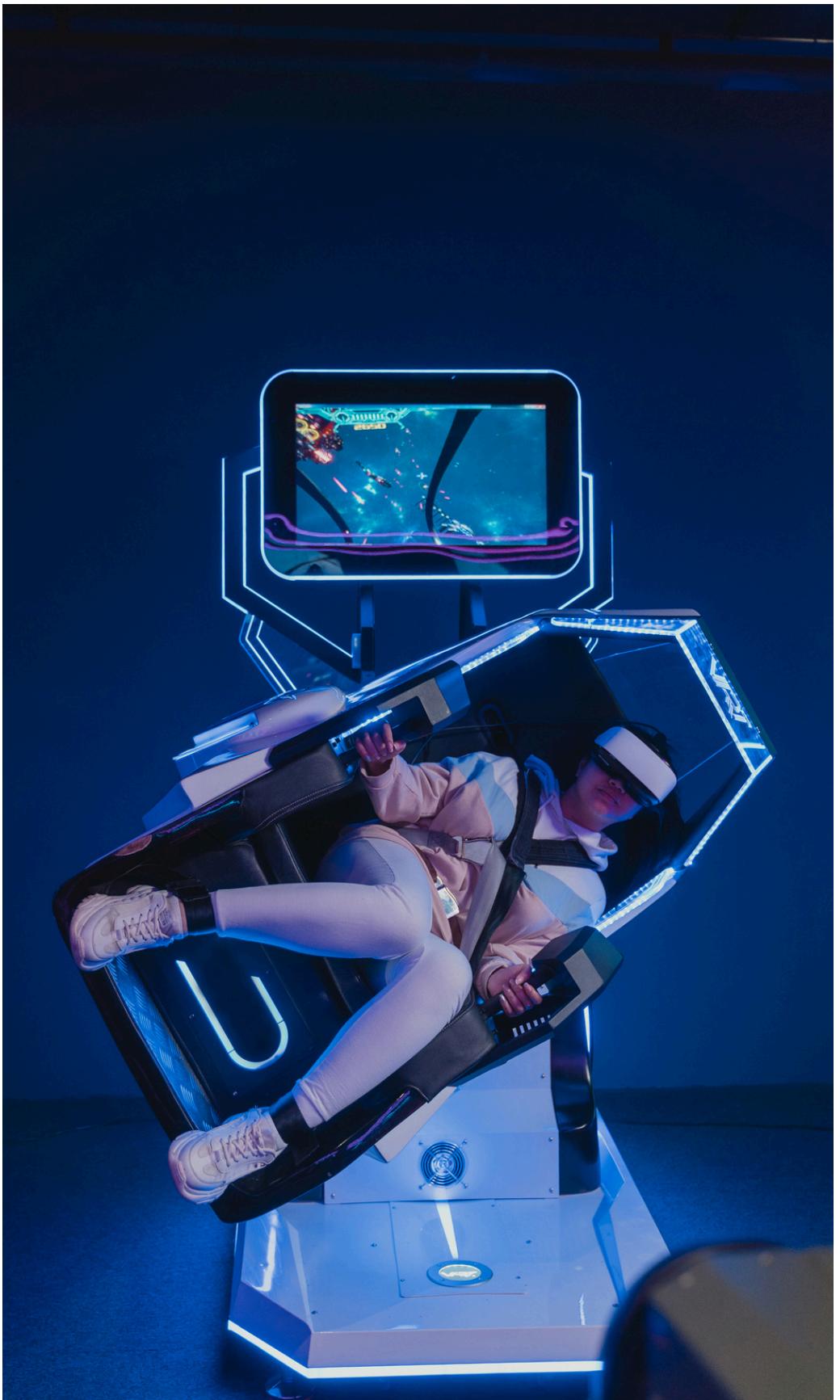




Constraint Satisfaction Problems

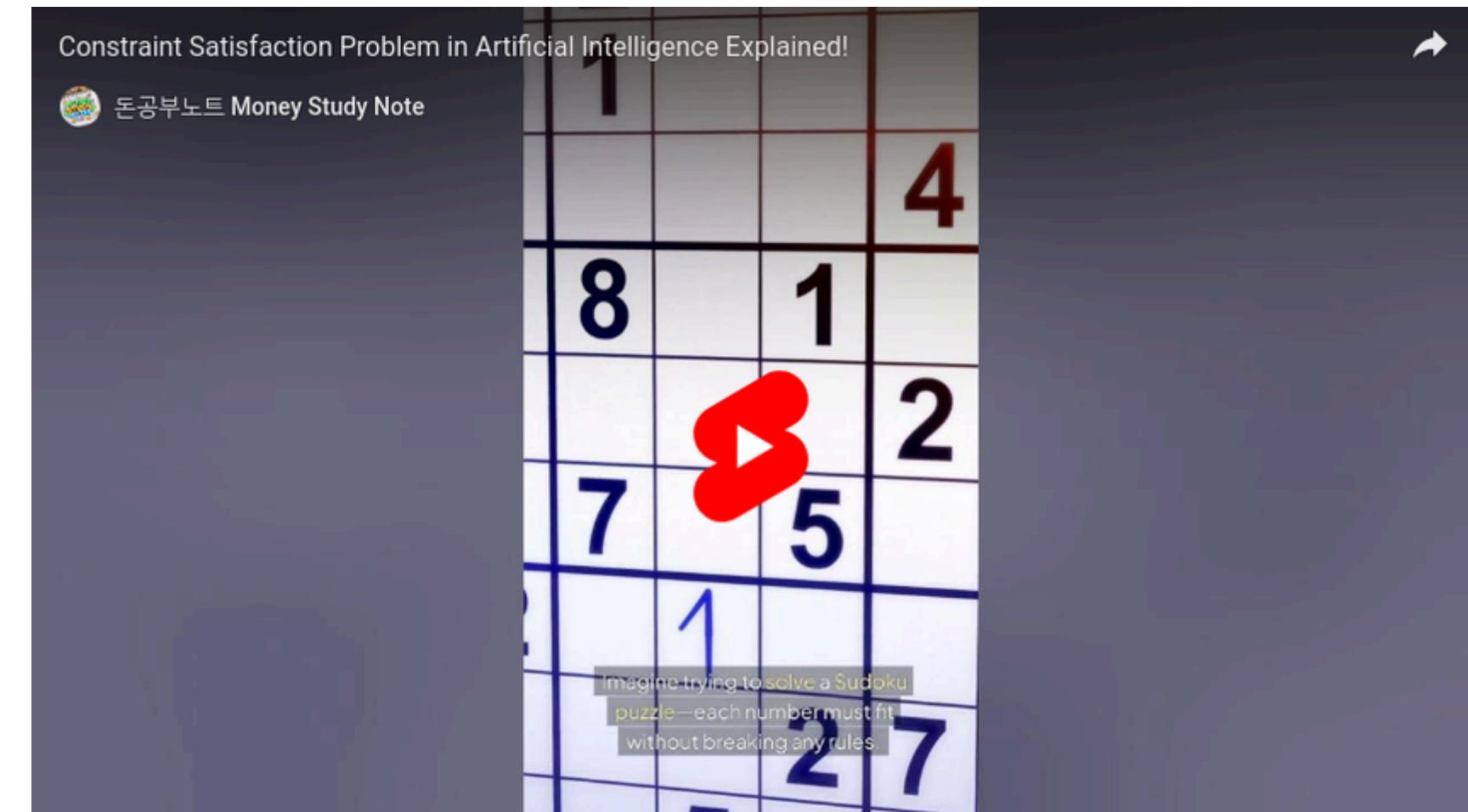


Agenda

- Definición
- Elementos de un CSP
- Ejemplos
- Métodos de resolución
- Aplicaciones

Definición

Son un tipo de problema en inteligencia artificial en los que se busca asignar valores a un conjunto de variables, cumpliendo ciertas restricciones o condiciones predefinidas.





Elementos

01

Conjunto de Variables (X)

Un conjunto finito de variables X

02

Dominios (D)

Cada variable X tiene un dominio D de posibles valores que puede tomar.

03

Restricciones (C)

Un conjunto de condiciones que especifican combinaciones permitidas de valores para ciertas variables

04



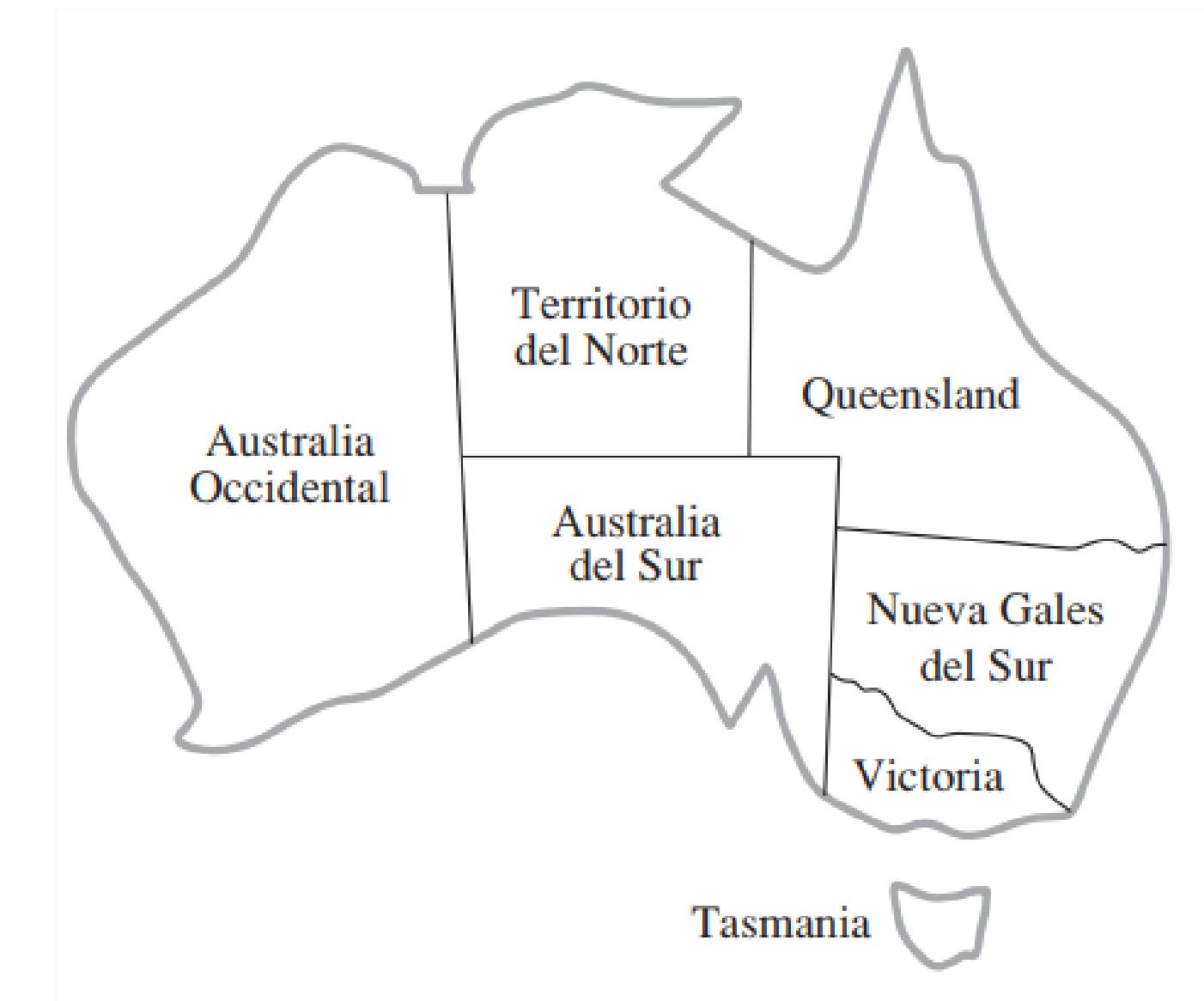
Tipos de Restricciones

- 01 **Unitarias** Afectan solo una variable
- 02 **Binarias** Afectan a dos variables
- 03 **High-Order** Afectan a tres o más variables
- 04

Ejemplo

Problema de Coloreo de Mapas

Dado un mapa con varias regiones, el objetivo es asignar un color a cada región de manera que ninguna región vecina comparta el mismo color.



Componentes del CSP

Variables: Cada región del mapa

Si tomamos el mapa de Australia con las siguientes regiones:

- WA (Australia Occidental)
- NT (Territorio del Norte)
- SA (Australia del Sur)
- Q (Queensland)
- NSW (Nueva Gales del Sur)
- V (Victoria)
- T (Tasmania)

$$\mathcal{X} = \{WA, NT, Q, NSW, V, SA, T\}$$

Dominios: Un conjunto de colores posibles

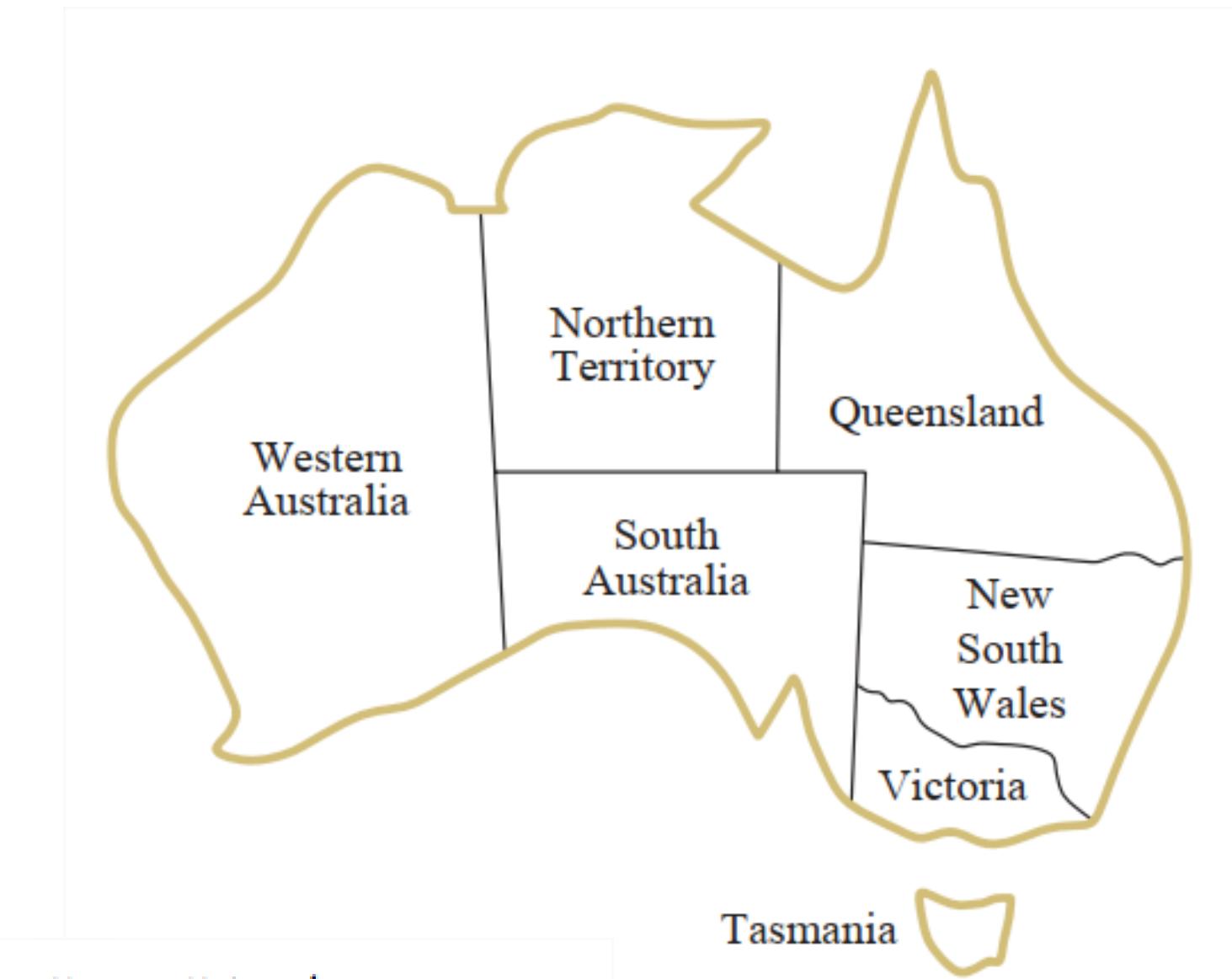
$$D = \{\text{rojo}, \text{verde}, \text{azul}\}$$

Restricciones: Dos regiones adyacentes no pueden tener el mismo color.

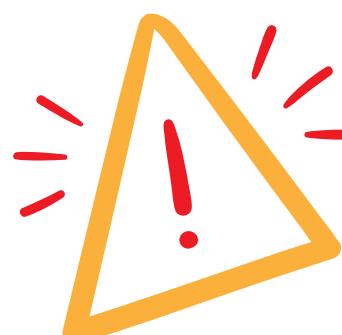
Cada región puede tener uno de tres colores: {rojo, verde, azul}. Las restricciones vienen de los límites geográficos entre regiones.

Restricciones

WA \neq NT
WA \neq SA
NT \neq SA
NT \neq Q
SA \neq Q
SA \neq NSW
SA \neq V
Q \neq NSW
NSW \neq V



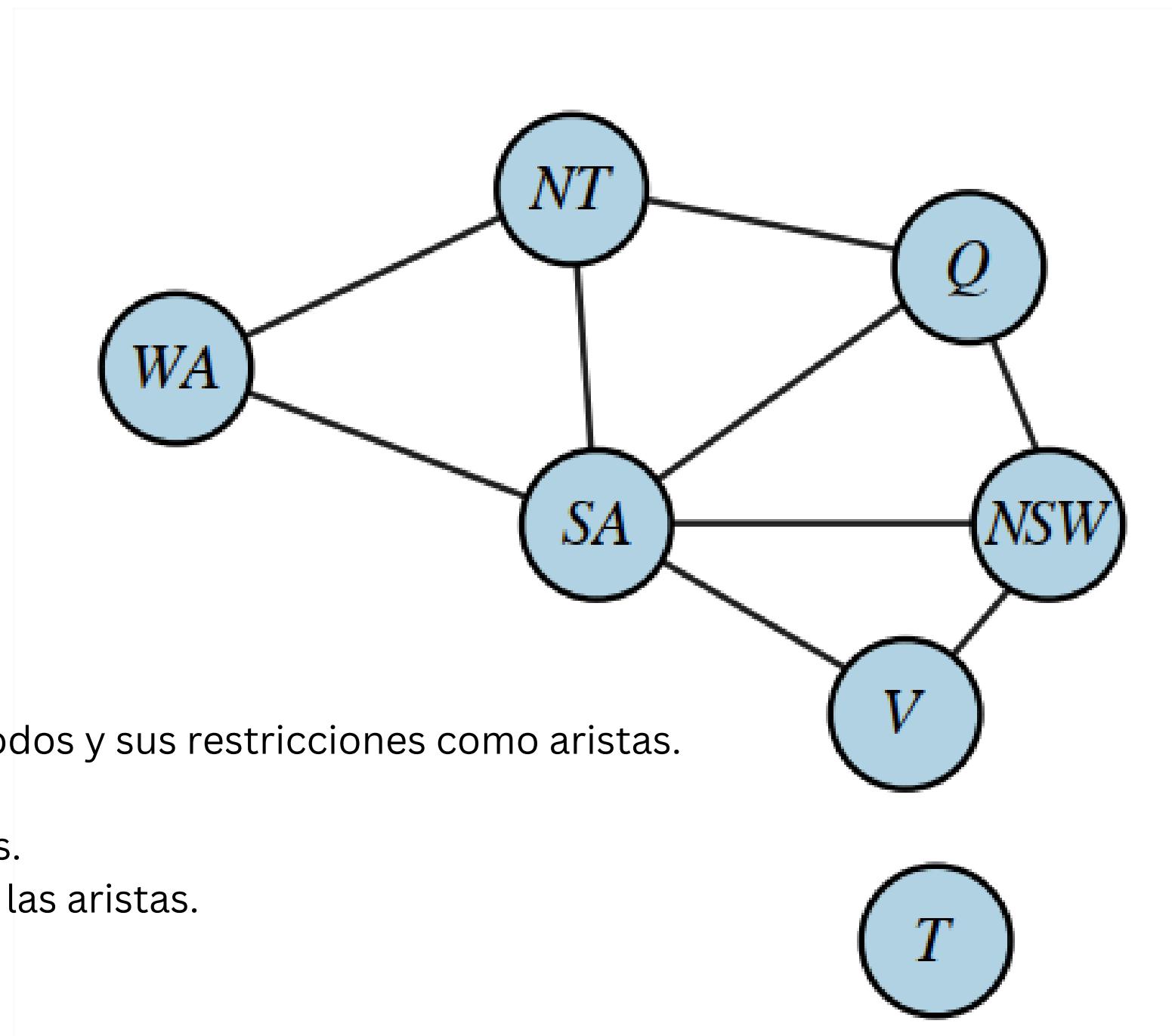
$$\mathcal{C} = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, \\ WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}.$$



Tasmania (T) no tiene restricciones porque no comparte frontera con ninguna otra región.



Grafo de Restricciones



Un grafo permite representar regiones como nodos y sus restricciones como aristas.

Ejemplo con Australia:

- Las regiones (WA, NT, SA, etc.) son los nodos.
- Las restricciones (WA ≠ NT, SA ≠ Q, etc.) son las aristas.

Esto ayuda a visualizar qué regiones están conectadas y deben cumplir la restricción de colores distintos.

Implementing CSP Algorithm

Representing the problem

Selecting a variable

Selecting a value

Termination

Backtracking

Applying constraints



Algoritmos para CSP

- **Backtracking (búsqueda con retroceso):** asigna valores secuencialmente y retrocede si hay conflictos.
- **Forward Checking:** evita asignaciones que llevarían a un fallo, reduciendo el espacio de búsqueda.
- **Arc Consistency:** Asegura que todas las restricciones binarias sean consistentes. (AC-3) para reducir el dominio de valores posibles antes de la búsqueda.
- **Heurísticas:**
 - **Minimum Remaining Values (MRV):** Selecciona la variable con menos valores posibles.
 - **Degree Heuristic:** Selecciona la variable involucrada en más restricciones.

Backtracking

Arrays:

- **VARIABLES**
- **RESTRICCIONES**
- **DOMINIO:** se recorre en la función ORDEN-VALORES-DOMINIO()

función BÚSQUEDA-CON-VUELTA-ATRÁS(*psr*) **devuelve** una solución, o fallo
devolver VUELTA-ATRÁS-RECURSIVA({ }, *psr*)

función VUELTA-ATRÁS-RECURSIVA(*asignación*, *psr*) **devuelve** una solución, o fallo
si *asignación* es completa **entonces devolver** *asignación*
var \leftarrow SELECCIONA-VARIABLE-NOASIGNADA(VARIABLES[*psr*], *asignación*, *psr*)
para cada *valor* **en** ORDEN-VALORES-DOMINIO(*var*, *asignación*, *psr*) **hacer**
si *valor* es consistente con *asignación* de acuerdo a las RESTRICCIONES[*psr*] **entonces**
 añadir {*var* = *valor*} a *asignación*
 resultado \leftarrow VUELTA-ATRÁS-RECURSIVA(*asignación*, *psr*)
 si *resultado* \neq fallo **entonces devolver** *resultado*
 borrar {*var* = *valor*} de *asignación*
devolver fallo

Figura 5.3 Un algoritmo simple de vuelta atrás para problemas de satisfacción de restricciones. El algoritmo se modela sobre la búsqueda primero en profundidad recursivo del Capítulo 3. Las funciones SELECCIONA-VARIABLE-NOASIGNADA y ORDEN-VALORES-DOMINIO pueden utilizarse para implementar las heurísticas de propósito general discutidas en el texto.

Pasos del Algoritmo Aplicado al Ejemplo

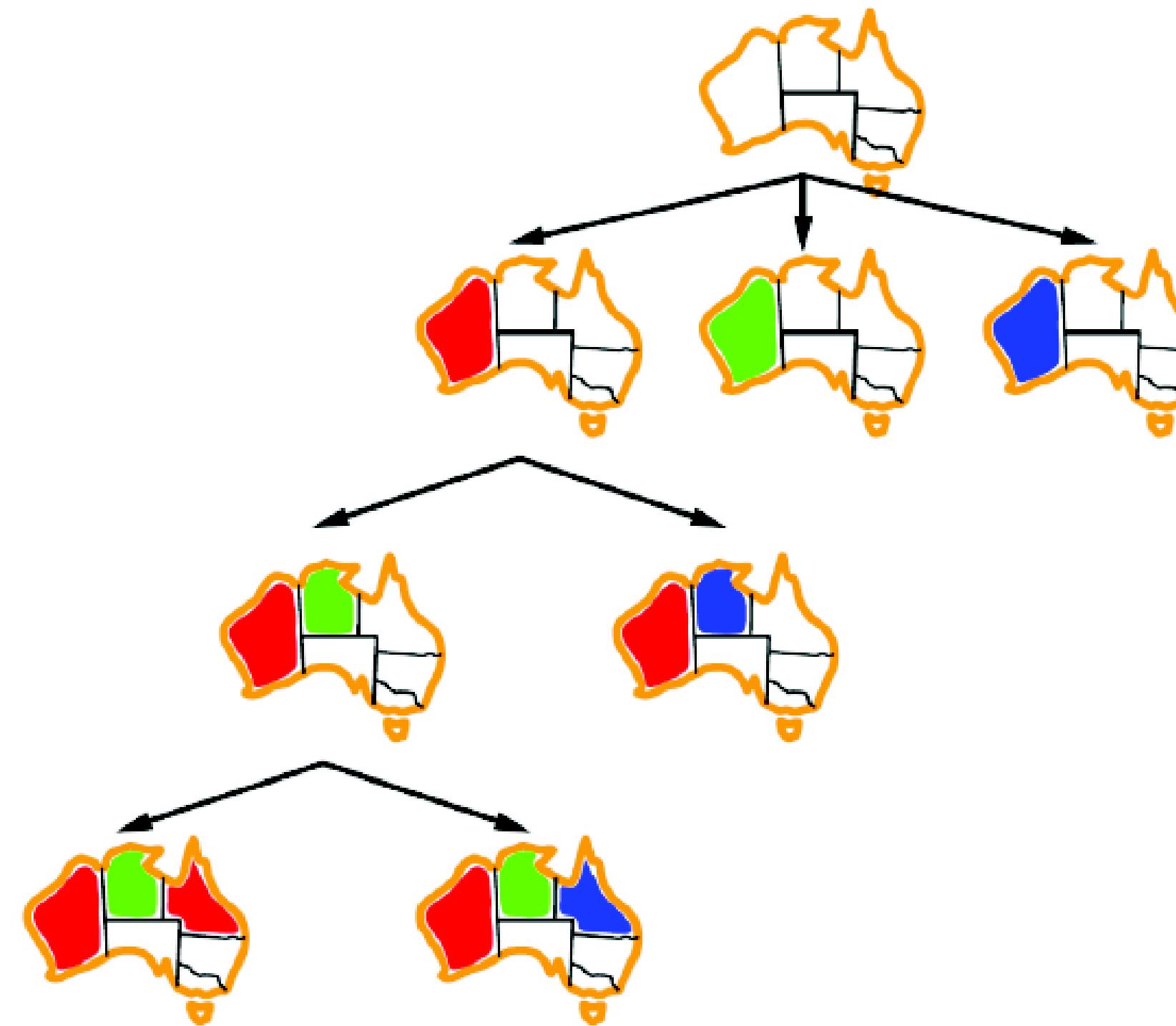
El backtracking explora asignaciones de colores y retrocede si encuentra un conflicto.

Se sigue el siguiente proceso:

1. Seleccionar una variable no asignada: Elige una región sin color asignado.
2. Asignar un valor: Asigna un color del dominio a la región seleccionada.
3. Verificar restricciones: Comprueba si la asignación viola alguna restricción con las regiones adyacentes ya coloreadas.
4. Si no hay violación, continúa con la siguiente región.
5. Si hay violación, deshaz la asignación y prueba con otro color.
6. Repetir: Repite el proceso hasta que todas las regiones estén coloreadas o se determine que no hay solución con los colores disponibles.
7. Retroceder (Backtrack): Si no se puede asignar un color válido a una región, retrocede a la región anterior y prueba con otro color.

Pasos del Algoritmo Aplicado al Ejemplo

El backtracking explora asignaciones de colores y retrocede si encuentra un conflicto.



Ejecución Paso a Paso

Paso 1: Iniciar con una asignación vacía

Asignación inicial: { }

Paso 2: Seleccionar la primera variable

Variable elegida: WA

Colores disponibles: {Rojo, Verde, Azul}

- ◆ Asignamos WA = Rojo

Asignación parcial: { WA = Rojo }

Paso 3: Seleccionar la siguiente variable

Variable elegida: NT

Colores disponibles: {Rojo, Verde, Azul}

- ◆ Probamos NT = Rojo

✗ No es válido porque WA = Rojo y WA ≠ NT (backtracking).

- ◆ Probamos NT = Verde

✓ Válido.

Asignación parcial: { WA = Rojo, NT = Verde }

Ejecución Paso a Paso

Paso 4: Seleccionar la siguiente variable

Variable elegida: SA

Colores disponibles: {Rojo, Verde, Azul}

- ◆ Probamos SA = Rojo
- ✗ No es válido porque WA = Rojo y WA ≠ SA (backtracking).
- ◆ Probamos SA = Azul
- ✓ Válido.

Asignación parcial: { WA = Rojo, NT = Verde, SA = Azul }

Paso 5: Seleccionar la siguiente variable

Variable elegida: Q

Colores disponibles: {Rojo, Verde, Azul}

- ◆ Probamos Q = Rojo
- ✗ No es válido porque NT = Verde y NT ≠ Q (backtracking).
- ◆ Probamos Q = Azul
- ✗ No es válido porque SA = Azul y SA ≠ Q (backtracking).
- ◆ Probamos Q = Verde
- ✓ Válido.

Asignación parcial: { WA = Rojo, NT = Verde, SA = Azul, Q = Verde }

Ejecución Paso a Paso

Paso 6: Seleccionar la siguiente variable

Variable elegida: NSW

Colores disponibles: {Rojo, Verde, Azul}

◆ Probamos NSW = Rojo

✓ Válido.

Asignación parcial: { WA = Rojo, NT = Verde, SA = Azul, Q = Verde, NSW = Rojo }

Paso 7: Seleccionar la siguiente variable

Variable elegida: V

Colores disponibles: {Rojo, Verde, Azul}

◆ Probamos V = Rojo

✗ No es válido porque NSW = Rojo y NSW ≠ V (backtracking).

◆ Probamos V = Verde

✓ Válido.

Asignación parcial: { WA = Rojo, NT = Verde, SA = Azul, Q = Verde, NSW = Rojo, V = Verde }

Ejecución Paso a Paso

Paso 8: Seleccionar la última variable

Variable elegida: T

Tasmania (T) no tiene restricciones con otras regiones, por lo que se puede asignar cualquier color.

- ◆ Asignamos T = Rojo
- ✓ Válido.

Asignación final: { WA = Rojo, NT = Verde, SA = Azul, Q = Verde, NSW = Rojo, V = Verde, T = Rojo }

¡Solución encontrada con backtracking!

Colores asignados a cada región:

- WA = Rojo
- NT = Verde
- SA = Azul
- Q = Verde
- NSW = Rojo
- V = Verde
- T = Rojo

Forward Checking

Qué hace:

Después de asignar un valor a una variable, elimina inmediatamente los valores inconsistentes de los dominios de las variables vecinas (relacionadas por restricciones).

Cuándo se aplica:

Durante la búsqueda (por ejemplo, en backtracking), solo cuando se asigna un valor a una variable.

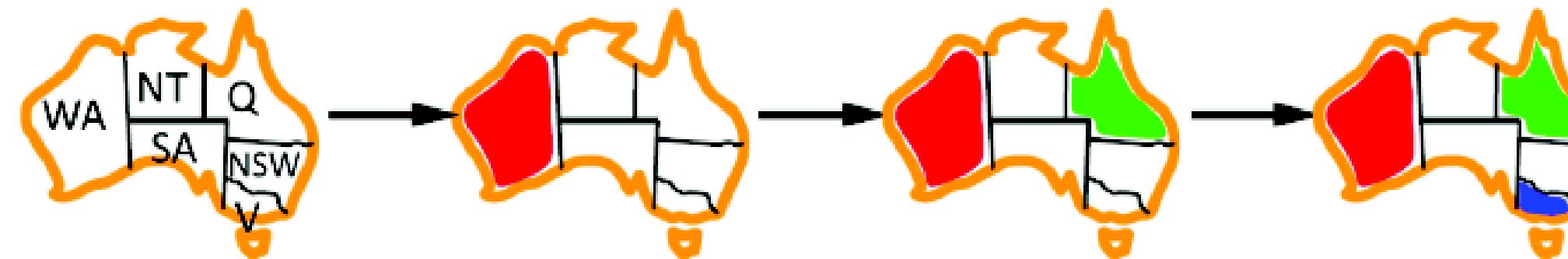
Ejemplo:

Imagina un Sudoku. Si colocas un "5" en una celda, el Forward Checking eliminaría el "5" de las celdas vecinas (misma fila, columna y región).

Limitación:

Solo mira un paso hacia adelante (variables directamente relacionadas). No garantiza que todas las restricciones del problema sean consistentes.

Forward Checking



	WA	NT	Q	NSW	V	SA
Paso 1	■ Red ■ Green ■ Blue					
Paso 2	■ Red	■ Green ■ Blue	■ Red ■ Green ■ Blue	■ Red ■ Green ■ Blue	■ Red ■ Green ■ Blue	■ Green ■ Blue
Paso 3	■ Red	■ ■ Blue	■ Green	■ Red ■ Blue	■ Red ■ Green ■ Blue	■ Blue
Paso 4	■ Red	■ ■ Blue	■ Green	■ Red	■ ■ ■ Blue	

Regla para este ejemplo, resolvemos de izquierda a derecha, tomando la variable con mayor numero de opciones disponibles

NOTA: Podemos concluir en el paso 4, que la actual distribucion no llevara a un estado de solucion consistente

Arc Consistency

Qué hace:

Asegura que todos los pares de variables conectadas por una restricción sean consistentes.

Cuándo se aplica:

Generalmente como preprocesamiento (antes de la búsqueda) o durante la búsqueda para reducir dominios.

Ejemplo:

En un problema de colorear un mapa, la consistencia de arco revisa que, para cada región adyacente, los colores disponibles no se solapen.

Ventaja:

Es más completa que el Forward Checking, ya que revisa todas las restricciones entre pares de variables, no solo las afectadas por una asignación reciente.

Arc Consistency

Ayuda a no cometer errores antes de empezar. Si revisas todas las reglas y eliminas las combinaciones que no funcionan

Para que haya **consistencia de arco** entre X e Y, cada valor en Dx debe tener al menos un valor en Dy que satisfaga la restricción, y viceversa.

Por ejemplo:

Si $X=1$, entonces Y puede ser 2,3,4 (todos son mayores que 1)

Si $X=2$, entonces Y puede ser 3,4 (mayores que 2).

Si $X=3$, entonces Y solo puede ser 4 (mayor que 3).

Variables: X e Y

Dominios:

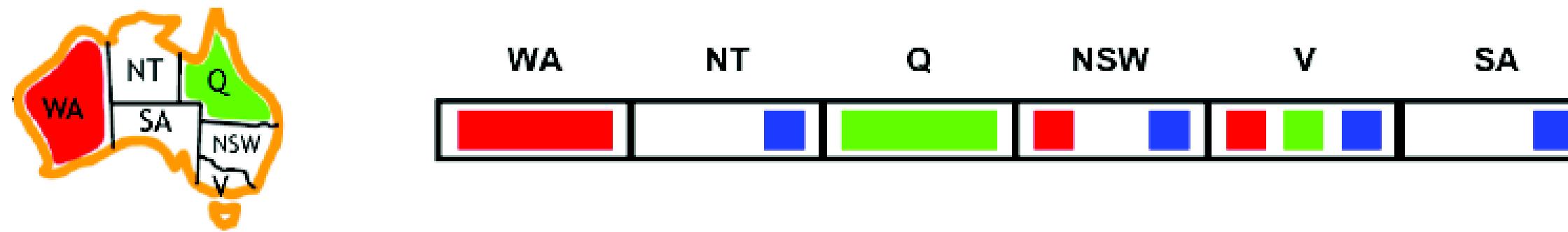
$D_x = \{1, 2, 3\}$

$D_y = \{2, 3, 4\}$

Restricciones: $X < Y$

Arc Consistency

Es una forma de propagación donde se asegura que todos los CSP son consistentes.



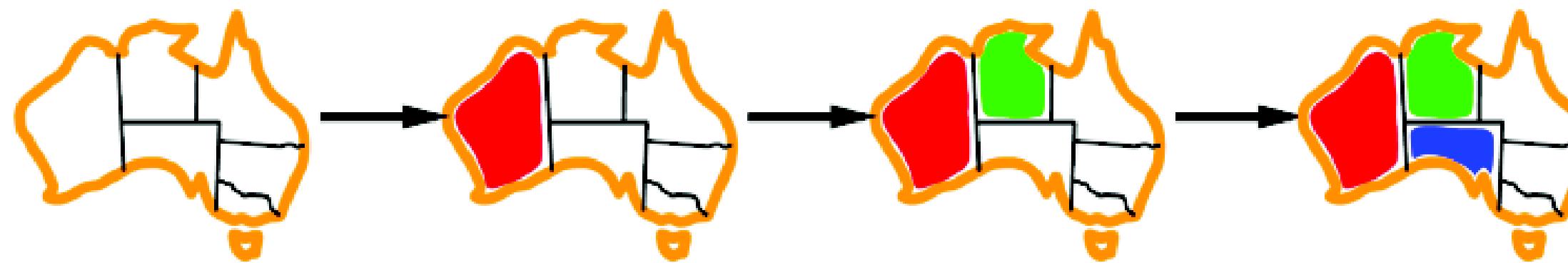
- Si X pierde un valor, los vecinos de X deben ser verificados
- Puede ser ejecutado como un pre-procesador o despues de cada asignación

Diferencias

Característica	Forward Checking	Consistencia de Arco
Alcance	Local (solo variables vecinas)	Global (todos los pares de variables)
Momento de aplicación	Durante la búsqueda (tras asignar valor)	Antes o durante la búsqueda
Complejidad	Más rápida (solo mira variables vecinas)	Más costosa (revisa todos los pares)
Pruning (poda)	Elimina valores de variables vecinas.	Elimina valores de cualquier variable

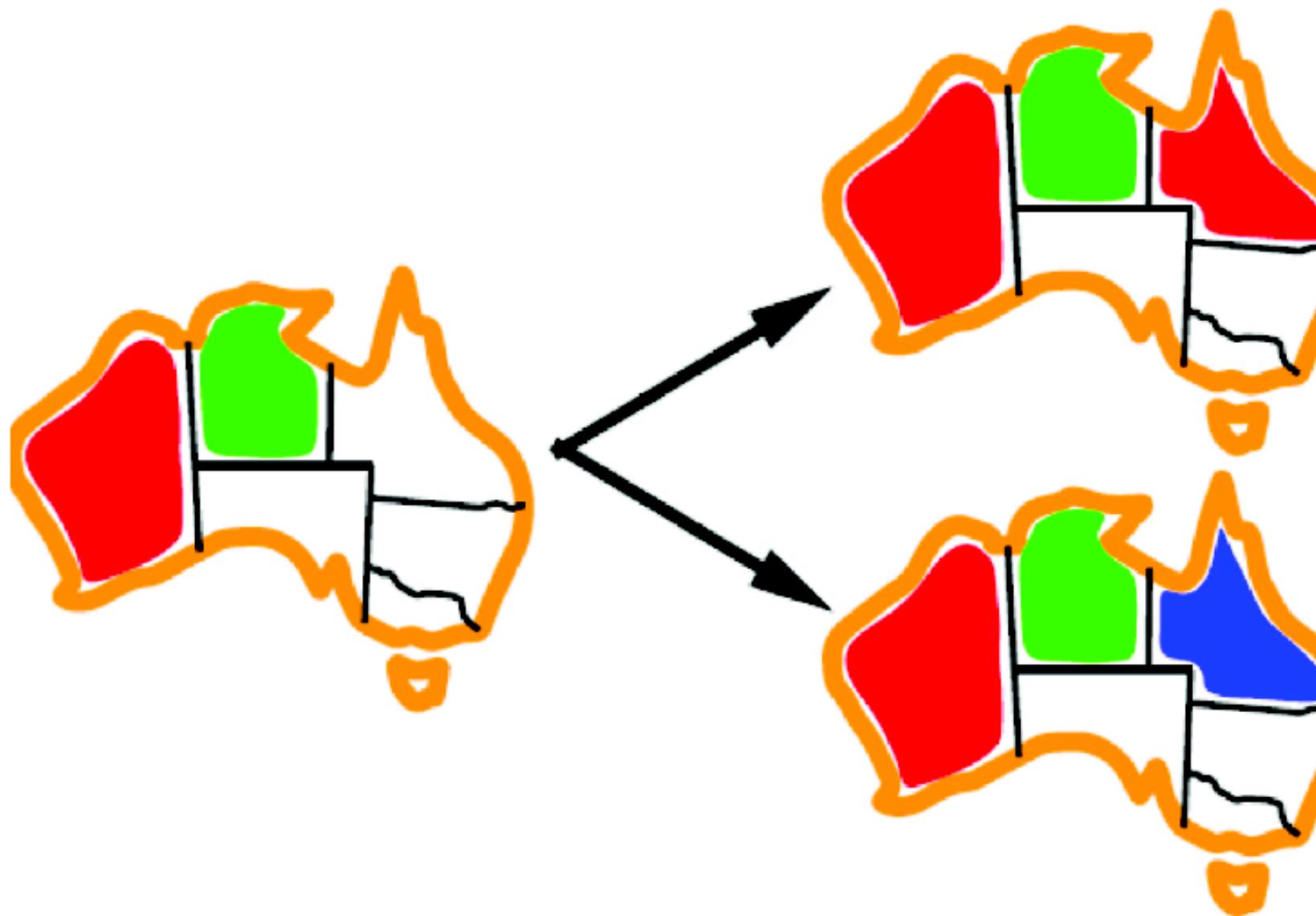
Minimum Remaining Value

Elegir la **variable** con la menor cantidad de valores posibles en su dominio



Least Constraining Value

Elegir el **valor** que presente mayores opciones en el futuro



Aplicaciones

Horarios y Planificación

- *Ejemplo:* Asignar horarios de clases a profesores y aulas sin que haya conflictos de asignaciones.
- *Aplicación:* Planificación académica, gestión de turnos en hospitales o empresas.



Aplicaciones

Resolución de Juegos de Lógica

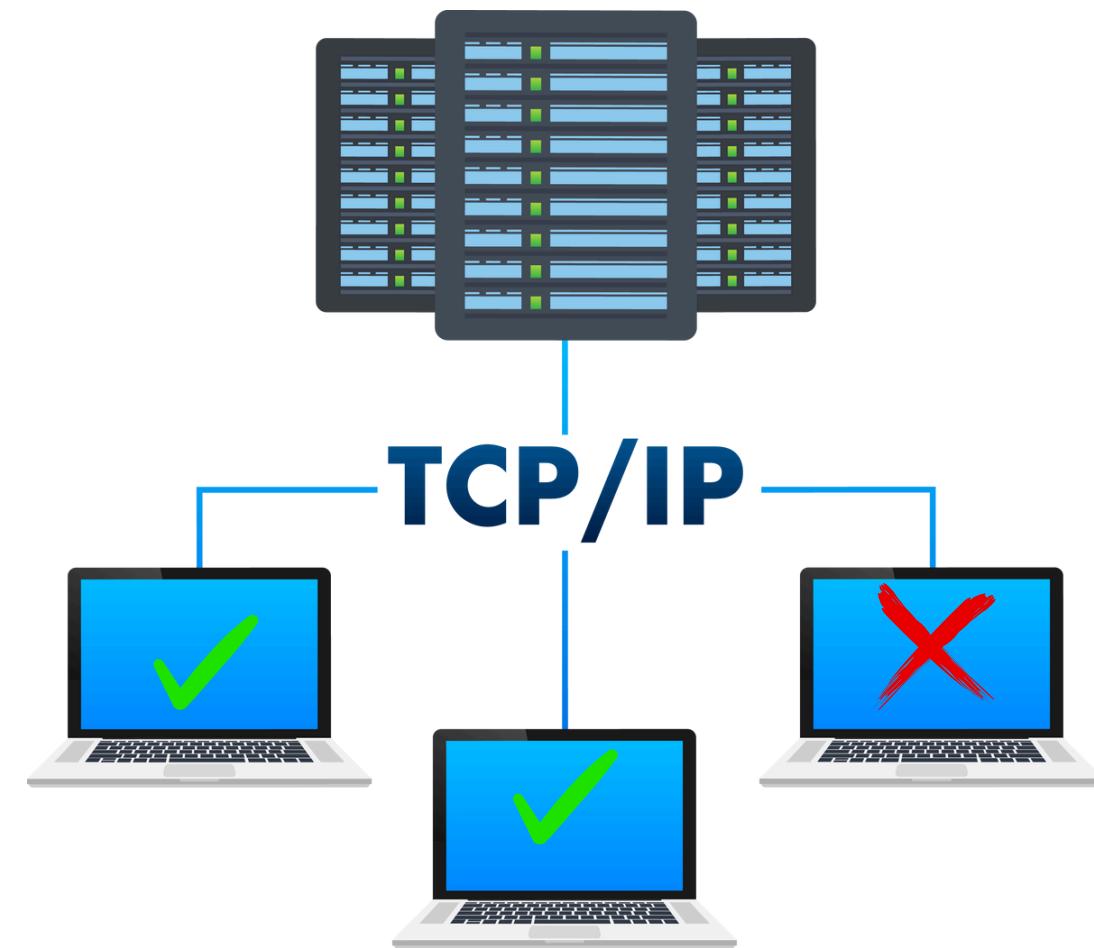
- *Ejemplo:* Llenar una cuadrícula de Sudoku cumpliendo restricciones de números en filas, columnas y subcuadrículas.
- *Aplicación:* Creación de videojuegos, optimización de tableros en juegos de estrategia.



Aplicaciones

Redes de Computadoras

- *Ejemplo:* Asignar direcciones IP y rutas en una red sin que haya conflictos.
- *Aplicación:* Optimización de redes de datos, seguridad en comunicación digital.



Aplicaciones

Logística y Transporte

- *Ejemplo:* Rutas de Entrega (Ej: Amazon, UPS).
- *Aplicación:* Optimizar rutas de reparto para minimizar tiempo y combustible, respetando ventanas de entrega y capacidad de vehículos.
- *Restricciones:* Tráfico, peso máximo, horarios de clientes



Resumen - Mapa Conceptual

