# Constructive Logic (15-317), Fall 2024
# Assignment 9: Prolog

Constructive Logic Staff
(Instructor: Karl Crary)

Due: Wednesday, November 6, 2024, 11:59 pm

This is an Prolog coding assignment. Please submit a file named "`hw.pl`" to "Homework 9." You can find directions for running Prolog on Andrew at the end.

## 1 Coloring Maps

Graph coloring is an interesting problem in graph theory. A graph coloring is an assignment of colors to each vertex such that no two adjacent vertices have the same color. Of particular interest is a coloring using a minimum number of colors; this number is called the *chromatic number* of the graph. The four-color theorem states that any planar graph[1] can be colored using at most four colors. The theorem was proved in 1976 using a computer program, and has caused much controversy (is a computer proof really a proof?). It has since been formally verified using the Coq theorem prover in 2005.

As a consequence of this theorem, any map can be colored with at most four colors such that no adjacent regions have the same color. This is because every map can be represented by a planar graph, with one vertex for each region, and an edge between two vertices if and only if their corresponding regions are adjacent.

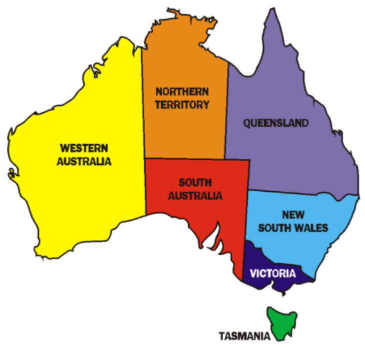

Figure 1: Australia (more colorful than necessary)

Consider, for example, Australia's map in Figure 1. Observe that this map uses more colors than necessary, although this might make it more visually appealing.

[1] A graph that can be drawn on the plane with no crossing edges.

**Task 1** (40 points). Implement a predicate `color_graph/3`, that is, a 3-argument predicate `color_graph(`*nodes*`, `*edges*`, `*colors*`)` that associates with the graph (*nodes*, *edges*) all of the valid 4-colorings of the graph. Submit your implementation in a file named `hw.pl`.

The query `color_graph(`*nodes*`, `*edges*`, `*coloring*`)` should check if *coloring* is a valid coloring, and the query `color_graph(`*nodes*`, `*edges*`, C)` should find every valid coloring `C`, one at a time, by backtracking. Note that `C` is a single coloring, not a list of colorings. (But each coloring **is** itself a list.)

- The *nodes* argument will be a list that looks something like:

  ```
  [node(a), node(b), node(c)]
  ```

  meaning that the valid nodes are `a`, `b`, and `c`.

- The *edges* argument will be a list that looks something like:

  ```
  [edge(a,b),edge(a,c), edge(b,c)]
  ```

  meaning there are edges connecting `a` to `b`, `a` to `c`, and `b` and `c`.

- The `coloring` result that you produce should look something like:

  ```
  [(a, red), (b, blue), (c, green)]
  ```

  meaning that `a` is colored red, `b` is colored blue, and `c` is colored green.

For efficiency reasons, you may prefer to find all valid colorings without repetition, but this is not required. Once all valid solutions have been found via backtracking, the predicate should fail.

Your implemementation should provide:

- A `color` predicate defined by the clauses:

  ```
  color(red).
  color(blue).
  color(green).
  color(yellow).
  ```

  This is included in the starter code.

- The code for the `color_graph` predicate, together with any helper predicates you choose to use. The `color_graph` predicate should have mode `color_graph(+`*nodes*`, +`*edges*`, -`*coloring*`)`, but you do not need to prove it.

Your solution does not need to be very long. The reference solution is just 19 lines of Prolog, including the color definitions.

For your convenience, we have provided you with a Prolog script (`coloring_tests.pl`) to test your implementation. You can use it by loading it into Prolog (after loading your code), and running `test.`

## 2 Return of the Theorem Prover

Now that you are experts in implementing G4ip in Standard ML, it is time to try implementing it in Prolog.

**Task 2** (60 points). Implement a predicate `prove/1`, that is, a one-argument predicate `prove(`*proposition*`)`, that proves constructive logic propositions. The `prove` predicate should have mode `prove(+`*proposition*`)`, but you do not need to prove it. Include your implementation as part of your `hw.pl` file.

You should represent propositions in Prolog using the following conventions. Remember, Prolog is untyped, so misspelling an operator will not generate an error, even though your program will fail to work properly.

| connective | blackboard | Prolog |
|---|---|---|
| truth | $T$ | `tt` |
| falsity | $F$ | `ff` |
| and | $A \wedge B$ | `and(A, B)` |
| or | $A \vee B$ | `or(A, B)` |
| implies | $A \supset B$ | `imp(A, B)` |
| atomic propositions | $P$ | `atom(p)` |

**We recommend that you implement your new theorem prover from scratch. Do not translate your Standard ML implementation into Prolog! You will make the problem much more difficult, because this problem is designed to be directly expressible as a logic program.**

For your convenience, we have provided you with a Prolog script (`prover_tests.pl`) to test your implementation. You can use it by loading it into Prolog (after loading your code), and running `test`. **Do not include the testing script in your submission.**

## A Running Prolog

To run Prolog on Andrew:

1. SSH into `unix.andrew.cmu.edu`.
2. Execute this command: `/afs/andrew/course/15/317/bin/317setup`

   This will add a line to your `.bashrc` file to add the 317 `bin` directory to your path. If you use a shell other than Bash, you may need to do something slightly different.
3. Run Prolog with the `prolog` command.
4. Load your prolog file using "`consult(hw).`" or just "`[hw].`" Do not say `hw.pl` here; the `.pl` extension is assumed.
5. Enter queries.
6. Exit Prolog using control-D or "`halt.`"

## B G4ip rules

The rules of G4ip (a.k.a. Dyckhoff's contraction-free sequent calculus) are as follows. In these rules, $P$ stands for atomic propositions.

$$\frac{P \in \Delta}{\Delta \longrightarrow P}\ \mathit{init}$$

$$\frac{\Delta, A, B \longrightarrow C}{\Delta, A \wedge B \longrightarrow C}\ {\wedge}L \qquad \frac{\Delta \longrightarrow A \quad \Delta \longrightarrow B}{\Delta \longrightarrow A \wedge B}\ {\wedge}R$$

$$\frac{\Delta, A \longrightarrow C \quad \Delta, B \longrightarrow C}{\Delta, A \vee B \longrightarrow C}\ {\vee}L \qquad \frac{\Delta \longrightarrow A}{\Delta \longrightarrow A \vee B}\ {\vee}R1 \qquad \frac{\Delta \longrightarrow B}{\Delta \longrightarrow A \vee B}\ {\vee}R2$$

$$\frac{\Delta \longrightarrow C}{\Delta, T \longrightarrow C}\ \mathsf{T}L \qquad \frac{}{\Delta \longrightarrow T}\ \mathsf{T}R \qquad \frac{}{\Delta, F \longrightarrow C}\ \mathsf{F}L$$

$$\frac{\Delta, A \longrightarrow B}{\Delta \longrightarrow A \supset B}\ {\supset}R$$

The rules above are the same as in the reduced sequent calculus. The following are the strange rules of G4ip that replace ${\supset}L$:

$$\frac{\Delta, A \longrightarrow D}{\Delta, \mathsf{T} \supset A \longrightarrow D}\ \mathsf{T}{\supset}L \qquad \frac{\Delta \longrightarrow D}{\Delta, \mathsf{F} \supset A \longrightarrow D}\ \mathsf{F}{\supset}L$$

$$\frac{\Delta, A \supset B \supset C \longrightarrow D}{\Delta, (A \wedge B) \supset C \longrightarrow D}\ {\wedge}{\supset}L \qquad \frac{\Delta, A \supset C, B \supset C \longrightarrow D}{\Delta, (A \vee B) \supset C \longrightarrow D}\ {\vee}{\supset}L$$

$$\frac{P \in \Delta \quad \Delta, A \longrightarrow D}{\Delta, P \supset A \longrightarrow D}\ P{\supset}L$$

$$\frac{\Delta, B \supset C, A \longrightarrow B \quad \Delta, C \longrightarrow D}{\Delta, (A \supset B) \supset C \longrightarrow D}\ {\supset}{\supset}L$$

The rules ${\wedge}L$, ${\wedge}R$, ${\vee}L$, $\mathsf{T}L$, $\mathsf{T}R$, $\mathsf{F}L$, ${\supset}R$, $\mathsf{T}{\supset}L$, $\mathsf{F}{\supset}L$, ${\wedge}{\supset}L$, and ${\vee}{\supset}L$ are asynchronous. The rules *init*, ${\vee}R1$, ${\vee}R2$, $P{\supset}L$, and ${\supset}{\supset}L$ are synchronous.

## C G4ip in the Inversion Calculus

Three judgements indicate different states in the proof-search process. First decompose $A$ on the right, then decompose members of $\Omega$ on the left, then search through synchronous rules:

$$\begin{array}{ll} \text{right inversion} & \Delta;\Omega \xrightarrow{R} A \\ \text{left inversion} & \Delta;\Omega \xrightarrow{L} C^+ \\ \text{search} & \Delta \xrightarrow{S} C^+ \end{array}$$

Here, $P$ refers only to atomic propositions, $A^+$ (etc.) refers to right synchronous propositions, $A^-$ (etc.) refers to left synchronous propositions, $\Delta$ contains only left synchronous propositions, and $\Omega$ contains arbitrary propositions. Left and right synchronous propositions are given by the grammar:

$$\begin{array}{lll} \text{left synchronous} & A^- & ::= \; P \mid P \supset B \mid (B \supset C) \supset D \\ \text{right synchronous} & A^+ & ::= \; P \mid B \vee C \mid F \end{array}$$

**Right Inversion**

$$\frac{\Delta;\Omega \xrightarrow{R} A \quad \Delta;\Omega \xrightarrow{R} B}{\Delta;\Omega \xrightarrow{R} A \wedge B}\wedge R \qquad \frac{}{\Delta;\Omega \xrightarrow{R} T}\mathsf{T}R \qquad \frac{\Delta;\Omega, A \xrightarrow{R} B}{\Delta;\Omega \xrightarrow{R} A \supset B}\supset R$$

**Switch**

$$\frac{\Delta;\Omega \xrightarrow{L} P}{\Delta;\Omega \xrightarrow{R} P} LR_P \qquad \frac{\Delta;\Omega \xrightarrow{L} A \vee B}{\Delta;\Omega \xrightarrow{R} A \vee B} LR_\vee \qquad \frac{\Delta;\Omega \xrightarrow{L} F}{\Delta;\Omega \xrightarrow{R} F} LR_F$$

**Left Inversion**

$$\frac{\Delta;\Omega, A, B \xrightarrow{L} C^+}{\Delta;\Omega, A \wedge B \xrightarrow{L} C^+}\wedge L \qquad \frac{\Delta;\Omega \xrightarrow{L} C^+}{\Delta;\Omega, T \xrightarrow{L} C^+}\mathsf{T}L \qquad \frac{\Delta;\Omega, A \xrightarrow{L} C^+ \quad \Delta;\Omega, B \xrightarrow{L} C^+}{\Delta;\Omega, A \vee B \xrightarrow{L} C^+}\vee L$$

$$\frac{}{\Delta;\Omega, F \xrightarrow{L} C^+}\mathsf{F}L \qquad \frac{\Delta;\Omega, A \xrightarrow{L} D}{\Delta;\Omega, \mathsf{T} \supset A \xrightarrow{L} D}\mathsf{T}{\supset}L \qquad \frac{\Delta;\Omega \xrightarrow{L} D}{\Delta;\Omega, \mathsf{F} \supset A \xrightarrow{L} D}\mathsf{F}{\supset}L$$

$$\frac{\Delta;\Omega, A \supset B \supset C \xrightarrow{L} D}{\Delta;\Omega, (A \wedge B) \supset C \xrightarrow{L} D}\wedge{\supset}L \qquad \frac{\Delta;\Omega, A \supset C, B \supset C \xrightarrow{L} D}{\Delta;\Omega, (A \vee B) \supset C \xrightarrow{L} D}\vee{\supset}L$$

**Shift**

$$\frac{\Delta, P;\Omega \xrightarrow{L} C^+}{\Delta;\Omega, P \xrightarrow{L} C^+}\mathit{shift}_P \qquad \frac{\Delta, P \supset B;\Omega \xrightarrow{L} C^+}{\Delta;\Omega, P \supset B \xrightarrow{L} C^+}\mathit{shift}_{P\supset} \qquad \frac{\Delta, (A \supset B) \supset C;\Omega \xrightarrow{L} D^+}{\Delta;\Omega, (A \supset B) \supset C \xrightarrow{L} D^+}\mathit{shift}_{\supset\supset}$$

**Search**

$$\frac{\Delta \xrightarrow{S} C^+}{\Delta;\cdot \xrightarrow{L} C^+}\mathit{search} \qquad \frac{P \in \Delta}{\Delta \xrightarrow{S} P}\mathit{init} \qquad \frac{\Delta;\cdot \xrightarrow{R} A}{\Delta \xrightarrow{S} A \vee B}\vee R1 \qquad \frac{\Delta;\cdot \xrightarrow{R} B}{\Delta \xrightarrow{S} A \vee B}\vee R2$$

$$\frac{P \in \Delta \quad \Delta; A \xrightarrow{L} D^+}{\Delta, P \supset A \xrightarrow{S} D^+}P{\supset}L \qquad \frac{\Delta; B \supset C, A \xrightarrow{R} B \quad \Delta; C \xrightarrow{L} D^+}{\Delta, (A \supset B) \supset C \xrightarrow{S} D^+}{\supset\supset}L$$