

CSP come ricerca nello spazio degli stati

- CSP:
 - **state** e' definito da **variabili** X_i con **valori** presi dai **domini** D_i
 - **goal test** e' un insieme di **vincoli** che specificano le combinazioni di valori permesse (in modo intensionale).
 - **Operatori** possono essere (labeling) assegnamenti di valori a variabili
- Linguaggio di formalizzazione generale
- Algoritmi **general-purpose** per la soluzione.

1

CSP come ricerca (con propagazione)

- **Stato iniziale**: assegnamento vuoto { }
 - **Funzione Successore**: assegna un valore ad una variabile non ancora legata in modo che sia legale con gli assegnamenti gia' fatti.
 - fallisci se non esiste
 - **Goal test**: l'assegnamento e' completo (tutte le variabili sono legate).
1. Schema identico per tutti i CSPs
 2. Profondita limitata n se n sono le variabili.
 - usa depth-first search
 3. La strada e' irrilevante.
 4. Problema commutativo con d^n foglie.

2

Standard Backtracking

- Depth-first search per CSPs con singolo assegnamento di variabili e chiamata standard **backtracking**
- Standard Backtracking search e' l'algoritmo di ricerca non-informata basilare per CSP.

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return RECURSIVE-BACKTRACKING({}, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(Variables[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment according to Constraints[csp] then
      add { var = value } to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
    remove { var = value } from assignment
  return failure
```

3

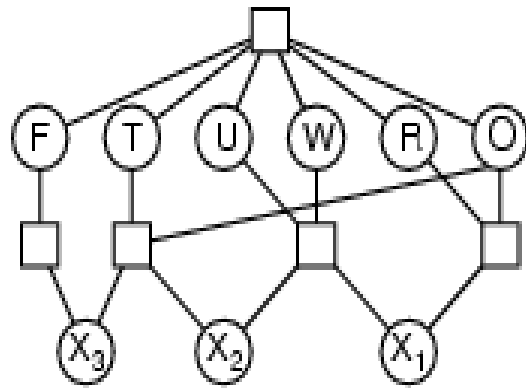
Differenti CSP

- Variabili Discrete
 - Domini Finiti:
 - n variables, con dimensione d (quelli che vedremo)
 - Domini Infiniti:
 - Interi, stringhe ecc,
 - e.g., job- scheduling, variabili rappresentano giorni di inizio-fine per ogni lavoro
 - *vincoli di durata* $StartJob_1 + 5 \leq StartJob_3$
- Variabili Continue
 - Programmazione lineare (Ricerca Operativa)

4

Esempio: Criptoaritmetica

$$\begin{array}{r} \text{TWO} \\ + \text{TWO} \\ \hline \text{FOUR} \end{array}$$



- **Variables:** $F, T, U, W, R, O, X_1, X_2, X_3$
- **Domains:** $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- **Constraints:** *Alldiff* (F, T, U, W, R, O)
 - $O + O = R + 10 \cdot X_1$
 - $X_1 + W + W = U + 10 \cdot X_2$
 - $X_2 + T + T = O + 10 \cdot X_3$
 - $X_3 = F, T \neq 0, F \neq 0$

5

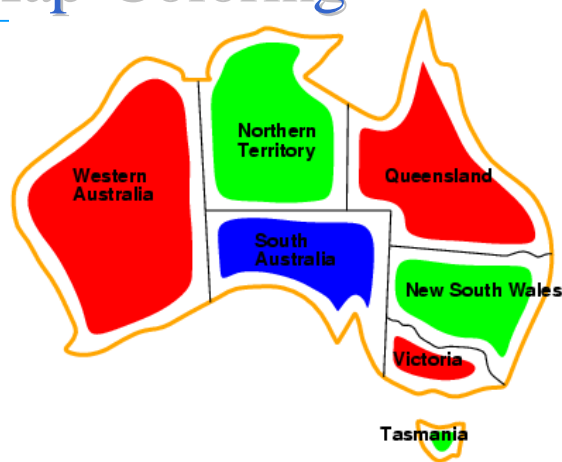
Esempio: Map-Coloring



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

6

Esempio: Map-Coloring

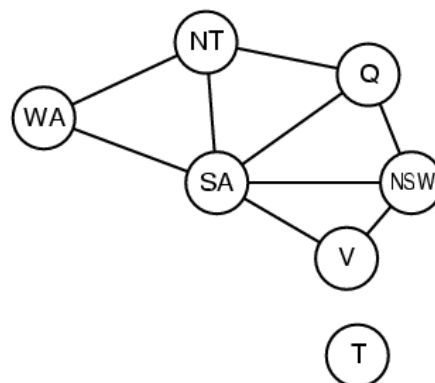


- Le soluzioni sono assegnamenti **completi** e **consistenti**, e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

7

Constraint graph

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



8

Backtracking : esempio



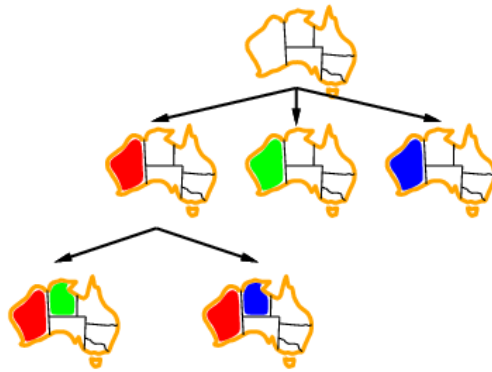
9

Backtracking : esempio



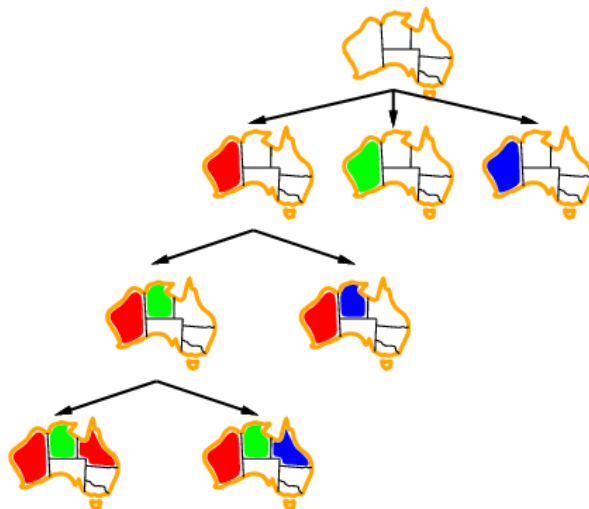
10

Backtracking : esempio



11

Backtracking : esempio



12

Most constrained variable

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



13

Most constraining variable

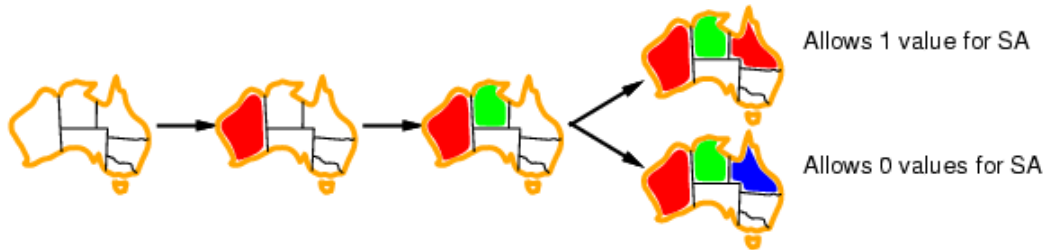
- Most constraining variable:
 - Scegli la variabile con il maggior numero di vincoli sulle altre variabili rimaste (si applica quando ci sono piu' variabili con uguale numero di valori nel dominio).



14

Least constraining value

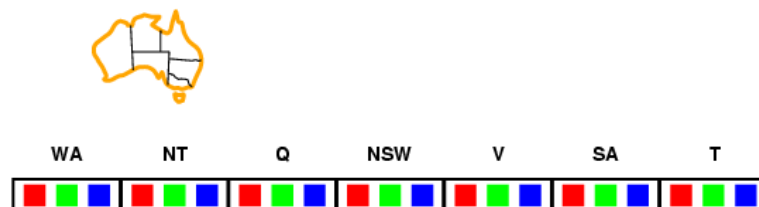
- Data una variabile, scegli il valore meno vincolante, cioè che rende impossibili o inconsistenti meno assegnamenti delle variabili rimaste.



15

Forward checking

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



16

Forward checking



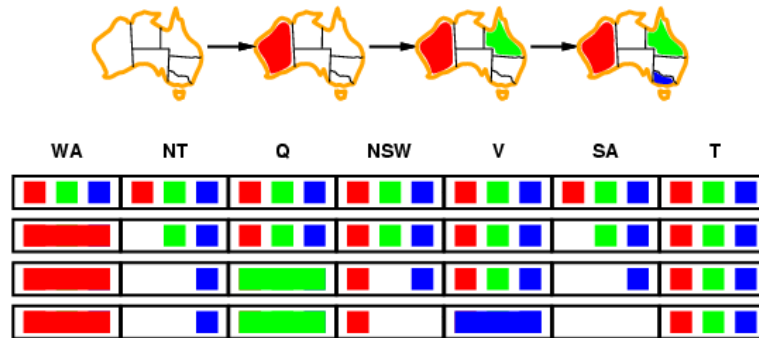
WA	NT	Q	NSW	V	SA	T
Red	Green	Blue	Red	Green	Blue	Red
Red	Green	Blue	Red	Green	Blue	Red
Red	Green	Blue	Red	Green	Blue	Red

Forward checking



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

Forward checking



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!

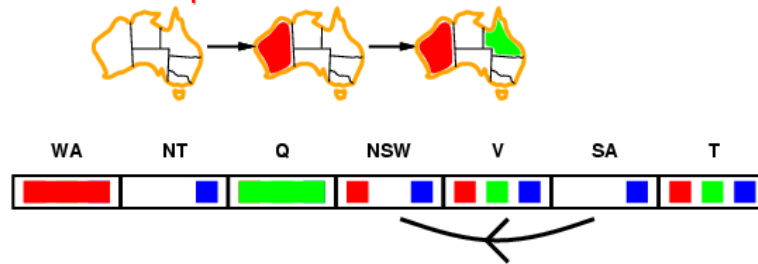


Arc consistency

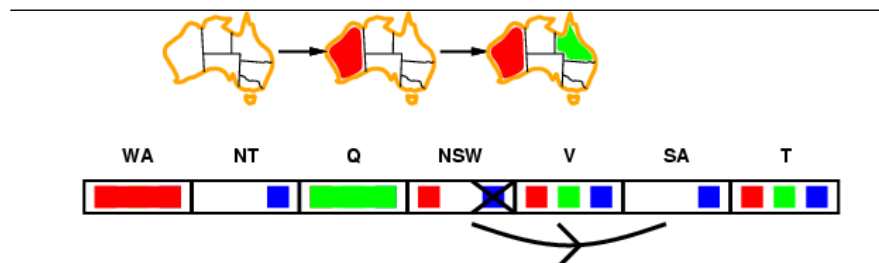


- La piu' semplice forma di propagazione, rende ogni arco consistente.
- $X \rightarrow Y$ e' consistente se e solo se

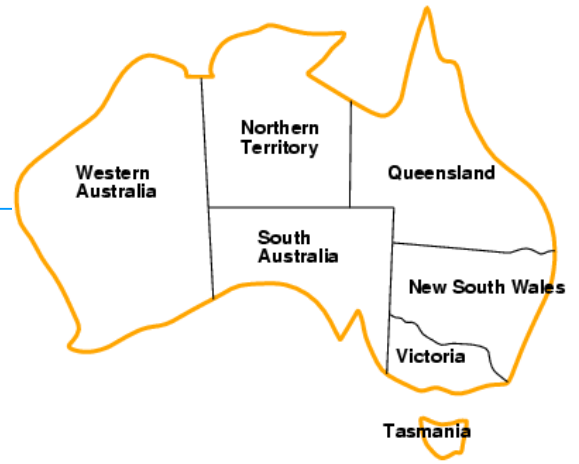
Per ogni valore di X c'e' almeno un valore ammissibile per Y



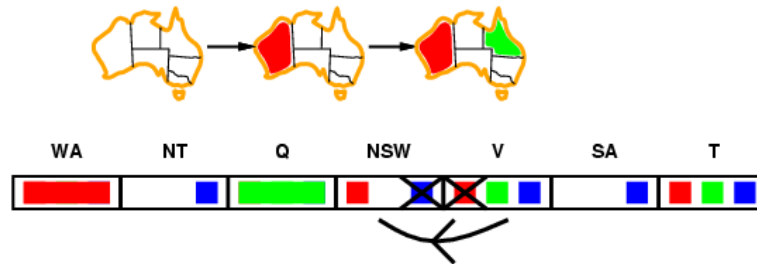
Arc consistency



Arc consistency



- SE X perde un valore, devo ricontrollare I "vicini" di X.



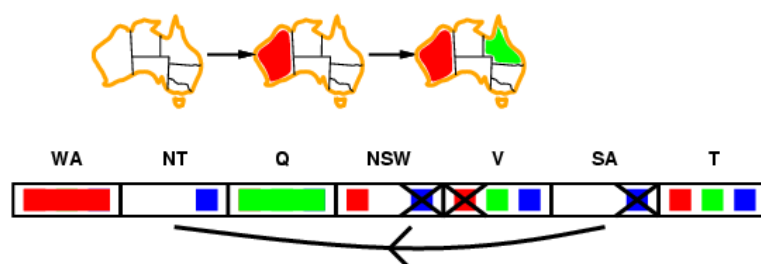
23

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.



24

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
```