanoitowers.pl
File Edit Browse Compile Prolog Pce Help
hanoitowers.pl
move(1, X, Y, _) :-
write('Move top disk from '),
write(X),
write(' to '),
write(Y),
nl.
move(N, X, Y, Z) :-
N>1,
M is N-1,
move(M, X, Z, Y),
move(1, X, Y, _),
move(M, Z, Y, X).
```

Figure 4. Code in the Prolog editor.

```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.3)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/Frida/Documents/Prolog/hanoitowers.pl compiled 0.00 sec, 2 clauses
?- move(3, left, right, center).
Move top disk from left to right
Move top disk from left to center
Move top disk from right to center
Move top disk from left to right
Move top disk from center to left
Move top disk from center to right
Move top disk from left to right
true.
```

Figure 5. Solution with 3 disks.

```
?- move(4, left, right, center).
Move top disk from left to center
Move top disk from left to right
Move top disk from center to right
Move top disk from left to center
Move top disk from right to left
Move top disk from right to center
Move top disk from left to center
Move top disk from left to right
Move top disk from center to right
Move top disk from center to left
Move top disk from right to left
Move top disk from center to right
Move top disk from left to center
Move top disk from left to right
Move top disk from center to right
true.
```

Figure 6. Solution with 4 disks.



```
?- move(5, left, right, center).  
Move top disk from left to right  
Move top disk from left to center  
Move top disk from right to center  
Move top disk from left to right  
Move top disk from center to left  
Move top disk from center to right  
Move top disk from left to right  
Move top disk from left to center  
Move top disk from right to center  
Move top disk from right to left  
Move top disk from center to left  
Move top disk from right to center  
Move top disk from left to right  
Move top disk from left to center  
Move top disk from right to center  
Move top disk from left to right  
Move top disk from center to left  
Move top disk from center to right  
Move top disk from left to right  
Move top disk from center to left  
Move top disk from right to center  
Move top disk from right to left  
Move top disk from center to left  
Move top disk from center to right  
Move top disk from left to right  
Move top disk from left to center  
Move top disk from right to center  
Move top disk from left to right  
Move top disk from center to left  
Move top disk from center to right  
Move top disk from left to right  
true .
```

Figure 7. Solution with 5 disks.

To check the results I used an online simulator because I do not have the physical game of the Towers of Hanoi at my house. The simulator lets us choose between 3, 4 or 5 disks, and it shows us the number of moves we are making while we are playing. I chose to play it with 4 disks.

The challenging thing here is that it has to be solved with the fewest moves, so I tried to do it this way but I didn't succeed. I solved it but not with the fewest moves, therefore, I decided to try it out with the results of the execution of the code, so I followed the “instructions” for solving the puzzle with four disks (Figure 6). And as we can see in Figure 8 the results were correct, we solved the puzzle with the fewest number of moves.

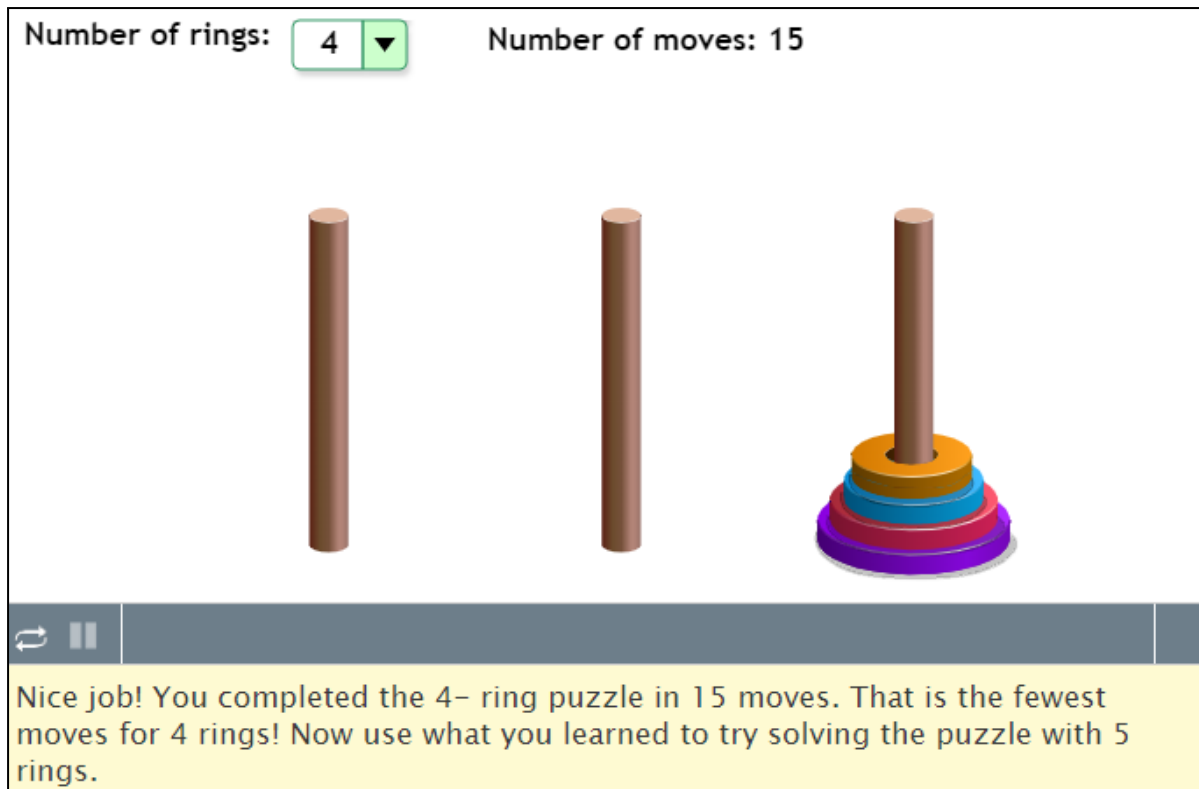


Figure 8. Online simulator of the Hanoi Towers with 4 disks solved in the webpage Learning and the Adolescent Mind (see in References).

## Conclusion

The Towers of Hanoi is a fun and classic problem in the world of programming, because it allows us to plan and solve problems by making us think logically and algorithmically to compute a solution, which is important in becoming a programmer. And we can use all this information written in a special syntax so that Prolog can understand it to analyze the relationships of the facts and rules of the puzzle, so we can solve it in the most efficient way.

## References

- Calapini, C. (2022). *Introduction to Prolog: A Programming Language for Artificial Intelligence*. Retrieved February 11, 2023, from: <https://blog.devgenius.io/introduction-to-prolog-a-programming-language-for-artificial-intelligence-320b75455381>
- Educative. (n.d.). *What is the Tower of Hanoi problem?* Retrieved February 10, 2023, from: <https://www.educative.io/answers/what-is-the-tower-of-hanoi-problem>
- Facts, Rules and Queries. (n.d.). Retrieved February 12, 2023, from: <http://www.cs.trincoll.edu/~ram/cpsc352/notes/prolog/factsrules.html>
- Learning and the Adolescent Mind. (n.d.). *Classroom Tools - Towers of Hanoi*. Retrieved February 9, 2023, from: [http://learningandtheadolescentmind.org/resources\\_02\\_towers.html](http://learningandtheadolescentmind.org/resources_02_towers.html)

Scientific American. (2017). *The Tower of Hanoi*. Retrieved February 12, 2023, from:  
<https://www.scientificamerican.com/article/the-tower-of-hanoi/>  
TutorialsPoint. (n.d.). *Prolog - Basics*. Retrieved February 10, 2023, from:  
[https://www.tutorialspoint.com/prolog/prolog\\_basics.htm](https://www.tutorialspoint.com/prolog/prolog_basics.htm)  
variable. (n.d.). Retrieved February 12, 2023, from:  
<http://www.cse.unsw.edu.au/~billw/dictionaries/prolog/variable.html>