CECS 342 Assignment 4 - Logic Languages with SWI-Prolog

August 23, 2021

Assignment Description. The purpose of this assignment is to give you some experience programming with logic programming. You will be using SWI-Prolog for this assignment.

Part One - Logic Puzzles. To successfully complete this program, you will have to use lists in Prolog and provide solutions to the following logic problems along with your Prolog source code.

The three logic puzzles are as follows:

It's a Tie. I have provided a sample solution for this one on Beachboard. Use this as a model for your own solutions.

Imaginary Friends. The first part of your assignment is to write (and turn in) a Prolog program to solve this logic puzzle. Once you understand the "It's a Tie" solution, this should be easy. Do this one first.

Star Tricked. The second part is to write (and turn in) a Prolog program to solve this logic puzzle. This one is a little trickier--you need to figure out how to describe the ordering of weekdays--but it's not too hard.

These problems make unusually extensive use of the NOT operator, `\+`. Here's what you need to know about this operator:

- `\+` works as you would expect when all variables are completely instantiated (bound).
- Prolog tries to prove things, so if expression `E` contains unbound variables, `\+E` will try to find variable bindings that will make `E` true.

It is easy to check whether a solution is correct: Just check whether it satisfied each of the numbered rules in the problem.

IT'S A TIE BY JENNY ROBERTS ★

Four colleagues recently got into a discussion about some of the flamboyant patterns showing up on neckties these days. As a joke, each man arrived at work the next day sporting the most ridiculous tie he could find (no two men wore ties with the same pattern — one tie was decorated with smiling cupids). None of the men had to venture outside of his own closet, as each had received at least one such tie from a different relative! From the following clues, can you match each man with the pattern on his flamboyant tie, as well as determine the relative who presented each man with his tie?

Solution is on page 54.

1. The tie with the grinning leprechauns wasn't a present from a daughter.
2. Mr. Crow's tie features neither the dancing reindeer nor the yellow happy faces.
3. Mr. Speigler's tie wasn't a present from his uncle.
4. The tie with the yellow happy faces wasn't a gift from a sister.
5. Mr. Evans and Mr. Speigler own the tie with the grinning leprechauns and the tie that was a present from a father-in-law, in some order.
6. Mr. Hurley received his flamboyant tie from his sister.

	Cupids	Happy faces	Leprechauns	Reindeer	Daughter	Father-in-law	Sister	Uncle
Mr. Crow								
Mr. Evans								
Mr. Hurley								
Mr. Speigler								
Daughter								
Father-in-law								
Sister								
Uncle								

IMAGINARY FRIENDS BY KEITH KING ★

Grantville's local library recently sponsored a writing contest for young children in the community. Each of four contestants (including Ralph) took on the task of bringing to life an imaginary friend in a short story. Each child selected a different type of animal (including a moose) to personify, and each described a different adventure involving this new friend (one story described how an imaginary friend had formed a rock band). From the following clues, can you match each young author with his or her imaginary friend and determine the adventure the two had together?

Solution is on page 54.

1. The seal (who isn't the creation of either Joanne or Lou) neither rode to the moon in a spaceship nor took a trip around the world on a magic train.
2. Joanne's imaginary friend (who isn't the grizzly bear) went to the circus.
3. Winnie's imaginary friend is a zebra.
4. The grizzly bear didn't board the spaceship to the moon.

	Grizzly bear	Moose	Seal	Zebra	Circus	Rock band	Spaceship	Train
Joanne								
Lou								
Ralph								
Winnie								
Circus								
Rock band								
Spaceship								
Train								

STAR TRICKED BY KEITH KING ★

Last week, four UFO enthusiasts made sightings of unidentified flying objects in their neighborhood. Each of the four reported his or her sighting on a different day, and soon the neighborhood was abuzz with rumors of little green men. By the weekend, though, the government stepped in and was able to give each person a different, plausible explanation of what he or she had "really" seen. Can you determine the day (Tuesday through Friday) each person sighted a UFO, as well as the object that it turned out to be?

Solution is on page 54.

	Ms. Barrada	Ms. Gort	Mr. Klatu	Mr. Nikto	Balloon	Clothesline	Frisbee	Water tower
Tuesday								
Wednesday								
Thursday								
Friday								
Balloon								
Clothesline								
Frisbee								
Water tower								

1. Mr. Klatu made his sighting at some point earlier in the week than the one who saw the balloon, but at some point later in the week than the one who spotted the Frisbee (who isn't Ms. Gort).
2. Friday's sighting was made by either Ms. Barrada or the one who saw a clothesline (or both).
3. Mr. Nikto did not make his sighting on Tuesday.
4. Mr. Klatu isn't the one whose object turned out to be a water tower.

Deliverables. Upload your *.pl* source code and a text file with the queries asked to solve each puzzle to Beachboard Dropbox. Your assignment **will not be graded** unless the query file is included along with your source code.

Part Two - Adventure Game. Remember back to the first week of class? Guess what? You get to make another adventure game! This time, it'll be in Prolog.

Your assignment is to write an **adventure game** in SWI-Prolog (almost like we did with Inform!). The requirements are exactly the same as the Inform assignment: pick any theme you like for your adventure game: rescue, survival, treasure hunt, "a day in the life," or whatever else appeals to you.

Copy the file *adventure.pl* (on Beachboard) and use it as a starting point. This is an absolutely boring game consisting of one room, one object, and one direction you can go (but going in that direction takes you back to the same room). Add to this code to create your own game; if it doesn't do what you want, fix it so that it does. This is free code, to use or modify any way you like. If you don't want to use it, that's okay too.

You can get additional ideas from the file *spider.pl* (also on Beachboard).

Your program should contain one (or more, if you like) of each of the following:

- **Locked door.** In its most boring form, you must find a key and use it to unlock a door, thus giving you access to one or more additional rooms. With a little more imagination: You aren't admitted without a badge. You need to buy a ticket. You must give the troll a gold piece before you can cross the bridge. Waving the magic wand causes the rainbow bridge to appear. Et cetera. Any sort of locked door puzzle will do.
- **Hidden object.** Boring form: You open a box and find something inside. More interesting: You break open a treasure chest. You use the combination to a safe. You peer into the crystal ball. You buy the candy bar from the vending machine. You disassemble the robot to get some part out of it.

- **Incomplete object.** Your flashlight needs batteries. Your gun needs bullets. Your car needs gas. Your bicycle has a flat tire. You need a computer to get at the information on a floppy disk. You are a zombie and need a brain.

- **Limited resouces.** You have a limited amount of time (to find the bomb before it goes off) or money (to buy the things you need), or food, drink, or sleep (so you don't collapse), or some other resource. Maybe you can find more resources in the game, maybe you can't. Depending on just what you decide to do, you may want to figure out how to do arithmetic in Prolog.

You should have a `start/0` predicate (similar to the one provided in the source code of the *adventure.pl* file) that I can use to start your game and find out what commands you have added. Don't make me look at the code to figure out how to run your program!

Also include an *inventory* command (abbreviation: `i`) to tell what the player is currently holding.

Play the game by running Prolog and typing queries into it. Prolog can easily read Prolog terms, but reading anything else is awkward, and not worth learning how to do. (However, if you want to ask the user for a number, you can use the `read(X)` predicate; a number is a term. Just remember to type a period after the number.)

You may simply re-create the game that you made for the first assignment (as long as it meets all of the requirements), or you may do something completely new.

Deliverables. Submit your *.pl* source code, a transcript of a sample run of your program, and a *readme.txt* file that briefly describes your game, and in particular briefly describes your locked door, hidden object, incomplete object, and limited resource. Make sure to also **include the query file** (or, at the very minmum, clear instructions on how to run your game) in order to receive credit.