SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Week 3 Exercises

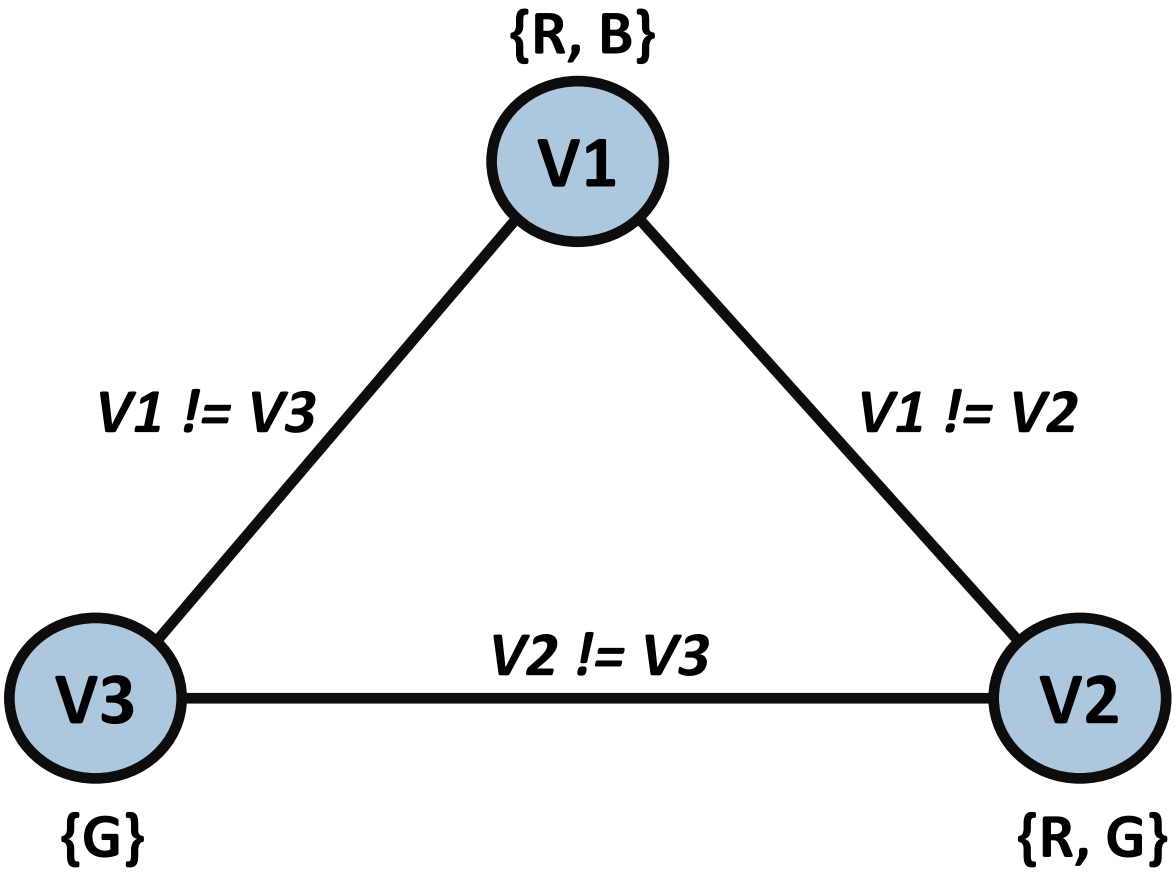
Heuristics and Constraint Satisfaction Problems

PROF. LIM KWAN HUI

50.021 Artificial Intelligence

The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.

Exercise: CSPs



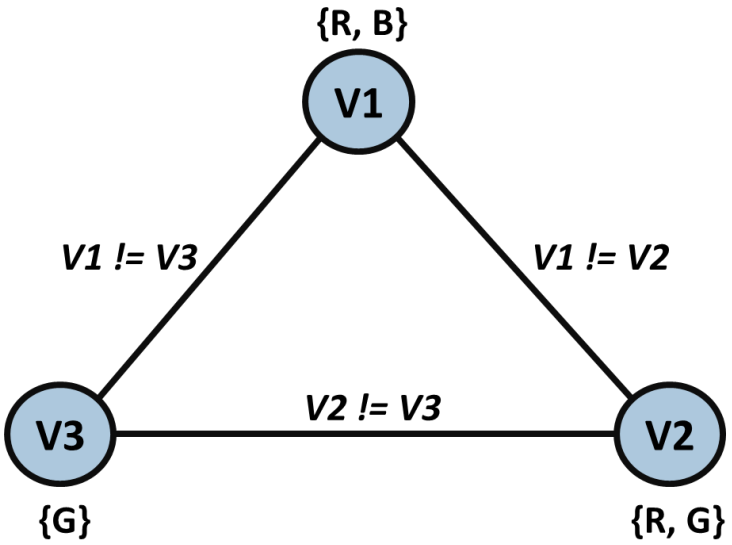
Exercise: CSPs

- Using the graph shown, run **Pure Backtracking**. Assume that we search based on a fixed variable ordering (V1, V2, then V3) and fixed value ordering (R, G, then B). Show all assignments even if it is immediately found to be inconsistent upon testing.

```
Must assign before you check constraint, cannot then
backtrack

        V1 {R}
  V2 {R}, V2 {G}
        V3 {G}


        V1 {B}
   V2 {R}, V2 {G}
 V3 {G}
```




```
ANS:
- V1 R
- V2 R (inconsistent assignment, backtrack)
- V2 G
- V3 G
- V1 B (backtrack to V1, because V2 only has G left, no other possible values)
- V2 R
- V3 G
```

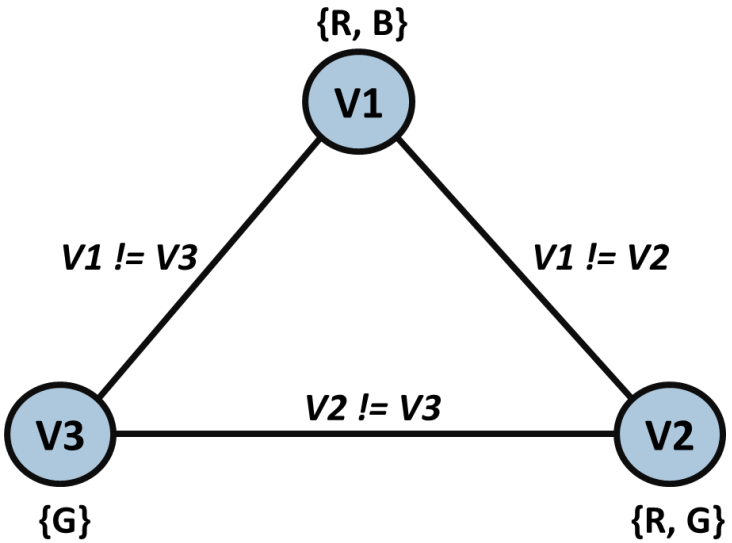
Exercise: CSPs

- Using the graph shown, run **Backtracking with Forward Checking**. Assume that we search based on a fixed variable ordering (V1, V2, then V3) and fixed value ordering (R, G, then B). Show all assignments even if it is immediately found to be inconsistent upon testing.

```
Check constraint, if cannot then don't assign

                V1 {R}
        V2 {R} – cannot, V2 {G}
                            V3 {G} – cannot


             V1 {B}
          V2 {R}
       V3 {G}
```




```
ANS:
- V1 R (check V2 R, cannot assign, delete inconsistent values R from neighbours)
- V2 G (check V3 G, cannot assign, delete inconsistent values G from neighbours)
- V1 B (backtrack to V1, because V2 only has G left, no other possible values)
- V2 R
- V3 G

solution: V1 B, V2 R, V3 G
```

Exercise: CSPs

- Using the graph shown, run the **AC-3 algorithm**. Show your steps, including the arcs/edges considered and the domains/values for each variable (first example given). List down the remaining domains/values for each variable after running AC-3.
  - V1→V2, D1=RB, D2=RG

```
X → Y is consistent iff for every value x of X there is some allowed y
need to check all edges in all directions

Queue: all edges

V1 -> V2, D1 = RB, D2 = RG (don't delete R from V1 yet, as there is
allowance)
V2 -> V1, D1 = RG, D1 = RB
V2 -> V3, D2 = RG, D3 = G (delete G from V2, add incoming edges to V2)
V3 -> V2, D3 = G , D2 = R (OK)
V1 -> V2, D1 = RB, D2 = R (delete R from V1, add incoming edges to V1)
V2 -> V1, D2 = R , D1 = B (OK)

remaining domains/values
V1 -> V2, D1 = B, D2 = R (OK)
V1 -> V3, D1 = B, D3 = G (OK)
V2 -> V3, D2 = R, D3 = G (OK)


solution: D1 = B, D2 = R, D3 = G
```