Imperial College London – Department of Computing

MSc in Advanced Computing
MSc in Computing Science (Specialism)

531: Prolog
'Frog Puzzle' – Assessed

Issued: 21 November 2018 **Due:** 29 November 2018

Introduction

The aim of this exercise is to give you an introduction to the basics of the Sicstus `clpfd` library for CLP(FD). You will write a CLP(FD) program to solve a puzzle, and then compare it to a pure Prolog program for the same task. The puzzle involves arranging nine cards, so that the coloured frogs printed on them all line up with one another. You are provided with a sketch of a CLP(FD) solution. The puzzle and the solution were provided by Frank Kriwaczek.

What To Do

Obtain the Exercise Files

- Clone your skeleton repository: (replacing the occurrence of *`login`* with your DoC login):

  `git clone https://gitlab.doc.ic.ac.uk/lab1819_autumn/531_frog_`*`login`*`.git`

  Or, if you have set up ssh key access you can use:

  `git clone git@gitlab.doc.ic.ac.uk:lab1819_autumn/531_frog_`*`login`*`.git`

- This will create a new directory called `531_frog_`*`login`*. Inside you will find the following files / directories:
  - `Frog Puzzle.pdf` — This file describes the frog puzzle.
  - `frog.pl` — this is the source file you should edit. It contains partial programs for you to complete.
  - `.git` and `.gitignore`

Part 1 (70% of the marks for the exercise)

Open the `Frog Puzzle.pdf file` and read the description of the puzzle. Then complete the partial CLPFD program in `frog.pl` by adding to it where indicated. (The details may be slightly different from those of Frank Kriwaczek's solution.)

Part 2 (30% of the marks for the exercise)

The 'Frog Puzzle' is a nice example, but not ideal because it is also relatively easy to solve in Prolog, without using CLP(FD). You do this by interleaving the generation of candidate configurations with constraint checking: in this example it is quite easy to do this. The file `frog.pl` also contains the skeleton of a program `frogProlog/1`. Your task is to complete this program by adding to it where indicated.

Timings

If you would like to investigate the relative performance of your programs you can obtain execution times using the Sicstus system predicate `statistics/2`. When called with the keyword `runtime`, this predicate returns a list `[TEver,TLast]` of integers, where `TEver` is the elapsed CPU time (in milliseconds) since Prolog was started, and `TLast` is the elapsed CPU time (in milliseconds) since the last call to `statistics/2` with key `runtime`. So, to obtain timings for `N` executions of the `frog/1` program you could write:

```
frogN(N, Solution, T) :-
  statistics(runtime, _),
  frogN(N),
  statistics(runtime, [_,T]),
  frog(Solution).

frogN(0).
frogN(N) :-
  N > 0, M is N - 1,
  frog(_), !,
  frogN(M).
```

Timings are *not* part of the submission of this exercise. They are for your own interest.

Testing

Your work will be automatically tested. Before submitting it please ensure that:

- Your program is written in `frog.pl`. Ensure that your program loads the Sicstus `clpfd` library.
- Your program *COMPILES WITHOUT ERRORS* on the **Linux Sicstus** system installed on the lab machines.

- Please do *not* include error messages and other forms of output in your submitted programs besides those specified above. (You may include them in your own versions of course, but please do not include them in the submitted version.)

You can verify that your code runs and produces sensible output by requesting an autotest online via `https://teaching.doc.ic.ac.uk/labts`.

Submission

Submit By: 29 November 2018

Submission is a two stage process.

1. **Push To Gitlab.** Use `git add`, `git commit` and `git push` to update the Gitlab server with the changes you have made to the skeleton repository. Use `git status` to confirm that you have no local changes you have not pushed, and then inspect the files on Gitlab:

   `https://gitlab.doc.ic.ac.uk`

2. **Submit directly to CATE.** Go to LabTS (`https://teaching.doc.ic.ac.uk/labts`), find your list of commits for this exercise and click the **Submit to CATe** button for the commit you want to submit.

Assessment

This exercise is worth 15% of the marks allocated for coursework on this module. Your solutions will be marked mainly on the correctness of your programs. Marks will be deducted for overcomplicated solutions. No 'cuts' are necessary anywhere in your solutions to this exercise.

Return of Work and Feedback

This exercise will be marked and returned by **14 December 2018**. Feedback on your solution will be given on the returned copy.