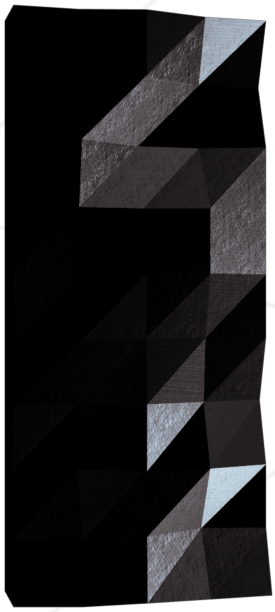


Cluedo

An introductory rush to logic programming

42 Staff pedago@42.fr

Summary: This document contains instructions for the Cluedo rush.

Contents

Chapter I

Foreword

```
KATIE: In my world everyone is a pony, and they all eat rainbows,
and poop butterflies.

HORTON: A person's a person, no matter how small.
```

Unable to choose which of these quotes was most relevant, we put both.

Chapter II

Introduction

This rush aims to introduce you to logical programming with Prolog.

Logic programming is a declarative programming style : the program describes facts and rules, and leaves the problem's resolution to the interpreter's care.

A Prolog program completely removes the procedure to solve the problem, it is content to describe what is true, and the inference engine resolves the queries by confronting them to the knowledge base. For this reason, a Prolog program is often much shorter than a procedural program solving the same problem.

This rush is progressive. First, you need to work on a little exercise to understand the syntax of the language. Then you have a classic puzzle to solve. And finally comes the part really playful : the resolution of a riddle a little touffue !

You will find that the code to be rendered each time is ultimately not very long, it is "enough" (certainly easier to say than to do!) to understand what is happening. Your work in pairs will therefore be mainly a work of understanding and sharing of knowledge.

Discuss ! Exchange ! Communicate !

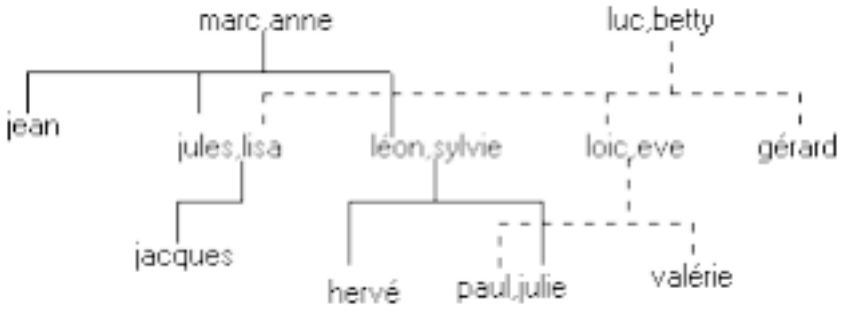
Chapter III

Step 1: Feed the knowledge base

The first day, there was the knowledge base.

You must build the knowledge base corresponding to the schema below, by setting the following predicates :
- woman/1
- man/1
- husband_of/2
- wife_of/2
- father-in-law_of/2
- mother-in-low_of/2
- child_of/2

As well as the ancestor_of/2 predicate, which must be usable on a family tree of any size.



Hint : if you are clever, you will define some of these predicates as rules...

Be very rigorous, your knowledge base will be heavily questioned for support. It must be perfect and exhaustive! The declaration of knowledge is a crucial step on the way to solving a problem.

Chapter IV

Step 2: Handle lists

Now that you know how to declare a set of knowledge, you will be able to start solving problems such as, for example, integrals.

Your next task is therefore the resolution of a great classic, known as Einstein's Enigma. There she is :

```
There are five houses of five different colors lined up along a road.
In each of these houses lives a person of different nationality.
Each of these people drinks a different drink, smokes a different cigar
and has a different pet.

- The Englishman lives in the red house.
- The Swede has dogs.
- The Dane drinks tea.
- The green house is to the left of the white house.
- The owner of the green house drinks coffee.
- The person who smokes Pall Mall has birds.
- The owner of the yellow house smokes Dunhill.
- The person living in the center house drinks milk.
- The Norwegian lives in the first house.
- The man who smokes Blend lives next to the one who has cats.
- The man who has a horse is the neighbor of the one who smokes Dunhill.
- The owner who smokes Blue Master drinks beer.
- The German smokes of the prince.
- The Norwegian lives right next to the blue house.
- The man who smokes Blend has a neighbor who drinks water.

Question : Who has the fish ?
```

That's right, who has the fish? Your Prolog program must provide the final set of knowledge, completed.

```
Little hint :
The simplest solution is to have a list of houses, each of
these houses containing variables representing the different
characteristics of a house.  The general rule that seeks the solution
would thus try to unify for each of these variables the clues given
to him.  The idea is "I have a list of houses such as one of its
elements has such and such a characteristic, a other of its elements
to such and such characteristic, etc."
```

Chapter V

Step 3: The goat, the wolf and the cabbage

And finally we arrive at the culmination of our logical epic : the resolution of the famous enigma of the goat, the wolf and the cabbage !

Here is the statement of this riddle:

```
A brave peon is on the edge of a river with his goat, his wolf, and a cabbage. He has
only a skinny boat to cross the river, and this boat can only support
two beings / objects at a time. Obviously, it is always  need someone to bring back
to advance the boat.

To this is added the harsh law of nature: he can not leave goat and cabbage together
without supervision (the goat would eat the cabbage), nor leave the goat with the
wolf (the goat would eat the wolf - what, you have not seen
BlackSheep ?).

How to get everyone on the other side without bloodshed ?
```

Your program must start with an predicate arity which equals 0 and have successive states that solve the problem. There must be one and only one solution.

```
Attention on XGP you are obliged to put a newline ('nl') after a
write.
```

Chapter VI

Bonus part

Bonuses will only be evaluated if Steps 1 and 2 (at least) are perfect.

As a bonus, we propose to solve the riddles of your choice, even to create them! Have fun, share on the forum the riddles that interested you, create new ones for your colleagues...

Masturbate your brains, it does not make you deaf. :)

Chapter VII

Instructions and advices

- You will use the Prolog XGP interpreter on dumps.
- There are well-developed resources on the internet to get started and understand how the Prolog inference engine works. www.learnprolognow.org is one example, but there are others : share your resources!
- You are entitled to predicates predefined by Prolog, including write/1 and list manipulation predicates.
- For the three imposed exercises, you must make the files `step1.pl`, `step2.pl` and `step3.pl`.
- You are free to name your bonuses as you see fit.
- You will have, in defense, comment and explain your (some) lines of code. You are not asked to capture all the subtleties of the inference engine, but if you are not able to explain the logic of your program, you will have 0.
- You must return to the root of your rendering repository a file named `author` containing the logins of the two members of the group followed by a '\n', such as :

```
$> cat -e ./author
login1$
login2$
$>
```

Chapter VIII

Turn-in and peer-evaluation

Turn in your work using your GiT repository, as usual. Only the work that's in your repository will be graded during the evaluation.