AI 2002
Artificial Intelligence

Dr. Hashim Yasin

Forward Checking

Arc Consistency

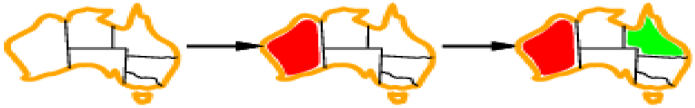
- A variable in a CSP is **arc-consistent** if every value in its domain satisfies the variable’s ***binary constraints***.
- ***Arc consistency*** *eliminates values from the domain of variables that can never be part of a consistent solution.*
- Directed arc $(\boldsymbol{V_i}, \boldsymbol{V_j})$ is arc consistent if

  $\forall \boldsymbol{x} \in \boldsymbol{D_i} \qquad \exists \boldsymbol{y} \in \boldsymbol{D_j}$

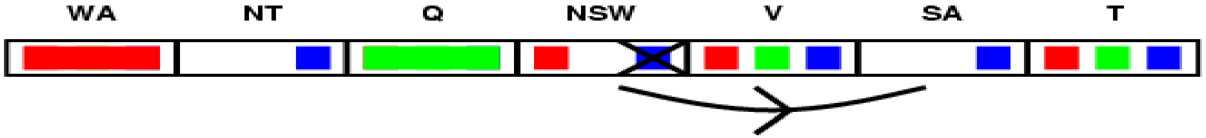
  such that $(\boldsymbol{x}, \boldsymbol{y})$ is allowed by constraint

Northern Territory
Western Australia
Queensland
South Australia
New South Wales
Victoria
Tasmania

- For every value

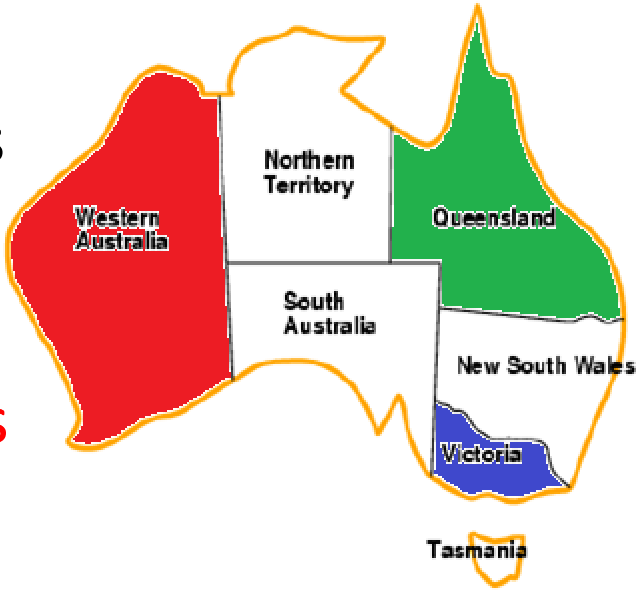


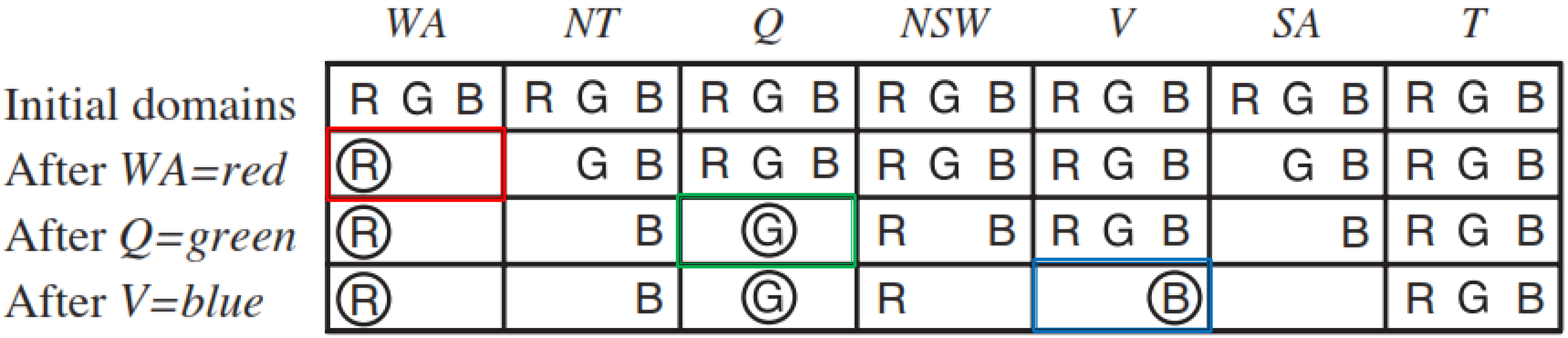
Inconsistant arc



Forward Checking

- Keep track of remaining legal values for unassigned variables
- Whenever a variable is assigned a value, the forward-checking process establishes **arc consistency** for it:

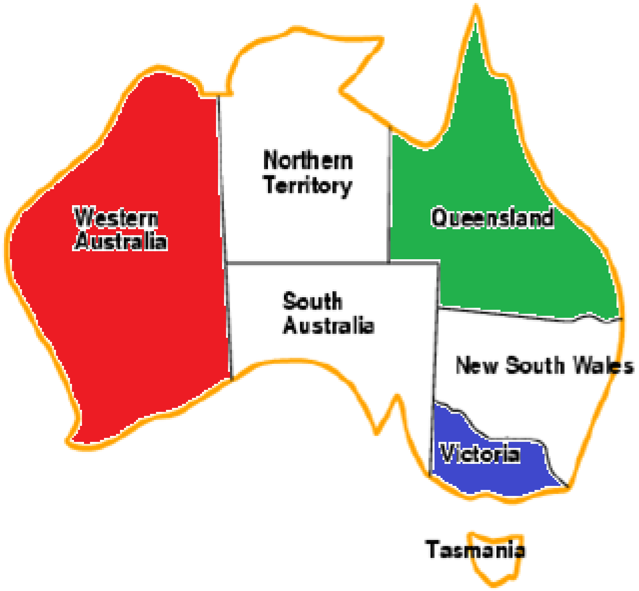




	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	Ⓡ	G B	R G B	R G B	R G B	G B	R G B
After *Q=green*	Ⓡ	B	Ⓖ	R B	R G B	B	R G B
After *V=blue*	Ⓡ	B	Ⓖ	R	Ⓑ		R G B

Forward Checking

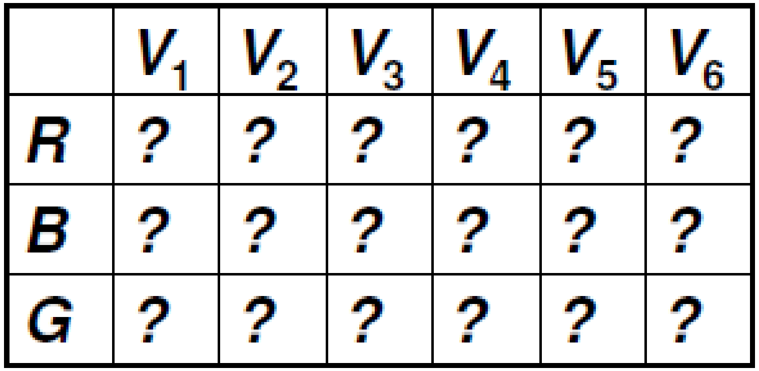
- Hence, forward checking has detected that the partial assignment {WA=red, Q=green, V =blue} is **inconsistent** and the algorithm will therefore backtrack immediately.



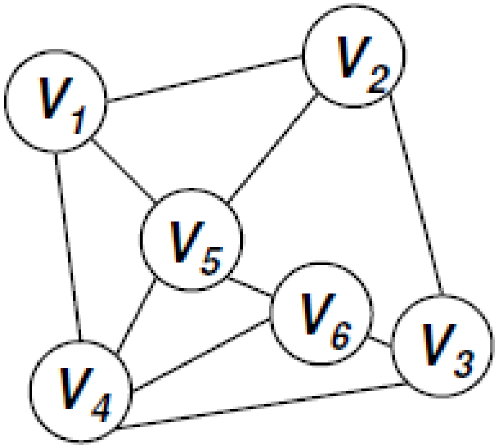
	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	Ⓡ	G B	R G B	R G B	R G B	G B	R G B
After *Q=green*	Ⓡ	B	Ⓖ	R B	R G B	B	R G B
After *V=blue*	Ⓡ	B	Ⓖ	R	Ⓑ		R G B

Forward Checking … Example 2

- Keep track of remaining legal values for unassigned variables
- Whenever a variable is assigned a value, the forward-checking process establishes **arc consistency** for it:

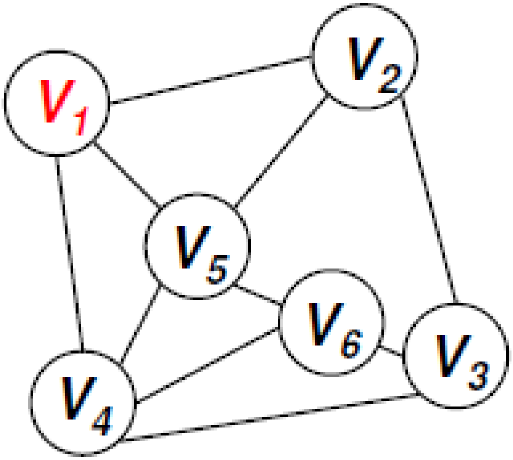


	V_1	V_2	V_3	V_4	V_5	V_6
R	?	?	?	?	?	?
B	?	?	?	?	?	?
G	?	?	?	?	?	?

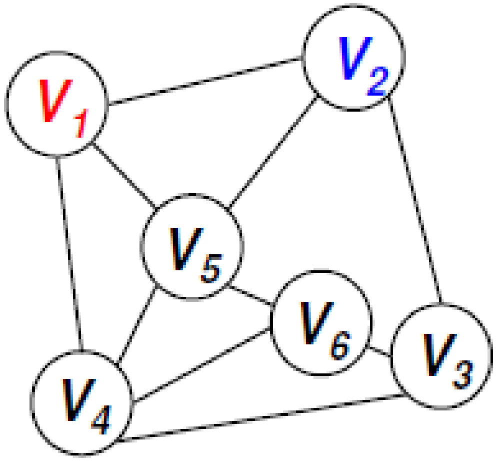


Forward Checking ... Example 2

	V_1	V_2	V_3	V_4	V_5	V_6
R	O	X	?	X	X	?
B		?	?	?	?	?
G		?	?	?	?	?

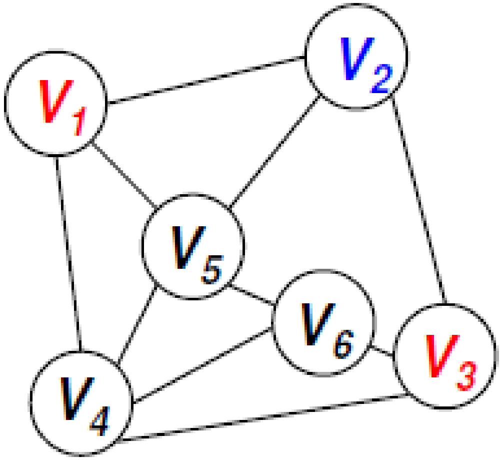


	V_1	V_2	V_3	V_4	V_5	V_6
R	O		?	X	X	?
B		O	X	?	X	?
G			?	?	?	?

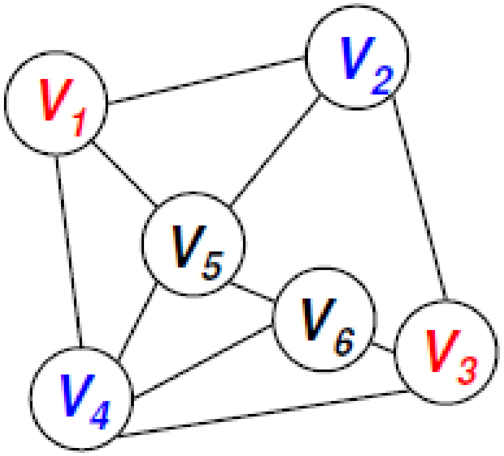


Forward Checking … Example 2

	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O	X	X	X
B		O		?	X	?
G				?	?	?

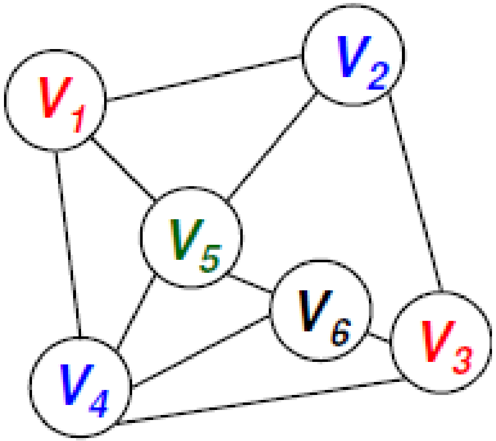


	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O		X	X
B		O		O	X	X
G					?	?



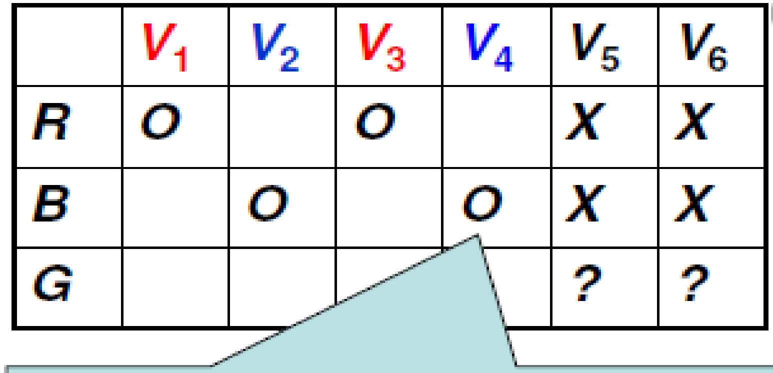
Forward Checking ... Example 2

	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O			X
B		O		O		X
G					O	X

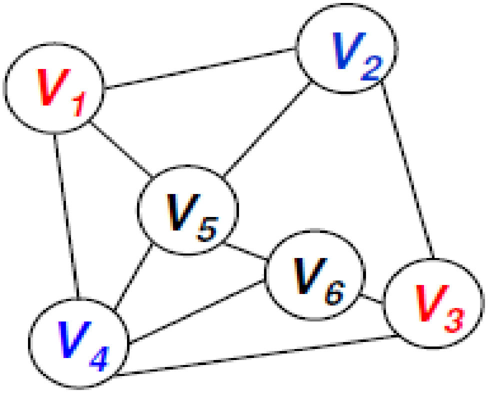


There are no valid assignments left for V_6 we need to backtrack

Forward Checking ... Example 2



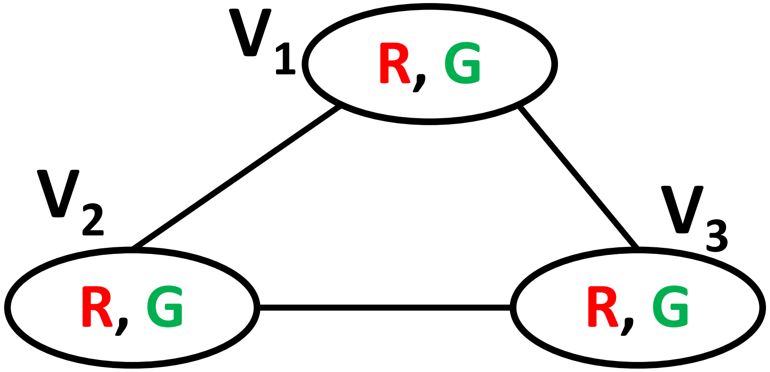
	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O		X	X
B		O		O	X	X
G					?	?



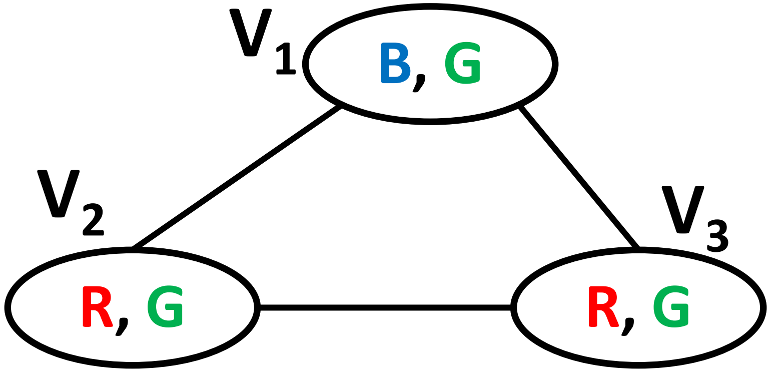
At this point, it is already obvious that this branch will not lead to a solution because there are no consistent values in the remaining domain for V_5 and V_6.

- Forward checking **does not detect all the inconsistencies**, only those that can be detected by looking at the constraints which contain the **current variable**.
- Can we look ahead further?

Arc consistency is not enough in general



Arc consistent but
NO solutions



Arc consistent but
TWO solutions
B, R, G
B, G, R

Need to do search to find solutions (if any)

Constraint Propagation

Constraint Propagation

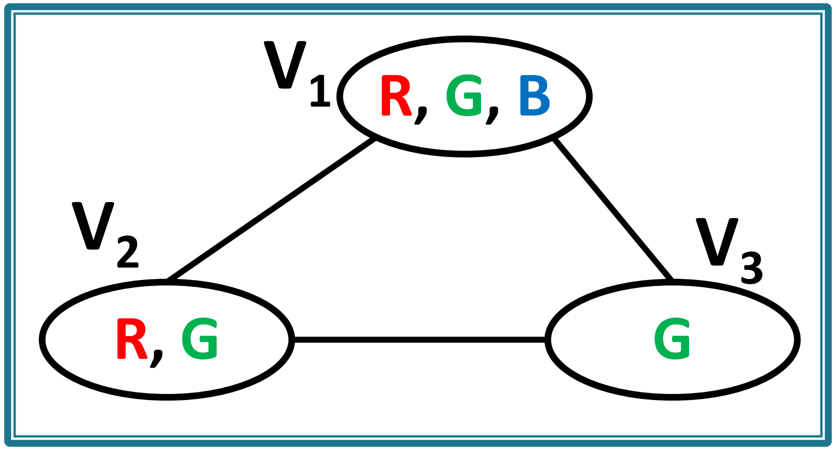
- ***V*** = variable being assigned at the current level of the search
- Set variable ***V*** to a value in ***D(V)***
- For every variable ***V'*** connected to ***V***:
  - Remove the values in ***D(V')*** that are **inconsistent** with the assigned variables
  - For every variable ***V"*** connected to ***V'***:
    - Remove the values in ***D(V")*** that are **no longer possible candidates**
    - And do this again with the variables connected to ***V"***
      - ……..until no more values can be discarded

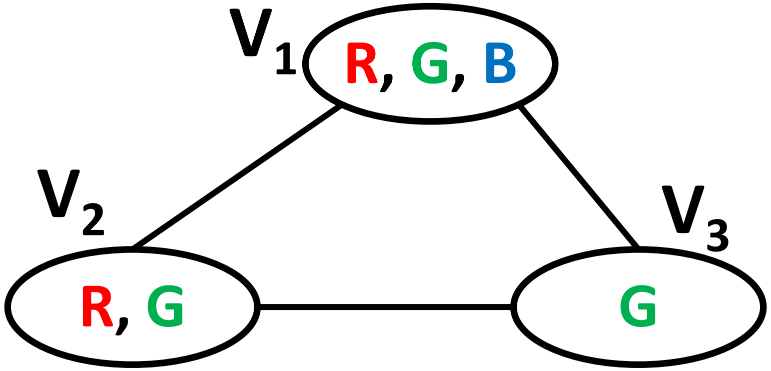
Forward Checking

Constraint Propagation

Constraint Propagation

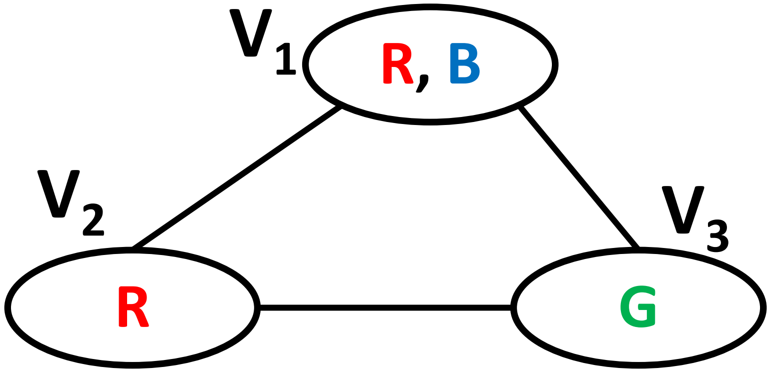
Arc examined	Value deleted
$V_1 - V_2$	None

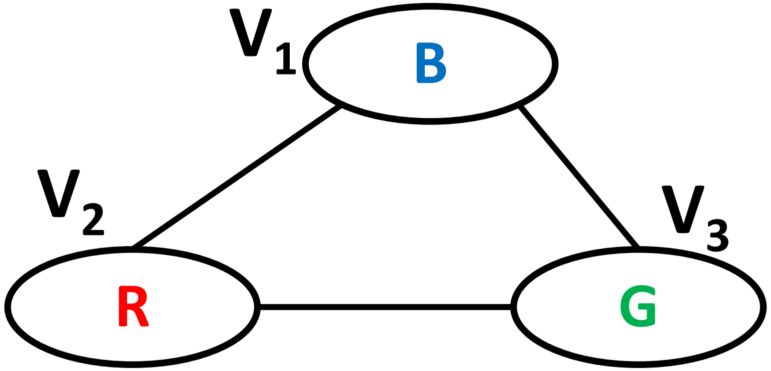




Constraint Propagation

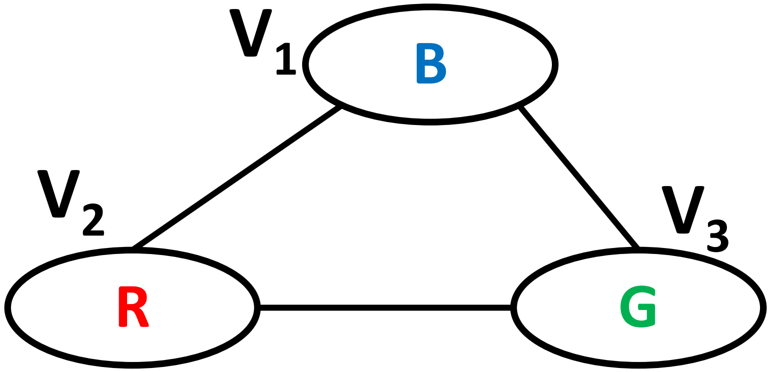
Arc examined	Value deleted
$V_1 – V_2$	None
$V_1 – V_3$	V_1 (G)
$V_2 – V_3$	V_2 (G)
$V_1 – V_2$	V_1 (R)





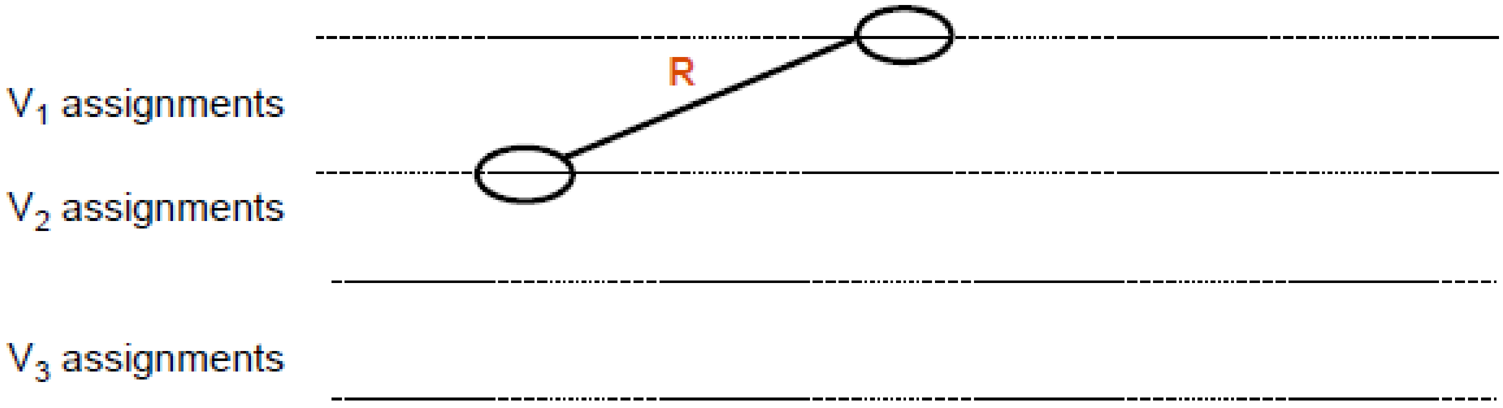
Constraint Propagation

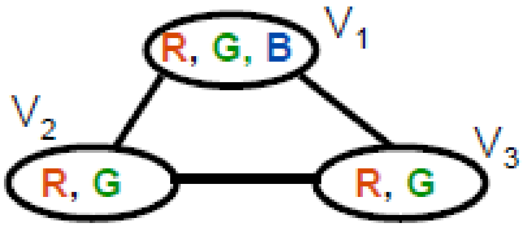
Arc examined	Value deleted
$V_1 - V_2$	None
$V_1 - V_3$	V_1 (G)
$V_2 - V_3$	V_2 (G)
$V_1 - V_2$	V_1 (R)
$V_1 - V_3$	None
$V_2 - V_3$	None



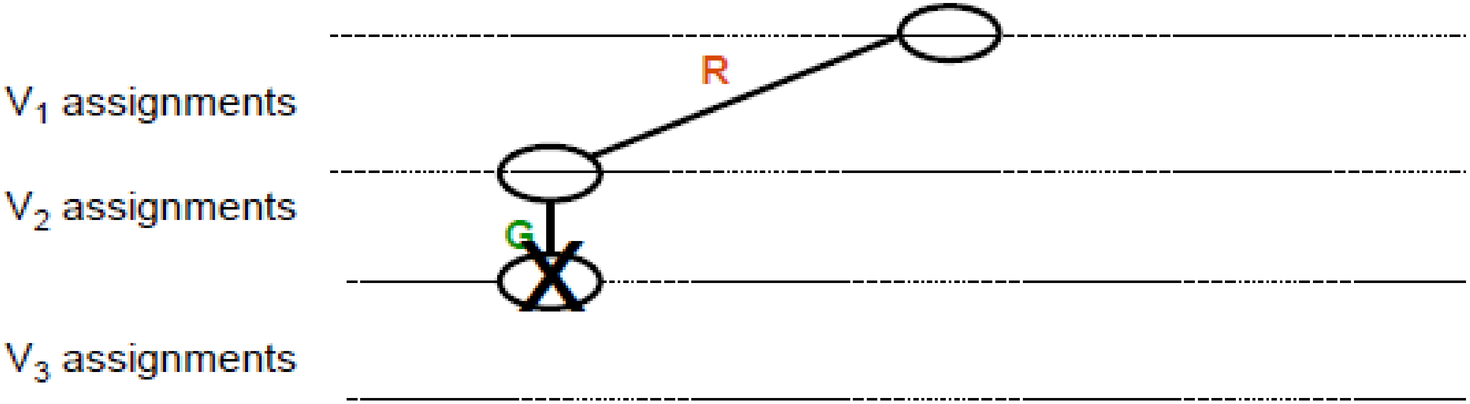
Forward Checking with Backtracking

Forward Checking with Backtracking

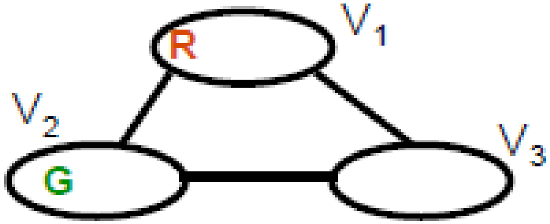




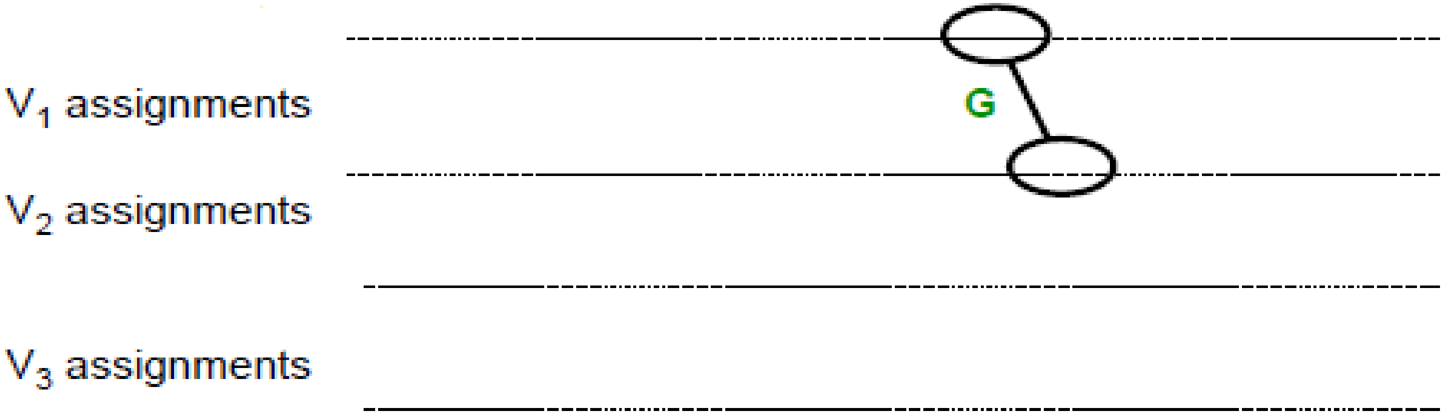
Forward Checking with Backtracking



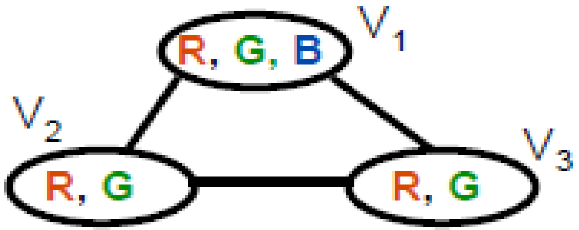
We have a conflict whenever a domain becomes empty.



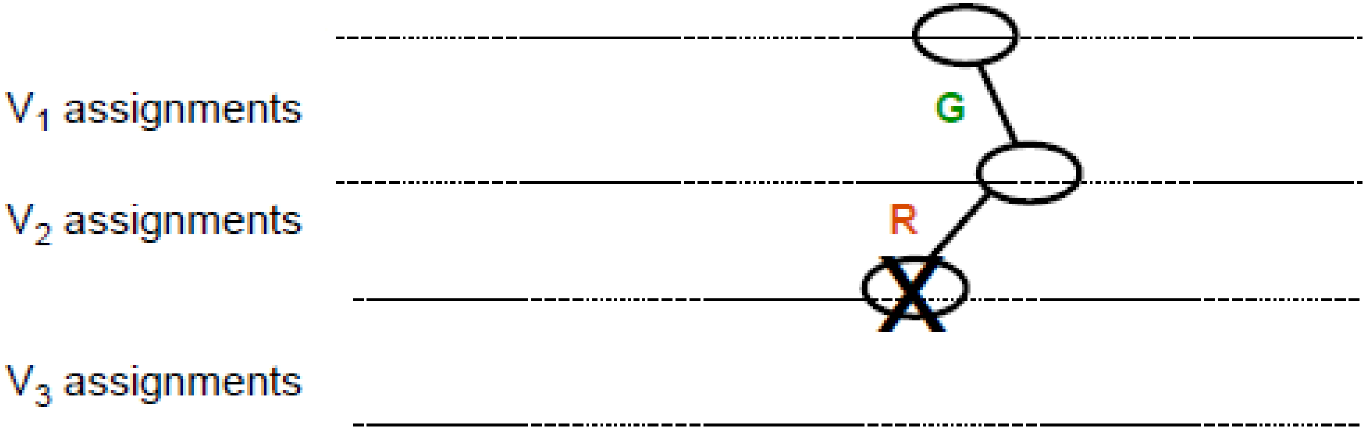
Forward Checking with Backtracking

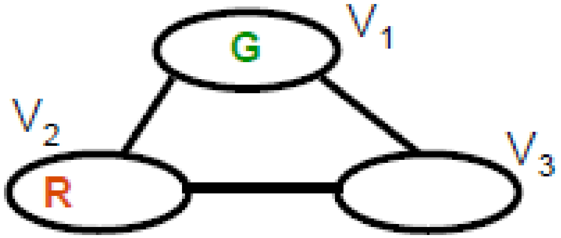


When backing up, need to restore domain values, since deletions were done to reach consistency with tentative assignments considered during search.

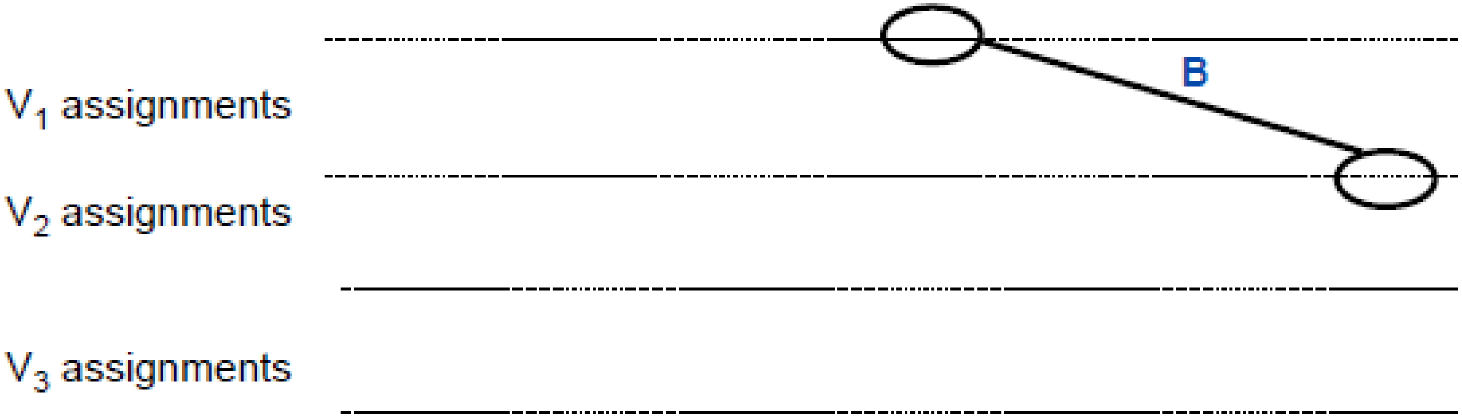


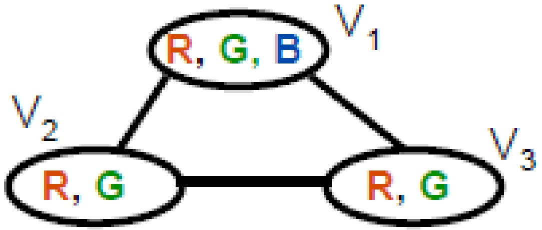
Forward Checking with Backtracking



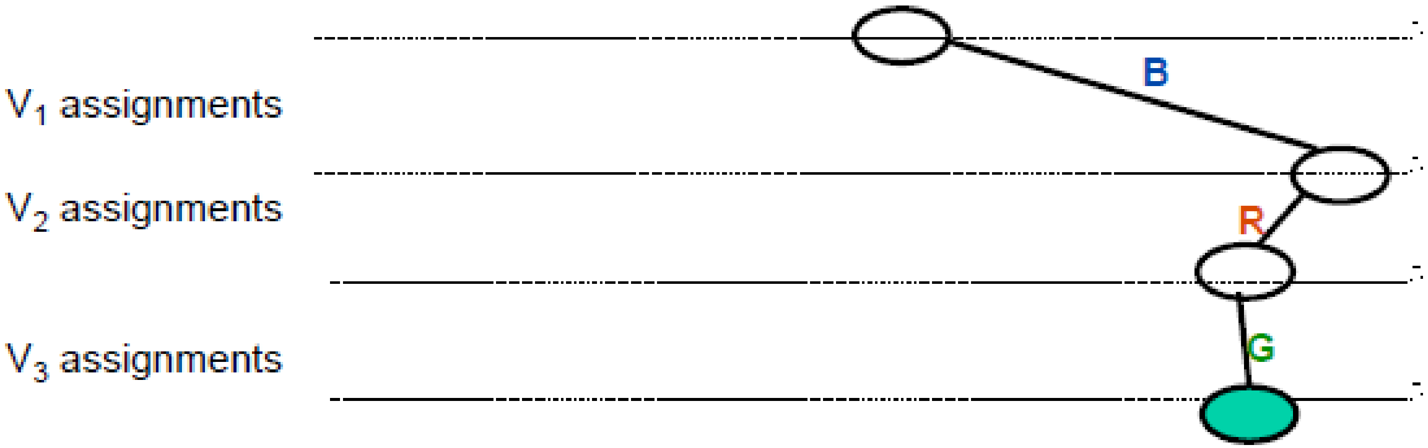


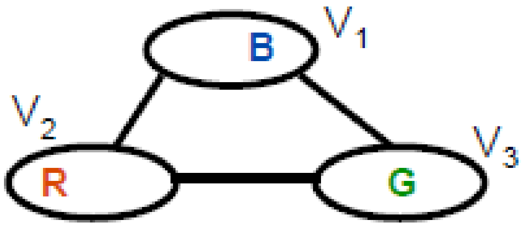
Forward Checking with Backtracking



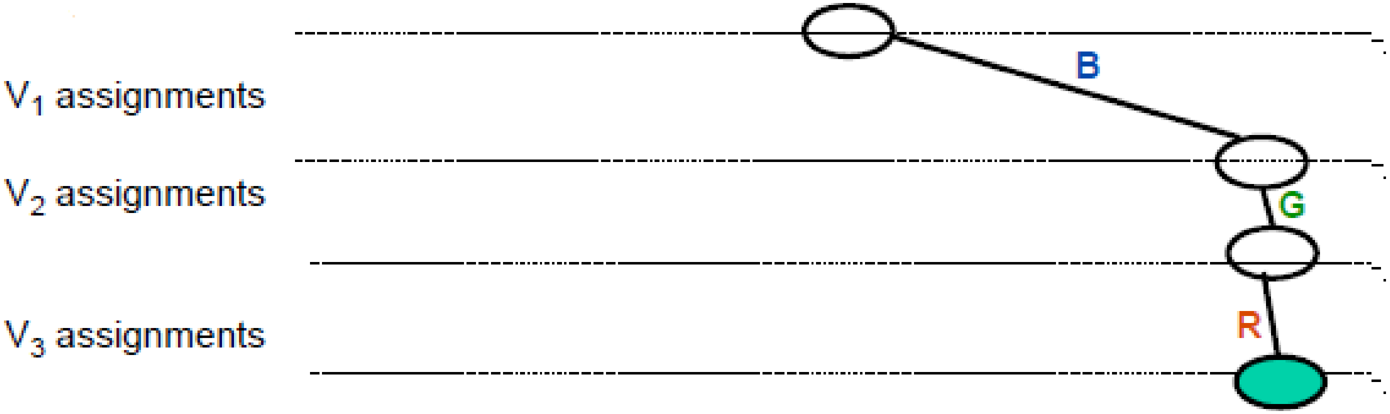


Forward Checking with Backtracking

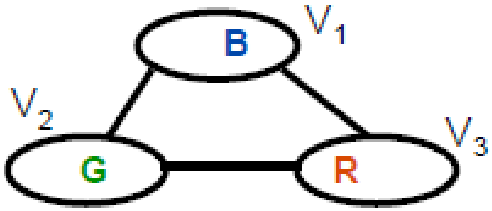




Forward Checking with Backtracking



No need to check previous assignments



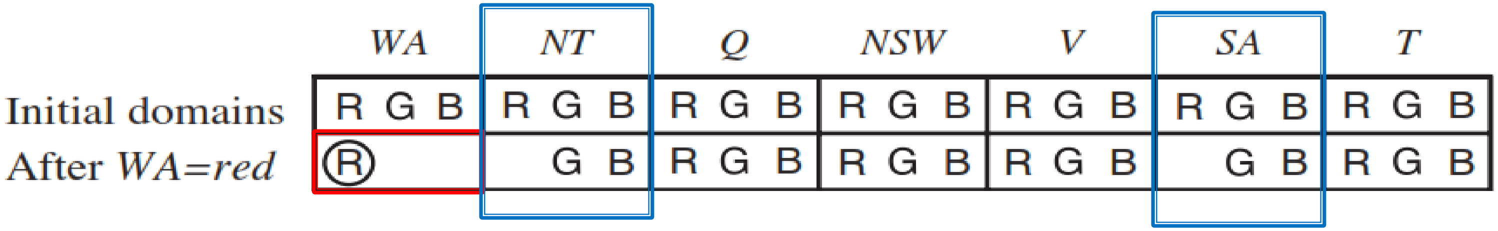
FC-BT with dynamic ordering

- Traditional **backtracking uses fixed ordering** of variables & values.
- The simplest strategy for selecting unassigned variable is to choose the next unassigned variable in order, $\{X_1, X_2, \ldots\}$.
- Other is the random order or place variables with many constraints first.
- Can be modified by choosing an **order dynamically** as the search proceeds.

FC-BT with dynamic ordering

Most Constrained Variable (Minimum Remaining Values (MRV)):

- when doing forward-checking, ***pick variable*** **with** ***fewest "legal"*** **values** to assign next (minimizes branching factor)
  - The MRV heuristic usually performs better than a random or static ordering, sometimes by a factor of 1,000 or more.

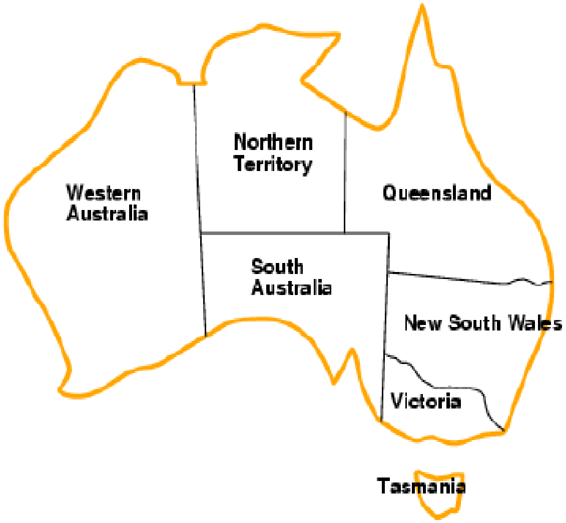


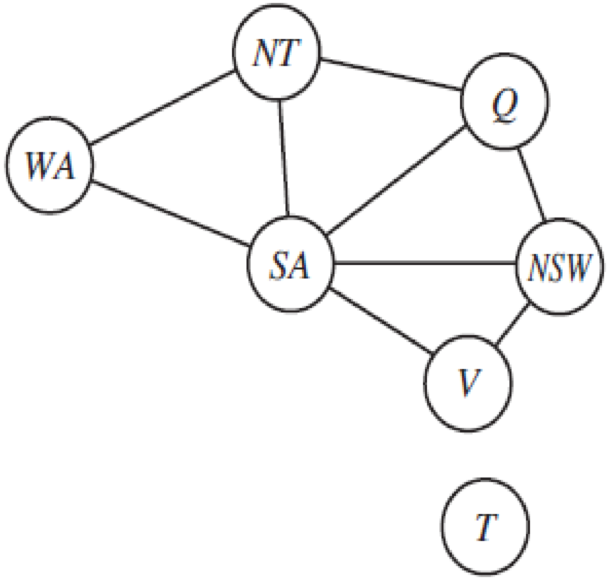
	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	Ⓡ	G B	R G B	R G B	R G B	G B	R G B

FC-BT with dynamic ordering

Degree Heuristic:

- It attempts to reduce the branching factor on future choices by *selecting the variable* that is involved in the ***largest number of constraints***.
  - SA is the variable with highest degree, 5.



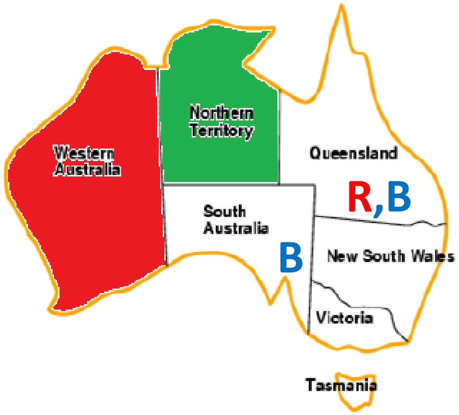


FC-BT with dynamic ordering

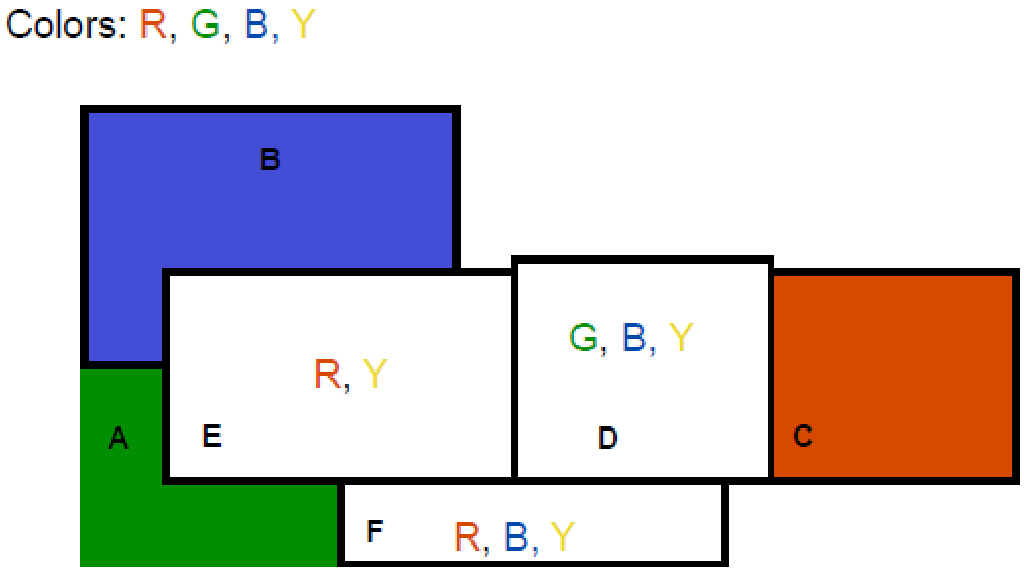
Least Constrained Value:

- **choose value that rules out the *smallest number* of values** in variables connected to the chosen variable by constraints.
- We have generated the partial assignment for **WA=red** and **NT =green**. What would be our next choice for **Q**. Blue would be a bad choice because it eliminates the last legal value left for Q's neighbor, SA. *The least-constraining-value heuristic therefore prefers **red** to blue*. *(eliminates fewest values from neighbouring domains)*

  - This combination improves feasible n-queens performance from about n = 30 with just FC to about n = 1000 with FC & ordering.



FC-BT with dynamic ordering



- Which **country** should we colour next
- What **colour** should we pick for it?

E most-constrained variable (smallest domain)

RED least-constraining value (eliminates fewest values from neighbouring domains)

Reading Material

- **Artificial Intelligence,** A Modern Approach

  Stuart J. Russell and Peter Norvig

  - **Chapter 6.**