

[AIMA] Russel, Stuart J., Peter Norvig, "Artificial Intelligence, A Modern Approach" 3rd Ed., Prentice Hall, New Jersey, 2010

KI091322 Kecerdasan Buatan

Materi 7: Pencarian dgn. Batasan Kondisi (*Constraint Satisfaction Problems*)

Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember Surabaya

2012

Pengembangan Bahan Ajar sebagai Pendukung Student Centered-Learning
(SCL) melalui e-Learning : Share ITS

Latar Belakang CSP

- Teknik pencarian terdahulu...
 - *uninformed, informed, local, adversarial*
- ...untuk memecahkan problem dengan mencari *state* yang bisa menjadi solusi
- Struktur internal *state* setiap problem tidak sama
- *Constraint Satisfaction Problems* (CSP) melakukan konfirmasi struktur internal *state* yang memenuhi syarat *goal test*

Representasi dengan CSP

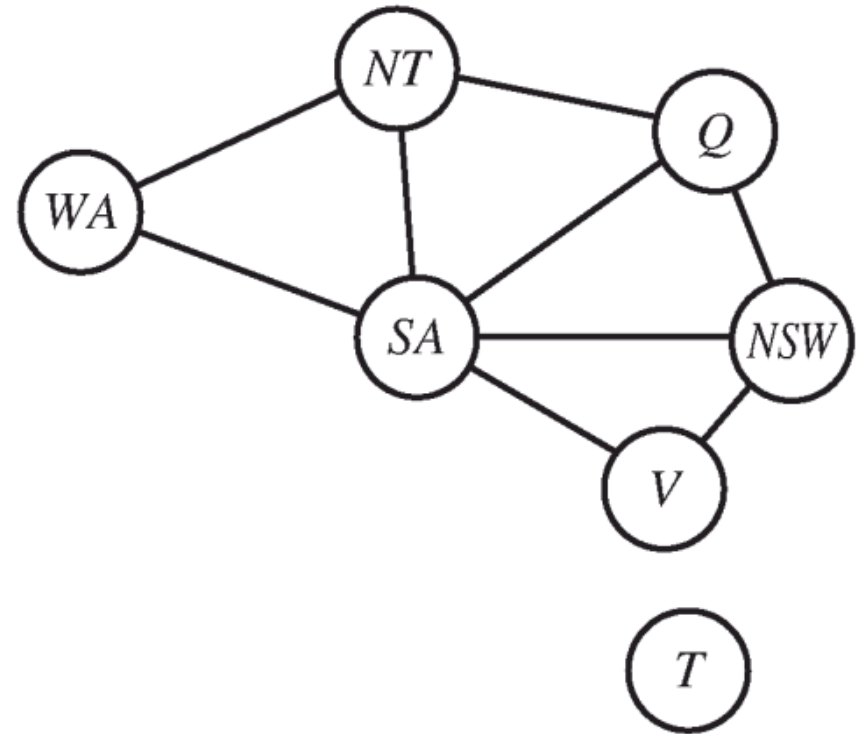
- Contoh Problem: Pemetaan 3 warna pada peta Australia
 - Tidak boleh ada warna sama pada negara bagian yang berbatasan
- Terdapat set variabel X_1, X_2, \dots, X_N .
 - Variabel WA, NT, Q, NSW, V, SA , dan T untuk negara bagian Western Australia, Northern Territory, Queensland, New South Wales, Victoria, South Australia, Tasmania
- Terdapat set batasan (*constraint*) C_1, C_2, \dots, C_M .
 - Setiap variabel negara bagian memiliki kemungkinan warna *red, green, blue*
- Setiap variabel X_i memiliki kemungkinan nilai (*values*) dari domain D_i yang konsisten atau tidak melanggar aturan *constraint*

Jenis *Constraint*

- *Unary constraint* untuk satu variabel
 - $SA \neq \text{green}$
- *Binary constraint* untuk pasangan variabel
 - $SA \neq WA$
- *Higher-order constraint* untuk >2 variabel
- *Preference constraint (soft constraint)*
 - Representasi *red is better than green* adalah pemberian nilai bobot yang berbeda

=> *constrained optimization problem*.

Representasi CSP sebagai *graph*



Contoh: nilai WA dibatasi dengan *constraint* dari nilai NT dan SA, sehingga *node* WA memiliki *edge* terhubung ke *node* NT dan SA

T tidak dipengaruhi apapun, sehingga *node* T menjadi *subgraph*

Inisialisasi Pewarnaan Peta Australia



- **Variables** WA, NT, Q, NSW, V, SA, T
- **Domains** $D_i = \{\text{red}, \text{green}, \text{blue}\}$
- **Constraints**: adjacent region memiliki warna berbeda
- e.g., $WA \neq NT$, atau (WA, NT) in $\{(\text{red}, \text{green}), (\text{red}, \text{blue}), (\text{green}, \text{red}), (\text{green}, \text{blue}), (\text{blue}, \text{red}), (\text{blue}, \text{green})\}$

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

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

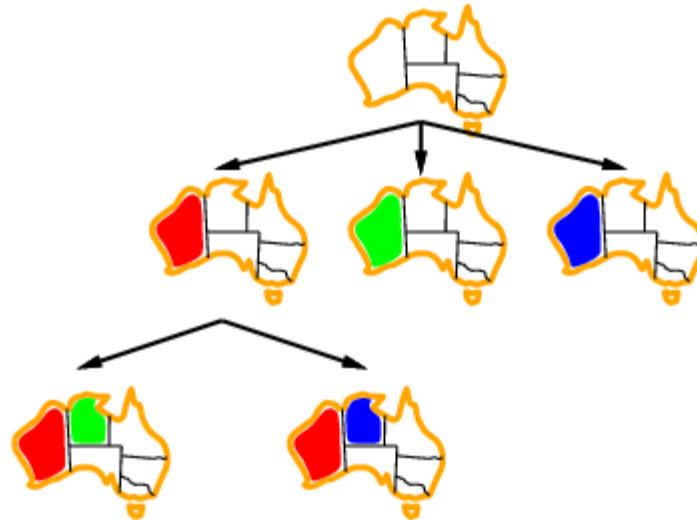
$SA \neq WA$ is a shortcut for $\langle (SA, WA), SA \neq WA \rangle$

$\{(\text{red}, \text{green}), (\text{red}, \text{blue}), (\text{green}, \text{red}), (\text{green}, \text{blue}), (\text{blue}, \text{red}), (\text{blue}, \text{green})\}$

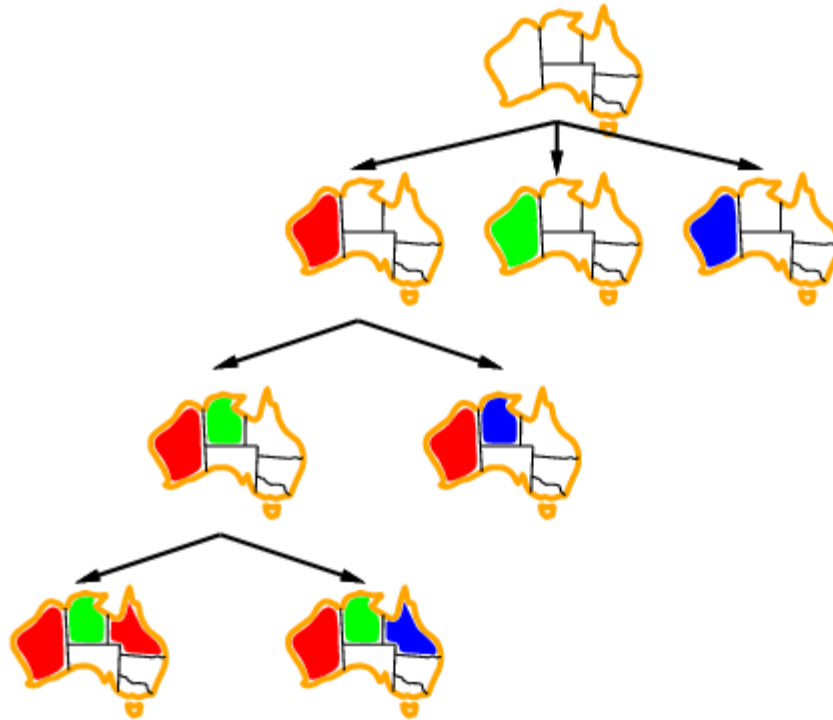
Contoh Pewarnaan Peta Australia



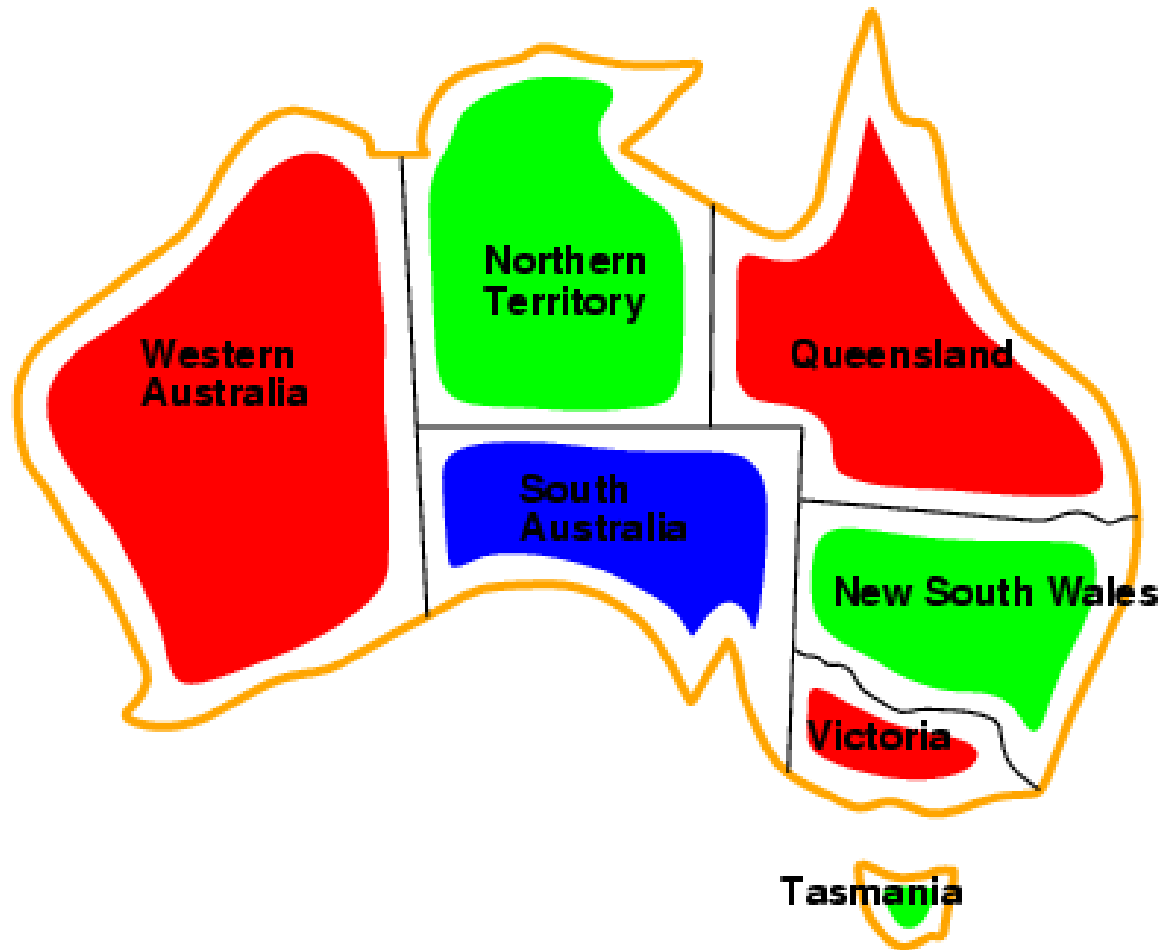
Contoh Pewarnaan Peta Australia



Contoh Pewarnaan Peta Australia



Contoh Solusi Pewarnaan Peta Australia



CSP vs *Search State-Space*

- Jika problem diselesaikan dengan pencarian
 - Ada $3^5 = 243$ kombinasi nilai warna setiap variabel
 - Jumlah anggota pada *state-space* = 243
- Jika problem diselesaikan dengan CSP
 - Hanya $2^5 = 32$ kombinasi karena satu warna sudah terpilih (pengurangan sampai 87%)
 - Jumlah anggota pada *state-space* = 32

Algoritma untuk Solusi CSP

- Backtracking Search
 - Pemilihan *state* yang memenuhi syarat *constraint*
- Constraint Propagation (inference, dugaan solusi)
 - Menggunakan *constraint* untuk mengurangi jumlah kemungkinan nilai pada variabel dan sebaliknya
 - pendekatan: *forward checking*, *arc consistency*

Algoritma CSP 1: backtracking

Backtracking Search = Depth-first search (DFS) untuk CSP dengan pemberian nilai pada single-variable (= *uninformed search*)

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure  
  return BACKTRACK( $\{ \}$ , csp)
```

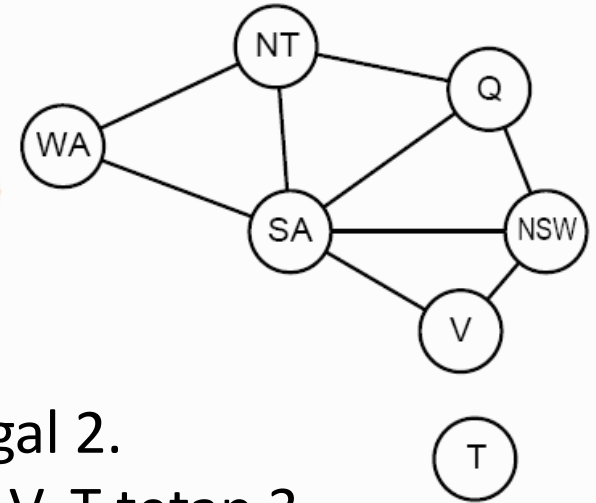
```
function BACKTRACK(assignment, csp) returns a solution, or failure  
  if assignment is complete then return assignment  
  var  $\leftarrow$  SELECT-UNASSIGNED-VARIABLE(csp)  
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do  
    if value is consistent with assignment then  
      add  $\{var = value\}$  to assignment  
      inferences  $\leftarrow$  INFERENCE(csp, var, value)  
      if inferences  $\neq$  failure then  
        add inferences to assignment  
        result  $\leftarrow$  BACKTRACK(assignment, csp)  
        if result  $\neq$  failure then  
          return result  
      remove  $\{var = value\}$  and inferences from assignment  
  return failure
```

Langkah Penting Backtracking

Hal yang menjadi isu:

- SELECT-UNASSIGNED-VARIABLE (urutan pemilihan variabel)
 - *Most Constrained***ed** Variable (*Minimum Remaining Values*, MRV)
 - Pilih variabel dengan variasi kemungkinan nilai paling sedikit
 - Jika >1 variabel, maka digunakan *Most Constraining Variable*
 - *Most Constraining***ing** Variable (MCV)
 - Pilih variabel yang memiliki jumlah *constraint* lebih banyak
- ORDER-DOMAIN-VALUES (urutan pemilihan nilai dari variabel)
 - *Least Constraining Value* (LCV)
 - Pilih nilai variabel yang memiliki constraint lebih sedikit untuk variabel lain
 - Atau pilih nilai variabel yang membuat variabel lain memiliki kemungkinan pilihan nilai lebih banyak

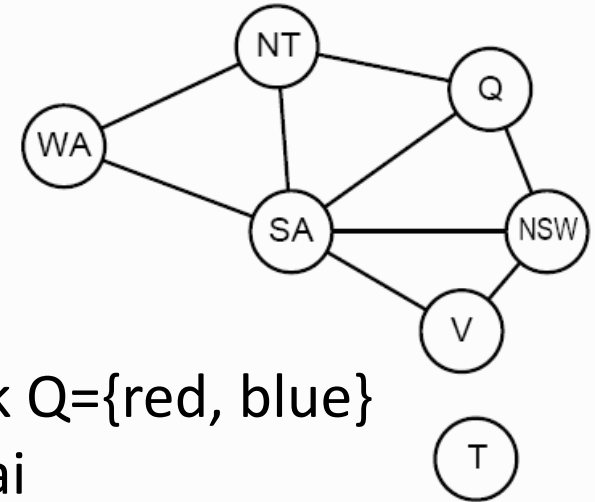
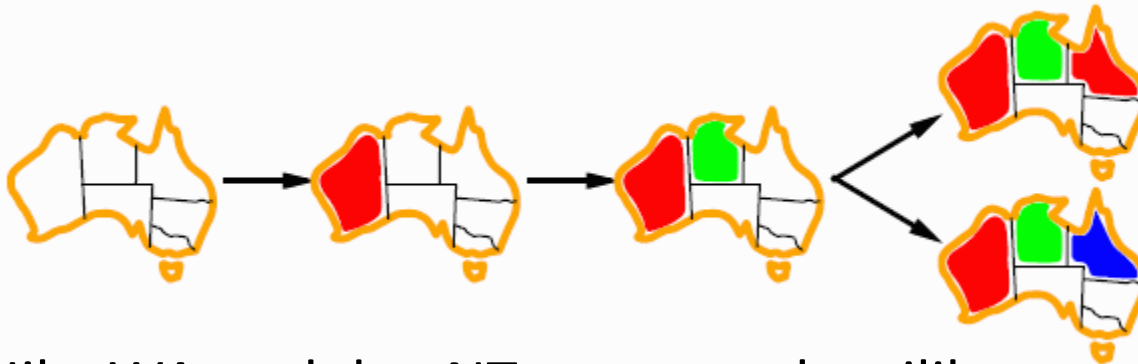
Contoh Penggunaan MRV dan MCV



Jika WA=red, maka pilihan untuk NT & SA tinggal 2.
Tidak berbatasan dengan WA, pilihan Q, NSW, V, T tetap 3.
MRV memilih NT & SA karena memiliki variasi kemungkinan nilai paling sedikit ($2 < 3$)

Lakukan MCV karena masih ada 2 pilihan: NT atau SA
Jumlah *constraint* NT ada 3 (node terhubung) -> WA, SA, Q
Jumlah *constraint* SA ada 5 (node terhubung) -> WA, NT, Q, NSW, V
MCV memilih SA karena memiliki jumlah *constraint* lebih banyak

Contoh Penggunaan LCV



Jika WA=red dan NT=green maka pilihan untuk Q={red, blue}

Jika Q = red maka variabel SA ada 1 pilihan nilai

(2 constraint, $SA \neq \text{red}$ & $SA \neq \text{green}$)


Jika Q = blue maka variabel SA tidak ada pilihan nilai

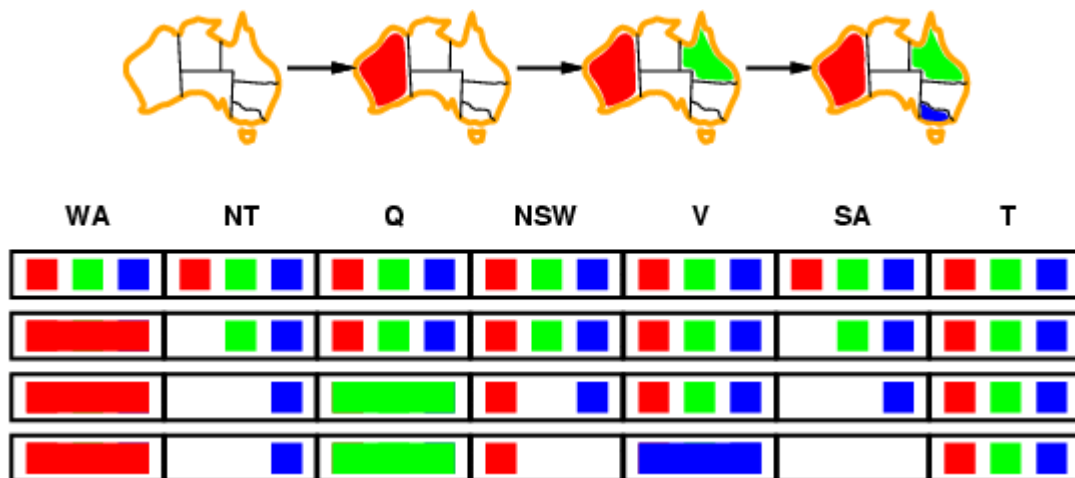
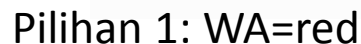
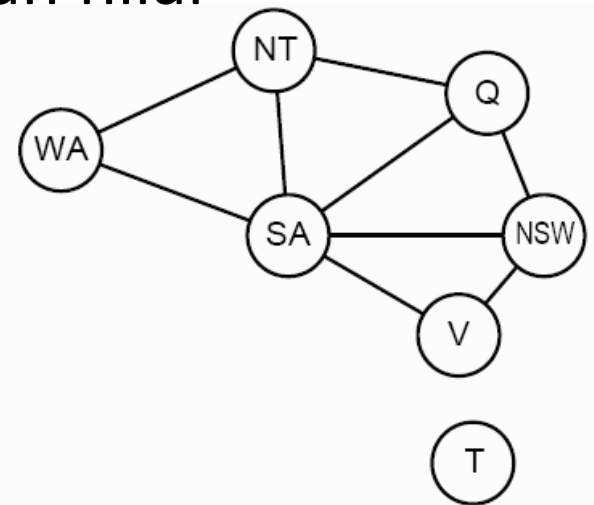
(3 constraint, $SA \neq \text{red}$ & $SA \neq \text{green}$ & $SA \neq \text{blue}$)

LCV akan memilih Q=red karena:

- red membuat pilihan constraint lebih sedikit untuk variabel SA
- atau
- red membuat pilihan nilai variabel SA lebih banyak ($1 > 0$)

Algoritma CSP 2: constraint propagation

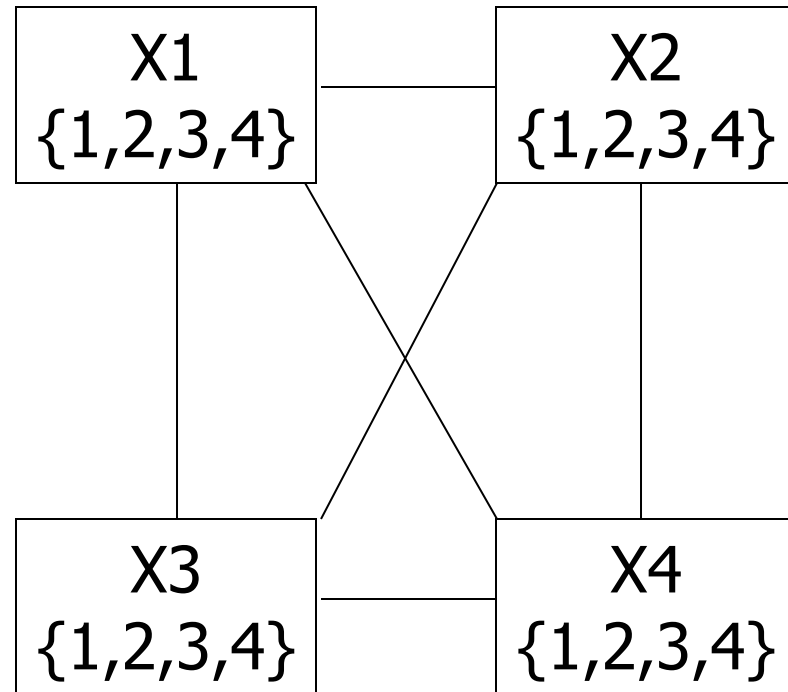
- Pendekatan *forward checking*
 - Mencatat (*keep track*) kemungkinan nilai yang konsisten dengan *constraint* untuk semua variabel
 - Pencarian dihentikan jika salah satu variabel sudah tidak memiliki kemungkinan nilai
 - backtrack dengan pilihan nilai lain
- 



Contoh Forward Checking: 4-Queens Problem

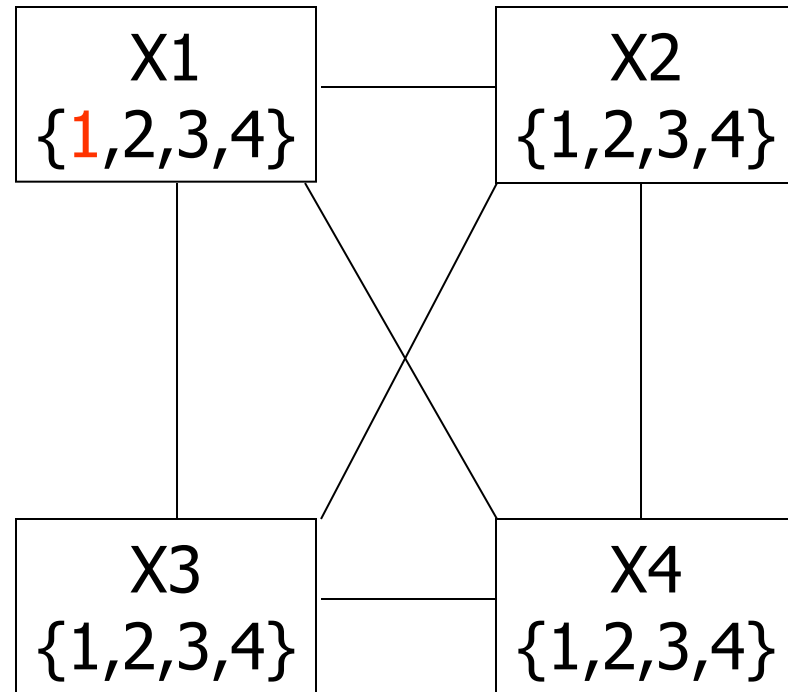
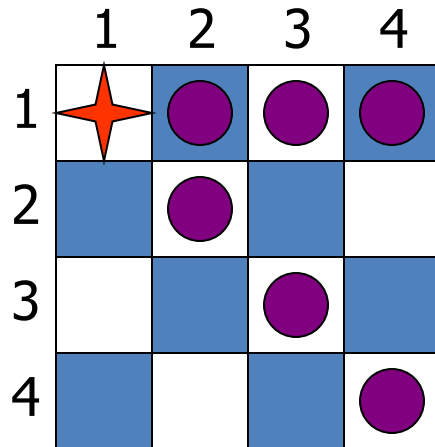
	1	2	3	4
1				
2				
3				
4				

[sumber: *slide CMSC 421 Artificial Intelligence, Bonnie J. Dorr, Univ. of Maryland*]










Representasi *complete graph* untuk 4 variabel problem 4-Queens saat inisialisasi

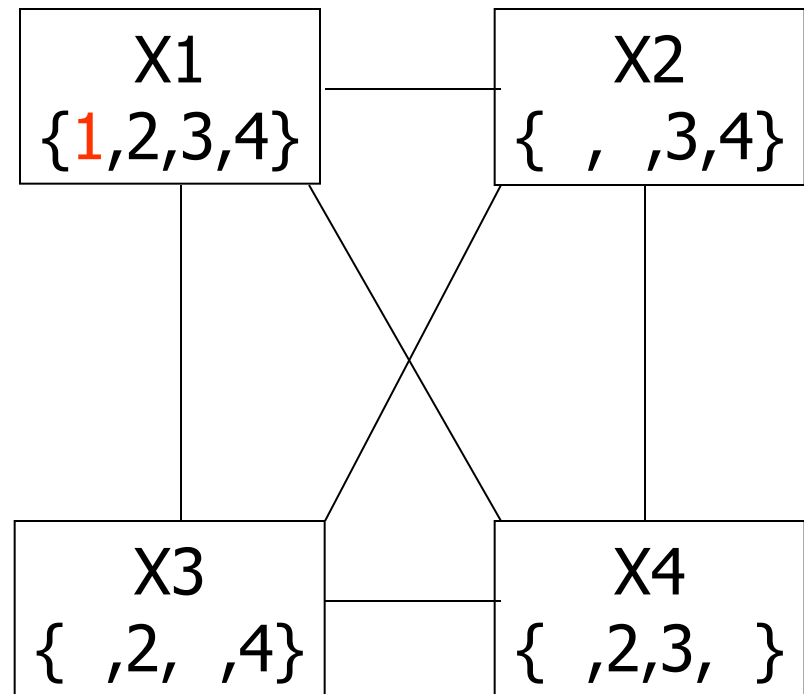
Contoh Forward Checking: 4-Queens Problem



Pilih variabel X1=1

Contoh Forward Checking: 4-Queens Problem

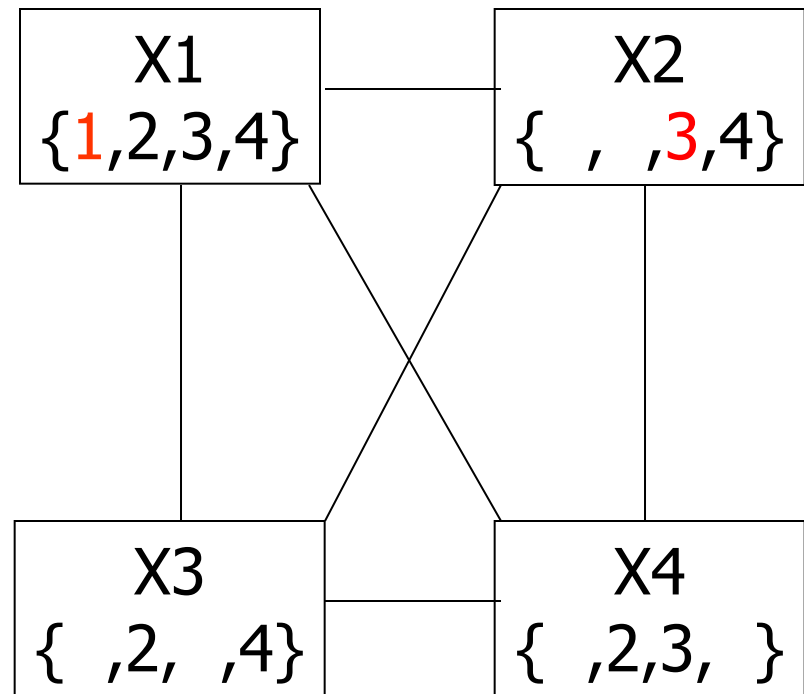
	1	2	3	4
1				
2				
3				
4				



Pilih variabel $X1=1$, sisa nilai variabel $X2$, $X3$, $X4$ dengan forward checking

Contoh Forward Checking: 4-Queens Problem

	1	2	3	4
1	★	●	●	●
2		●	●	
3		★	●	●
4			●	●

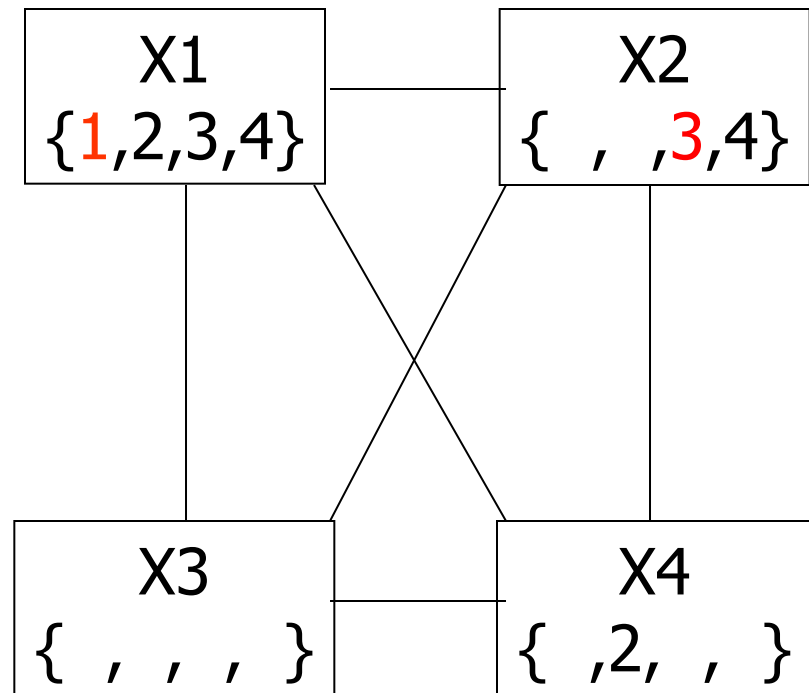


Pilih variabel X1=1 dan X2=3

Contoh Forward Checking: 4-Queens Problem

	1	2	3	4
1	★	●	●	●
2	●	●	●	
3		★	●	●
4	●		●	●

Pilih variabel $X1=1$ dan $X2=3$, membuat variabel $X3$ tidak memiliki nilai \rightarrow backtrack utk $X2=4$

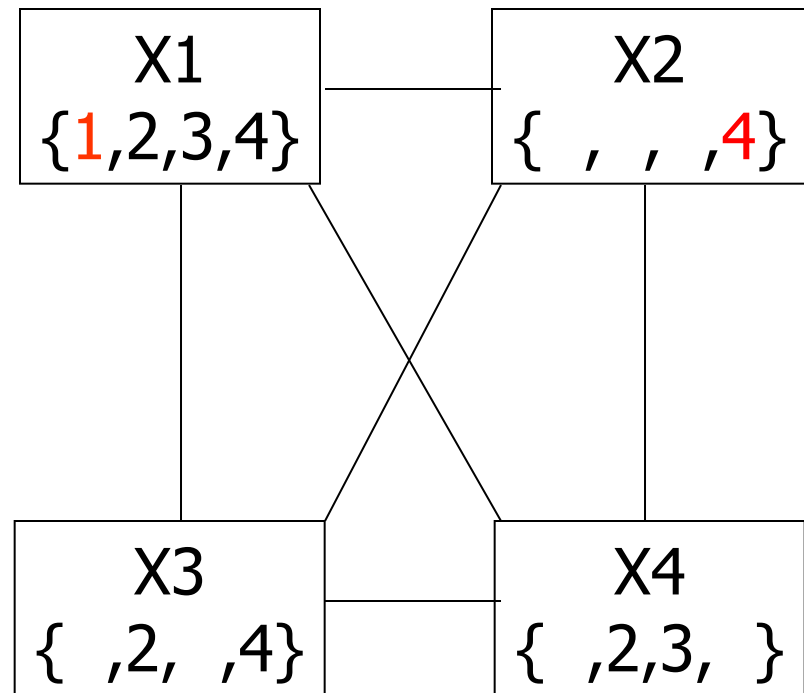


Pilih variabel $X1=1$ dan $X2=3$, sisa nilai variabel $X3$, $X4$ dengan forward checking

Contoh Forward Checking: 4-Queens Problem

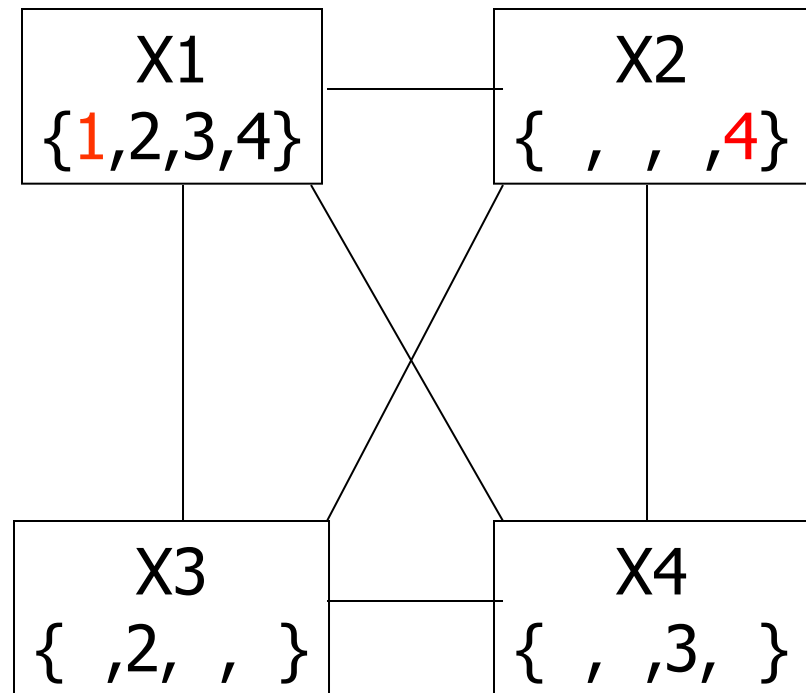
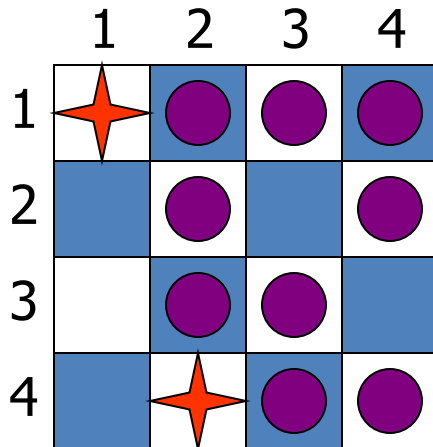
	1	2	3	4
1	★	●	●	●
2	■	●	■	●
3	□	●	●	■
4	■	★	●	●

Pilih variabel $X1=1$ dan $X2=3$, membuat variabel $X3$ tidak memiliki nilai \rightarrow backtrack utk $X2=4$



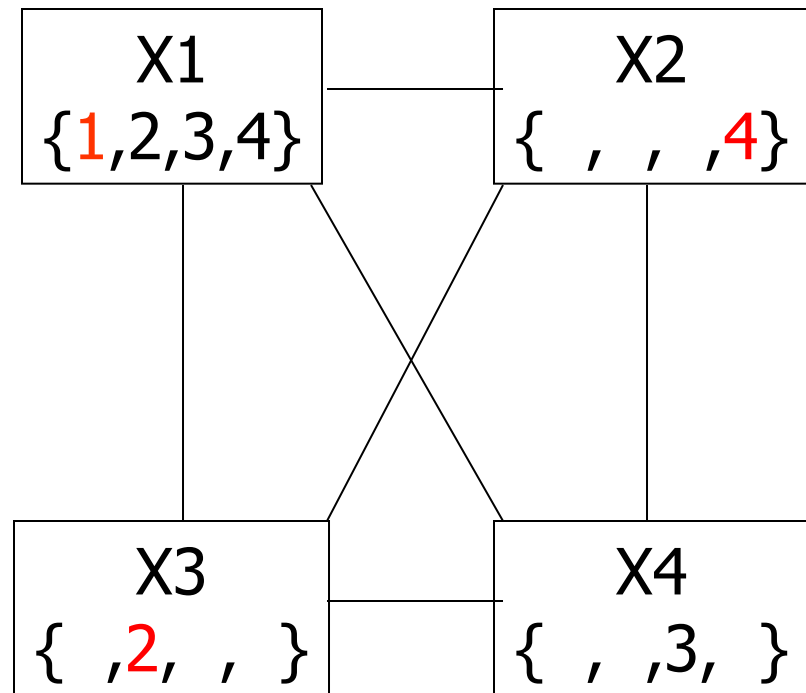
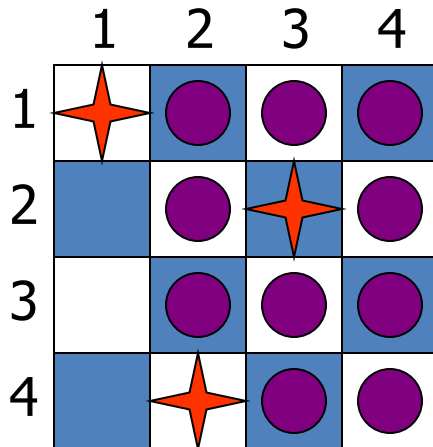
Kembali ke state setelah variabel $X1=1$,
Lalu nilai $X2=3$ dibuang dan set $X2=4$

Contoh Forward Checking: 4-Queens Problem



Pilih variabel $X1=1$ dan $X2=4$ ($X2=3$ sudah dibuang), sisa nilai variabel $X3, X4$ dengan forward checking

Contoh Forward Checking: 4-Queens Problem

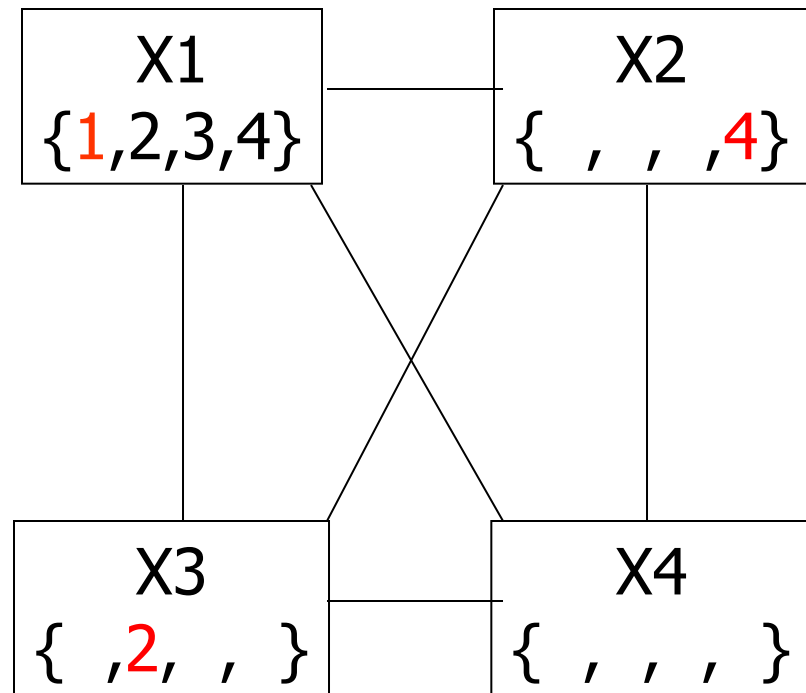


Pilih variabel X1=1 dan X2=4 dan X3=2

Contoh Forward Checking: 4-Queens Problem

	1	2	3	4
1	★	●	●	●
2	■	●	★	●
3	□	●	●	●
4	■	★	●	●

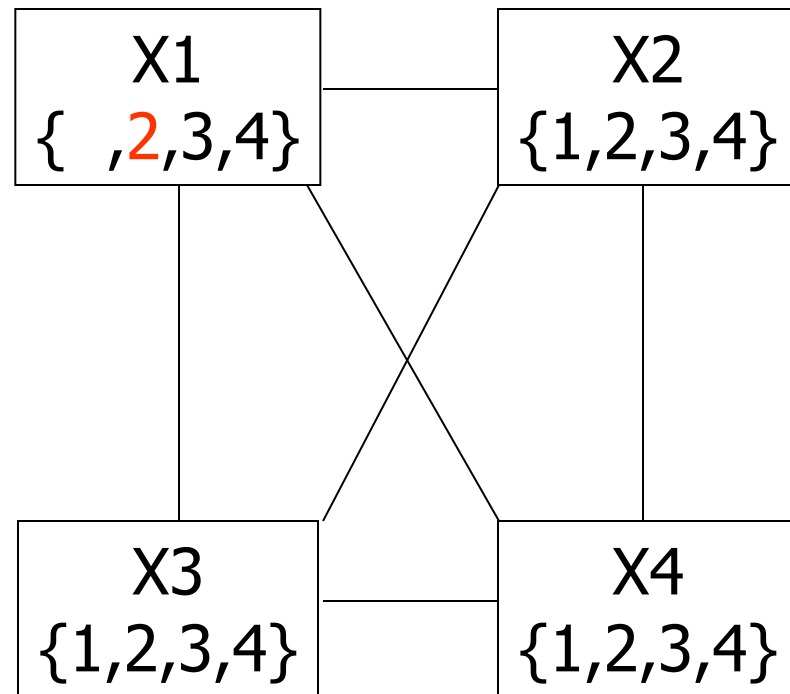
Pilih variabel $X1=1$ dan $X2=4$ dan $X3=2$, membuat variabel $X4$ tidak memiliki nilai ->
Backtrack $X3$ tidak bisa karena saat $X2=4$, nilai $X3=2$;
Backtrack $X2$ tidak bisa karena sudah tdk ada nilainya;
JADI backtrack $X1=2$



Pilih variabel $X1=1$ dan $X2=4$ dan $X3=2$, sisa nilai variabel $X4$ dengan forward checking

Contoh Forward Checking: 4-Queens Problem

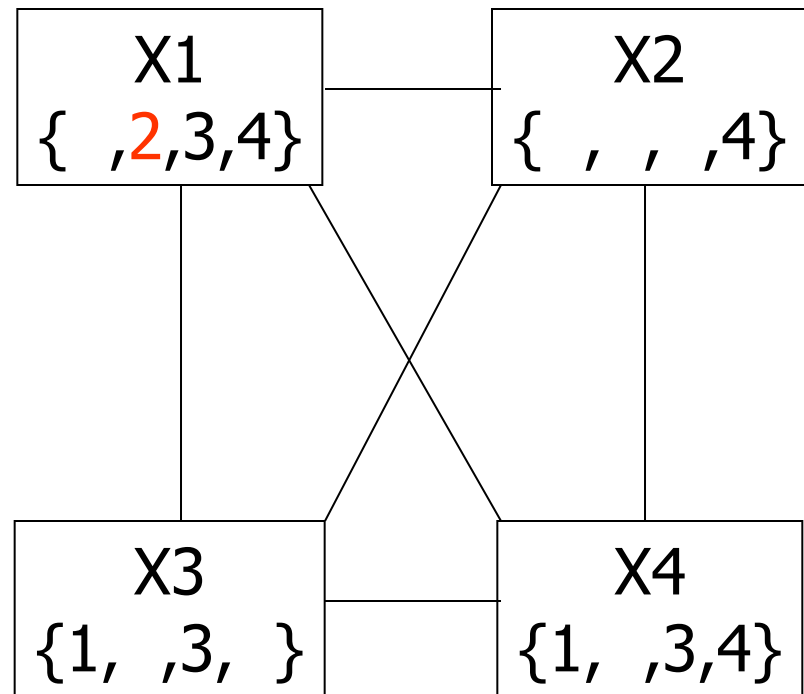
	1	2	3	4
1		●		
2	★	●	●	●
3		●		
4			●	



Pilih variabel $X1=2$, nilai $X1=1$ sudah dibuang karena tidak dapat menghasilkan solusi

Contoh Forward Checking: 4-Queens Problem

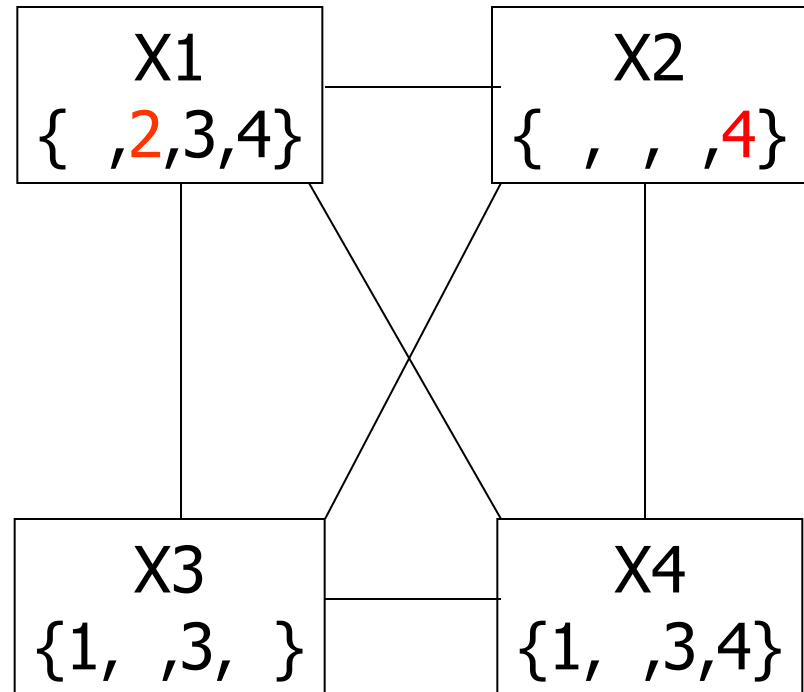
	1	2	3	4
1		●		
2	★	●	●	●
3		●		
4			●	



Pilih variabel X1=2, sisa nilai variabel X2, X3, X4 dengan forward checking

Contoh Forward Checking: 4-Queens Problem

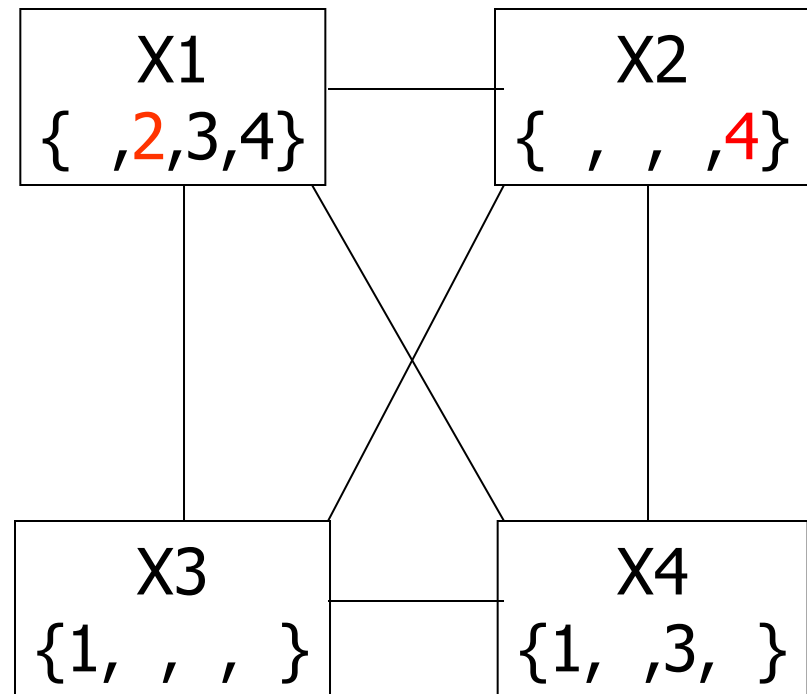
	1	2	3	4
1		●		
2	★	●	●	●
3		●	●	
4		★	●	●



Pilih variabel X1=2 dan X2=4

Contoh Forward Checking: 4-Queens Problem

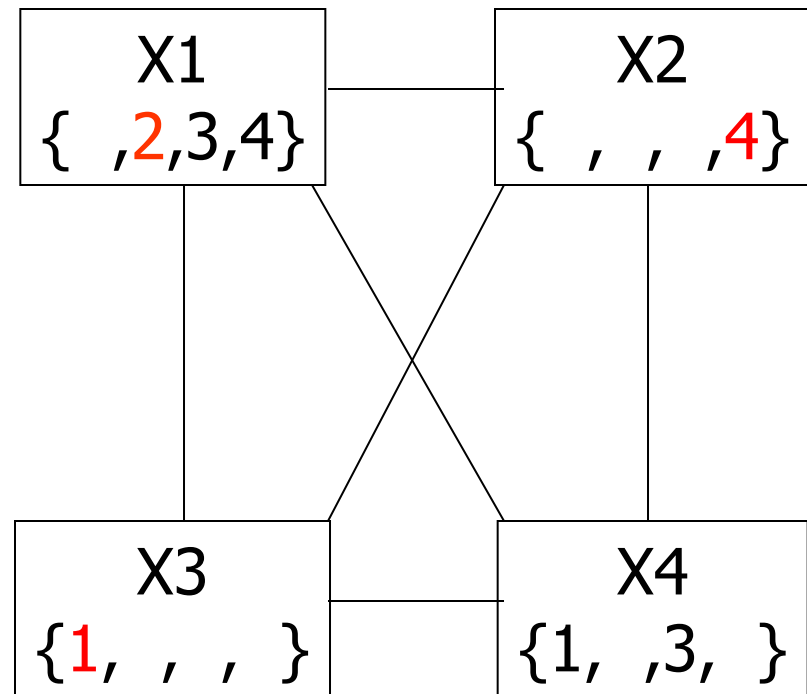
	1	2	3	4
1		●		
2	★	●	●	●
3		●	●	
4		★	●	●



Pilih variabel X1=2 dan X2=4,
sisa nilai variabel X3, X4 dengan forward checking

Contoh Forward Checking: 4-Queens Problem

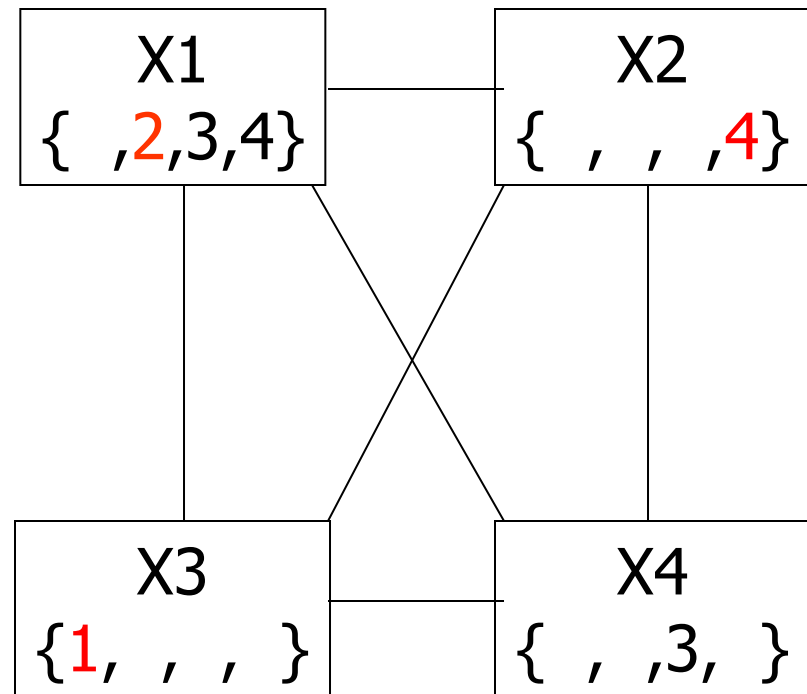
	1	2	3	4
1		●	★	●
2	★	●	●	●
3		●	●	
4		★	●	●



Pilih variabel X1=2 dan X2=4 dan X3=1

Contoh Forward Checking: 4-Queens Problem

	1	2	3	4
1		●	★	●
2	★	●	●	●
3		●	●	
4		★	●	●



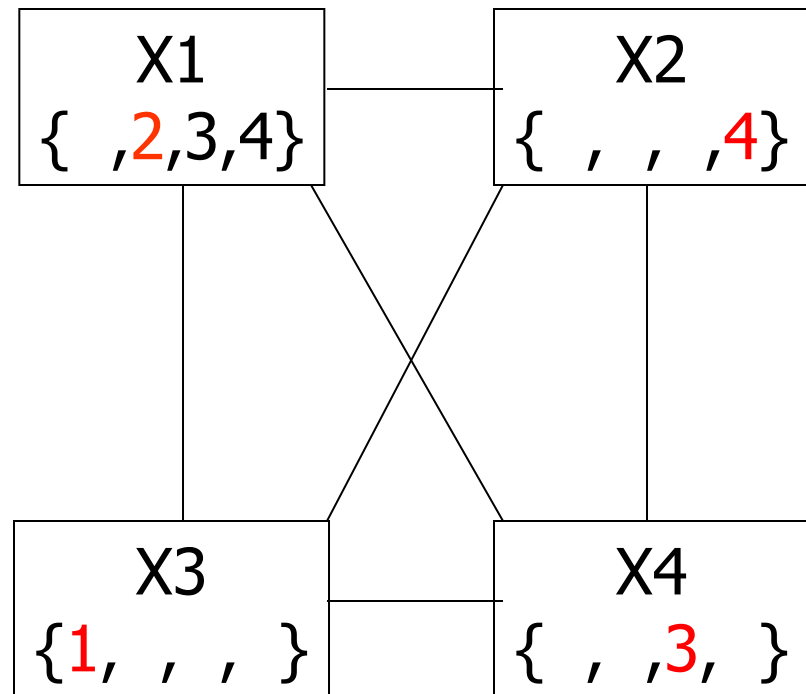
Pilih variabel $X1=2$ dan $X2=4$ dan $X3=1$,
sisa nilai variabel $X4$ dengan forward checking

Contoh Forward Checking: 4-Queens Problem

	1	2	3	4
1		●	★	●
2	★	●	●	●
3		●	●	★
4		★	●	●

Pilihan solusi lain dapat dilakukan dengan mencoba

$X1=\{3, 4\}$



Pilih variabel $X1=2$ dan $X2=4$ dan $X3=1$ dan $X4=3$; karena sudah tidak ada nilai lain $X4$ maka solusi ditemukan

Pseudocode Backtracking Search (BT)

CATATAN:

$BT(A, U)$

if A is complete then

 return A

end if

Remove a variable X from U

for all values $x \in D(X)$ do

 if $X = x$ is consistent with A according to the constraints then

 Add $X = x$ to A

$result \leftarrow BT(A, U)$

 if $result \neq failure$ then

 return $result$

 end if

 Remove $X = x$ from A

 end if

end for

return $failure$

A : variabel CSP yang nilainya sudah diset

U : variabel CSP yang nilainya belum diset

D : domain nilai variabel yang belum diset

Pseudocode untuk Backtracking Search (BT) dengan Forward Checking (FC)

BT+FC(A, U, D)

if A is complete then

 return A

end if

Remove a variable X from U

for all values $x \in D(X)$ do

 if $X = x$ is consistent with A according to the constraints then

 Add $X = x$ to A

$D' \leftarrow D$ (Save the current domains)

 for all $Y \in U$ (i.e., Y an unassigned variable), $Y - - - X$ (i.e., Y a neighbor of X in the constrained graph) do

 Remove values for Y from $D'(Y)$ that are inconsistent with A

 end for

 if for all $Y \in U, Y - - - X$, we have $D'(Y)$ not empty then

$result \leftarrow \text{BT+FC}(A, U, D')$

 if $result \neq failure$ then

 return $result$

 end if

 end if

 Remove $X = x$ from A

 end if

end for

return *failure*

CATATAN:

A: variabel CSP yang nilainya sudah diset

U: variabel CSP yang nilainya belum diset

D: domain nilai variabel yang belum diset

Algoritma CSP 2: constraint propagation

Pendekatan ARC CONSISTENCY

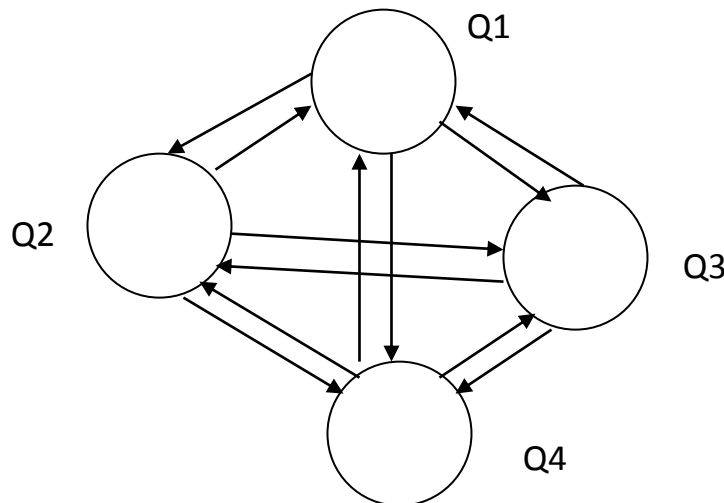
Terdapat constraint graph **G** berisi variabel X_1, \dots, X_n , dengan constraint $\{X_i \rightarrow X_j\}$, dan setiap X_i memiliki set nilai $\mathbf{v}(X_i)$. Arc $X_i \rightarrow X_j$ akan konsisten jika

$$\forall v \in \mathbf{v}(X_i) \exists w \in \mathbf{v}(X_j) \wedge v, w \text{ is consistent}$$

Arc $X_i \rightarrow X_j$ akan **tidak** konsisten jika

$$\exists v \in \mathbf{v}(X_i) \forall w \in \mathbf{v}(X_j) \Rightarrow v, w \text{ is inconsistent.}$$

Untuk membuat arc menjadi konsisten, maka ada nilai v yang harus dibuang.



Representasi *complete constraint graph* untuk 4 variabel problem 4-Queens

Contoh Arc Consistency (4 queens problem)

Hilangkan nilai v pada variabel Q_x jika terdapat nilai Q_y yang membuat v inconsistent dengan semua nilai Q_y

4	-		5	9
3	-		4	8
2	-	2		7
1	○	1	3	6
	Q1	Q2	Q3	Q4

Angka 1-9 menunjukan urutan penghilangan nilai pada domain variabel

Start $Q_1=1$, arc consistency cek nilai yang inconsistent tanpa melakukan set nilai di variabel Q_2 , Q_3 , Q_4

Contoh Arc Consistency (4 queens problem)

Hilangkan nilai v pada variabel Q_x jika terdapat nilai Q_y yang membuat v inconsistent dengan semua nilai Q_y

4	-		6	9
3	-	3	5	
2	○	2	4	8
1	-	1		7
	Q1	Q2	Q3	Q4

Angka 1-9 menunjukan urutan penghilangan nilai pada domain variabel

Start $Q_1=2$, arc consistency cek nilai yang inconsistent tanpa melakukan set nilai di variabel Q_2 , Q_3 , Q_4

Constraint Propagation:

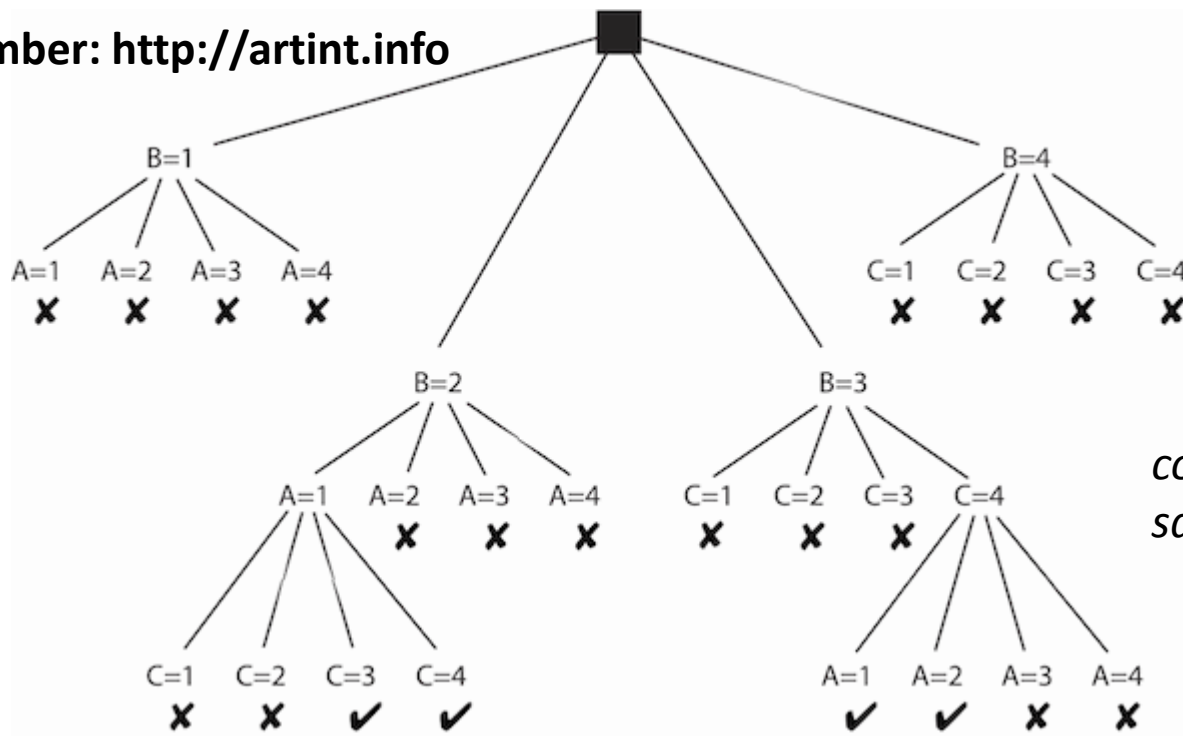
forward checking (FC) vs arc consistency (AC)

- FC memberikan solusi dengan waktu yang lebih cepat dibanding AC
- AC membutuhkan waktu lebih lama namun melakukan *pruning* lebih efektif pada *state space*
- Ada banyak pendekatan *consistency* lain yang memberikan solusi lebih baik namun waktu lebih lama (sumber: AIMA)
 - Node consistency, Path consistency, dll

Contoh Problem dengan Solusi AC

- Terdapat problem CSP dengan variabel A, B, C
- Domain nilai setiap variabel = $\{1,2,3,4\}$
- Constraint yang ada : $A < B$ and $B < C$

sumber: <http://artint.info>



contoh tree untuk salah satu solusi