Yachay Tech University

Mathematical and Computational Logic

Prolog Lab 6: Map Coloring

1. Overview

In this lab you will model and solve the Map Coloring problem using Constraint Logic Programming over Finite Domains (CLP(FD)) in SWI-Prolog. You will first color the regions of Australia, then extend the model to color South America, experimenting with different numbers of colors and labeling strategies.

2. Learning Goals

- Translate a CSP into variables, domains, and constraints.
- Use CLP(FD) constraints (#=, #\=, #/<, #/\, #\/).
- Understand propagation vs. search and the role of labeling/2.
- Extend a base model to a new dataset (South America).

3. Quick Primer: CLP(FD)

Load the finite domain library with:

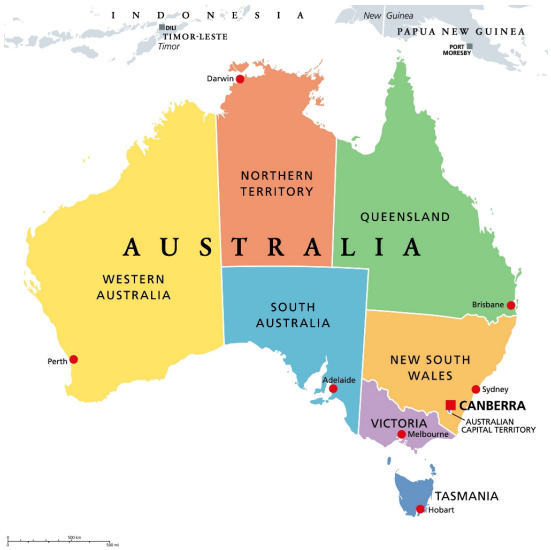
```
:- use_module(library(clpfd)).
```

Key ideas:

- Variables have integer domains: X in 1..4
- Constraints relate variables: A #\= B, A #= B+1
- Propagation narrows domains before search
- labeling/2 searches remaining choices and can optimize strategies

4. Australia Map (Adjacency)

We use common region codes: wa, nt, sa, q, nsw, v, t.



Reference Map

Adjacency facts (undirected; record each pair once):

```
adjacent(wa, nt).
adjacent(wa, sa).
adjacent(nt, sa).
adjacent(nt, q).
adjacent(sa, q).
adjacent(sa, nsw).
adjacent(sa, v).
adjacent(q, nsw).
adjacent(nsw, v).
```

5. Base Model Pattern

Variables: one color per region (integers 1..K). For example, K=3 for three colors.
Constraints: for every adjacency (A,B), ColorA #\= ColorB.
Search: labeling/2 assigns concrete colors.

6. Reference Predicate Design

We'll design these predicates:

- regions_au/1 — returns the ordered list of region atoms for Australia.
- edges_au/1 — returns a list of pairs A-B representing adjacencies.
- color_names/1 — color name mapping [1=red, 2=green, 3=blue, 4=yellow].
- map_color(Vars, Edges, K) — core constraint model for any map.
- colorize_au(K, Vars) — ties it all together for Australia.
- pretty_color_by_region(Regions, Vars) — prints Region=ColorName.

7. Running the Australia Model

Example queries after loading your .pl file:

?- colorize_au(3, Vars), writeln(Vars).

?- colorize_au(3, Vars), regions_au(Rs), pretty_color_by_region(Rs, Vars).

Try labeling strategies:

?- regions_au(Rs), edges_au(Es), map_color(Vs, Es, 3), labeling([ffc], Vs), pretty_color_by_region(Rs, Vs).

8. South America Extension

Define regions (order is up to you, e.g., ar, bo, br, cl, co, ec, gy, gfr, py, pe, su, uy, ve) and adjacency pairs (land borders).

Generate the color mapping for the South America region.

9. Lab Tasks

Part A — Australia (guided):

1) Implement regions_au/1 and edges_au/1.
2) Implement map_color/3 for K colors.
3) Solve with K=3 and K=4. Compare solutions.
4) Print Region=ColorName using pretty_color_by_region/2.

Part B — South America (extension to the previous code):

5) Implement regions_sa/1 and edges_sa/1 from adjacency facts.
6) Solve with K=3 and K=4; observe feasibility and solution variety.
7) Try labeling strategies: labeling([ffc]), labeling([min]), etc.

10. Tips and Troubleshooting

• If you see instantiation or domain errors, ensure domains are set with Var in 1..K before applying constraints.
• Use #\= for inequality, not =\= (which is arithmetic comparison, not a CLP(FD) constraint).
• Start simple (Australia), then extend to South America.
• Prefer compact adjacency lists and maplist to apply constraints cleanly.

11. Deliverables

• A .pl file that solves Australia and South America map coloring.
• Console printout of one valid coloring for each map (K=3 and K=4).