

INF623

2024/1



Inteligência Artificial

A9: Problemas de satisfação de restrição I

IA na mídia



03/04/2024

Google Vlogger

Modelo desenvolvido pela Google para animação de imagens com base em texto e imagem.

<https://enriccorona.github.io/vlogger/>

Plano de aula

- ▶ Problemas de satisfação de restrições
- ▶ Exemplos
- ▶ Tipos de restrições
- ▶ Grafo de restrições
- ▶ Busca com retrocesso

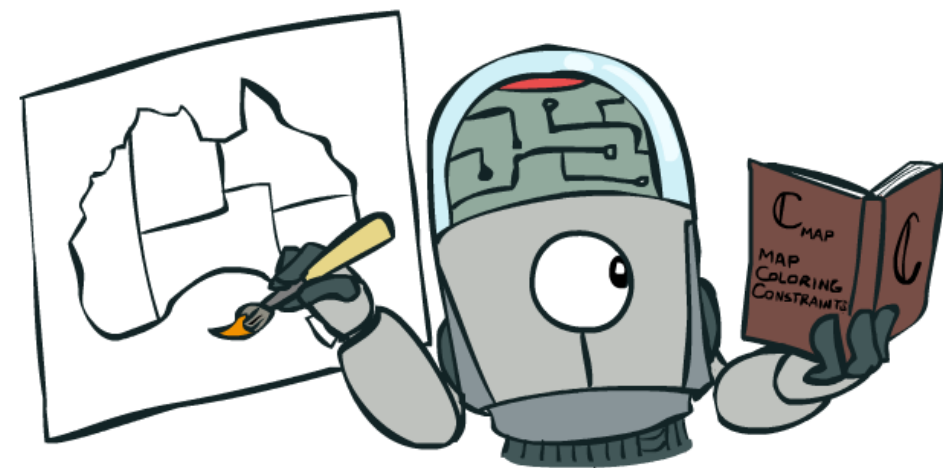
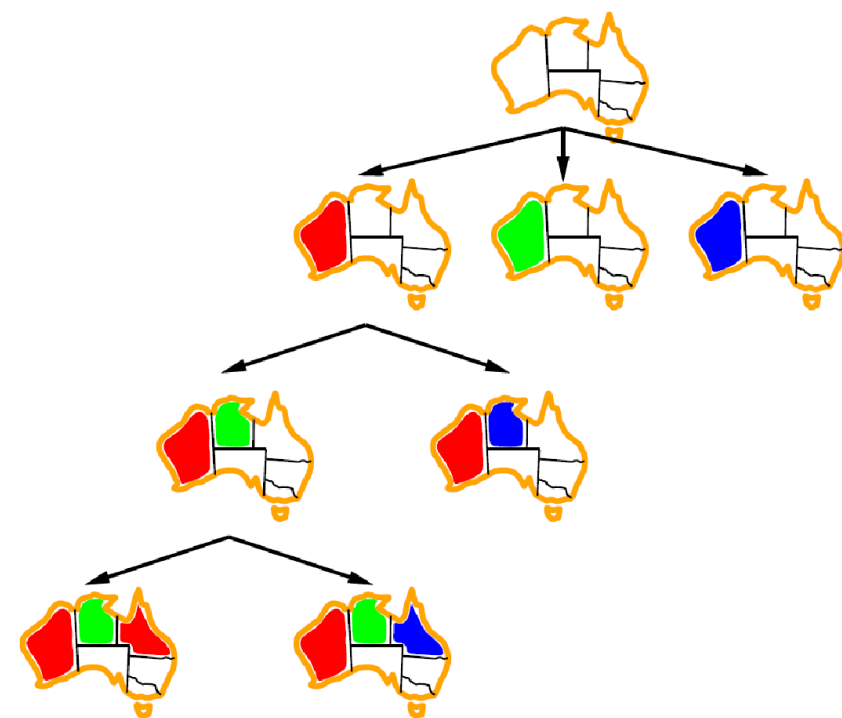
Exemplo 1: coloração de mapas

Considere o problema de colorir o mapa de um país com um número restrito de cores de tal forma que nenhum de seus estados adjacentes possuam a mesma cor.



Agentes racionais para jogos

Para resolver problemas desse tipo, chamados de **Problemas de Satisfação de Restrição (PSRs)**, um agente assume que o mundo é representado por um **espaço de estados fatorados** e que objetivo é encontrar uma solução que satisfaça um **conjunto de restrições**.



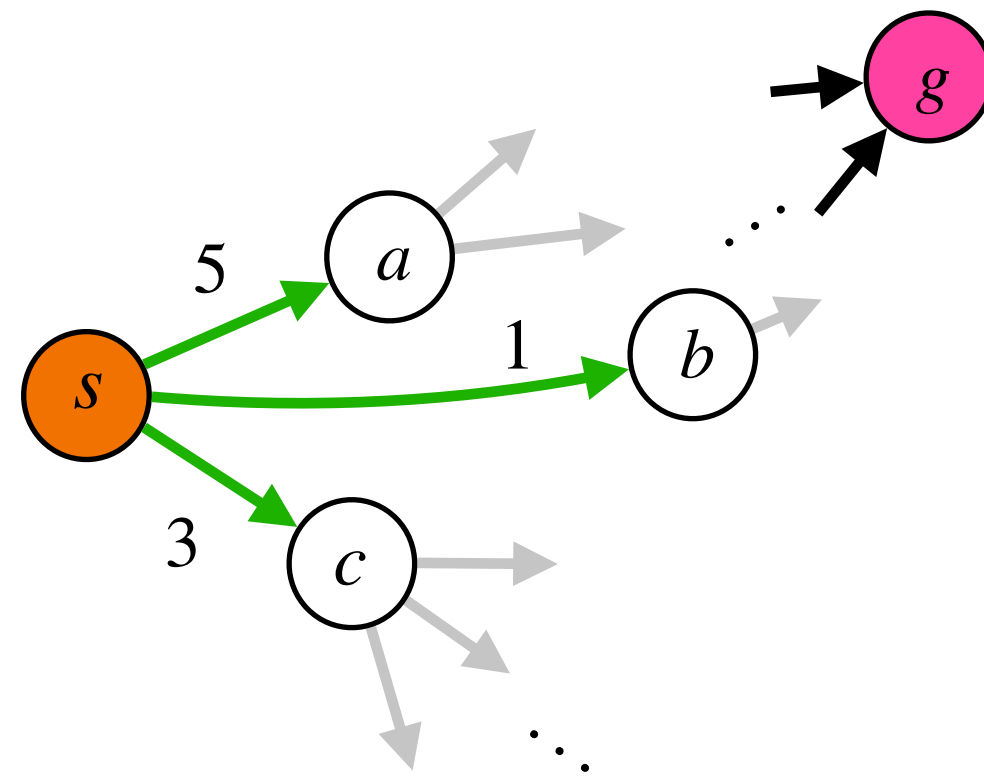
- ▶ **Estados** são **fatorados** em um conjunto de variáveis;
- ▶ Um **conjunto de restrições** especifica combinações possíveis de variáveis;
- ▶ Uma **solução** é uma atribuição de valores para todas as variáveis que satisfaça o conjunto de restrições

Problemas de satisfação de restrição

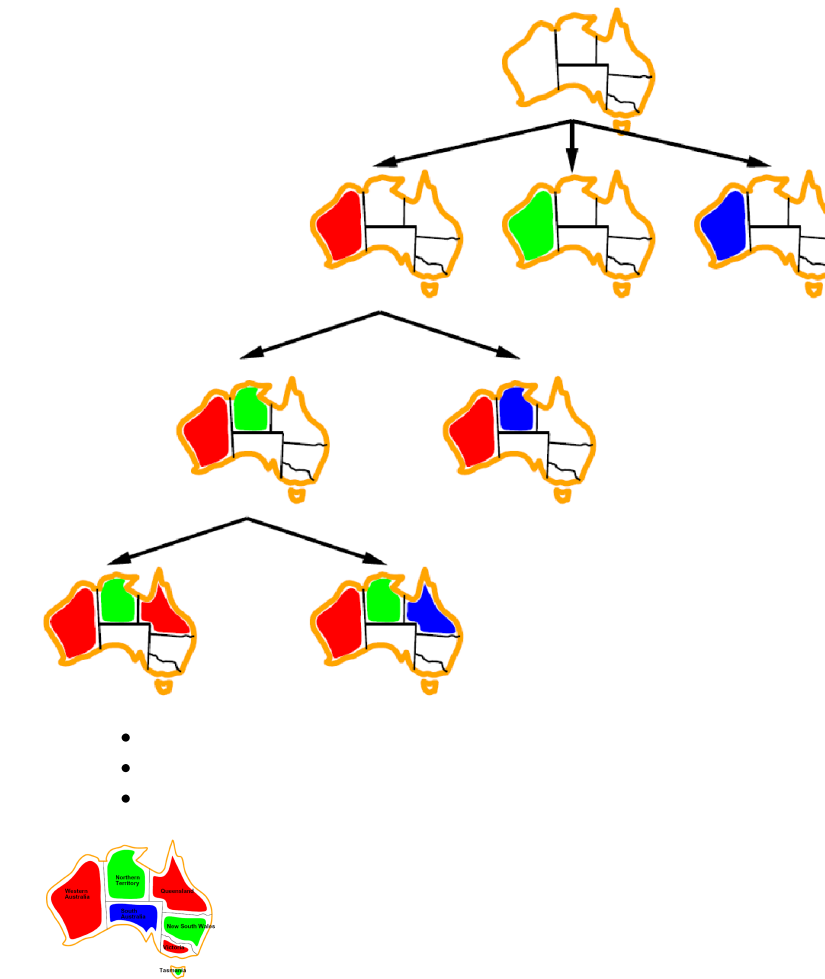
Um **problema de satisfação de restrição** pode ser definido pela tupla (X, D, C) , onde:

- ▶ X é um conjunto de **variáveis** $\{X_1, \dots, X_n\}$
- ▶ D é um conjunto de **domínios** $\{D_1, \dots, D_n\}$
 - ▶ Um domínio D_i é um conjunto de valores possíveis $\{v_1, \dots, v_k\}$ para X_i (e.g., variáveis booleanas – $\{V, F\}$)
 - ▶ Diferentes variáveis podem ter diferentes domínios com tamanhos diferentes
- ▶ C é um conjunto de **restrições** que especificam combinações de valores possíveis
 - ▶ Cada restrição C_j consiste em um par (Escopo, Relação), onde:
 - ▶ *Escopo* é uma tupla de variáveis que participam na restrição (e.g., (X_1, X_2));
 - ▶ *Relação* pode ser expressa:
 - ▶ *Explicitamente* – pelo conjunto de todas as tuplas de valores que satisfazem a restrição (e.g.,);
 - ▶ *Implicitamente* – por uma função que retorna se uma tupla faz parte da relação
- ▶ Uma solução é uma atribuição $\{X_i = v_i, X_j = v_j, \dots\}$ que seja **completa** e **consistente**
 - ▶ **Atribuição completa:** cada variável tem um valor atribuído
 - ▶ **Atribuição consistente:** não viola nenhuma restrição

Busca no espaço de estados vs. PSRs



PSRs são geralmente NP-completos, mas existem algumas subclasses de problemas que podem ser resolvidos de maneira eficiente

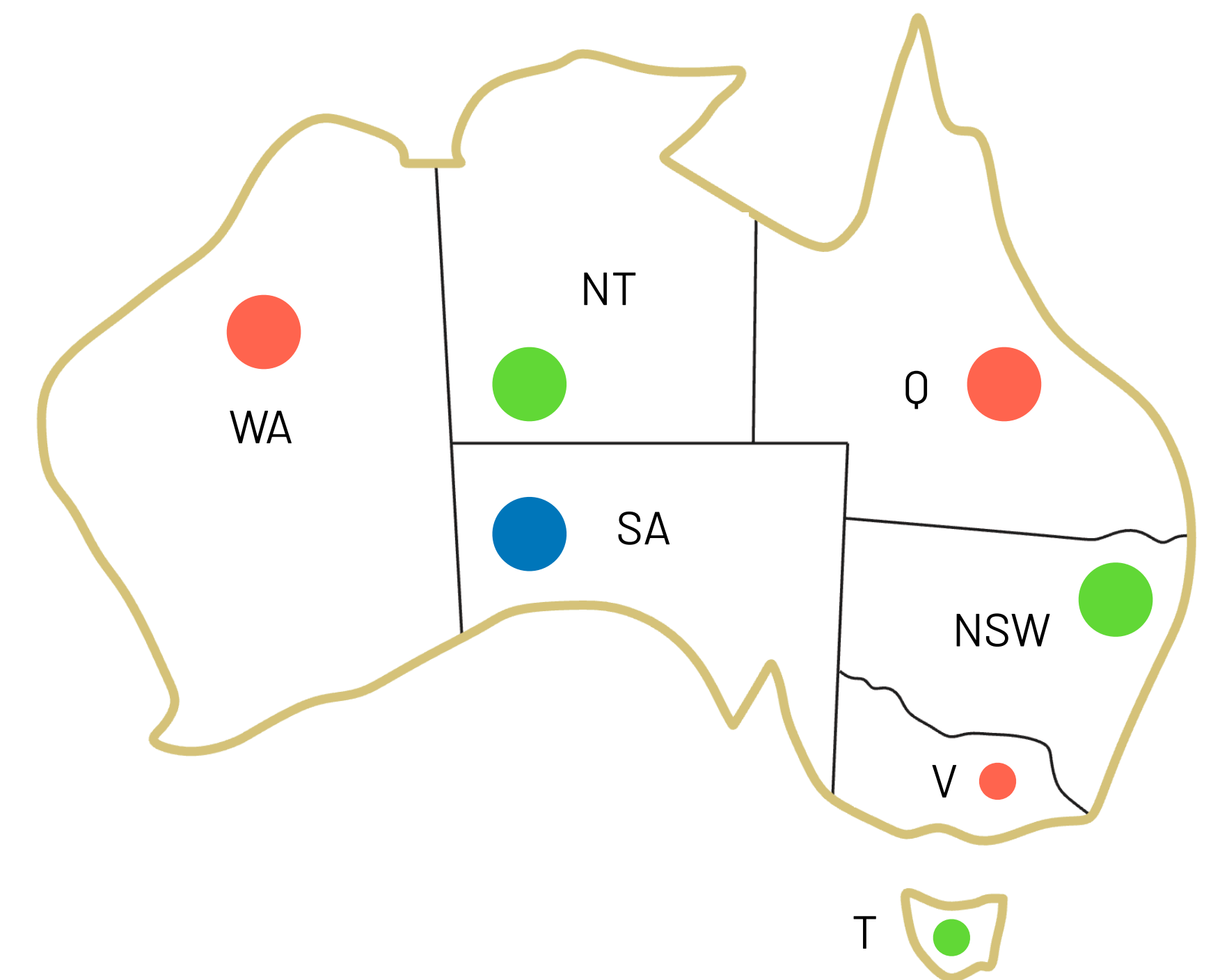


- ▶ Objetivo é encontrar um caminho entre s e g
- ▶ Busca no espaço de **estados**
- ▶ **Função de ações A:** dado um estado, gerar próximos estados
- ▶ Função heurística h específica do problema guia a busca em direção da solução

- ▶ Objetivo é encontrar g (uma atribuição de valores que satisfaça o conjunto de restrições)
- ▶ Busca no espaço de **estados fatorados** (atribuições)
- ▶ **Função de ações A:** atribuir um valor a uma variável
- ▶ A busca é guiada pela própria estrutura dos estados (abordagem mais geral)

Exemplo 1: coloração de mapas

- ▶ **Variáveis:** $X = \{WA, NT, Q, NSW, V, SA, T\}$
- ▶ **Domínios:** $D = \{\text{Vermelho, Verde, Azul}\}$
- ▶ **Restrições:** Regiões adjacentes devem ter cores diferentes
 - ▶ Explicitamente:
 - $(WA, NT) \in \{(\text{Vermelho, Verde}), (\text{Vermelho, Azul}), \dots\}$
 - $(WA, SA) \in \{(\text{Vermelho, Verde}), (\text{Vermelho, Azul}), \dots\}$
 - ...
 - ▶ Implicitamente:
 - $C = \{WA \neq NT, WA \neq SA, NT \neq Q, NT \neq SA,$
 $Q \neq SA, Q \neq NSW, NSW \neq SA, NSW \neq V, V \neq SA\}$
- ▶ **Exemplo de solução**
 - $\{WA = \text{Vermelho}, NT = \text{Verde}, Q = \text{Vermelho}, NSW = \text{Verde},$
 $V = \text{Vermelho}, SA = \text{Azul}, T = \text{Verde}\}$



Exemplo 2: Sudoku

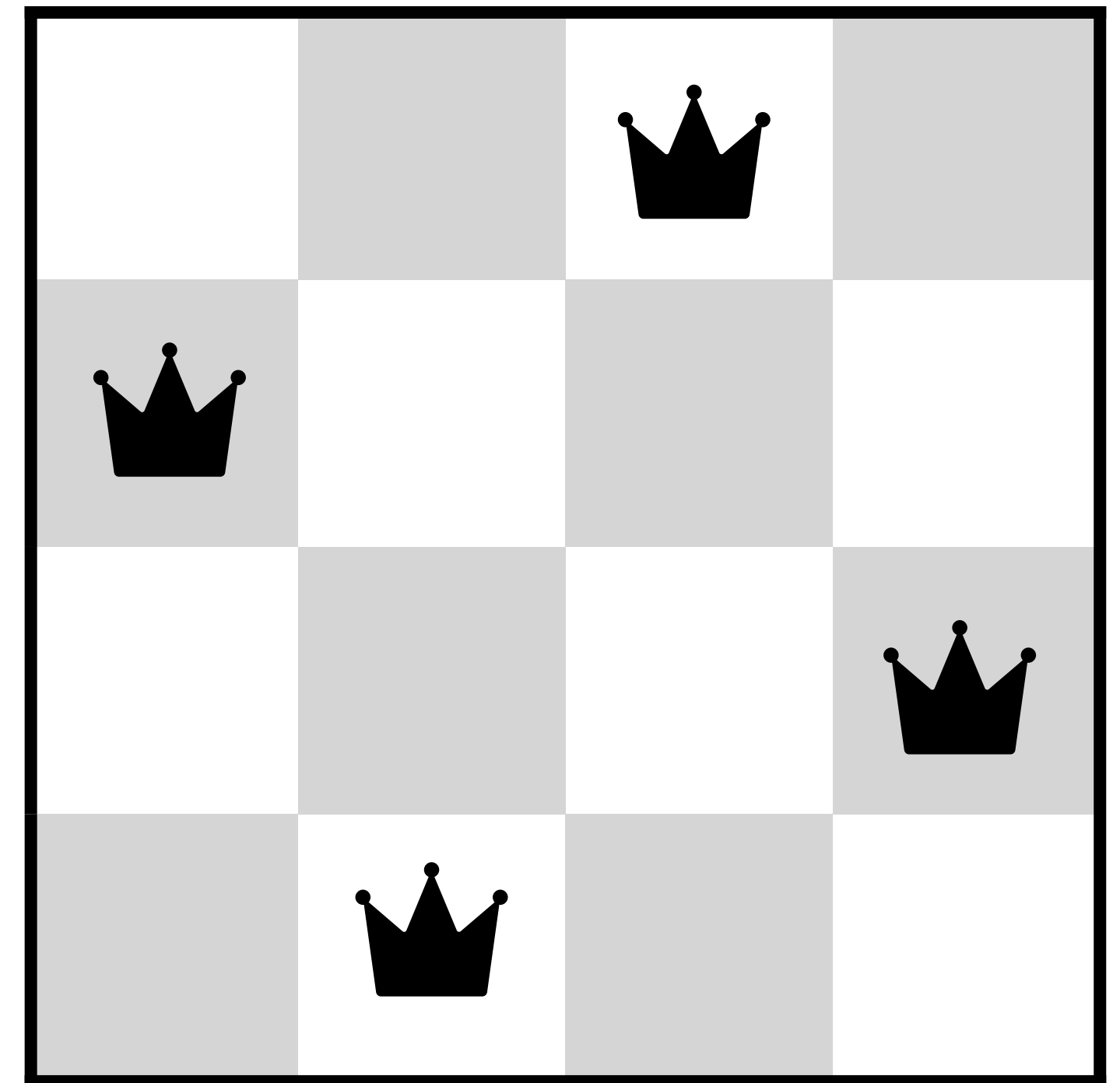
- ▶ **Variáveis:** Uma variável $X_{i,j}$ para cada célula da matriz.
- ▶ **Domínios:** $D = \{1,2,3,4,5,6,7,8,9\}$
- ▶ **Restrições (Implicitamente):**
 - ▶ **Linhas**
 - ▶ Todas as 9 variáveis diferentes para linha 1 ($\{X_{1,1}, X_{1,2}, \dots, X_{1,9}\}, \neq$)
 - ▶ Todas as 9 variáveis diferentes para linha 2 ($\{X_{2,1}, X_{2,2}, \dots, X_{2,9}\}, \neq$)
 - ▶ ...
 - ▶ **Colunas**
 - ▶ Todas as 9 variáveis diferentes para coluna 1 ($\{X_{1,1}, X_{2,1}, \dots, X_{9,1}\}, \neq$)
 - ▶ Todas as 9 variáveis diferentes para coluna 2 ($\{X_{1,2}, X_{2,2}, \dots, X_{9,2}\}, \neq$)
 - ▶ ...
 - ▶ **Quadrantes**
 - ▶ Todas as 9 variáveis diferentes do 1º quadrante:
($\{X_{1,1}, X_{1,2}, X_{1,3}, X_{2,1}, X_{2,2}, X_{2,3}, X_{3,1}, X_{3,2}, X_{3,3}\}, \neq$)
 - ▶ Todas as 9 variáveis diferentes do 2º quadrante:
($\{X_{1,4}, X_{1,5}, X_{1,6}, X_{2,4}, X_{2,5}, X_{2,6}, X_{3,4}, X_{3,5}, X_{3,6}\}, \neq$)
 - ▶ ...

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

Exercício: 4-rainhas

Defina o problema das 4-rainhas como um PSR.

- ▶ **Variáveis:** _____
- ▶ **Domínios:** _____
- ▶ **Restrições:** _____
- ▶ **Exemplo de solução:** _____



Exercício: 4-rainhas

Formalização 1:

► **Variáveis:** Uma variável $X_{i,j}$ para cada posição do tabuleiro.

► **Domínios:** $D = \{0,1\}$

► **Restrições:**

$$\forall i, j, k (X_{ij}, X_{ik}) \in \{(0,0), (0,1), (1,0)\}$$

$$\forall i, j, k (X_{ij}, X_{kj}) \in \{(0,0), (0,1), (1,0)\}$$

$$\forall i, j, k (X_{ij}, X_{i+k,j+k}) \in \{(0,0), (0,1), (1,0)\}$$

$$\forall i, j, k (X_{ij}, X_{i+k,j-k}) \in \{(0,0), (0,1), (1,0)\}$$

$$\sum X_{ij} = N$$

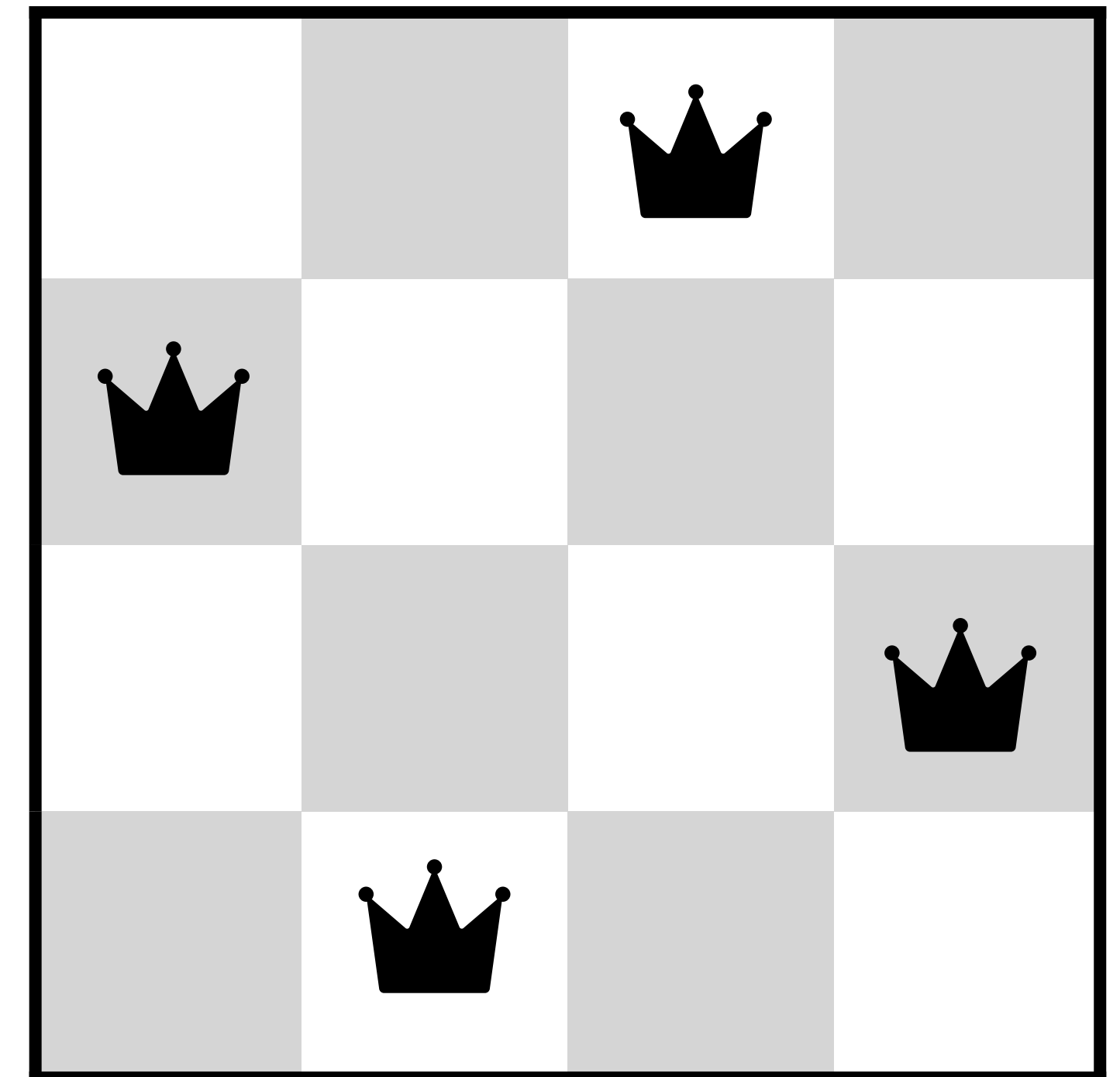
► **Exemplo de solução:**

$$\{X_{1,1} = 0, X_{1,2} = 0, X_{1,3} = 1, X_{1,4} = 0$$

$$X_{2,1} = 1, X_{2,2} = 0, X_{2,3} = 1, X_{2,4} = 0$$

$$X_{3,1} = 0, X_{3,2} = 0, X_{3,3} = 1, X_{3,4} = 1$$

$$X_{4,1} = 0, X_{4,2} = 1, X_{4,3} = 1, X_{4,4} = 0\}$$



Exercício: 4-rainhas

Formalização 2:

▶ **Variáveis:** Uma variável Q_i para cada coluna do tabuleiro.

▶ **Domínios:** $D = \{1,2,3,4\}$

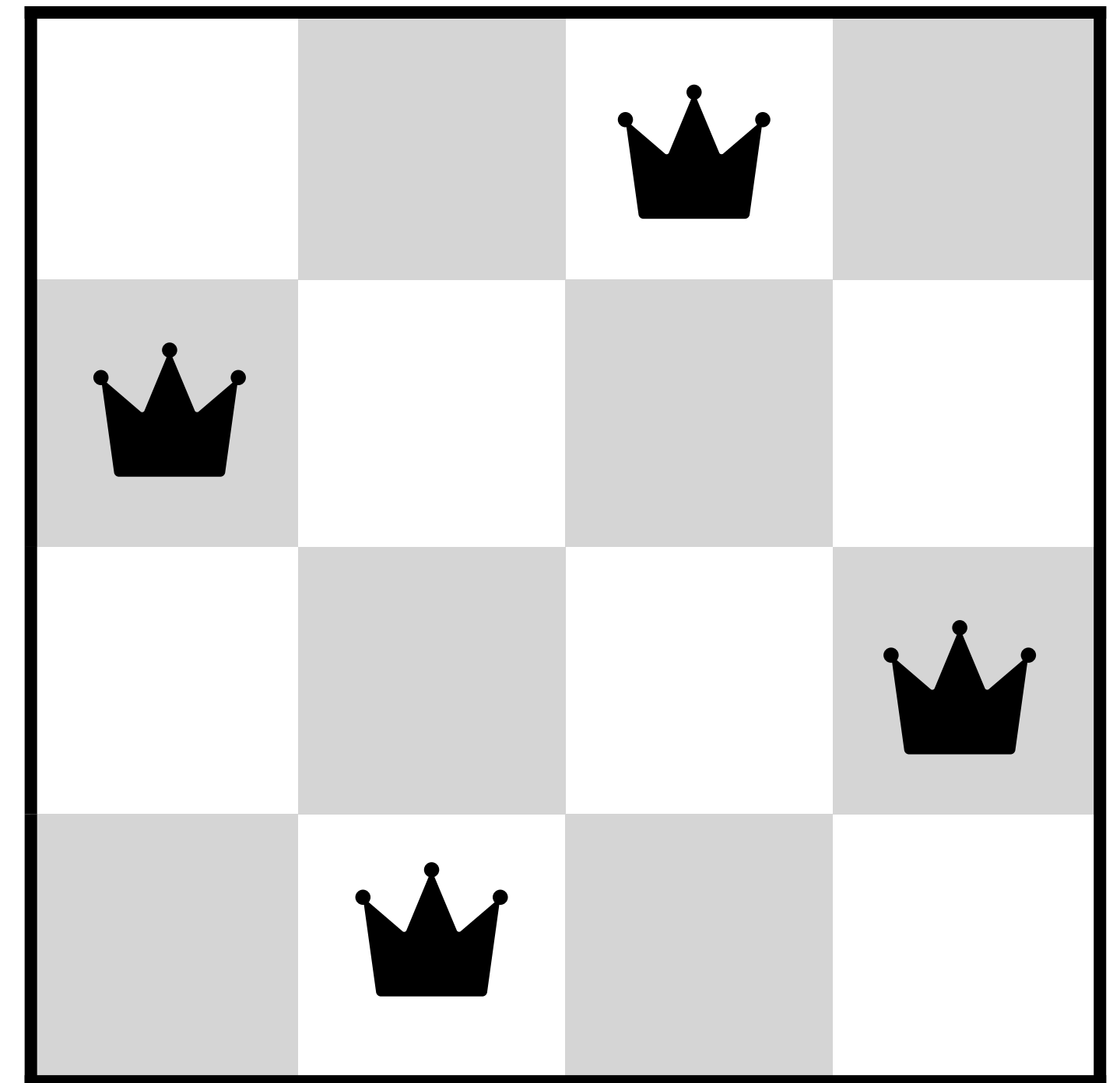
▶ **Restrições:**

Implicitamente: $\forall_{i,j} \text{ nao-se-atacam}(Q_i, Q_j)$

Explicitamente: $(Q_1, Q_2) \in \{(1,3), (1,4), \dots\}$
...

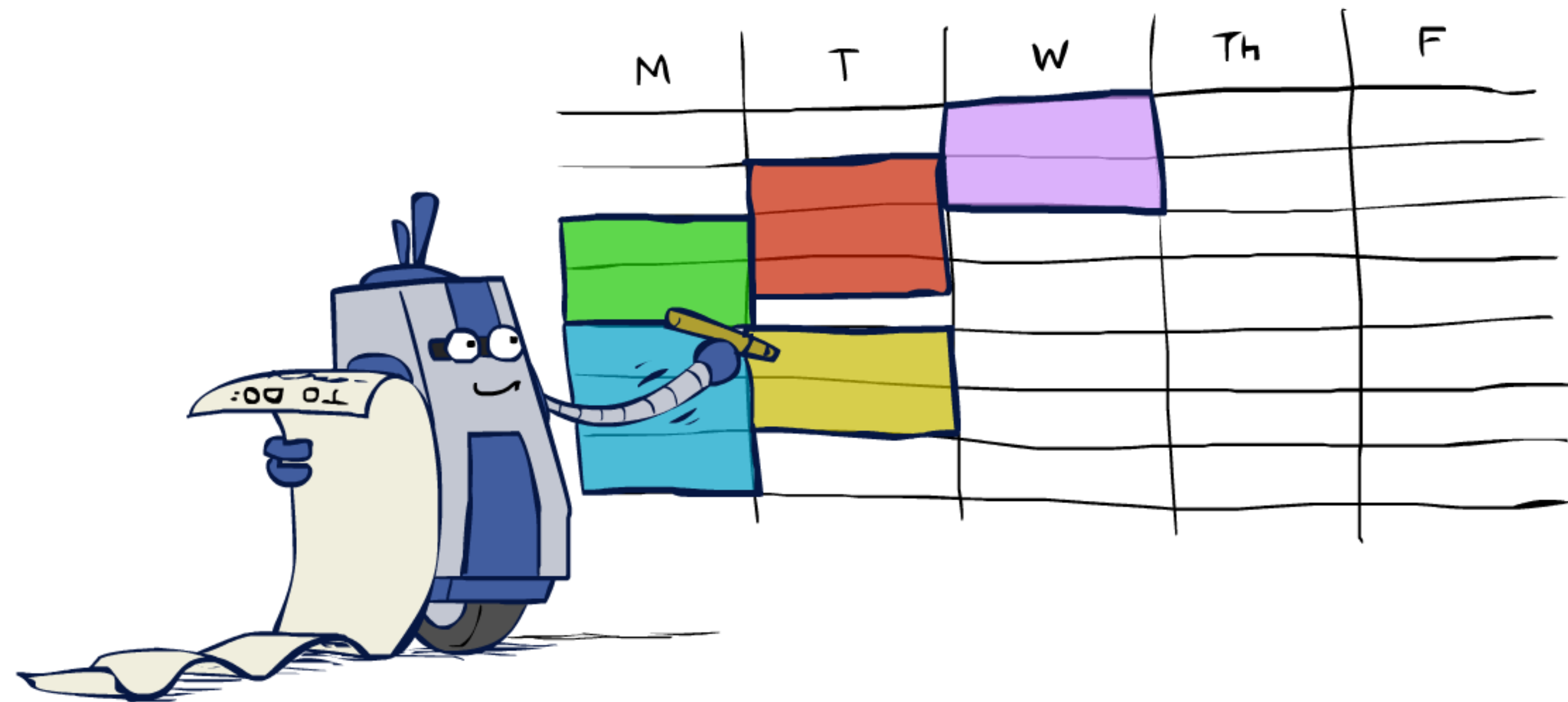
▶ **Exemplo de solução:**

$\{Q_1 = 2, X_2 = 4, X_3 = 1, X_4 = 3\}$



Aplicações

- ▶ Agendamento: quando podemos nos encontrar?
- ▶ Alocar horários: que aulas são oferecidas, quando e onde?
- ▶ Atribuição: quem ensina qual aula?
- ▶ Configuração de hardware
- ▶ Agendamento de transportes
- ▶ Automação de fábricas
- ▶ Layout de circuito
- ▶ Diagnóstico de erro
- ▶ ... muito mais!



Tipos de restrições

- ▶ **Unárias:** restringem uma variável

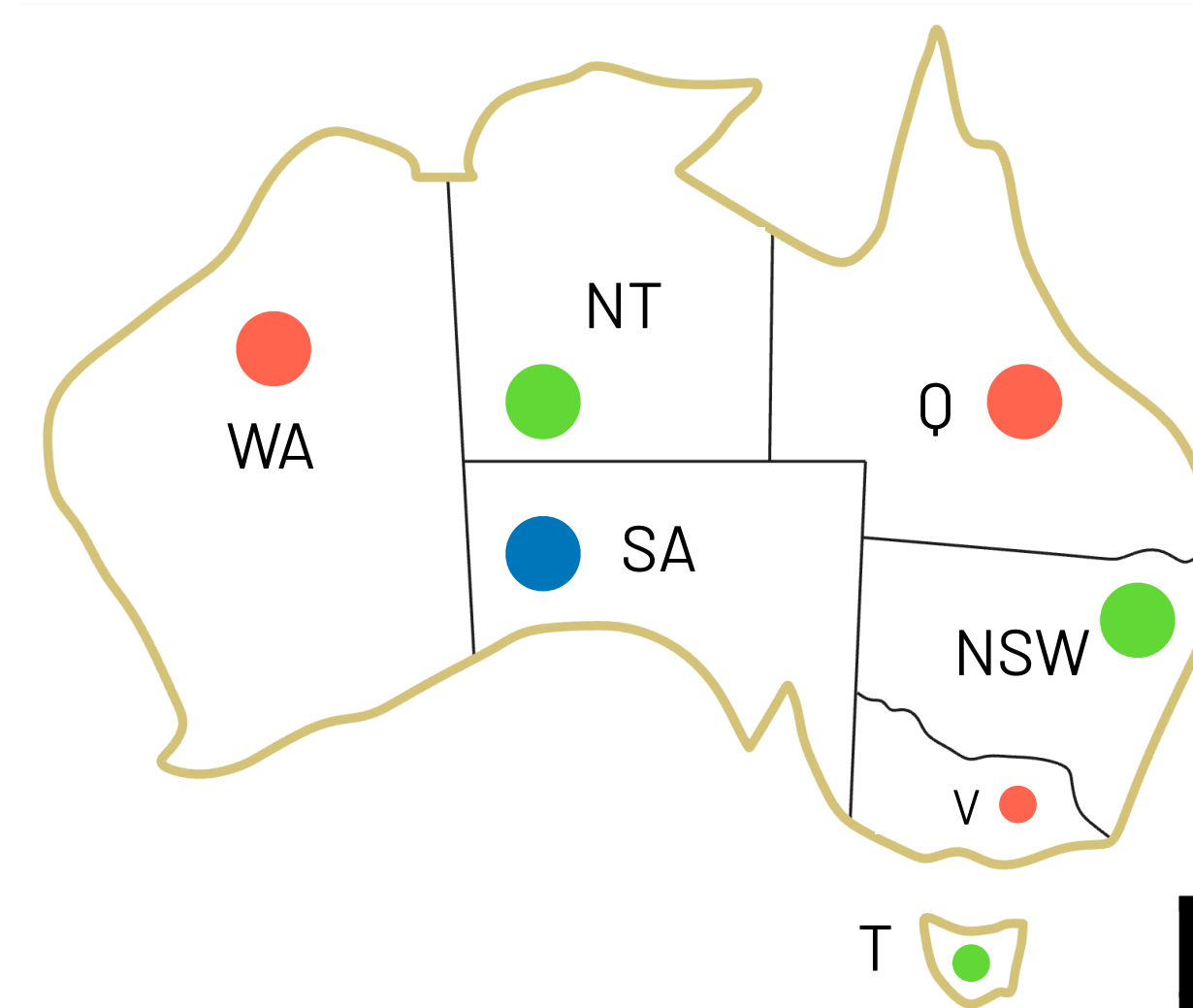
Exemplo: WA não pode ser colorido de Verde

- ▶ **Binárias:** restringem duas variáveis

Exemplo: estados adjacentes não podem ter a mesma cor

- ▶ **Globais:** restringem mais de duas variáveis

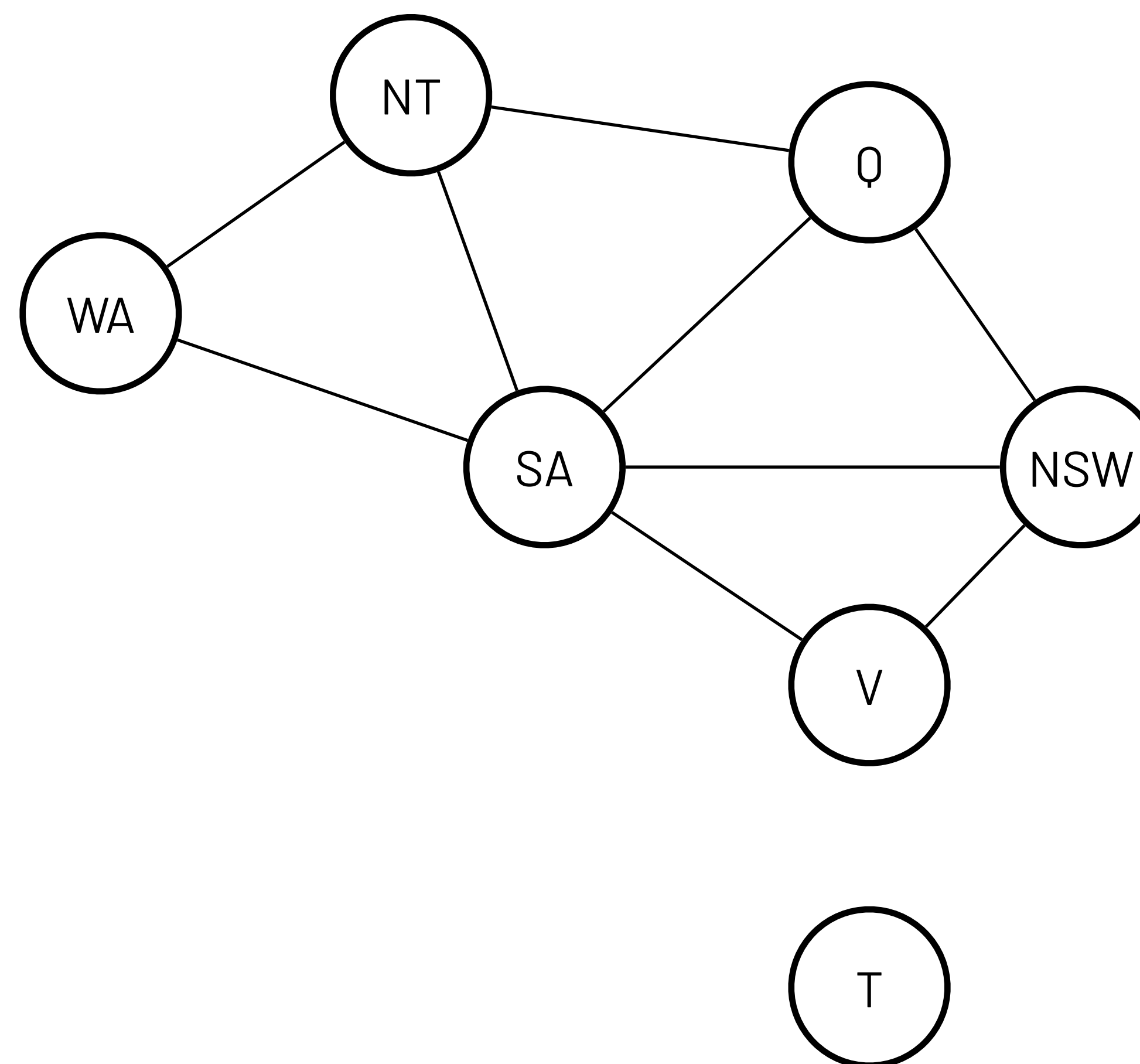
Exemplo: todas as células de uma linha do Sudoku



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

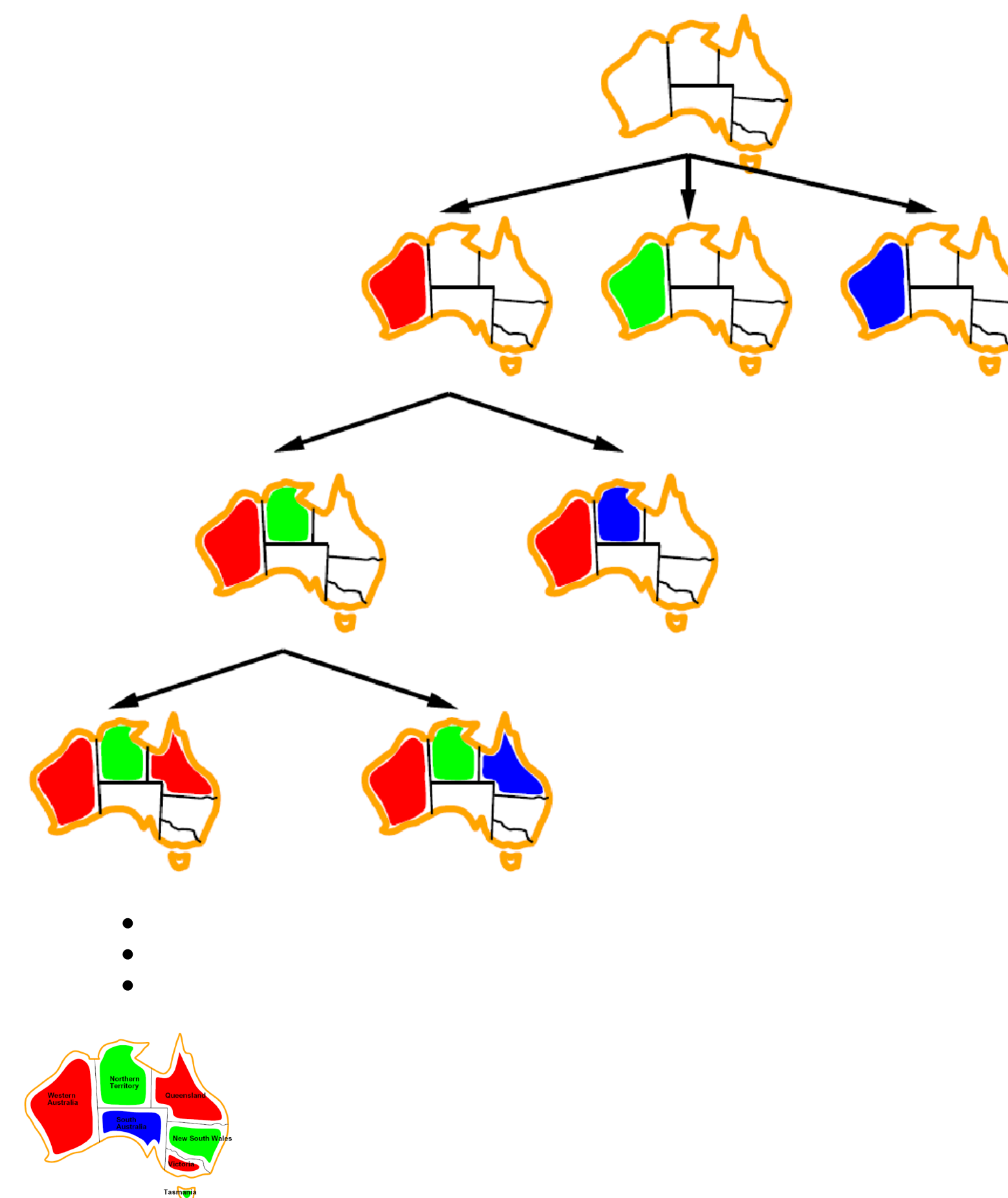
Grafo de restrições

- ▶ **Vértices** são variáveis
- ▶ **Arestas** representam restrições entre variáveis
 - ▶ Não especificam o tipo de restrição, mas apenas que ela existe
- ▶ Algoritmos de PSRs usam a estrutura do grafo de restrições para guiar a busca.
 - ▶ Exemplo: Tasmania é um problema independente!



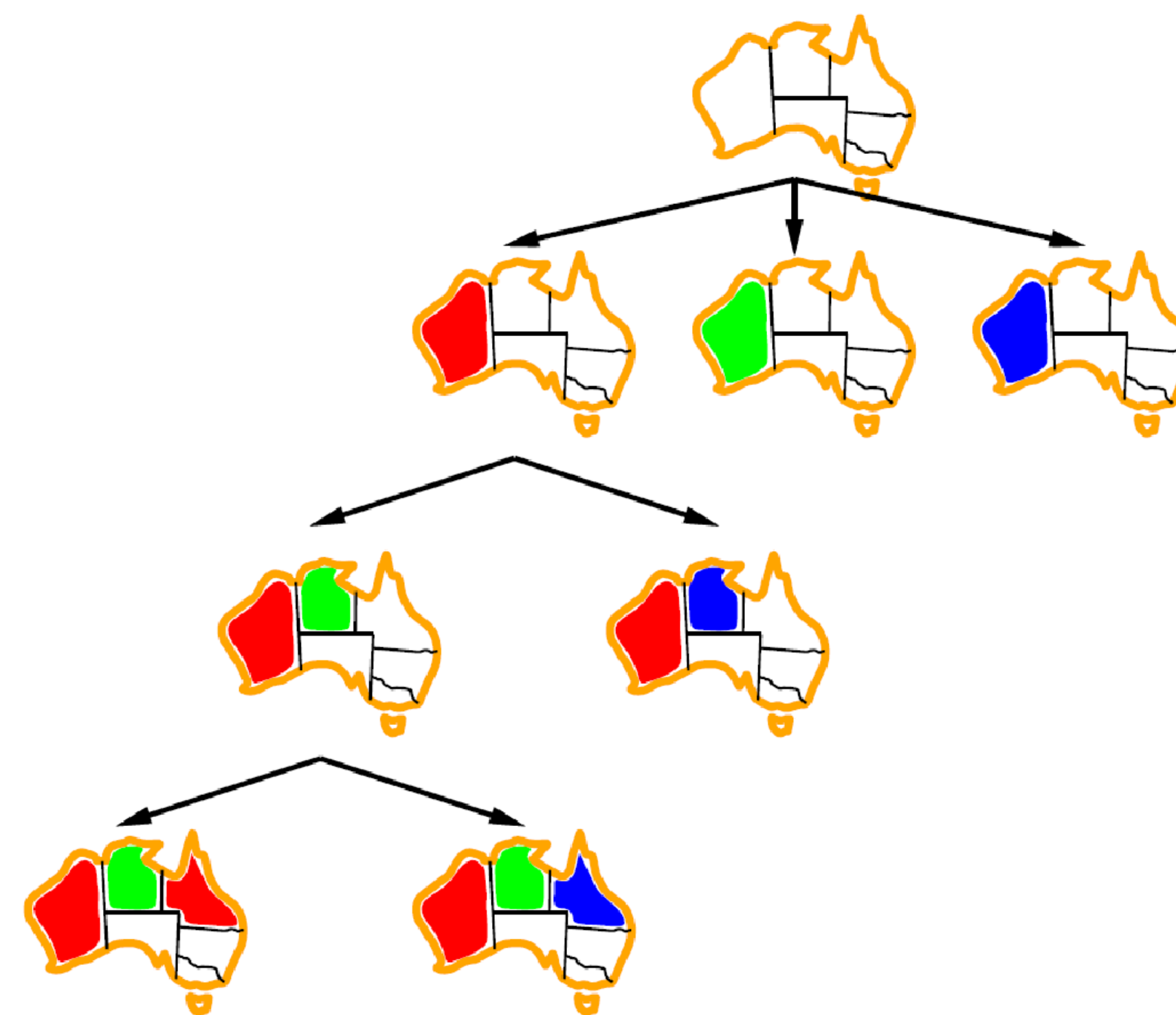
Resolvendo PSRs

- ▶ Formulação de busca para PSRs:
 - ▶ Estados são definidos por **atribuições parciais**:
 - ▶ **Estado Inicial**: atribuição vazia { }
 - ▶ **Função de Ação**: atribuir um valor para uma variável aberta
 - ▶ **Estado final**: a atribuição atual é **completa** a **consistente**
 - ▶ Como seria o desempenho da BFS?
 - ▶ Todas as soluções estão nas folhas! A BFS exploraria todos os níveis sistematicamente antes de encontrar uma solução



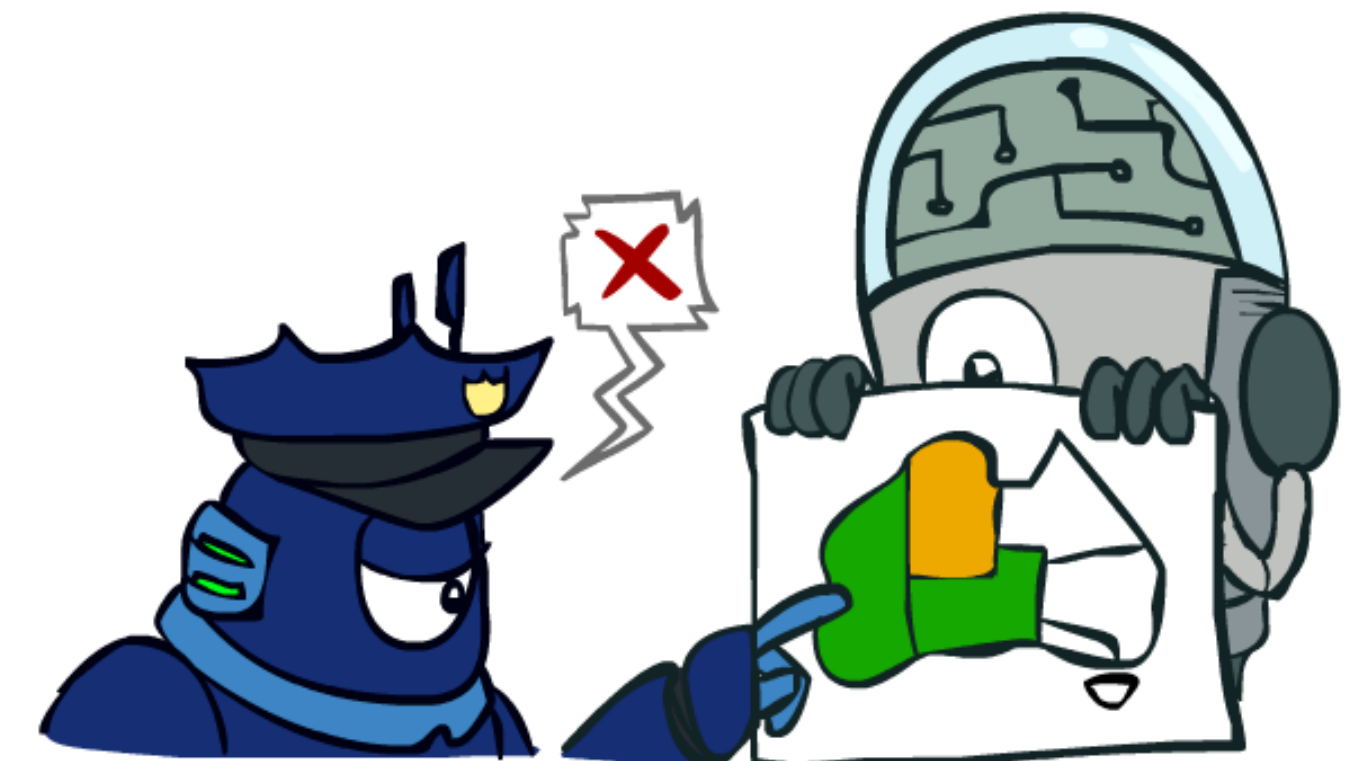
Busca com retrocesso para PSRs: ideia geral

- ▶ **Busca com retrocesso** (*backtracking search*) é o algoritmo sem informação básico para resolver PSRs
- ▶ Modificar **busca em profundidade (DFS)** adicionando duas ideias:
 - ▶ **Ideia 1: Modificar uma única variável por vez**
 - ▶ Atribuição de variáveis é comutativa
 - ▶ Definir uma ordem para as variáveis
 - ▶ **Ideia 2: Verifique restrições a cada expansão**
 - ▶ Se chegarmos em um estado que não satisfaz as restrições, podemos retroceder!
 - ▶ Verificar restrições pode envolver computação não-constante
- ▶ Esse algoritmo consegue resolver o N-rainhas com $N \approx 25$



Busca com retrocesso para PSRs

```
def busca-retrocesso(X, D, C):  
1.     return retrocesso-recursivo({}, X, D, C)  
  
def retrocesso-recursivo(assign, X, D, C):  
1.     if is-complete(assign):  
2.         return assign  
3.     var = select-unsassigned-variable(X, assign) # Ideia 1: ordenar variáveis  
4.     for value in sort(var, assign, X, D):  
5.         if is-consistent(assign, value, C): # Ideia 2: verificar restrições  
6.             assign[var] = value  
7.             result = retrocesso-recursivo(assign, X, D, C)  
8.             if result != None:  
9.                 return result  
10.            del assign[var]  
11.    return None
```



Próxima aula

A10: Problemas de satisfação de restrição II

Ordenação de variáveis e valores, inferência em PSRs e estruturas de problemas