UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Intelligenza Artificiale

Problemi di soddisfacimento di vincoli

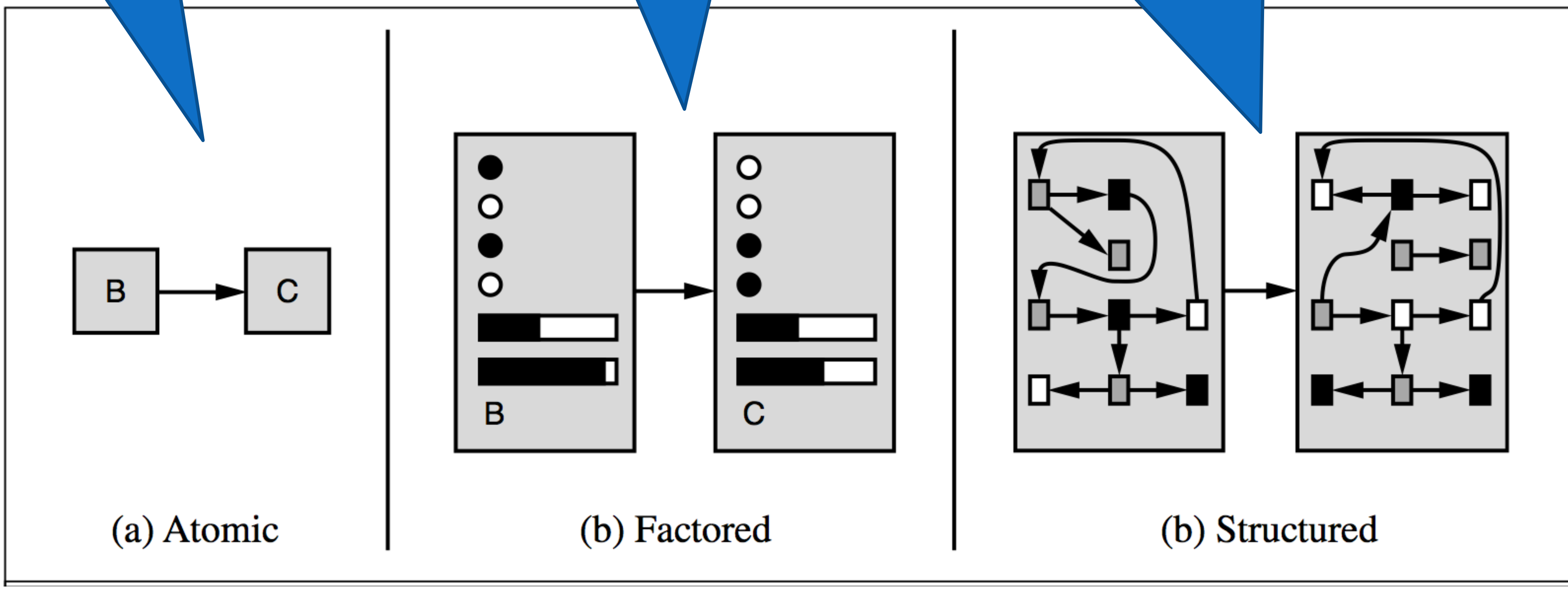
a.a. 2022/2023

Rappresentare gli stati

Usato da:
Algoritmi di ricerca
HMM

Usato da algoritmi di:
CSP
Pianificazione
Reti bayesiane

Usato da:
logica del primo ordine
Modelli probabilistici
Apprendimento basato sulla conoscenza
NLP



(a) Atomic (b) Factored (b) Structured

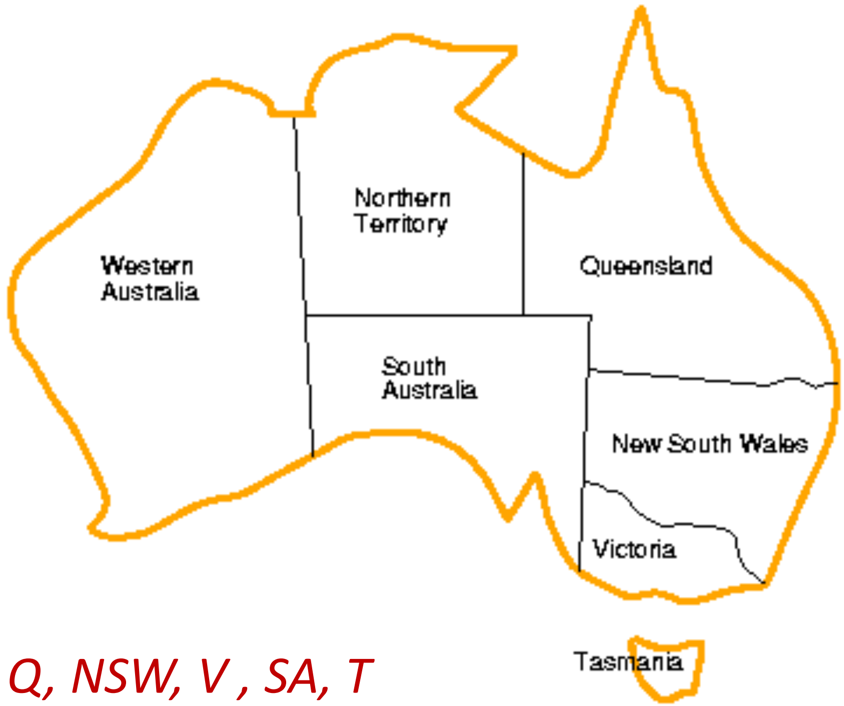
Applicazioni

- Problemi di allocazione di risorse
  - Distribuzione degli aerei negli stand negli aeroporti
  - Distribuzione dei container nei porti e nei cargo
  - Pianificazione delle attività del personale
    - Gestione dei turni
    - Gestione dei periodi di carico
- Problemi di scheduling
  - Pianificazione della produzione
  - Produzione della tabella oraria di treni e trasporti urbani
- Gestione e configurazione di reti di telecomunicazione
  - Pianificazione della stesura dei cavi
  - Posizionamento degli access point in reti WiFi
- Progettazione di circuiti digitali
- Realizzazione di assistenti intelligenti per applicazioni Web

Formulazione di problemi CSP

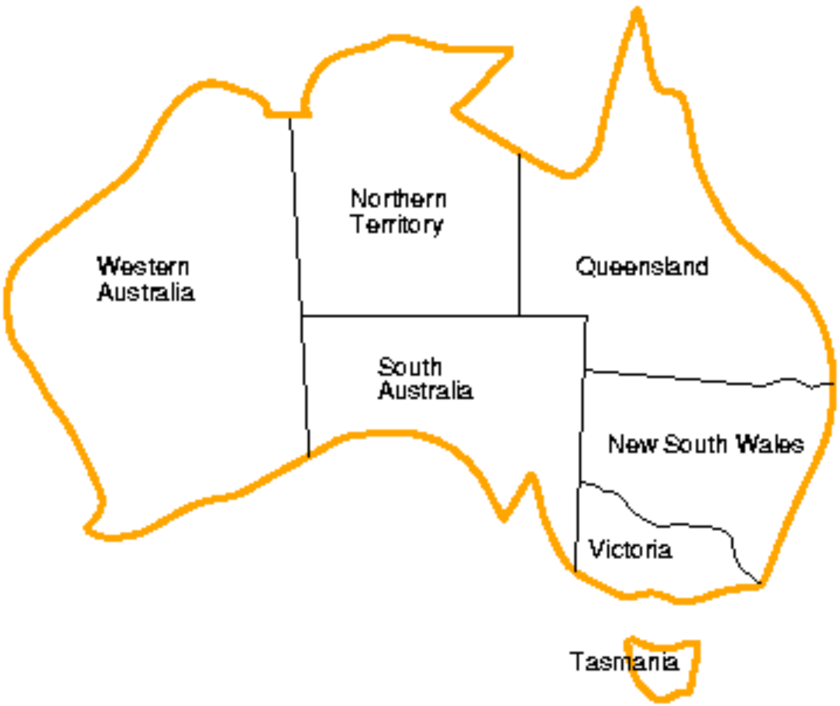
- *Problema*:
  - insieme di variabili: $X_1\ X_2 \ldots X_n$
  - con associato dominio: $D_1\ D_2 \ldots D_n$
  - insieme di vincoli (relazioni tra le variabili): $C_1\ C_2 \ldots C_m$
- Un *modello* di un CSP è un assegnamento di valori a variabili che soddisfa tutti i vincoli (ad es. $X_1 < X_2$)
- *Stato:* un assegnamento parziale di valori a variabili $\{X_i = v_i,\ X_j = v_j \ldots\}$
- *Stato iniziale*: { }
- *Azioni*: assegnamento di un valore a una variabile
- *Soluzione* *(goal test)*: un assegnamento completo (le variabili hanno tutte un valore) e consistente (i vincoli sono soddisfatti)

Colorazione di una mappa

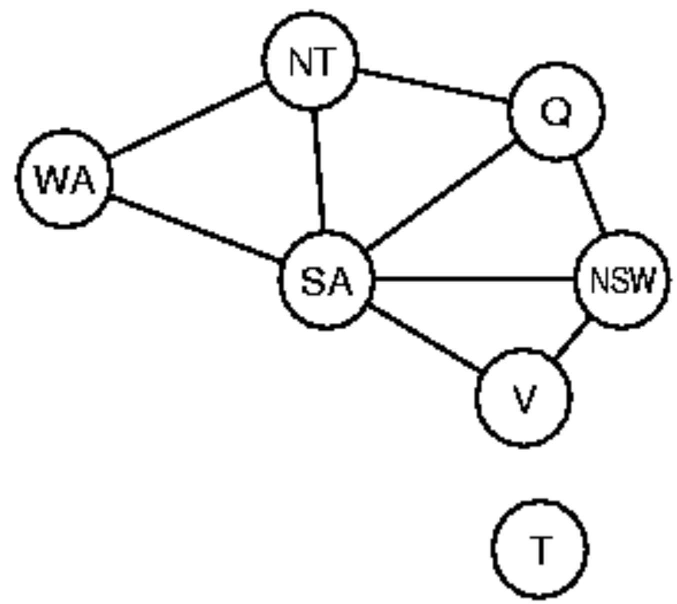


- Variabili *WA, NT, Q, NSW, V , SA, T*
- Domini D_i = *{red, green, blue}*
- Vincoli: regioni adiacenti devono avere colori differenti cioè,
  - WA ≠ NT (se il linguaggio lo permette), oppure
  - (WA, NT) ∈ {(red, green), (red, blue), (green, red), (green, blue), . . .}

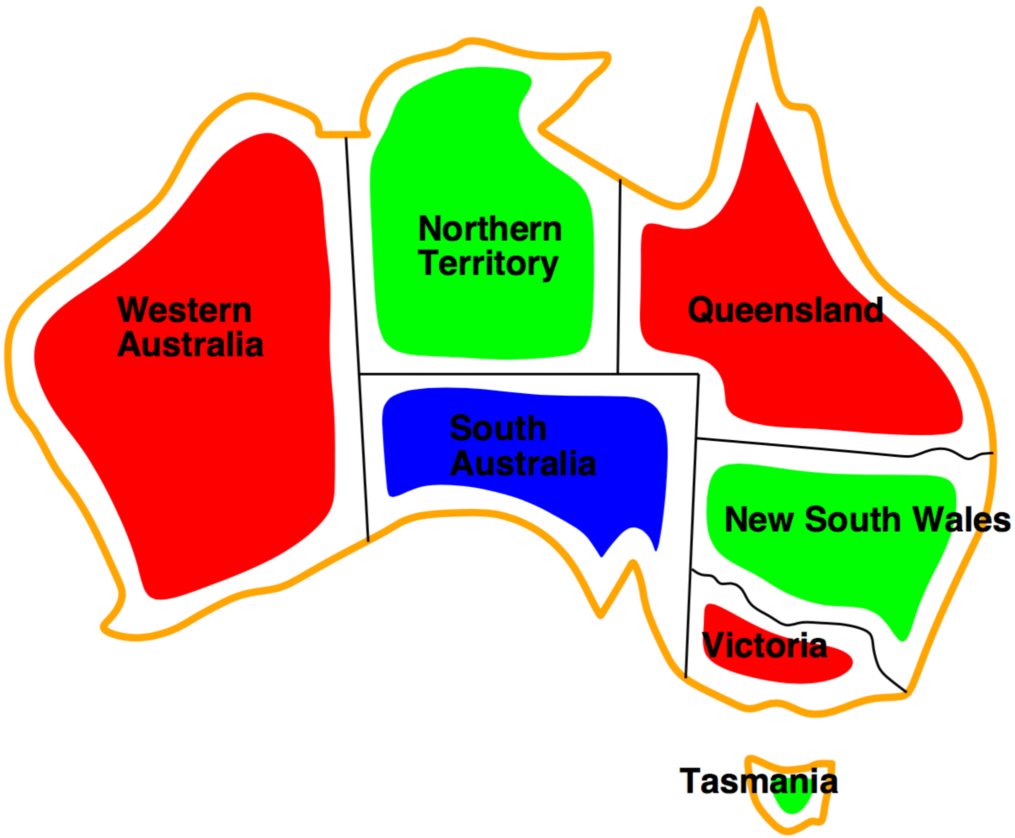
Esempio: colorazione di una mappa



Grafo dei vincoli



Soluzione



- Le soluzioni sono assegnazioni che soddisfano tutti i vincoli, cioè, *{WA=red, NT =green, Q=red, NSW =green, V =red, SA=blue, T =green}*

Tipi di problemi CSP

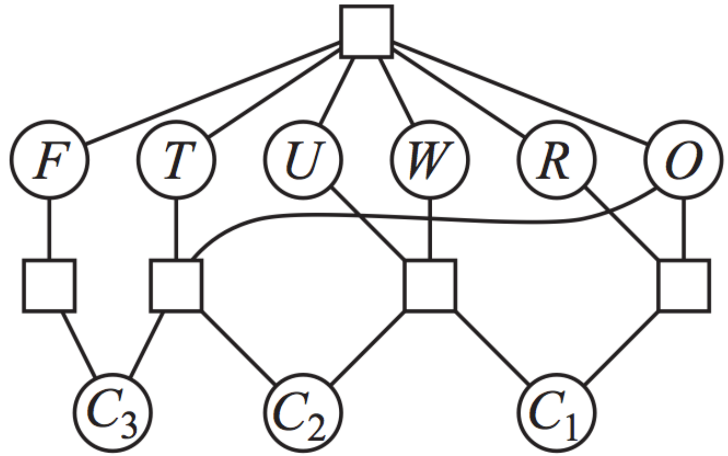
- Variabili discrete con domini finiti o infiniti
  - CSP booleani (valori *vero* e *falso*)
- Variabili con domini continui (programmazione lineare)
  - Vincoli espressi come uguaglianze o disuguaglianze lineari

- I vincoli possono essere:
  - unari (es. “x pari”)
  - binari (es. “x > y”)
  - di grado superiore (es. x+y = z)

- Vincoli assoluti o di preferenza

Giochi Enigmistici

$$\begin{array}{r} T\;W\;O \\ +\;T\;W\;O \\ \hline F\;O\;U\;R \end{array}$$



- Ogni lettera = cifra diversa
- *Tuttediverse*(F, T, U, W, R, O)
- $O + O = R + 10\ C_1$
- $C_1 + W + W = U + 10C_2$
- $C_2 + T + T = O + 10C_3$
- $C_3 = F$
- Rappresentato in un ipergrafo di vincoli - nodi e ipernodi (quadrati) che rappresentano vincoli n-ari

Giochi Enigmistici

Solution State:-

Y = 2
D = 7
S = 9
R = 8
N = 6
E = 5
O = 0
M = 1
C1 = 1
C2 = 0
C3 = 0

	C3(0)	C2(1)	C1(1)	
	S(9)	E(5)	N(6)	D(7)
+	M(1)	O(0)	R(8)	E(5)
M(1)	O(0)	N(6)	E(5)	Y(2)

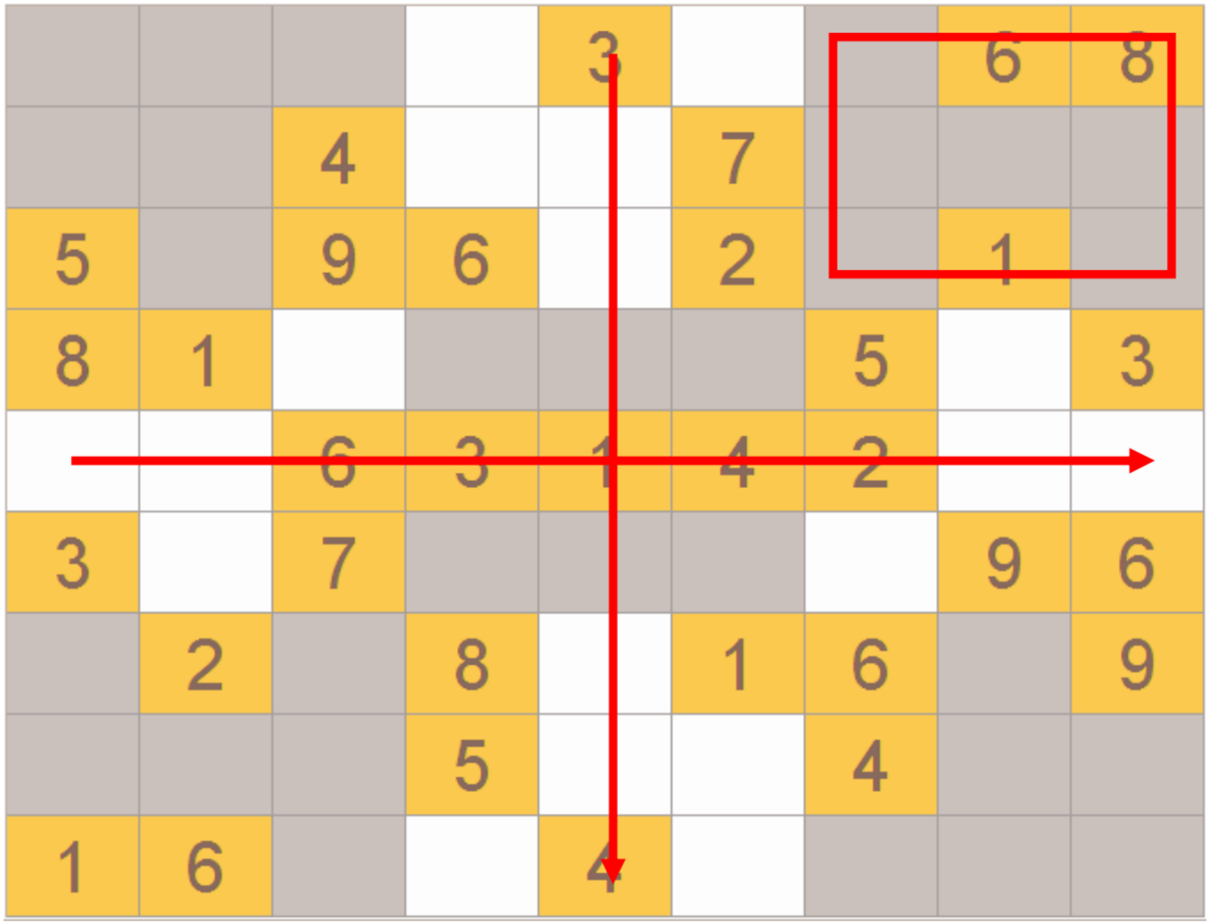
Sudoku

- Le celle possono assumere un valore numerico
- I valori numerici possibili per le celle sono gli interi compresi tra 1 e 9 (estremi inclusi)

				3			6	8
		4			7			
5		9	6		2		1	
8	1					5		3
		6	3	1	4	2		
3		7					9	6
	2		8		1	6		9
			5			4		
1	6			4				

Sudoku - regole

- Le regole sono semplici
- I valori nelle celle devono essere tutti diversi tra loro
  - Per ogni riga
  - Per ogni colonna
  - Per ogni blocco (3x3)



Definizione del problema come CSP

- Un problema di Soddisfacimento di Vincoli è composto da
  - Un insieme di variabili
    - Le celle del Sudoku, i simboli dei problemi Cripto-aritmetici
  - Un insieme di domini da cui attingere i valori da assegnare alle singole variabili
    - [1..9] per il Sudoku, [0..9] per il problemi Cripto-aritmetici
  - Un insieme di vincoli
    - Le regole del Sudoku o dei problemi Cripto-aritmetici

Soluzione di un CSP

- Una soluzione di un CSP è

  Un insieme di valori presi dai domini che assegnati alle rispettive variabili soddisfano tutti i vincoli

- Un CSP può avere
  - Una soluzione
    - Il Sudoku è ben posto
  - Zero soluzioni
    - Il Sudoku è sbagliato...c'è stato un errore di stampa
  - Più soluzioni
    - Il Sudoku non è ben posto...non dovrebbe succedere

Risolvere problemi di CSP

- Algoritmi di propagazione di vincoli (inferenza)
  - Utilizzano i vincoli per ridurre il numero di valori legali per una variabile, che a sua volta può ridurre i valori legali per un'altra variabile e cosi via.
  - Algoritmo AC-3
- Algoritmi di ricerca
  - Operano su assegnamenti di valori alle variabili
  - Backtracking
- Alternanza di ricerca ed inferenza
  - Backtracking con forward checking
  - Backtracking con MAC
- Algoritmi di ricerca locale

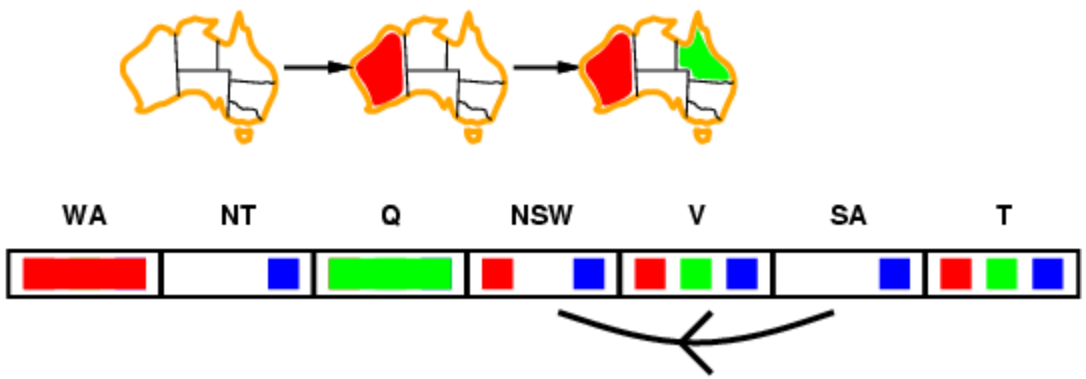
Inferenza nei CSP: consistenza di nodo

- Una singola variabile è nodo-consistente se tutti i valori del suo dominio soddisfano i suoi vincoli unari

- Ad esempio, se gli australiani del sud non amano il verde, possiamo rendere il nodo SA consistente eliminando il verde dal suo dominio

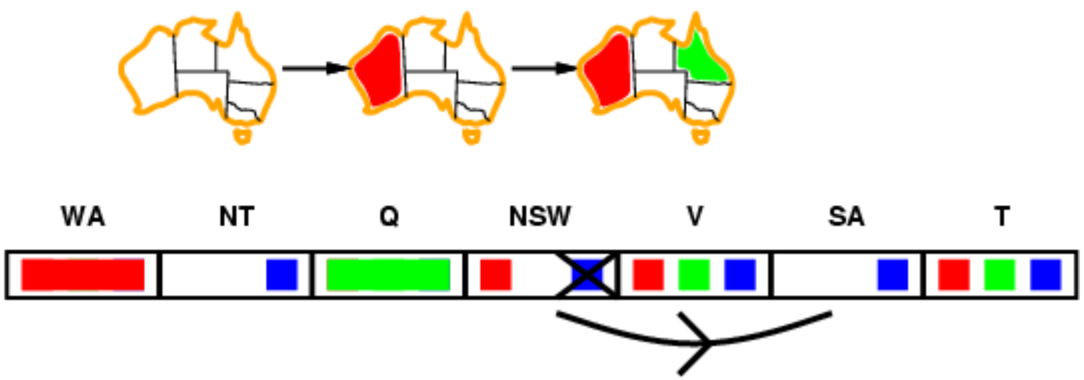
Inferenza nei CSP: propagazione dei vincoli

- La forma più semplice di propagazione rende ogni arco consistente
- L'arco X → Y è consistente se e solo se per ogni valore x di X c'è qualche y consentito



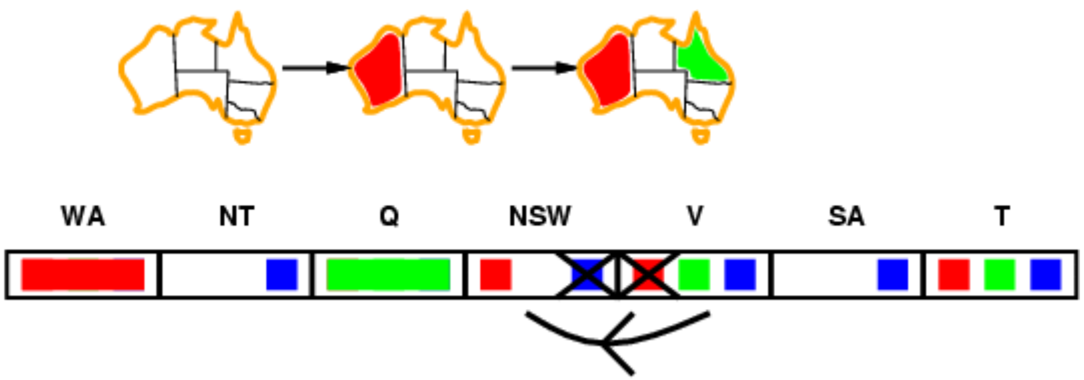
Inferenza nei CSP: propagazione dei vincoli

- La forma più semplice di propagazione rende ogni arco consistente
- L'arco X → Y è consistente se e solo se per ogni valore x di X c'è qualche y consentito



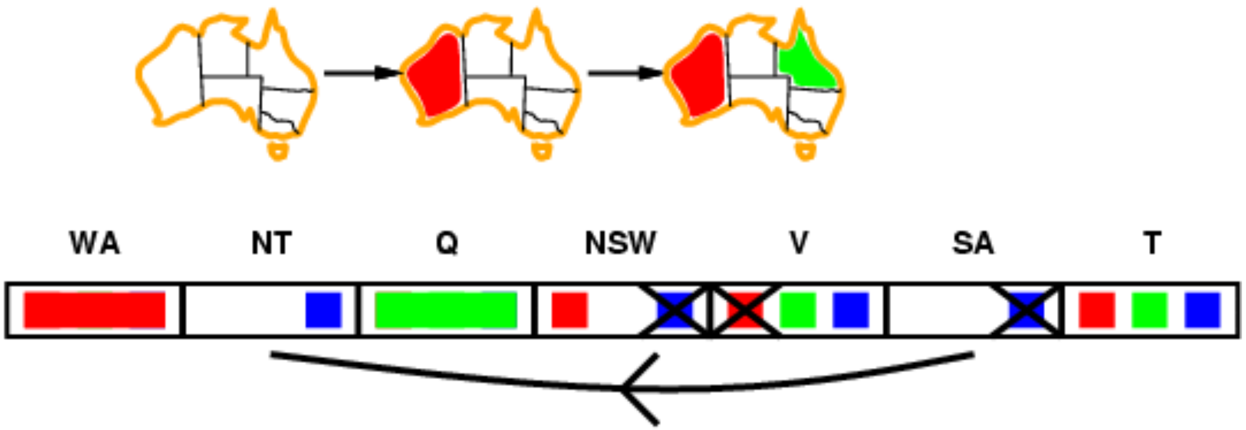
Inferenza nei CSP: constraint propagation

- La forma più semplice di propagazione rende ogni arco consistente
- X → Y è consistente se e solo se per ogni valore x di X c'è qualche y consentito



- Se X perde un valore, i vicini di X devono essere ricontrollati

Inferenza nei CSP: constraint propagation



- Se X perde un valore, i vicini di X devono essere ricontrollati
- Può essere eseguito dopo ogni assegnazione

Arc consistency algorithm AC-3

```
function AC-3( csp) returns the CSP, possibly with reduced domains
   inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
   local variables: queue, a queue of arcs, initially all the arcs in csp

   while queue is not empty do
      (Xi, Xj) ← REMOVE-FIRST(queue)
      if RM-INCONSISTENT-VALUES(Xi, Xj) then
         for each Xk in NEIGHBORS[Xi] do
            add (Xk, Xi) to queue
```

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

- Time complexity: $O(cd^3)$
  - Numero max di volte che un arco entra in queue è *d*
  - Controllo consistenza arco $O(d^2)$

Cercare una soluzione: ricerca

- La tecnica di soluzione più semplice è detta Generate & Test (G&T)
  - Si assegna un valore ad ogni variabile
  - Si verifica se tutti i vincoli sono soddisfatti
    - Sì, è stata trovata una soluzione
    - No, si prova con valori diversi
- Il procedimento continua finché non ci sono più assegnamenti nuovi da provare
  - Nel frattempo si è passati per (tutte) le soluzioni

Applicazione al sudoku

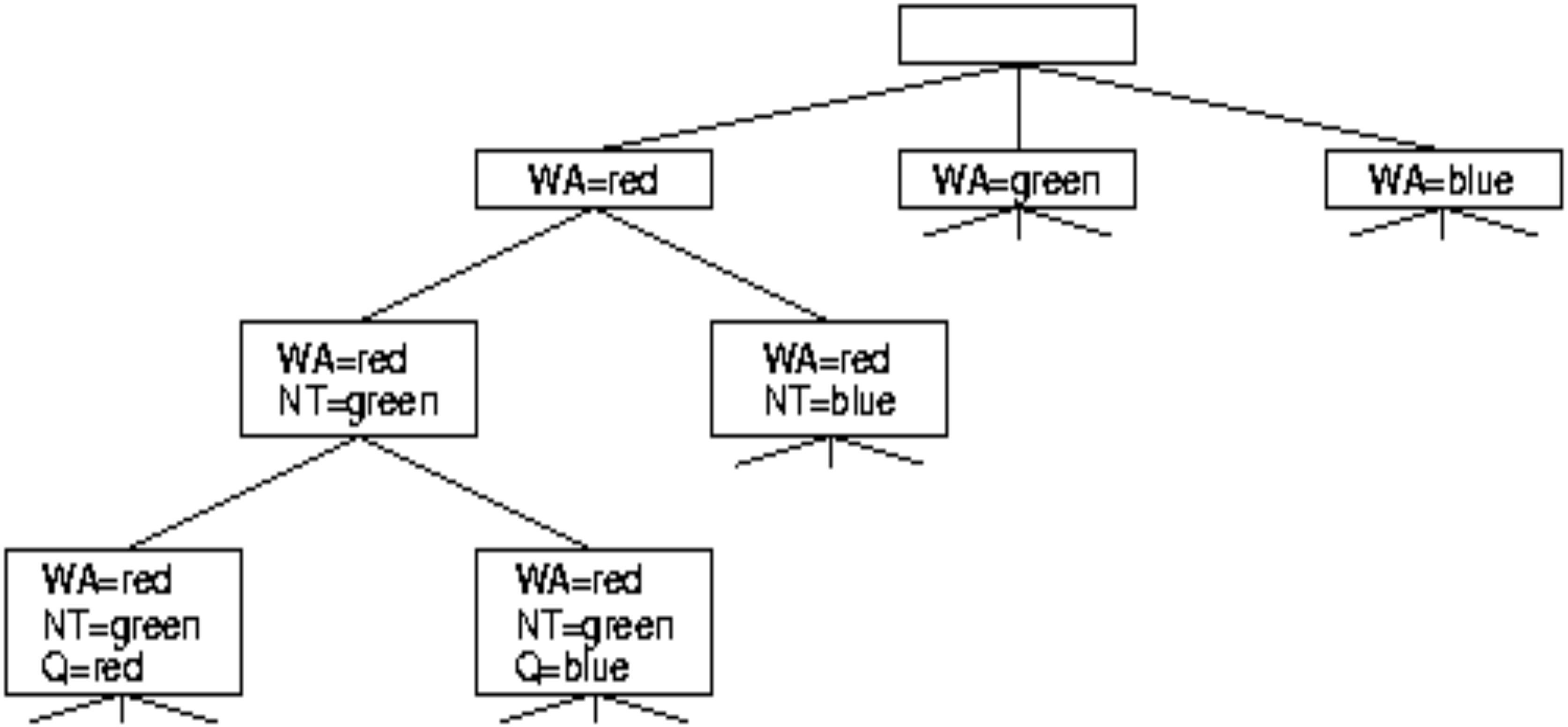
1. Si mette un numero in ogni cella vuota
2. Si verifica che le regole siano rispettate
3. Se qualche regola è violata, si prova con altri numeri

- Se abbiamo n variabili ed un dominio con d valori le foglie dell'albero sono d^n
- Il G&T non è realistico per problemi complessi
  - Nel caso del Sudoku, con 31 celle già riempite, abbiamo $9^{50} \approx 5 \cdot 10^{47}$ possibilità da esplorare!

Strategie di ricerca

- *Ricerca con backtracking* (BT): ad ogni passo si assegna una variabile e si torna indietro in caso di fallimento (*Generate and Test*).
- *Controllo anticipato* della violazione dei vincoli: è inutile andare avanti fino alla fine e poi controllare; si può fare *backtracking* non appena si scopre un vincolo violato.
- La ricerca è naturalmente limitata in profondità dal numero di variabili

Esempio di backtracking



Euristiche per problemi CSP *indipendenti dal dominio*

1. Quale variabile scegliere?
2. Quali valori scegliere?
3. Qual è l'influenza di una scelta sulle altre variabili? (*Propagazione di vincoli*)
4. Come evitare di ripetere i fallimenti? (*Backtracking intelligente*)

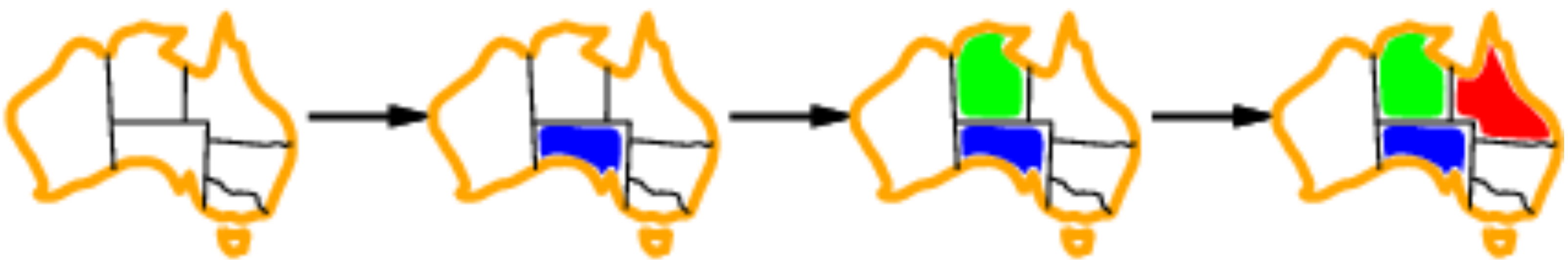
Scelta delle variabili

1. MRV (Minimum Remaining Values): scegliere la variabile che ha meno valori possibili, la variabile *più vincolata*.

   Si scoprono prima i fallimenti.

2. In base *al grado (DH – Degree Heuristic):* scegliere la variabile coinvolta in più vincoli con le altre variabili (la *variabile più vincolante*)

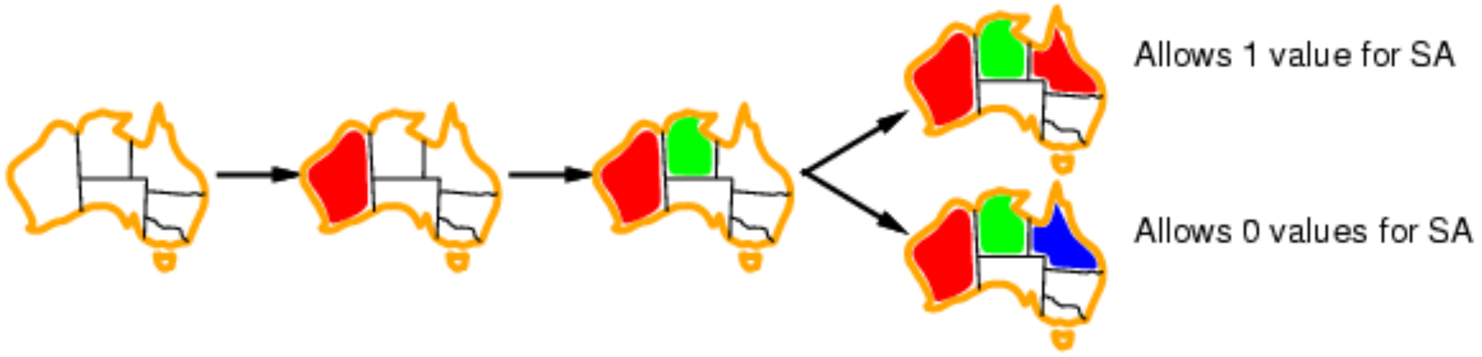
   Da usare a parità di MRV



Scelta dei valori

- Scelta la variabile come scegliere il valore da assegnare?

1. *Valore meno vincolante* (LCV - Least constraining value): quello che esclude meno valori per le altre variabili direttamente collegate con la variabile scelta
   - Cerca sempre di lasciare la massima flessibilità ai successivi assegnamenti di variabili



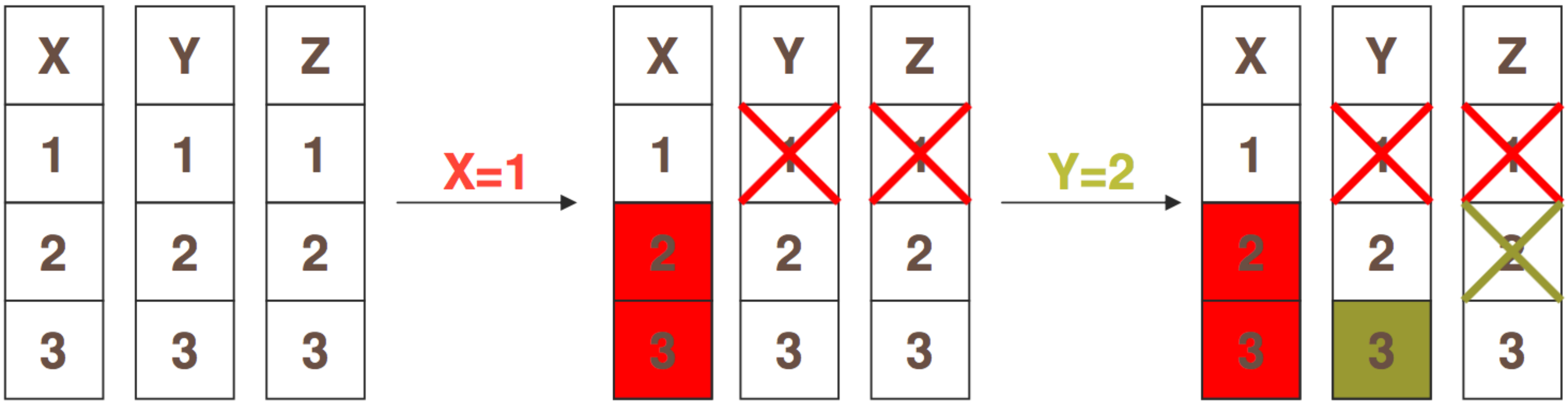
Propagazione di vincoli

1. Verifica in avanti (*forward checking* o FC): assegnato un valore ad una variabile si possono eliminare i valori incompatibili per le altre variabili collegate da vincoli (un passo solo)

2. *Propagazione di vincoli:* si itera il *forward checking*; se una variabile ha il suo dominio ristretto per effetto del *forward checking* si vanno a controllare le variabili collegate …

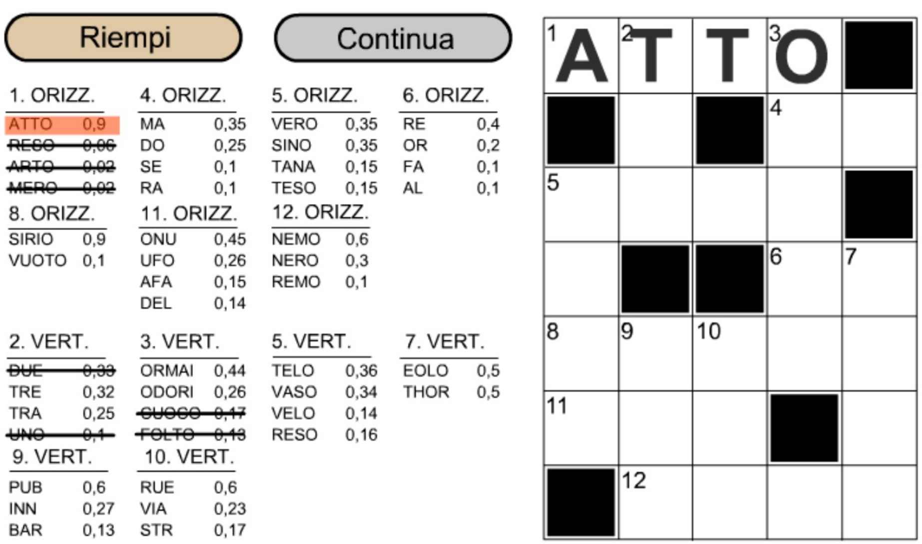
Propagazione dei Vincoli

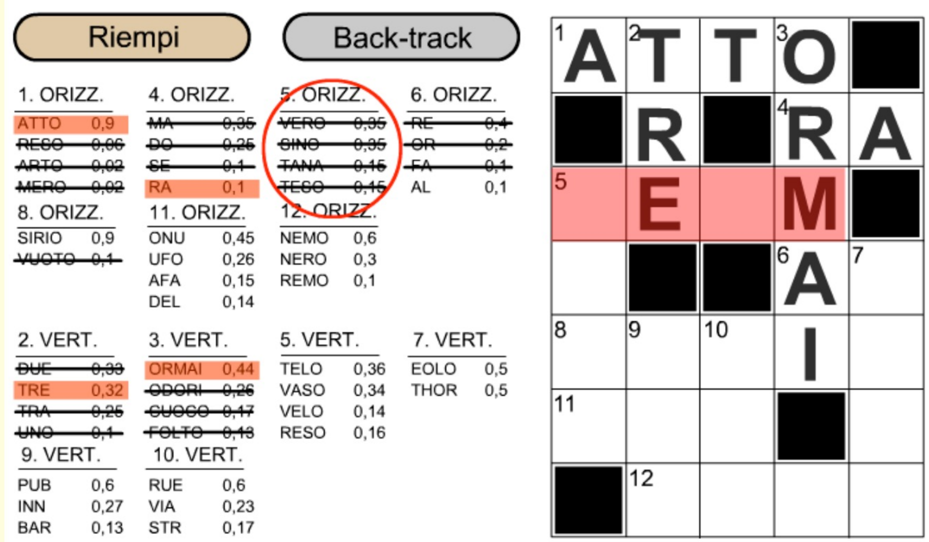
- In generale una volta assegnato un valore ad una variabile, i vincoli eliminano dei valori possibili dai domini delle altre variabili



Forward checking

- E’ applicabile all’interno dell’algoritmo di backtracking come tecnica per stabilire quando tornare indietro.
- Ogni volta che si raggiunge uno stato non-consistente (con almeno una variabile priva di valori rimasti -> *node-consistency*) si effettua il backtracking.



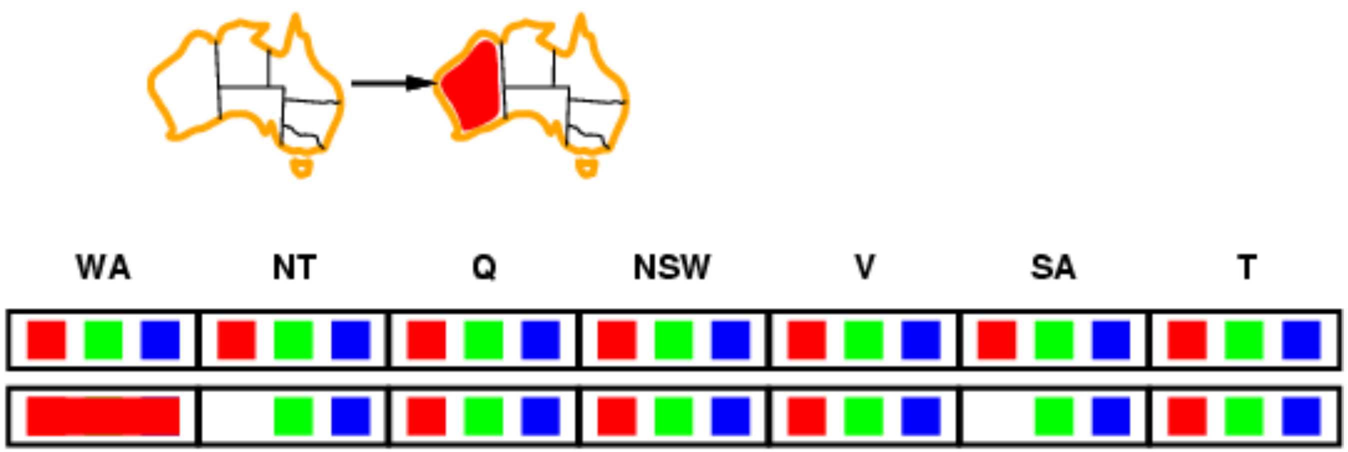


Forward checking

- Idea:
  - Tenere traccia dei valori legali possibili per le variabili non assegnate
  - Terminare la ricerca quando una qualsiasi variabile non ha valori legali

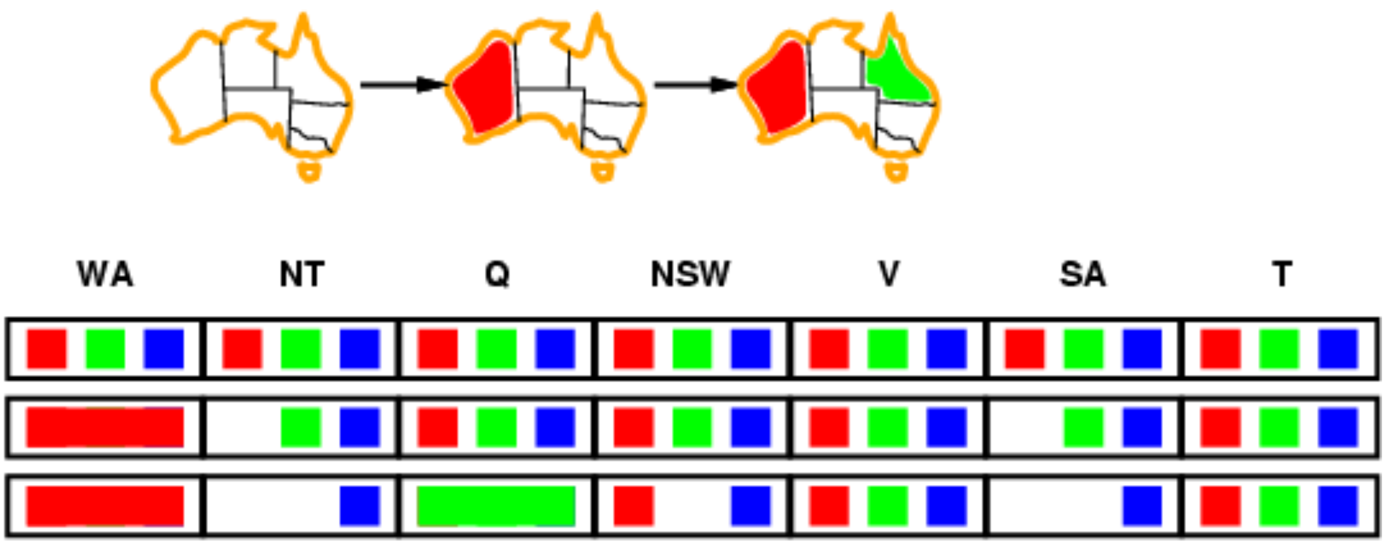
Forward checking

- Idea:
  - Tenere traccia dei valori legali possibili per le variabili non assegnate
  - Terminare la ricerca quando una qualsiasi variabile non ha valori legali



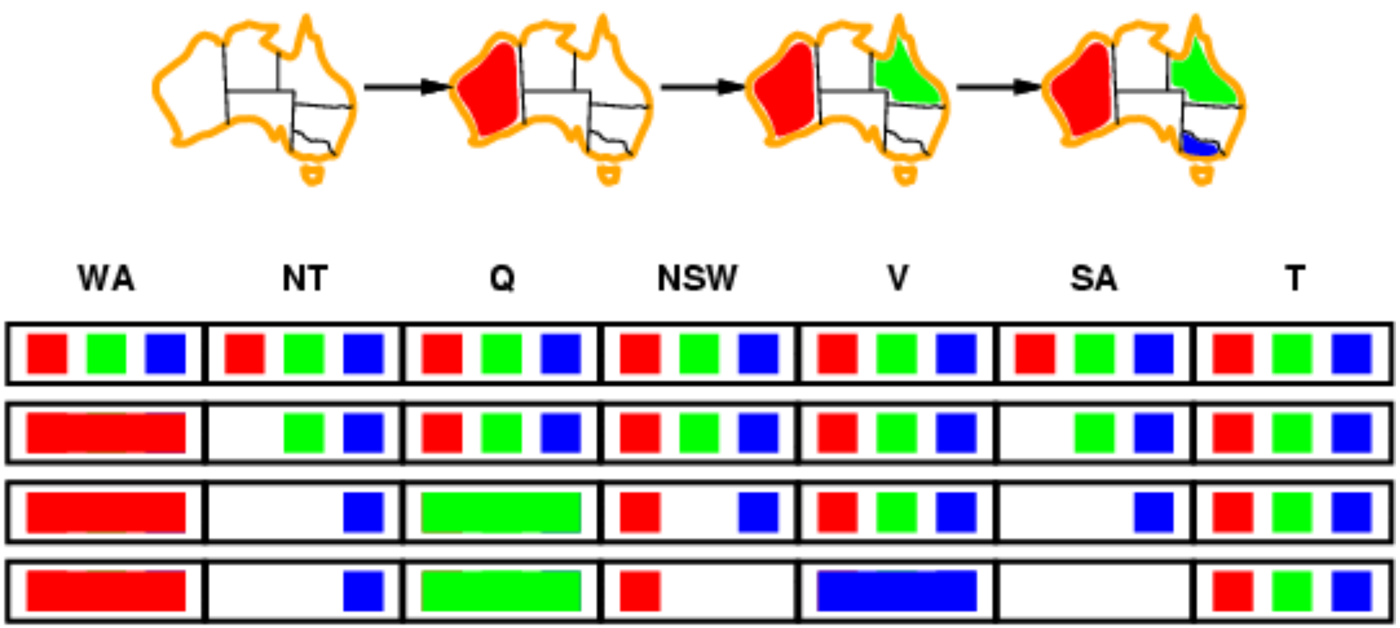
Forward checking

- Idea:
  - Tenere traccia dei valori legali possibili per le variabili non assegnate
  - Terminare la ricerca quando una qualsiasi variabile non ha valori legali



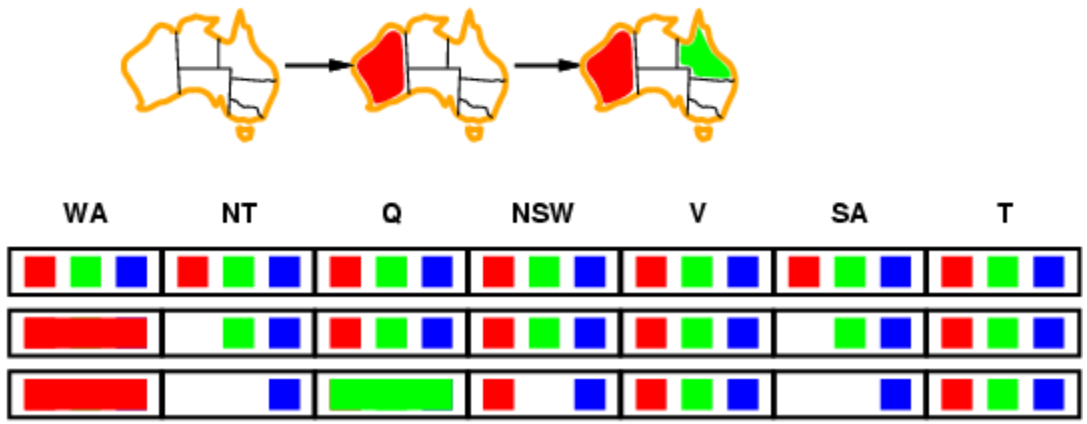
Forward checking

- Idea:
  - Tenere traccia dei valori legali possibili per le variabili non assegnate
  - Terminare la ricerca quando una qualsiasi variabile non ha valori legali



Propagazione di vincoli

- Il forward chaining propaga le informazioni da variabili assegnate a variabili non assegnate, ma non fornisce una rilevazione precoce dei fallimenti:



- NT e SA non possono essere blue!
- La propagazione dei vincoli ripetutamente impone dei vincoli locali

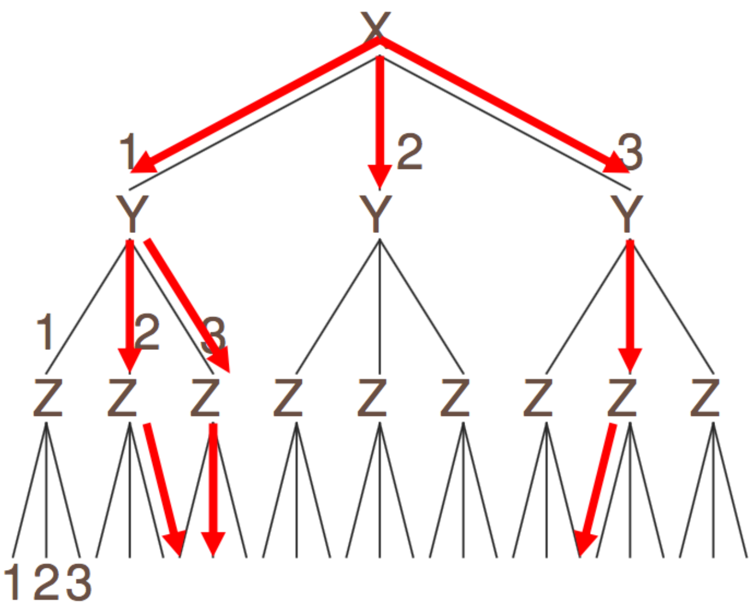
Consistenza degli archi

- Un metodo veloce per propagare i vincoli.
  - Nel grafo di vincoli, un arco orientato da A a B è consistente se per ogni valore x di A c'è almeno un valore *y* di B consistente con x.
  - Quello che si fa è controllare la consistenza degli archi all'inizio e dopo ogni assegnamento (MAC – Maintaining Arc Consistency)
- Dopo un assegnamento di una variabile, si sfrutta il grafo dei vincoli per togliere i valori non più ammissibili delle altre variabili
  - Ad ogni assegnamento i valori dei domini delle variabili vengono ridotti
  - E quindi si tagliano dei rami dell'albero

Albero di Ricerca del MAC

Consideriamo il CSP

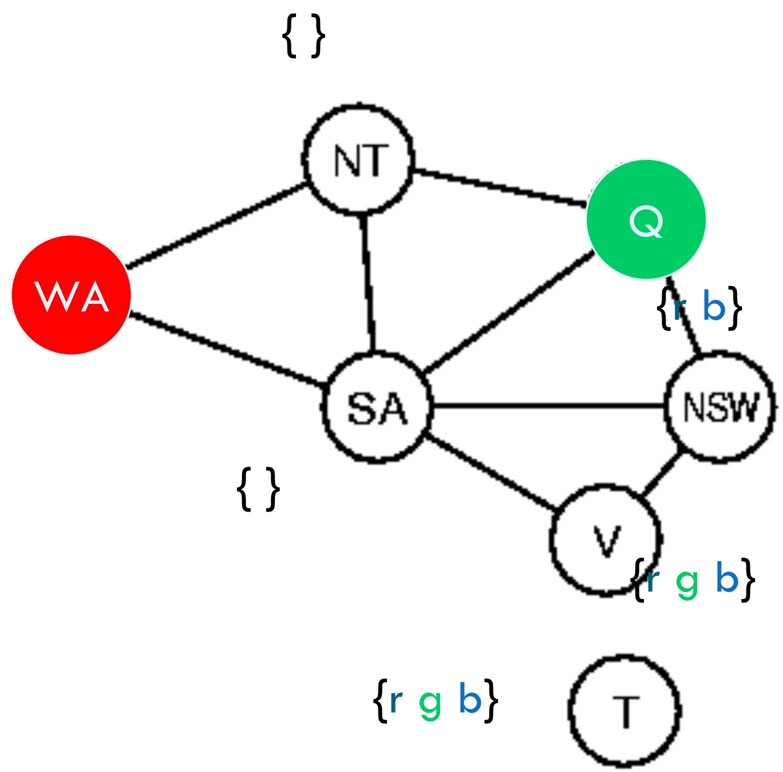
- Variabili X, Y, Z
- Domini $D_X = D_Y = D_Z = \{1, 2, 3\}$
- Vincoli $X \neq Y$ e $Y \neq Z$ e $X \neq Z$



X	Y	Z
1	1	1
1	1	2
1	1	3
1	2	1
1	2	2
1	2	3
1	3	1
1	3	2
1	3	3
2	1	1
	...	
3	3	3

Esempio di MAC

- WA=r
- Q=g
- Si scopre subito che non va bene



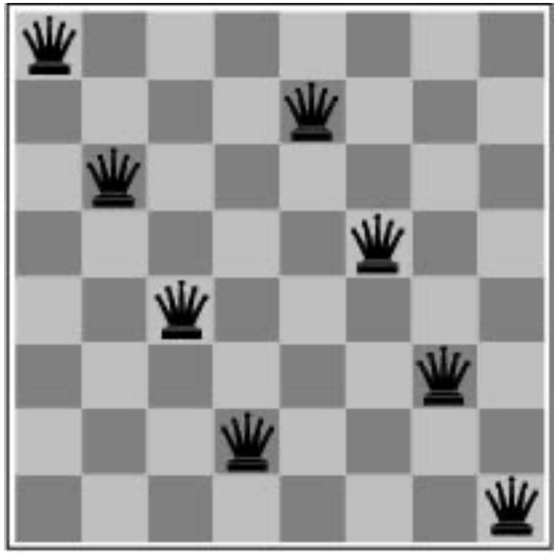
CSP con miglioramento iterativo

- Si parte con tutte le variabili assegnate (tutte le regine sulla scacchiera) e ad ogni passo si modifica l'assegnamento ad una variabile per cui un vincolo è violato (si muove una regina minacciata su una colonna).
- È un algoritmo di *riparazione euristica*.
- Un'euristica nello scegliere un nuovo valore potrebbe essere quella dei *conflitti minimi*: si sceglie il valore che crea meno conflitti.

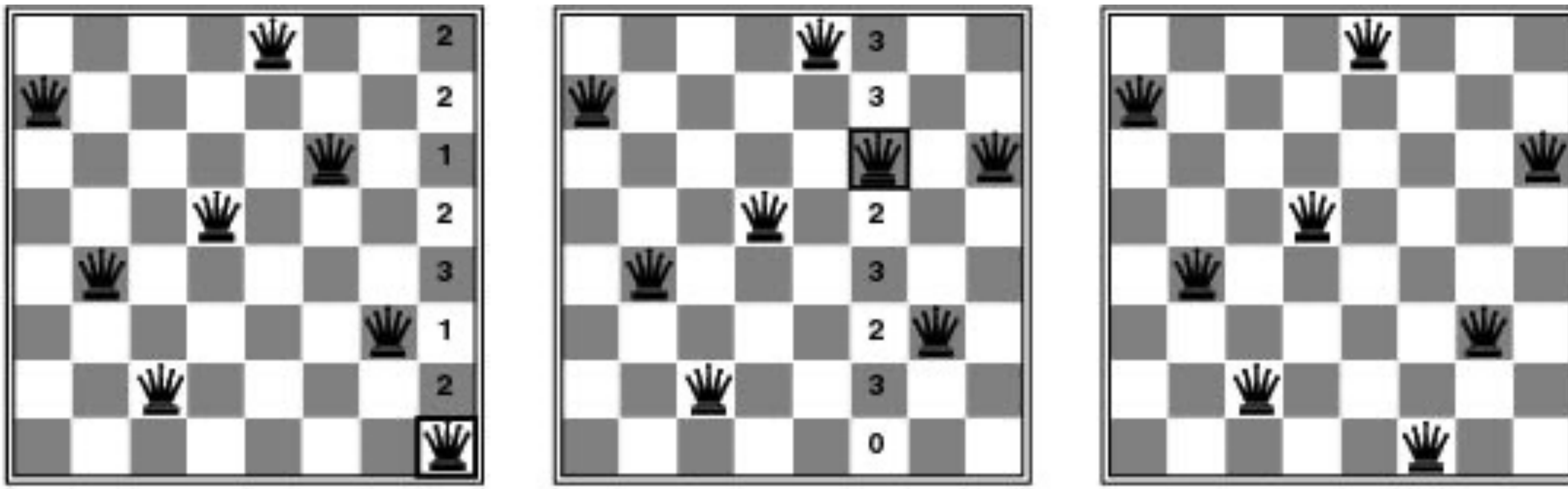
Il problema delle 8 regine come CSP

Formulazione come CSP:

- V_i: posizione della regina nella colonna i-esima
- D_i: {1 … 8}
- Vincoli di "*non-attacco*" tra V_1 e V_2: {<1,3> <1,4> <1,5>… <1,8><2,4><2,5> …<2,8> …}



Esempio



- *Molto efficace*: 1 milione di regine in 50 passi!
- Utilizzabile in problemi reali di progettazione e *scheduling* (anche real-time)

Metodi Riparativi (ricerca locale)

- Di solito ci sono una serie di plateau
  - Potrebbero esserci milioni di assegnamenti che distano solo un conflitto dalla soluzione
  - Mosse laterali per uscire dai plateau
  - Tabù search: tiene un elenco di stati visitati di recente cosi si evita di ritornarci
  - Simulated annealing
- La ricerca locale può essere usata in ambienti online quando il problema può cambiare
  - Scheduling voli aerei in caso di imprevisti (mal tempo)

Metodi Riparativi vs. Costruttivi

- Gli algoritmi costruttivi funzionano bene soprattutto su CSP-binari (o con pochi vincoli e pochi valori):
  - Es: complessità di AC-3 = $O(nk^3)$ dipende da k=valori e n=archi
- Diventano poco gestibili con Constraint Networks ad arità maggiore e con molti valori.
  - Es: N-regine con N > 106.
- Gli algoritmi riparativi, locali, forniscono **meno garanzie teoriche**, ma nel caso di problemi molto complessi risultano efficienti nella pratica.
- Gli algoritmi riparativi non sono completi. Si tratta infatti di algoritmi “locali”.

Riassunto

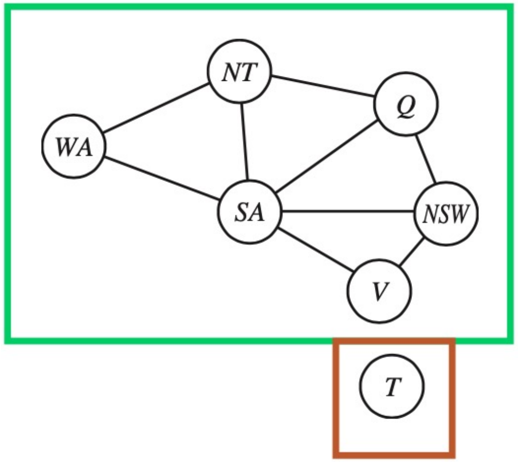
- Algoritmi di propagazione di vincoli (inferenza)
  - Utilizzano i vincoli per ridurre il numero di valori legali per una variabile, che a sua volta può ridurre i valori legali per un'altra variabile e cosi via.
  - Algoritmo AC-3
- Algoritmi di ricerca
  - Operano su assegnamenti di valori alle variabili
  - Backtracking
- Alternanza di ricerca ed inferenza
  - Backtracking con forward checking
  - Backtracking con MAC
- Ricerca locale

Sotto problemi indipendenti

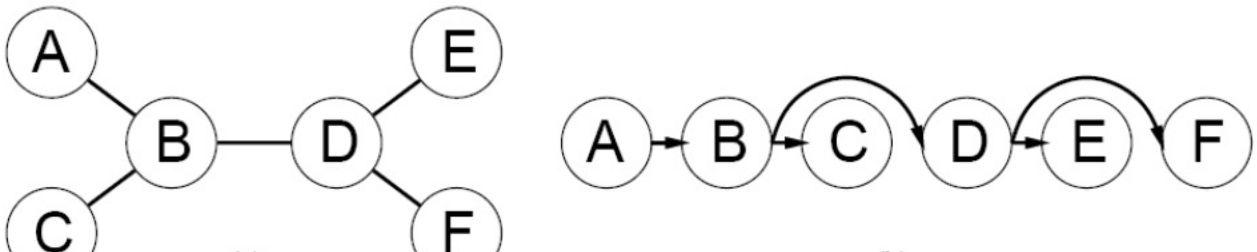
- La struttura del problema, rappresentata dal grafico dei vincoli, può essere utilizzata per trovare rapidamente soluzioni. Esamineremo problemi con una struttura specifica e strategie per migliorare il processo di ricerca di una soluzione.
- Il primo caso ovvio è quello dei **sottoproblemi indipendenti**.
- Nell'esempio di colorazione della mappa, colorare la Tasmania e colorare la terraferma sono sotto-problemi indipendenti:
  - qualsiasi soluzione la terraferma combinata con qualsiasi soluzione per la Tasmania produce una soluzione per l'intera mappa.
- Ciascun **componente collegato** del grafico del vincolo corrisponde a un sotto-problema CSP_i
- Se l'assegnazione S_i è una soluzione di CSP_i, allora $\bigcup_i S_i$ è una soluzione di $\bigcup_i CSP_i$

Sotto problemi indipendenti

- Il risparmio nel tempo di calcolo è rilevante
- *n* # variabili
- *c* # variabili per sotto-problemi
- *d* dimensione del dominio
- *n/c* problemi indipendenti
- $O(d^c)$ complessità nel risolverne uno
- $O(d^c\ n/c)$ *lineare* sul numero di variabili *n* piuttosto che $O(d^n)$ esponenziale!
- Dividere un CSP booleano con 80 variabili in quattro sotto-problemi riduce il tempo di soluzione nel caso peggiore dalla vita dell'universo a meno di un secondo !!!



La struttura dei problemi: alberi



(a) (b)

a) Un grafo di vincoli è un albero quando due nodi sono collegati da un solo percorso; possiamo scegliere qualsiasi variabile alla radice di un albero. A in fig (b).
b) Scelta una variabile come radice, l'albero induce un **ordinamento topologico** sulle variabili. I figli di un nodo sono elencati dopo il loro genitore.

Directed Arc Consistency (DAC)

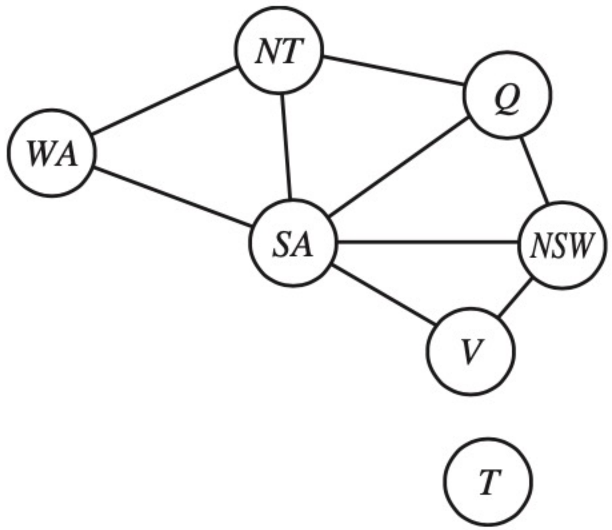
- Un CSP è definito come **direct arc-consistent** secondo un ordinamento delle variabili X_1, X_2, ... X_n se e solo se ogni X_i è arco consistente con ogni X_j per j> i.
- Possiamo fare in modo che un grafo a forma di albero sia *direct arc-consistent* in un passaggio sulle *n* variabili; ogni passaggio deve confrontare fino a *d* possibili valori di dominio per due variabili, per un tempo totale di $O(nd^2)$.
- Tree-CSP-solver:
  1. Procedendo da X_n a X_2, gli archi PARENT(X_i) $\rightarrow$ X_i sono consistenti riducendo il dominio di X_i, se necessario. Può essere fatto in un solo passaggio.
  2. Procedendo da X_1 a X_n assegnare valori alle variabili; non c'è bisogno di ***backtracking*** poiché ogni valore per un padre ha almeno un valore legale per il figlio.

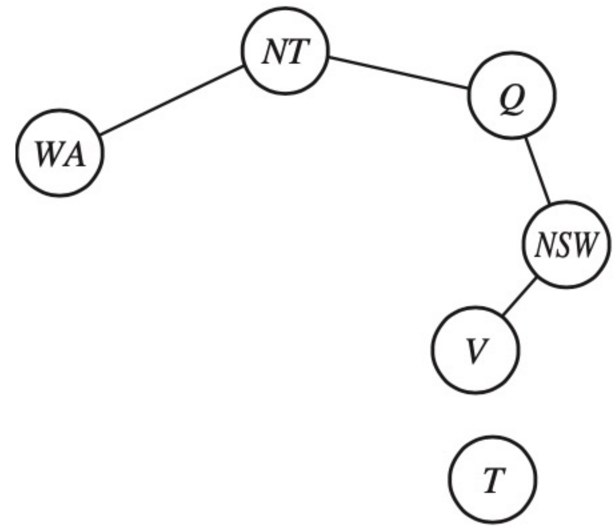
Tree-CSP-Solver

function TREE-CSP-SOLVER(*csp*) **returns** a solution, or failure
  inputs: *csp*, a CSP with components $X,\ D,\ C$

  $n \leftarrow$ number of variables in X
  $assignment \leftarrow$ an empty assignment
  $root \leftarrow$ any variable in X
  $X \leftarrow$ TOPOLOGICALSORT(X, $root$)
  for $j = n$ **down to** 2 **do**
    MAKE-ARC-CONSISTENT(PARENT(X_j), X_j)
    if it cannot be made consistent **then return** *failure*
  for $i = 1$ **to** n **do**
    $assignment[X_i] \leftarrow$ any consistent value from D_i
    if there is no consistent value **then return** *failure*
  return *assignment*

Ridurre grafi in alberi





a) Se potessimo eliminare l'Australia Meridionale, il grafico diventerebbe un albero.

b) Questo può essere fatto fissando un valore per SA (che in questo caso non ha importanza) e rimuovendo valori incoerenti dalle altre variabili.

Insieme di taglio dei Cicli

- In generale, dobbiamo applicare una strategia di suddivisione del dominio, provando con diversi assegnamenti:

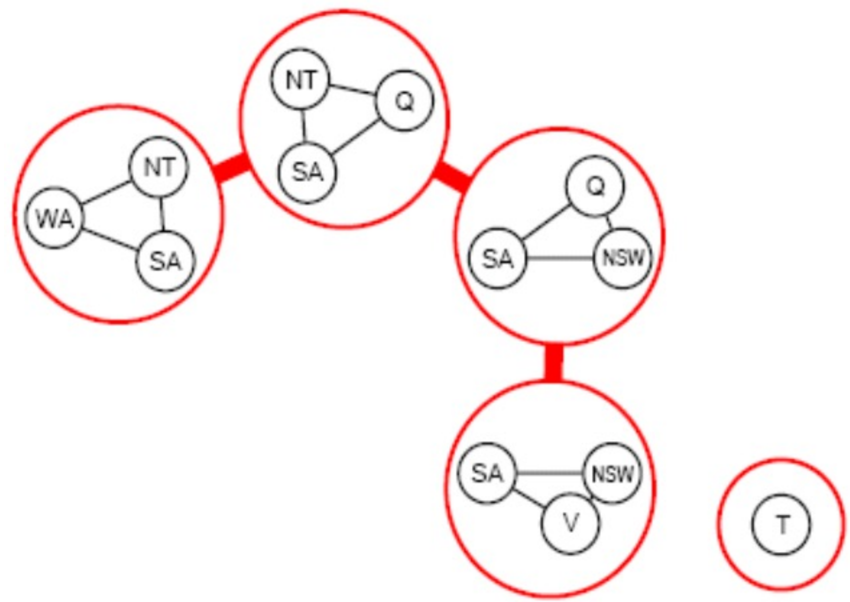
1. Scegli un sottoinsieme S delle variabili del CSP in modo tale che il grafo dei vincoli diventi un albero dopo la rimozione di S. S è chiamato **cycle cutset**.
2. Per ogni possibile assegnazione coerente alle variabili in S:
   a) rimuovere dai domini delle variabili rimanenti tutti i valori che sono incompatibili con l'assegnamento per S
   b) Se il rimanente CSP ha una soluzione, restituirla insieme all'assegnazione per S.

- Complessità temporale: $O(d^c (n - c) d^2)$
  - dove *c* è la dimensione del cycle cutset e *d* la dimensione del dominio
- Dobbiamo provare ciascuna delle combinazioni d^c di valori per le variabili in S, e per ogni combinazione dobbiamo risolvere un problema ad albero di dimensioni (n - c).

Decomposizione ad albero

a) L'approccio consiste in una decomposizione ad albero del grafo dei vincoli in una serie di sotto-problemi connessi.

b) Ogni sotto-problema viene risolto in modo indipendente e le soluzioni risultanti vengono quindi combinate in modo intelligente



Proprietà di una decomposizione

- Una decomposizione dell'albero deve soddisfare i tre requisiti seguenti:
  1. Ogni variabile nel problema originale appare in almeno uno dei problemi secondari.
  2. Se due variabili sono collegate da un vincolo nel problema originale, devono apparire insieme (insieme al vincolo) in almeno uno dei problemi secondari.
  3. Se una variabile appare in due sotto-problemi nella struttura, deve apparire in ogni sotto-problema lungo il percorso che collega quei sotto-problemi.
- Le condizioni 1-2 assicurano che tutte le variabili e i vincoli siano rappresentati nella decomposizione.
- La condizione 3 riflette il vincolo secondo il quale ogni data variabile deve avere lo stesso valore in ogni sotto-problema in cui appare; i collegamenti che uniscono i sotto-problemi nella struttura impongono questo vincolo.

Risolvere un problema decomposto

- Risolviamo ogni sotto-problema in modo indipendente. Se un problema non ha soluzione, il problema originale non ha soluzione.
- Mettere insieme le soluzioni. Risolviamo un meta-problema definito come segue:
  - Ogni sotto-problema è una "mega-variabile" il cui dominio è l'insieme di tutte le soluzioni per il sotto-problema

    Ex. Dom (X1) = {⟨WA = r, SA = b, NT = g⟩ ...} le 6 soluzioni al primo sottoproblema
  - I vincoli assicurano che le soluzioni dei sottoproblemi assegnino gli stessi valori alle variabili che condividono.
- Idealmente dovremmo trovare, tra i molti possibili, una decomposizione dell'albero con **larghezza minima** (la dimensione del sottoproblema più grande - 1). Questo è NP-hard. Se *w* è la larghezza minima dell'albero delle possibili decomposizioni dell'albero, la complessità è O (nd^{w+1}).