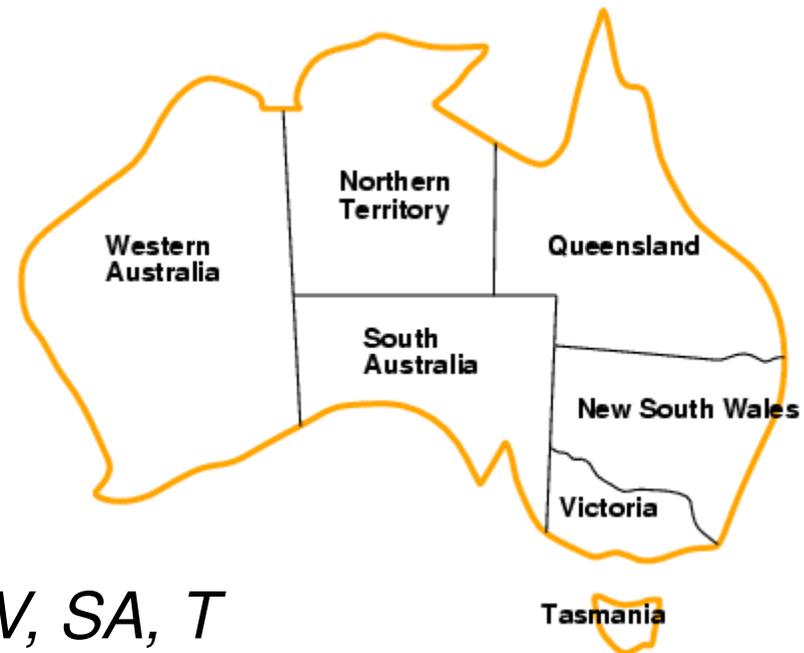


# Esempio: Map-Coloring

---



- **variabili**  $WA, NT, Q, NSW, V, SA, T$
- **Domini**  $D_i = \{\text{red, green, blue}\}$
- **vincoli**: regioni adiacenti devono avere colori diversi
- e.g.,  $WA \neq NT$ , or  $(WA, NT)$  in  $\{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), (\text{blue, red}), (\text{blue, green})\}$

# Esempio: Map-Coloring

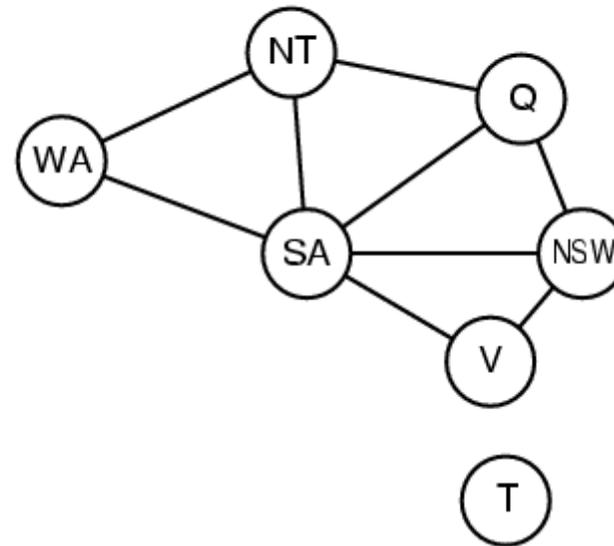
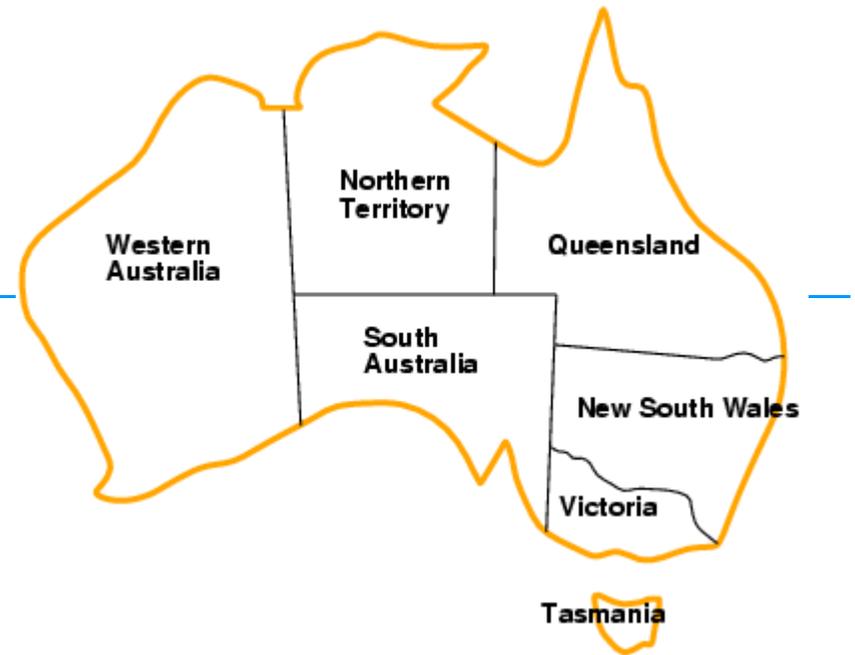
---



- Le soluzioni sono assegnamenti **completi** e **consistenti**, e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

# Constraint graph

- **CSP binario:** ogni vincolo si correla a due variabili
- **Constraint graph:** nodi sono variabili, archi sono vincoli



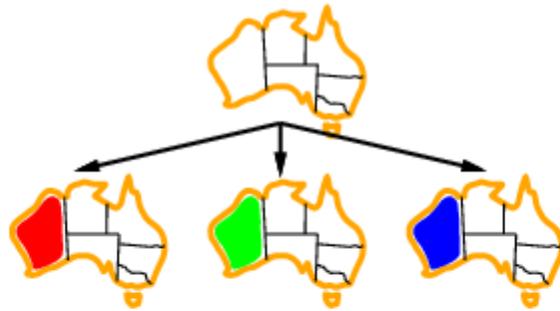
# Backtracking : esempio

---



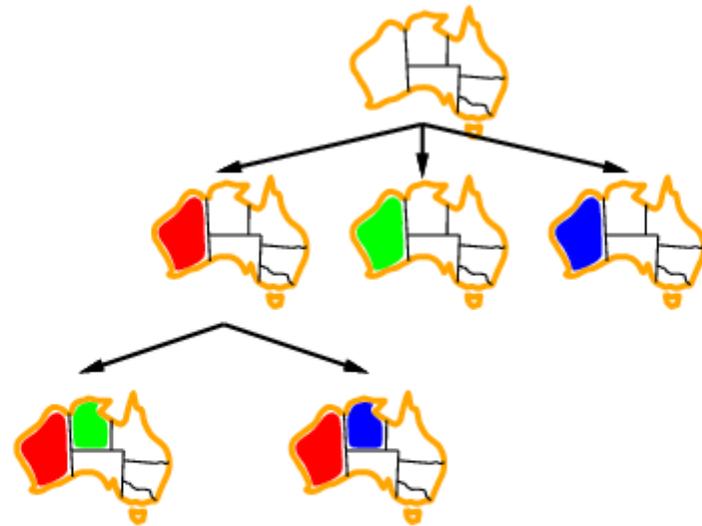
# Backtracking : esempio

---



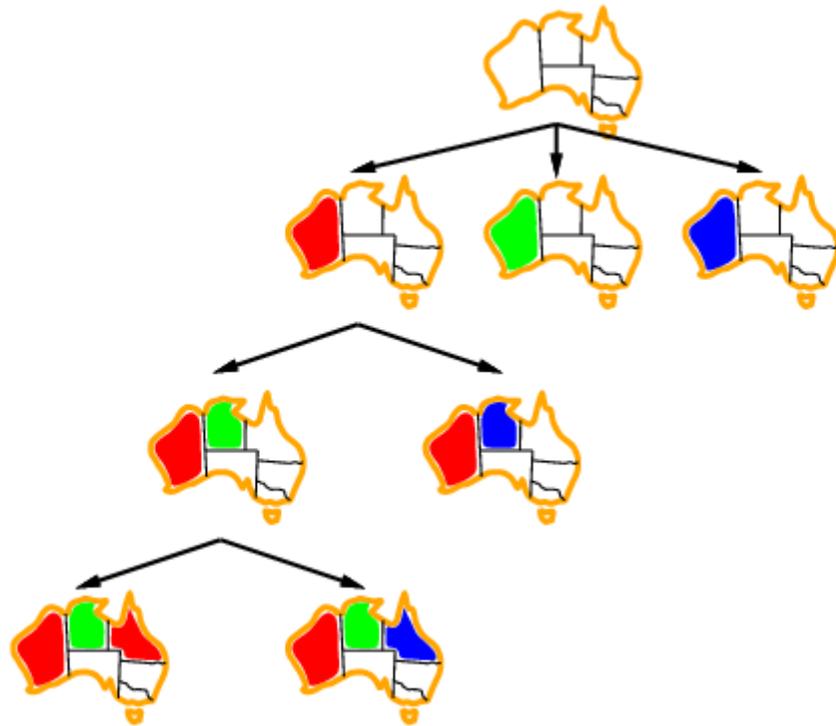
# Backtracking : esempio

---



# Backtracking : esempio

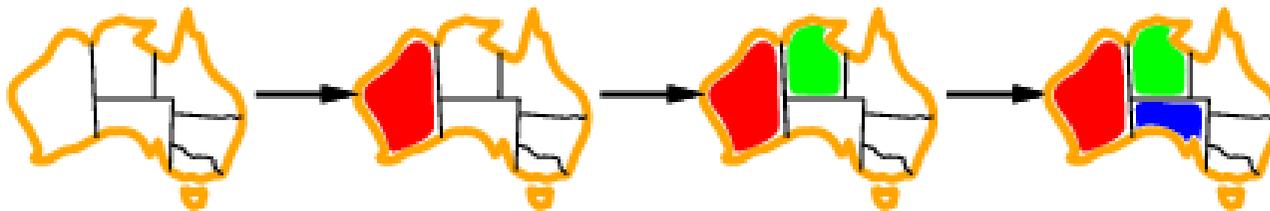
---



# Most constrained variable

---

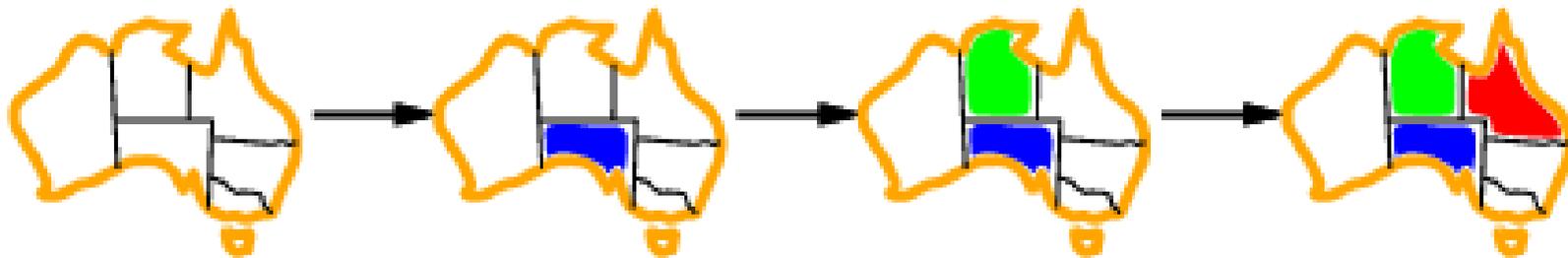
- Most constrained variable:  
Scegli la variabile con meno valori ammessi
- minimum remaining values (MRV) o first fail



# Most constraining variable

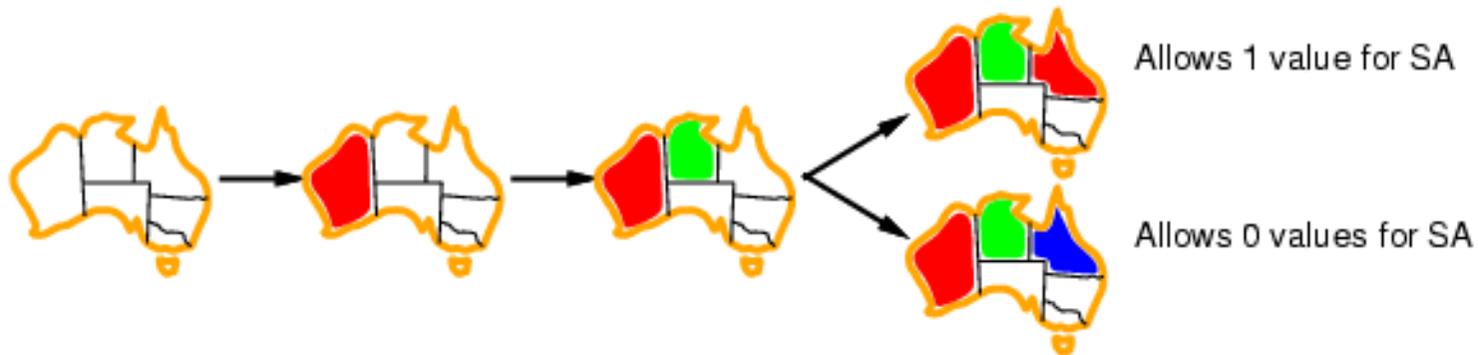
---

- Most constraining variable:
  - Scegli la variabile con il maggior numero di vincoli sulle altre variabili rimaste (si applica quando ci sono piu' variabili con uguale numero di valori nel dominio).



# Least constraining value

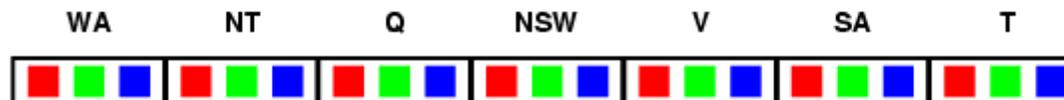
- Data una variabile, scegli il valore meno vincolante, cioè che rende impossibili o inconsistenti meno assegnamenti delle variabili rimaste.



# Forward checking

---

- Idea:
  - Tenere traccia dei valori legali per le variabili rimanenti
  - Fallire quando non ci sono piu' valori legali.







# Forward checking



WA	NT	Q	NSW	V	SA	T
■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■
■■■■	■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■
■■■■	■ ■ ■	■■■■	■ ■ ■	■ ■ ■	■ ■ ■	■ ■ ■
■■■■	■ ■ ■	■■■■	■ ■ ■	■■■■	■ ■ ■	■ ■ ■

# Constraint Propagazione di vincoli e forward checking

- Forward checking propaga l'informazione da variabili assegnate a variabili non assegnate, ma non consente di individuare subito situazioni inconsistenti.
- NT e SA non possono essere entrambe blue
- **Constraint propagation** fra variabili non assegnate!



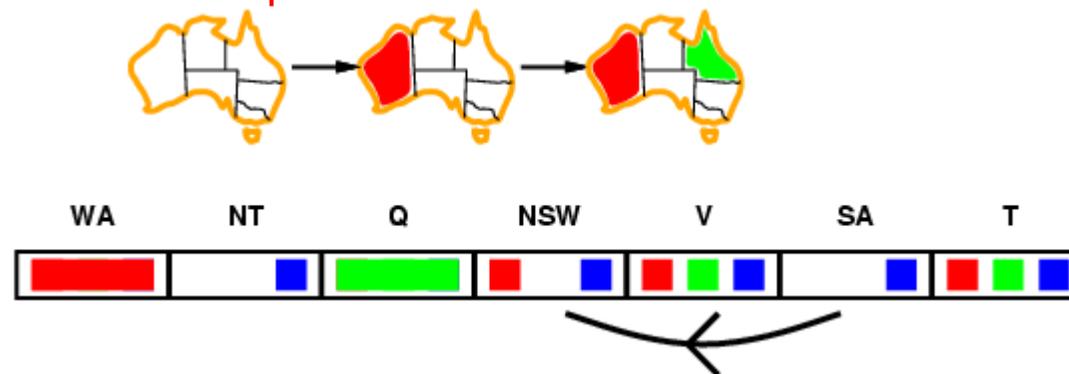
WA	NT	Q	NSW	V	SA	T
Red	Green	Blue	Red	Green	Blue	Red
Red	Green	Blue	Red	Green	Blue	Red
Red	Blue	Green	Red	Blue	Green	Red

# Arc consistency

- La piu' semplice forma di propagazione, rende ogni arco consistente.

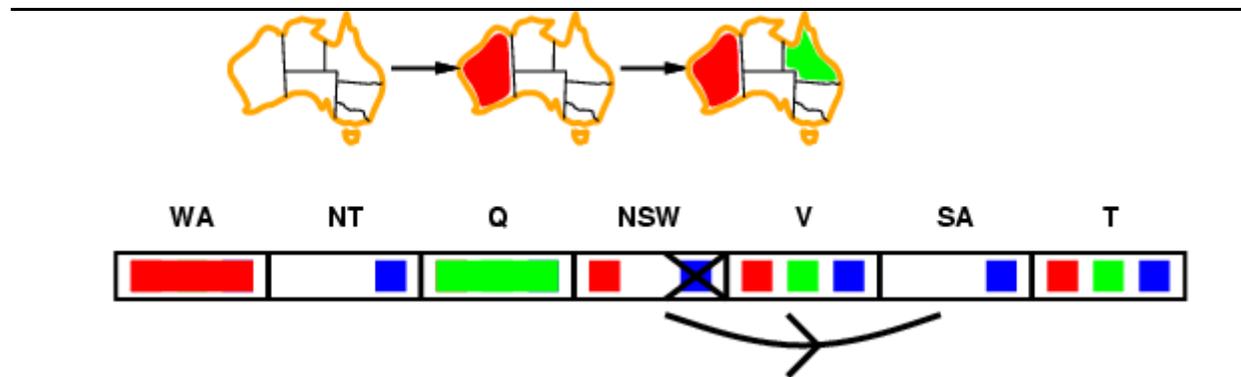
- $X \rightarrow Y$  e' consistente se e solo se

Per **ogni** valore di  $X$  c'e' almeno un **valore ammissibile** per  $Y$



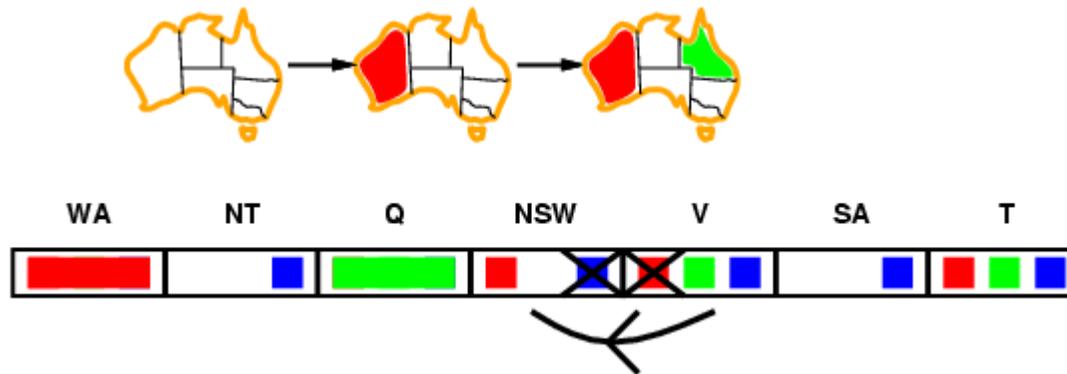
# Arc consistency

---



# Arc consistency

- SE  $X$  perde un valore, devo ricontrollare I “vicini” di  $X$ .



# Arc consistency

I fallimenti sono trovati con l'Arc consistency prima che con il forward checking

Puo' girare come pre-processor, oppure dopo ogni assegnamento.



# Arc-Consistency: Algorithm

---

```
function AC-3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables  $\{X_1, X_2, \dots, X_n\}$ 
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
     $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\textit{queue})$ 
    if RM-INCONSISTENT-VALUES( $X_i, X_j$ ) then
      for each  $X_k$  in NEIGHBORS[ $X_i$ ] do
        add  $(X_k, X_i)$  to queue



---


function RM-INCONSISTENT-VALUES( $X_i, X_j$ ) returns true iff remove a value
  removed  $\leftarrow$  false
  for each  $x$  in DOMAIN[ $X_i$ ] do
    if no value  $y$  in DOMAIN[ $X_j$ ] allows  $(x, y)$  to satisfy constraint( $X_i, X_j$ )
      then delete  $x$  from DOMAIN[ $X_i$ ]; removed  $\leftarrow$  true
  return removed
```