

# Kecerdasan Buatan/ Artificial Intelligence

## Constraint Satisfaction Problem (CSP)

Rekyan Regasari Mardi Putri, ST, MT

Lailil Muflikhah, S.Kom, M.Sc

**Imam Cholissodin, S.Si., M.Kom**

M. Ali Fauzi, S.Kom, M.Kom

# Pokok Bahasan

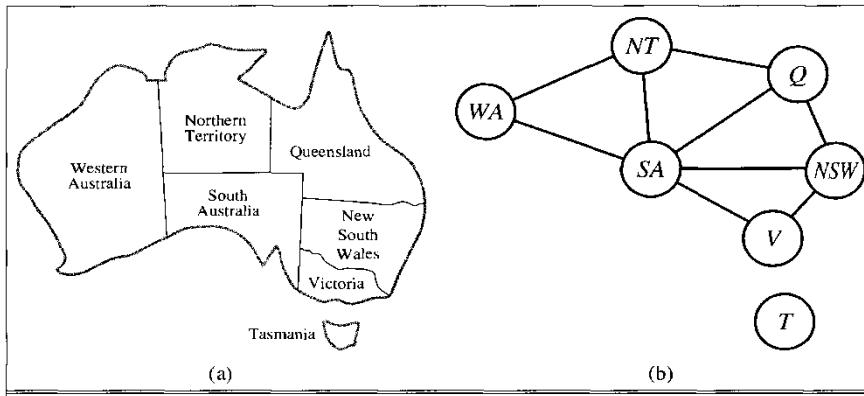
1. Constraint Satisfaction Problems (CSP)
2. Pencarian Backtracking untuk CSP
3. Local search untuk CSPs
4. Latihan Individu + Tugas Kelompok
5. Quiz 1

# Constraint Satisfaction Problem

- **CSP** atau Constraint Satisfaction Problem adalah permasalahan yang tujuannya adalah mendapatkan suatu kombinasi variabel-variabel tertentu yang memenuhi aturan-aturan (*constraints*) tertentu.
- **State** didefinisikan dengan *variables*  $X_i$  yang mempunyai *values* dari domain  $D_i$ .
- **Goal Test** adalah sebuah himpunan *constraints* yang memberikan kombinasi yang diijinkan untuk mengisi variabel.

# Contoh CSP

## □ Mewarnai Peta

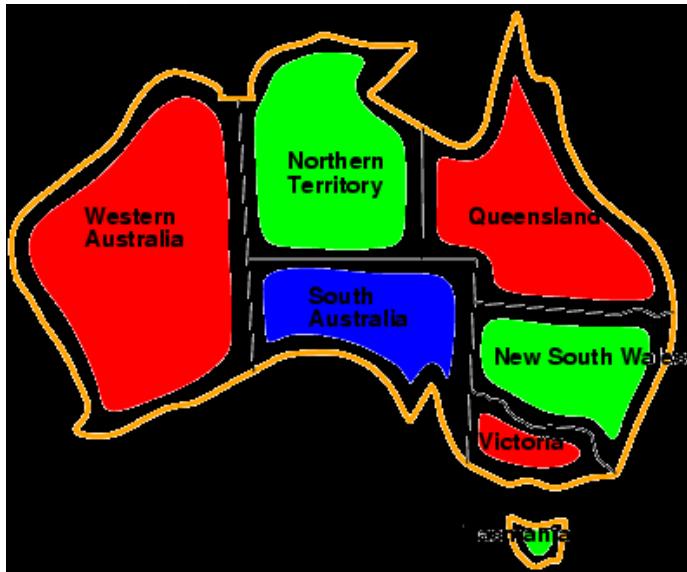


**Figure 5.1** (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem. The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

- Variabel: WA, NT, Q, NSW, V, SA, T
- Ranah:  $D_i = \{\text{red, green, blue}\}$
- Syarat: 2 wilayah yang berbatasan harus berbeda warna:
  - $WA \neq NT, NT \neq SA, \dots$
  - $(WA, NT) \in \{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), \dots\}$

# Contoh CSP

## □ Contoh Solusi CSP : Mewarnai Peta

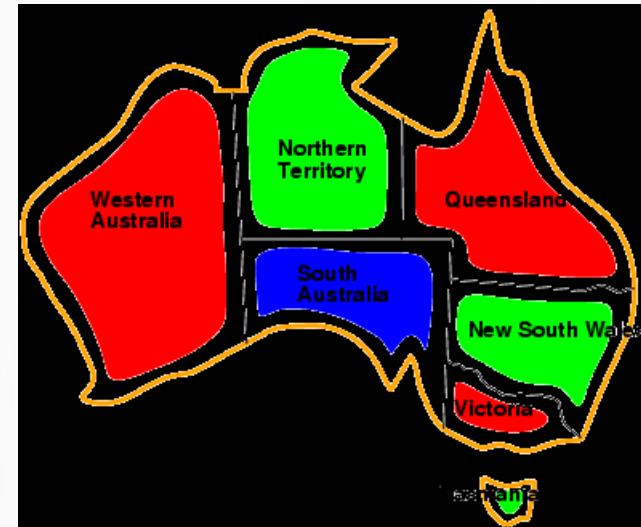
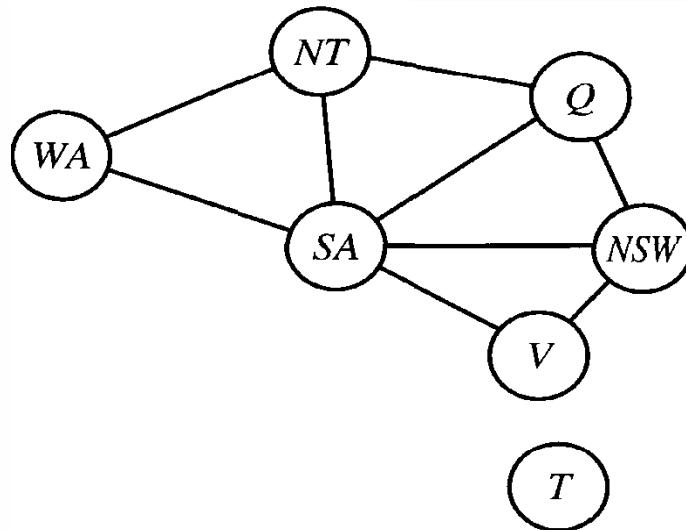


- Solusi adalah pemberian nilai setiap variabel yang memenuhi syarat, mis : {WA=*red*, NT=*green*, Q=*red*, NSW=*green*, V=*red*, SA=*blue*, T=*green*}

# Contoh CSP

## □ Constraint graph

- **Binary** CSP : sebuah constraint menyangkut hubungan maks. 2 variable.
- **Constraint graph** : representasi di mana node adalah variable, edge adalah constraint, mis:



# Backtracking

- ❑ Algoritma backtracking search (penelusuran kembali) adalah suatu bentuk algoritma depth-first-search.
- ❑ Jika solusi partial melanggar constraint, backtracking melakukan langkah kembali ke solusi partial sebelumnya.
- ❑ Pada algoritma backtracking, teknik look ahead digunakan untuk meramalkan efek pemilihan variabel branching untuk mengevaluasi nilai-nilai variabel tersebut.
- ❑ Forward checking adalah salah satu cara untuk melakukan look ahead. Forward checking mencegah terjadinya konflik dengan melarang nilai-nilai yang belum diisikan ke variable untuk dipakai.
- ❑ Ketika suatu nilai diisikan ke suatu variabel, nilai yang berada di domain dari variabel yang konflik tersebut akan dihilangkan dari domain.

# Minimum Remaining Value

- ❑ Adalah suatu teknik yang dikembangkan untuk menangani masalah kemungkinan besar gagal pada pencarian menggunakan CSP.
- ❑ MRV berkerja dengan memilih variabel yang memiliki domain legal dan paling sedikit (memiliki kemungkinan untuk membuat suatu dead-end paling besar) untuk diisikan terlebih dulu.

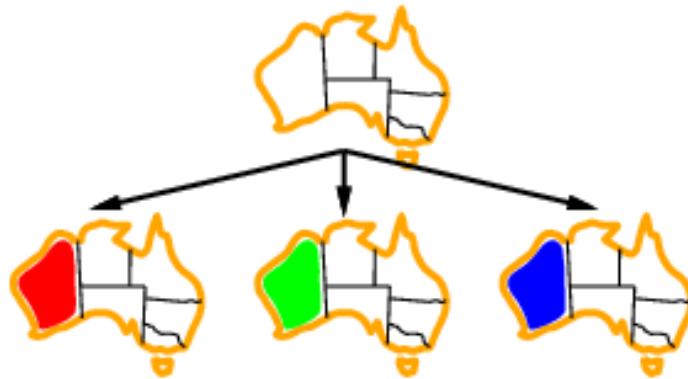
# Backtracking Search

- ❑ Variable assignment berlaku komutatif, dalam arti: [WA=red lalu NT=green] sama saja [NT=green lalu WA=red]
- ❑ Pada tiap level, hanya perlu meng-assign satu variabel  $b = d$
- ❑ Depth first search pada CSP dengan assignment satu variabel tiap level disebut backtracking search.

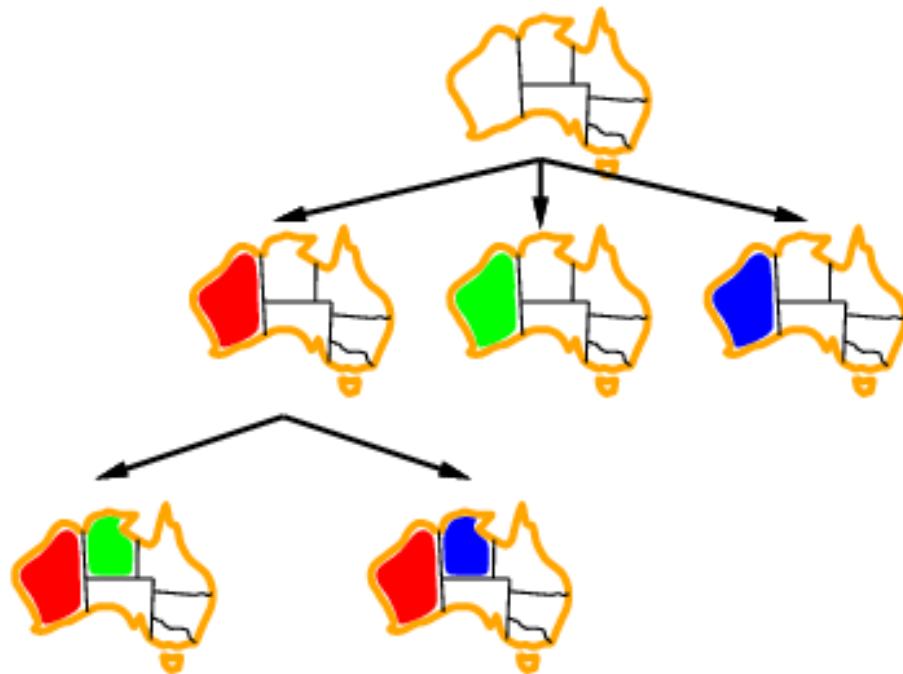
# Backtracking example



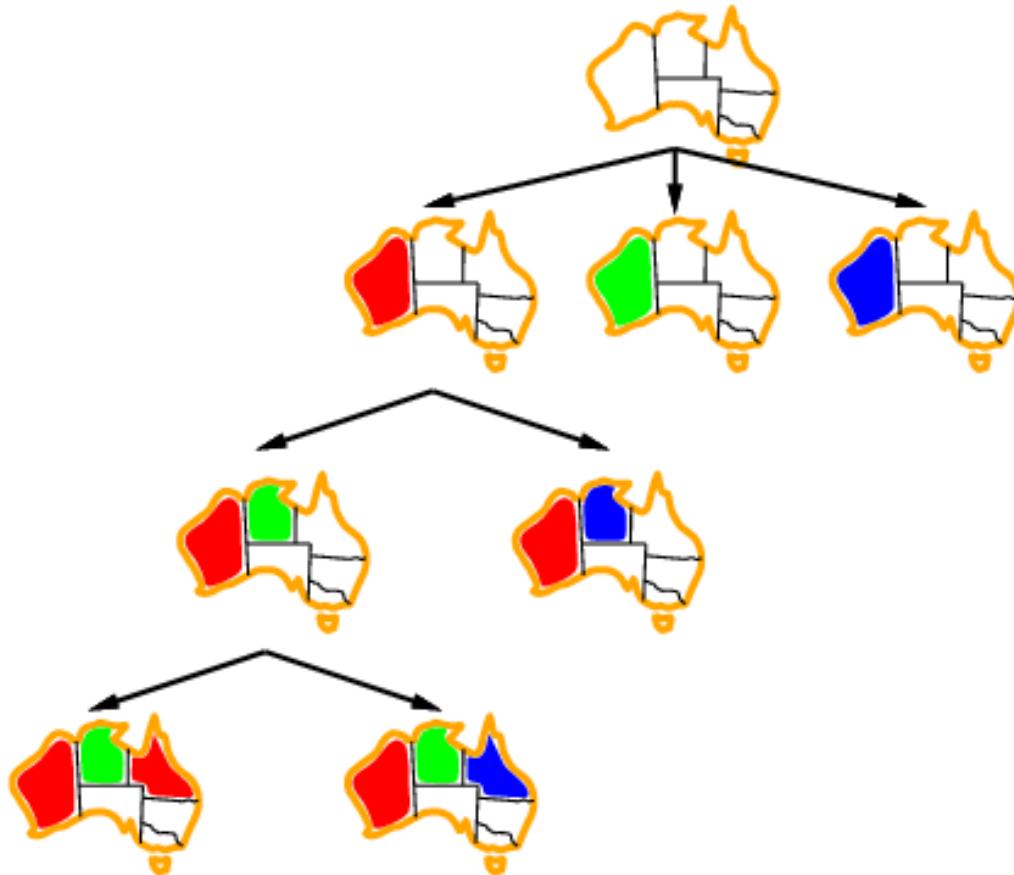
# Backtracking example



# Backtracking example



# Backtracking example

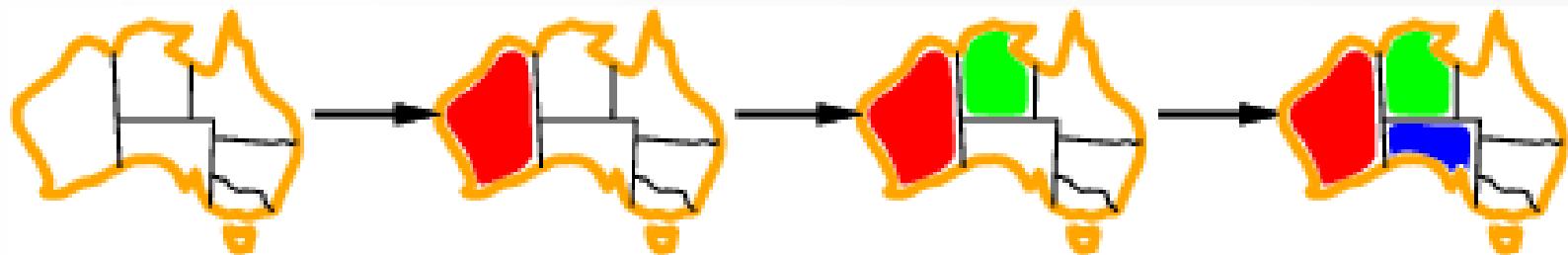


# Memperbaiki kinerja backtracking

- Urutan pemilihan variable dan nilai mempengaruhi kinerja backtracking
- Terdapat beberapa strategi yang berlaku secara umum (general-purpose) :
  1. Variable mana yang perlu di-assign terlebih dulu?
  2. Nilai apakah yang perlu dicoba terlebih dulu?
  3. Apakah kita bisa mendeteksi kepastian failure lebih awal?
  4. Apakah kita bisa memanfaatkan struktur masalah?
  5. CSP? (Representasinya jelas!)

# Prinsip 1: Most Constrained Variable

- Variabel yang paling dibatasi
  - Pilih variable yang memiliki kemungkinan nilai sah paling sedikit

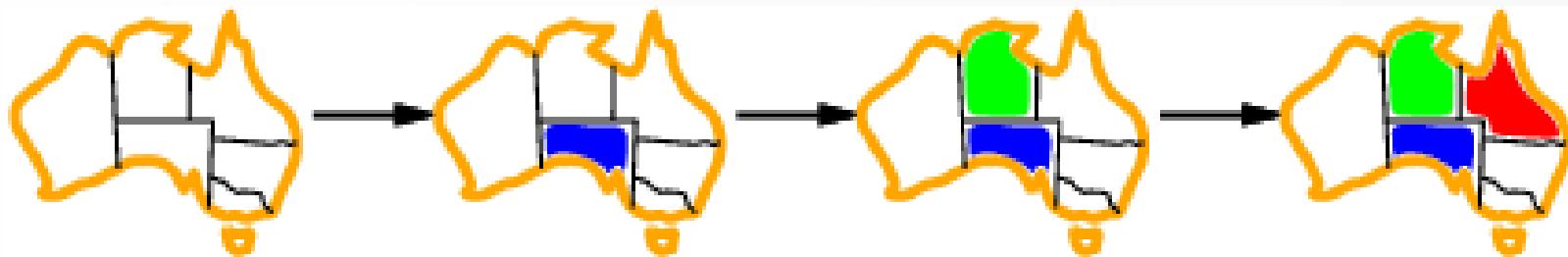


- a.k.a. minimum remaining values (MRV) heuristic  
also called most constrained variable or “fail first”

# Prinsip 2: Most Constraining Variable

## □ Variable paling membatasi

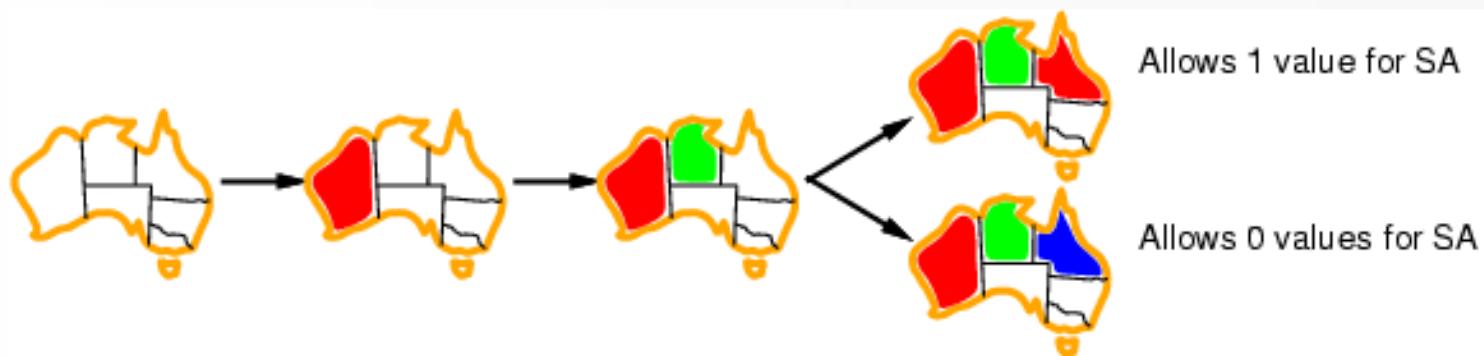
- Pilih variable yang terlibat constraint dengan variable lain (yang belum di-assign) yang paling banyak.
- Tie – breaker : gunakan kalau ada 2 atau lebih variable yang sama bagusnya berdasarkan prinsip 1.



# Prinsip 3: Least Constraining Value

## □ Nilai paling membebasi

- Pilih nilai yang menimbulkan batasan kemungkinan nilai variable lain (yang belum di-assign) yang paling sedikit.



- Jika ketiga prinsip ini digunakan, masalah n-queens dengan  $n=1000$  bisa diselesaikan!

# Forward checking



- Catat kemungkinan nilai sah untuk semua variable yang belum di-assign. Jika ada sebuah variable yang tidak ada kemungkinan nilai sah, langsung failure (backtrack).



WA

NT

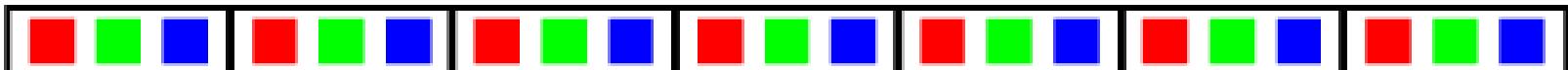
9

NSW

W

SA

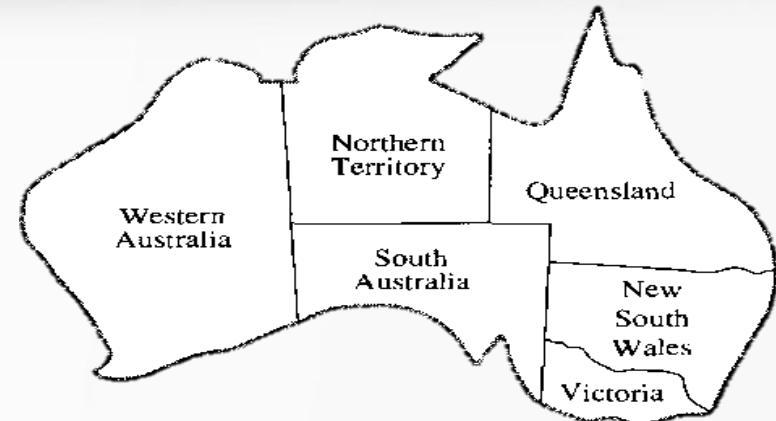
T



# Forward checking

## ❑ Idenya :

- Simpan nilai valid untuk variable yang belum di-assign
- Bila salah satu variable tidak mempunyai kemungkinan nilai yang valid maka search dihentikan

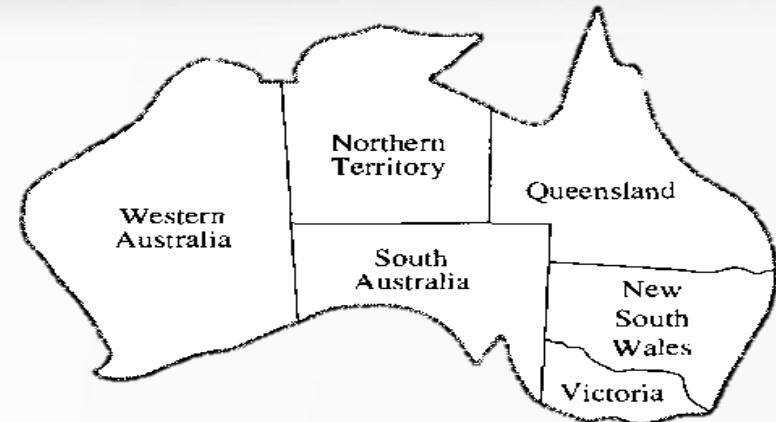


WA	NT	Q	NSW	V	SA	T
■ Red	■ Green	■ Blue	■ Red	■ Green	■ Blue	■ Red
■ Red			■ Red	■ Green	■ Blue	■ Red

# Forward checking

## ❑ Ideanya :

- Simpan nilai valid untuk variable yang belum di-assign
- Bila salah satu variable tidak mempunyai kemungkinan nilai yang valid maka search dihentikan

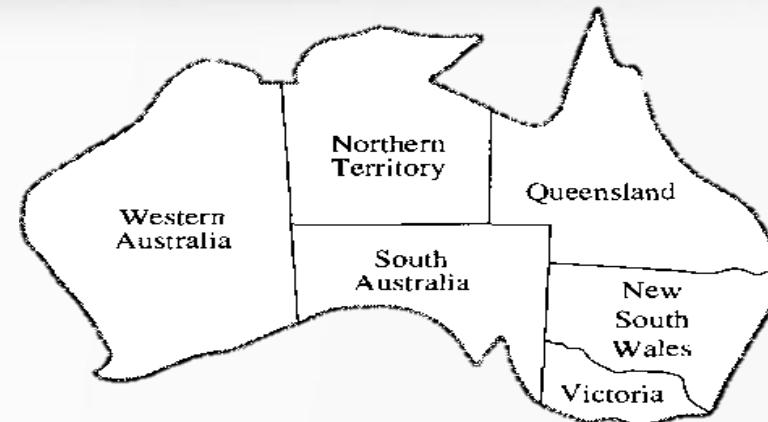


WA	NT	Q	NSW	V	SA	T
■ Red ■ Green ■ Blue						
■ Red		■ Green ■ Blue	■ Red ■ Green ■ Blue	■ Red ■ Green ■ Blue	■ Green ■ Blue	■ Red ■ Green ■ Blue
■ Red			■ Red	■ Blue	■ Red ■ Green ■ Blue	

# Forward checking

# □ Idenya :

- Simpan nilai valid untuk variable yang belum di-assign
  - Bila salah satu variable tidak mempunyai kemungkinan nilai yang valid maka search dihentikan

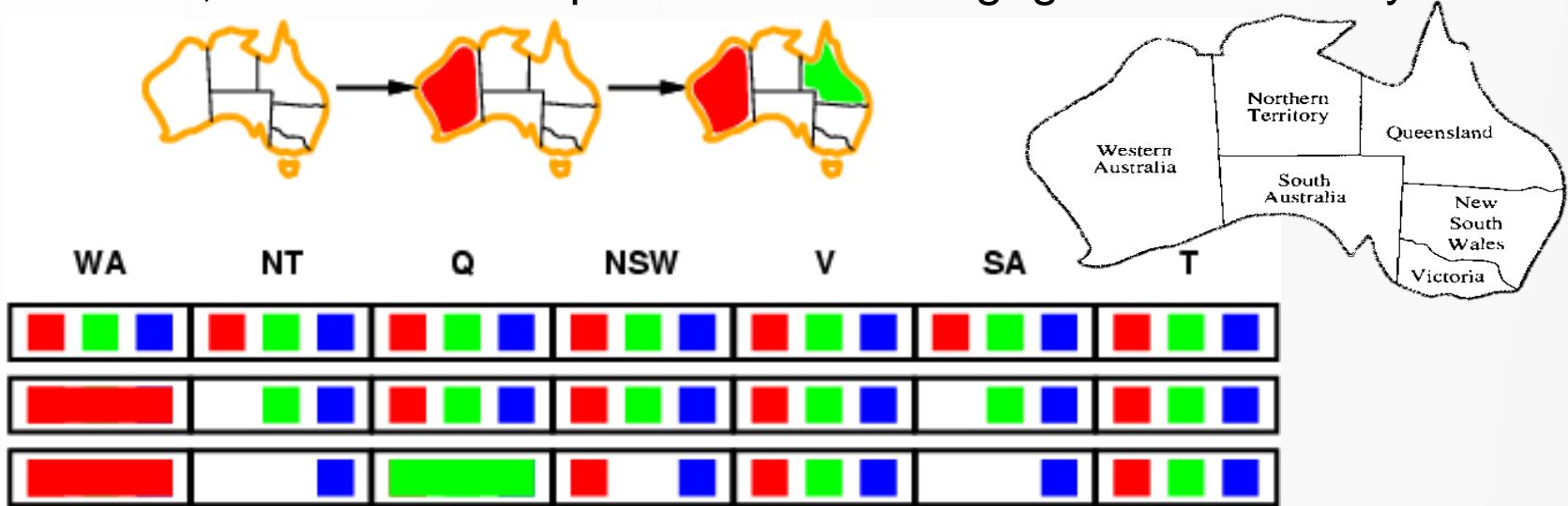


The figure consists of seven horizontal bars, one for each state/territory. Each bar is composed of three segments: a red segment on the left, a white segment in the middle, and a blue segment on the right. The width of each segment varies, indicating the relative proportion of that color in the respective region.

Region	Red (R)	White (W)	Blue (B)
WA	Large	Medium	Small
NT	Very Large	Very Small	Medium
Q	Medium	Medium	Medium
NSW	Medium	Very Small	Large
V	Medium	Medium	Medium
SA	Medium	Very Small	Large
T	Very Large	Very Small	Medium

# Constraint propagation

- Forward checking memberikan informasi dari variabel yang dialokasi, namun tidak dapat mendeteksi kegagalan sebelumnya.

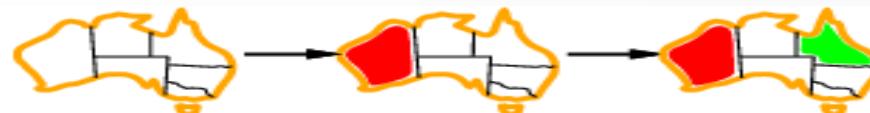


Note :

- Forward checking mem-propagasi (meneruskan) informasi dari variable yang sudah di-assign ke yang belum.
- Secara umum, ini disebut constraint propagation.
- Namun, tidak semua failure bisa di-deteksi secara dini.

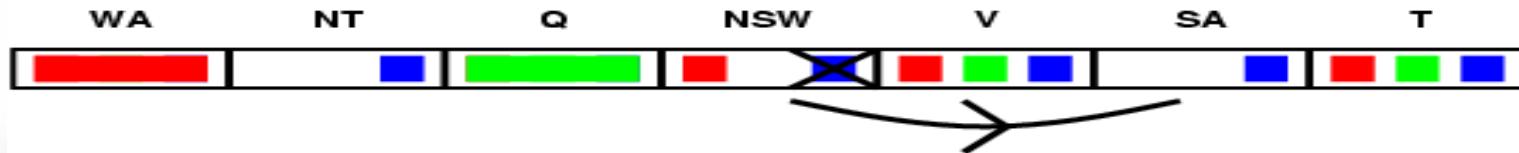
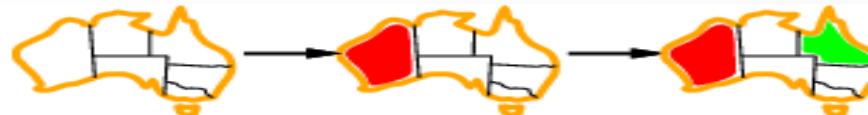
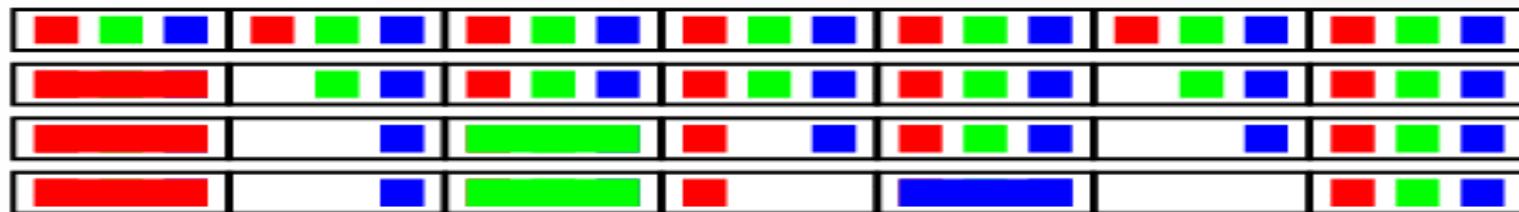
# Arc consistency

- ❑ Bentuk sederhana dari propagasi, membuat arc consistent
- ❑  $X \rightarrow Y$  is consistent iff for **every** value  $x$  of  $X$  there is **some** allowed  $y$



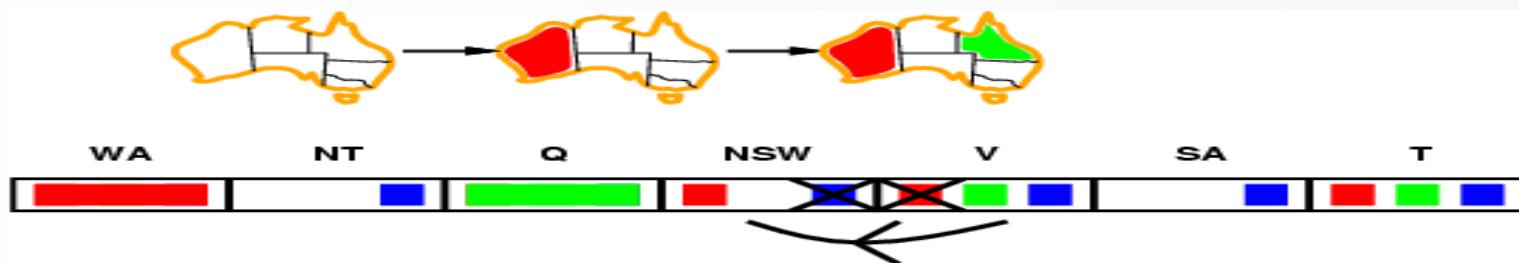
# Arc consistency

- ❑ Bentuk sederhana dari propagasi, membuat arc consistent
- ❑  $X \rightarrow Y$  is consistent iff for **every** value  $x$  of  $X$  there is **some** allowed  $y$



# Arc consistency

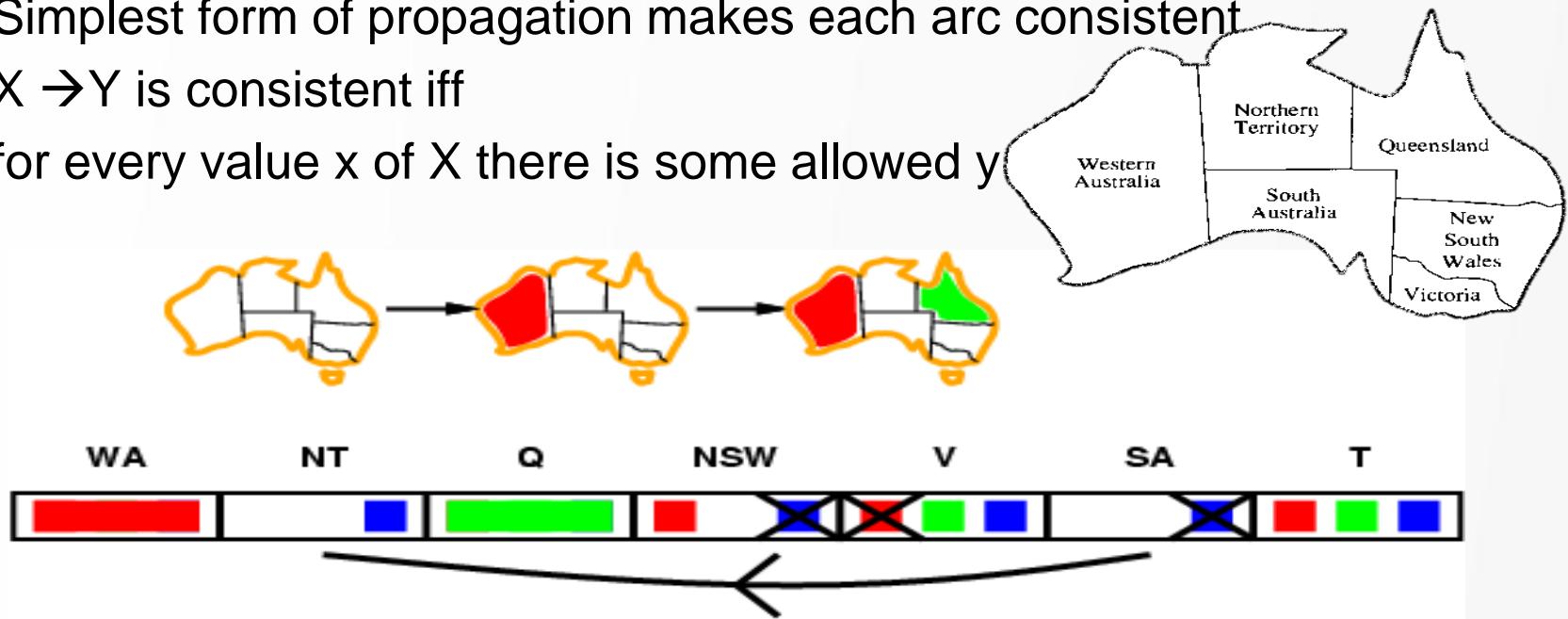
- Bentuk sederhana dari propagasi, membuat arc consistent
- $X \rightarrow Y$  is consistent iff  
for **every** value  $x$  of  $X$  there is **some** allowed  $y$



- Jika  $X$  kehilangan suatu nilai, neighbors dari  $X$  perlu diperiksa ulang.

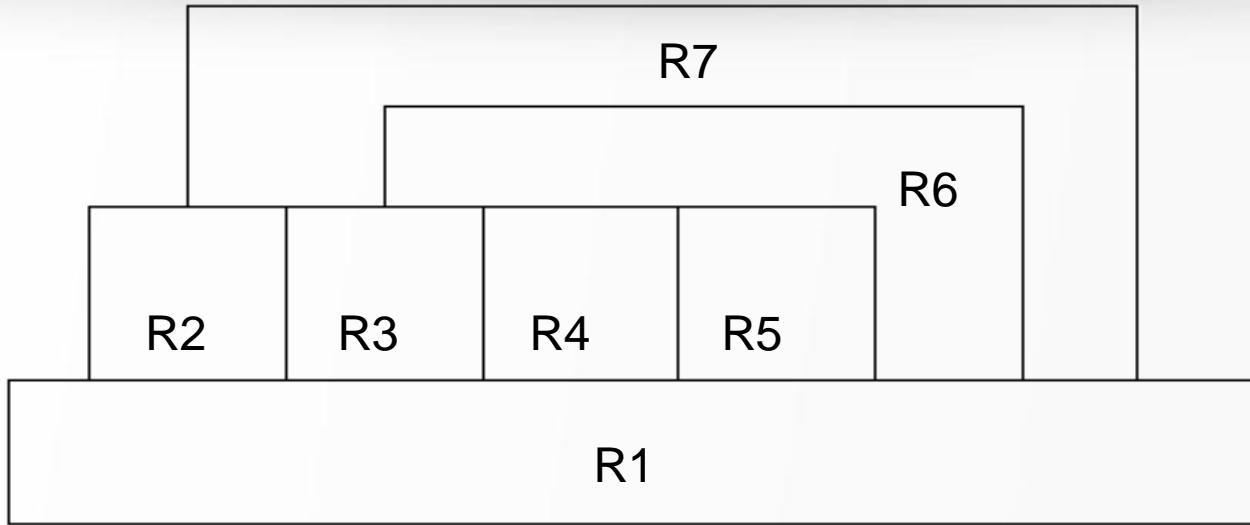
# Arc consistency

- ❑ Bentuk sederhana dari propagasi, membuat arc consistent
- ❑ Simplest form of propagation makes each arc consistent
- ❑  $X \rightarrow Y$  is consistent iff
- ❑ for every value  $x$  of  $X$  there is some allowed  $y$



- ❑ Jika  $X$  kehilangan suatu nilai, neighbors dari  $X$  perlu diperiksa ulang
- ❑ Arc consistency detects failure lebih dini dari pada forward checking
- ❑ Dapat dijalankan sebagai preprocessor atau setelah setiap assignment.

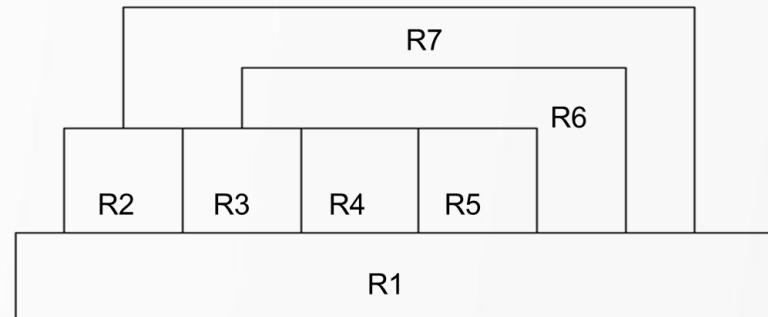
# Constraint Propagation: Map Coloring



- Dengan **heuristic** yang admissible dan consistent, A\* pasti complete dan optimal.
- Isikan bidang (R1..R7) di atas dengan warna : Merah, Kuning, Hijau, Biru.
- Bidang bertetangga tidak boleh memiliki warna yang sama.
  1. Apakah variabel yang Anda gunakan?
  2. Apakah domain yang tersedia?
  3. Bagaimana Anda mengevaluasi constraints-nya?

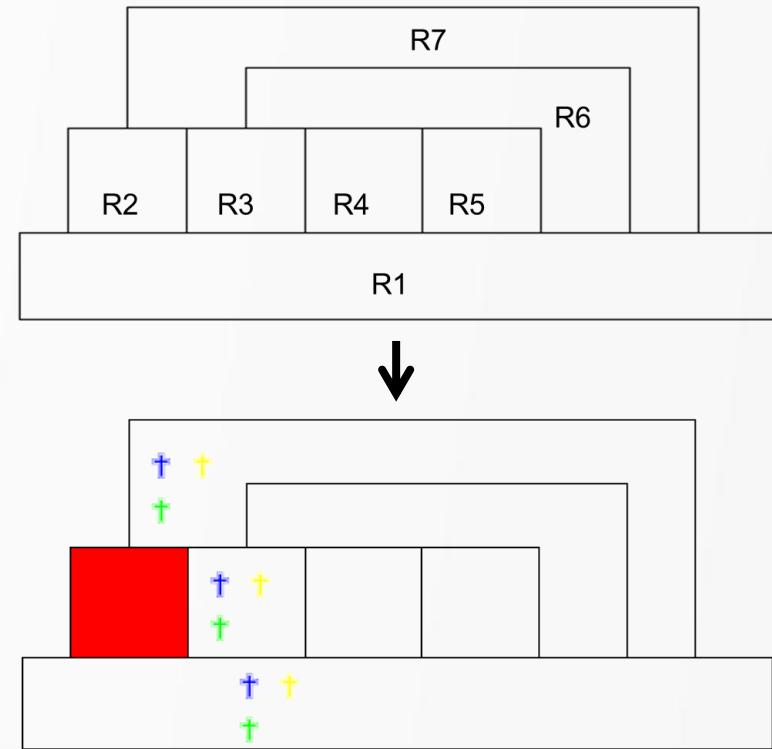
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



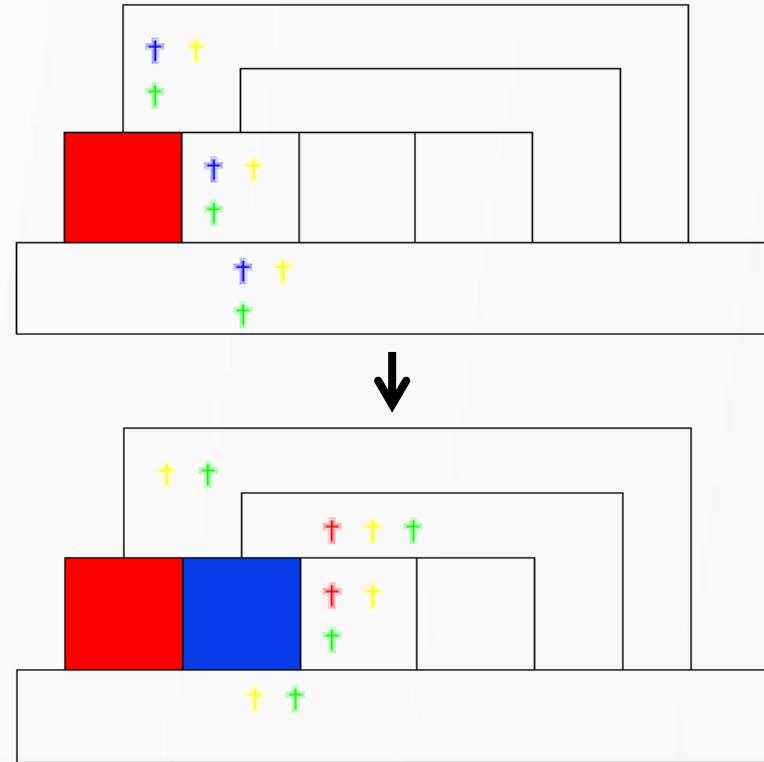
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



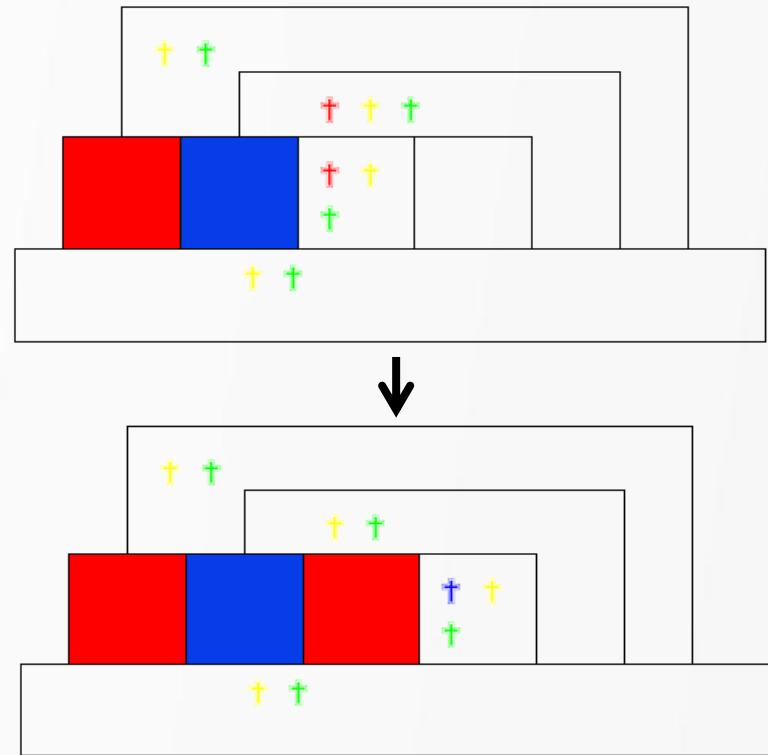
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



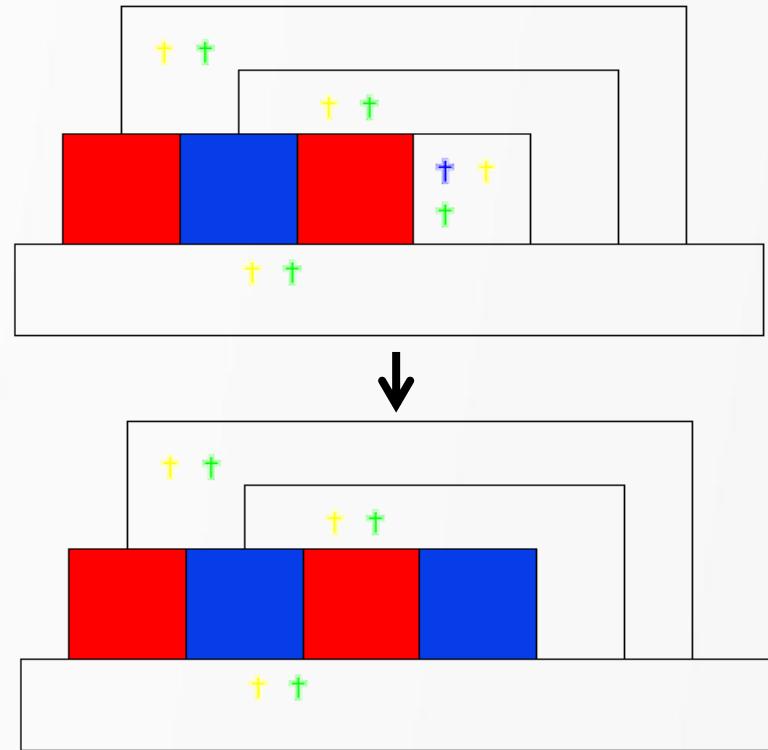
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



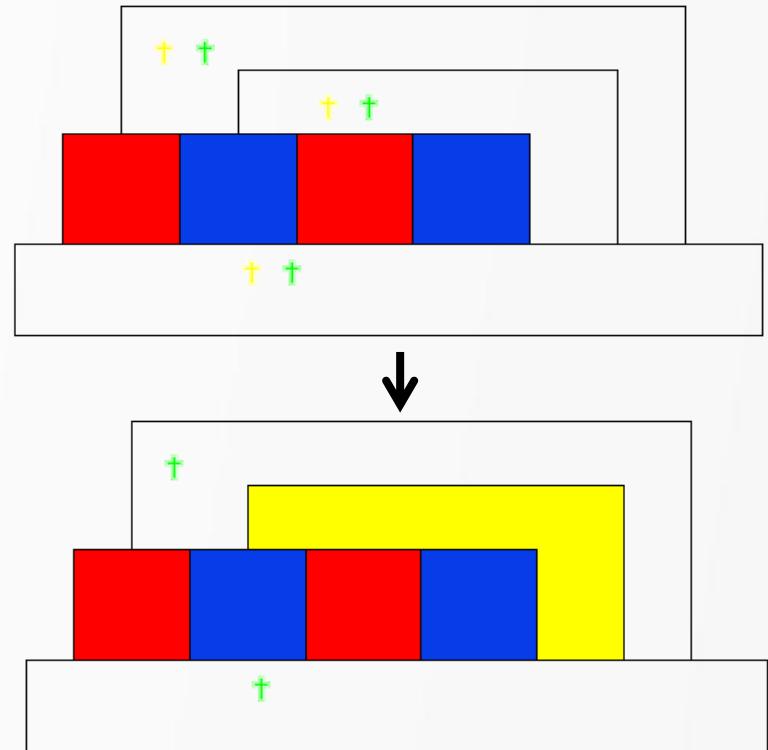
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



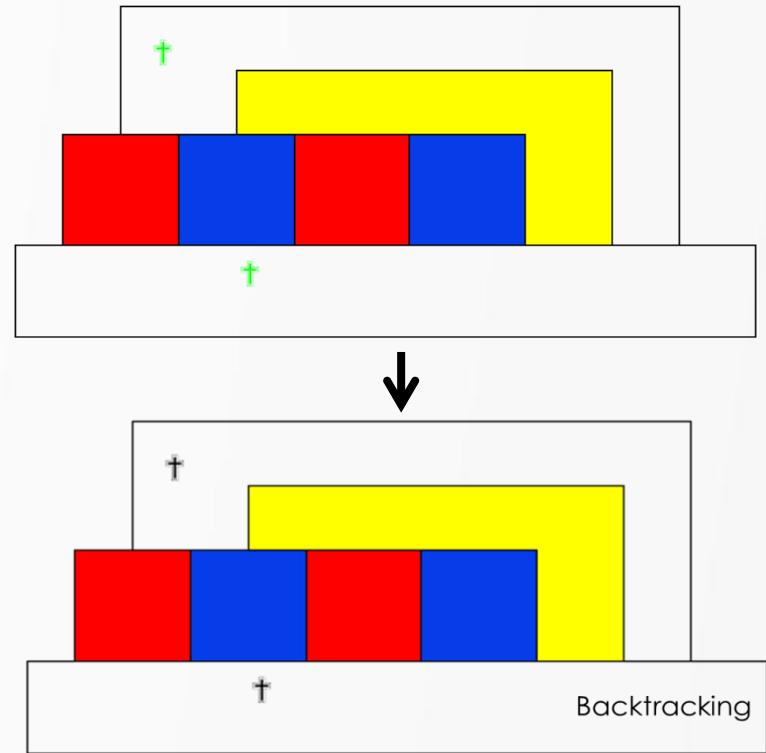
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



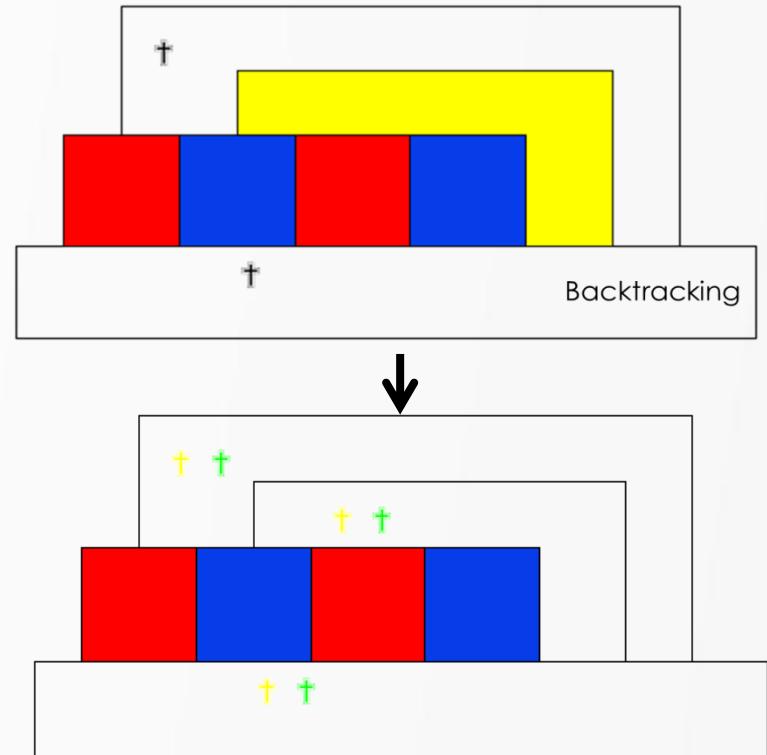
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



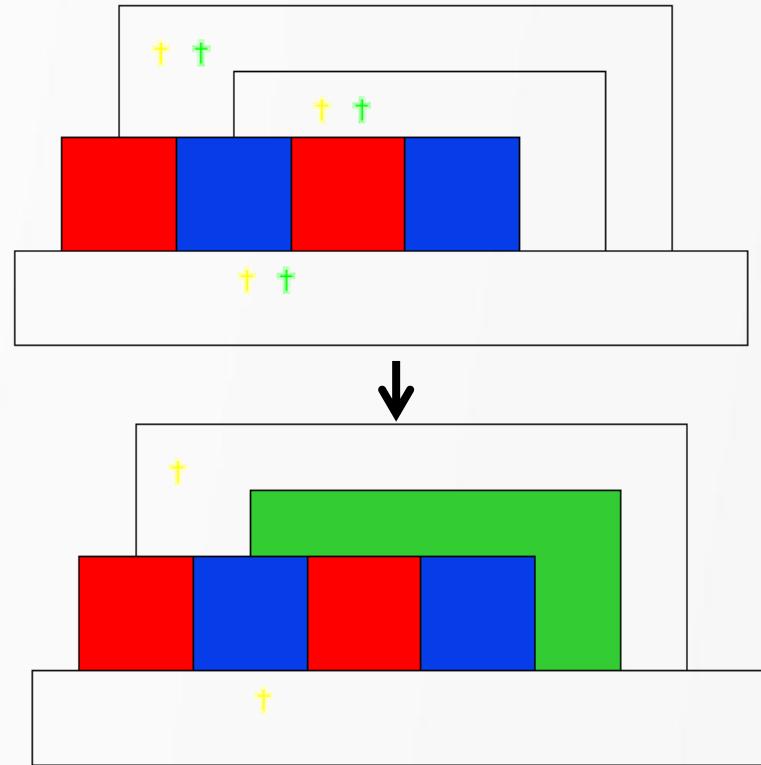
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



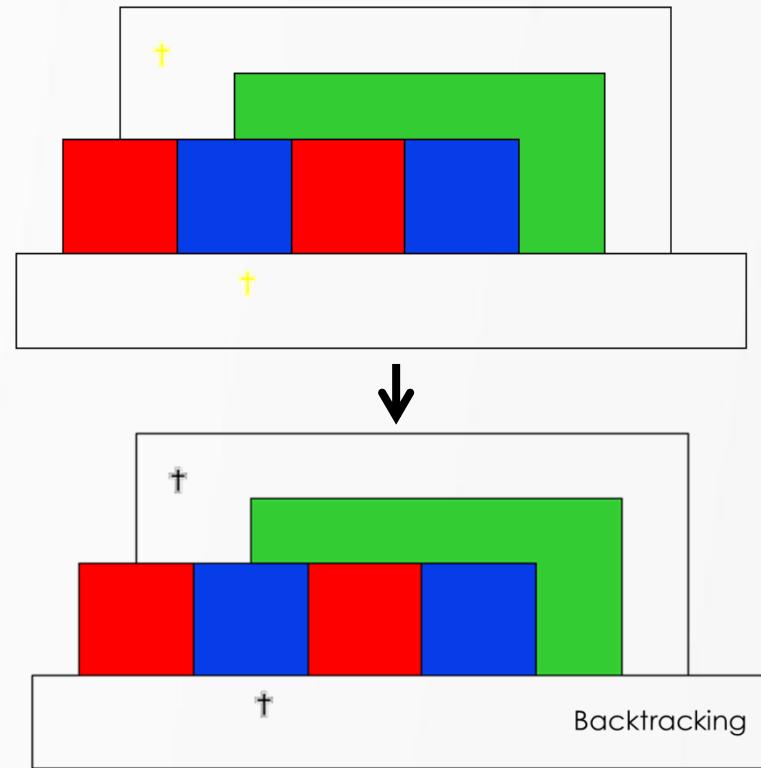
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



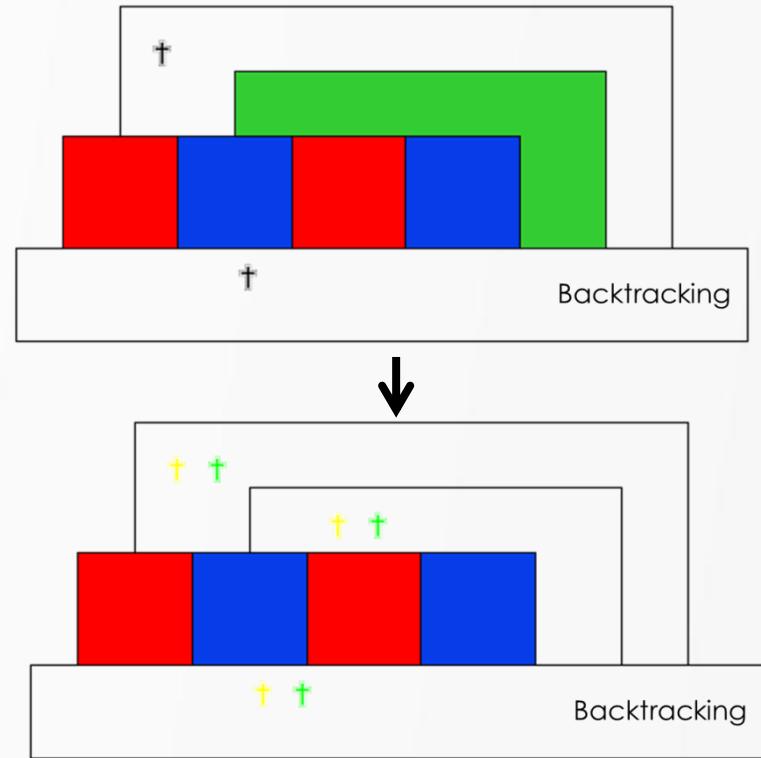
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



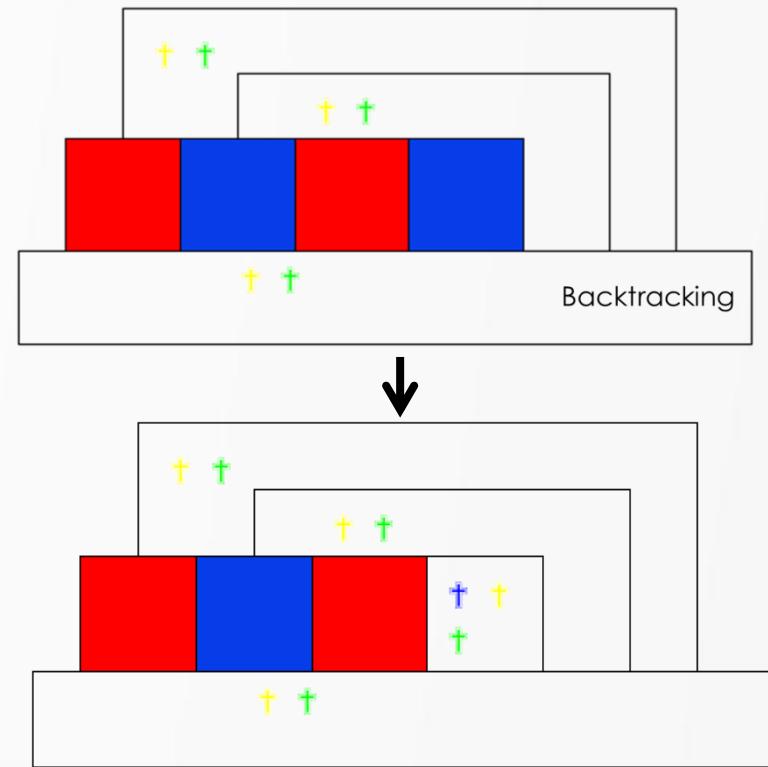
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



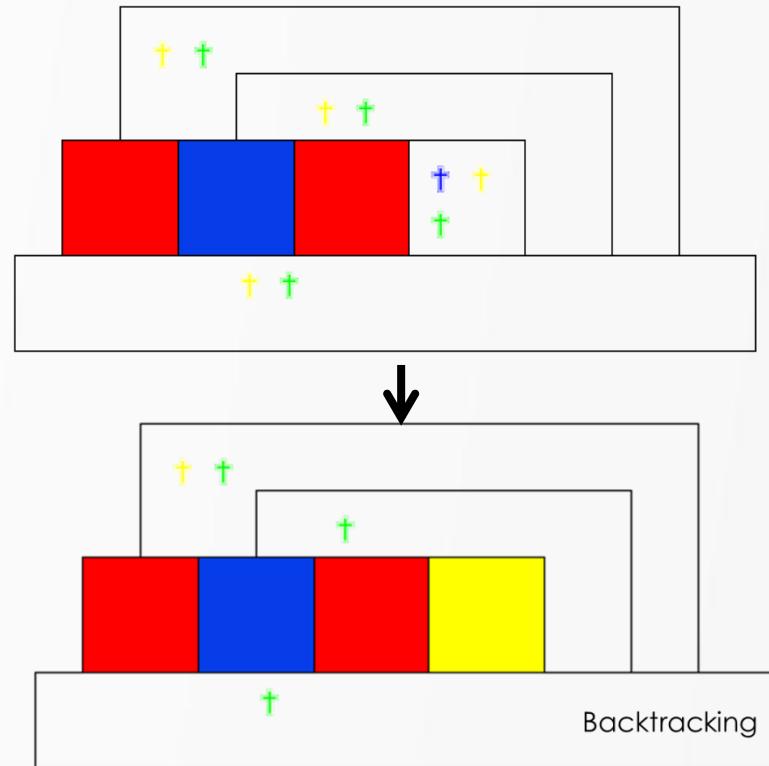
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



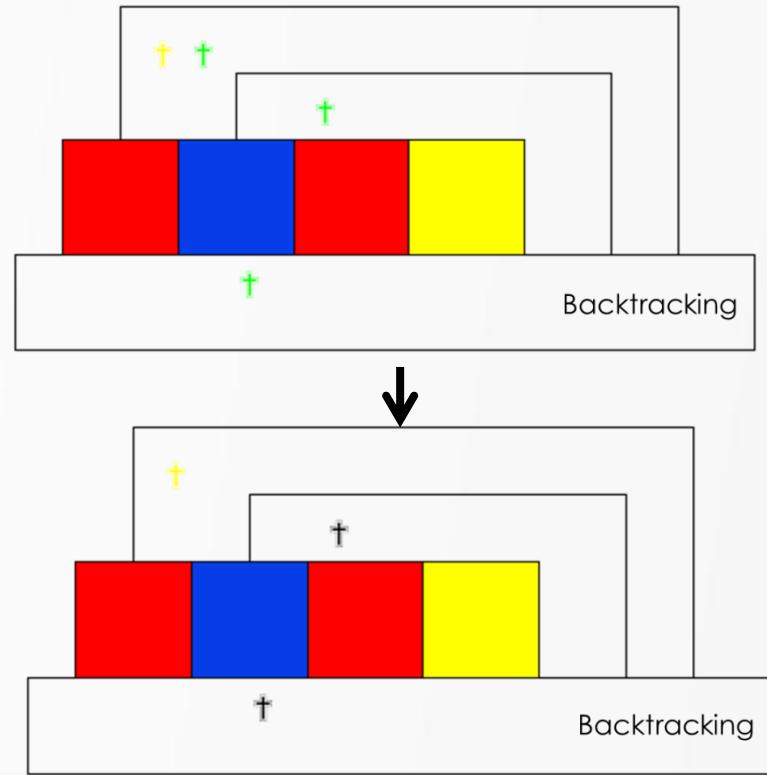
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



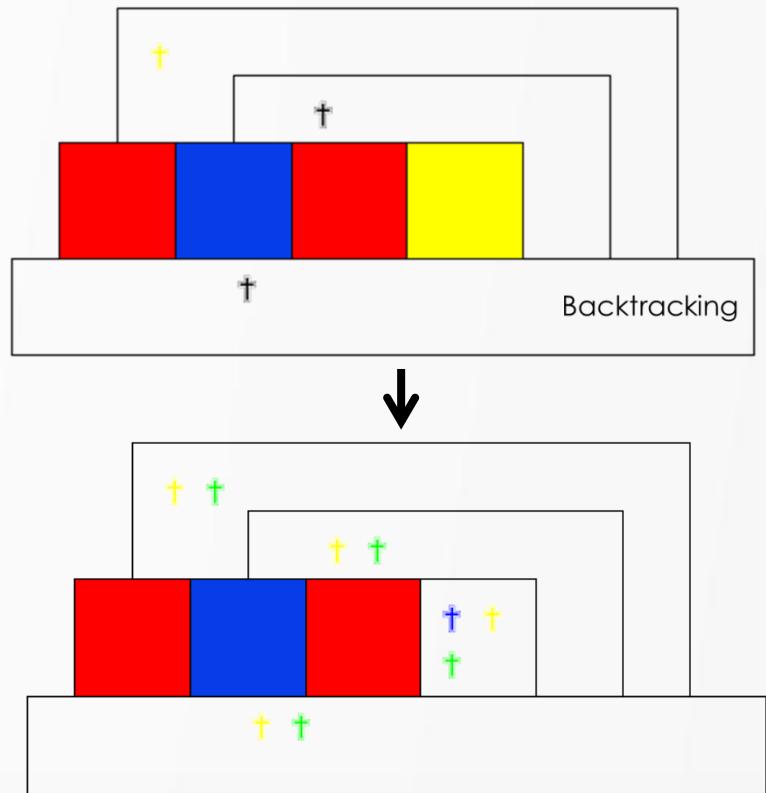
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



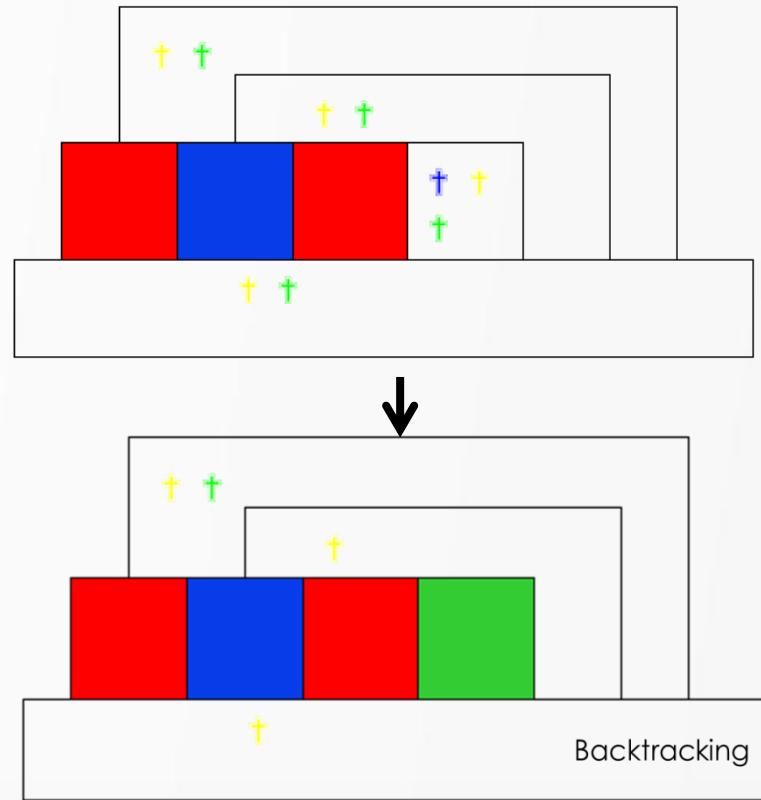
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



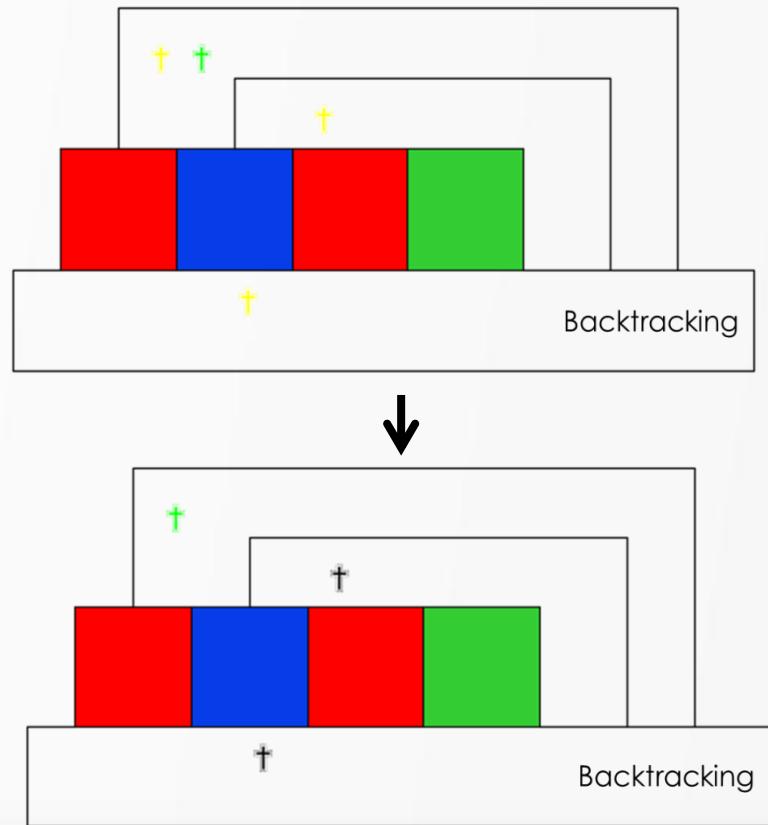
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



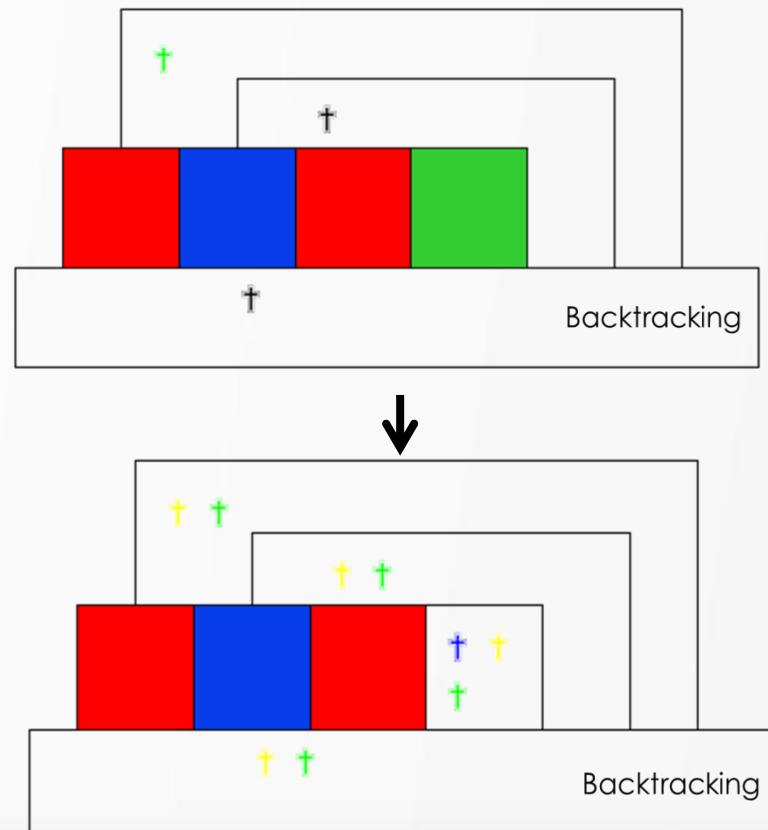
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



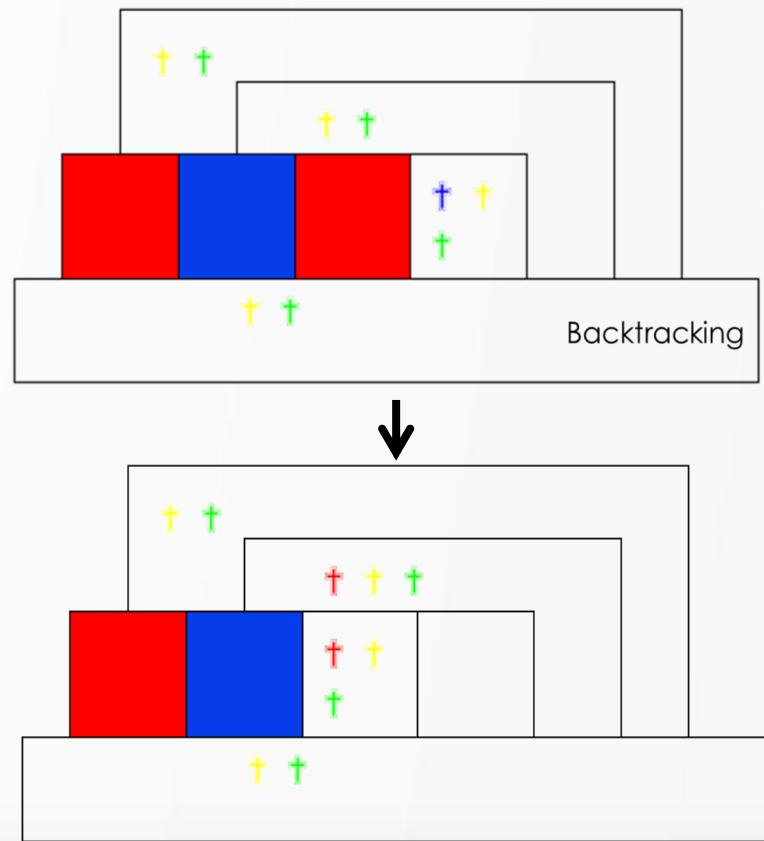
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



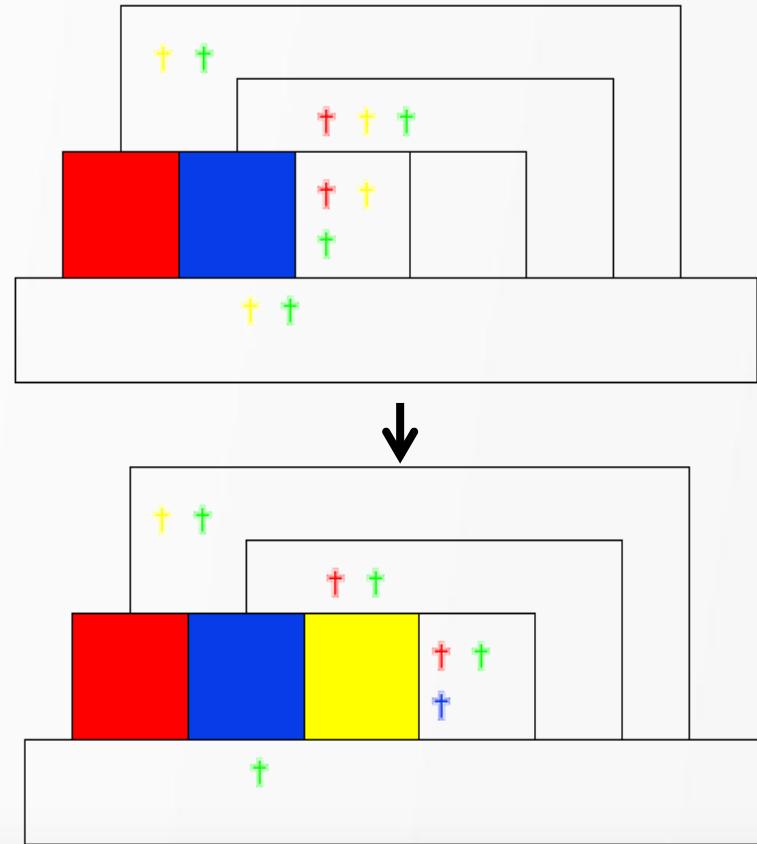
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



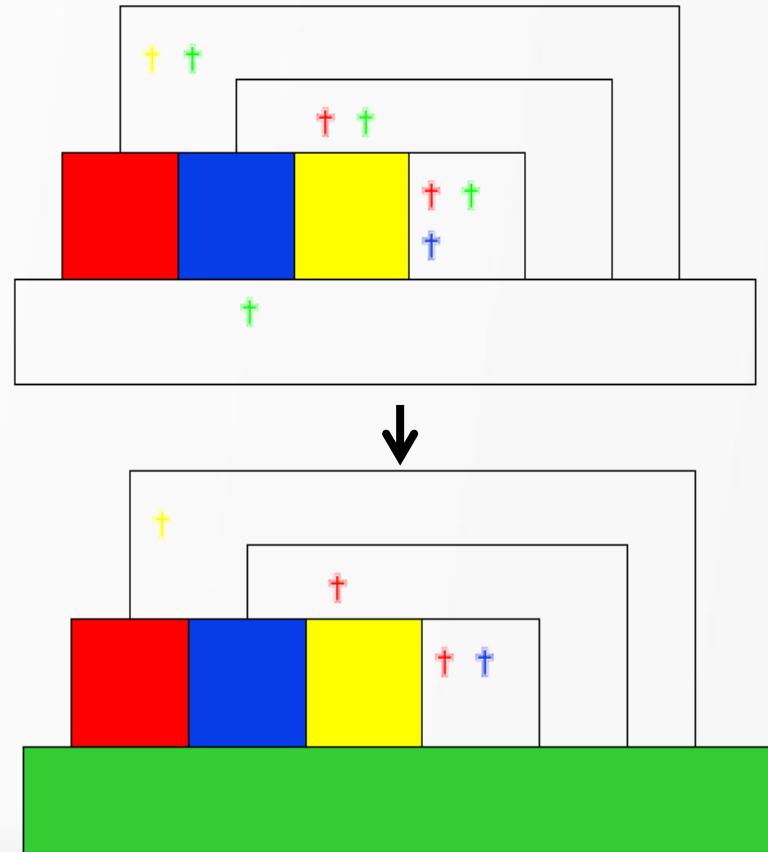
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



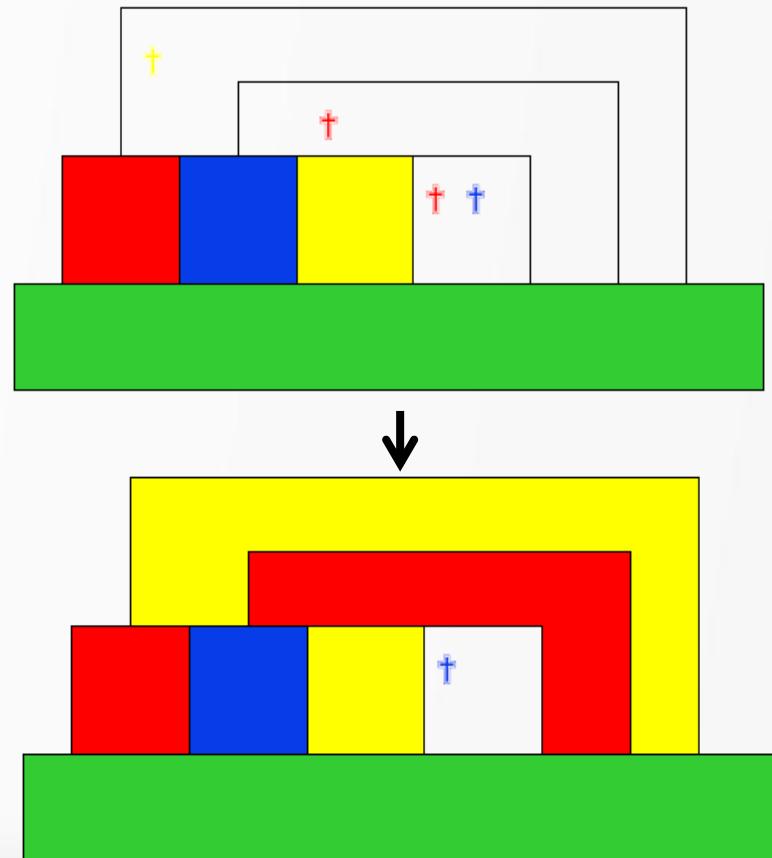
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



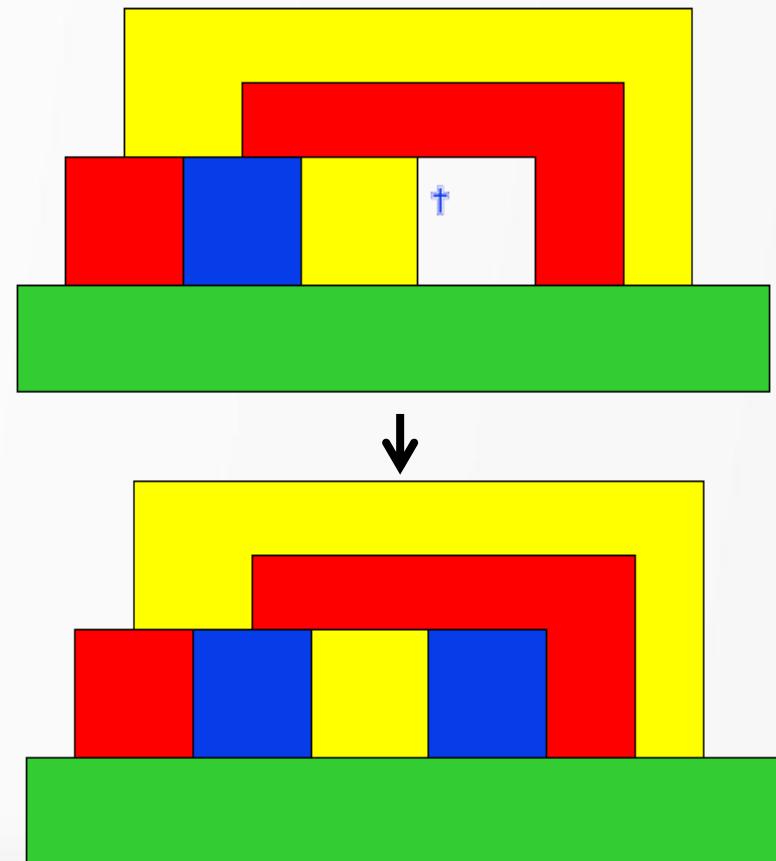
# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



# Constraint Propagation: Map Coloring

- Variabel yang harus diisi: R1, .. R7
- Domain yang tersedia: warna (**merah**, **kuning**, **hijau**, **biru**)
- Constraints :
  1.  $R1 \neq R2, \dots, R7,$
  2.  $R2 \neq R3,$
  3.  $R3 \neq R4,$
  4.  $R4 \neq R5,$
  5.  $R5 \neq R6,$
  6.  $R6 \neq R7$



# Selesai

