# Satisfacción de restricciones

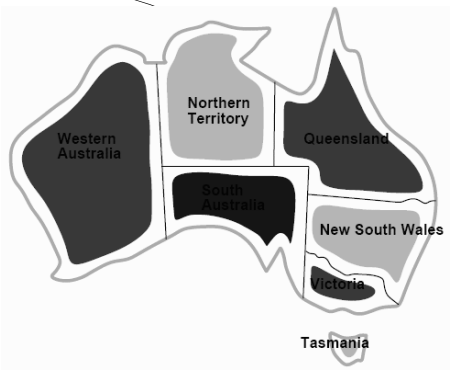
# El origen

- http://www.youtube.com/watch?v=495nCzxM9PI

# Algunos problemas

- Colorear mapas
  - Tengo 7 estados, tres colores
  - Dos estados no tienen el mismo color
- Asignar profesores a un horario
  - Tengo 20 profesores, para 20 clases
  - Los profesores deben dar su clase en la que son expertos


Variables $WA, NT, Q, NSW, V, SA, T$
Domains $D_i = \{red, green, blue\}$
Constraints: adjacent regions must have different colors
e.g., $WA \neq NT$ (if the language allows this), or
$(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), \ldots\}$




# El tipo de problemas

- Variables:
  - $X_1 \ldots X_n$
- Dominios para cada variable:
  - $D_1 \ldots D_n$
- Restricciones
  - $R_1 \ldots R_m$
- Satisfacción de restricciones

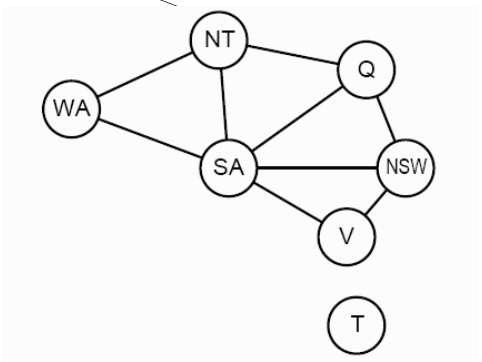
## Asignación y el objetivo

- Una asignación
  - $\{X_1 = v_1 \ldots X_n = v_n\}$ donde $v_1 \subseteq D_1 \ldots v_n \subseteq D_n$
- El objetivo
  - $\{X_1 = v_1 \ldots X_n = v_n\}$ donde $v_1 \subseteq D_1 \ldots v_n \subseteq D_n$
    Tal que
    $\{X_1 = v_1 \ldots X_n = v_n\}$ satisface a $R_1 \ldots R_m$

## SR un problema

## SR como "problema"

- Estado inicial
- Acción
- Prueba de arribo
- Función de costo




## Tipos de problemas de SR

Por su dominio

- Discretos
  - 8-queen, colorando
  - Variables booleanas: 3SAT
- Infinitos
  - No se puede enumerar las restricciones!!
  - Lenguaje de restricciones
    - Lineares y no lineares
  - Continuos: Continuos : programación lineal, cuadrática, segundo orden

## Tipos de problemas de SR

Por el tipo de restricciones

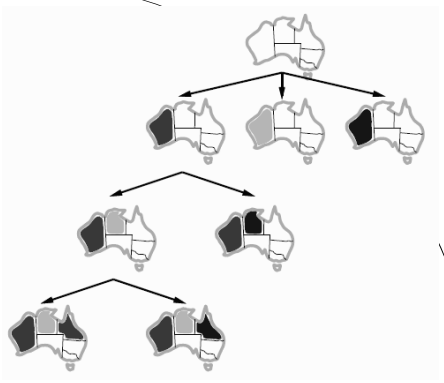
- Unarias: restricción para una sola variable
- Binarias: restricción para dos variables
- De orden superior
  - Todas las variables son diferentes
- Absolutas y de preferencia

## SR con búsqueda

- Supongamos que tenemos
  - n variables
  - d posibles asignaciones para cada variablle
- Con preferencia por anchura
  - Factor de ramificación ??
  - Profundidad ??
  - Máxima secuencia ??

## SR con búsqueda

- Conmutatividad
  - Si el orden de aplicar las acciones no tiene efecto en el resultado
- Backtracking es aplicable a problemas de SR
- El orden en asignar las variables es importante!!!

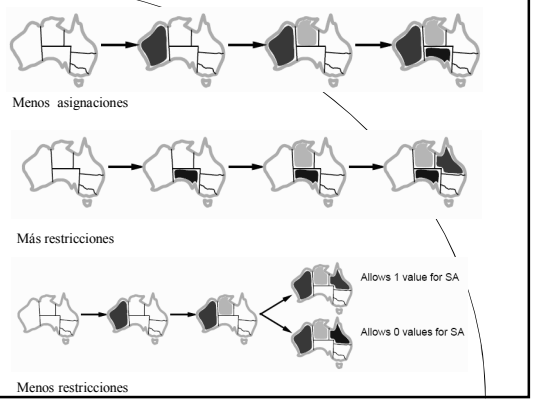


## Las preguntas del millón

- ¿Qué variable asignar en el siguiente paso?
- ¿Qué repercusiones tiene mi asignación actual con respecto a las siguientes?
- ¿Cuándo fallamos en la búsqueda, podemos evitar este camino en la búsqueda?

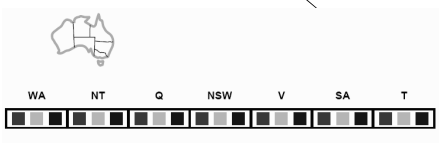
## El orden

- ¿Cómo selecciono la siguiente variable a asignar?
  - El orden en que me los dieron
  - La variable con la asignación con los menos asignaciones posibles.
    - No reduce 3 colores, pero sí a 8-queen/n-queen

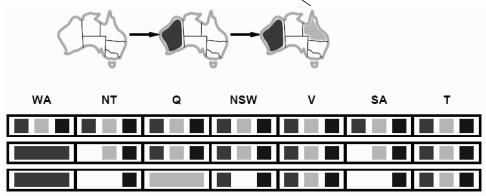


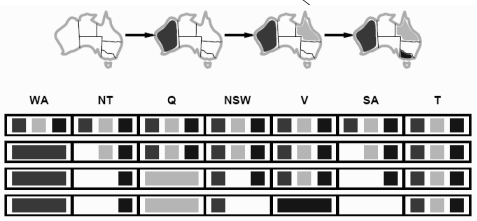
## Repercusiones

- Si asigno un valor a una variable, ese valor va a restringir los valores posibles para las variables que no han sido asignadas
  - X es v
  - Colecto las variables Y que no han sido asignadas y que comparten restricciones con X
  - Elimino del dominio de las variables colectadas aquellos valores que entren en conflicto con la asignacion X es v
- Chequeo anticipado (forward checking)



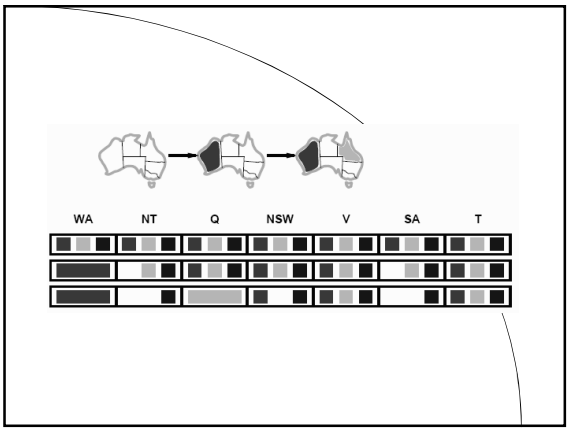




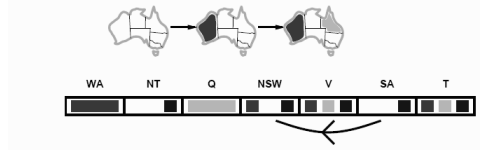

## Repercusiones

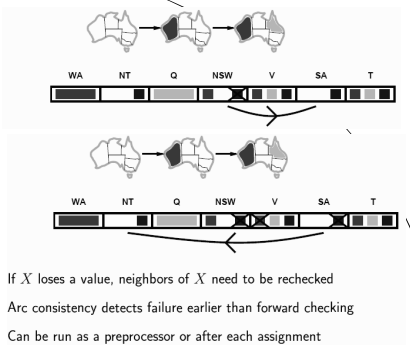
- Chequeo anticipado
  - Únicamente revisa las variables compatibles con X
  - ¿Qué pasa con los valores de los dominios eliminados?
- Chequeo de consistencia de arcos
  - Verificar las consecuencias de modificar los dominios
- Algoritmo C-3, más caro que chequeo anticipado, pero vale la pena




Simplest form of propagation makes each arc consistent

$X \rightarrow Y$ is consistent iff
for *every* value $x$ of $X$ there is *some* allowed $y$




If $X$ loses a value, neighbors of $X$ need to be rechecked

Arc consistency detects failure earlier than forward checking

Can be run as a preprocessor or after each assignment

## Casos especiales

- Todas diferentes ??
  - Es posible aplicar una restricción de orden superior para reducir los dominios
- A lo más???
  - Los valores de unas variables a lo más deben sumar una cantidad

## Fallos

- Identificar el conjunto de variables conflictivas
  - El conjunto de variables conectados por restricciones a una asignación
- Brinco hacia atrás:
  - A la variable más reciente del conjunto conflictivo
  - O regresar, a las variables antes del conjunto conflictivo

## Búsqueda local

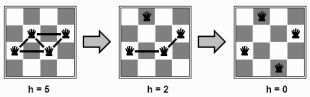
- Un solo estado en cada paso!!

States: 4 queens in 4 columns ($4^4 = 256$ states)

Operators: move queen in column

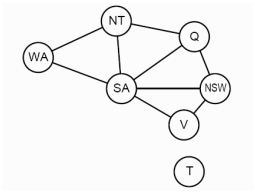
Goal test: no attacks

Evaluation: $h(n)$ = number of attacks

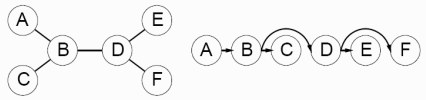


## Naturaleza del problema

- Los problemas vienen con una estructura
- ¿Podemos utilizar esa estructura?

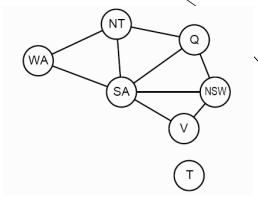



1. Choose a variable as root, order variables from root to leaves such that every node's parent precedes it in the ordering




2. For $j$ from $n$ down to 2, apply RemoveInconsistent$(Parent(X_j), X_j)$
3. For $j$ from 1 to $n$, assign $X_j$ consistently with $Parent(X_j)$

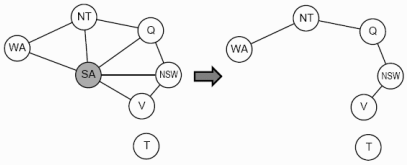
- Problema:




- No es un árbol

- Dos soluciones:
  - Hacerlo un árbol a través de una asignación
  - Crear un meta árbol

Conditioning: instantiate a variable, prune its neighbors' domains




Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree

Cutset size $c$ $\Rightarrow$ runtime $O(d^c \cdot (n-c)d^2)$, very fast for small $c$