دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

مسائل ارضای محدودیت

«هوش مصنوعی: یک رهیافت نوین»، فصل ۶

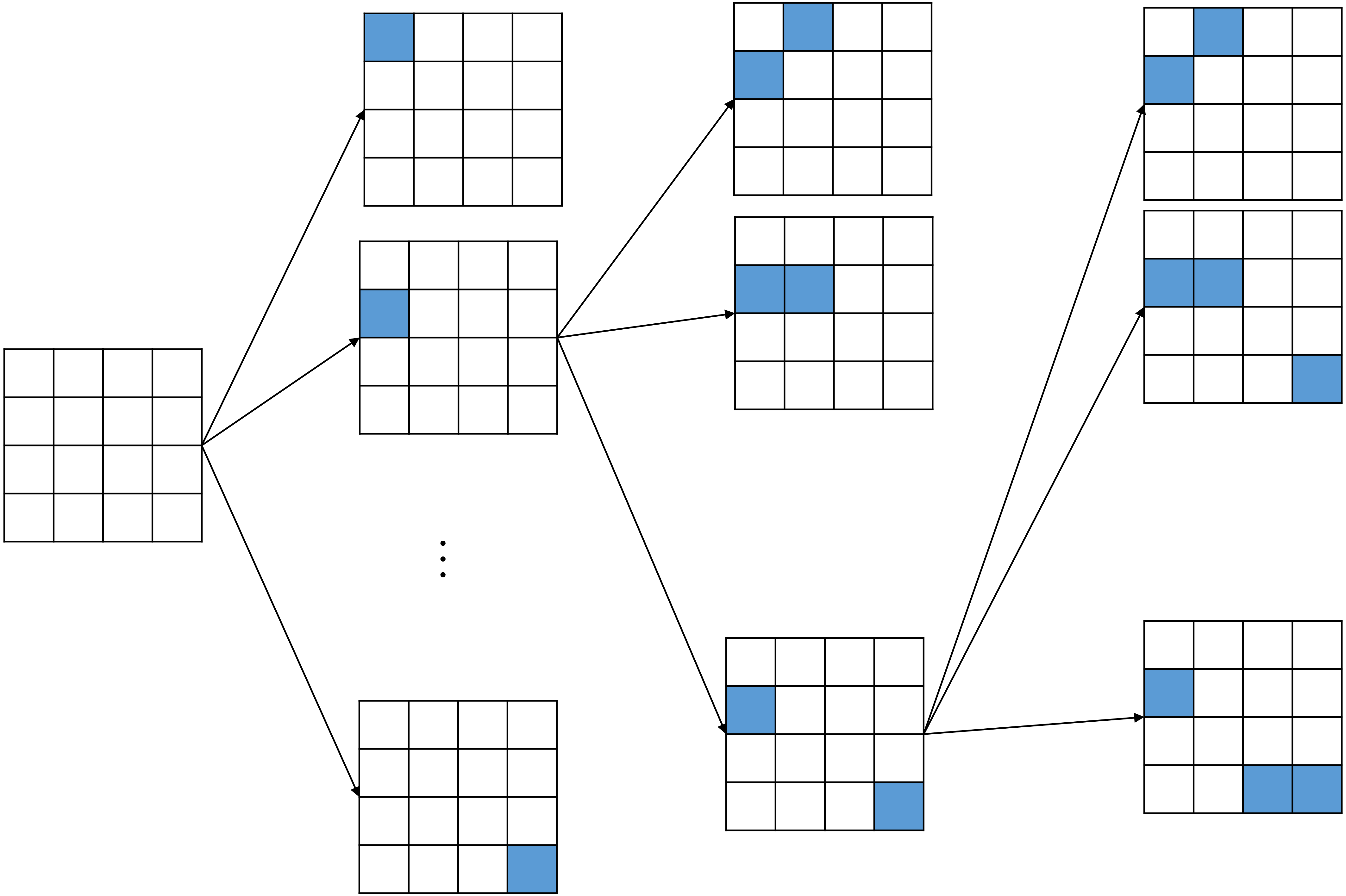
ارائه‌دهنده: سیده فاطمه موسوی

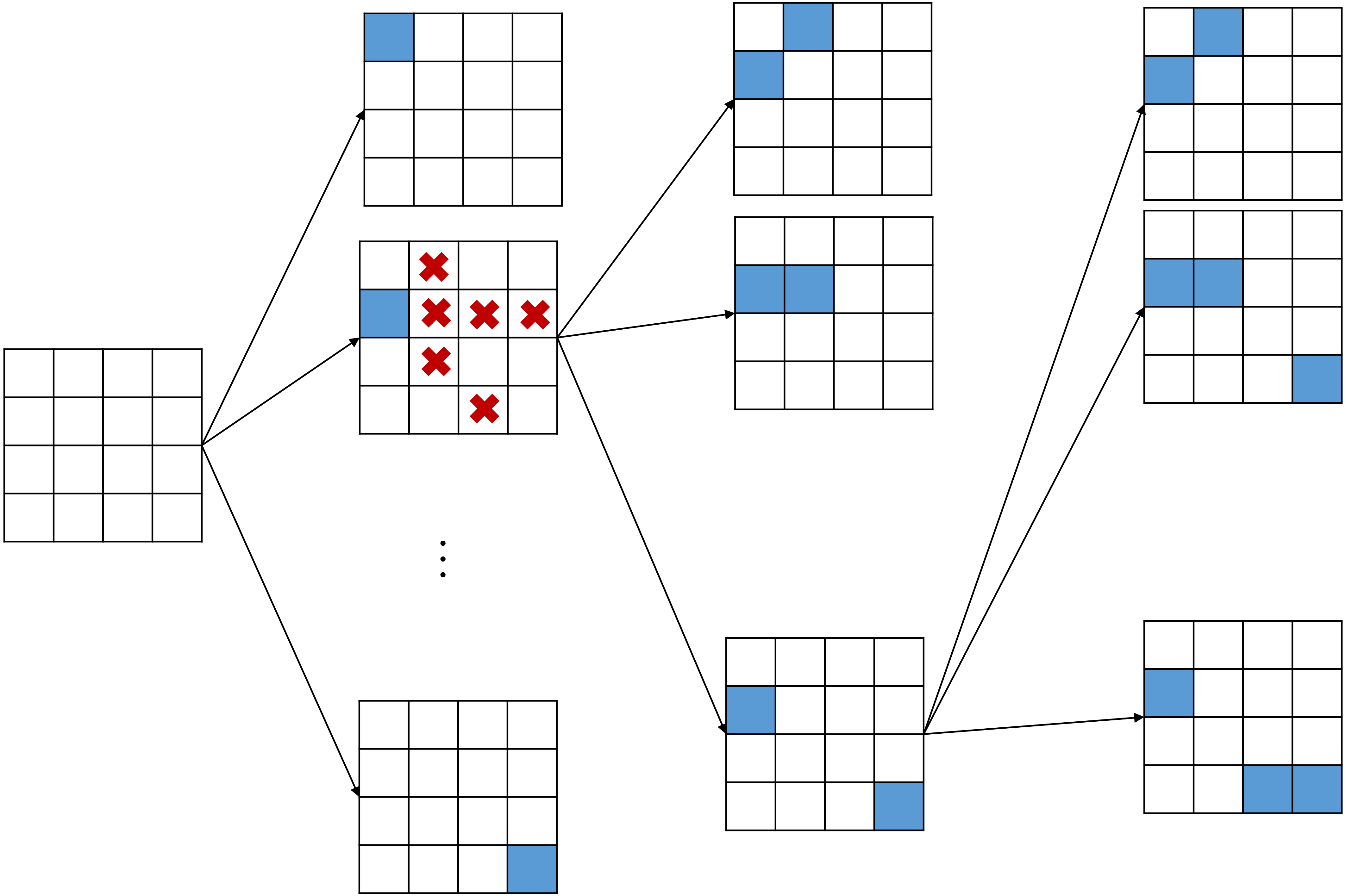
نیم‌سال اول ۱۴۰۰–۱۳۹۹

رئوس مطالب

- تعریف مسائل ارضای محدودیت (Constraint Satisfaction Problems)
- انتشار محدودیت: استنتاج در CSPها
- جستجوی عقب‌گرد برای CSPها
- جستجوی محلی در CSPها
- ساختار مسائل

تعریف مسائل ارضای محدودیت





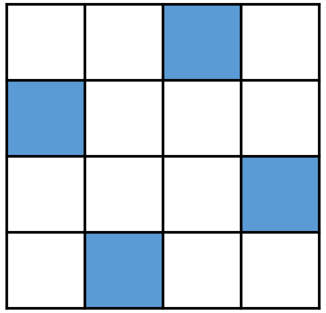
تعریف مسائل ارضای محدودیت

- یک مسئله‌ی ارضای محدودیت شامل سه جزء زیر است:
  - مجموعه‌ای از **متغیرها** $X=\{X_1,X_2,\dots, X_n\}$
  - مجموعه‌ای از **دامنه‌ها** $D=\{D_1,D_2,\dots, D_n\}$
    - هر دامنه‌ی D_i شامل مجموعه‌ای از مقادیر مجاز $\{v_1,\dots,v_k\}$ برای متغیر X_i است.
  - مجموعه‌ای از **محدودیت‌ها** $C=\{C_1,C_2,\dots, C_m\}$
    - هر محدودیت C_i یک زوج <scope, rel> است که scope چندتایی است که در محدودیت شرکت می‌کند و rel رابطه‌ای است که مقادیری که متغیرها می‌توانند بگیرند را تعریف می‌کند.
    - برای مثال اگر بخواهیم رابطه‌ی نابرابری بین دو متغیر که هر دو دامنه‌ی $\{A, B\}$ هستند را نشان دهیم یکی از دو روش زیر را می‌توانیم به کار ببریم:

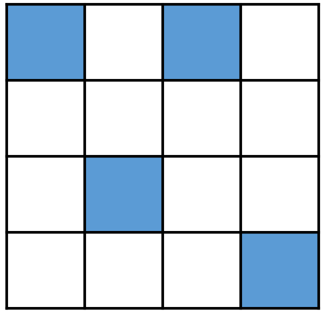
$$<(X_1, X_2), X_1 \neq X_2> \qquad <(X_1, X_2), [(A, B), (B, A)]>$$

تعریف مسائل ارضای محدودیت ...

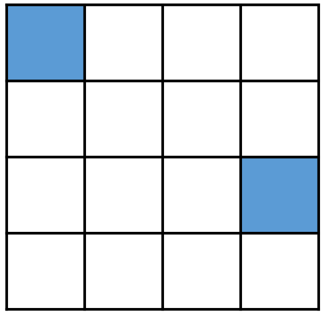
- برای حل یک مسئله ارضای محدودیت می‌بایست فضای حالت و راه‌حل تعریف شود.
- فضای حالت: هر حالت یک انتساب مقادیر به برخی یا همه متغیرها است
  - $\{X_i=v_i, X_j=v_j,\ldots\}$ (انتساب جزئی)
- راه‌حل (هدف): یک انتساب **کامل** و **سازگار** از مقادیر به متغیرها
  - سازگار: انتسابی که هیچ یک از محدودیت‌ها را نقض نمی‌کند.
  - کامل: به هر یک از متغیرها مقداری نسبت داده شده باشد.



$\{X_1=2, X_2=4, X_3=1, X_4=3\}$

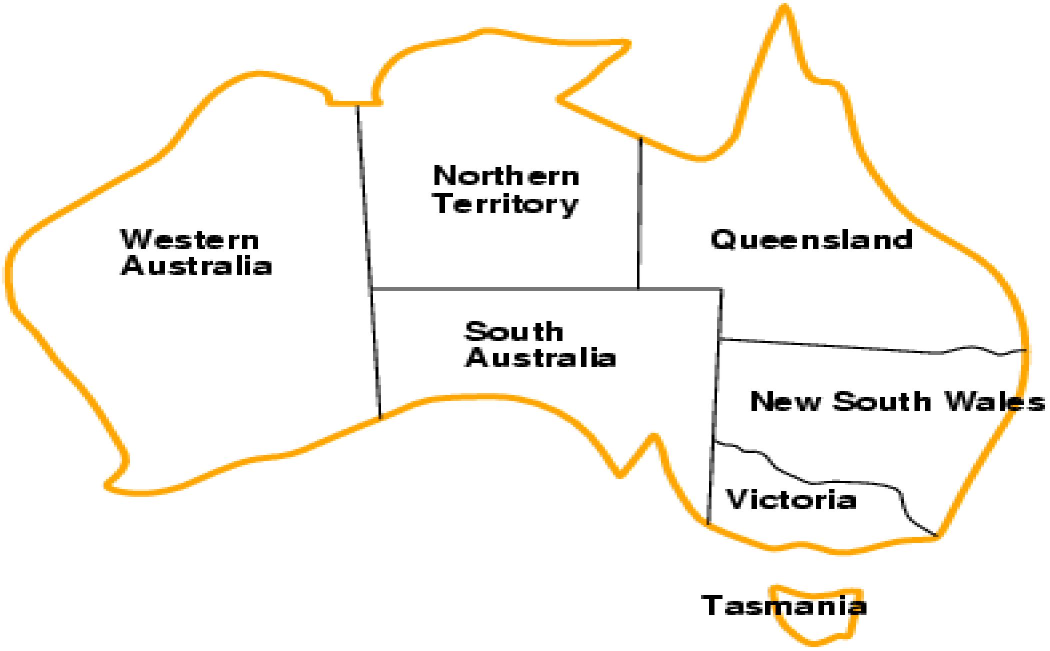


$\{X_1=1, X_2=3, X_3=1, X_4=4\}$



$\{X_1=1, X_4=3\}$

مثال: رنگ آمیزی نقشه



- تعریف مسئله رنگ‌آمیزی نقشه
  - متغیرها: هر یک از ناحیه‌ها

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

  - دامنه‌ها: سه رنگ قرمز، سبز و آبی

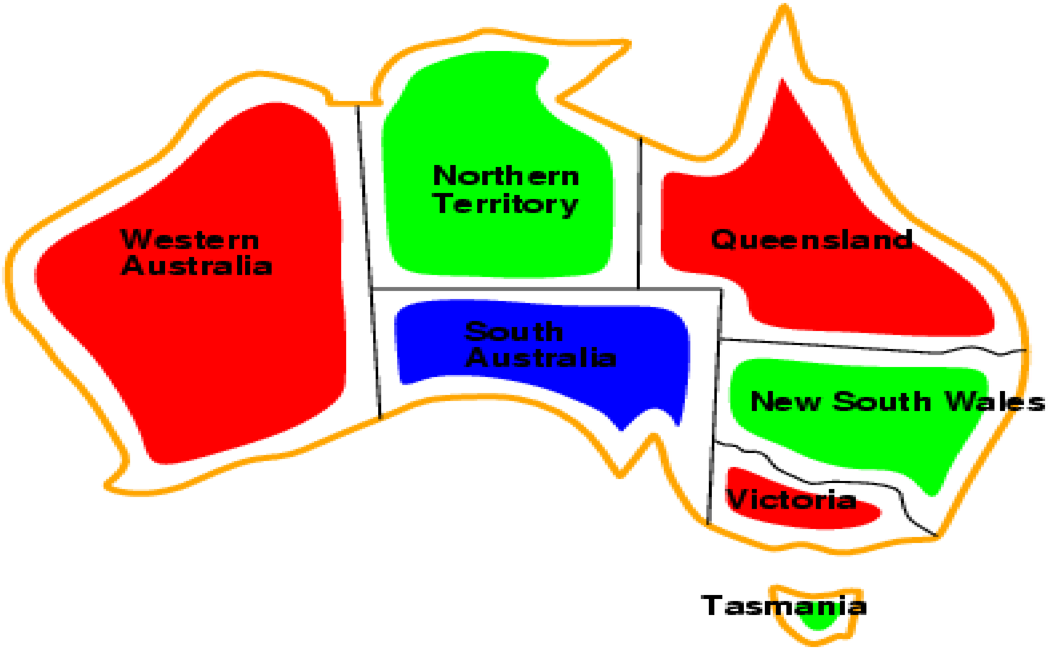
$D_i = \{red, green, blue\}$

  - محدودیت‌ها: نواحی همسایه باید رنگ متفاوتی داشته باشند.

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

مثال: رنگ آمیزی نقشه

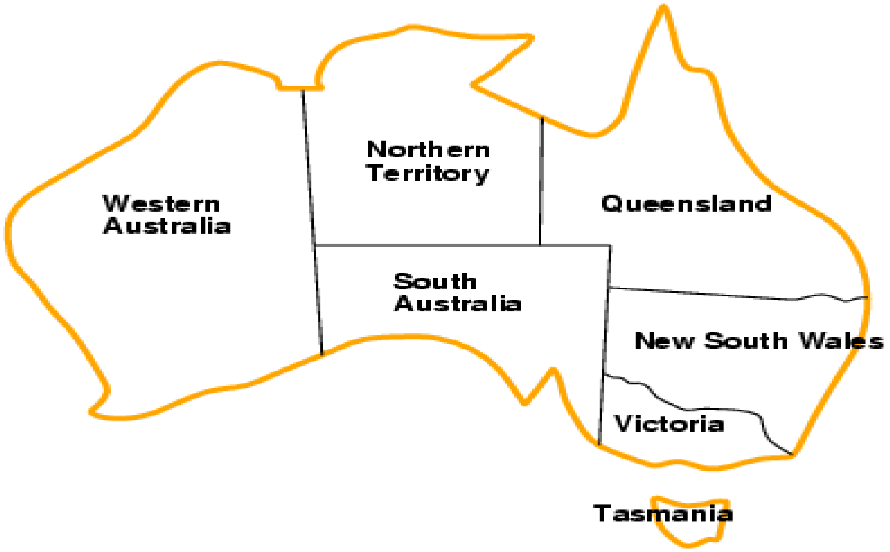
- راه‌حل مسئله‌ی رنگ‌آمیزی
  - یک انتساب کامل و سازگار برای مثال

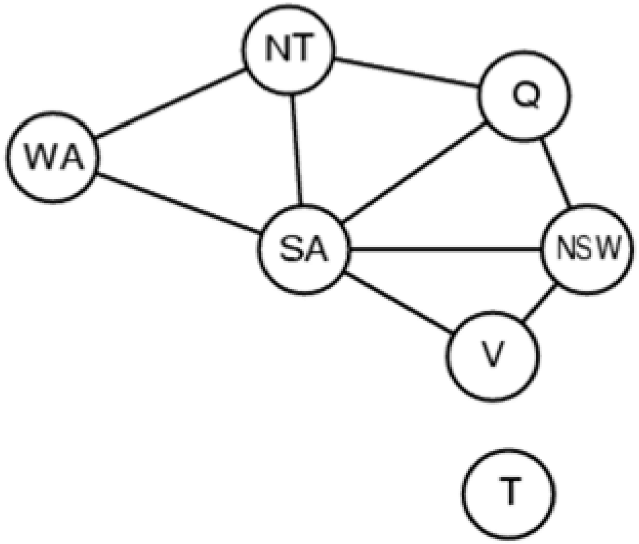


{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green}

گراف محدودیت

- گراف محدودیت: گرافی که گره‌ها در آن نشان‌دهنده‌ی متغیرها و یال‌ها نشان‌دهنده‌ی محدودیت‌های بین متغیرها است.



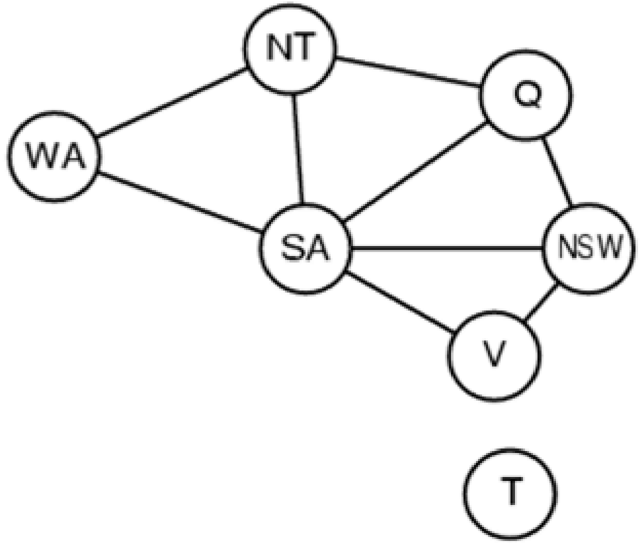


دلایل استفاده از CSPها

- CSPها یک نمایش طبیعی برای رنج گسترده‌ای از مسائل ارائه می‌کنند.
  - اغلب حل مسئله با استفاده از CSPها آسان‌تر است.
- به دلیل نمایش استاندارد حالت‌ها (مجموعه متغیرها با مقدارشان)، می‌توان تابع مابعد و آزمون هدف را به شکل کلی نوشت به طوریکه برای هر مسئله CSP قابل اعمال باشد.
- می‌توان هیوریستیک‌های کلی و کارایی ایجاد کرد که برای تمام مسائل CSP قابل استفاده باشند.
  - طراحی این هیوریستیک‌ها نیاز به تخصص اضافی در دامنه خاص مسأله ندارد.

دلایل استفاده از CSPها ...

- حل‌کننده‌های CSP می‌توانند به سرعت بخش بزرگی از فضای حالت را حذف کنند.
  - برای مثال در رنگ‌آمیزی نقشه با انتخاب {SA=blue} می‌توان نتیجه گرفت که هیچ یک از ۵ متغیر همسایه نمی‌تواند رنگ آبی داشته باشد.
  - بدون مزیت بردن از انتشار محدودیت تعداد انتساب ۵ متغیر دیگر $۳^{۵}$=۲۴۳ خواهد بود.
  - با انتشار محدودیت این تعداد برابر با $۲^{۵}$=۳۲ می‌شود. (۸۷٪ کاهش)
- بسیاری از مسائلی که برای جستجوی فضای حالت منظم، رام‌نشدنی (intractable) هستند می‌توانند به سرعت هنگامی که به یک CSP فرموله می‌شوند حل گردند.
  - هنگامی که متوجه شدیم یک انتساب جزئی منجر به یک راه‌حل نمی‌شود از ادامه دادن و مقداردهی بیشتر این انتساب خودداری می‌کنیم.



مثال: برنامه‌ریزی وظایف

- یکی از مسائل رایج کارخانه‌ها، برنامه‌ریزی وظایف روزانه‌ی آن‌ها است.
- برای مثال: مسئله‌ی برنامه‌ریزی اسمبل کردن یک ماشین
  - متغیرها: کل کار از وظایفی تشکیل شده است و می‌توان هر وظیفه را به‌عنوان یک متغیر مدل نمود.
  - دامنه‌ها: مقدار هر متغیر زمان شروع وظیفه است. (یک عدد برحسب دقیقه)
  - محدودیت‌ها: یک وظیفه باید قبل از وظیفه دیگر انجام شود (برای مثال یک چرخ باید قبل از قرار گرفتن قالپاق نصب شود)
    - وظایف زیادی می‌توانند به صورت هم‌زمان اجرا شوند.
    - یک وظیفه یک مقدار زمان مشخص برای تکمیل شدن نیاز دارد.

مثال: برنامه‌ریزی وظایف ...

- فرض کنید این مسئله دارای ۱۵ وظیفه زیر است
  - نصب محورهای جلو و عقب
  - اضافه کردن هر چهار چرخ (جلو، عقب، راست و چپ)
  - جای دادن مهره‌ها برای هر چهار چرخ
  - اضافه کردن قالپاق
  - بازرسی مونتاژ نهایی

$$\begin{aligned} X = \{ & Axle_F, Axle_B, Wheel_{RF}, Wheel_{LF}, Wheel_{RB}, Wheel_{LB}, Nuts_{RF}, \\ & Nuts_{LF}, Nuts_{RB}, Nuts_{LB}, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect\} \end{aligned}$$

مثال: برنامه‌ریزی وظایف ...

- هرگاه وظیفه T_1 باید قبل از وظیفه‌ی T_2 انجام شود و T_1 برای تکمیل شدن به d_1 زمان احتیاج داشته باشد، یک محدودیت حسابی به شکل $T_1+d_1 \leq T_2$ خواهیم داشت.

$$\begin{array}{ll}
Axle_F + 10 \leq Wheel_{RF}; & Axle_F + 10 \leq Wheel_{LF}; \\
Axle_B + 10 \leq Wheel_{RB}; & Axle_B + 10 \leq Wheel_{LB} \\
Wheel_{RF} + 1 \leq Nuts_{RF}; & Nuts_{RF} + 2 \leq Cap_{RF}; \\
Wheel_{LF} + 1 \leq Nuts_{LF}; & Nuts_{LF} + 2 \leq Cap_{LF}; \\
Wheel_{RB} + 1 \leq Nuts_{RB}; & Nuts_{RB} + 2 \leq Cap_{RB}; \\
Wheel_{LB} + 1 \leq Nuts_{LB}; & Nuts_{LB} + 2 \leq Cap_{LB}\ .
\end{array}$$

- فرض می‌کنیم چهار کارگر برای نصب چرخ‌ها داریم، اما تنها یک ابزار برای نصب محور دارند. (ترکیب حسابی و منطقی محدودیت‌ها)

$$(Axle_F + 10 \leq Axle_B) \quad \textbf{or} \quad (Axle_B + 10 \leq Axle_F)$$

مثال: برنامه‌ریزی وظایف ...

- بازرسی بعد از تمام وظایف انجام می‌شود و ۳ دقیقه طول می‌کشد. پس برای هر متغیر به جز Inspect یک محدودیت به شکل $X+d_X \leq Inspect$ اضافه می‌کنیم.
- اگر یک شرط وجود داشته باشد که کل اسمبل کردن باید در ۳۰ دقیقه انجام شود، می‌توان با محدود کردن دامنه‌ی تمام متغیرها به این شرط رسید.

$$D_i=\{1,2,3,\ldots,27\}$$

انواع متغیرهای CSP

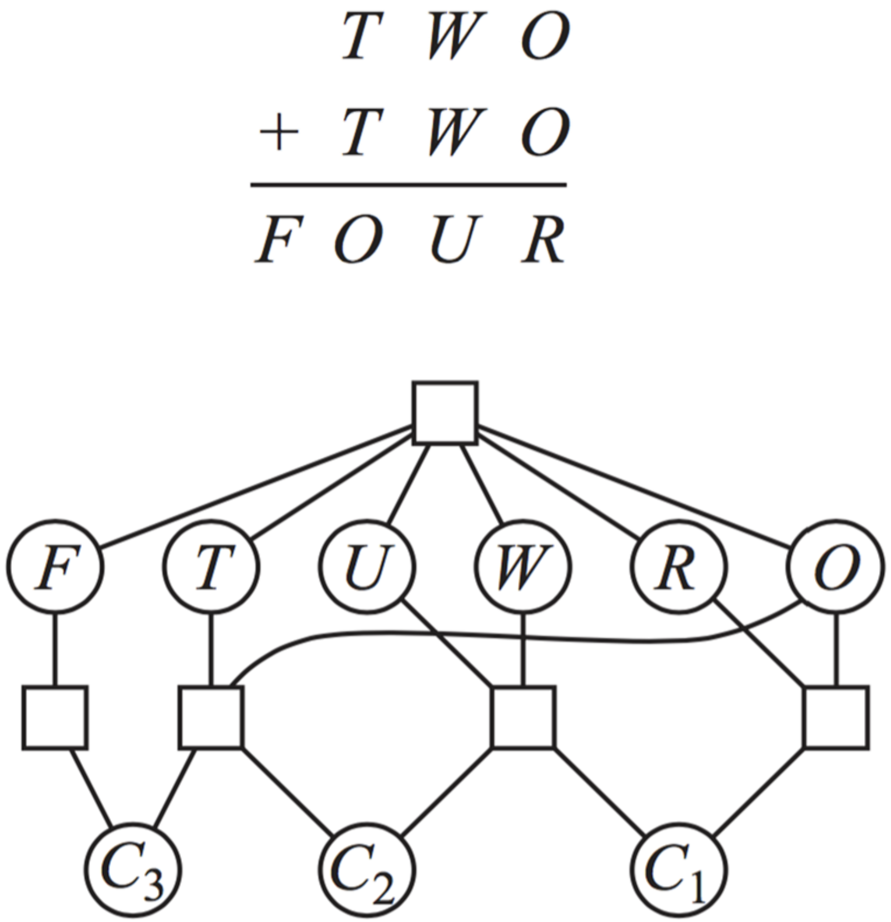
- متغیرهای گسسته
  - دامنه‌های محدود
    - n متغیر، اندازه هر دامنه d: تعداد انتساب های کامل $O(d^n)$
    - مثال: ۸ وزیر ، رنگ‌آمیزی نقشه و ...
  - دامنه‌های نامحدود
    - اعداد صحیح، رشته‌ها و ... مثال: در برنامه‌ریزی وظایف اگر deadline قرار نمی‌دادیم تعداد نامتناهی زمان شروع برای هر متغیر وجود داشت.
    - نیاز به زبان محدودیت دارند. مثال: $StartJob1 + 5 \leq StartJob3$
- متغیرهای پیوسته
  - مثال: زمان‌های شروع و پایان مشاهدات تلسکوپ فضایی هابل
  - اگر محدودیت‌ها تنها خطی باشند آن‌گاه مسئله قابل حل با برنامه‌ریزی خطی در زمان چندجمله‌ای نسبت به تعداد متغیرها خواهد بود.

انواع محدودیت‌ها از نظر تعداد متغیرها

- یگانی (unary) باعث محدود شدن مقدار یک متغیر منفرد می‌شود.
  - مثال: $<(SA), SA \neq green>$
- دوگانی (binary) محدودیت شامل یک زوج از متغیرها می‌باشد،
  - مثال: $<(SA, WA), SA \neq WA>$
- محدودیت مرتبه بالاتر(Higher-order) شامل سه یا بیشتر متغیر است.
  - مثال: مقدار Y بین X و Z است Between(X, Y, Z)
- محدودیت سراسری شامل تعداد دلخواه از متغیرها است.
  - مثال: Alldiff یعنی تمام متغیرهای موجود در محدودیت باید مقادیر متفاوت داشته باشند.

مثال: مسئله رمزنگاری

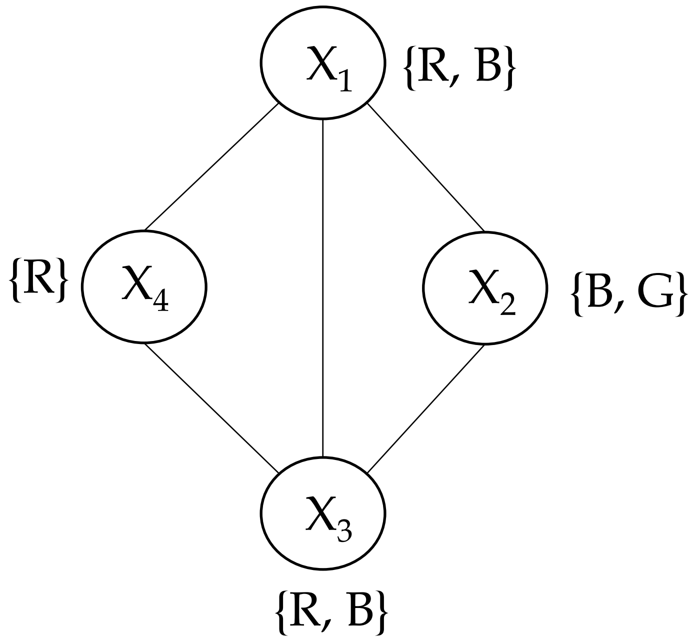
- Variables: $F\ T\ U\ W\ R\ O\ C_1\ C_2\ C_3$
- Domains: $\{0,1,2,3,4,5,6,7,8,9\}$
- Constraints: $Alldiff\ (F,T,U,W,R,O)$
  - $O + O = R + 10 \cdot C_1$
  - $C_1 + W + W = U + 10 \cdot C_2$
  - $C_2 + T + T = O + 10 \cdot C_3$
  - $C_3 = F,\ T \neq 0,\ F \neq 0$

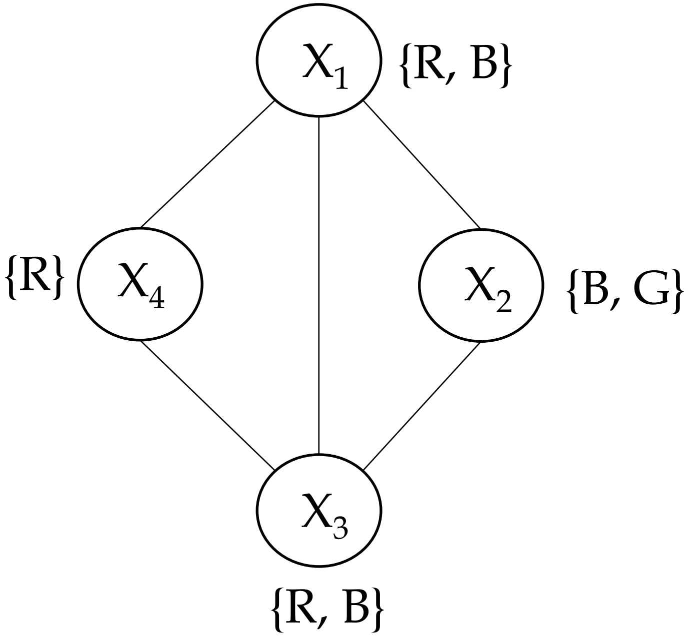


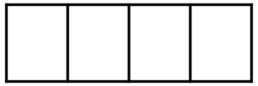
انواع محدودیت‌ها از نظر اولویت

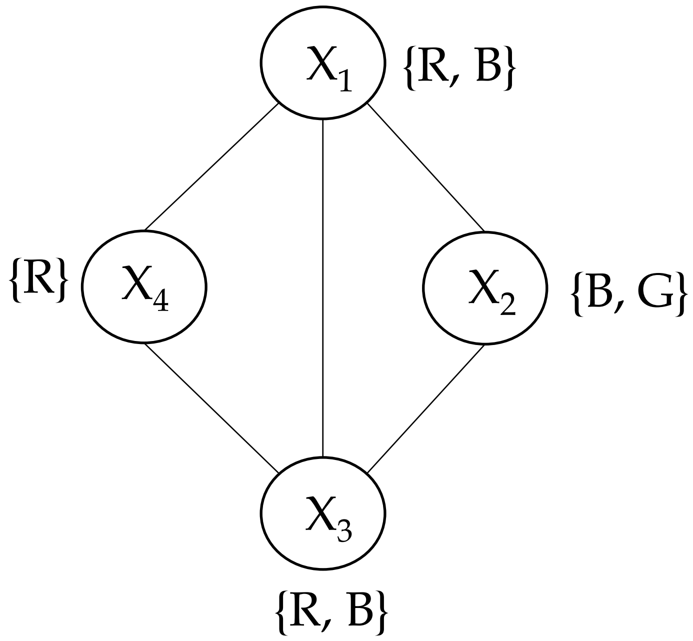
- محدودیت مطلق: محدودیت‌هایی که نباید هیچ یک از آن‌ها در راه‌حل نقض شود.
  - مثال: یک استاد نمی‌تواند به‌طور هم‌زمان در دو کلاس درس دهد.
- محدودیت اولویت‌دار: نشان می‌دهد کدام راه‌حل ارجح است.
  - مثال: استاد X کلاس‌های صبح را ترجیح می‌دهد و استاد Y کلاس‌های عصر را.
  - اگر برنامه‌ی زمانی داشته باشیم که برای استاد X در عصر کلاس گذاشته شده باشد باز هم یک راه‌حل خواهد بود اگر چه راه‌حل بهینه نیست.
- محدودیت‌های اولویت می‌توانند به‌صورت هزینه‌هایی بر روی انتساب هر یک از متغیرها کد شوند. با این فرموله‌بندی، CSP اولویت را می‌توان با روش‌های جستجوی بهینه‌سازی مبتنی‌بر مسیر یا محلی حل کرد.
  - کلاس عصر برای استاد X دو امتیاز منفی داشته باشد در حالی که کلاس صبح یک امتیاز مثبت.

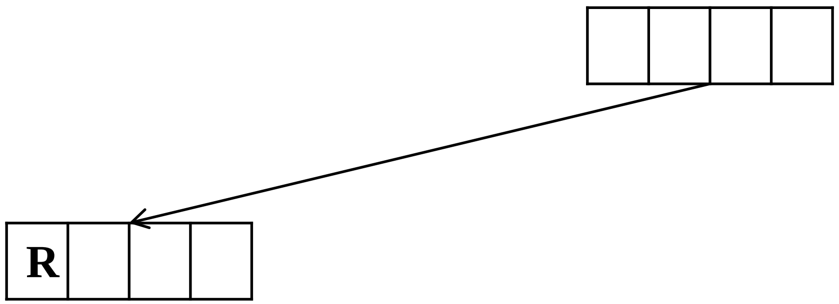
انتشار محدودیت: استنتاج در CSPها

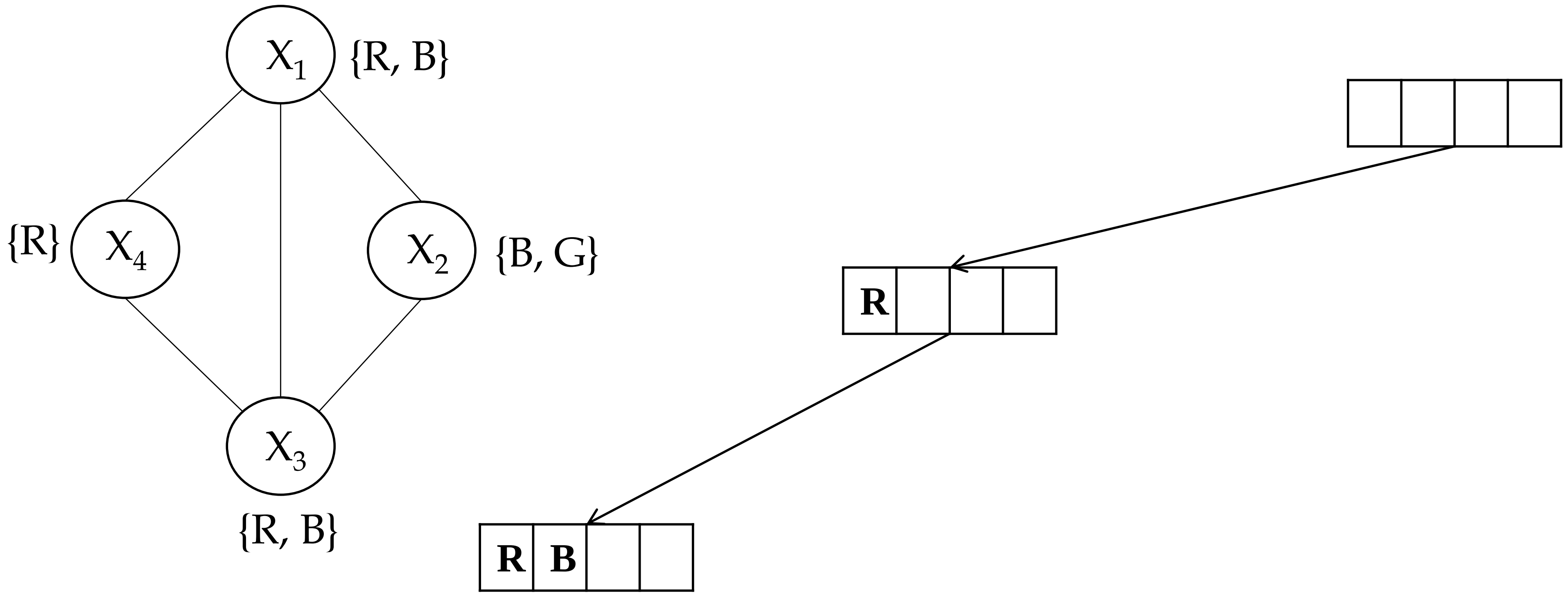

X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}


X_1
{R, B}
{R}
X_4
X_2
{B, G}
X_3
{R, B}

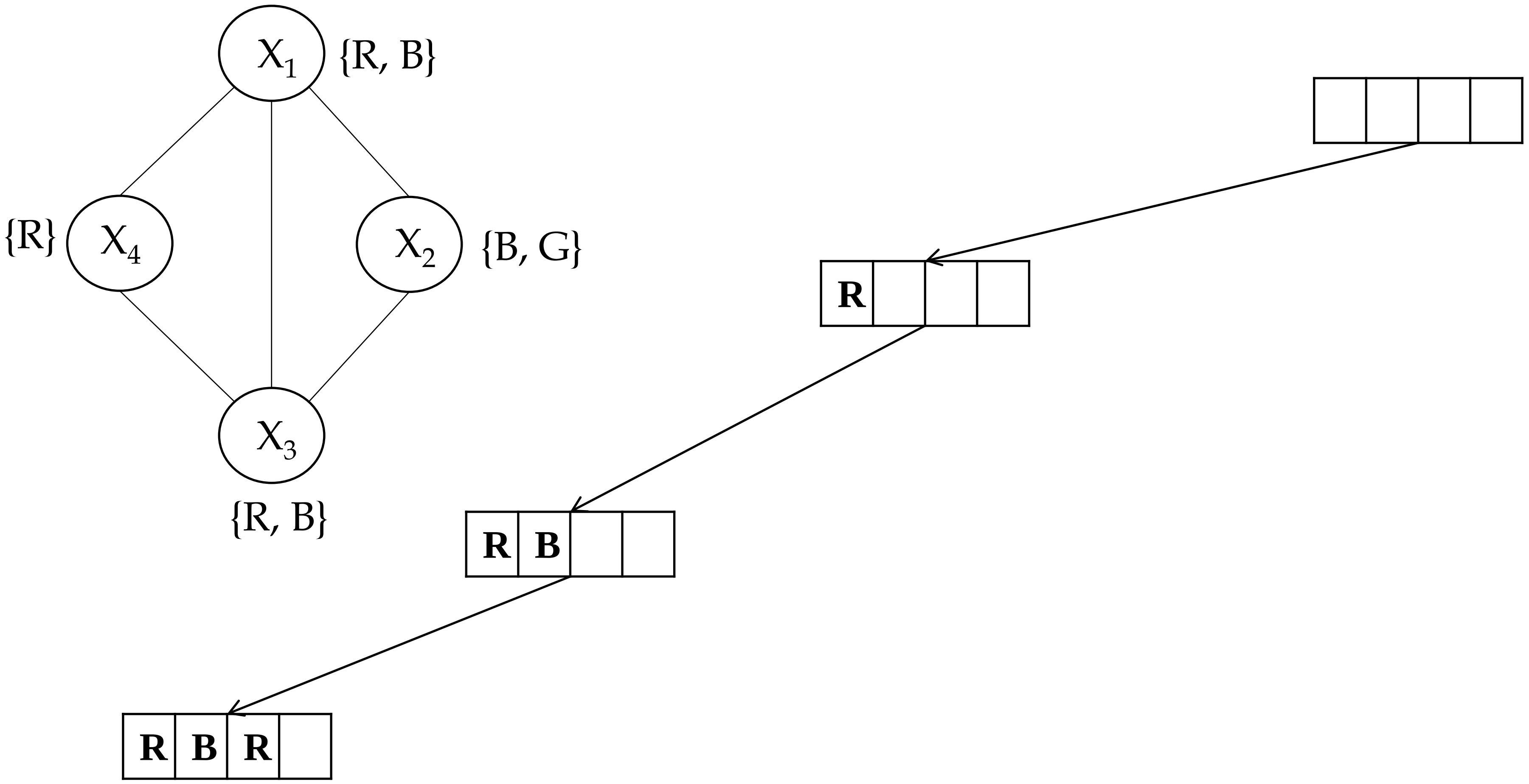



X1
{R, B}
{R}
X4
X2
{B, G}
X3
{R, B}

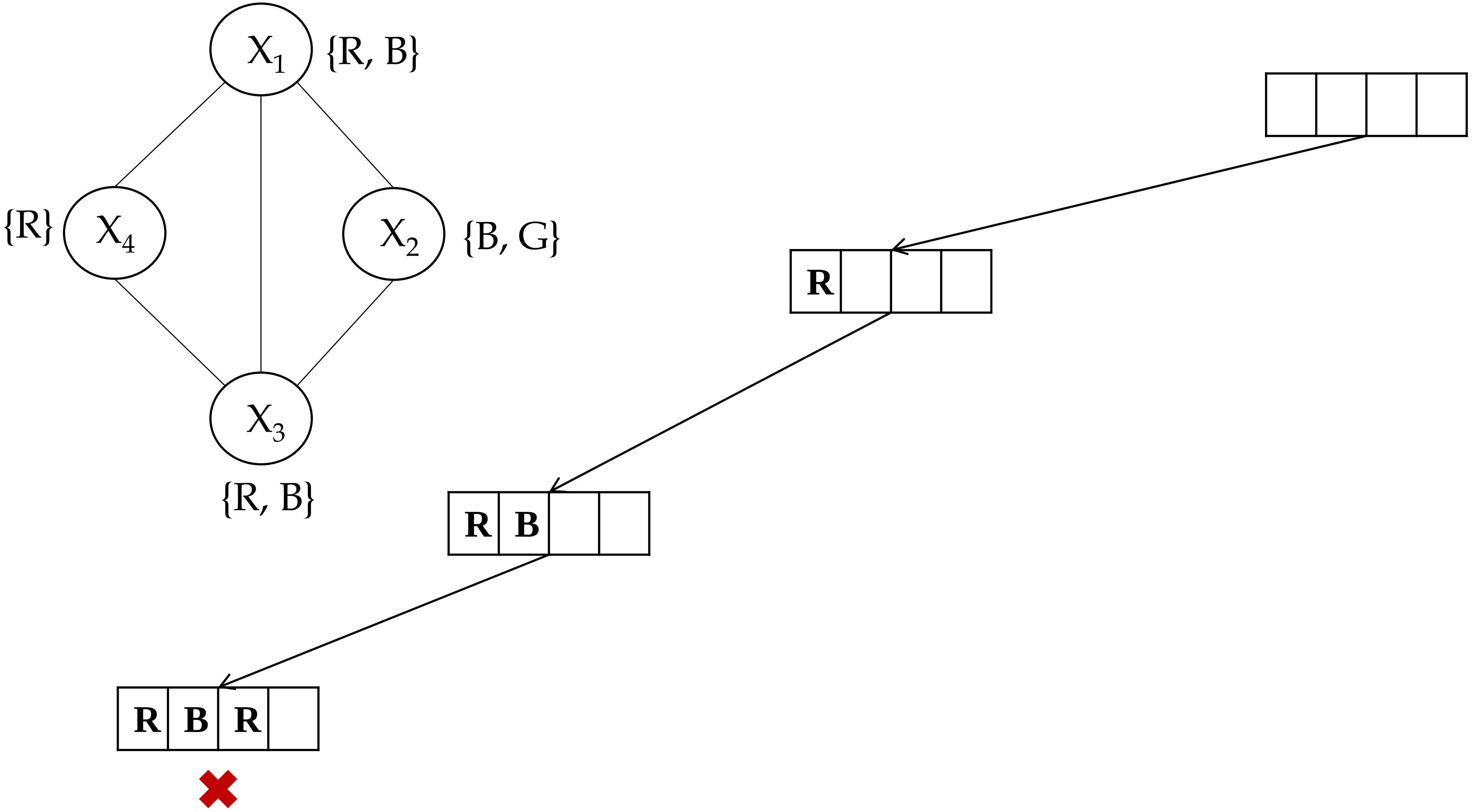

R

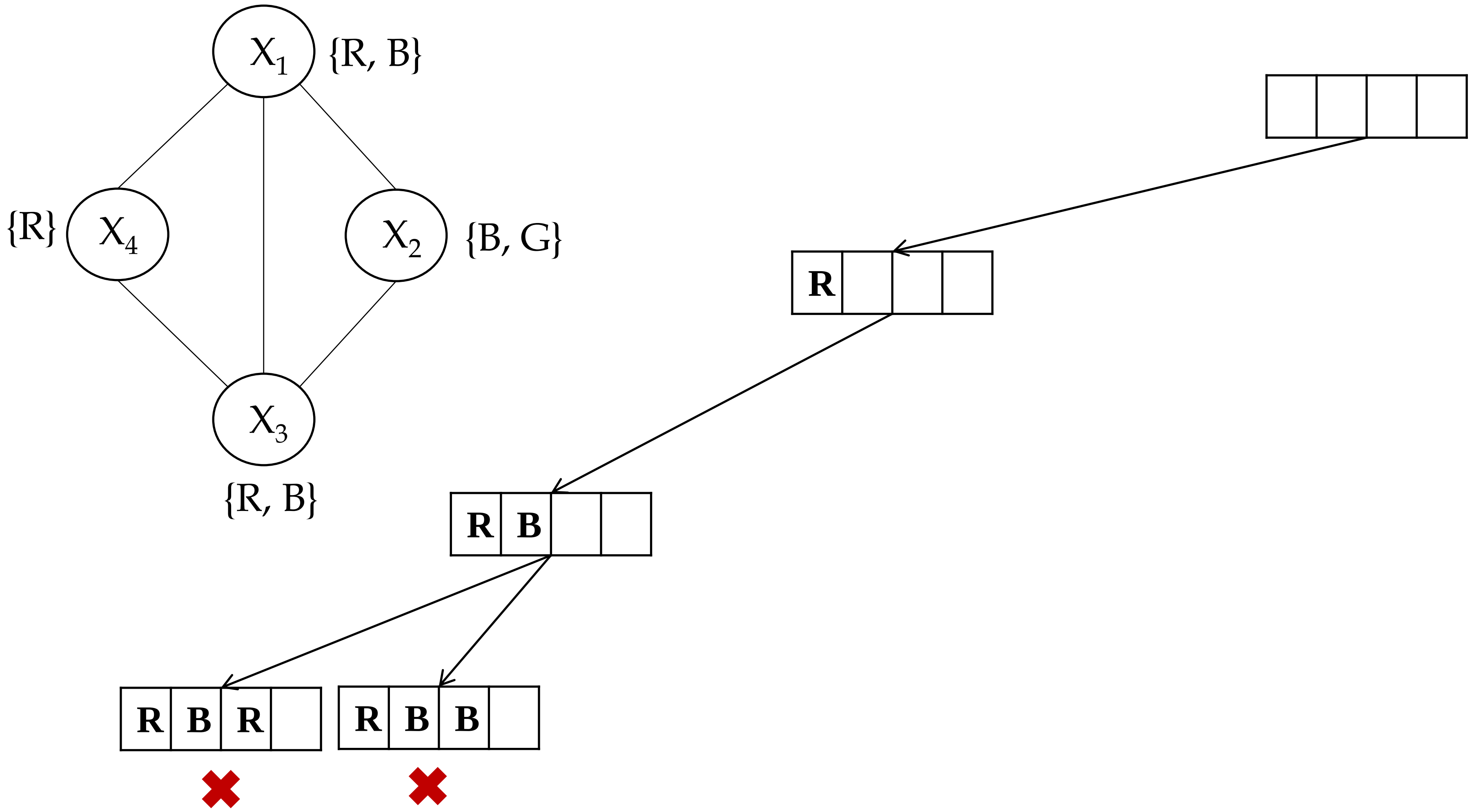


X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}
R
R B

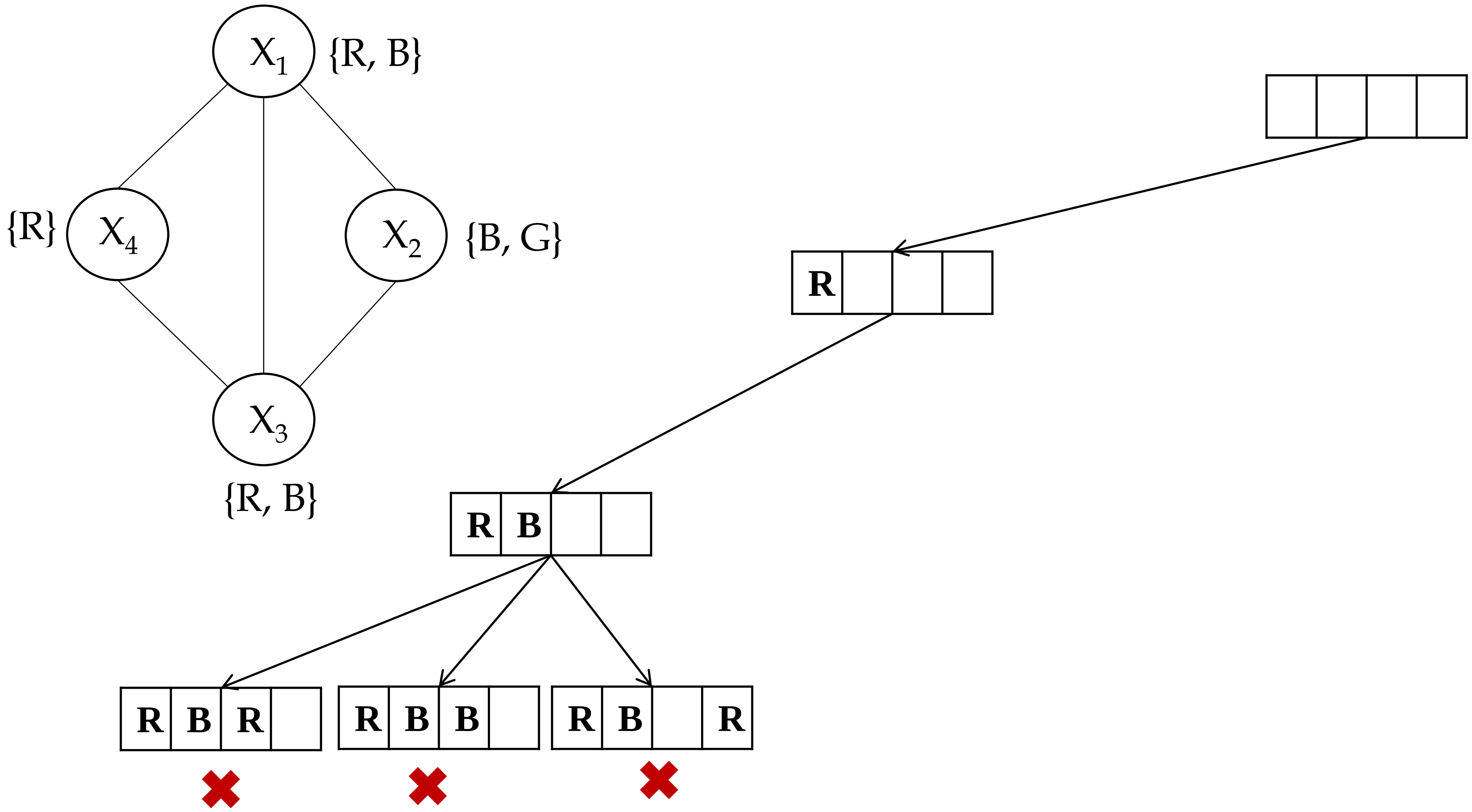


X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}
R
R B
R B R

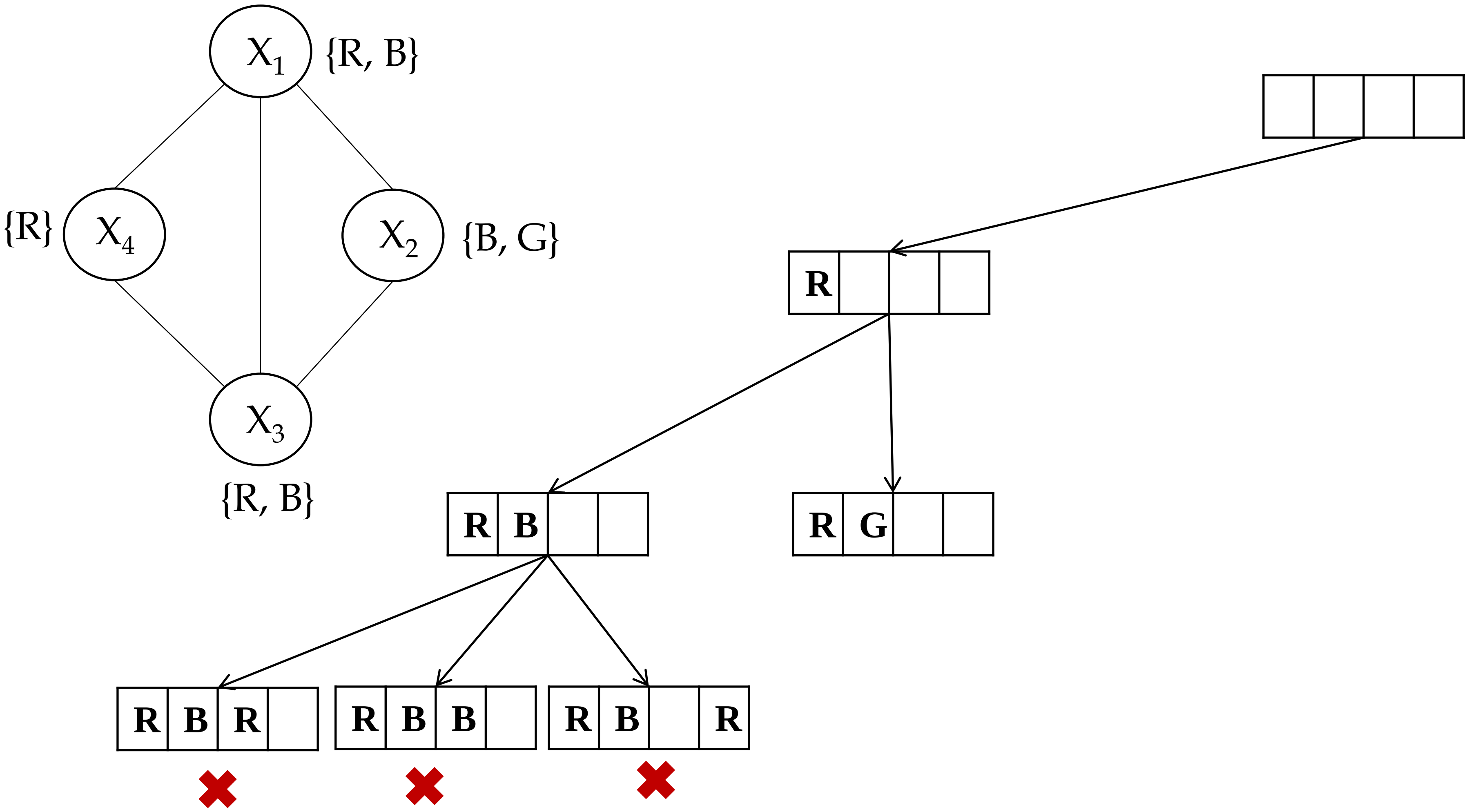

X1
{R, B}
X4
{R}
X2
{B, G}
X3
{R, B}
R
R B
R B R

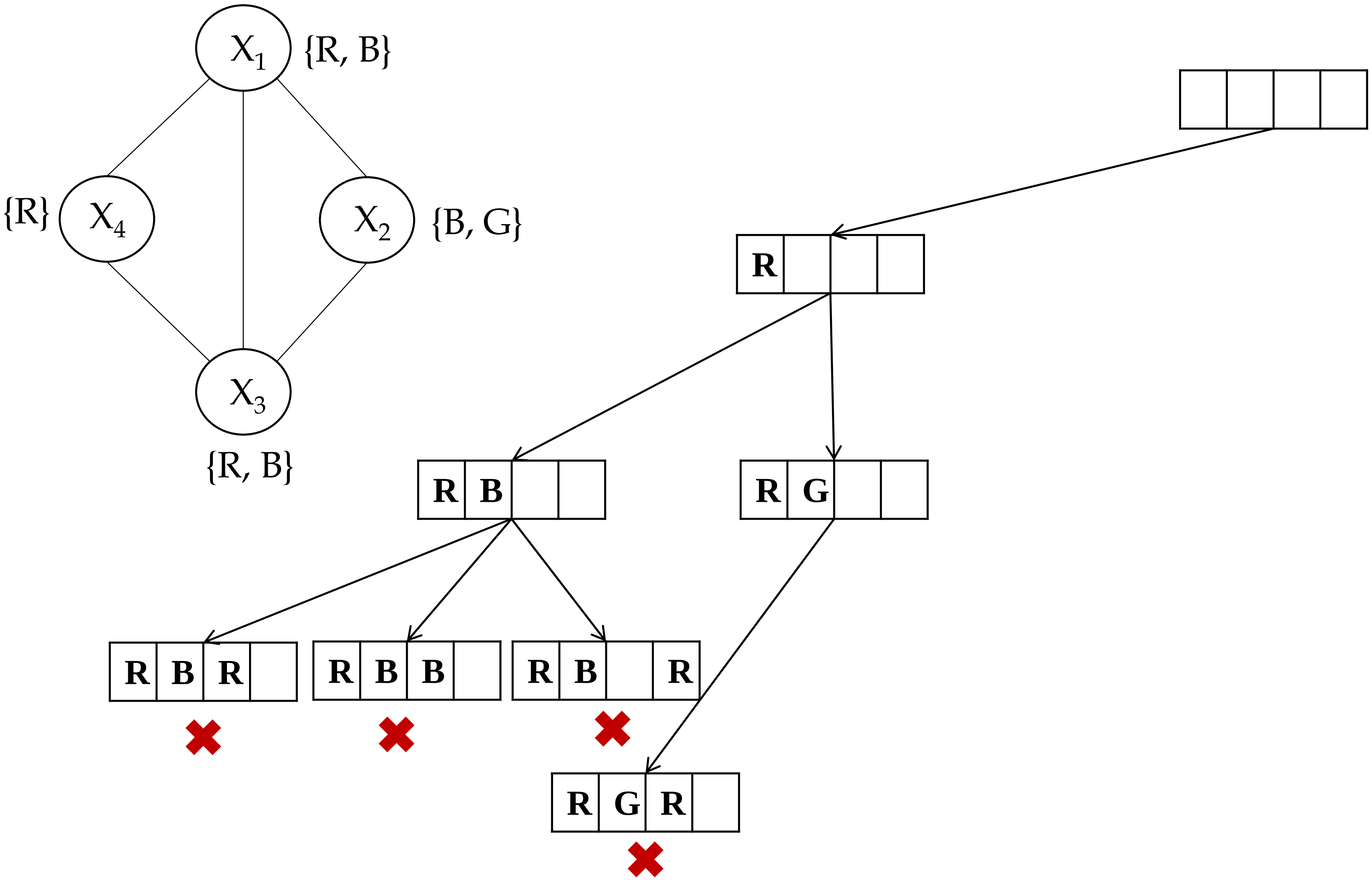


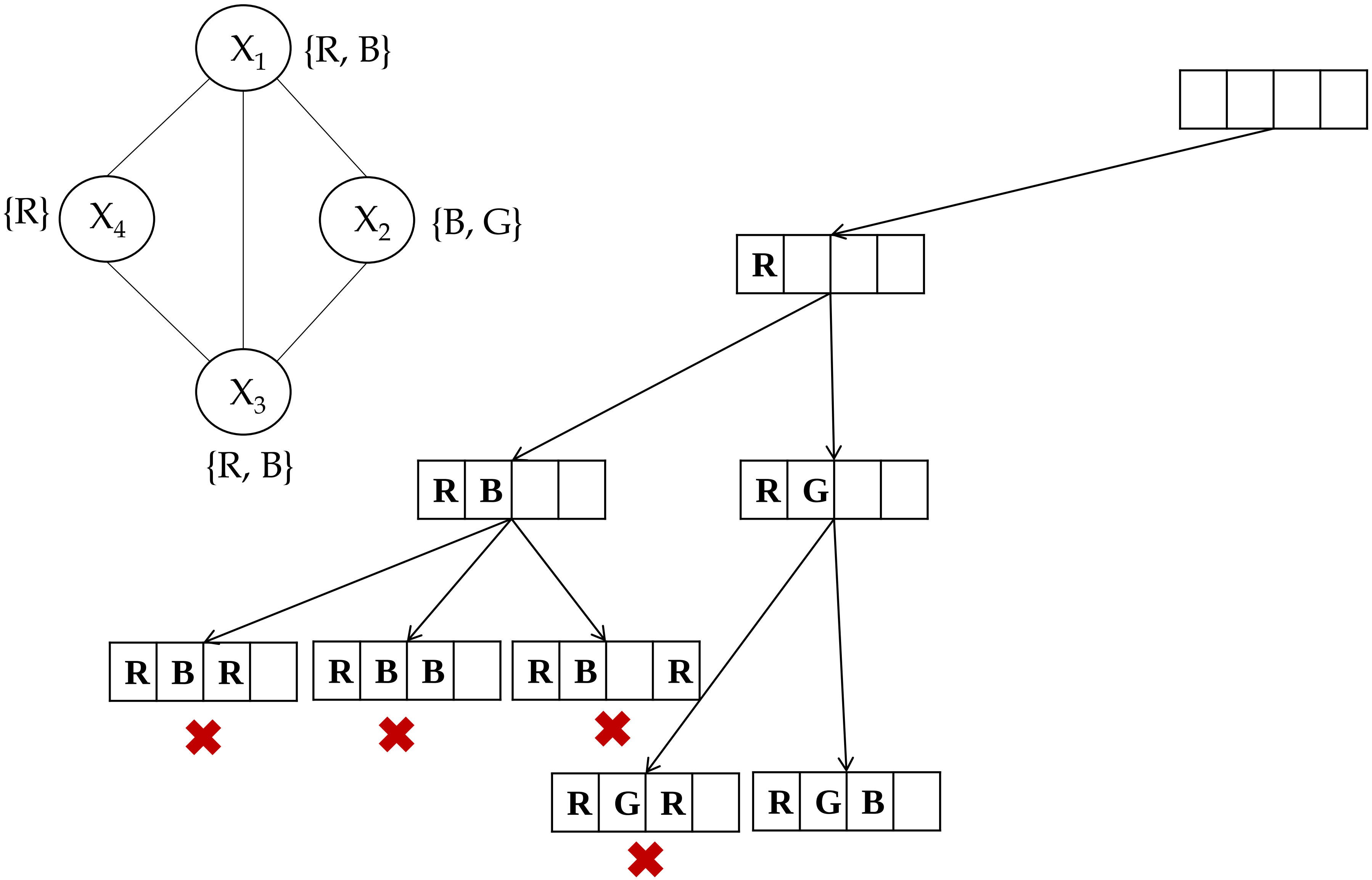
X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}
R
R B
R B R
R B B



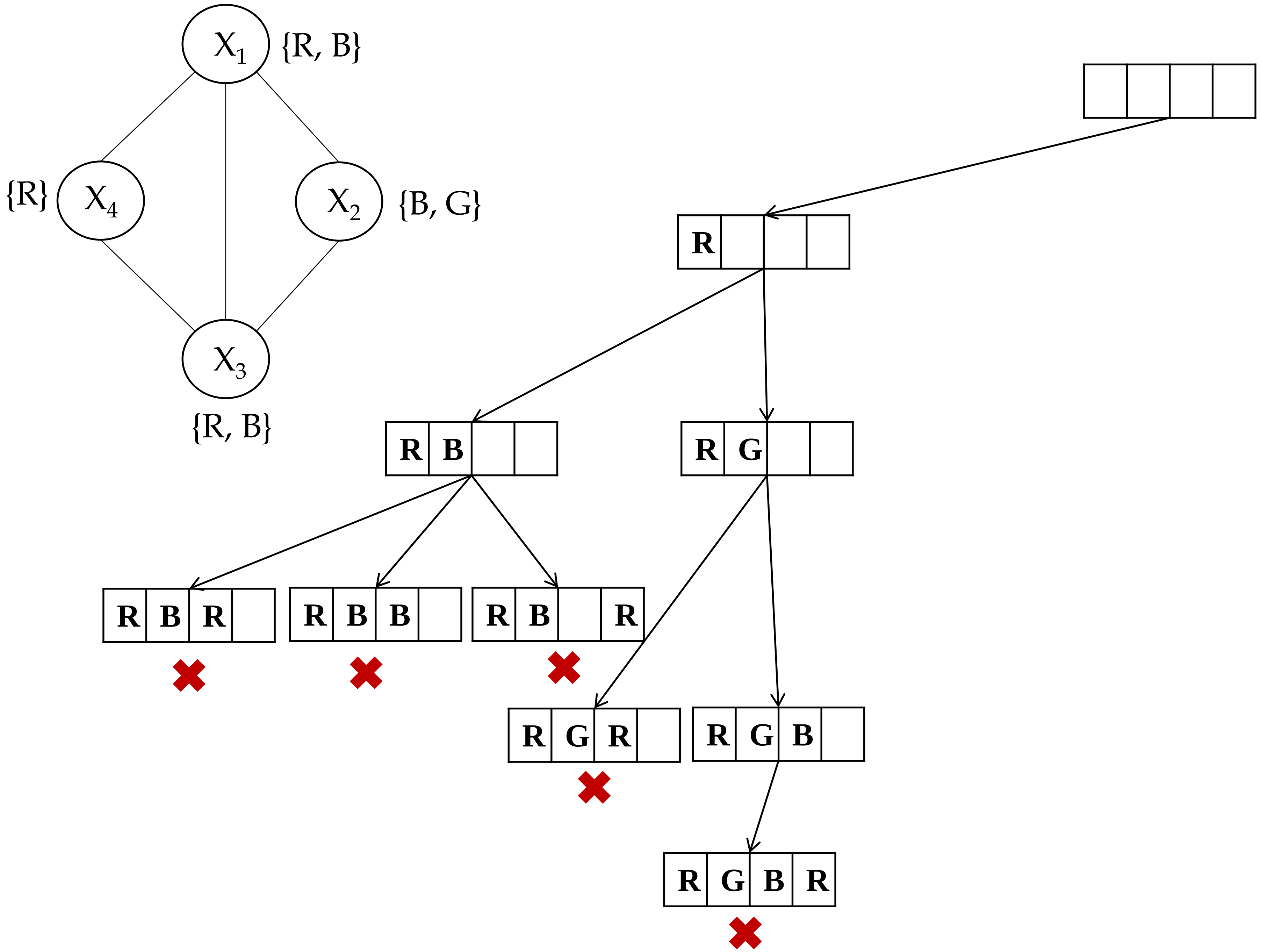
X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}
R
R B
R B R
R B B
R B R


X1
{R, B}
{R}
X4
X2
{B, G}
X3
{R, B}
R
R B
R G
R B R
R B B
R B R

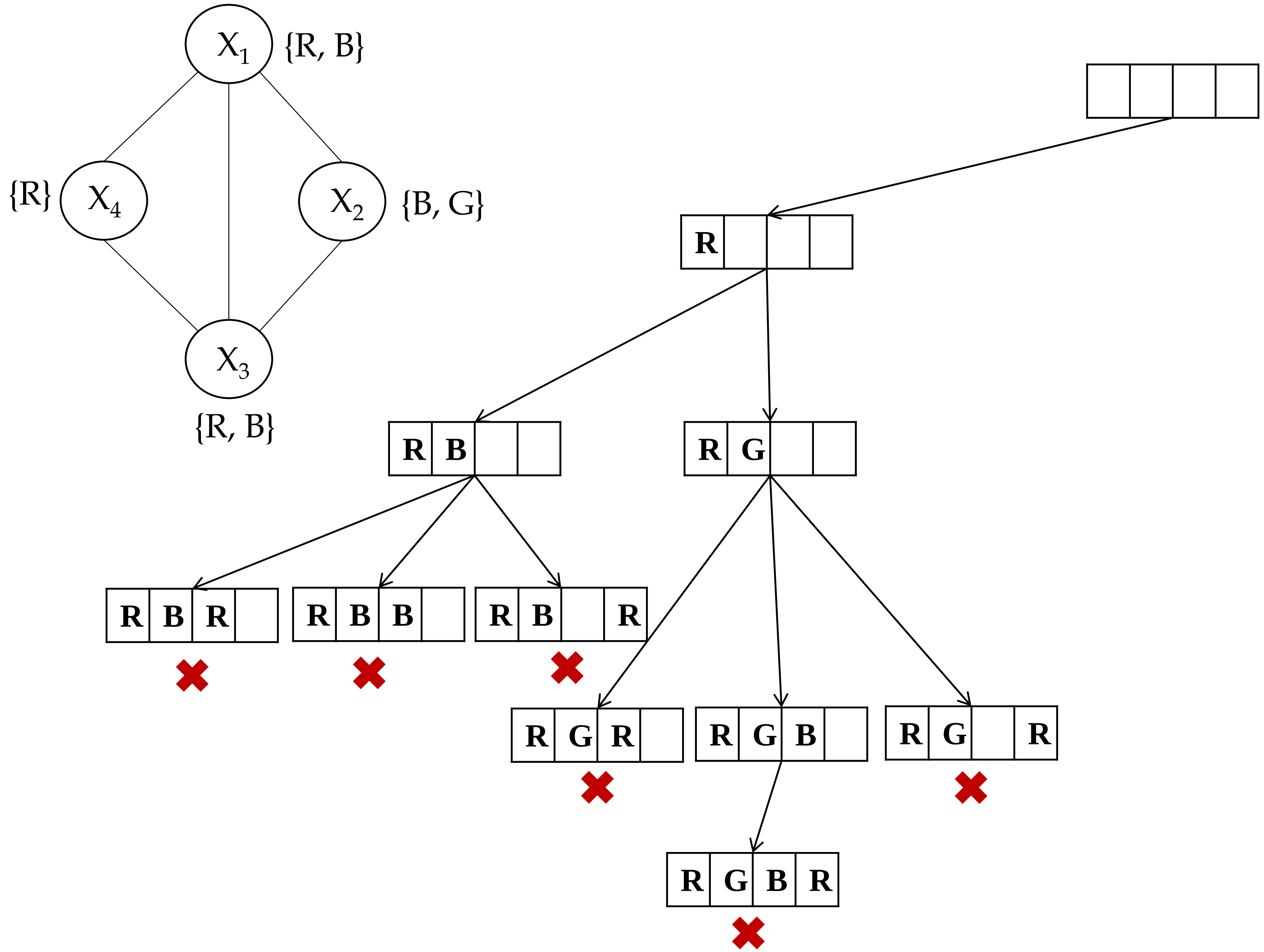

X1
{R, B}
{R}
X4
X2
{B, G}
X3
{R, B}
R
R B
R G
R B R
R B B
R B R
R G R



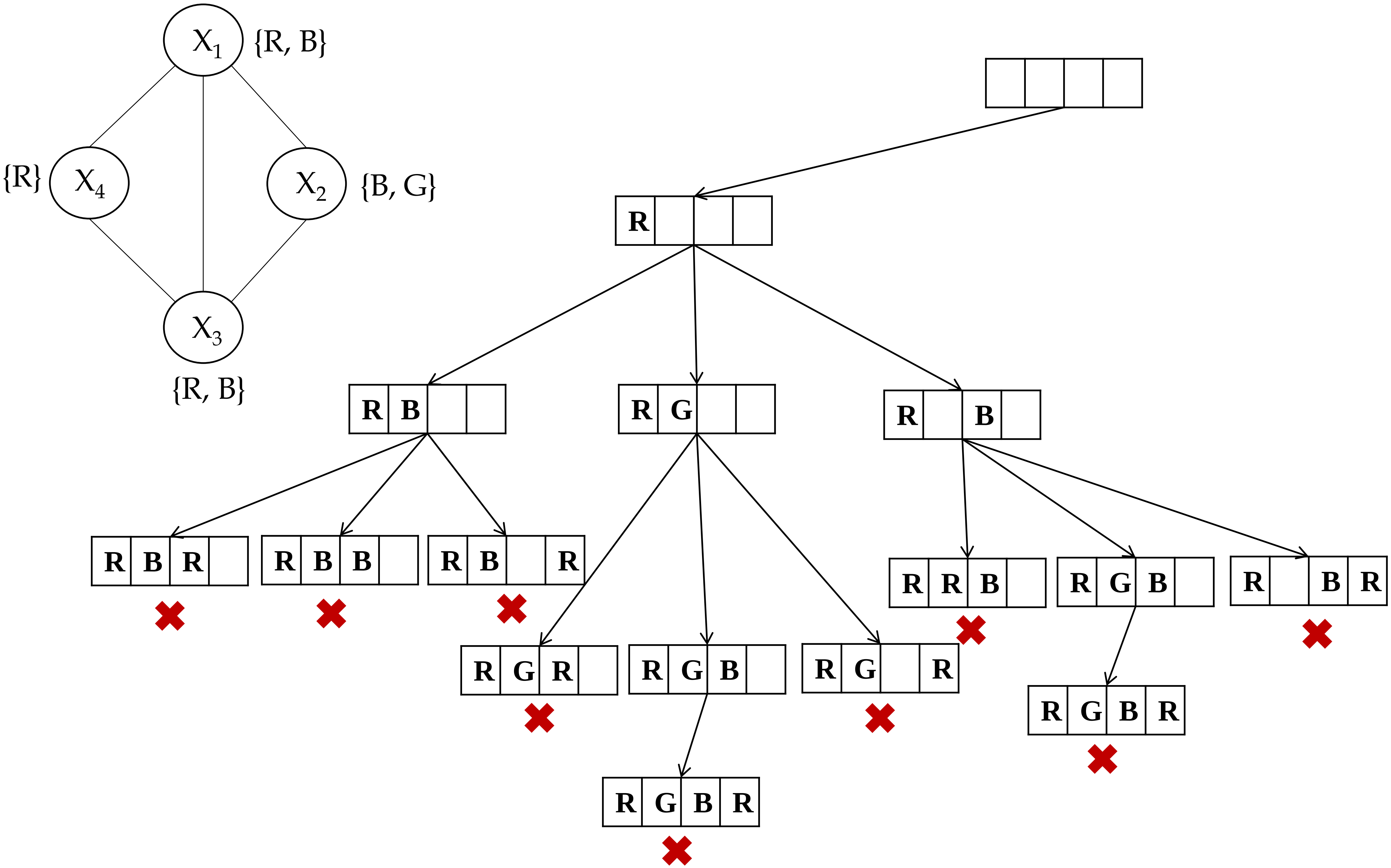
X1 {R, B}
{R} X4
X2 {B, G}
X3
{R, B}
R
R B
R G
R B R
R B B
R B R
R G R
R G B



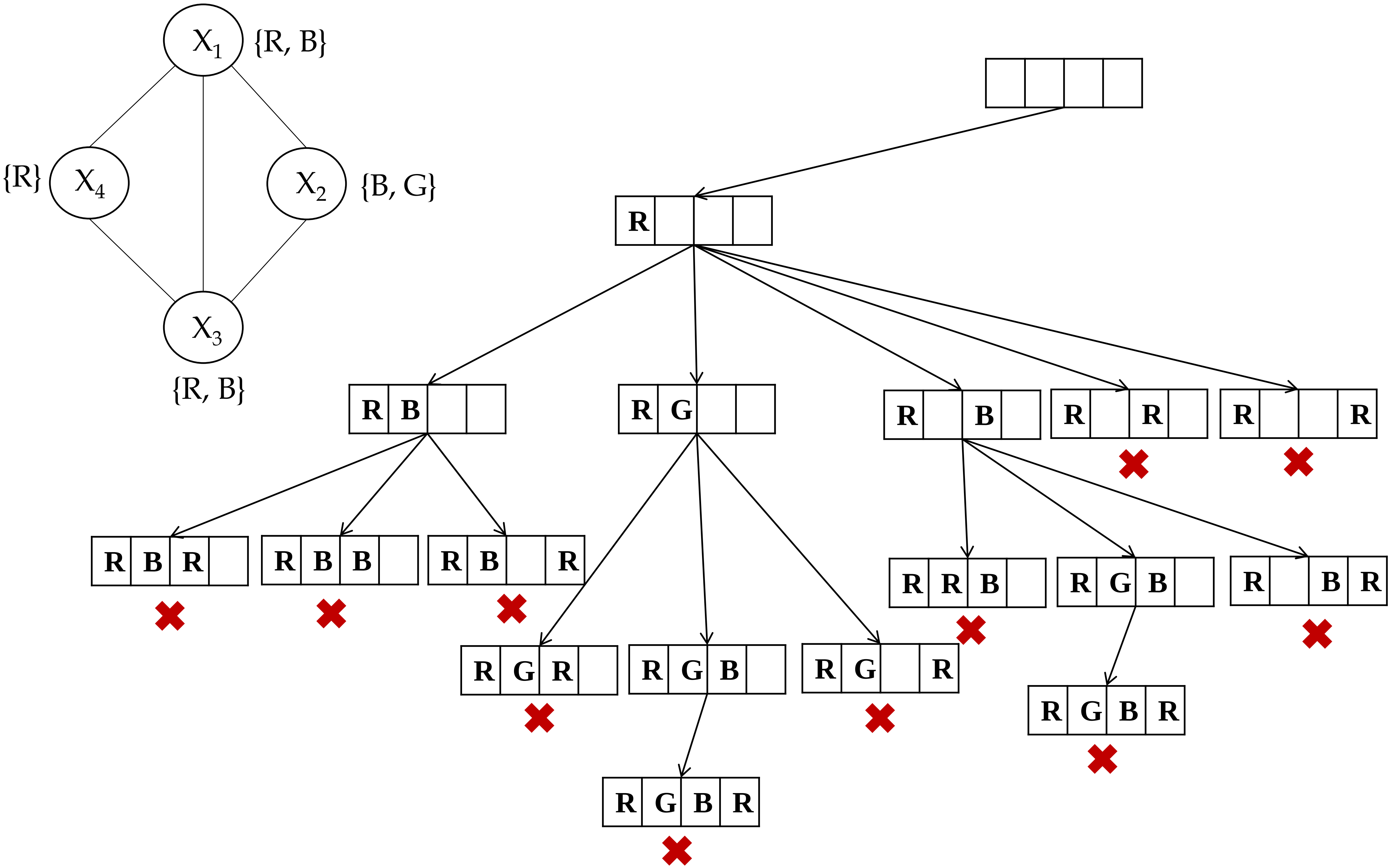
X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}
R
R B
R G
R B R
R B B
R B R
R G R
R G B
R G B R



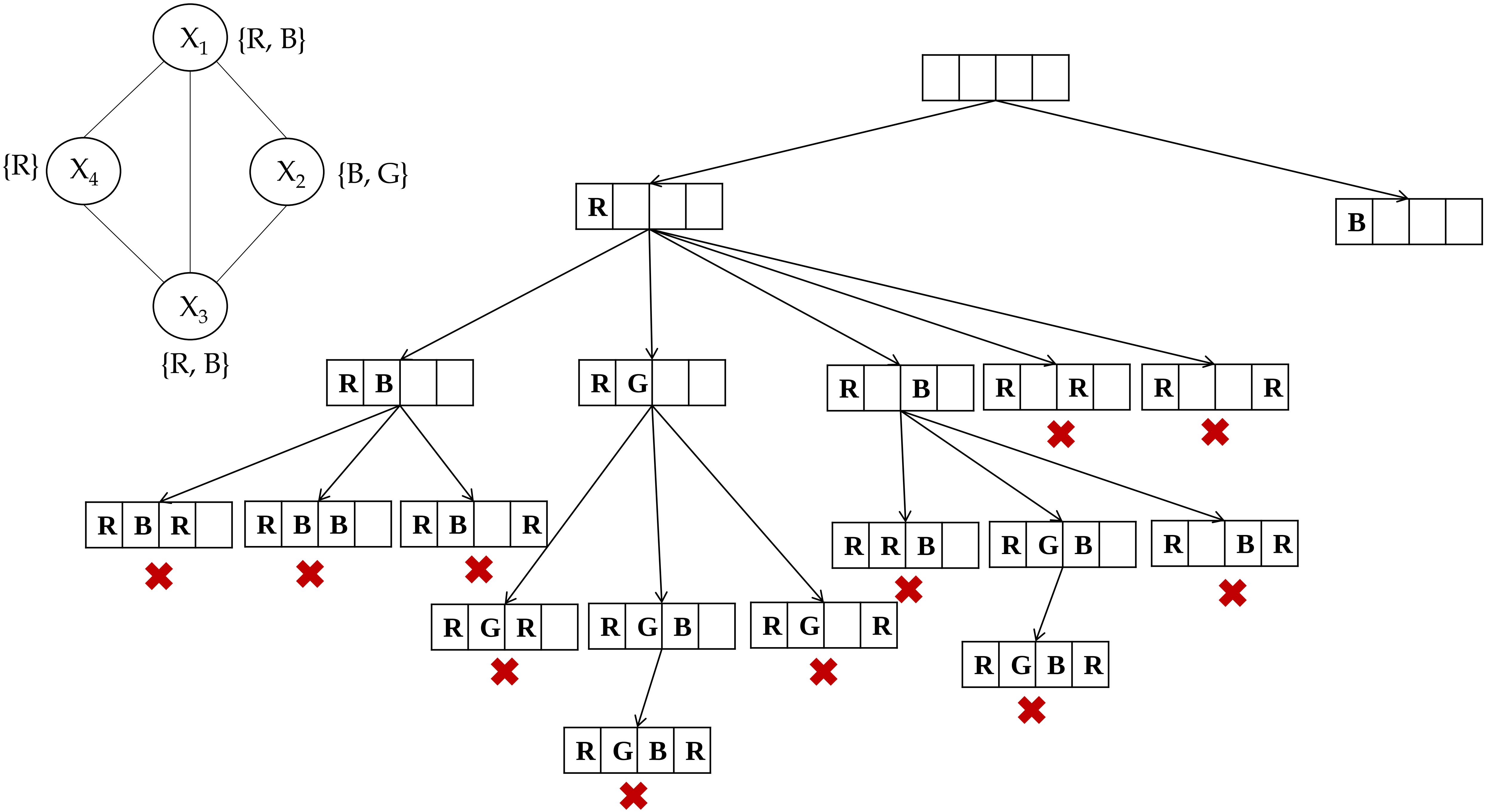
X1 {R, B}
{R} X4
X2 {B, G}
X3
{R, B}
R
R B
R G
R B R
R B B
R B R
R G R
R G B
R G R
R G B R



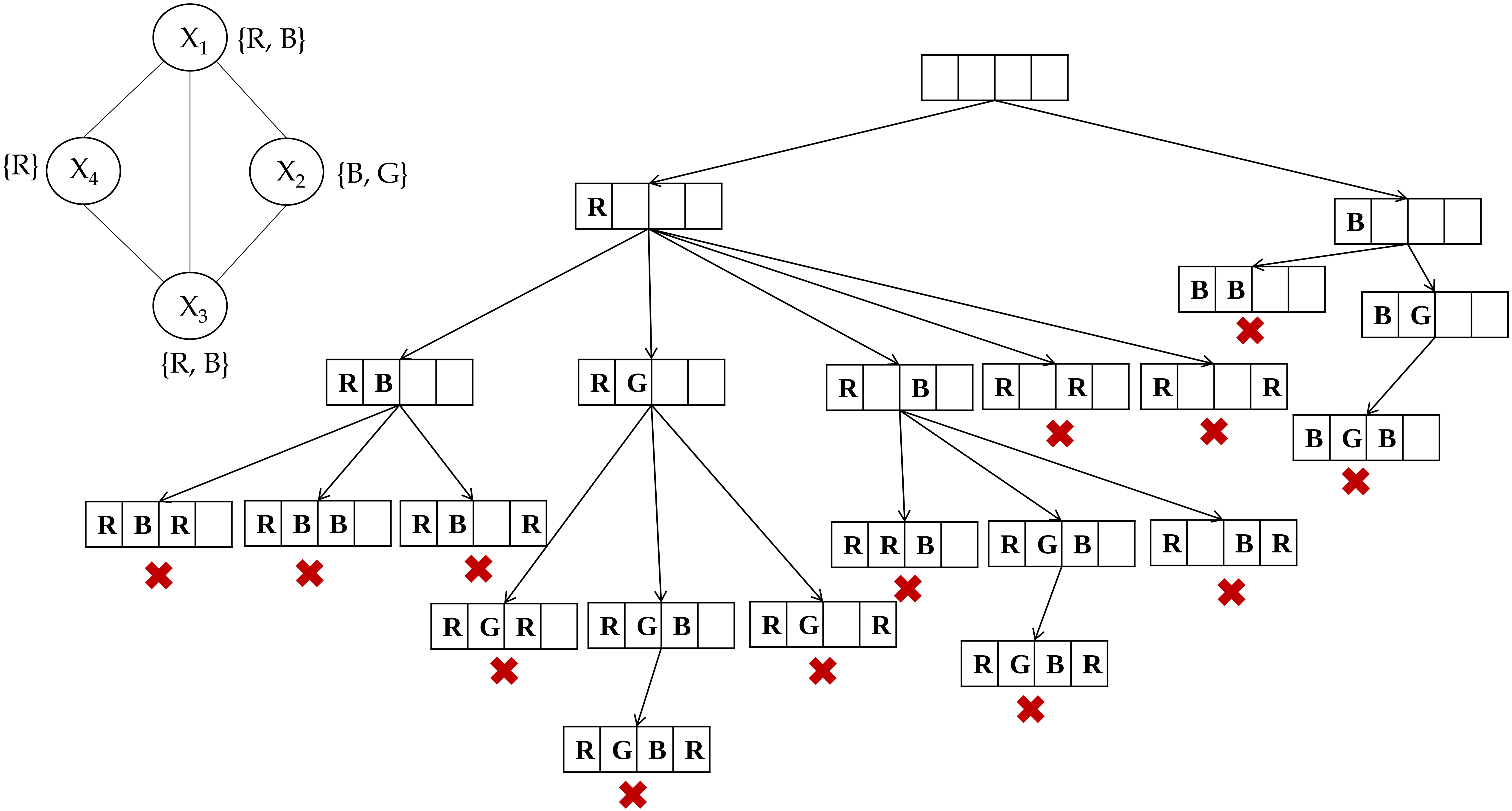
X1 {R, B}
X4 {R}
X2 {B, G}
X3 {R, B}
R
R B
R G
R B
R B R
R B B
R B R
R G R
R G B
R G R
R R B
R G B
R B R
R G B R
R G B R

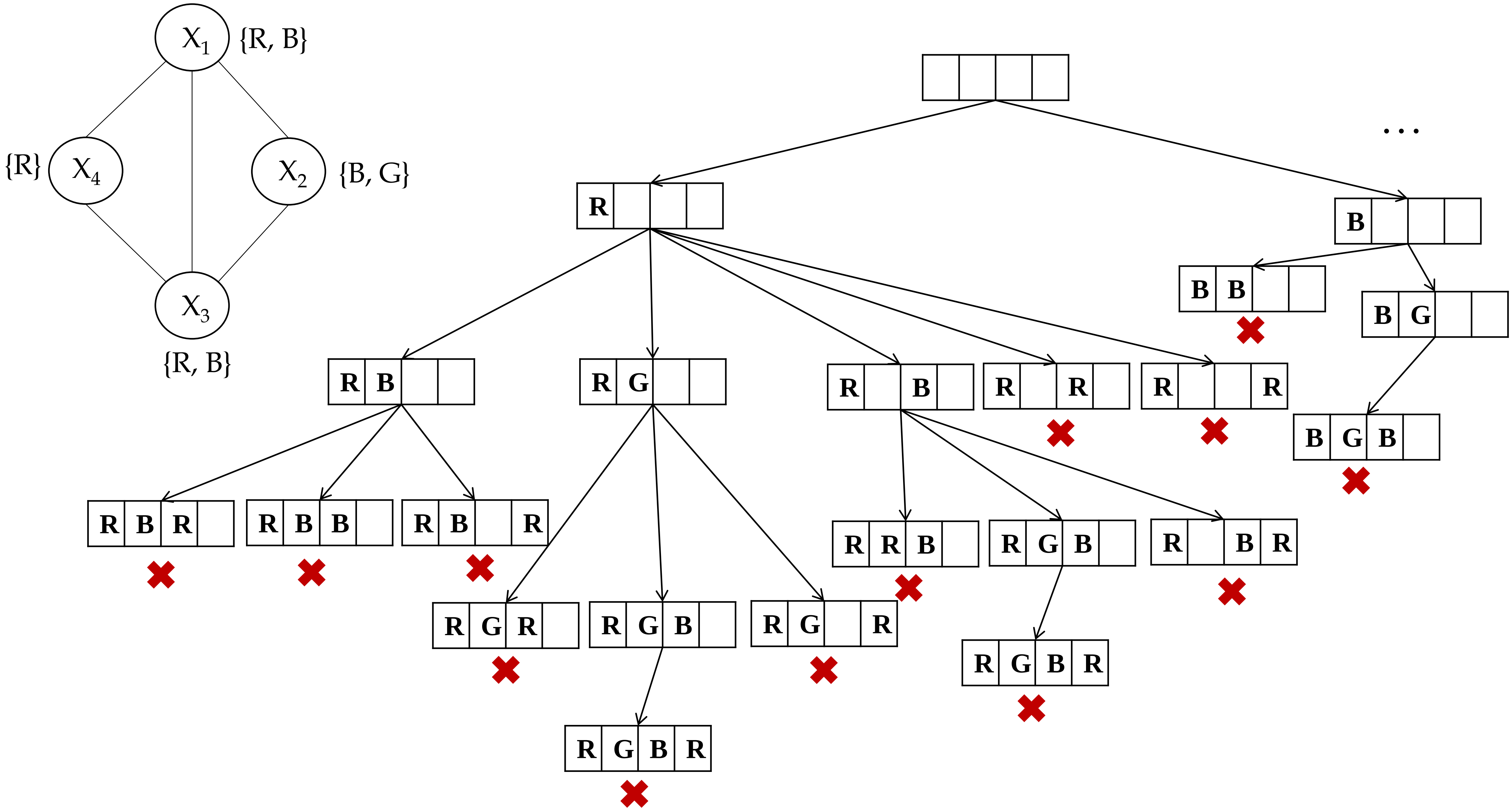


X1 {R, B}
{R} X4
X2 {B, G}
X3
{R, B}
R
R B
R G
R B
R R
R R
R B R
R B B
R B R
R G R
R G B
R G R
R R B
R G B
R B R
R G B R
R G B R

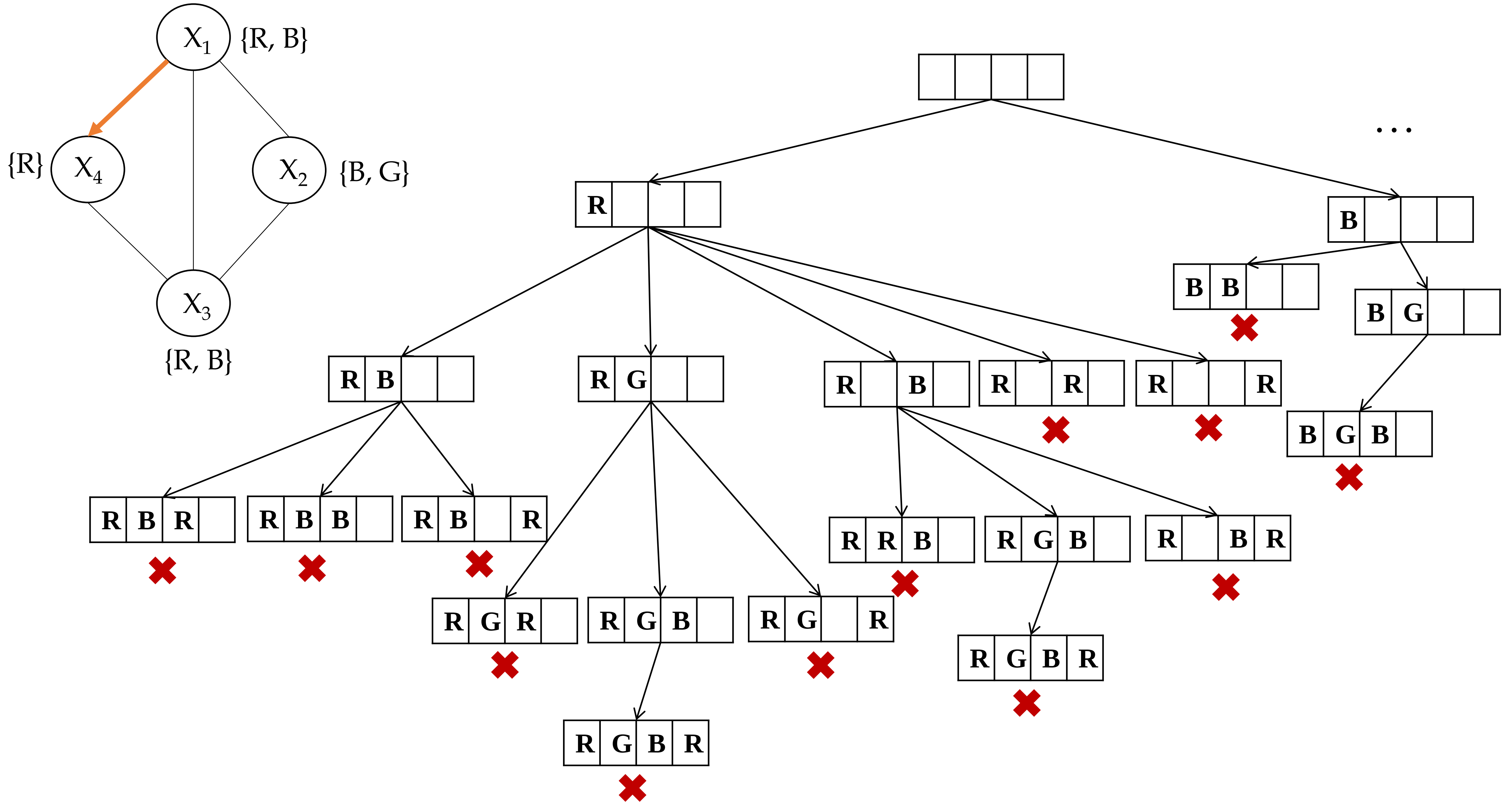


X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}
R
B
R B
R G
R B
R R
R R
R B R
R B B
R B R
R G R
R G B
R G R
R R B
R G B
R B R
R G B R
R G B R

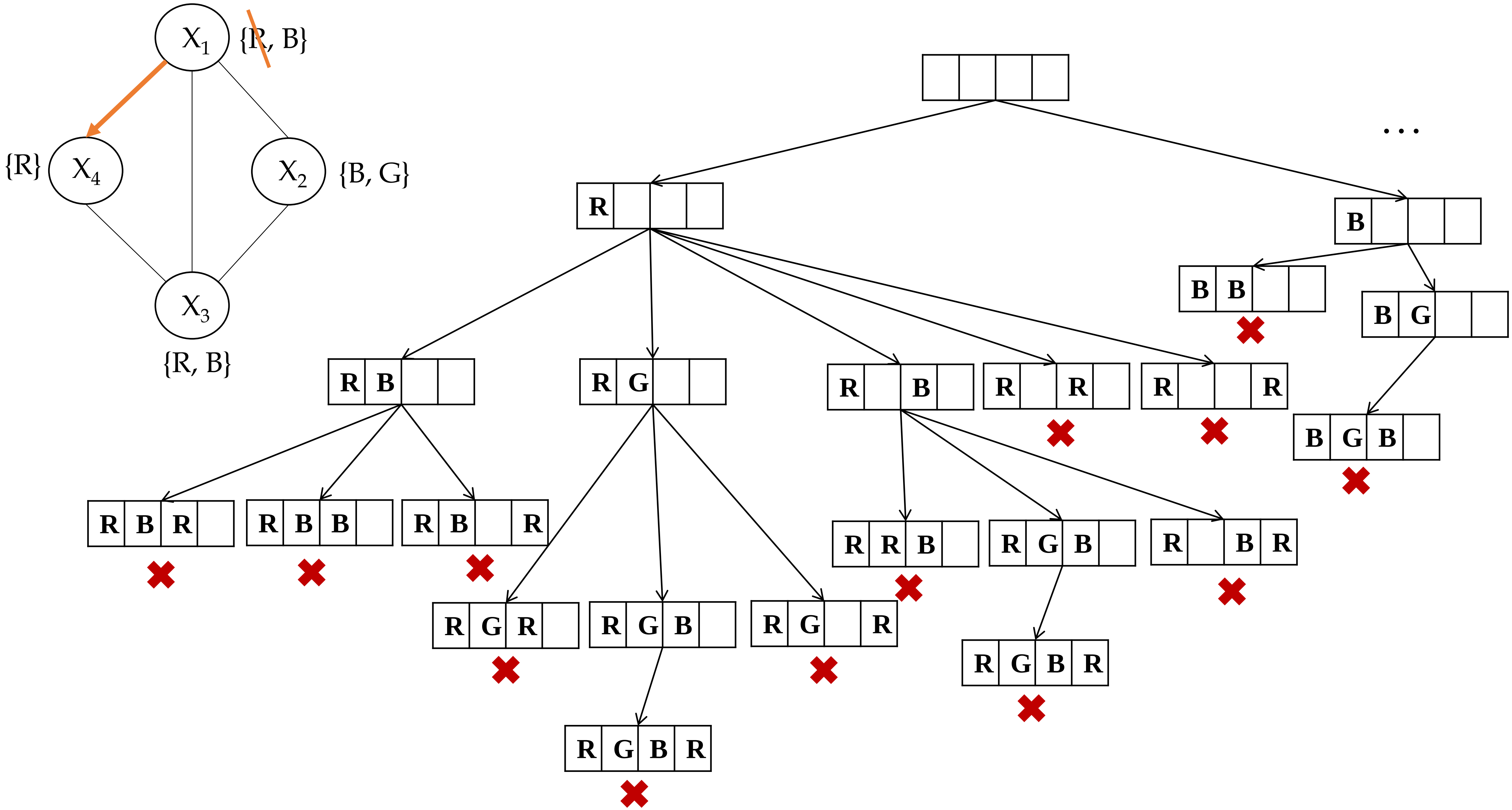

X_1 {R, B}
X_2 {B, G}
X_3 {R, B}
X_4 {R}



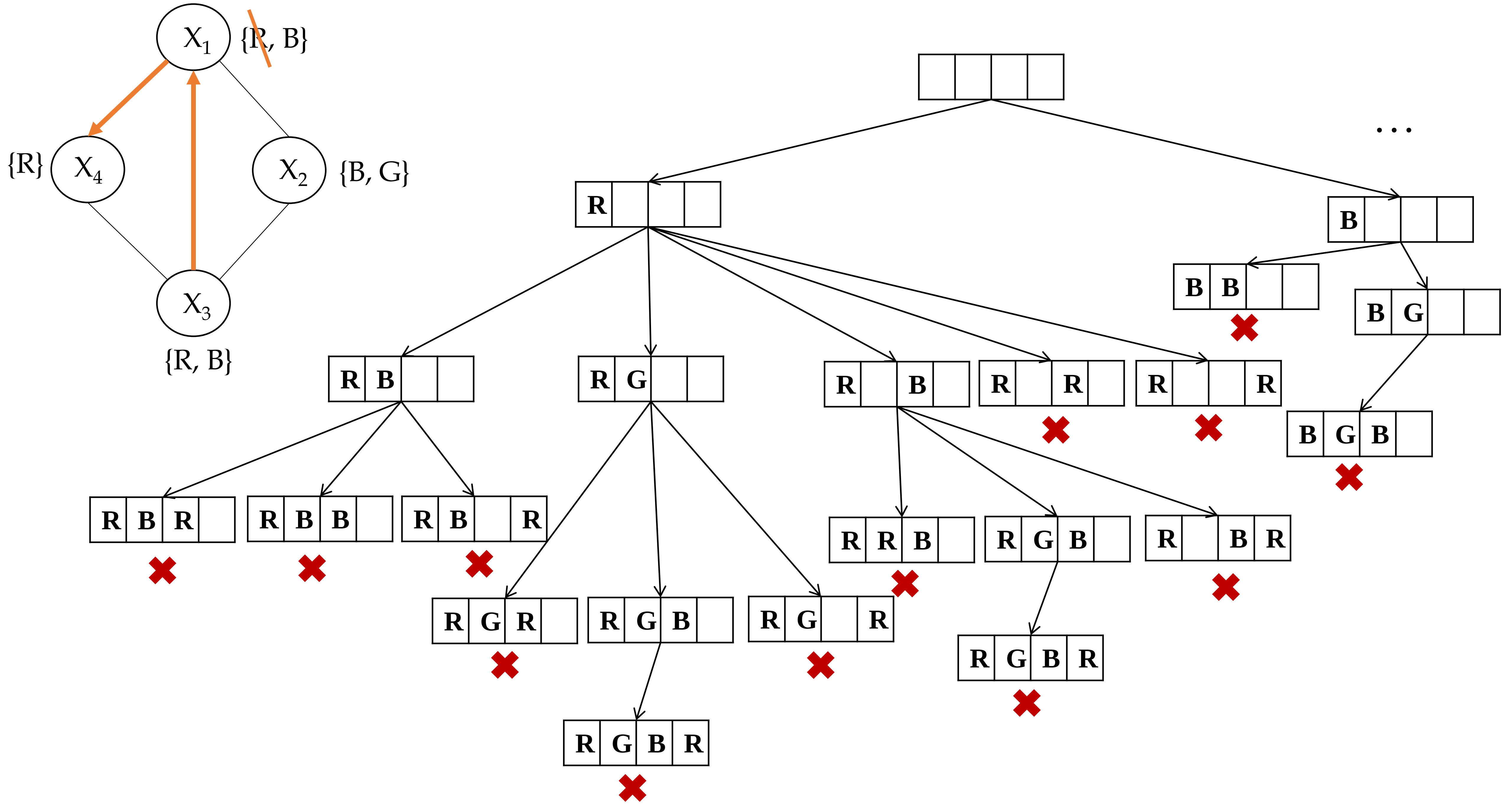
X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}

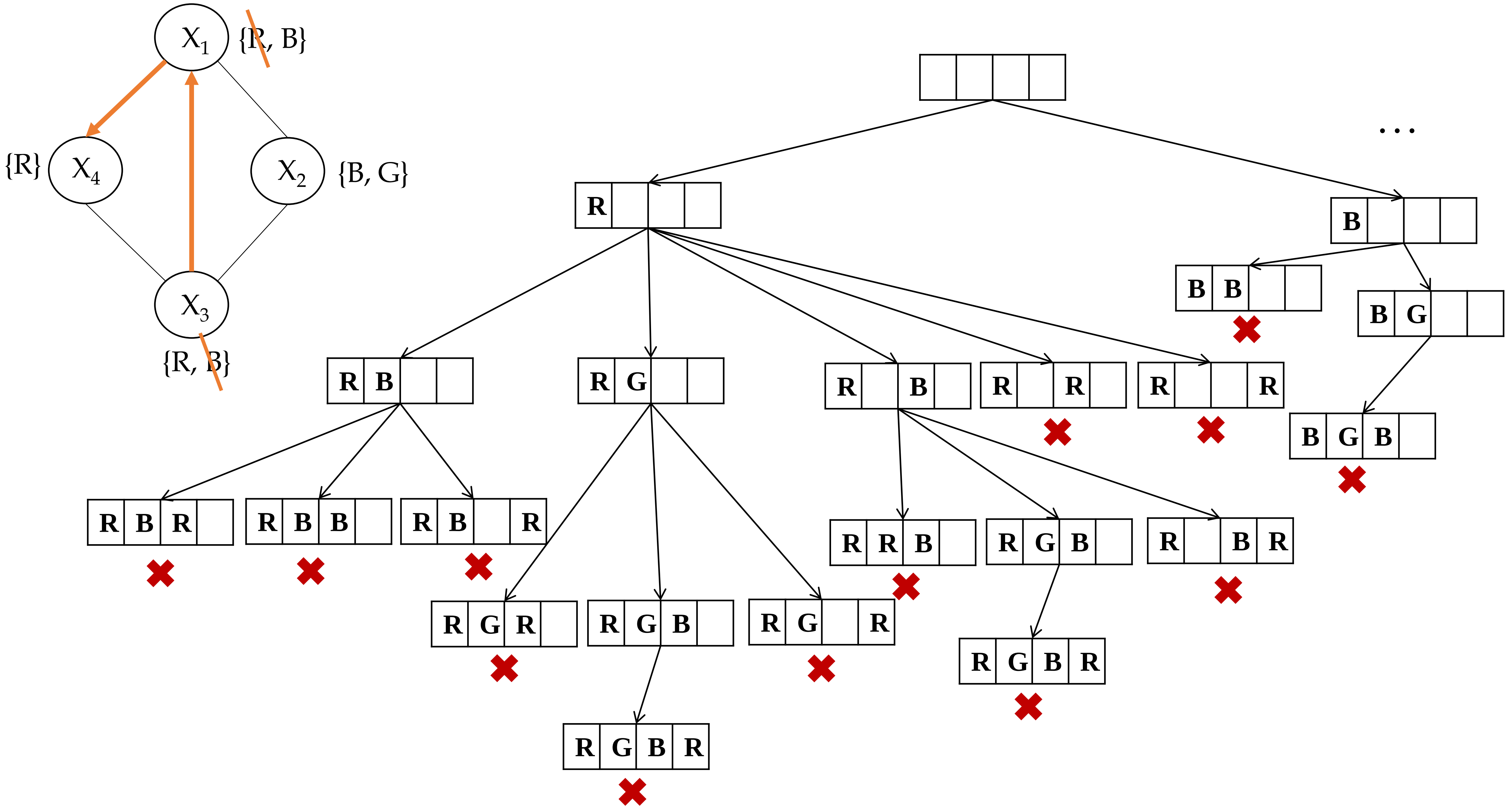


X1 {R, B}
{R} X4
X2 {B, G}
X3
{R, B}
R
B
R B
R G
R B
R R
R R
B B
B G
B G B
R B R
R B B
R B R
R G R
R G B
R G R
R G B R
R R B
R G B
R B R
R G B R

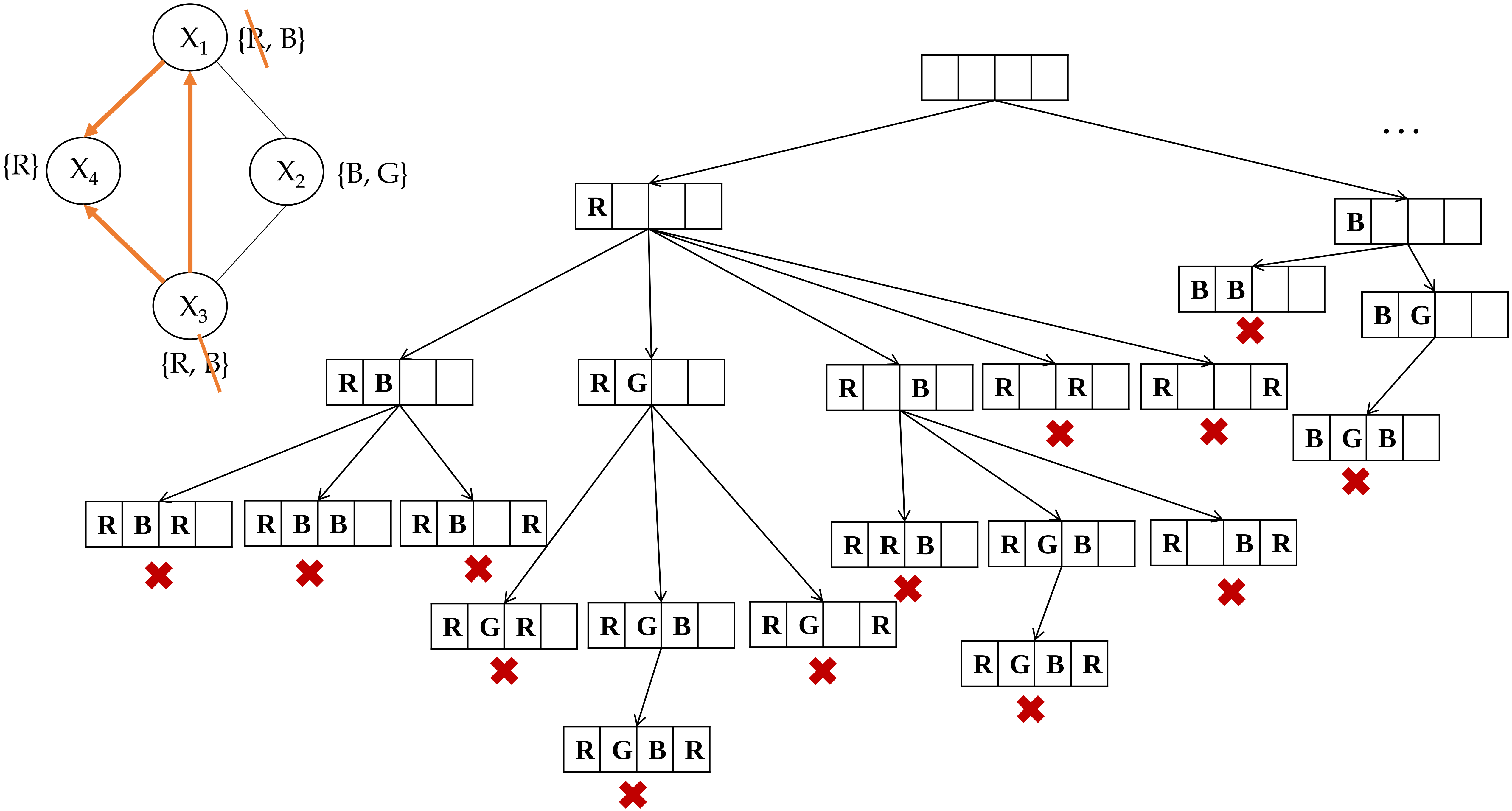


X1
{R, B}
X4
{R}
X2
{B, G}
X3
{R, B}
R
B
R B
R G
R B
R R
R R
B B
B G
B G B
R B R
R B B
R B R
R G R
R G B
R G R
R R B
R G B
R B R
R G B R
R G B R
...

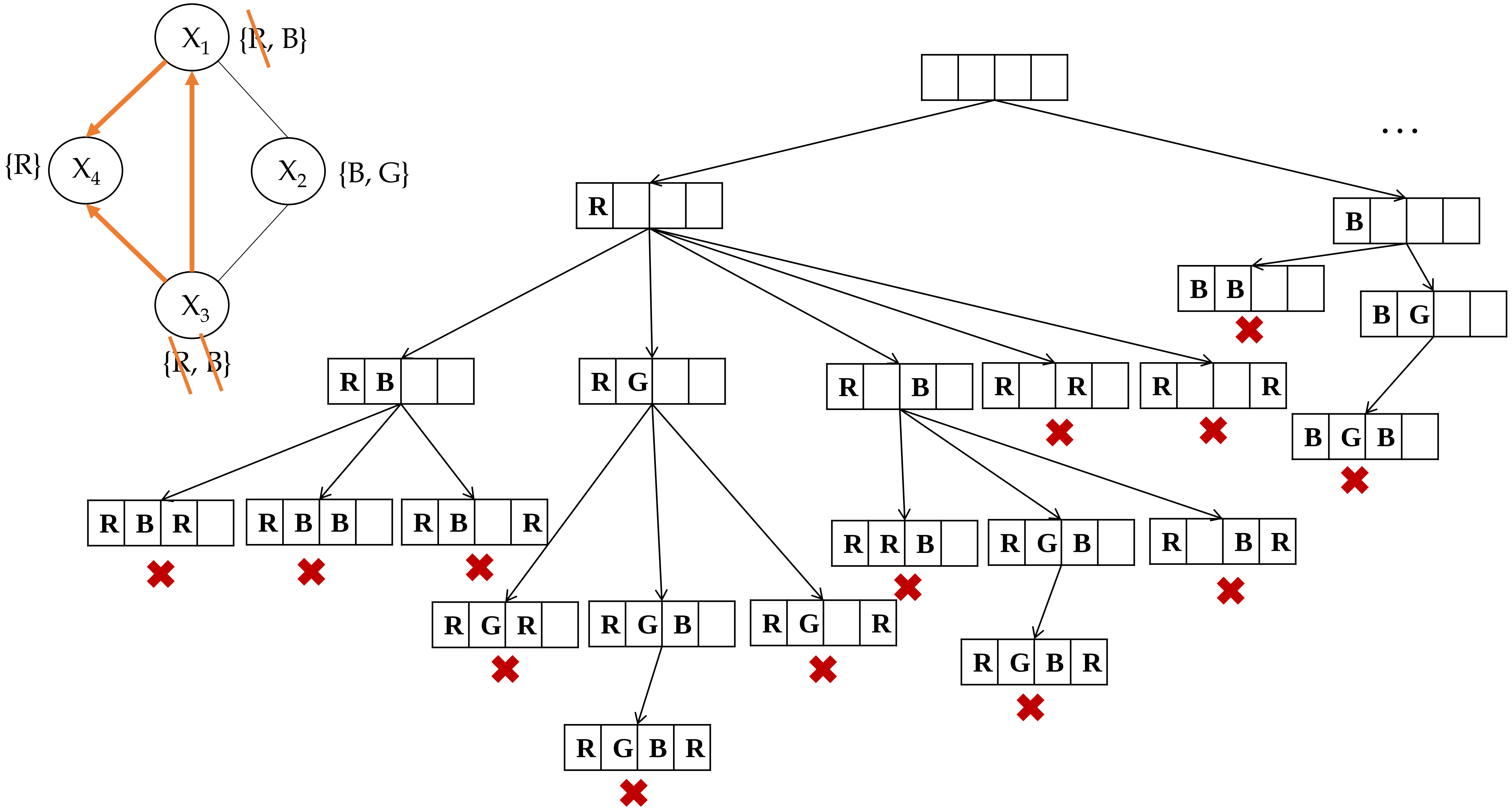

X_1
{R, B}
X_4
{R}
X_2
{B, G}
X_3
{R, B}
...



X_1
{R, B}
{R}
X_4
X_2
{B, G}
X_3
{R, B}
…



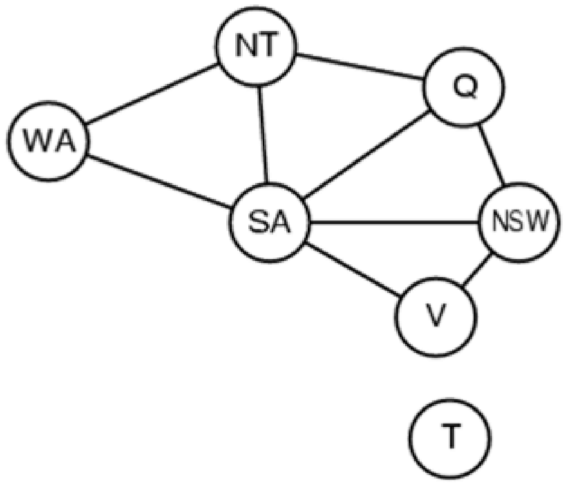
X_1 {R, B}
{R} X_4
X_2 {B, G}
X_3
{R, B}
R
B
R B
R G
R B
R R
R R
B B
B G
B G B
R B R
R B B
R B R
R G R
R G B
R G R
R R B
R G B
R B R
R G B R
R G B R
…



X1 {R, B}
{R} X4
X2 {B, G}
X3 {R, B}
R
B
R B
R G
R B
R R
R R
B B
B G
B G B
R B R
R B B
R B R
R G R
R G B
R G R
R R B
R G B
R B R
R G B R
R G B R
...

پایه‌ی استنتاج برای CSPها: سازگاری محلی

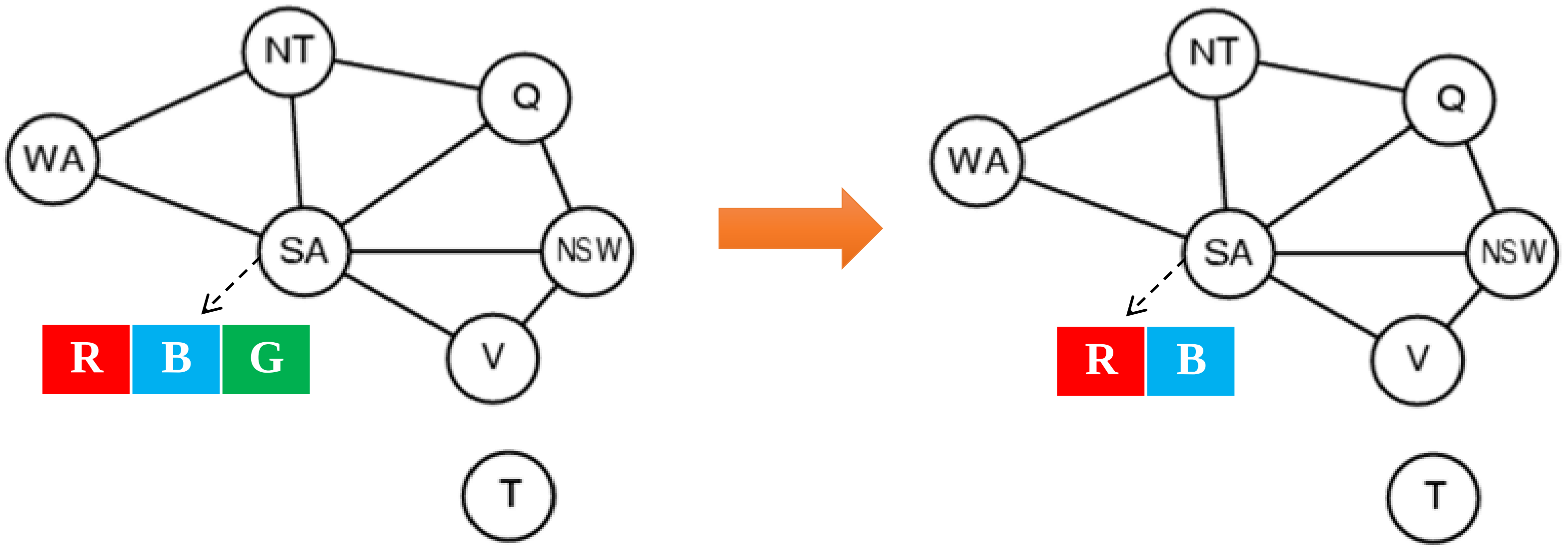
- در جستجوی فضای حالت منظم، یک الگوریتم تنها یک کار می‌تواند انجام دهد: جستجو
- اما در CSPها یک انتخاب برای انجام وجود دارد: جستجو یا انتشار محدودیت
  - جستجو: از امکانات مختلف انتساب متغیر یکی را انتخاب کند.
  - انتشار محدودیت: با استفاده از محدودیت‌ها تعداد مقادیر قانونی برای یک متغیر کم می‌شود که به نوبه‌ی خود می‌تواند مقادیر قانونی برای متغیرهای دیگر را نیز کاهش دهد و ...
- ایده‌ی کلیدی سازگاری محلی (local consistency) است.
  - فرایند اجرای سازگاری محلی در هر بخش از گراف محدودیت باینری، باعث می‌شود مقادیر ناسازگار از گراف حذف شوند.
  - انواع سازگاری محلی: سازگاری نود، سازگاری کمان، ...



سازگاری نود (Node-consistency)

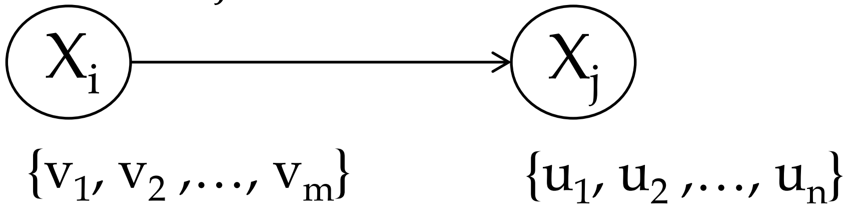
- یک متغیر سازگار نود است اگر تمام مقادیر در دامنه‌ی متغیر با محدودیت‌های یگانی متغیر سازگار باشد.
  - برای مثال اگر در نوعی مسئله رنگ‌آمیزی نقشه، SA رنگ سبز را قبول نکند، می‌توان سازگاری نود را با حذف رنگ سبز از دامنه آن برقرار ساخت.

$$D_{SA}=\{red, blue, green\} \text{ , } SA \neq green \rightarrow D_{SA}=\{red, blue\}$$



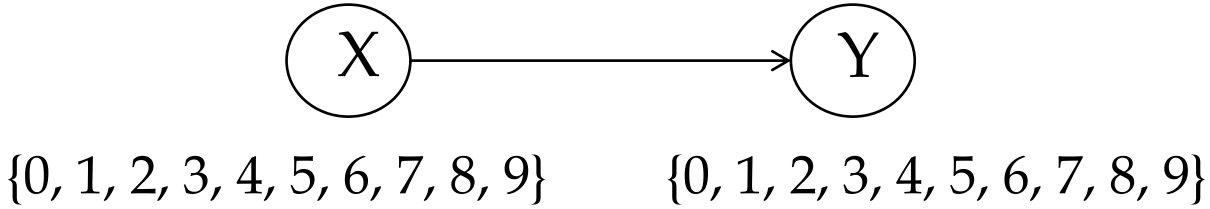
سازگاری کمان (Arc-consistency)

- به‌طور رسمی: X_i نسبت به متغیر دیگر X_j سازگار کمان است اگر برای هر مقدار در دامنه‌ی فعلی D_i مقداری در دامنه‌ی D_j وجود داشته باشد که محدودیت باینری روی کمان (X_i,X_j) را ارضا کند.



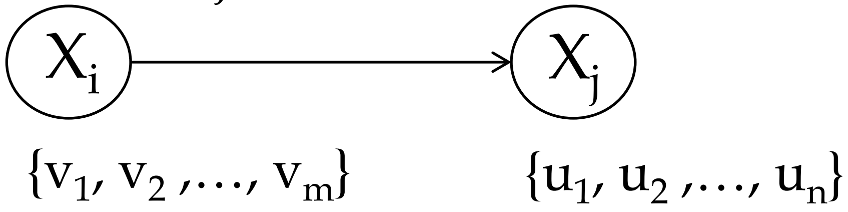
- یک متغیر سازگار کمان است اگر هر مقدار در دامنه‌ی آن محدودیت‌های دوتایی متغیر را مرتفع کند.

- مثال
  - متغیرها: $\{X, Y\}$
  - دامنه: $\{0,1,2,\ldots,9\}$
  - محدودیت: $Y=X^2$



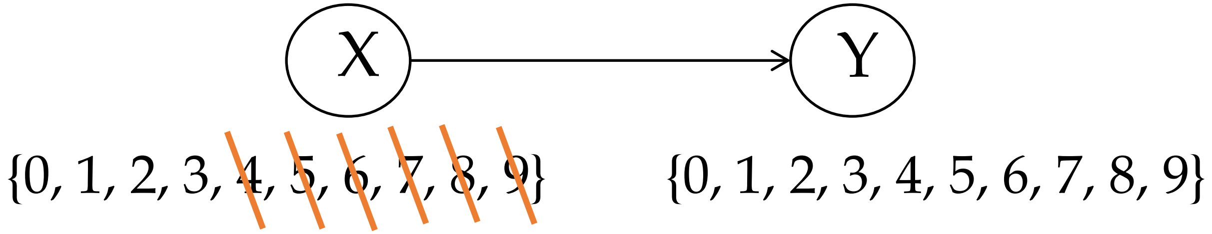
سازگاری کمان (Arc-consistency)

- به‌طور رسمی: X_i نسبت به متغیر دیگر X_j سازگار کمان است اگر برای هر مقدار در دامنه‌ی فعلی D_i مقداری در دامنه‌ی D_j وجود داشته باشد که محدودیت باینری روی کمان (X_i,X_j) را ارضا کند.



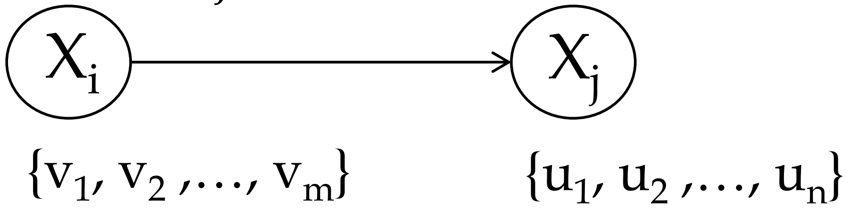
- یک متغیر سازگار کمان است اگر هر مقدار در دامنه‌ی آن محدودیت‌های دوتایی متغیر را مرتفع کند.

- مثال
  - متغیرها: $\{X, Y\}$
  - دامنه: $\{0,1,2,\ldots,9\}$
  - محدودیت: $Y=X^2$



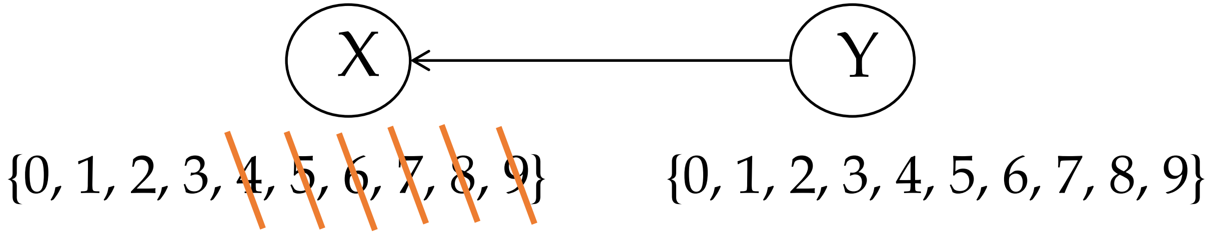
سازگاری کمان (Arc-consistency)

- به‌طور رسمی: X_i نسبت به متغیر دیگر X_j سازگار کمان است اگر برای هر مقدار در دامنه‌ی فعلی D_i مقداری در دامنه‌ی D_j وجود داشته باشد که محدودیت باینری روی کمان (X_i,X_j) را ارضا کند.



- یک متغیر سازگار کمان است اگر هر مقدار در دامنه‌ی آن محدودیت‌های دوتایی متغیر را مرتفع کند.

- مثال
  - متغیرها: $\{X, Y\}$
  - دامنه: $\{0,1,2,\ldots,9\}$
  - محدودیت: $Y=X^2$



سازگاری کمان (Arc-consistency)

- به‌طور رسمی: X_i نسبت به متغیر دیگر X_j سازگار کمان است اگر برای هر مقدار در دامنه‌ی فعلی D_i مقداری در دامنه‌ی D_j وجود داشته باشد که محدودیت باینری روی کمان (X_i,X_j) را ارضا کند.

$$X_i \longrightarrow X_j$$

$\{v_1, v_2 ,\ldots, v_m\}$ $\{u_1, u_2 ,\ldots, u_n\}$

- یک متغیر سازگار کمان است اگر هر مقدار در دامنه‌ی آن محدودیت‌های دوتایی متغیر را مرتفع کند.

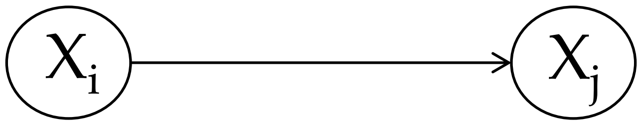
- مثال
  - متغیرها: $\{X, Y\}$
  - دامنه: $\{0,1,2,\ldots,9\}$
  - محدودیت: $Y=X^2$

$$X \longleftarrow Y$$

$\{0, 1, 2, 3, \cancel{4}, \cancel{5}, \cancel{6}, \cancel{7}, \cancel{8}, \cancel{9}\}$ $\{0, 1, \cancel{2}, \cancel{3}, 4, \cancel{5}, \cancel{6}, \cancel{7}, \cancel{8}, 9\}$

الگوریتم سازگاری کمان

برای هر کمان (X_i, X_j) در صف آن را از صف خارج کن



X_i را نسبت به X_j سازگار کن

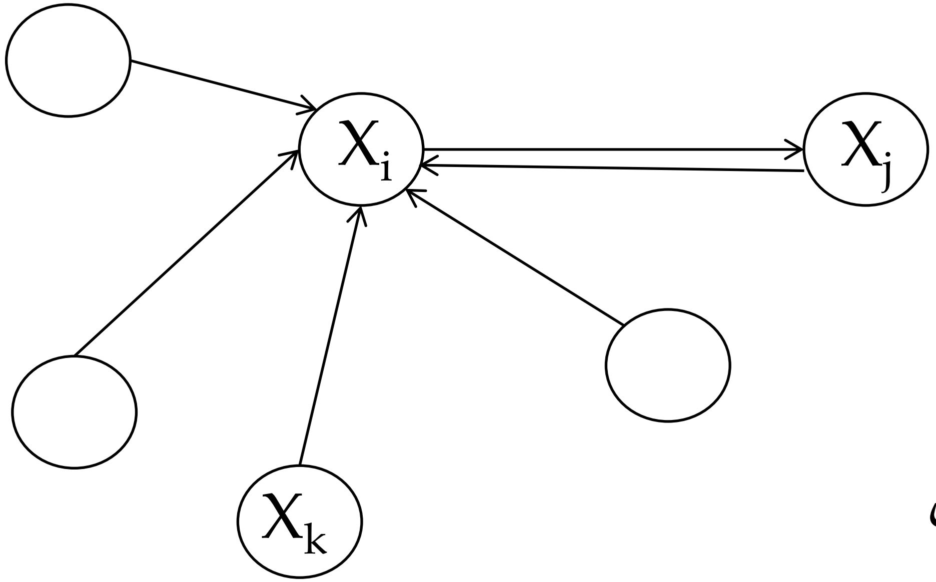
۱- اگر دامنه D_i بدون تغییر ماند آن‌گاه ادامه بده

۲- اگر $|D_i|=0$ آن‌گاه “false” برگردان

۳- هر همسایه X_k از X_i به‌جز X_j را به صف اضافه کن

- هنگامی که صف خالی شد CSP حاصل هم‌ارز با CSP اصلی خواهد بود
  - راه‌حل هر دو یکسان است اما متغیرهای با دامنه‌های کوچکتر منجر به جستجوی سریع‌تر می‌شود.

الگوریتم سازگاری کمان



برای هر کمان (X_i, X_j) در صف آن را از صف خارج کن

X_i را نسبت به X_j سازگار کن

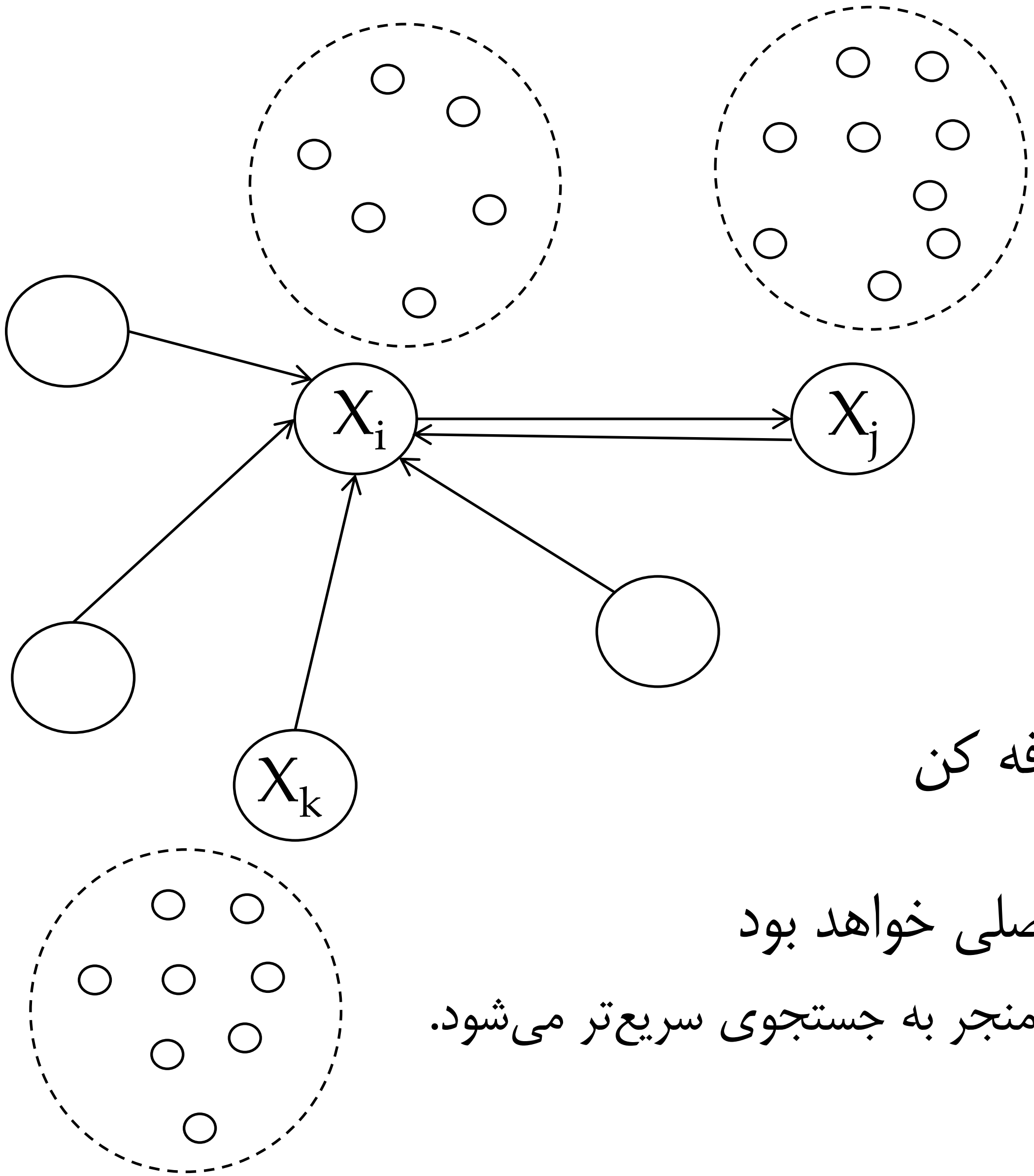
۱- اگر دامنه D_i بدون تغییر ماند آن‌گاه ادامه بده

۲- اگر $|D_i|=0$ آن‌گاه "false" برگردان

۳- هر همسایه X_k از X_i به‌جز X_j را به صف اضافه کن

- هنگامی که صف خالی شد CSP حاصل هم‌ارز با CSP اصلی خواهد بود
  - راه‌حل هر دو یکسان است اما متغیرهای با دامنه‌های کوچکتر منجر به جستجوی سریع‌تر می‌شود.

الگوریتم سازگاری کمان



برای هر کمان (X_i, X_j) در صف آن را از صف خارج کن

X_i را نسبت به X_j سازگار کن

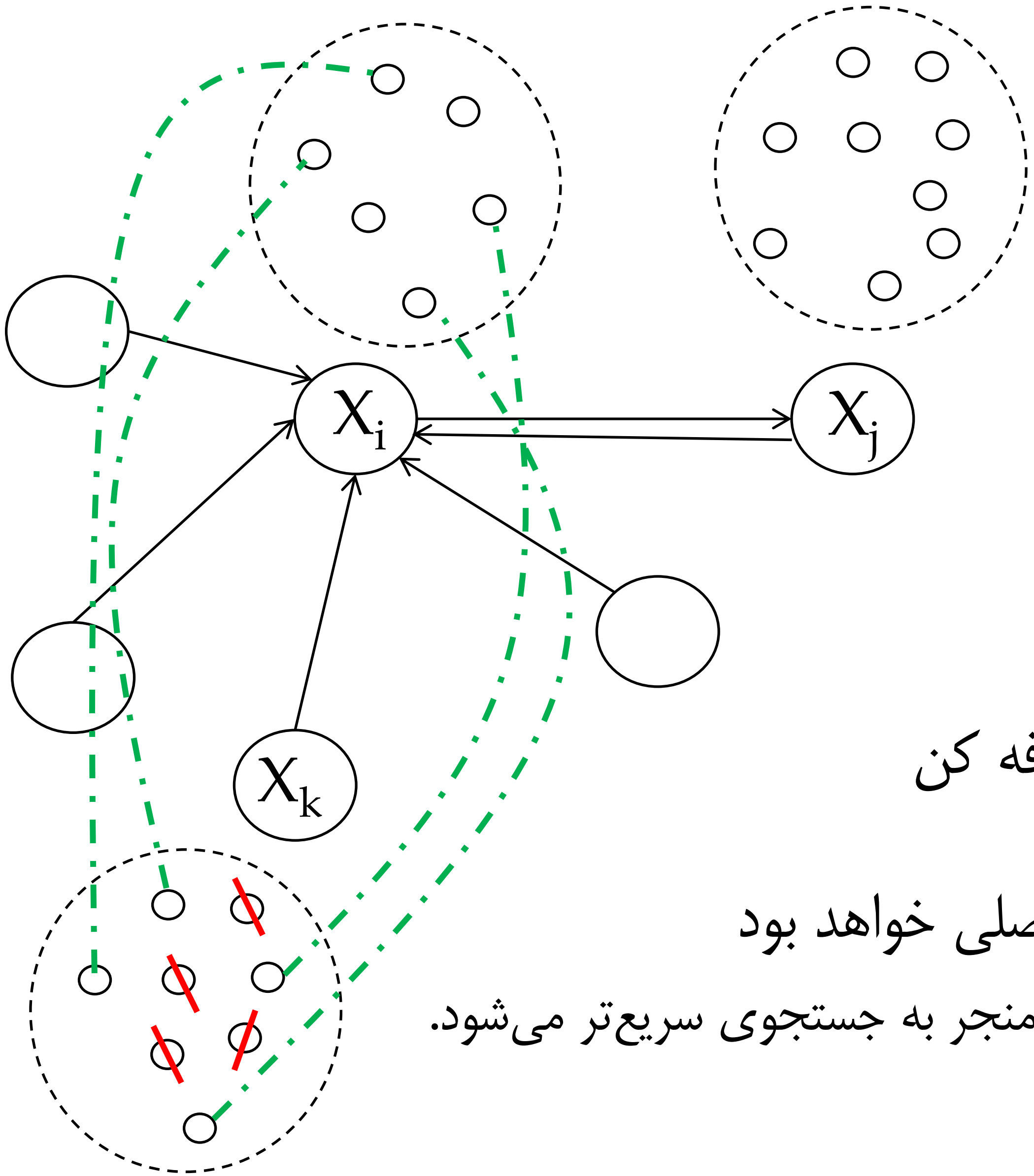
۱- اگر دامنه D_i بدون تغییر ماند آن‌گاه ادامه بده

۲- اگر $|D_i|=0$ آن‌گاه "false" برگردان

۳- هر همسایه X_k از X_i به‌جز X_j را به صف اضافه کن

- هنگامی که صف خالی شد CSP حاصل هم‌ارز با CSP اصلی خواهد بود
  - راه‌حل هر دو یکسان است اما متغیرهای با دامنه‌های کوچکتر منجر به جستجوی سریع‌تر می‌شود.

الگوریتم سازگاری کمان



برای هر کمان (X_i, X_j) در صف آن را از صف خارج کن

X_i را نسبت به X_j سازگار کن

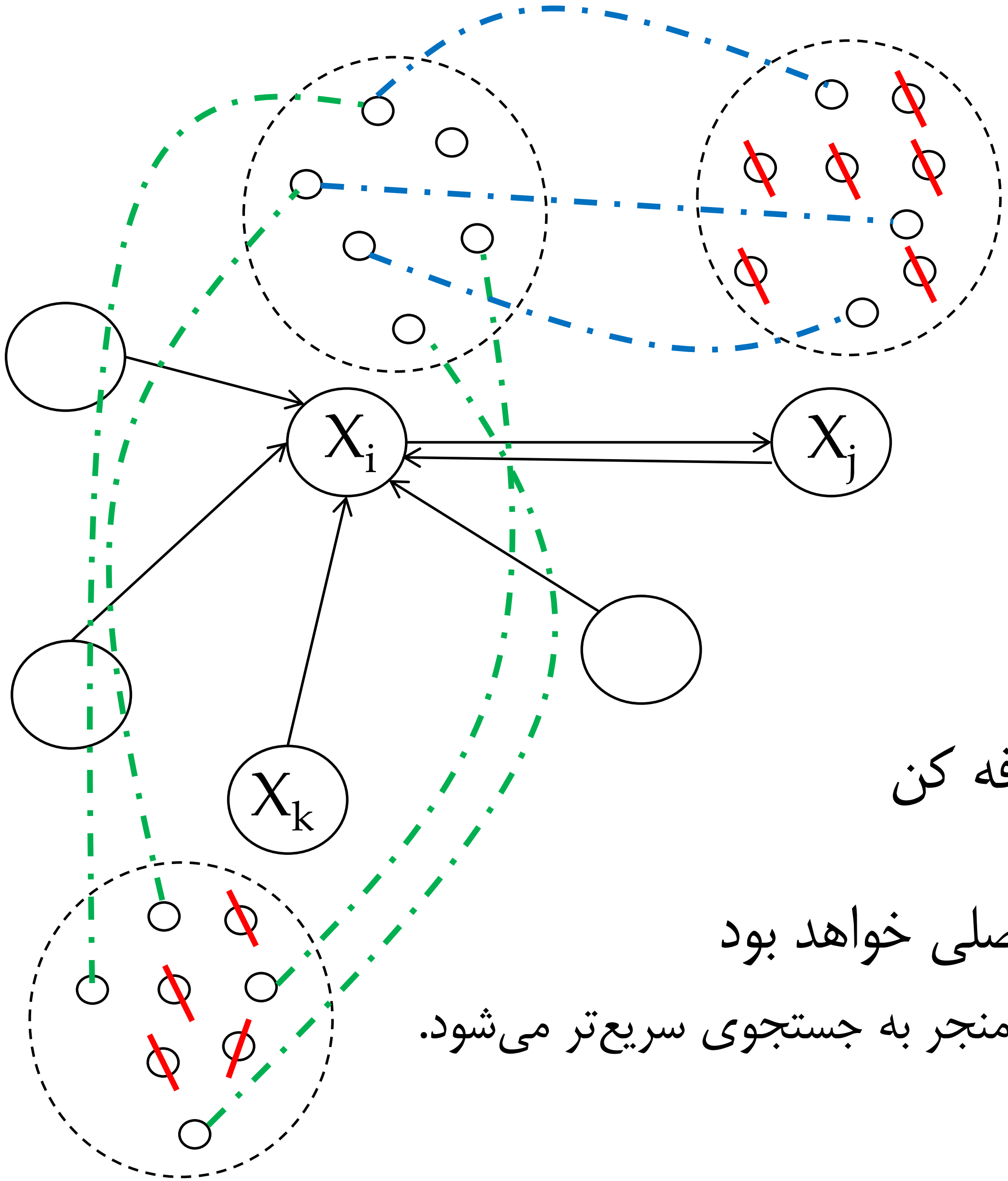
۱- اگر دامنه D_i بدون تغییر ماند آن‌گاه ادامه بده

۲- اگر $|D_i| = 0$ آن‌گاه “false” برگردان

۳- هر همسایه X_k از X_i به‌جز X_j را به صف اضافه کن

- هنگامی که صف خالی شد CSP حاصل هم‌ارز با CSP اصلی خواهد بود
  - راه‌حل هر دو یکسان است اما متغیرهای با دامنه‌های کوچکتر منجر به جستجوی سریع‌تر می‌شود.

الگوریتم سازگاری کمان



برای هر کمان (X_i, X_j) در صف آن را از صف خارج کن

X_i را نسبت به X_j سازگار کن

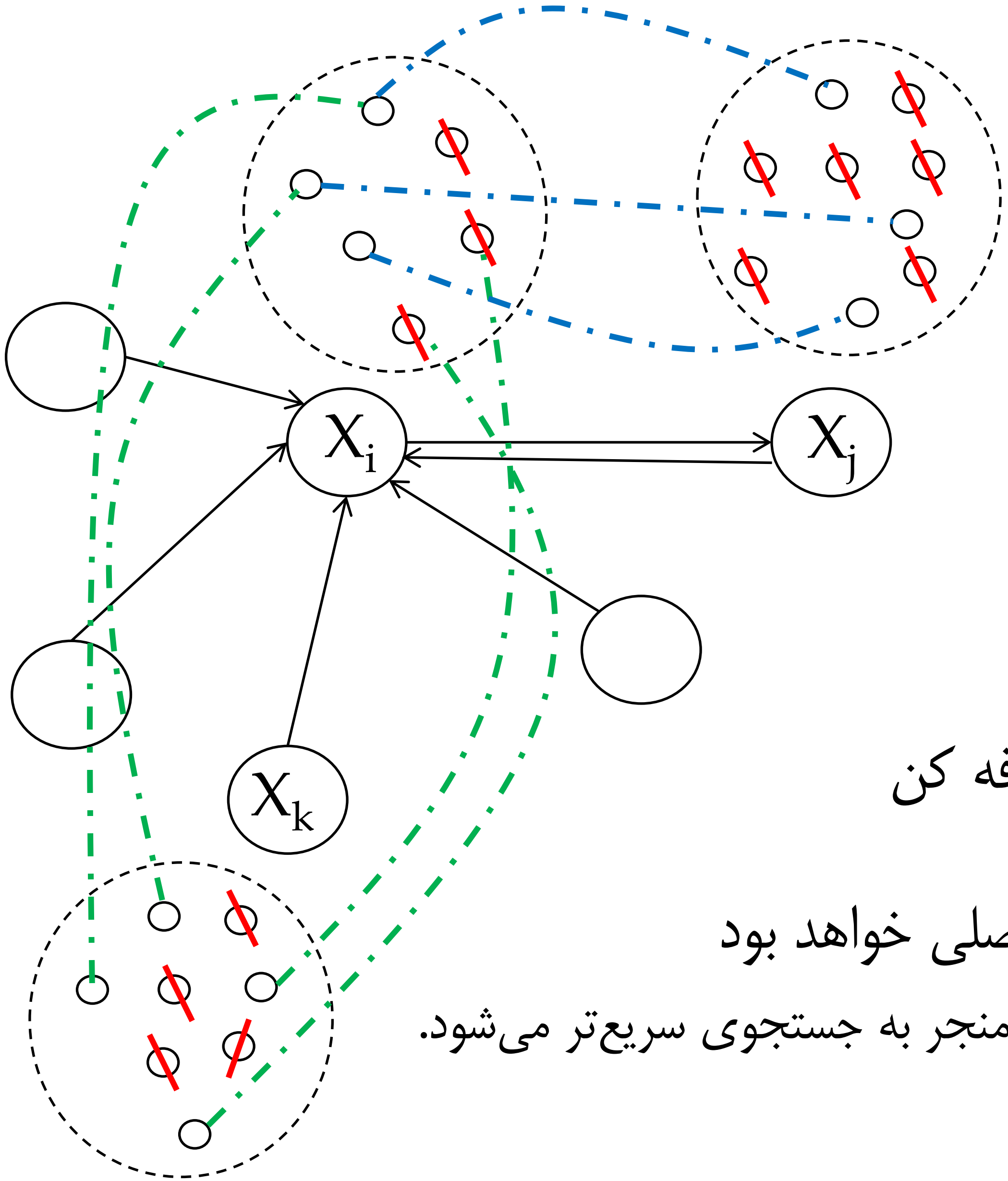
۱- اگر دامنه D_i بدون تغییر ماند آن‌گاه ادامه بده

۲- اگر $|D_i|=0$ آن‌گاه "false" برگردان

۳- هر همسایه X_k از X_i به‌جز X_j را به صف اضافه کن

- هنگامی که صف خالی شد CSP حاصل هم‌ارز با CSP اصلی خواهد بود
  - راه‌حل هر دو یکسان است اما متغیرهای با دامنه‌های کوچکتر منجر به جستجوی سریع‌تر می‌شود.

الگوریتم سازگاری کمان



برای هر کمان (X_i, X_j) در صف آن را از صف خارج کن

X_i را نسبت به X_j سازگار کن

۱- اگر دامنه D_i بدون تغییر ماند آن‌گاه ادامه بده

۲- اگر $|D_i|=0$ آن‌گاه “false” برگردان

۳- هر همسایه X_k از X_i به‌جز X_j را به صف اضافه کن

- هنگامی که صف خالی شد CSP حاصل هم‌ارز با CSP اصلی خواهد بود
  - راه‌حل هر دو یکسان است اما متغیرهای با دامنه‌های کوچکتر منجر به جستجوی سریع‌تر می‌شود.

الگوریتم سازگاری کمان AC-3

```
function AC-3(csp) returns false if an inconsistency is found and true otherwise
  inputs: csp, a binary CSP with components (X, D, C)
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (X_i, X_j) ← REMOVE-FIRST(queue)
    if REVISE(csp, X_i, X_j) then
      if size of D_i = 0 then return false
      for each X_k in X_i.NEIGHBORS - {X_j} do
        add (X_k, X_i) to queue
  return true
```

```
function REVISE(csp, X_i, X_j) returns true iff we revise the domain of X_i
  revised ← false
  for each x in D_i do
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j then
      delete x from D_i
      revised ← true
  return revised
```

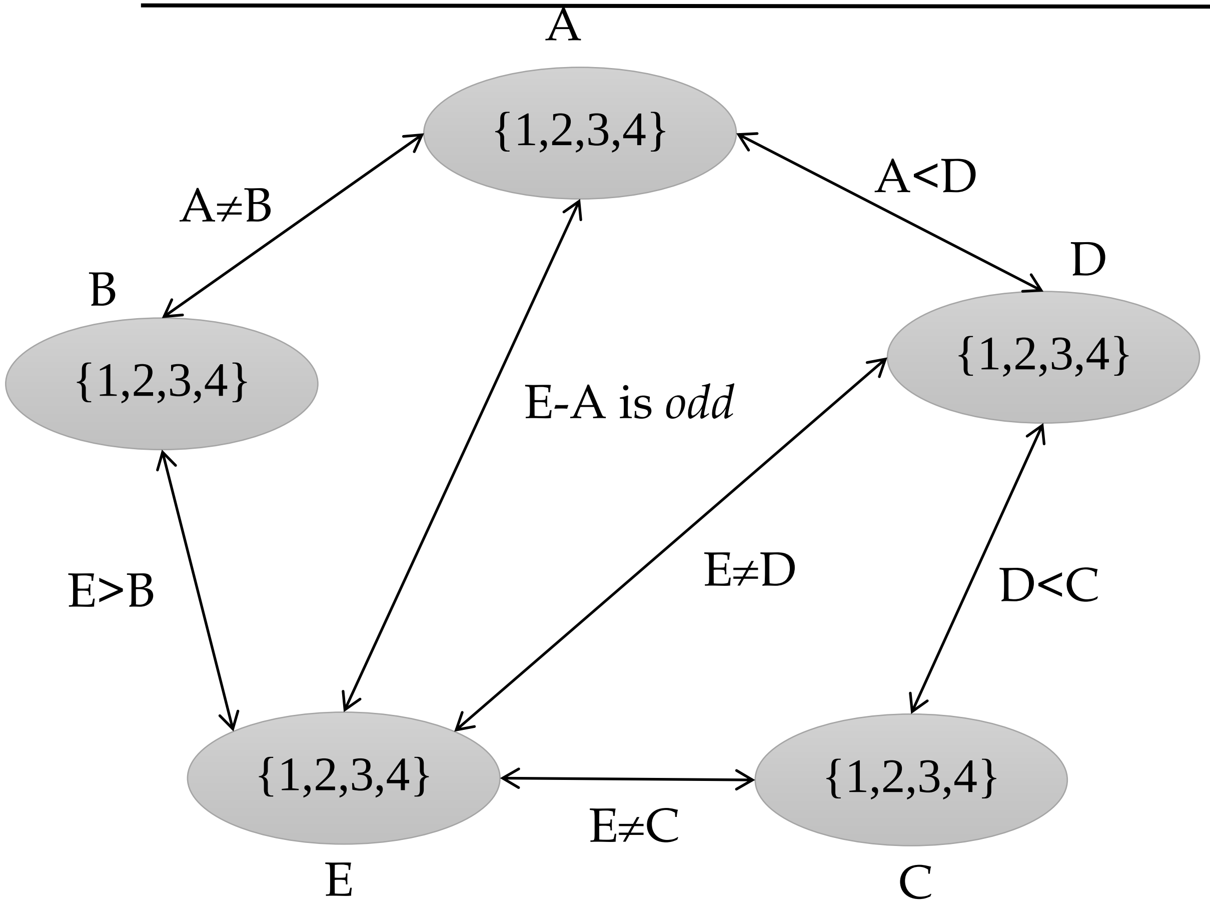
پیچیدگی الگوریتم AC-3

- یک CSP با شرایط زیر درنظر بگیرید
  - تعداد متغیرها: n
  - حداکثر سایز هر متغیر: d
  - تعداد محدودیت‌های دوتایی: c
- هر کمان (X_i, X_k) تنها d بار می‌تواند در صف قرار گیرد.
  - زیرا X_i حداکثر d مقدار برای حذف دارد.
- چک کردن سازگاری کمان: $O(d^2)$

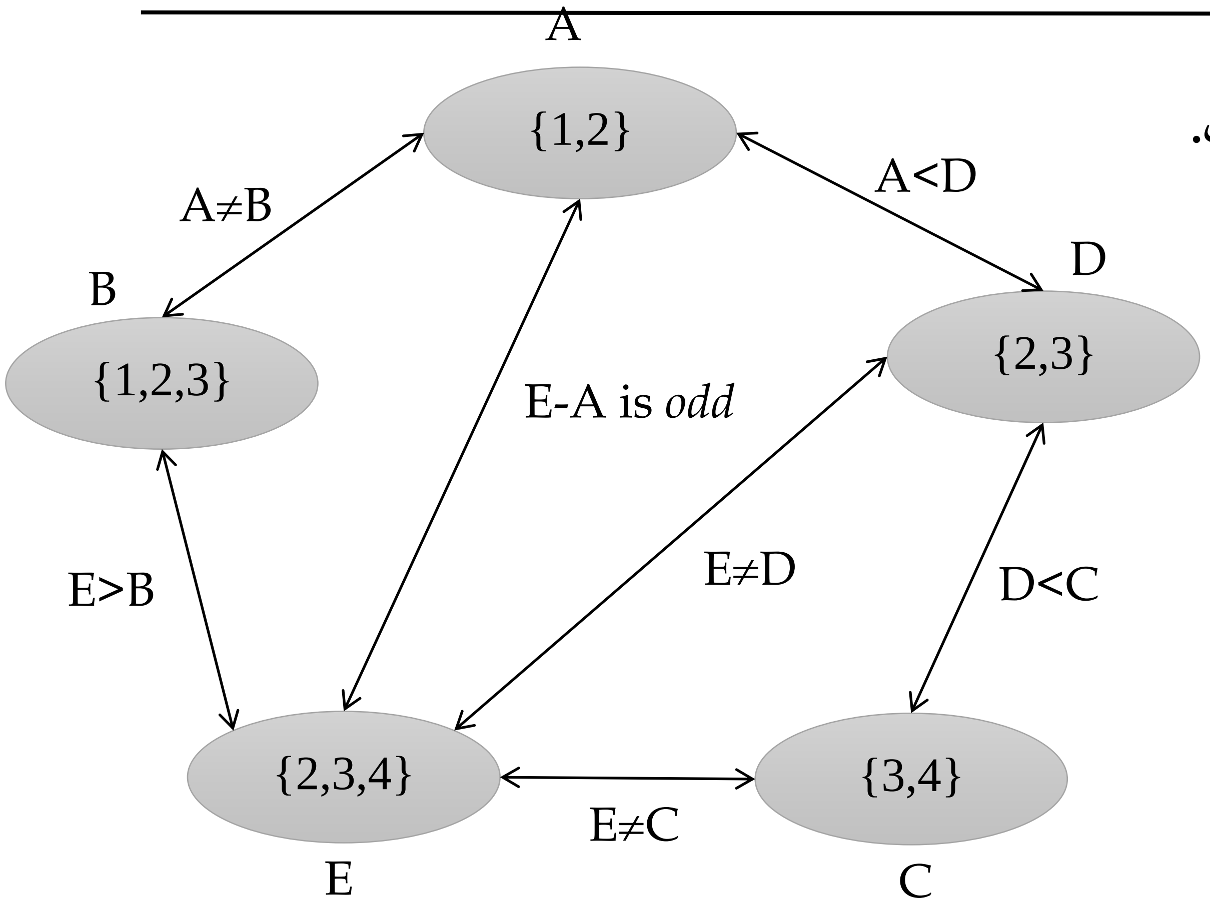
$O(cd^3)$

مثال– سازگاری کمان

گراف محدودیت زیر را در نظر بگیرید و مراحل الگوریتم AC-3 را بر روی آن نشان دهید



مثال– سازگاری کمان ...



صف اولیه مجموعه تمامی کمان‌های گراف محدودیت است.

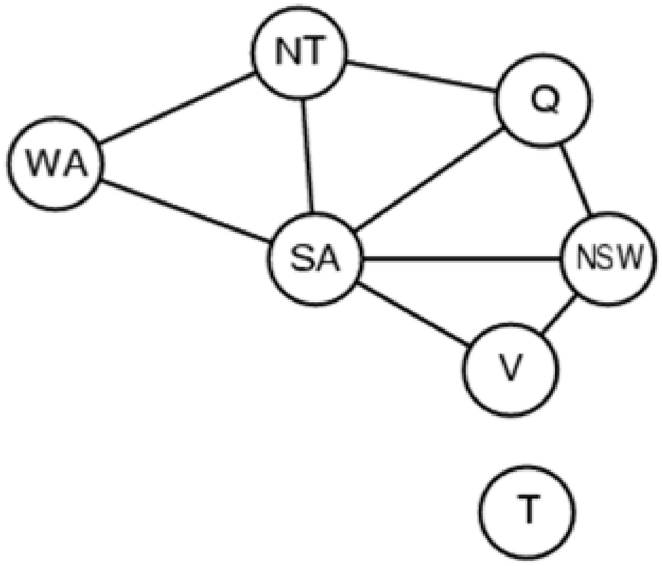
Queue= {<E,B>, <B,E>, <A,B>, <B,A>, …}

خلاصه‌ای از مراحل انجام شده توسط الگوریتم: نمایش مقادیر حذف شده با بررسی هر کمان

E=1	<E, B>
B=4	<B, E>
D=4	<D, C>
C=1	<C, D>
A=3, A=4	<A, D>
D=1	<D, A>
C=2	<C, D>

سازگاری کمان در مسئله رنگ‌آمیزی نقشه

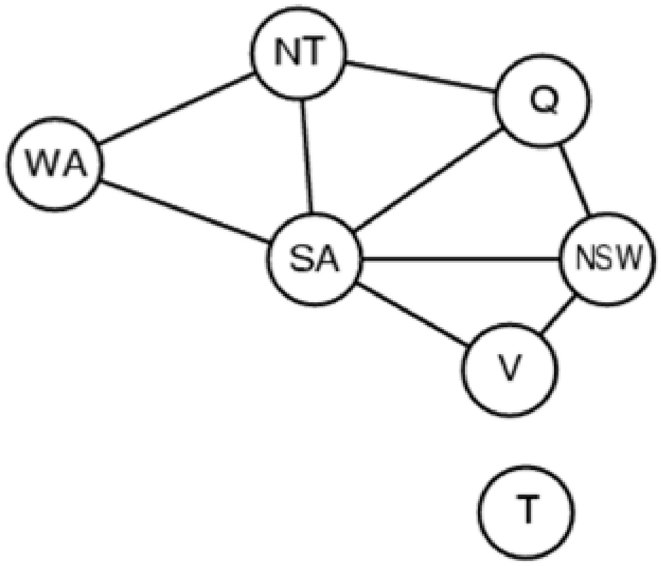
- در مورد یک مسئله رنگ‌آمیزی برای یک نقشه دلخواه، تمامی زوج متغیرها سازگار کمان هستند اگر $|D_i| \geq 2$ برای هر i برقرار باشد.
- مثال: فرض کنید هر ناحیه را بتوان تنها با دو رنگ قرمز و آبی رنگ‌آمیزی کرد.
  - آیا متغیرها در این حالت سازگار کمان هستند؟
  - آیا اعمال سازگاری کمان نیاز است؟
  - آیا راه‌حلی برای مسئله وجود دارد؟



- نتیجه: در برخی مسائل، سازگاری کمان برای استنتاج کافی نیست و ما نیاز به مفهوم قوی‌تری از سازگاری داریم.

سازگاری مسیر (Path-consistency)

- مجموعه‌ی دو متغیره $\{X_i,X_j\}$ نسبت به متغیر سوم X_m سازگار مسیر است اگر برای هر انتساب سازگار $\{X_i=a, X_j=b\}$ با $\{X_i,X_j\}$، یک انتساب برای X_m وجود داشته باشد که سازگاری روی $\{X_i,X_m\}$ و $\{X_m,X_j\}$ برقرار باشد.



- مثال: اعمال سازگاری مسیر در مورد رنگ‌آمیزی نقشه با دو رنگ
  - سازگاری {SA,WA} نسبت به NT؟

انتساب‌های سازگار به {SA,WA} به‌صورت زیر است:

{SA=red, WA=blue} یا {SA=blue, WA=red}

در این صورت NT={} خواهد بود و هیچ راه‌حلی برای مسئله وجود ندارد.

سازگاری مرتبه kام (k-consistency)

- یک CSP دارای سازگاری مرتبهٔ k است اگر برای هر مجموعهٔ k-1 عضوی از متغیرها و برای هر انتساب سازگار به آن‌ها، همیشه یک مقدار سازگار یافت شود که بتوان به هر متغیر kام انتساب داد.
  - سازگاری مرتبهٔ ۱ = سازگاری گره
  - سازگاری مرتبهٔ ۲ = سازگاری کمان
  - سازگاری مرتبهٔ ۳ = سازگاری مسیر

تست ۲

مسئله ارضای محدودیت رنگ کردن نقشه زیر را در نظر بگیرید. فرض کنید می‌خواهیم این نقشه را با تنها یک رنگ، رنگ‌آمیزی کنیم (شهرهای مجاور نباید هم‌رنگ باشند). با این فرض، گراف محدودیت این مسئله به ازای چه مقادیری از k دارای خاصیت k-consistency است؟

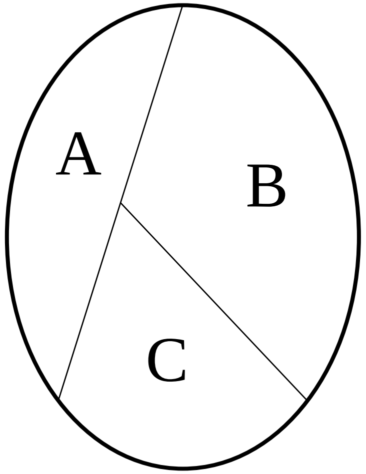
(مهندسی کامپیوتر دولتی ۹۳)

۱) فقط $k=0$

۲) فقط $k=1$

۳) $k=1$ و $k=2$

✓ ۴) $k=1$ و $k=3$



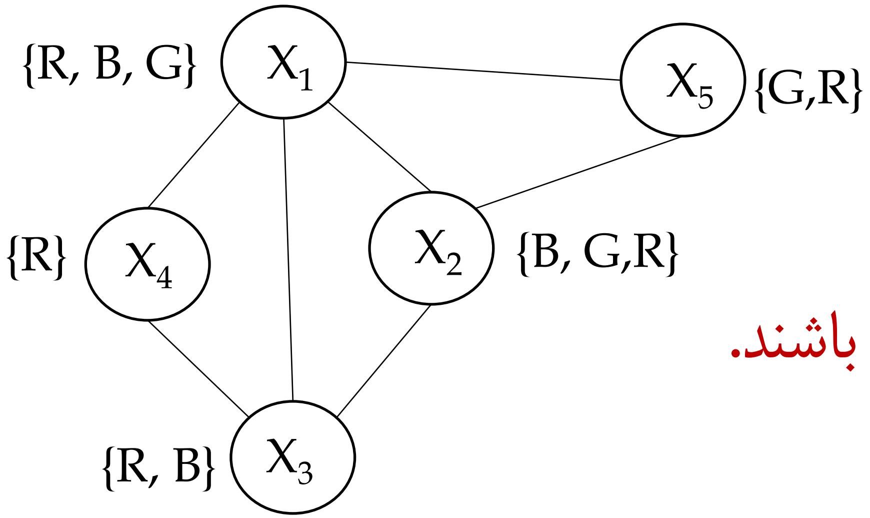
سازگاری قوی مرتبه k (Strongly k-consistent)

- یک CSP قویاً k سازگار است اگر دارای سازگاری مرتبهٔ k، مرتبهٔ k-1، مرتبهٔ k-2، تا سازگار مرتبهٔ ۱ نیز باشد.
- فرض کنید یک CSP با n گره هر کدام حداکثر با d مقدار داشته باشیم و این CSP دارای سازگاری قوی مرتبهٔ n است.
  - در ابتدا، یک مقدار سازگار را برای X_1 انتخاب می‌کنیم. از آنجا که گراف دارای سازگاری مرتبهٔ ۲ است، مطمئن هستیم که می‌توان مقداری سازگار یافت که بتوان به X_2 اختصاص داد و به ترتیب، می‌توان مقادیر سازگاری برای بقیه متغیرها یافت.
  - می‌توان این مسأله را بدون انجام عقب‌گرد، حل کرد.
  - راه‌حل مسأله حداکثر با مرتبهٔ زمانی $O(n^2d)$ پیدا می‌شود.

X_1	0
X_2	d
X_3	2d
⋮	⋮
X_n	(n-1)d

کدام سطح سازگاری؟

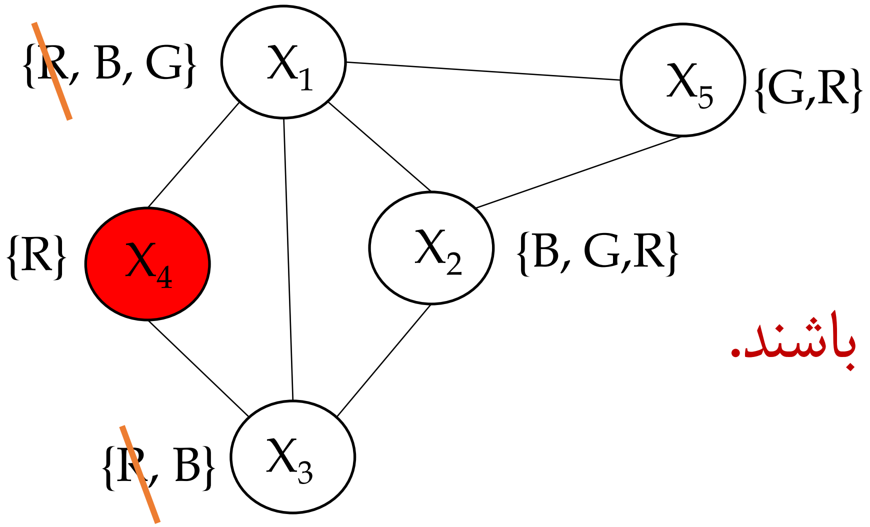
- هر الگوریتم برای برقرای سازگاری مرتبه k در بدترین حالت از نظر زمان و حافظه نمایی است.
- می‌بایست مصالحه‌ای (trade off) میان زمان مورد نیاز برای برقراری سازگاری kام و مقدار حذفیات از فضای جستجو برقرار کرد.
  - در عمل معمولا سازگاری ۲ انجام می‌شود و کم‌تر از سازگاری ۳ استفاده می‌شود.



برخورد با محدودیت‌های ویژه

۱- محدودیت Alldiff: تمامی متغیرها باید دارای مقادیر متفاوتی باشند.

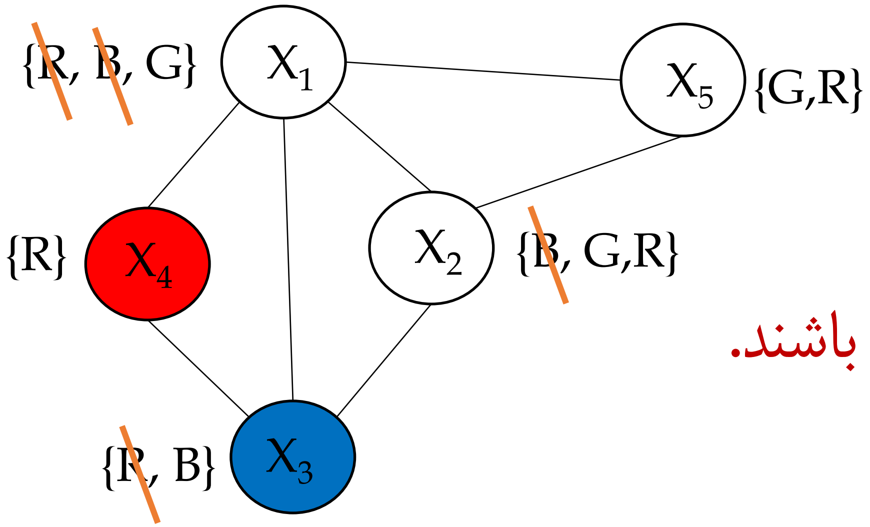
- یک شکل ساده از کشف ناسازگاری برای این محدودیت
  - اگر m متغیر شامل این محدودیت باشند ولی تنها n مقدار که n<m است برای انتساب موجود باشد دیگر این محدودیت نمی‌تواند برآورده شود.
- یک الگوریتم ساده برای اعمال محدودیت Alldiff
  - در ابتدا تمام متغیرهایی که دارای دامنهٔ واحد می‌باشند را حذف و مقدار مربوط به دامنه‌ی آن‌ها را نیز از دامنه‌ی سایر متغیرها حذف می‌کنیم.
  - این کار را برای تمام متغیرهای تک مقداره تکرار می‌کنیم.
  - اگر در پایان دامنه‌ی یک متغیر تهی شود یا تعداد متغیر بیشتر از تعداد مقادیر دامنه باقی مانده باشد، ناسازگاری در مسأله تشخیص داده خواهد شد.



برخورد با محدودیت‌های ویژه

۱– محدودیت Alldiff: تمامی متغیرها باید دارای مقادیر متفاوتی باشند.

- یک شکل ساده از کشف ناسازگاری برای این محدودیت
  - اگر m متغیر شامل این محدودیت باشند ولی تنها n مقدار که n<m است برای انتساب موجود باشد دیگر این محدودیت نمی‌تواند برآورده شود.
- یک الگوریتم ساده برای اعمال محدودیت Alldiff
  - در ابتدا تمام متغیرهایی که دارای دامنهٔ واحد می‌باشند را حذف و مقدار مربوط به دامنه‌ی آن‌ها را نیز از دامنه‌ی سایر متغیرها حذف می‌کنیم.
  - این کار را برای تمام متغیرهای تک مقداره تکرار می‌کنیم.
  - اگر در پایان دامنه‌ی یک متغیر تهی شود یا تعداد متغیر بیشتر از تعداد مقادیر دامنه باقی مانده باشد، ناسازگاری در مسأله تشخیص داده خواهد شد.



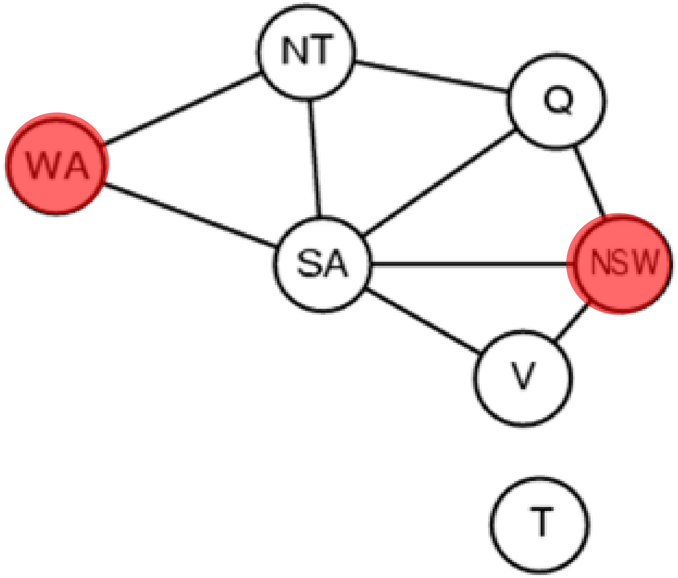
برخورد با محدودیت‌های ویژه

۱– محدودیت Alldiff: تمامی متغیرها باید دارای مقادیر متفاوتی باشند.

- یک شکل ساده از کشف ناسازگاری برای این محدودیت
  - اگر m متغیر شامل این محدودیت باشند ولی تنها n مقدار که n<m است برای انتساب موجود باشد دیگر این محدودیت نمی‌تواند برآورده شود.
- یک الگوریتم ساده برای اعمال محدودیت Alldiff
  - در ابتدا تمام متغیرهایی که دارای دامنهٔ واحد می‌باشند را حذف و مقدار مربوط به دامنه‌ی آن‌ها را نیز از دامنه‌ی سایر متغیرها حذف می‌کنیم.
  - این کار را برای تمام متغیرهای تک مقداره تکرار می‌کنیم.
  - اگر در پایان دامنه‌ی یک متغیر تهی شود یا تعداد متغیر بیشتر از تعداد مقادیر دامنه باقی مانده باشد، ناسازگاری در مسأله تشخیص داده خواهد شد.

برخورد با محدودیت‌های ویژه ...

۱- محدودیت Alldiff: تمامی متغیرها باید دارای مقادیر متفاوتی باشند.



- مثال: رنگ‌آمیزی نقشه
  - فرض کنید انتساب زیر صورت گرفته است

{WA=red, NSW=red}

  - محدودیت سراسری زیر را داریم

Alldiff(SA, NT, Q)

  - بعد از اعمال AC-3 دامنه متغیرها به‌صورت زیر خواهد شد

{green , blue}

  - سه متغیر و دو رنگ داریم پس محدودیت Alldiff ناسازگار می‌شود.

برخورد با محدودیت‌های ویژه ...

۲- محدودیت منبع (محدودیت بیشینه یا Atmost)

- مثال- انتساب افراد (منابع) به کارهای متفاوت
- فرض کنید $PA_1,\ldots,PA_4$ به تعداد افرادی اشاره می‌کند که برای انجام چهار کار متفاوت در نظر گرفته شده‌اند.
- محدودیتی که براساس آن، تعداد افراد در مجموع نباید از ۱۰ نفر بیشتر باشد به صورت atmost $(10,PA_1,PA_2,PA_3,PA_4)$ نمایش داده می‌شود.

برخورد با محدودیت‌های ویژه ...

۲– محدودیت منبع (محدودیت بیشینه یا Atmost)

- می‌توان با بررسی مجموع حداقل مقادیر دامنه‌های فعلی، ناسازگاری را تشخیص داد.
  - مثلاً اگر هر متغیر دارای دامنهٔ {۳،۴،۵،۶} باشد، محدودیت بیشینه، تأمین نمی‌شود.
- می‌توان برای اعمال سازگاری، مقدار بیشینه در هر دامنه را، به شرطی که با مقدار کمینه در دامنه‌های دیگر ناسازگار باشد، حذف نمود.
  - اگر هر متغیری در مثال فوق، دارای دامنهٔ {۲،۳،۴،۵،۶} باشد، می‌توان به منظور رسیدن به سازگاری، مقادیر ۵ و ۶ را از دامنهٔ متغیرها حذف نمود.

برخورد با محدودیت‌های ویژه ...

۳- محدودیت حدود (Bounds): در مسائلی که نمی‌توان دامنه‌ی هر متغیر را به‌صورت مجموعه‌ای از اعداد صحیح نشان داد، از حدود بالا و پایین برای نمایش استفاده می‌کنند و انتشار محدودیت حدود را به کار می‌برند.

- مثال- برنامه‌ریزی خطوط هوایی (ظرفیت هواپیما)
  - دو پرواز F_1 و F_2 وجود دارد که هر کدام به‌ترتیب ظرفیت ۱۶۵ و ۳۸۵ را دارند.
  - دامنه‌ی اولیه تعداد مسافران برای هر پرواز $D_1=[0,165]$ و $D_2=[0,385]$ است.
  - محدودیت: مجموع تعداد مسافران هر دو پرواز ۴۲۰ نفر شود.
  - انتشار محدودیت حدود: دامنه‌ها به $D_1=[35,165]$ و $D_2=[255,385]$ تبدیل می‌شوند.

حل سودوکو تنها از طریق استنتاج (بدون جستجو)

- متغیرها: هر یک از مربع‌های خالی
- دامنه: {1,2,3,4,5,6,7,8,9}

✫ محدودیت‌ها:

✫ ۹ محدودیت Alldiff برای سطرها

✫ ۹ محدودیت Alldiff برای ستون‌ها

✫ ۹ محدودیت Alldiff برای ناحیه‌های ۹تایی

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

جستجوی عقب‌گرد برای CSPها

برای حل بسیاری از مسائل CSP علاوه‌بر استنتاج نیاز به جستجو نیز وجود دارد. برای این منظور باید به فرموله‌سازی مسئله (افزایشی یا کامل) بپردازیم.

در این بخش با درنظر گرفتن فرموله‌سازی افزایشی الگوریتمی برای حل CSPها ارائه می‌دهیم.

فرموله‌بندی افزایشی CSPها

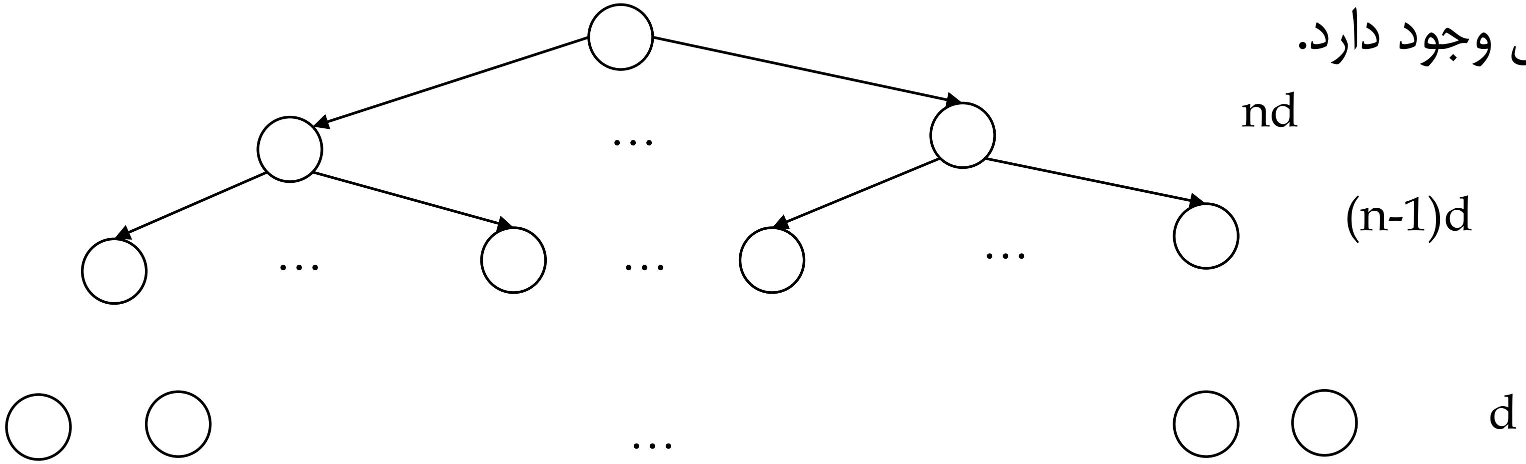
- حالات: مجموعه‌ی تمامی انتساب‌های سازگار
- حالت اولیه: تمام متغیرها بدون مقدار، یعنی انتساب تهی
- اقدام‌ها: انتساب مقدار به یکی از متغیرهایی که مقدار ندارد به طوری‌که با مقادیر متغیرهای قبلی ناسازگار نباشد.
  - در صورت عدم وجود انتساب‌های مجاز شکست می‌خورد.
- آزمون هدف: آیا انتساب فعلی کامل است؟
  - چون عملیات به‌گونه‌ای انجام می‌شود که هر انتساب سازگار باشد در این آزمون تنها بررسی می‌شود که تمام متغیرها مقداردهی شده‌اند یا خیر.
- هزینه مسیر: هزینه یکسان برای تمام مراحل (هزینه اهمیت ندارد)

استفاده از الگوریتم‌های جستجوی استاندارد

- برای تمام مسائل CSP این فرموله‌سازی به‌کار می‌رود.
- الگوریتم جستجوی عمق محدود برای یک CSP با n متغیر و سایز دامنه d
  - هر پاسخ در عمق n و با n متغیر ظاهر می‌شود.
  - در سطح یک فاکتور انشعاب برابر است با nd در سطح دو (n-1)d و …
  - بنابراین تعداد برگ‌های درخت جستجو برابر است با $n!d^n$

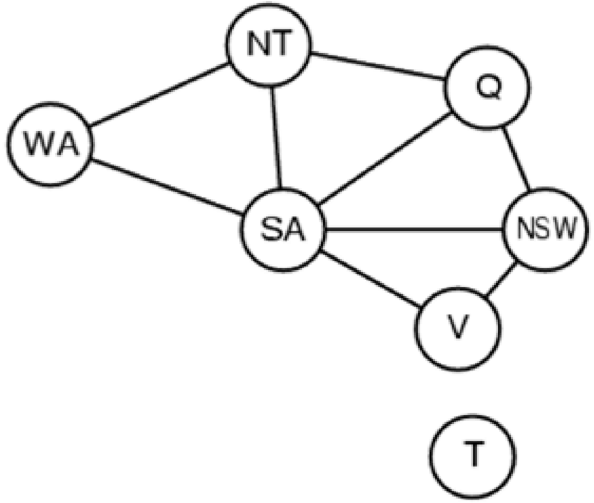
$$(nd) * [(n-1)d] * [(n-2)d] * \ldots d = n!d^n$$

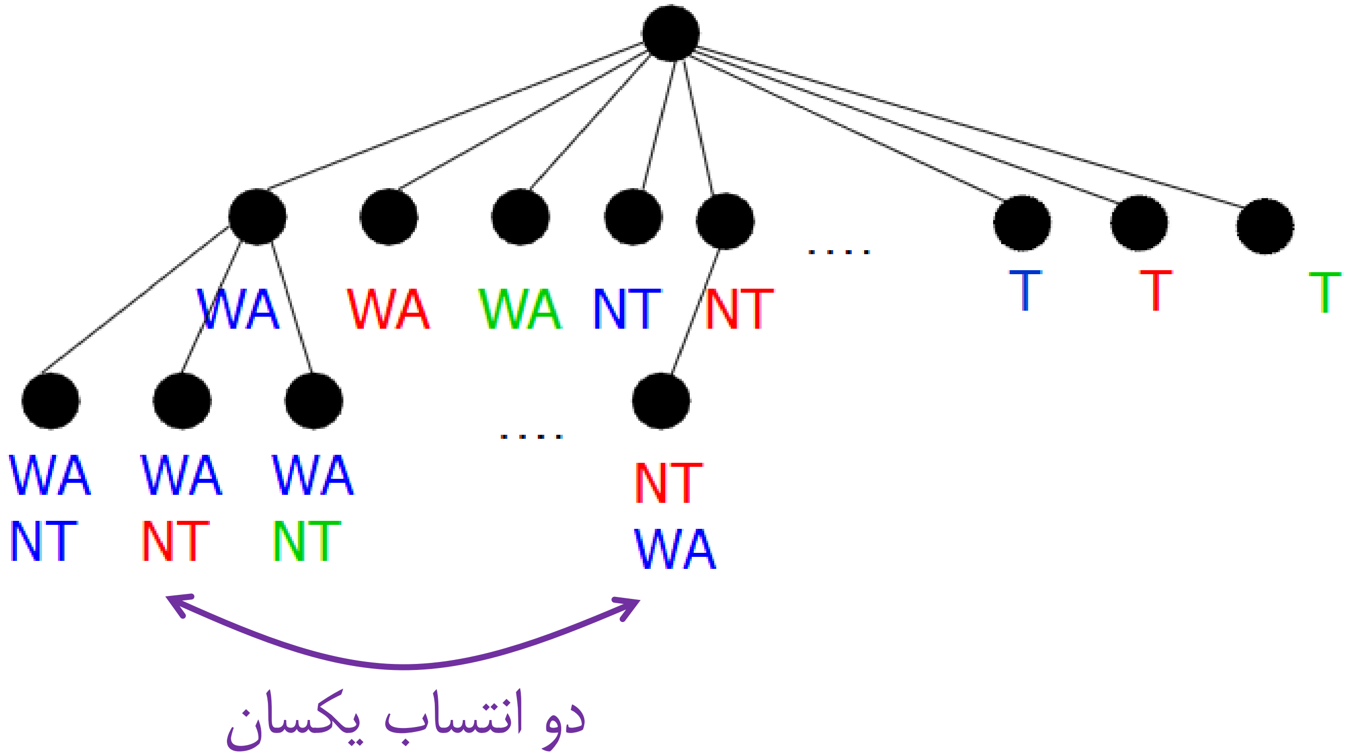
  - با این‌وجود تنها d^n انتساب کامل وجود دارد.



بهبود روش قبل با خاصیت جابه‌جایی CSPها

- یک مسئله دارای خاصیت جابه‌جایی (commutativity) است اگر ترتیب اِعمال هر مجموعه از اقدامات هیچ تأثیری بر روی نتیجه ایجاد نکند.
  - مستقل از ترتیب انتساب مقادیر به متغیرها، به انتساب‌های جزئی یکسانی خواهیم رسید.

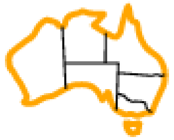




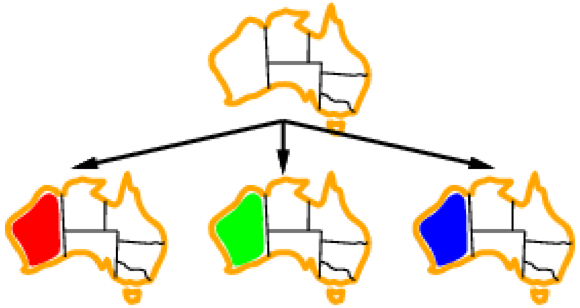
جستجوی عقب‌گرد (Backtracking)

- جستجوی عقب‌گرد، جستجوی اول عمقی است که در هر زمان برای یک متغیر مقداری در نظر گرفته می‌شود و اگر هیچ مقدار مجازی برای انتساب به یک متغیر باقی نمانده باشد به عقب برمی‌گردد.
  - یک متغیر بدون انتساب را انتخاب کن
  - تمام مقادیر دامنه آن متغیر را به نوبت امتحان کن
  - اگر مقداری سازگار با مقادیر انتساب‌یافته قبلی پیدا کردی به آن متغیر انتساب بده در غیر این صورت به عقب برگرد
- تعداد برگ‌ها؟ d^n
  - هرس زیردرخت‌ها

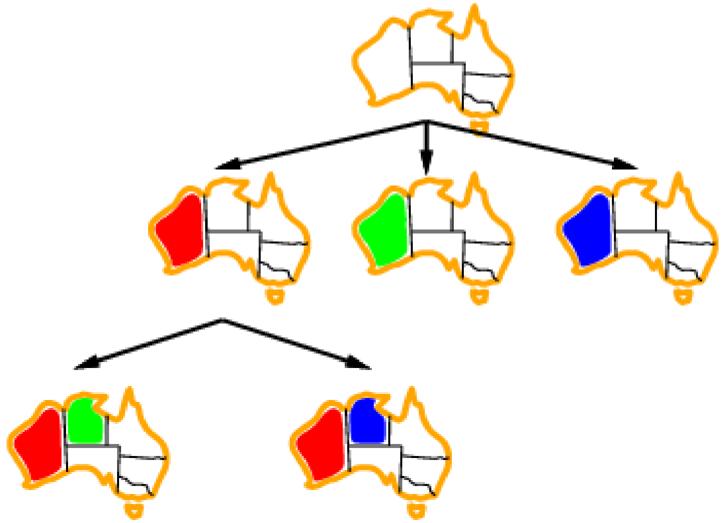
جستجوی عقب‌گرد–مثال رنگ‌آمیزی نقشه



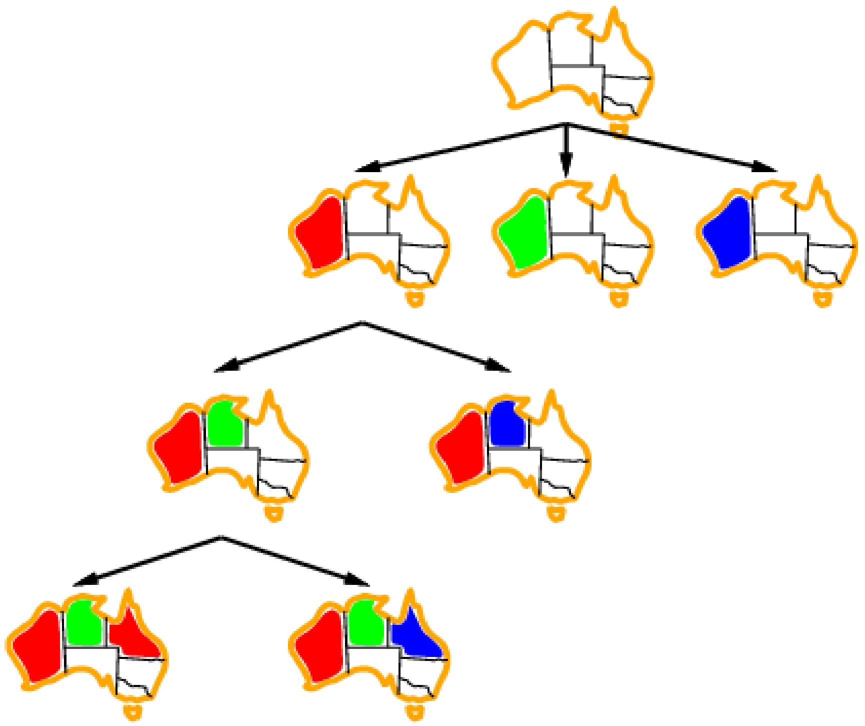
جستجوی عقب‌گرد–مثال رنگ‌آمیزی نقشه



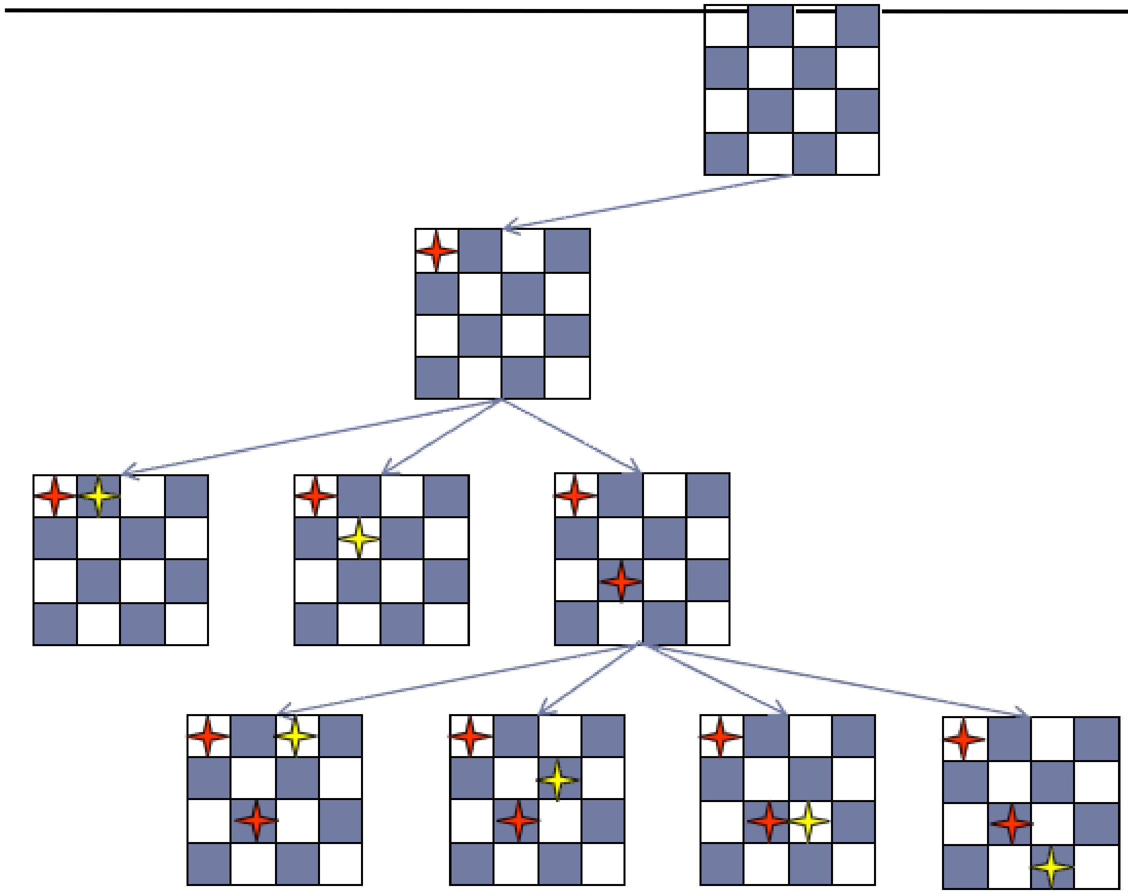
جستجوی عقب‌گرد–مثال رنگ‌آمیزی نقشه



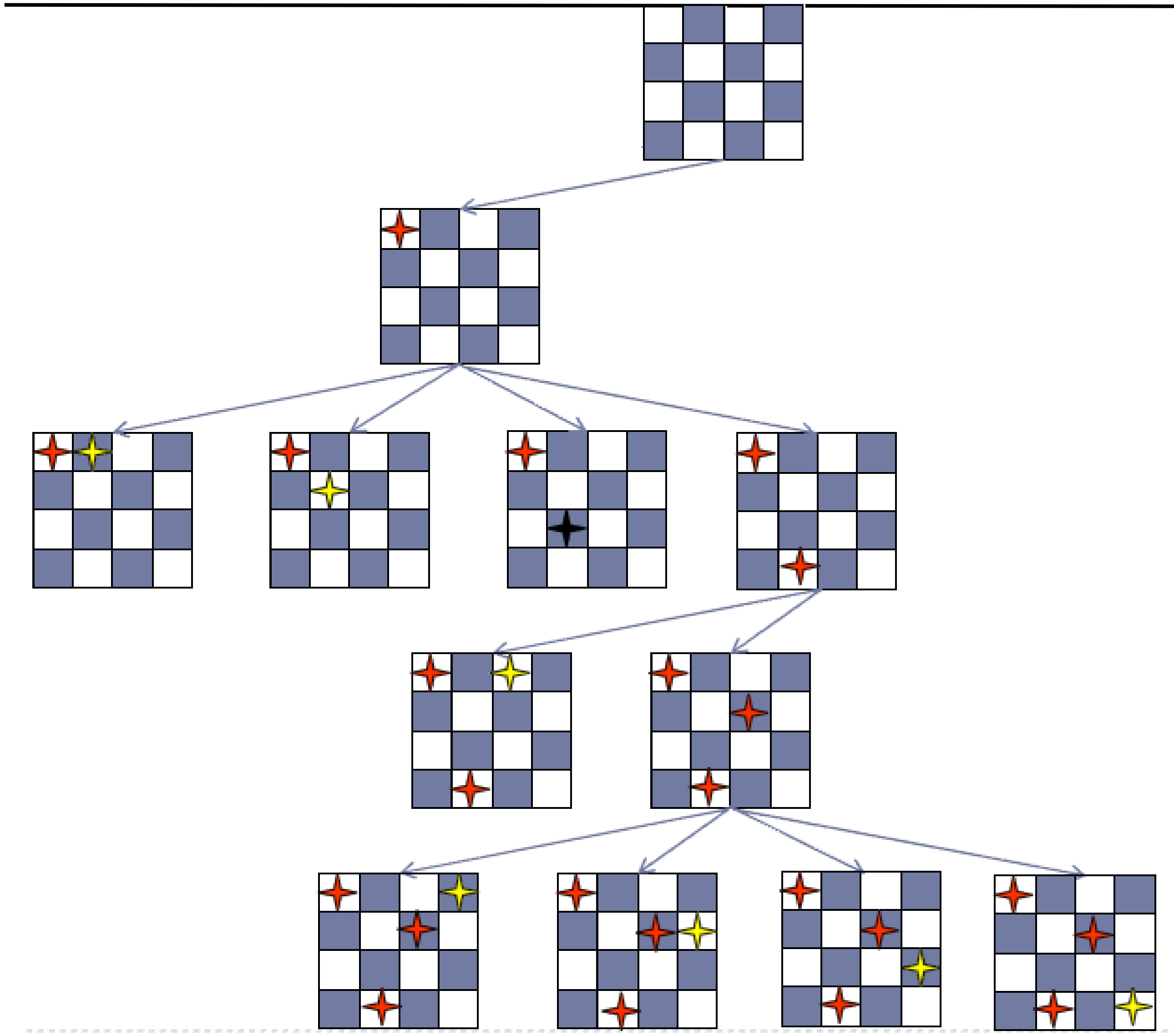
جستجوی عقب‌گرد–مثال رنگ‌آمیزی نقشه



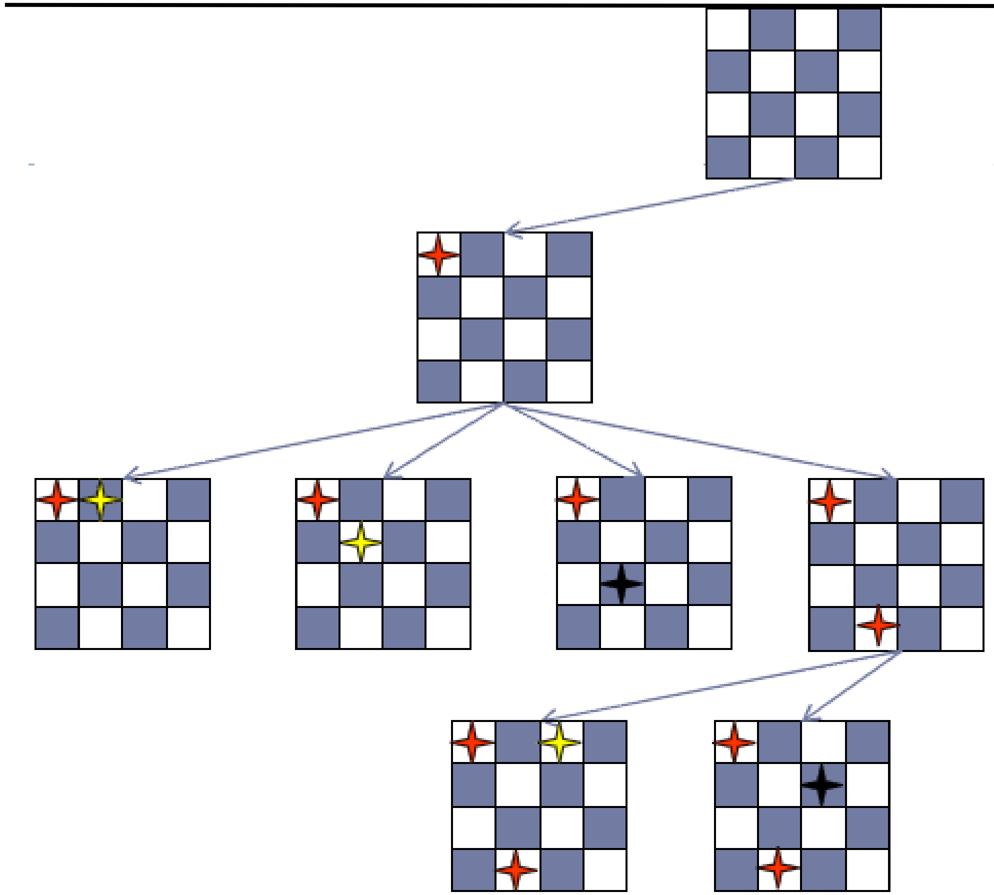
جستجوی عقب‌گرد–مثال ۴ وزیر



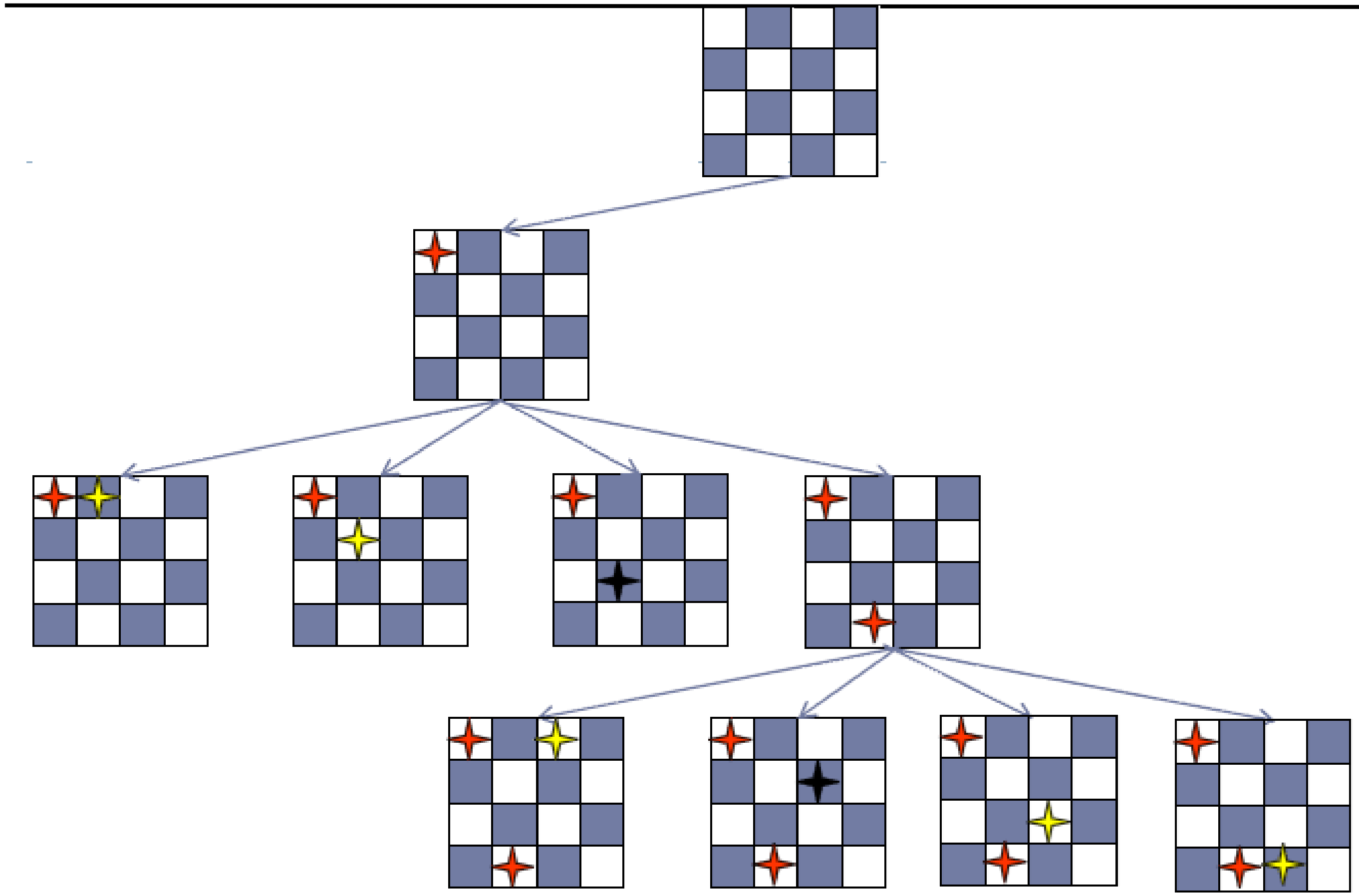
جستجوی عقب‌گرد–مثال ۴ وزیر



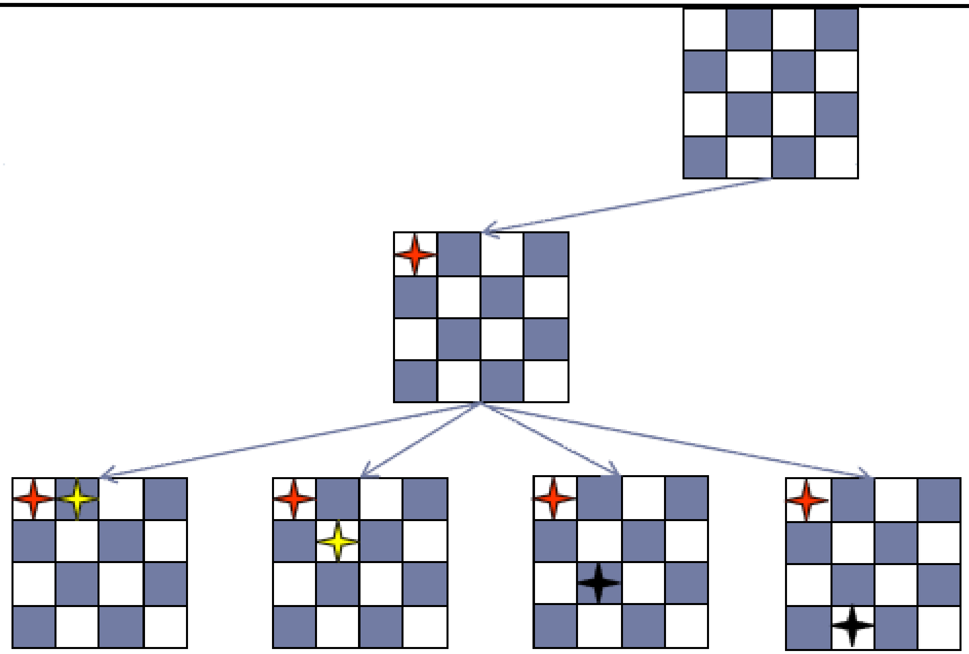
جستجوی عقب‌گرد–مثال ۴ وزیر



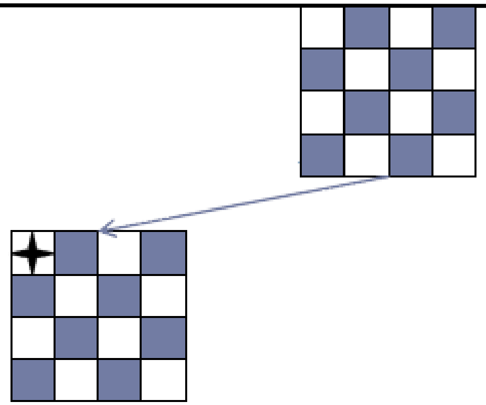
جستجوی عقب‌گرد–مثال ۴ وزیر



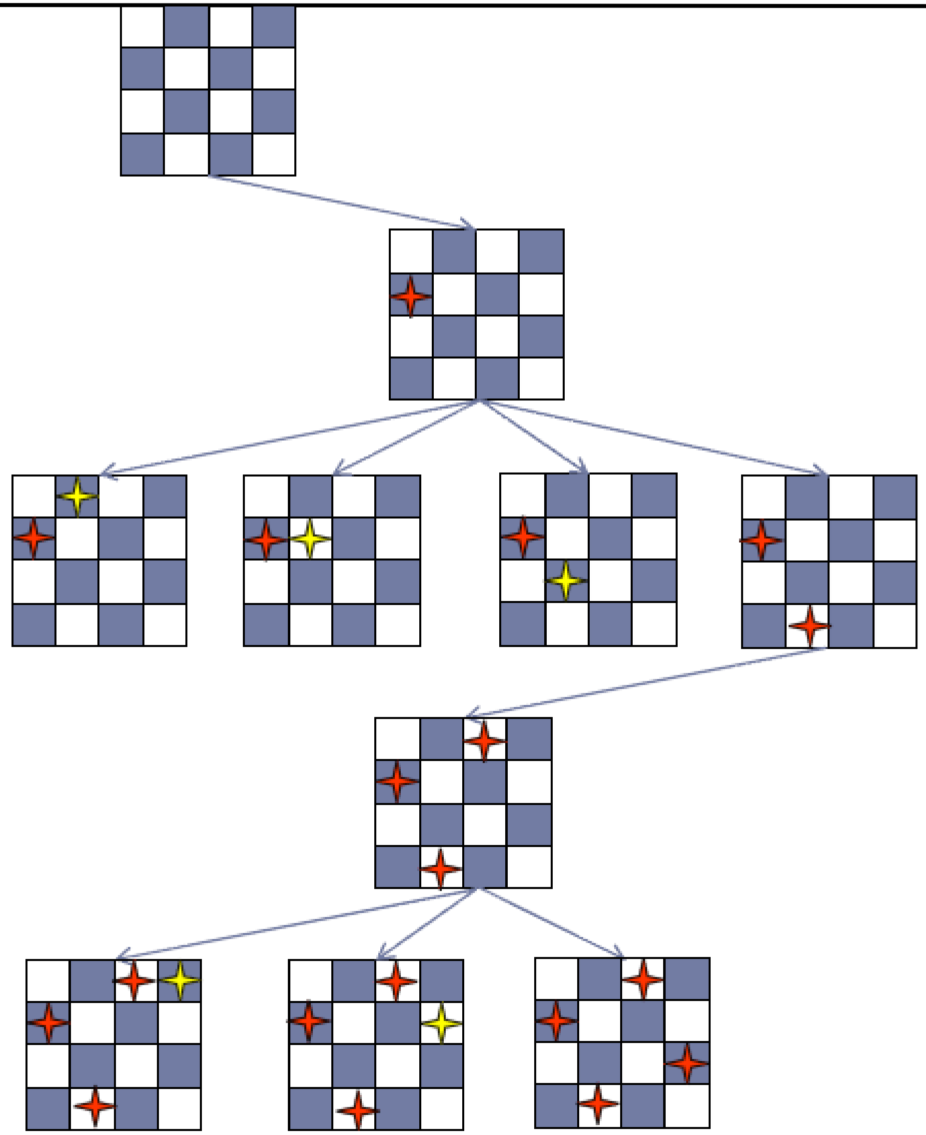
جستجوی عقب‌گرد–مثال ۴ وزیر



جستجوی عقب‌گرد–مثال ۴ وزیر



جستجوی عقب‌گرد–مثال ۴ وزیر



الگوریتم جستجوی عقب‌گرد

```
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 ← 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 ← INFERENCE(csp, var, value)
         if inferences ≠ failure then
            add inferences to assignment
            result ← BACKTRACK(assignment, csp)
            if result ≠ failure then
               return result
      remove {var = value} and inferences from assignment
   return failure
```

بهبود کارایی جستجوی عقب‌گرد

- جستجوی عقب‌گرد یک الگوریتم ناآگاهانه است و برای مسائل بزرگ کارآمد نیست.
- با پاسخ به سوالات زیر می‌توان به توابع هیوریستیک همه‌منظوره‌ای دست یافت که به جستجوی عقب‌گرد کمک می‌کنند.

۱- در مرحله‌ی بعد کدام متغیر باید مقداردهی شود؟ (SELECT-UNASSIGNED-VARIABLE)

۲- مقادیر متغیر انتخابی باید به چه ترتیبی امتحان شوند؟ (ORDER-DOMAIN-VALUES)

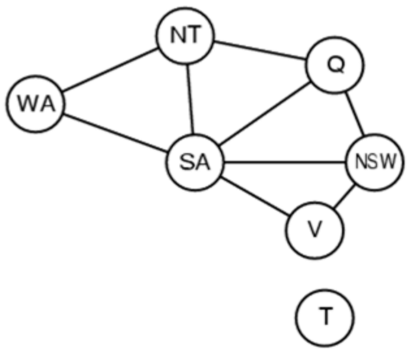
۳- چه استنتاج‌هایی باید در هر گام در جستجو انجام شود؟ (INFERENCE)

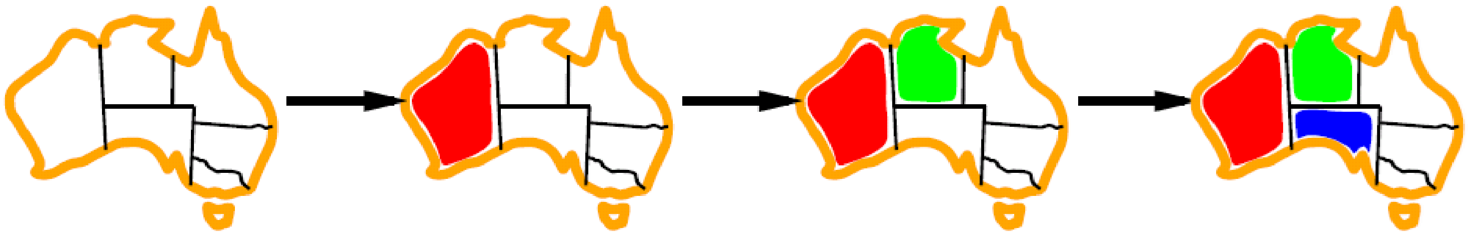
۴- آیا می توان شکست‌های حتمی را زودتر تشخیص داد؟

- فرض کنید در عقبگرد در مسأله ۸- وزیر، ۶ وزیر اول به گونه‌ای قرار گرفته‌اند که قرار دادن هشتمین وزیر را غیر ممکن می‌سازند. عقب‌گرد تمام مکان‌های ممکن برای وزیر هفتم را چک می‌کند، اگرچه مسأله غیر قابل حل است.

هیوریستیک MRV "متغیر با کم‌ترین مقدار باقی‌مانده"

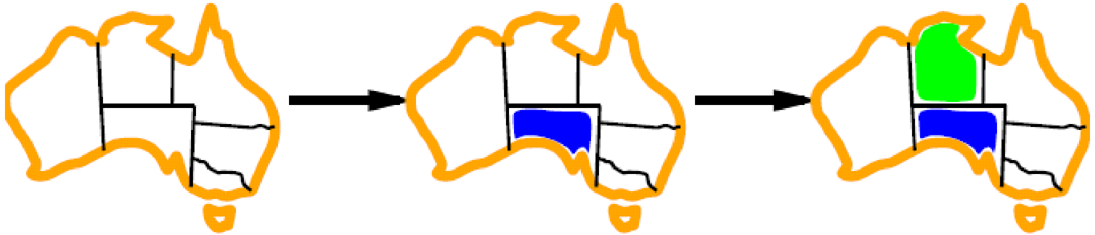
- هیوریستیک MRV (Minimum Remaining Value): انتخاب متغیری با کم‌ترین تعداد مقادیر مجاز
  - نام‌های دیگر: "متغیر با بیشترین محدودیت" یا "اول شکست"
  - هرس درخت جستجو: متغیری که بیشترین احتمال شکست در مسیر را دارد اول برمی‌گزیند و بدین وسیله درخت جستجو هرس می‌شود.
  - کشف سریع شکست: اگر در جریان کار متغیر x بدون مقدار مجاز باقی ماند MRV متغیر x را برمی‌گزیند و فورا شکست در جستجو تشخیص داده خواهد شد.



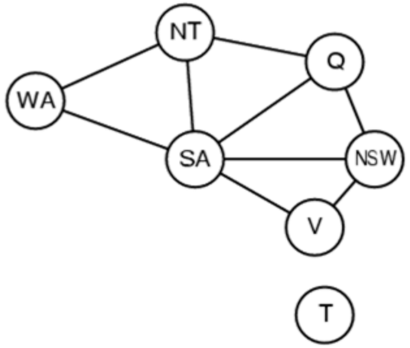


هیوریستیک درجه (degree)

- اولین ناحیه برای رنگ‌آمیزی نقشه کدام باشد بهتر است؟ آیا هیوریستیک MRV در انتخاب آن به ما کمک می‌کند؟
- هیوریستیک درجه: انتخاب متغیری که در تعداد بیشتری از محدودیت‌های متغیرهای بدون انتساب نقش دارد
  - کاربرد: گره‌گشایی برای MRV زمانی‌که تعداد مقادیر مجاز برای چندین متغیر یکسان باشد
  - منجر به کاهش ضریب انشعاب در متغیرهای بعدی می‌شود.

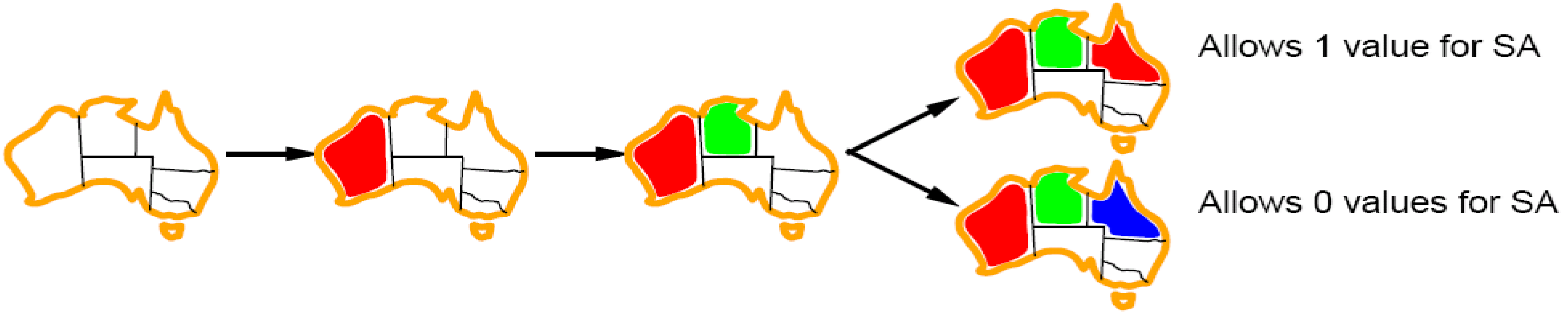


$d(WA) = 2$	$d(WA) = 1$	$d(WA) = 0$
$d(NT) = 3$	$d(NT) = 2$	$d(Q) = 1$
$d(Q) = 3$	$d(Q) = 2$	$d(NSW) = 2$
$d(NSW) = 3$	$d(NSW) = 2$	$d(V) = 1$
$d(V) = 2$	$d(V) = 1$	$d(T) = 0$
$d(SA) = 5$	$d(T) = 0$	
$d(T) = 0$		



هیوریستیک LCV "مقدار با حداقل محدودیت"

- هیوریستیک LCV (Least Constraining Value): مقداری را انتخاب می‌کند که کم‌ترین محدودیت را روی مقادیر معتبر متغیرهای بدون انتساب ایجاد کند.
  - این هیوریستیک سعی می‌کند بیشترین قابلیت انعطاف را برای انتساب‌های بعدی متغیرها فراهم آورد. به این ترتیب امکان رسیدن به یک انتساب نامعتبر را کم می‌کند و درنتیجه احتمال عقب‌گرد کردن را کم‌تر می‌کند و درنتیجه سریع‌تر به جواب می‌رسیم.
  - مناسب برای زمانی که فقط به دنبال یک راه‌حل هستیم نه تمام راه‌حل‌ها

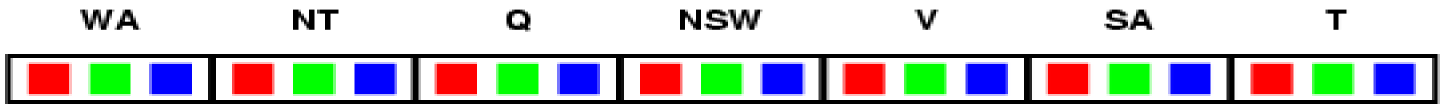


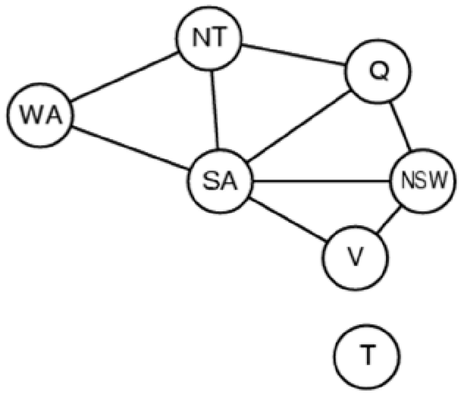
جایگاه استنتاج(inference) در مسائل CSP

- استنتاج به‌عنوان یک مرحله‌ی پیش پردازش
  - الگوریتم سازگاری یال AC-3
    - حذف مقادیر از دامنه‌های تمام متغیرها تا آن‌جا که تمام زوج متغیرها سازگار کمان شوند.
- استنتاج در خلال جستجو
  - وارسی روبه‌جلو (Forward checking)
    - هنگام انتخاب یک مقدار برای یک متغیر، کاهش‌های جدید دامنه بر روی متغیرهای بدون انتساب همسایه را استنتاج می‌کند.
  - حفظ سازگاری یال (Maintaining Arc Consistency-MAC)– انتشار محدودیت
    - وارسی روبه‌جلو + انتشار محدودیت‌ها به‌طور بازگشتی هنگامی که تغییرات بر روی دامنه‌ها بوجود می‌آید.

وارسی روبه‌جلو

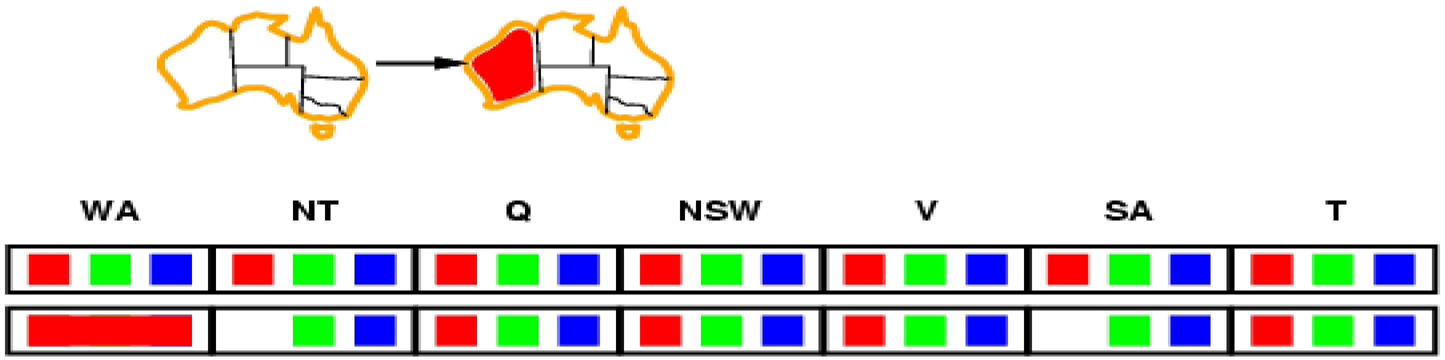
- هرگاه یک مقدار به متغیر X انتساب داده شد وارسی پیش‌رو سازگاری یال را بر روی آن اجرا می‌کند.
  - برای هر متغیر انتساب‌نیافته‌ی Y که توسط یک محدودیت با X در ارتباط است، هر مقدار ناسازگار با مقدار انتخاب‌شده برای X را از دامنه‌ی Y حذف می‌کند.
  - اگر برای یک متغیر هیچ مقدار مجازی باقی نماند، جستجو پایان می‌یابد.

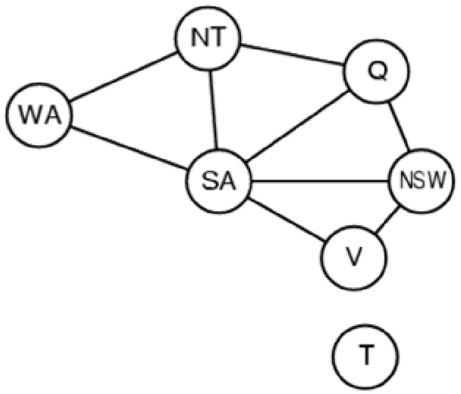




وارسی روبه‌جلو

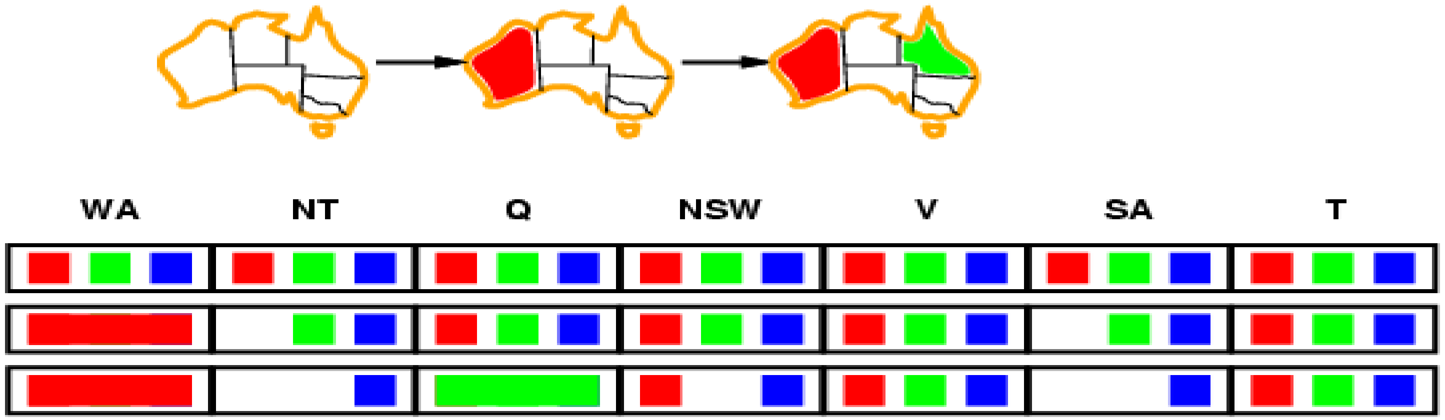
- هرگاه یک مقدار به متغیر X انتساب داده شد وارسی پیش‌رو سازگاری یال را بر روی آن اجرا می‌کند.
  - برای هر متغیر انتساب‌نیافته‌ی Y که توسط یک محدودیت با X در ارتباط است، هر مقدار ناسازگار با مقدار انتخاب‌شده برای X را از دامنه‌ی Y حذف می‌کند.
  - اگر برای یک متغیر هیچ مقدار مجازی باقی نماند، جستجو پایان می‌یابد.

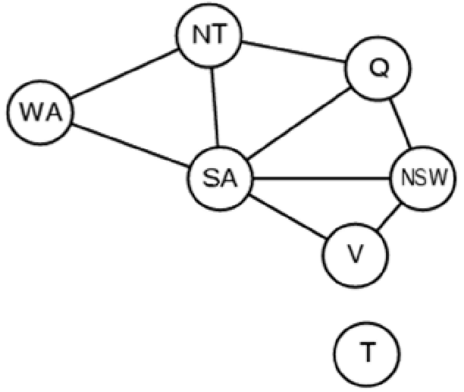




وارسی روبه‌جلو

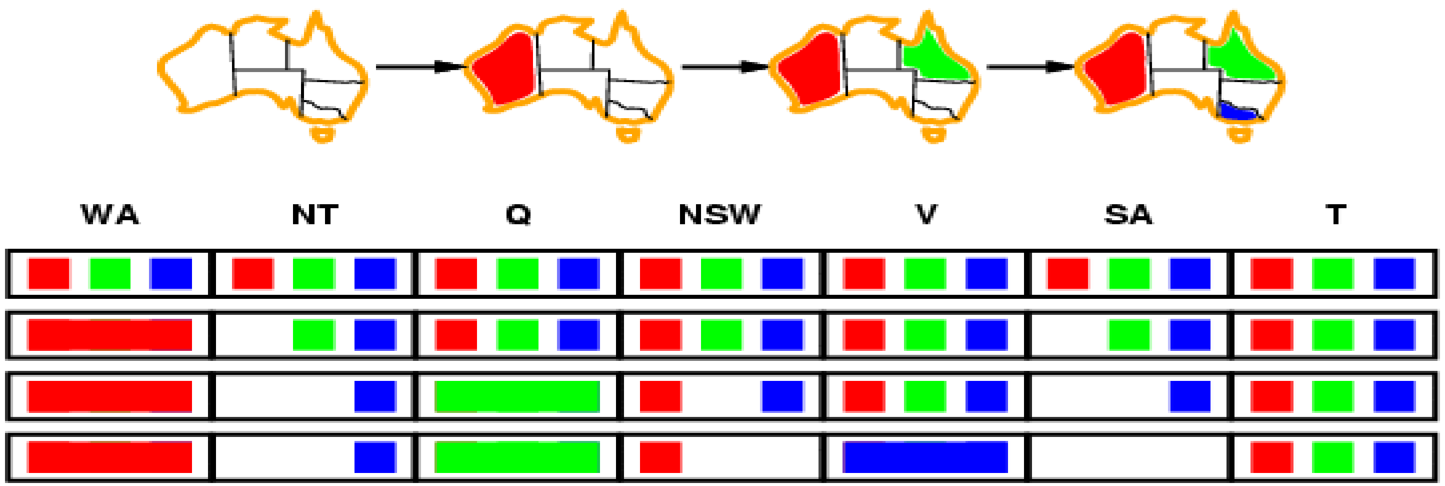
- هرگاه یک مقدار به متغیر X انتساب داده شد وارسی پیش‌رو سازگاری یال را بر روی آن اجرا می‌کند.
  - برای هر متغیر انتساب‌نیافته‌ی Y که توسط یک محدودیت با X در ارتباط است، هر مقدار ناسازگار با مقدار انتخاب‌شده برای X را از دامنه‌ی Y حذف می‌کند.
  - اگر برای یک متغیر هیچ مقدار مجازی باقی نماند، جستجو پایان می‌یابد.

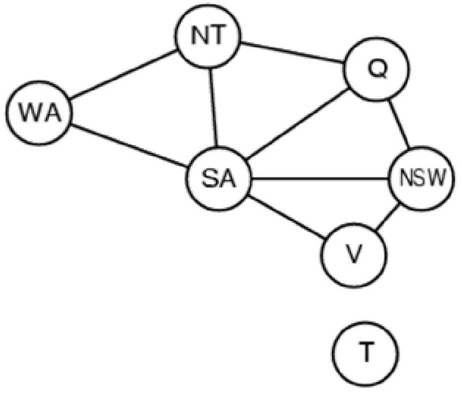




وارسی روبه‌جلو

- هرگاه یک مقدار به متغیر X انتساب داده شد وارسی پیش‌رو سازگاری یال را بر روی آن اجرا می‌کند.
  - برای هر متغیر انتساب‌نیافته‌ی Y که توسط یک محدودیت با X در ارتباط است، هر مقدار ناسازگار با مقدار انتخاب‌شده برای X را از دامنه‌ی Y حذف می‌کند.
  - اگر برای یک متغیر هیچ مقدار مجازی باقی نماند، جستجو پایان می‌یابد.

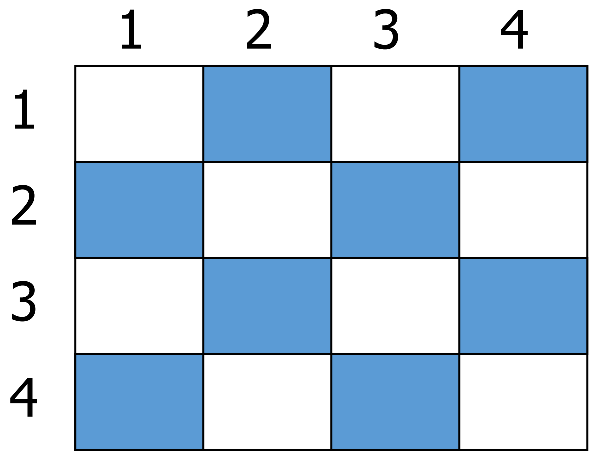


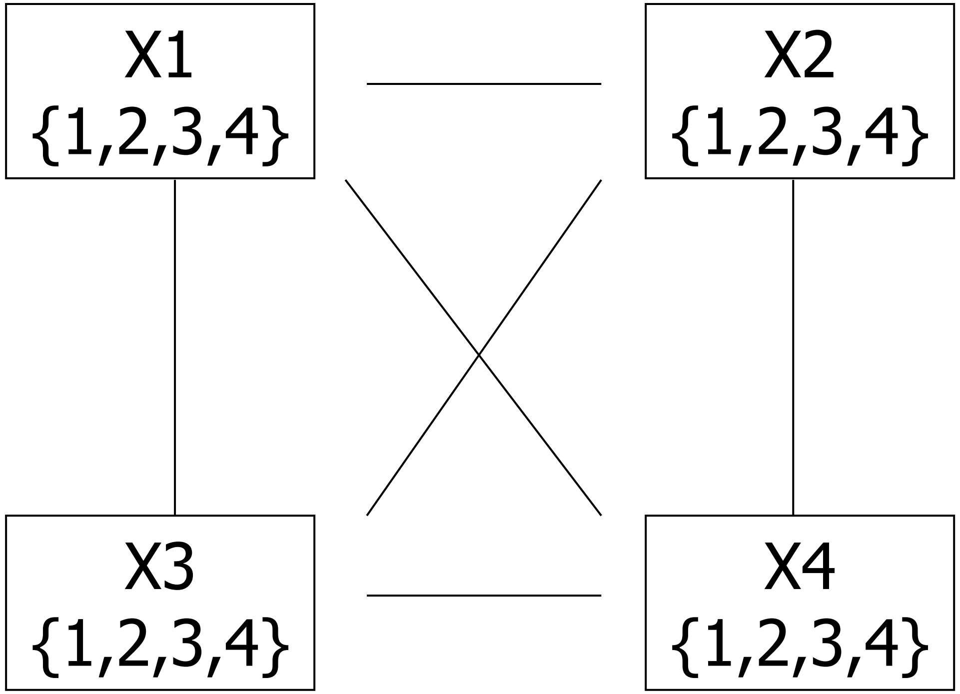


وارسی روبه‌جلو ...

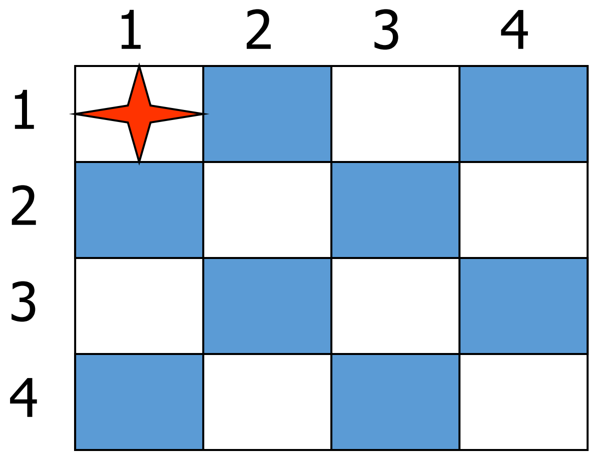
- وارسی روبه‌جلو راجع به انتخاب متغیر بعدی برای مقداردهی یا انتخاب مقدار برای متغیر نظری نمی‌دهد، فقط می‌تواند وقوع تضاد در آینده را تشخیص دهد.
  - در واقع بعد از هر مقداردهی به یک متغیر، وارسی روبه‌جلو را انجام می‌دهیم تا بررسی کنیم مقداردهی اخیر منجر به بن‌بست در آینده می‌شود یا خیر.
- برای بسیاری مسائل ترکیب MRV و وارسی روبه‌جلو کارآمدتر خواهد بود.
  - وارسی روبه‌جلو یک راه کارآمد برای محاسبه افزایشی اطلاعاتی است که MRV نیاز دارد.

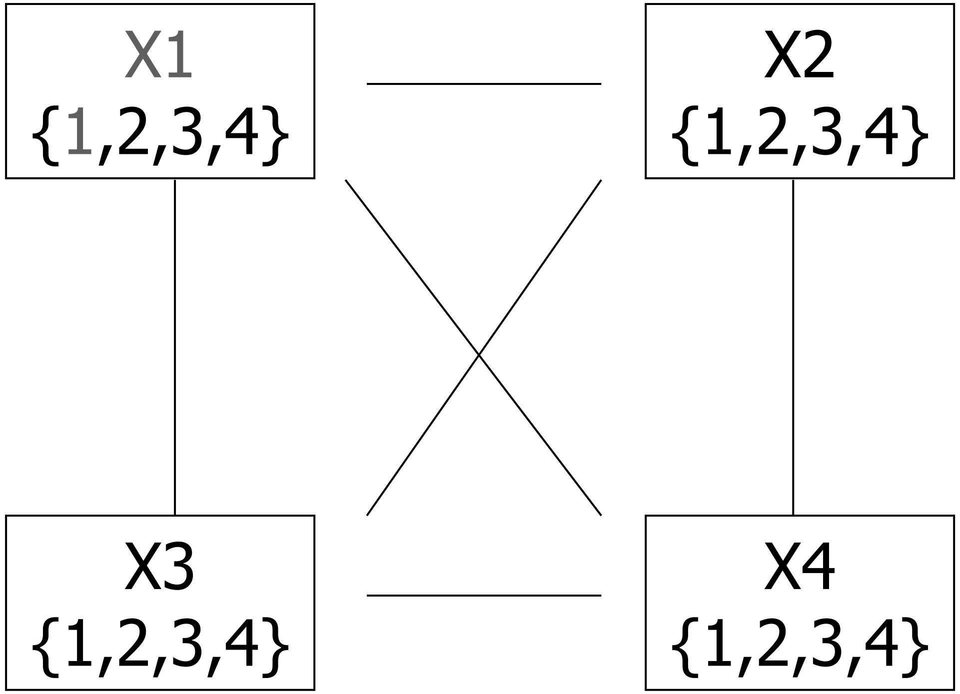
مسئله ۴ وزیر (FC+MRV)



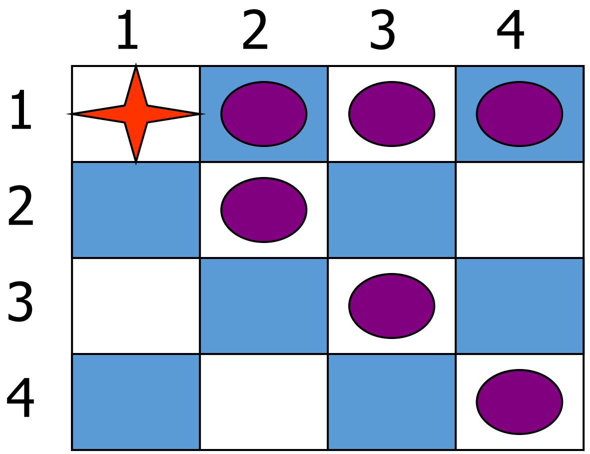


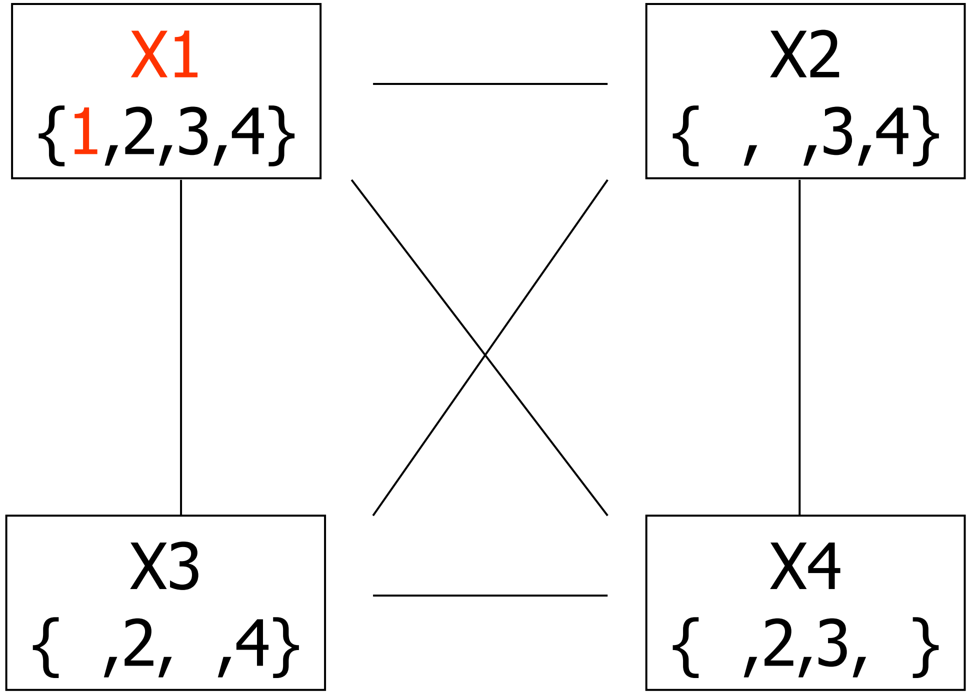
مسئله ۴ وزیر (FC+MRV)



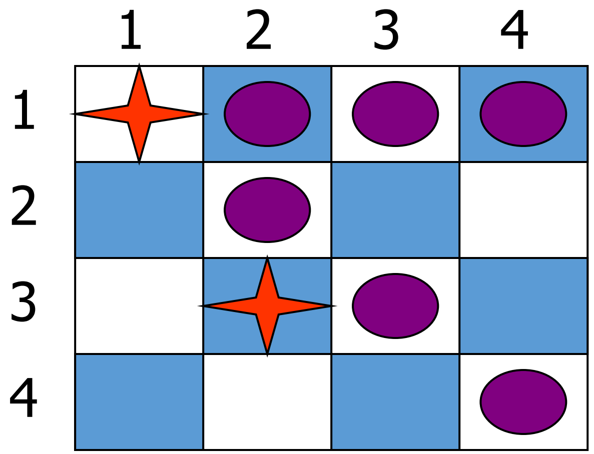


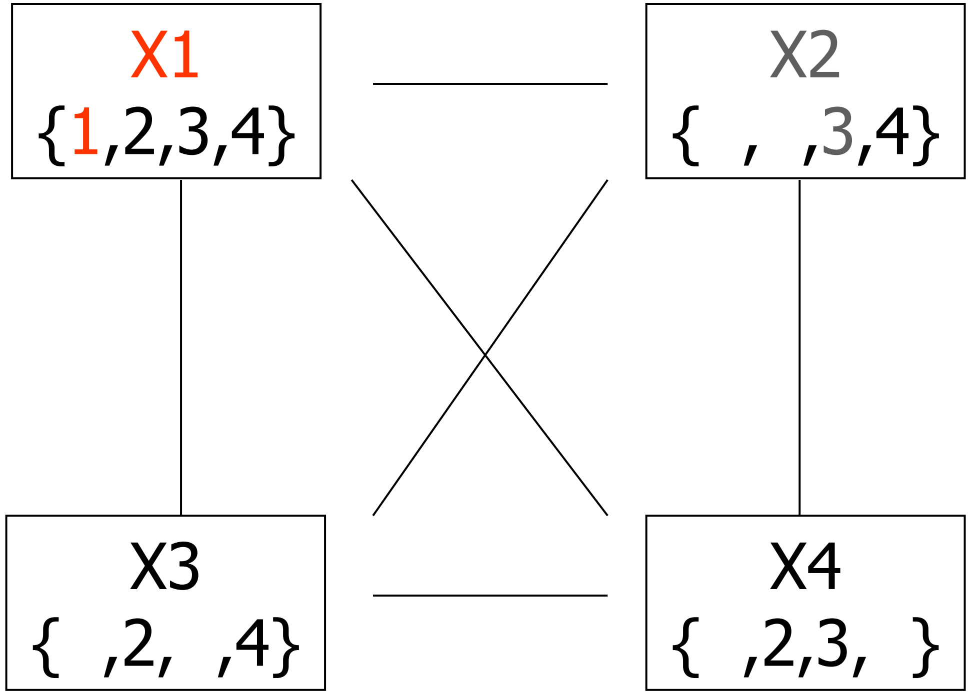
مسئله ۴ وزیر (FC+MRV)



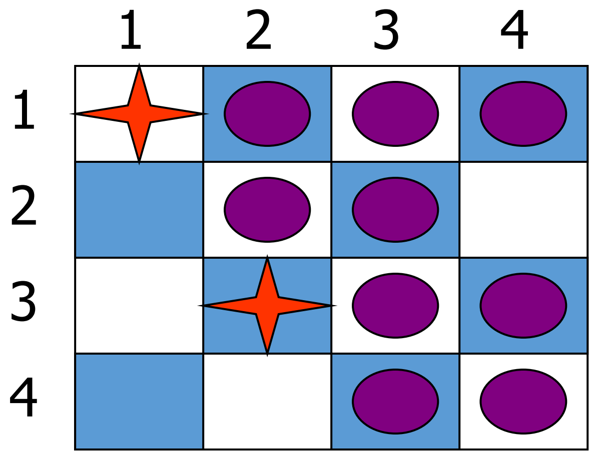


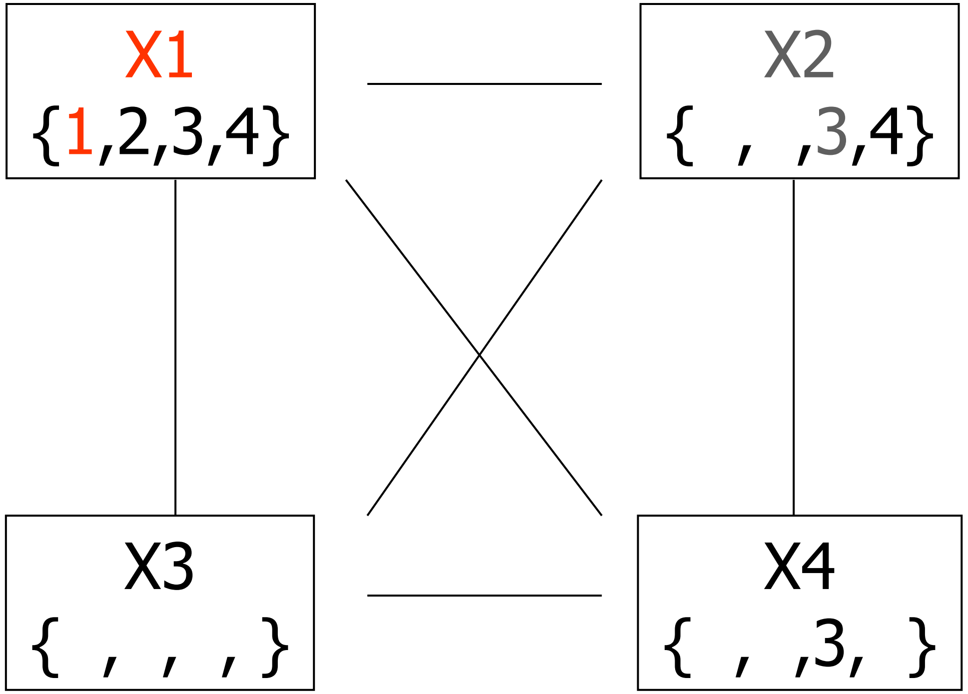
مسئله ۴ وزیر (FC+MRV)



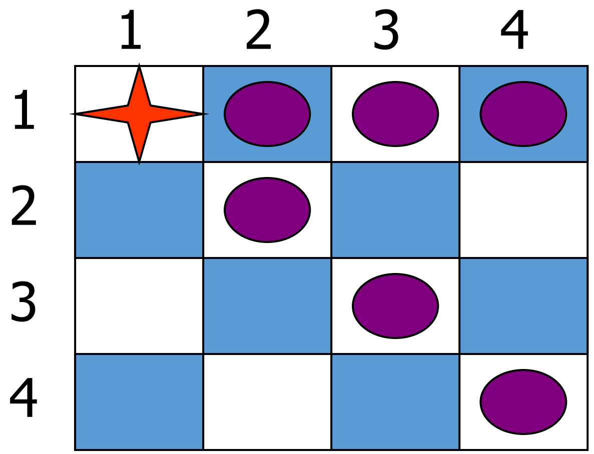


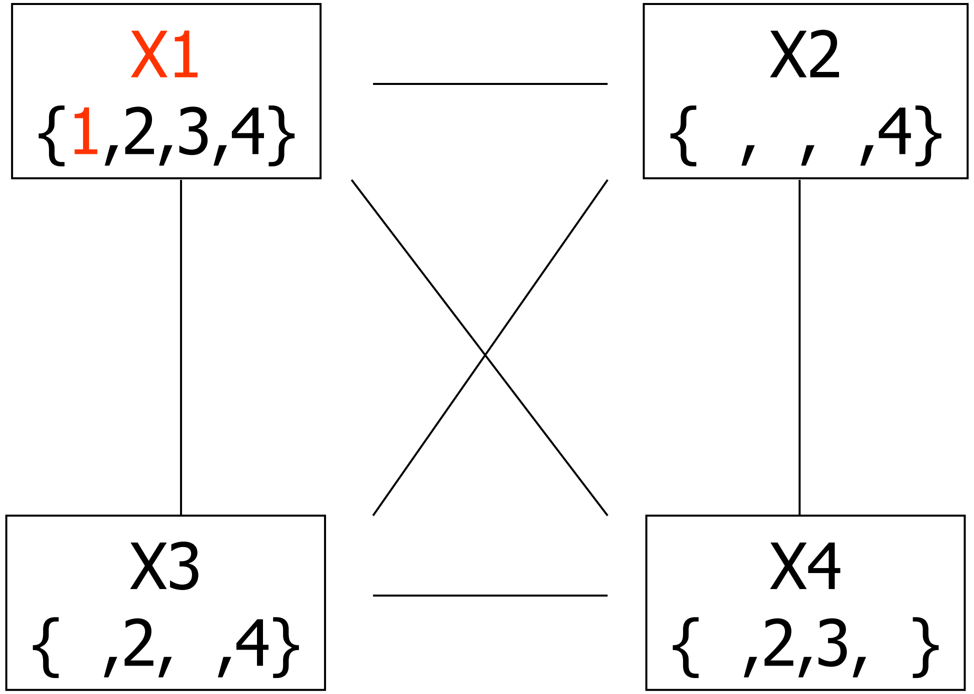
مسئله ۴ وزیر (FC+MRV)



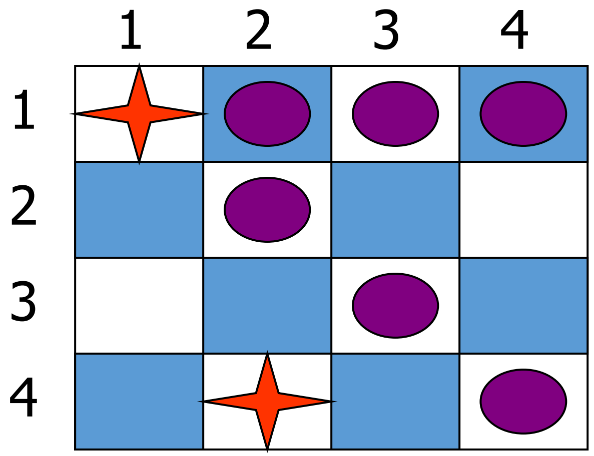


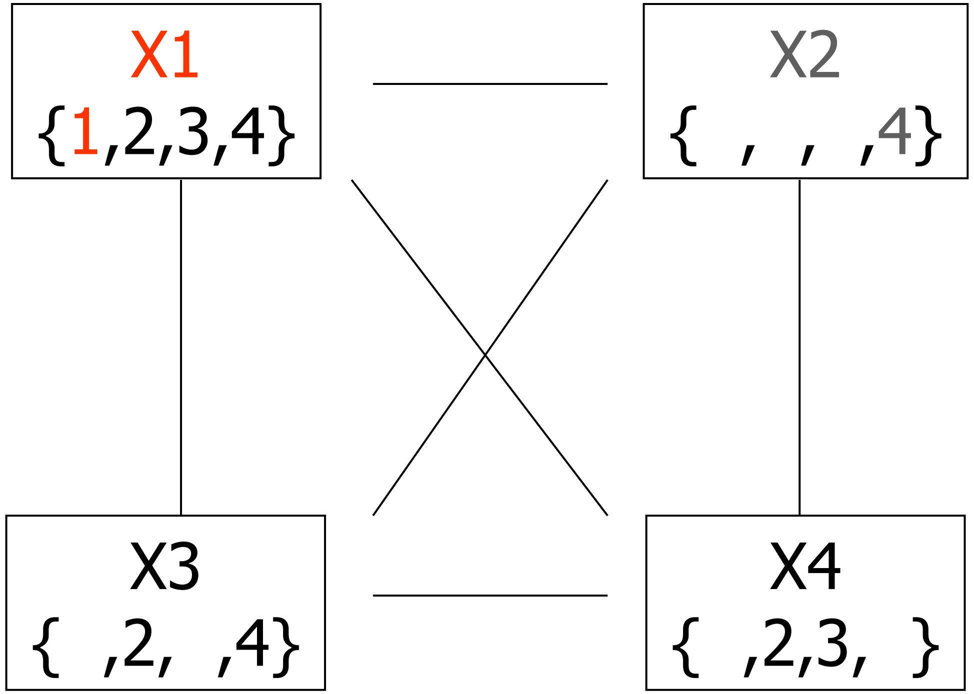
مسئله ۴ وزیر (FC+MRV)



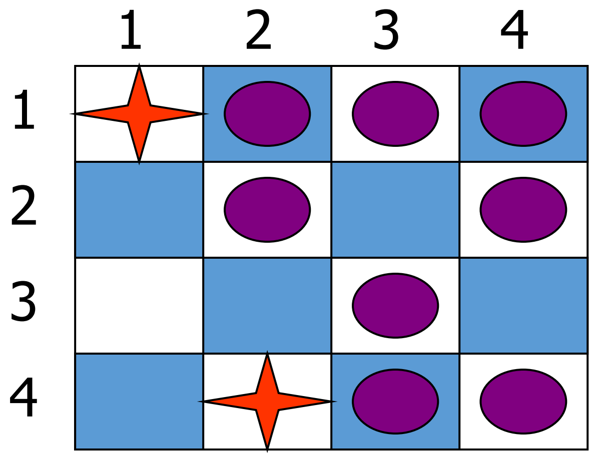


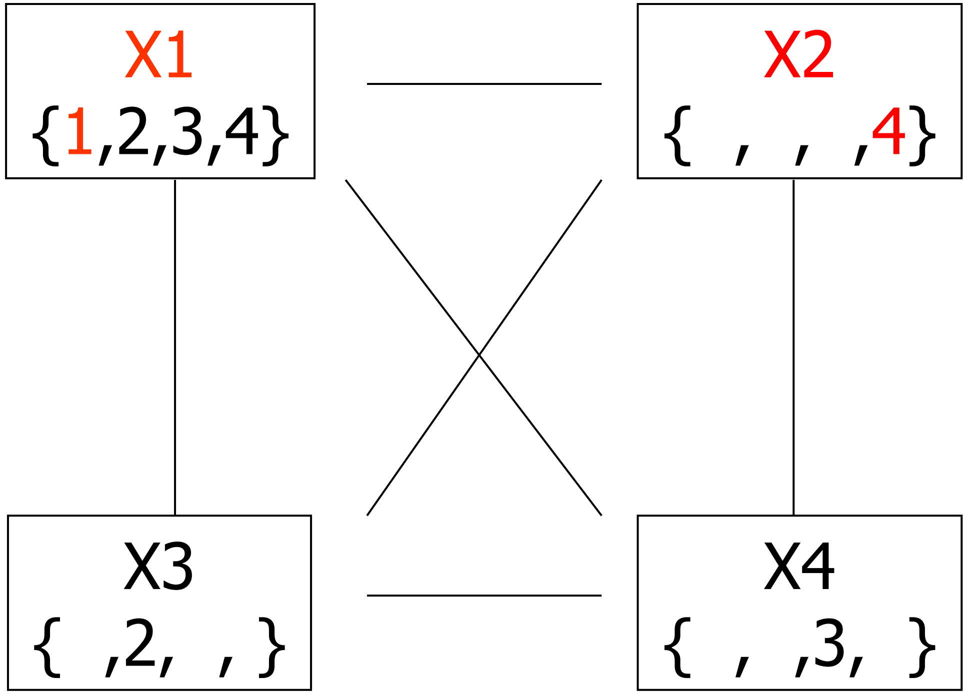
مسئله ۴ وزیر (FC+MRV)



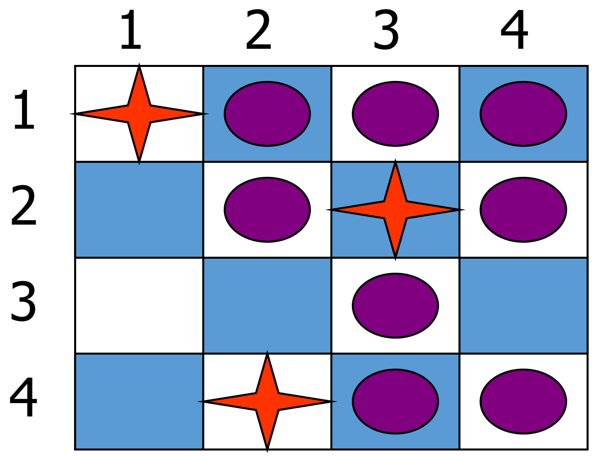


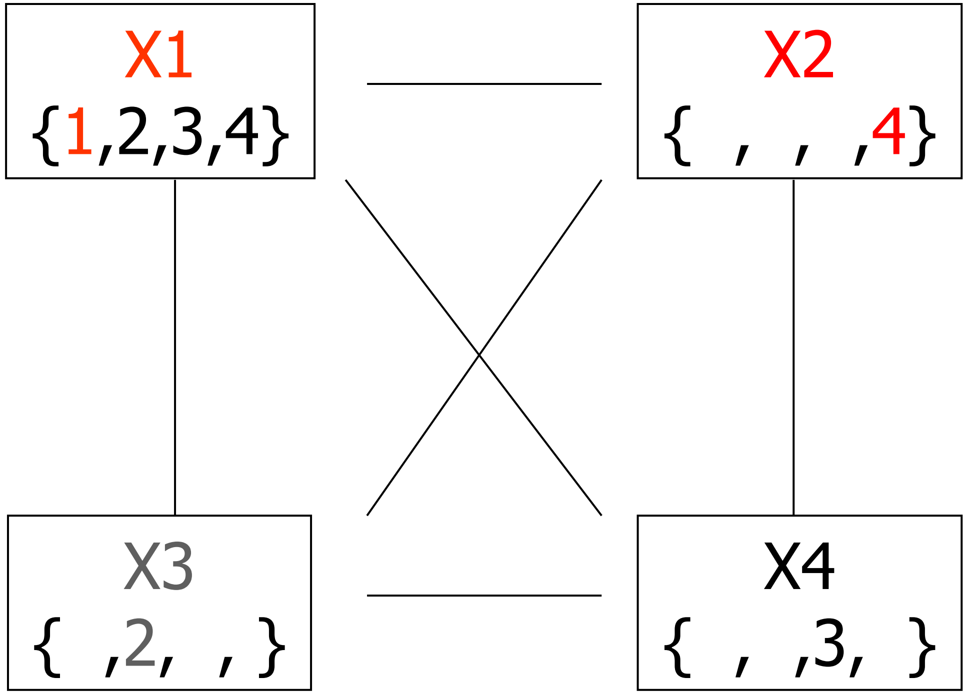
مسئله ۴ وزیر (FC+MRV)



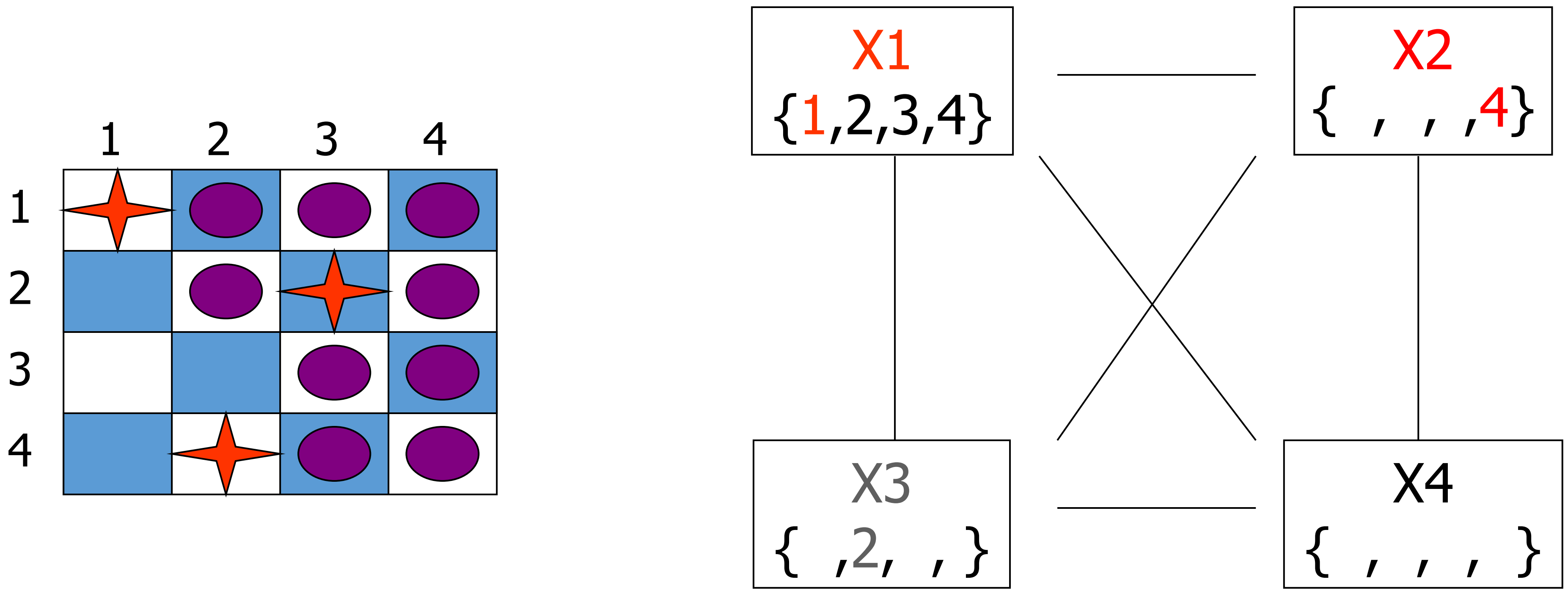


مسئله ۴ وزیر (FC+MRV)

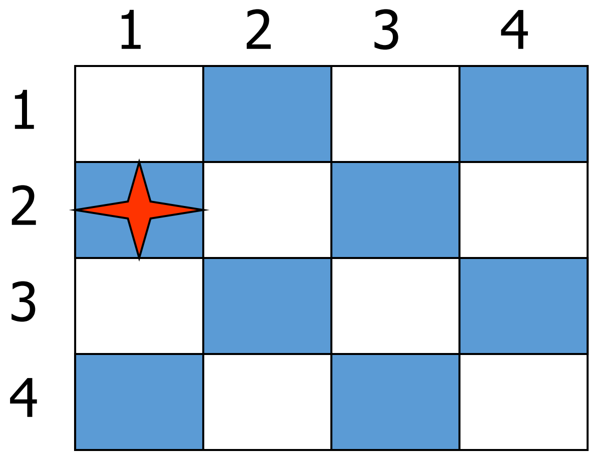


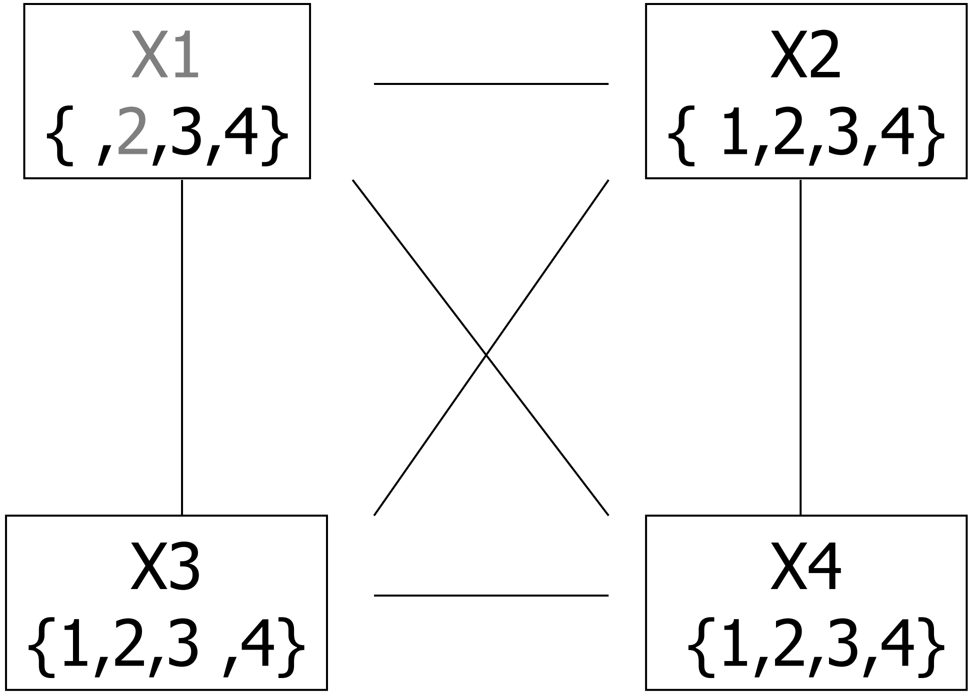


مسئله ۴ وزیر (FC+MRV)



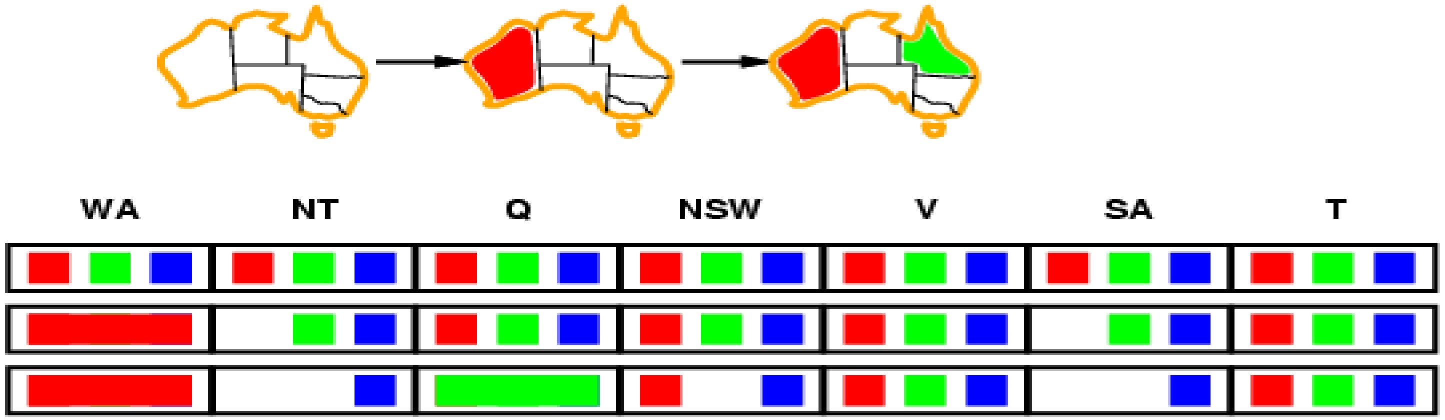
مسئله ۴ وزیر (FC+MRV)

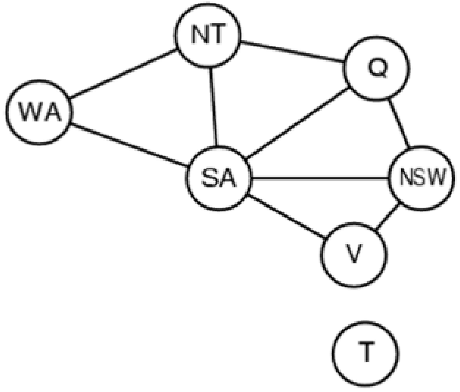




نیاز به انتشار محدودیت بیشتر

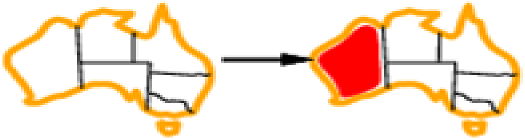
- وارسی رو به‌جلو محدودیت‌ها را از متغیرهای انتساب‌یافته به متغیرهای انتساب‌نیافته منتشر می‌کند، اما تمام شکست‌ها را نمی‌تواند در زودترین زمان ممکن تشخیص دهد.
  - در مثال زیر NT و SA هر دو نمی توانند آبی باشند!
  - وارسی روبه‌جلو به اندازه کافی به جلو نگاه نمی‌کند و متغیرهای دیگر را سازگار کمان نمی‌کند.

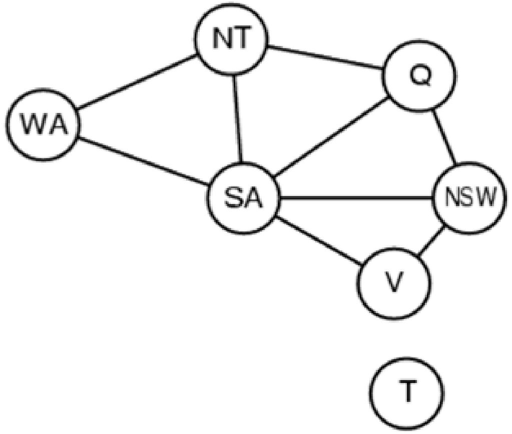




حفظ سازگاری یال (MAC)

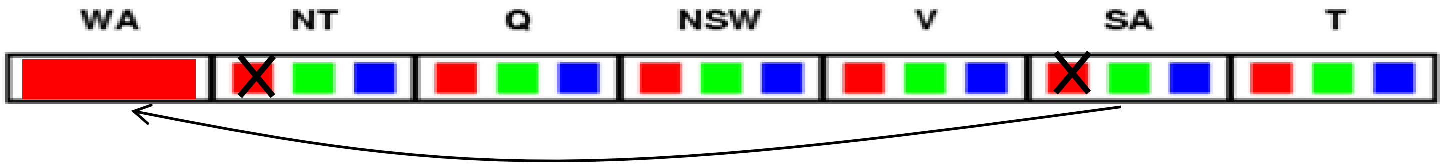
- پس از آن‌که یک مقدار به متغیر X_i نسبت داده شد، AC-3 فراخوانی می‌شود؛ اما ورودی به‌جای یک صف از همه‌ی یال‌ها در CSP، یال‌های (X_j,X_i) برای همه‌ی X_jهایی است که متغیرهای بدون انتساب بوده و همسایه‌ی X_i هستند.





{(NT, WA) ,(SA, WA)}

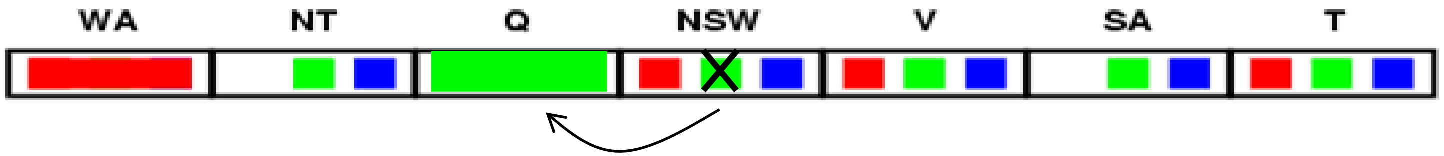
{(SA, WA), (Q, NT), (SA, NT)}



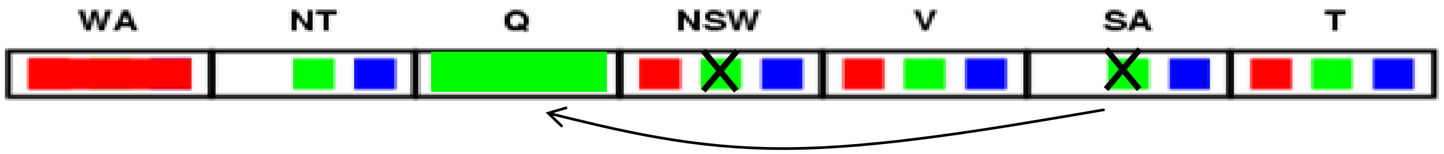
{(Q, NT), (SA, NT), (V, SA), (NSW, SA), (Q, SA), (NT, SA)}

حفظ سازگاری یال (MAC)

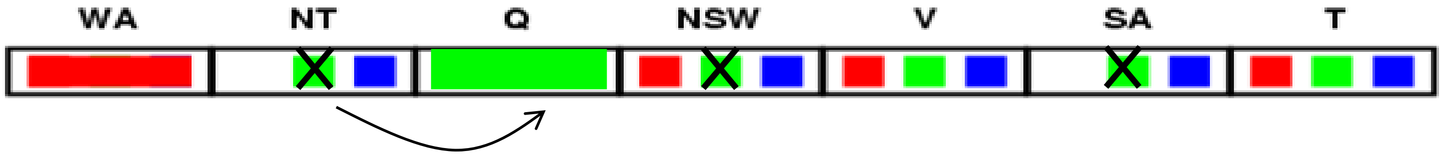
{(NSW, Q) ,(SA, Q), (NT, Q)}



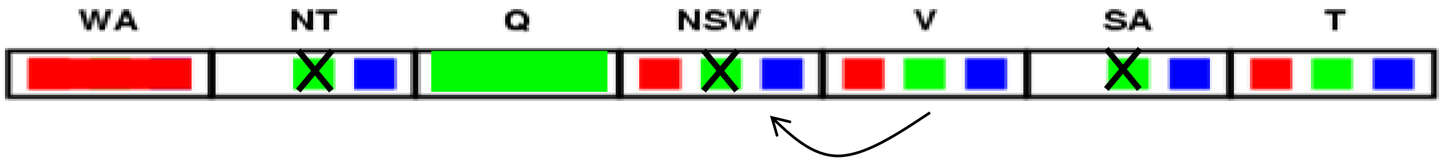
{(SA, Q), (NT, Q), (V, NSW), (SA, NSW)}

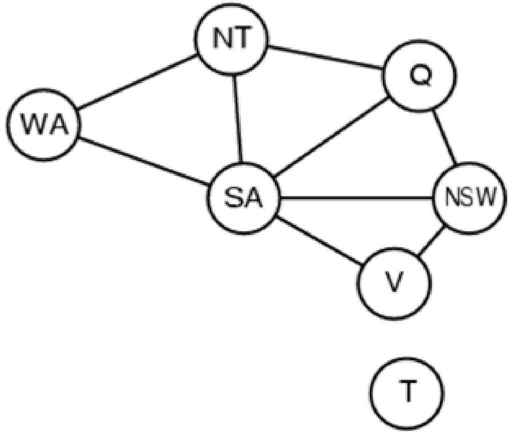


{(NT, Q), (V, NSW), (SA, NSW), (NSW, SA), (NT, SA), (V, SA)}

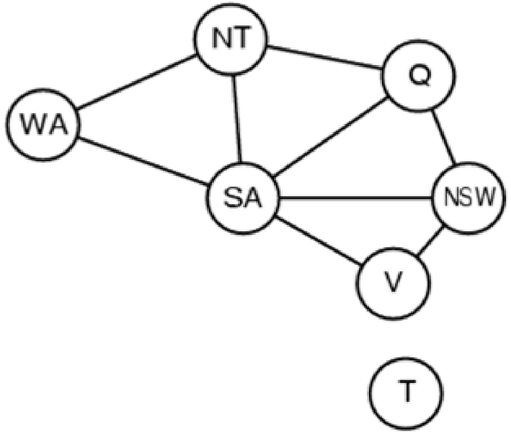


{(V, NSW), (SA, NSW), (NSW, SA), (NT, SA), (V, SA), (SA, NT)}

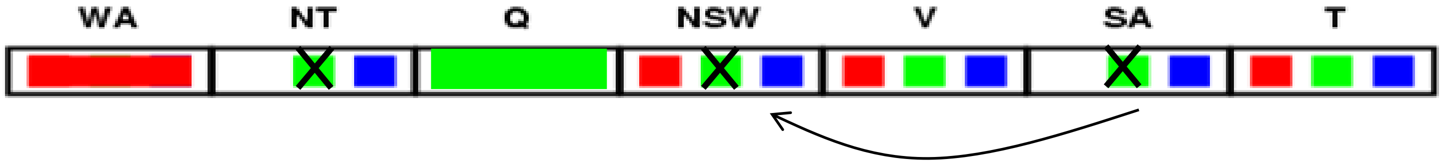




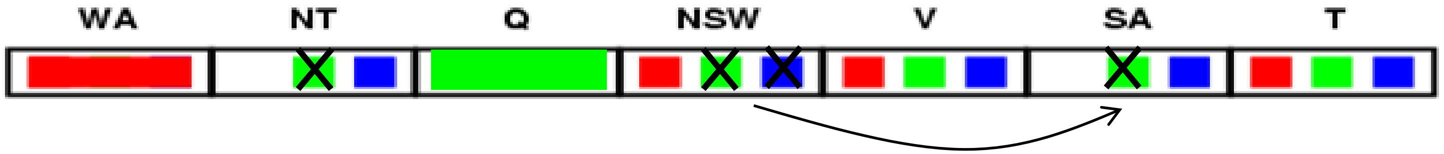
حفظ سازگاری یال (MAC)



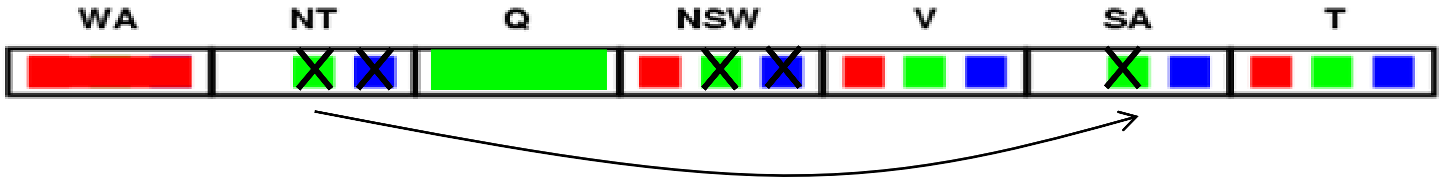
{(SA, NSW), (NSW, SA), (NT, SA), (V, SA), (SA, NT)}



{(NSW, SA), (NT, SA), (V, SA), (SA, NT)}



{(NT, SA), (V, SA), (SA, NT), (V, NSW)}



تست ۳

در حل مسئله ارضای محدودیت زیر (مسئله چهار وزیر)، وزیر شماره یک در خانه شماره ۲ قرار داده شده و خانه‌هایی که با علامت × مشخص شده‌اند، توسط الگوریتم Forward checking حذف شده است. در این مرحله می‌خواهیم الگوریتم Arc consistency را روی این مسئله اعمال کنیم. کدام مقدار و از دامنه کدام وزیر زودتر از سایر مقادیر حذف می‌شود؟ (Q_x,n) یعنی مقدار n از دامنه Q_x حذف می‌شود. (مهندسی کامپیوتر دولتی ۹۵)

	1	2	3	4
Q_1		✓		
Q_2	×	×	×	
Q_3		×		×
Q_4		×		

۱) $(Q_3,3)$ یا $(Q_3,1)$

۲) $(Q_3,3)$ یا $(Q_4,1)$

۳) $(Q_4,4)$ یا $(Q_4,1)$

۴) $(Q_4,4)$ یا $(Q_3,3)$ ✓

عقب‌گرد ساده

- در الگوریتم BACKTRACKING-SEARCH هنگامی که یکی از شاخه‌ها با شکست در جستجو مواجه می‌شود، الگوریتم به <u>متغیر قبلی</u> باز می‌گردد و مقداری جدید برای آن در نظر می‌گیرد.
  - مثال: فرض کنید ترتیب رنگ ناحیه‌ها به‌صورت زیر باشد

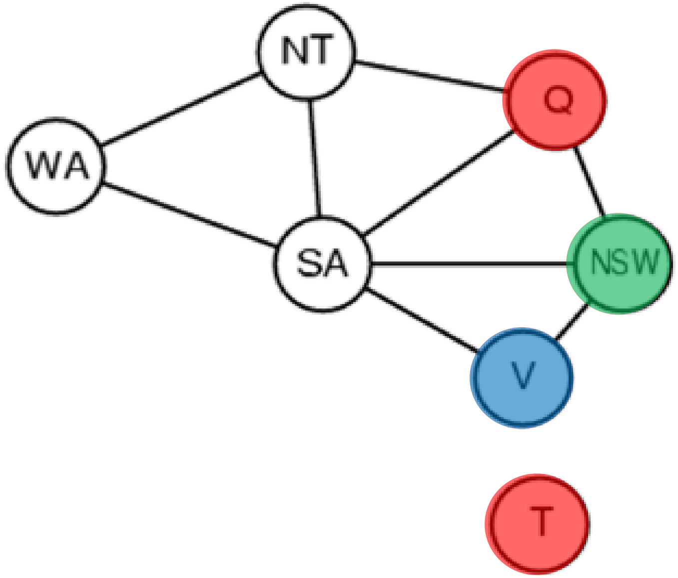
Q=red, NSW=green, V=blue, T=red

و فرض کنید متغیر بعدی برای مقداردهی SA باشد.

چه مقداری را می‌توان برای SA برگزید؟ هیچ مقداری موجود نیست.

متغیری که به آن برگشت انجام می‌شود کدام است؟ T

آیا انتخاب این متغیر برای برگشت کمکی می‌کند؟ خیر



پرش رو به عقب

- یک روش پس‌گرد هوشمندانه‌تر آن است که تمام مسیر را تا رسیدن به مجموعه‌ای از متغیرها که باعث شکست شده‌اند (مجموعه تناقض)، به عقب باز گردیم.
  - مجموعه تناقض (Conflict set) برای متغیر X عبارت است از مجموعه‌ای از متغیرهایی که قبلاً مقداردهی شده‌اند و به واسطهٔ یک محدودیت با X در ارتباطند.

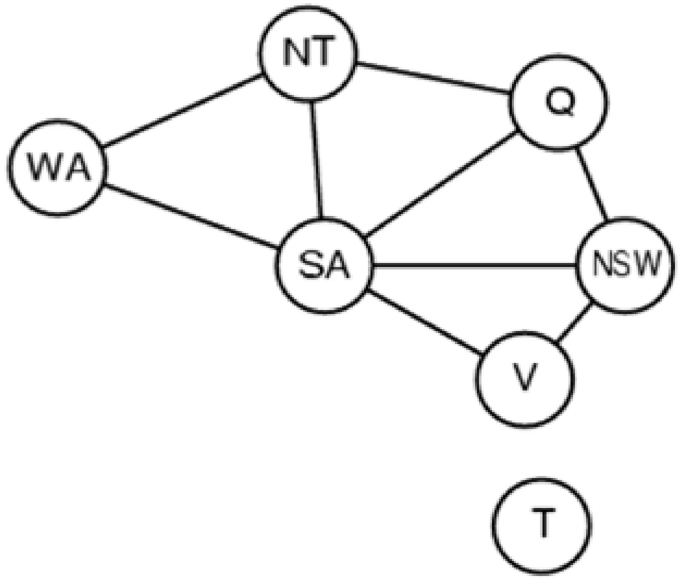
- روش پرش رو به عقب (Back jumping)
  - قبل از مقداردهی به یک متغیر: ”مجموعه تناقض“ را برای آن متغیر محاسبه و ذخیره کن
  - اگر تمام مقادیر متغیر فعلی X با انتساب‌های قبلی ناسازگار باشد: مسیر پیموده‌شده را تا رسیدن به آخرین متغیری که در مجموعهٔ تناقض مقداردهی شده است، به عقب برو و مقدار دیگری از آن متغیر را بررسی کن.

پرش رو به عقب ...

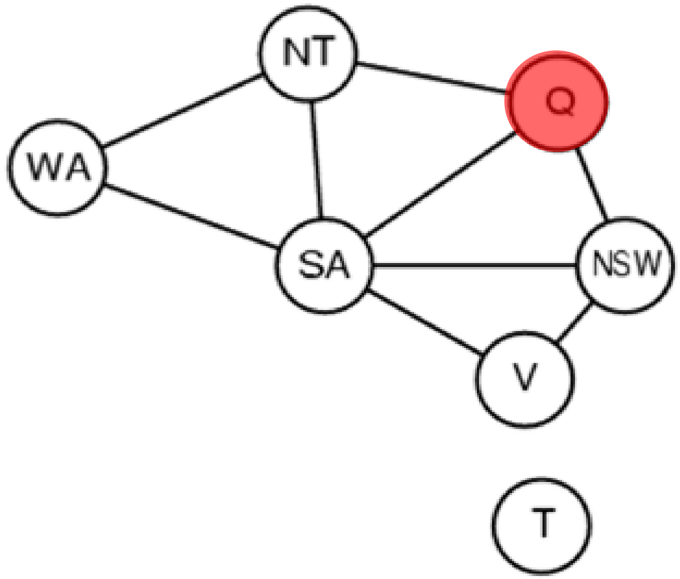
- مثال: ترتیب رنگ ناحیه‌ها را مانند مثال قبل فرض کنید

Q=red, NSW=green, V=blue, T=red

مجموعه تناقض را قبل از مقداردهی هر متغیر ذکر می‌کنیم



پرش رو به عقب ...



- مثال: ترتیب رنگ ناحیه‌ها را مانند مثال قبل فرض کنید

Q=red, NSW=green, V=blue, T=red

مجموعه تناقض را قبل از مقداردهی هر متغیر ذکر می‌کنیم

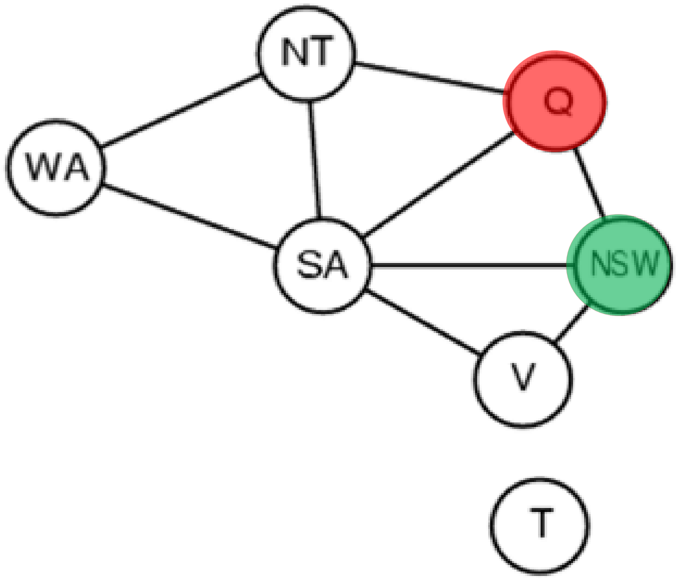
Q=red Conf(Q)={}

پرش رو به عقب ...

- مثال: ترتیب رنگ ناحیه‌ها را مانند مثال قبل فرض کنید

Q=red, NSW=green, V=blue, T=red

مجموعه تناقض را قبل از مقداردهی هر متغیر ذکر می‌کنیم



Q=red → NSW=green

Conf(Q)={}

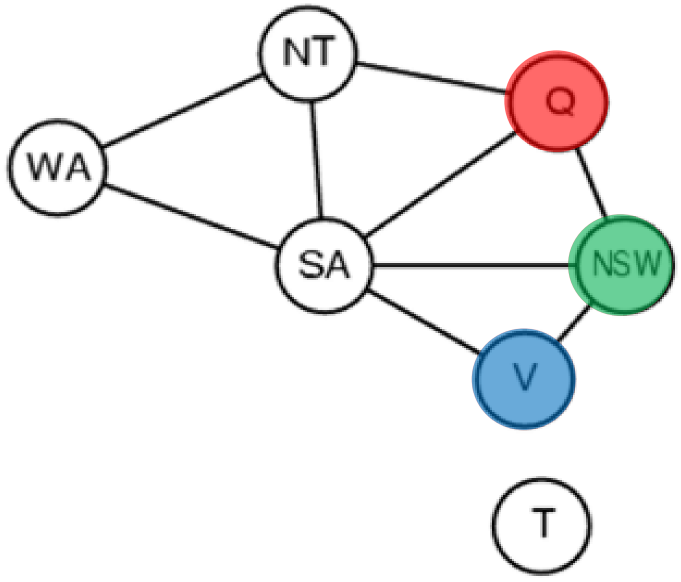
Conf(NSW)={Q}

پرش رو به عقب ...

- مثال: ترتیب رنگ ناحیه‌ها را مانند مثال قبل فرض کنید

Q=red, NSW=green, V=blue, T=red

مجموعه تناقض را قبل از مقداردهی هر متغیر ذکر می‌کنیم



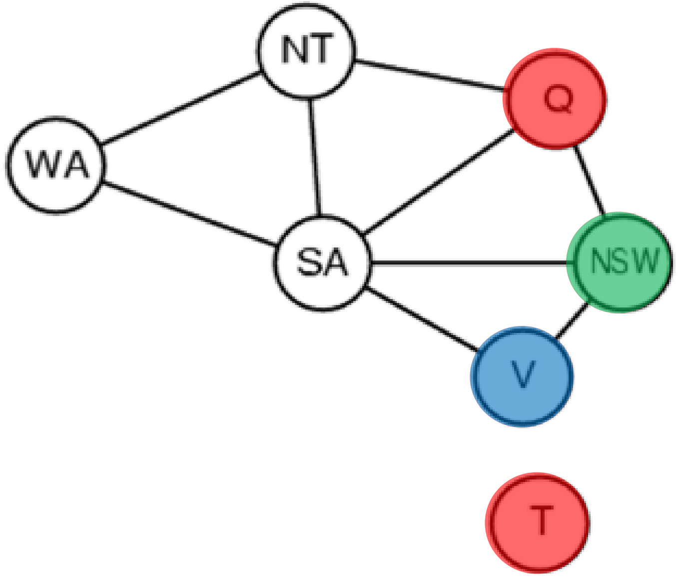
Q=red → NSW=green → V=blue

Conf(Q)={}

Conf(NSW)={Q}

Conf(V)={NSW}

پرش رو به عقب ...



- مثال: ترتیب رنگ ناحیه‌ها را مانند مثال قبل فرض کنید

Q=red, NSW=green, V=blue, T=red

مجموعه تناقض را قبل از مقداردهی هر متغیر ذکر می‌کنیم

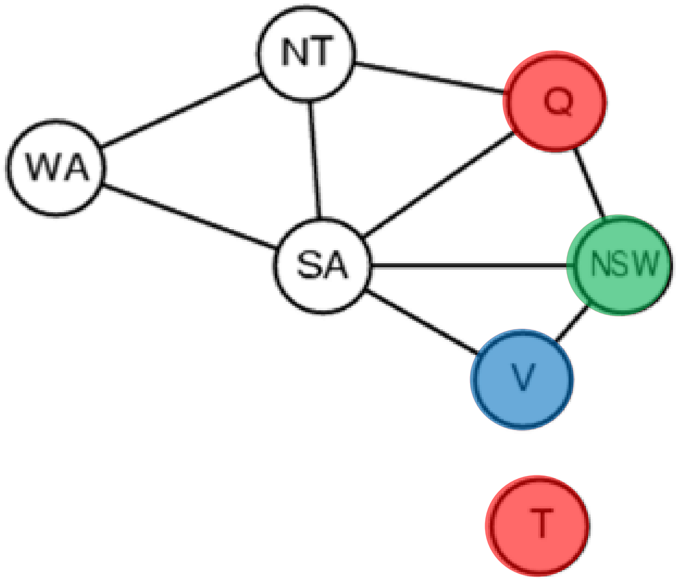
Q=red	Conf(Q)={}
NSW=green	Conf(NSW)={Q}
V=blue	Conf(V)={NSW}
T=red	Conf(T)={}

پرش رو به عقب ...

- مثال: ترتیب رنگ ناحیه‌ها را مانند مثال قبل فرض کنید

Q=red, NSW=green, V=blue, T=red

مجموعه تناقض را قبل از مقداردهی هر متغیر ذکر می‌کنیم



Q=red → Conf(Q)={}

NSW=green → Conf(NSW)={Q}

V=blue → Conf(V)={NSW}

T=red → Conf(T)={}

SA=? Conf(SA)={Q,NSW,V}

مقدار سازگاری برای SA وجود ندارد پس به متغیر V برمی‌گردیم.

پرش رو به عقب ...

- پرش رو به عقب هنگامی رخ می‌دهد که هر مقدار متعلق به دامنه با انتساب انجام شده دارای تناقض باشد. وارسی رو به جلو این حالت را از قبل تشخیص می‌دهد و از رسیدن به آن جلوگیری می‌کند.
  - هر شاخه‌ای که به وسیلهٔ پرش رو به عقب هرس می‌شود، می‌تواند به وسیله انجام وارسی رو به جلو نیز هرس شود.
- پرش رو به عقب ساده، در جستجوهایی که از روش‌های قوی‌تر بررسی سازگاری نظیر MAC استفاده می‌کنند زائد است.

تست ۴

فرض کنید در حل مسئله ارضای محدودیت به کمک روش پرش به عقب برای متغیر kام هیچ مقدار مناسبی باقی نمانده باشد و الگوریتم به متغیر jام عقب‌گرد کند و مقدار جدیدی برای متغیر jام تعیین شود. مقادیر انتخاب‌شده برای متغیرهای بعد از j تا k چه خواهد شد؟

(مهندسی کامپیوتر دولتی ۹۳)

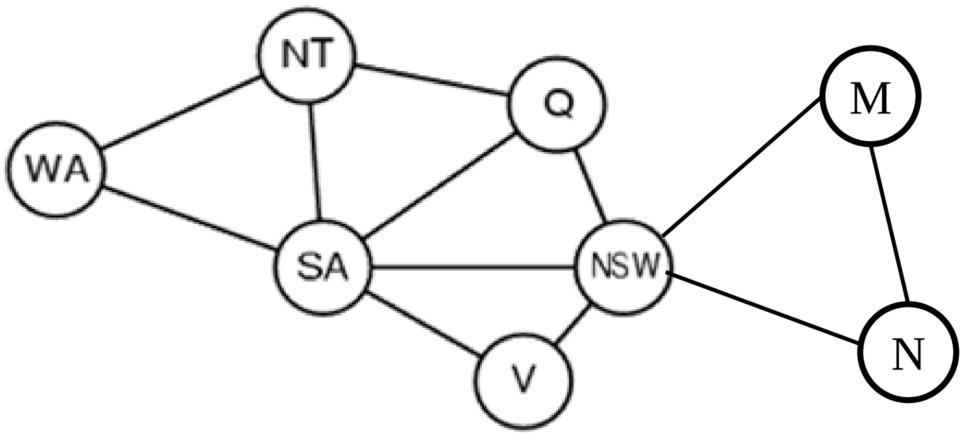
۱) فقط مقدار متغیرهای وابسته به j مجددا تعیین می‌شوند.

۲) فقط مقدار متغیرهای وابسته به k مجددا تعیین می‌شوند.

۳) مقادیر همه متغیرهای بین j و k حفظ می‌شوند.

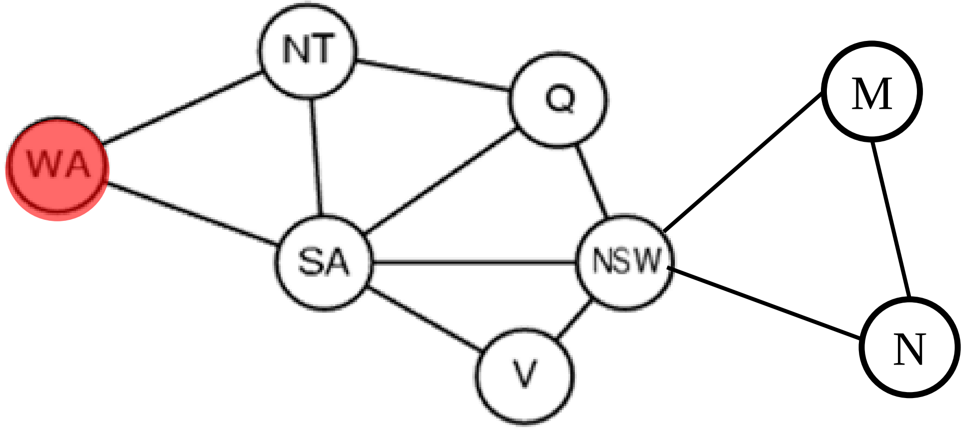
✓ ۴) مقادیر همه متغیرهای بین j و k مجددا تعیین می‌شوند.

تست ۴



تست ۴

WA=red Conf(WA)={}

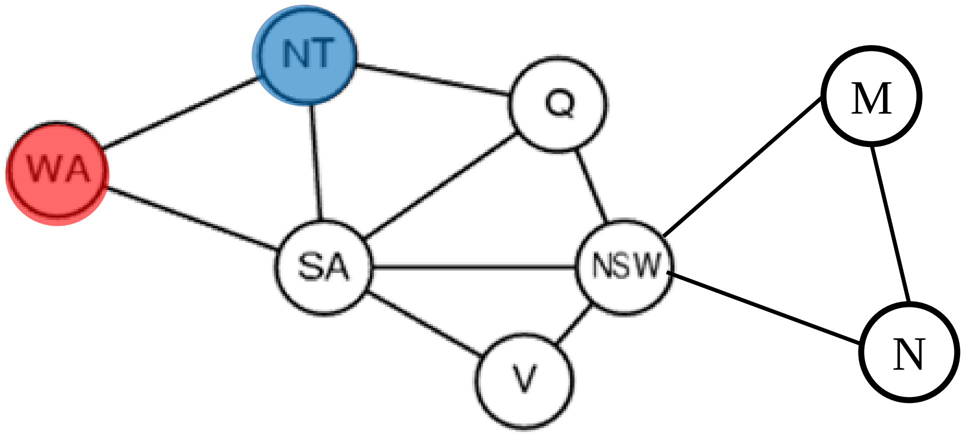


تست ۴

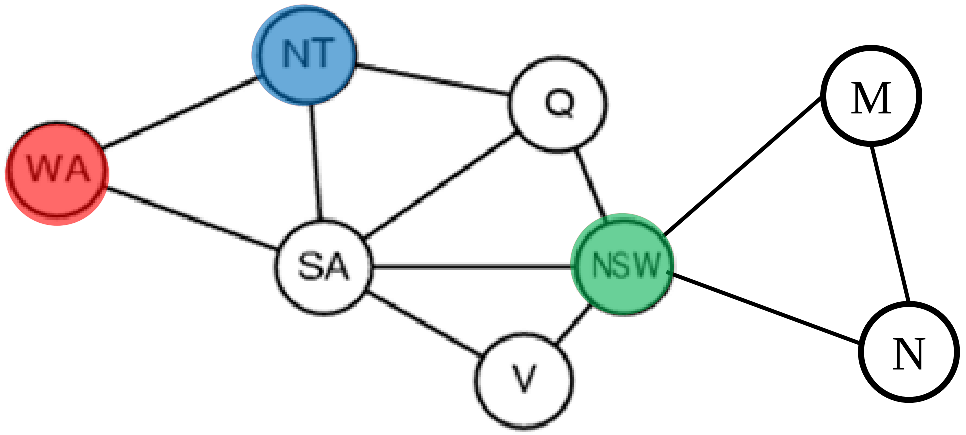
WA=red　　Conf(WA)={}

↓

NT=blue　　Conf(NT)={WA}



تست ۴



WA=red Conf(WA)={}

NT=blue Conf(NT)={WA}

NSW=green Conf(NSW)={}

تست ۴

WA=red Conf(WA)={}

↓

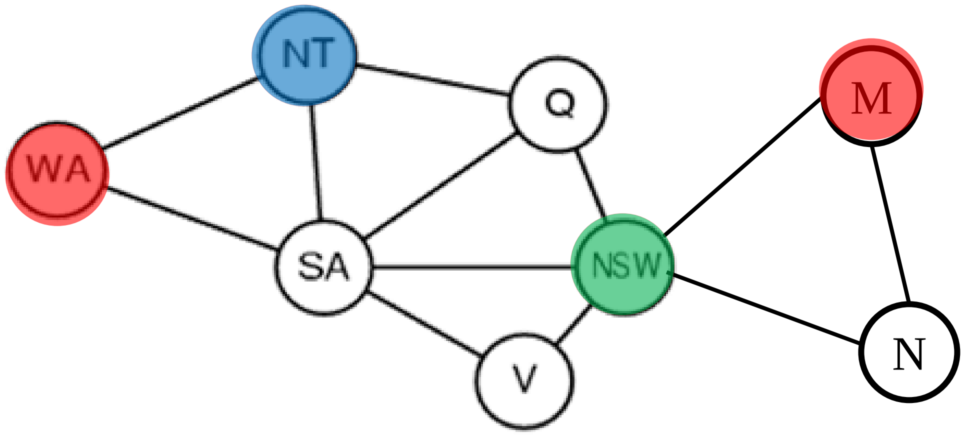
NT=blue Conf(NT)={WA}

↓

NSW=green Conf(NSW)={}

↓

M=red Conf(M)={NSW}



تست ۴

WA=red Conf(WA)={}

↓

NT=blue Conf(NT)={WA}

↓

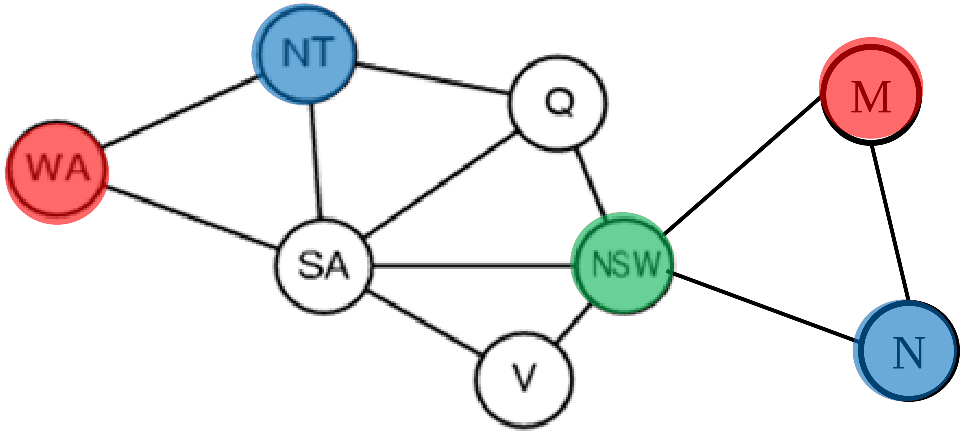
NSW=green Conf(NSW)={}

↓

M=red Conf(M)={NSW}

↓

N=blue Conf(N)={NSW, M}



WA=red Conf(WA)={}

↓

NT=blue Conf(NT)={WA}

↓

NSW=green Conf(NSW)={}

↓

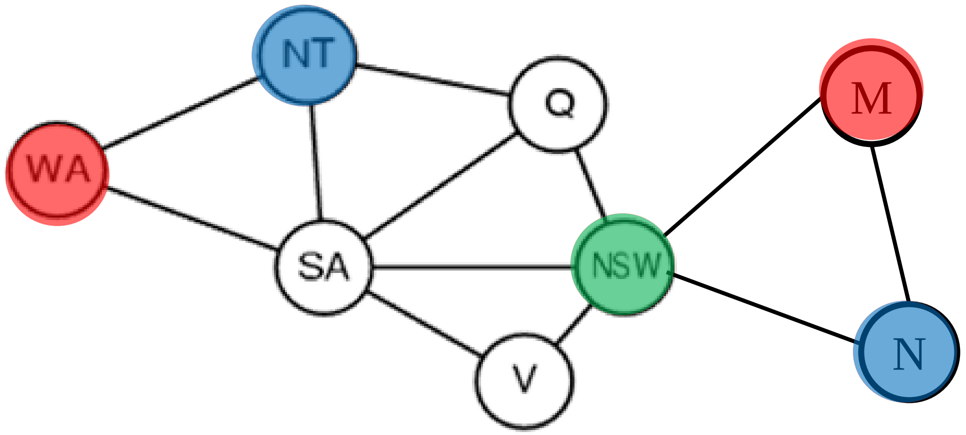
M=red Conf(M)={NSW}

↓

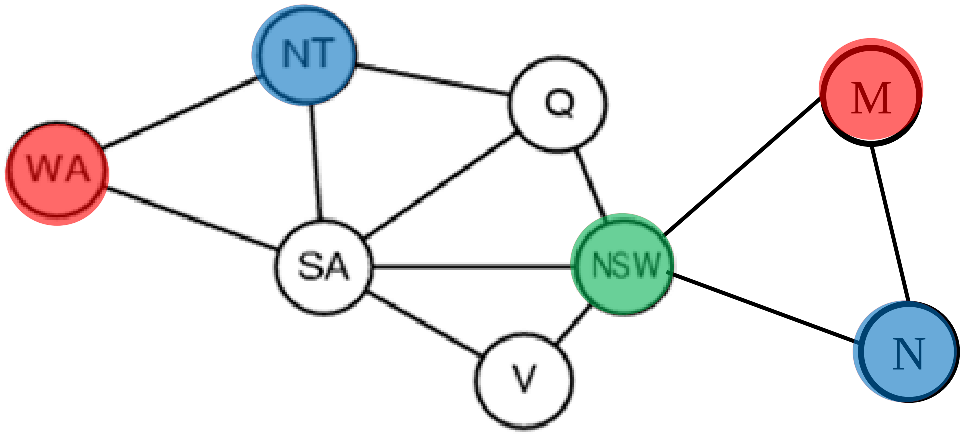
N=blue Conf(N)={NSW, M}

↓

SA=? Conf(SA)={WA, NT, NSW}



تست ۴



WA=red Conf(WA)={}

↓

NT=blue Conf(NT)={WA}

↓

NSW=green Conf(NSW)={}

↓

M=red Conf(M)={NSW}

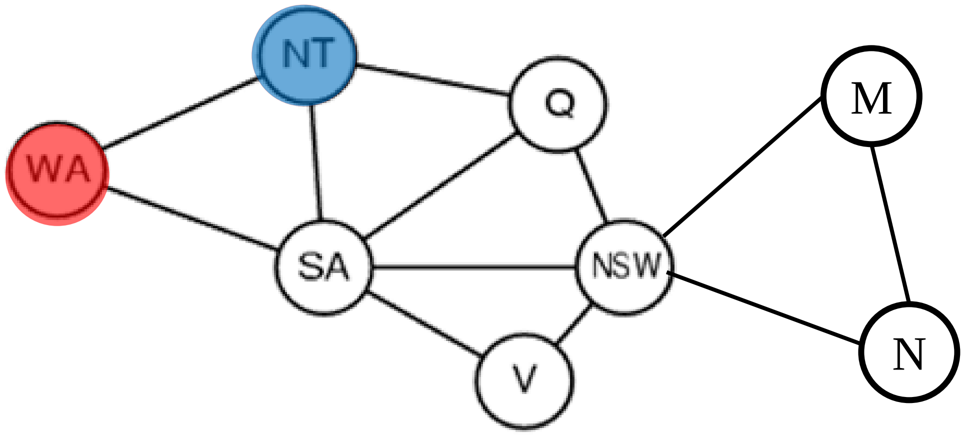
↓

N=blue Conf(N)={NSW, M}

↓

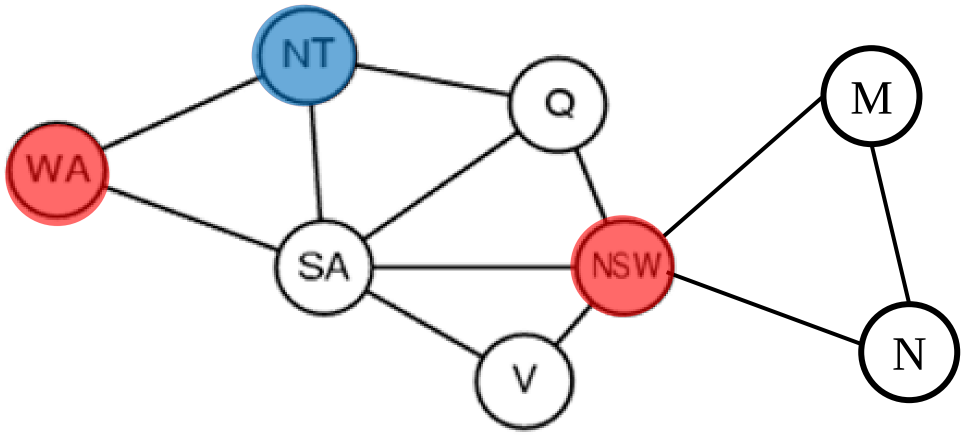
SA=? Conf(SA)={WA, NT, NSW}

تست ۴



WA=red	Conf(WA)={}
NT=blue	Conf(NT)={WA}
NSW=green	Conf(NSW)={}
M=red	Conf(M)={NSW}
N=blue	Conf(N)={NSW, M}
SA=?	Conf(SA)={WA, NT, NSW}

تست ۴



WA=red Conf(WA)={}

NT=blue Conf(NT)={WA}

NSW=green Conf(NSW)={}

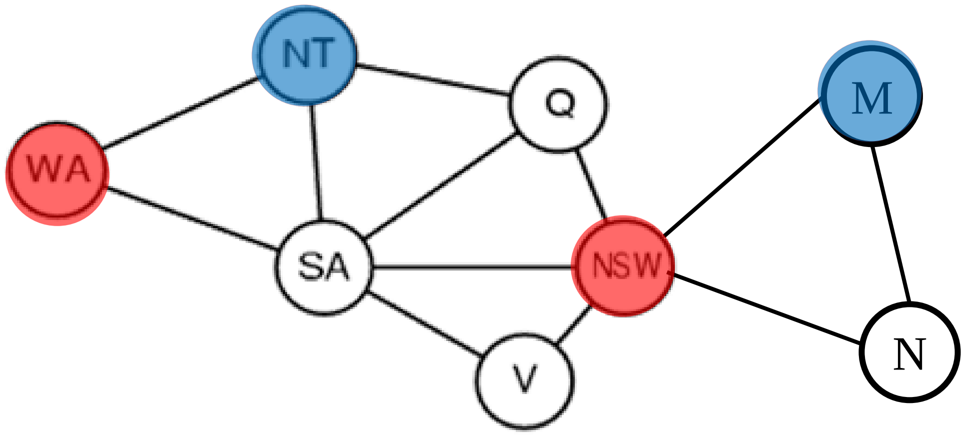
NSW=red Conf(NSW)={}

M=red Conf(M)={NSW}

N=blue Conf(N)={NSW, M}

SA=? Conf(SA)={WA, NT, NSW}

تست ۴



WA=red Conf(WA)={}

↓

NT=blue Conf(NT)={WA}

NSW=green Conf(NSW)={}

NSW=red Conf(NSW)={}

↓

M=red Conf(M)={NSW}

M=blue Conf(M)={NSW}

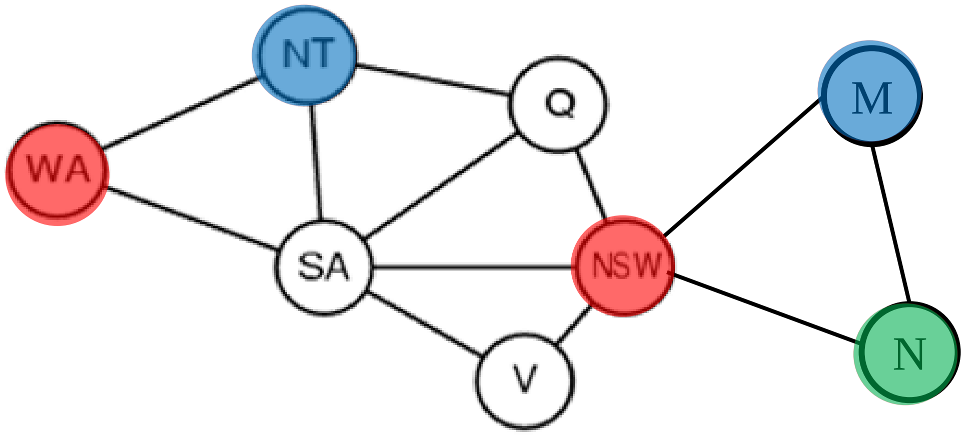
↓

N=blue Conf(N)={NSW, M}

↓

SA=? Conf(SA)={WA, NT, NSW}

تست ۴



WA=red Conf(WA)={}

NT=blue Conf(NT)={WA}

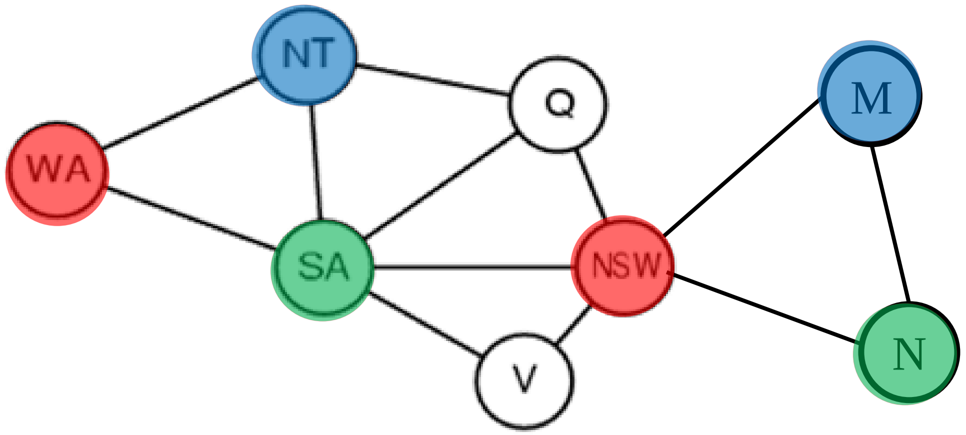
NSW=green Conf(NSW)={} | NSW=red Conf(NSW)={}

M=red Conf(M)={NSW} | M=blue Conf(M)={NSW}

N=blue Conf(N)={NSW, M} | N=green Conf(N)={NSW, M}

SA=? Conf(SA)={WA, NT, NSW}

تست ۴



WA=red   Conf(WA)={}

NT=blue   Conf(NT)={WA}

NSW=green   Conf(NSW)={}   NSW=red   Conf(NSW)={}

M=red   Conf(M)={NSW}   M=blue   Conf(M)={NSW}

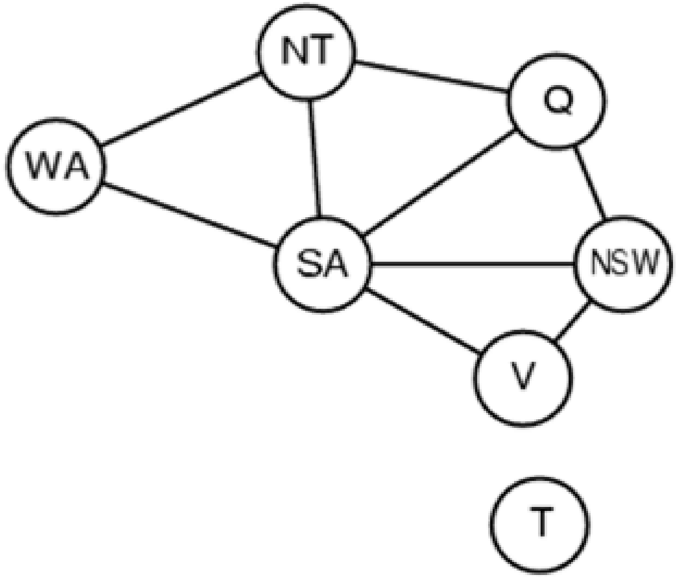
N=blue   Conf(N)={NSW, M}   N=green   Conf(N)={NSW, M}

SA=?   Conf(SA)={WA, NT, NSW}   SA=green   Conf(SA)={WA, NT, NSW}

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

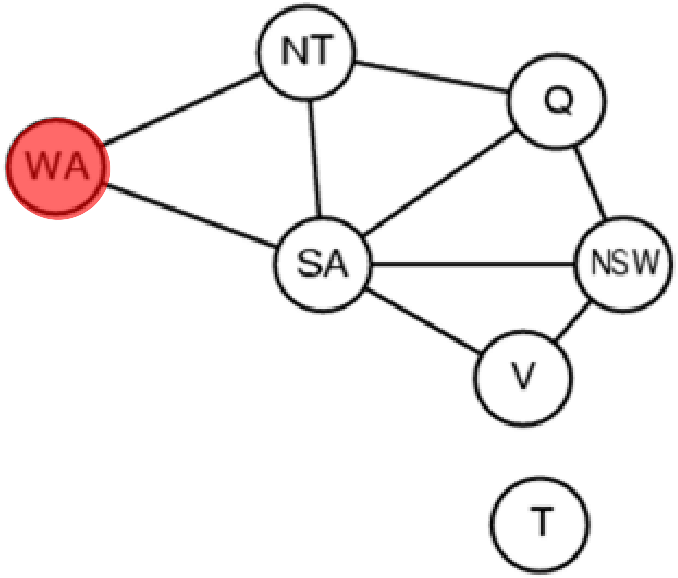


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red Conf(WA)={}

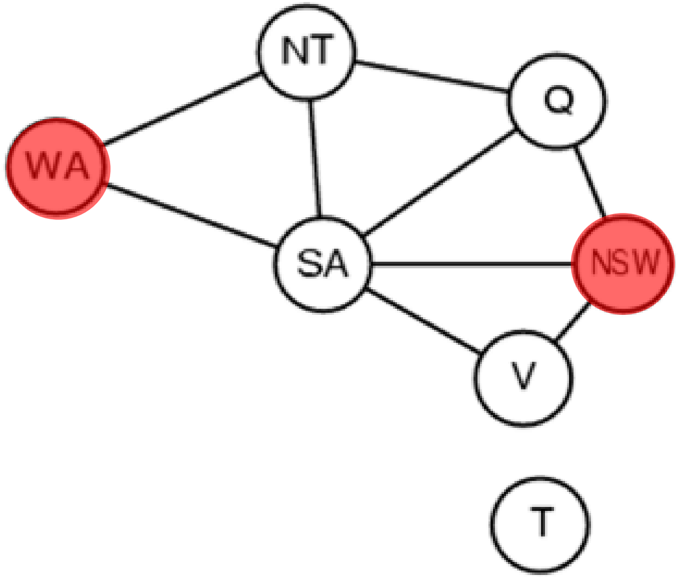


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

$$\begin{array}{ll} WA=red & Conf(WA)=\{\} \\ \downarrow & \\ NSW=red & Conf(NSW)=\{\} \end{array}$$



پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

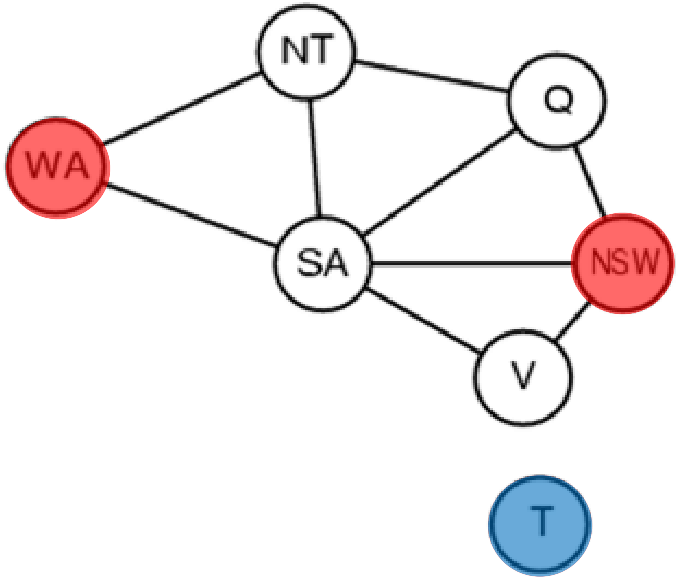
WA=red   Conf(WA)={}

↓

NSW=red   Conf(NSW)={}

↓

T=blue   Conf(T)={}

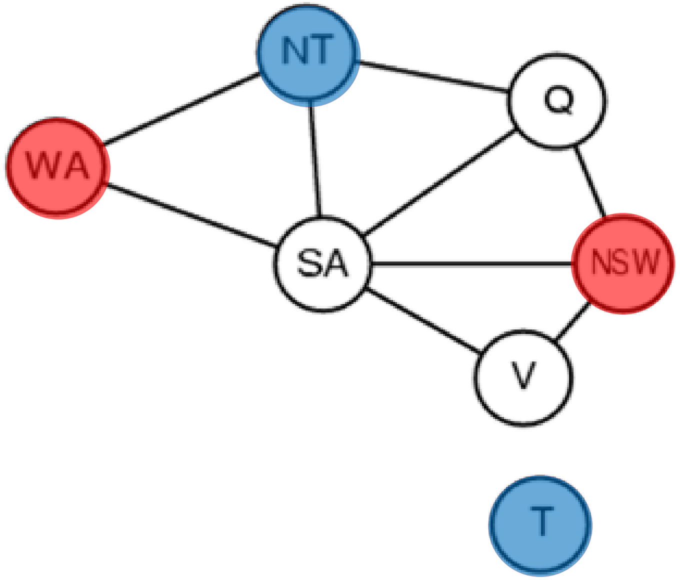


پرش رو به عقب

• مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red	Conf(WA)={}
↓	
NSW=red	Conf(NSW)={}
↓	
T=blue	Conf(T)={}
↓	
NT=blue	Conf(NT)={WA}

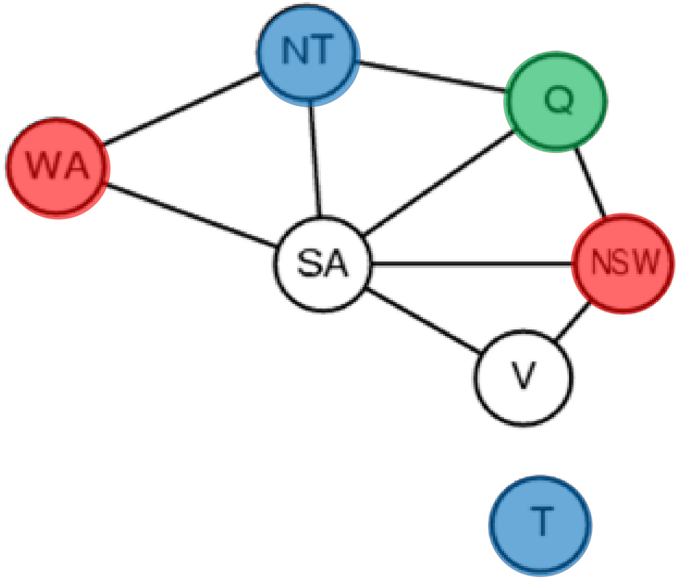


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red	Conf(WA)={}
↓	
NSW=red	Conf(NSW)={}
↓	
T=blue	Conf(T)={}
↓	
NT=blue	Conf(NT)={WA}
↓	
Q=green	Conf(Q)={NSW, NT}

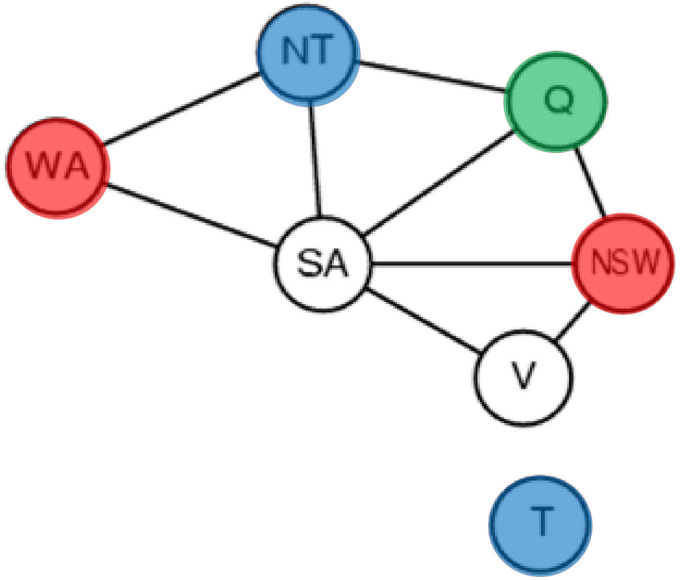


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red ↓	Conf(WA)={}
NSW=red ↓	Conf(NSW)={}
T=blue ↓	Conf(T)={}
NT=blue ↓	Conf(NT)={WA}
Q=green ↓	Conf(Q)={NSW, NT}
SA=?	Conf(SA)={WA, NSW, NT, Q}

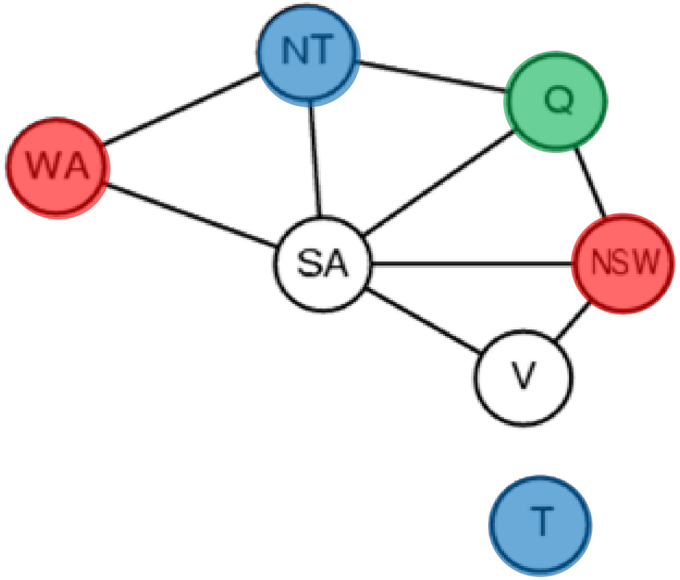


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red	Conf(WA)={}
↓	
NSW=red	Conf(NSW)={}
↓	
T=blue	Conf(T)={}
↓	
NT=blue	Conf(NT)={WA}
↓	
Q=green	Conf(Q)={NSW, NT}
↓	
SA=?	Conf(SA)={WA, NSW, NT, Q}

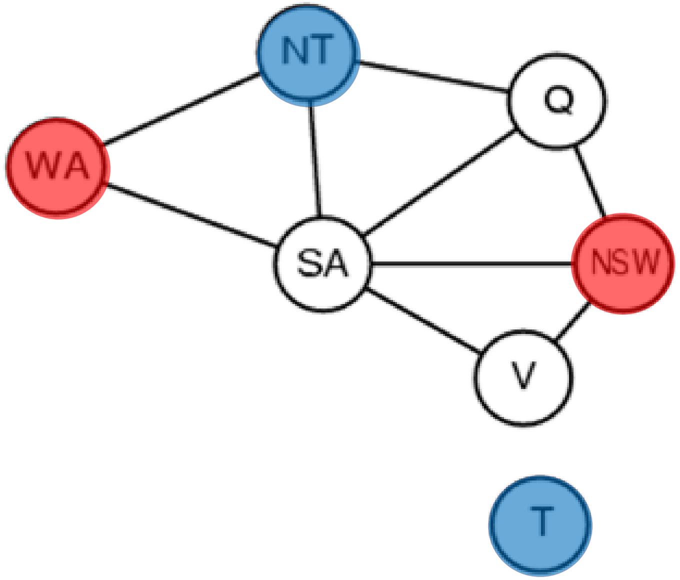


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red	Conf(WA)={}
↓	
NSW=red	Conf(NSW)={}
↓	
T=blue	Conf(T)={}
↓	
NT=blue	Conf(NT)={WA}
↓	
Q=green	Conf(Q)={NSW, NT}
↓	
SA=?	Conf(SA)={WA, NSW, NT, Q}

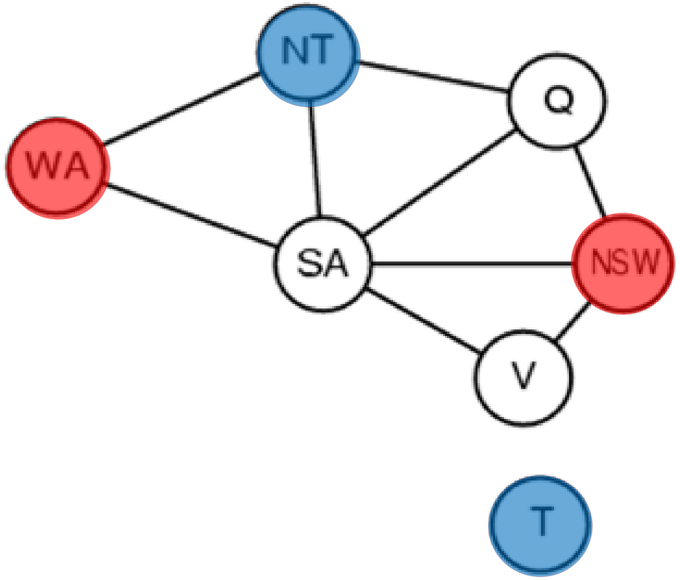


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red	Conf(WA)={}
NSW=red	Conf(NSW)={}
T=blue	Conf(T)={}
NT=blue	Conf(NT)={WA}
Q=green	Conf(Q)={NSW, NT}
SA=?	Conf(SA)={WA, NSW, NT, Q}

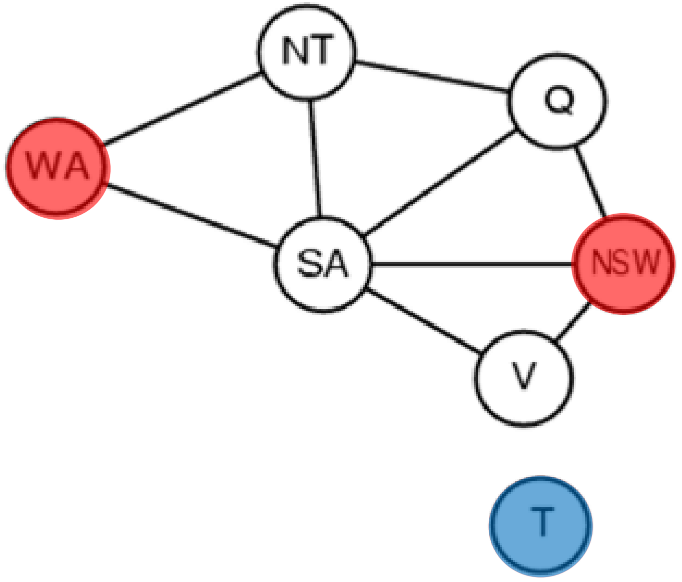


پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

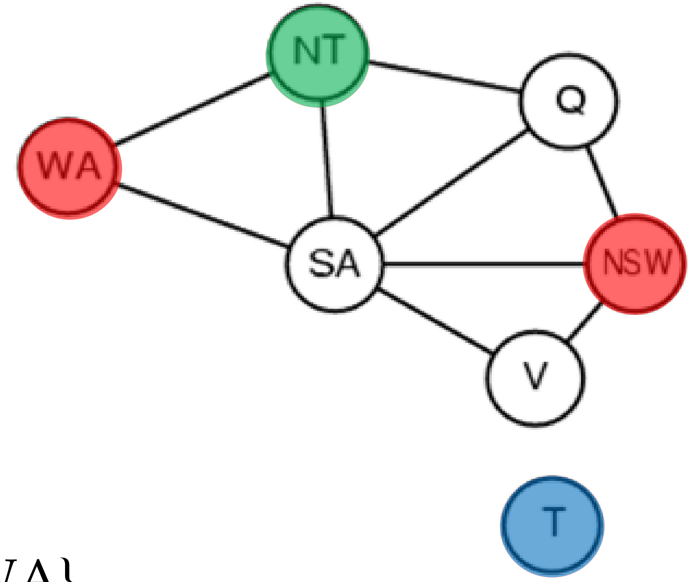
WA=red	Conf(WA)={}
↓	
NSW=red	Conf(NSW)={}
↓	
T=blue	Conf(T)={}
↓	
NT=blue ←	Conf(NT)={WA}
↓	
Q=green ←	Conf(Q)={NSW, NT}
↓	
SA=?	Conf(SA)={WA, NSW, NT, Q}



پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red　　Conf(WA)={}

NSW=red　　Conf(NSW)={}

T=blue　　Conf(T)={}

NT=blue　　Conf(NT)={WA}

NT=green　　Conf(NT)={WA}

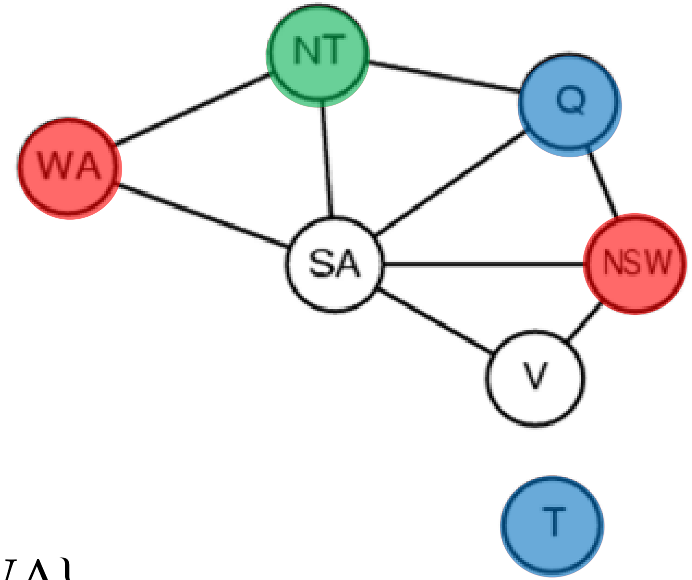
Q=green　　Conf(Q)={NSW, NT}

SA=?　　Conf(SA)={WA, NSW, NT, Q}

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

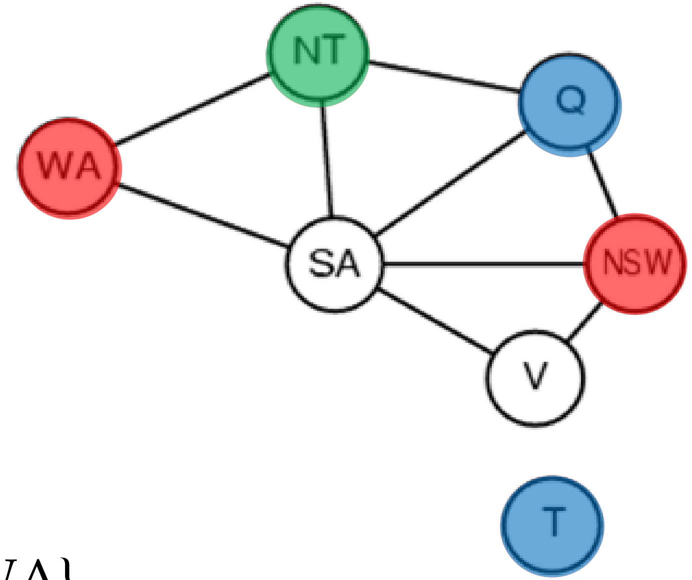


WA=red	Conf(WA)={}		
NSW=red	Conf(NSW)={}		
T=blue	Conf(T)={}		
NT=blue	Conf(NT)={WA}	NT=green	Conf(NT)={WA}
Q=green	Conf(Q)={NSW, NT}	Q=blue	Conf(Q)={NSW, NT}
SA=?	Conf(SA)={WA, NSW, NT, Q}		

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

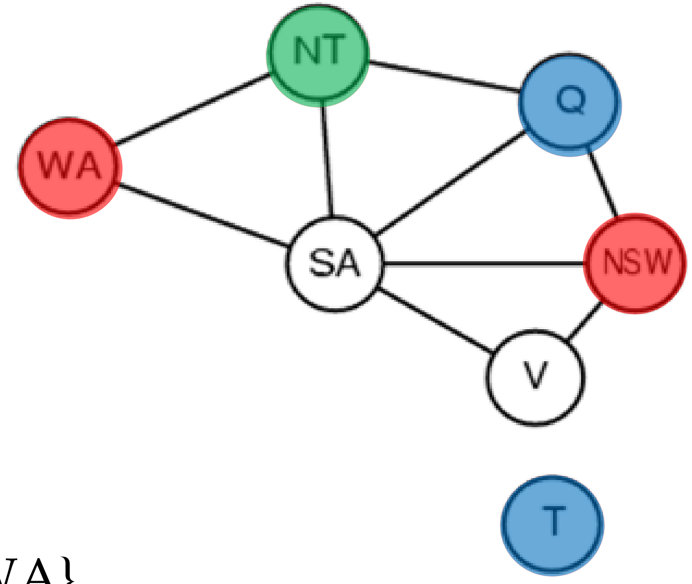


WA=red	Conf(WA)={}		
NSW=red	Conf(NSW)={}		
T=blue	Conf(T)={}		
NT=blue	Conf(NT)={WA}	NT=green	Conf(NT)={WA}
Q=green	Conf(Q)={NSW, NT}	Q=blue	Conf(Q)={NSW, NT}
SA=?	Conf(SA)={WA, NSW, NT, Q}	SA=?	Conf(SA)={WA, NSW, NT, Q}

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red   Conf(WA)={}

↓

NSW=red   Conf(NSW)={}

↓

T=blue   Conf(T)={}

↓

NT=blue   Conf(NT)={WA}

↓

Q=green   Conf(Q)={NSW, NT}

↓

SA=?   Conf(SA)={WA, NSW, NT, Q}

NT=green   Conf(NT)={WA}

↓

Q=blue   Conf(Q)={NSW, NT}

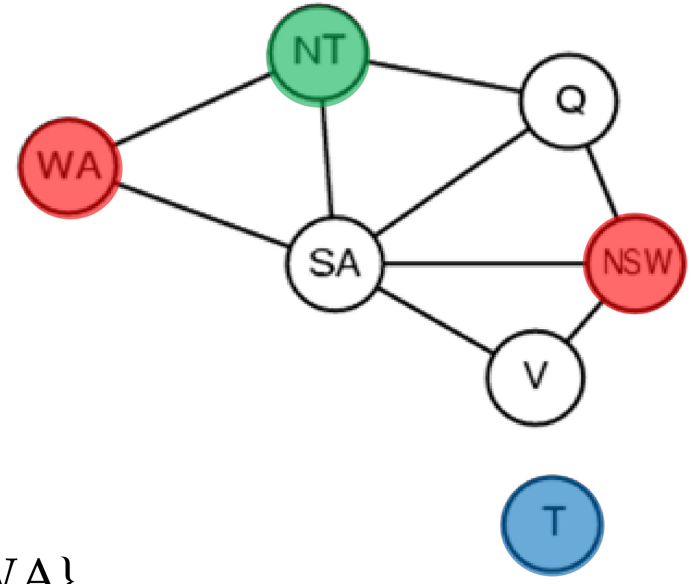
↓

SA=?   Conf(SA)={WA, NSW, NT, Q}

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red　　Conf(WA)={}

↓

NSW=red　　Conf(NSW)={}

↓

T=blue　　Conf(T)={}

↓

NT=blue　　Conf(NT)={WA}

↓

Q=green　　Conf(Q)={NSW, NT}

↓

SA=?　　Conf(SA)={WA, NSW, NT, Q}

NT=green　　Conf(NT)={WA}

↓

Q=blue　　Conf(Q)={NSW, NT}

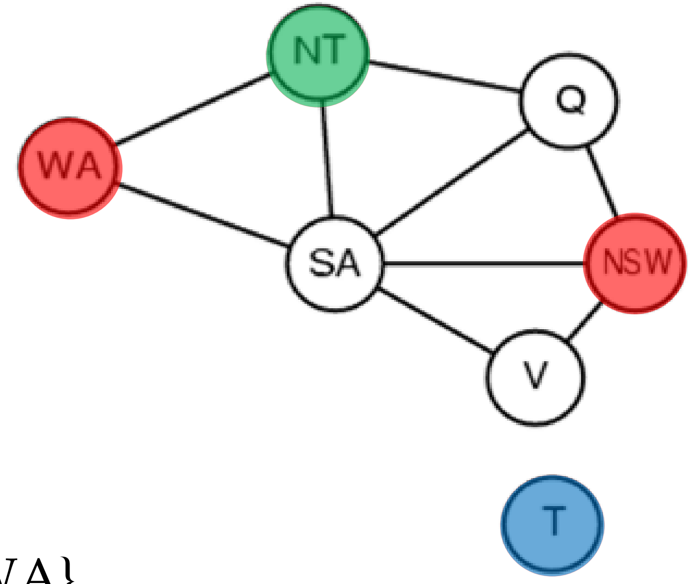
↓

SA=?　　Conf(SA)={WA, NSW, NT, Q}

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red　　Conf(WA)={}

NSW=red　　Conf(NSW)={}

T=blue　　Conf(T)={}

NT=blue　　Conf(NT)={WA}

Q=green　　Conf(Q)={NSW, NT}

SA=?　　Conf(SA)={WA, NSW, NT, Q}

NT=green　　Conf(NT)={WA}

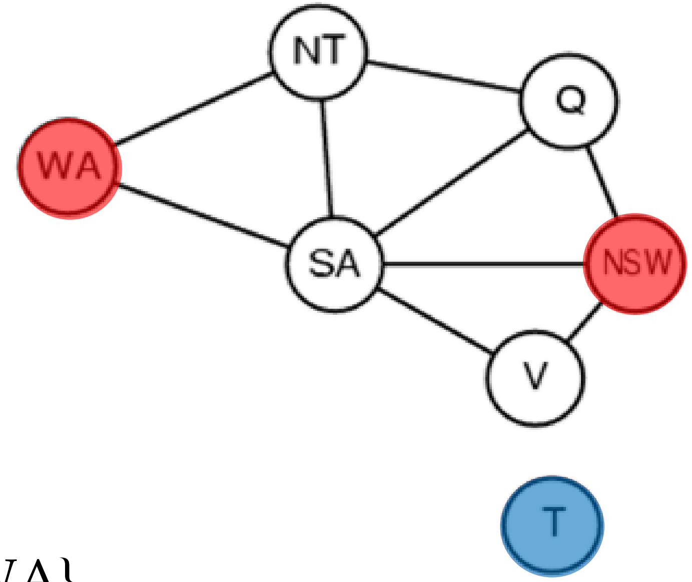
Q=blue　　Conf(Q)={NSW, NT}

SA=?　　Conf(SA)={WA, NSW, NT, Q}

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



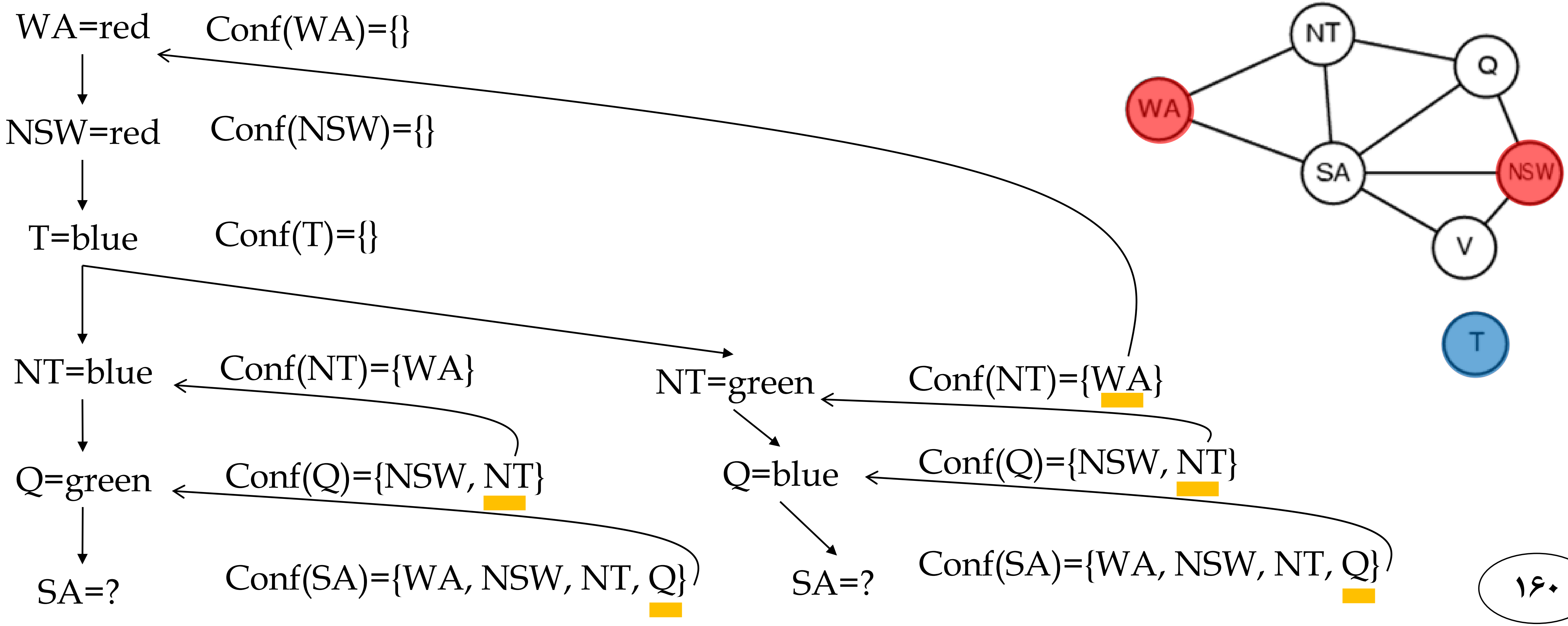
WA=red	Conf(WA)={}
NSW=red	Conf(NSW)={}
T=blue	Conf(T)={}
NT=blue	Conf(NT)={WA}
Q=green	Conf(Q)={NSW, NT}
SA=?	Conf(SA)={WA, NSW, NT, Q}

NT=green	Conf(NT)={WA}
Q=blue	Conf(Q)={NSW, NT}
SA=?	Conf(SA)={WA, NSW, NT, Q}

پرش رو به عقب

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



پرش رو به عقب با هدایت برخورد

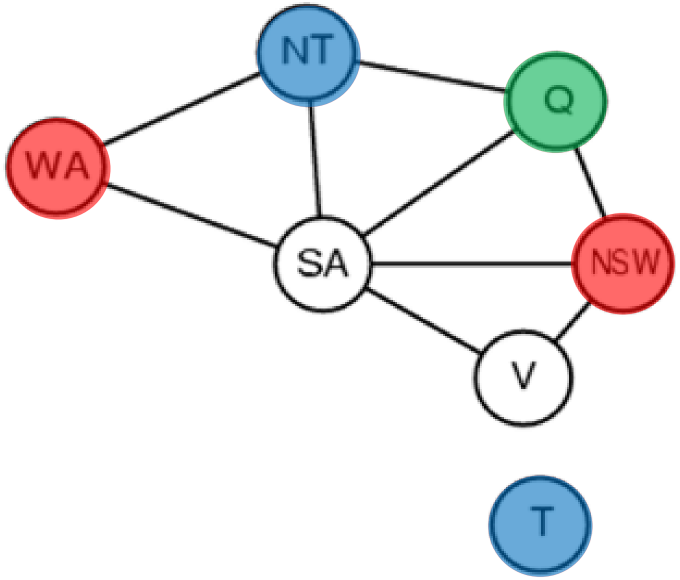
- تعریف دقیق مجموعه برخورد: مجموعه برخورد متغیر X عبارت است از مجموعه متغیرهایی که قبل از X مقدار گرفته‌اند و باعث می‌شوند متغیر X به همراه متغیرهای بعدی فاقد راه‌حل سازگار باشد.
- پرش رو به عقب با هدایت برخورد (Conflict-directed Backjumping)
  - در نظر بگیرید که X_j متغیر فعلی و $conf(X_j)$ مجموعهٔ تناقض آن باشد.
  - اگر تمام مقادیر ممکن برای X_j با شکست مواجه شوند، الگوریتم به X_i برخواهد گشت که آخرین متغیر مقدار داده شده و متعلق به مجموعهٔ تناقض X_j می‌باشد و انتساب زیر را انجام می‌دهیم:

$$conf\ (X_i) \leftarrow conf\ (X_i) \cup conf\ (X_j) - \{X_i\}$$

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



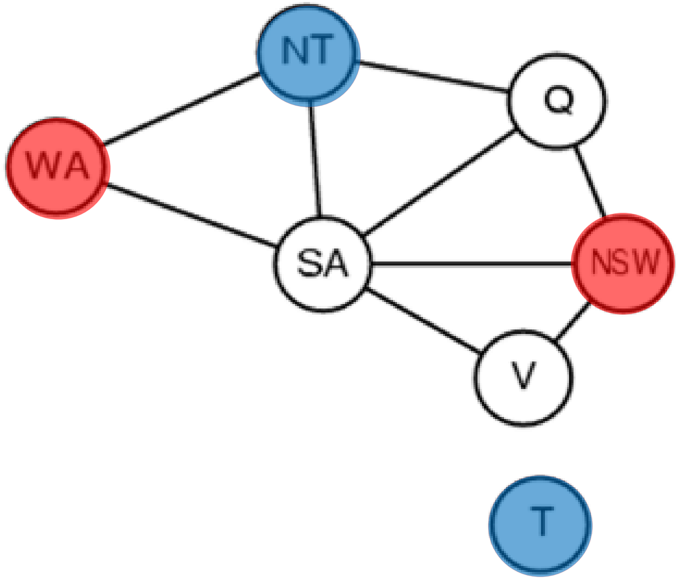
WA=red	Conf(WA)={}
↓ NSW=red	Conf(NSW)={}
↓ T=blue	Conf(T)={}
↓ NT=blue	Conf(NT)={WA}
↓ Q=green	Conf(Q)={NSW, NT} ⟶ Conf(Q)={WA, NSW, NT}
↓ SA=?	Conf(SA)={WA, NSW, NT, Q}

یعنی هیچ راه‌حل سازگاری با انتساب‌های فعلی WA, NSW, NT از Q=green به بعد وجود ندارد.

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red	Conf(WA)={}
↓	
NSW=red	Conf(NSW)={}
↓	
T=blue	Conf(T)={}
↓	
NT=blue	Conf(NT)={WA} ⟶ Conf(NT)={WA, NSW}
↓	
Q=green	Conf(Q)={WA, NSW, NT}
↓	
SA=?	Conf(SA)={WA, NSW, NT, Q}

یعنی هیچ راه‌حل سازگاری با انتساب‌های فعلی ,WA
NSWاز NT=blue به بعد وجود ندارد.

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?

WA=red   Conf(WA)={}

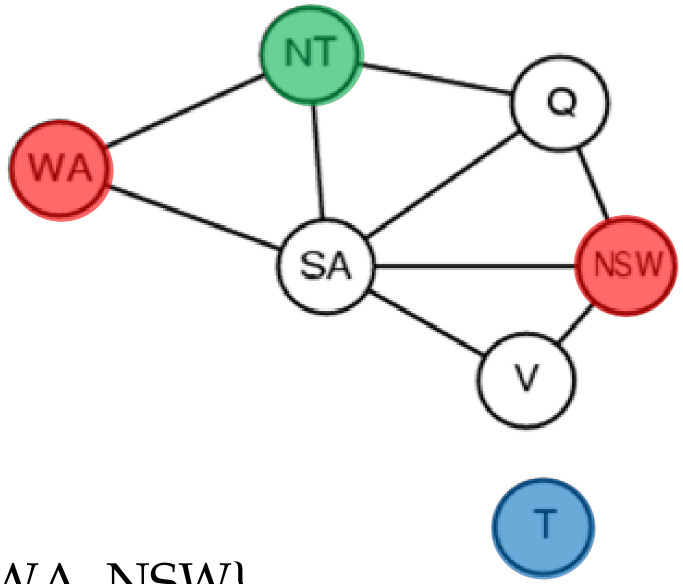
NSW=red   Conf(NSW)={}

T=blue   Conf(T)={}

NT=blue   Conf(NT)={WA, NSW}   NT=green   Conf(NT)={WA, NSW}

Q=green   Conf(Q)={WA, NSW, NT}

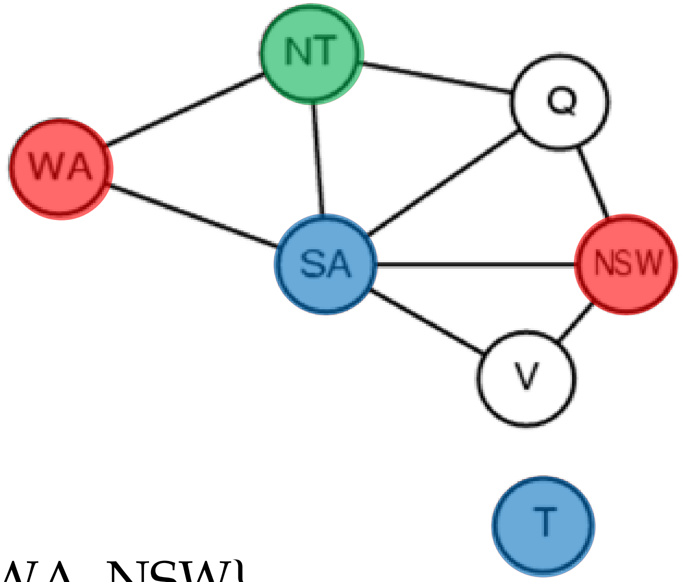
SA=?   Conf(SA)={WA, NSW, NT, Q}



پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red   Conf(WA)={}

NSW=red   Conf(NSW)={}

T=blue   Conf(T)={}

NT=blue   Conf(NT)={WA, NSW}

Q=green   Conf(Q)={WA, NSW, NT}

SA=?   Conf(SA)={WA, NSW, NT, Q}

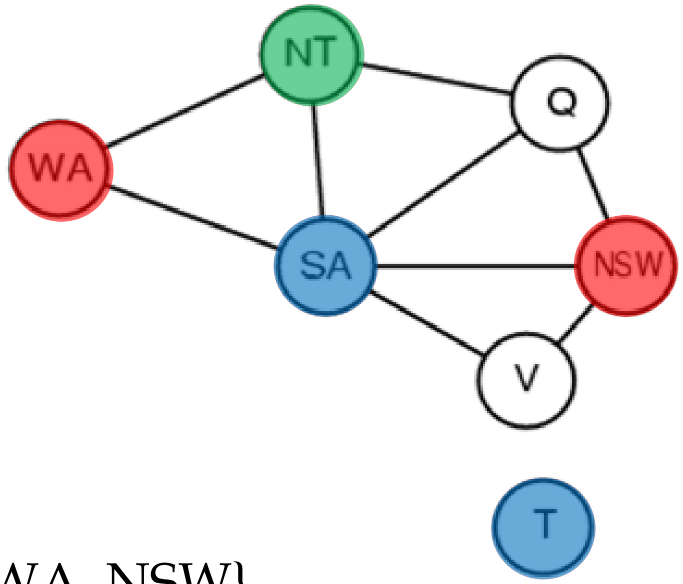
NT=green   Conf(NT)={WA, NSW}

SA=blue   Conf(SA)={WA, NSW, NT}

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red   Conf(WA)={}

NSW=red   Conf(NSW)={}

T=blue   Conf(T)={}

NT=blue   Conf(NT)={WA, NSW}   NT=green   Conf(NT)={WA, NSW}

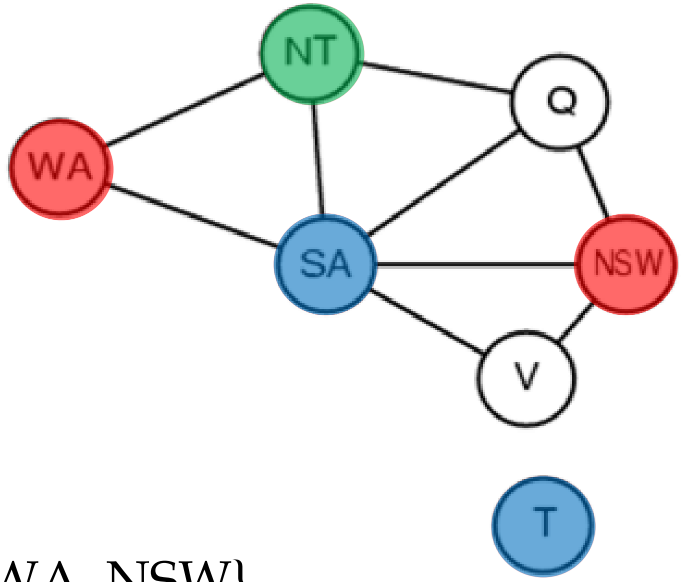
Q=green   Conf(Q)={WA, NSW, NT}   SA=blue   Conf(SA)={WA, NSW, NT}

SA=?   Conf(SA)={WA, NSW, NT, Q}   Q=?   Conf(Q)={NSW, NT, SA}

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red　　Conf(WA)={}

NSW=red　　Conf(NSW)={}

T=blue　　Conf(T)={}

NT=blue　　Conf(NT)={WA, NSW}

Q=green　　Conf(Q)={WA, NSW, NT}

SA=?　　Conf(SA)={WA, NSW, NT, Q}

NT=green　　Conf(NT)={WA, NSW}

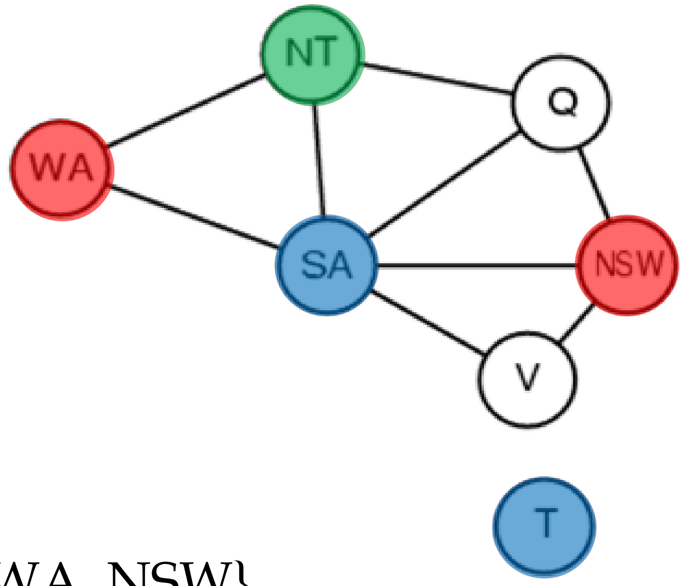
SA=blue　　Conf(SA)={WA, NSW, NT}

Q=?　　Conf(Q)={NSW, NT, SA}

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red   Conf(WA)={}

↓

NSW=red   Conf(NSW)={}

↓

T=blue   Conf(T)={}

↓

NT=blue   Conf(NT)={WA, NSW}   NT=green   Conf(NT)={WA, NSW}

↓

Q=green   Conf(Q)={WA, NSW, NT}   SA=blue   Conf(SA)={WA, NSW, NT}

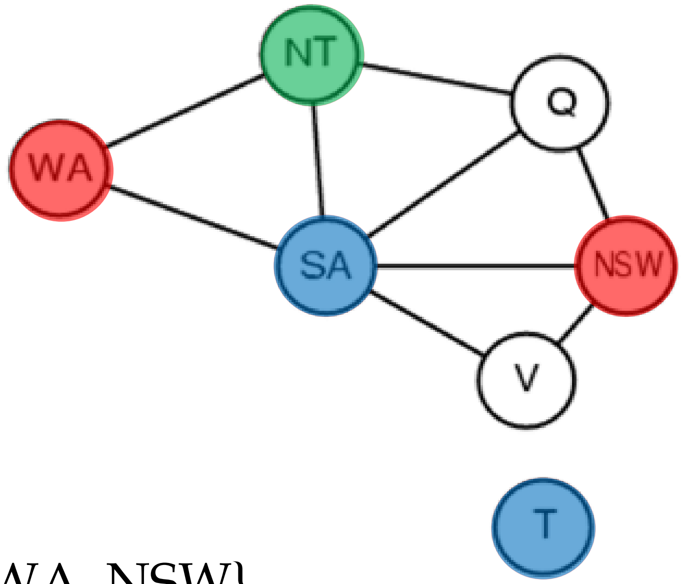
↓

SA=?   Conf(SA)={WA, NSW, NT, Q}   Q=?   Conf(Q)={NSW, NT, SA}

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red   Conf(WA)={}

NSW=red   Conf(NSW)={}

T=blue   Conf(T)={}

NT=blue   Conf(NT)={WA, NSW}

Q=green   Conf(Q)={WA, NSW, NT}

SA=?   Conf(SA)={WA, NSW, NT, Q}

NT=green   Conf(NT)={WA, NSW}

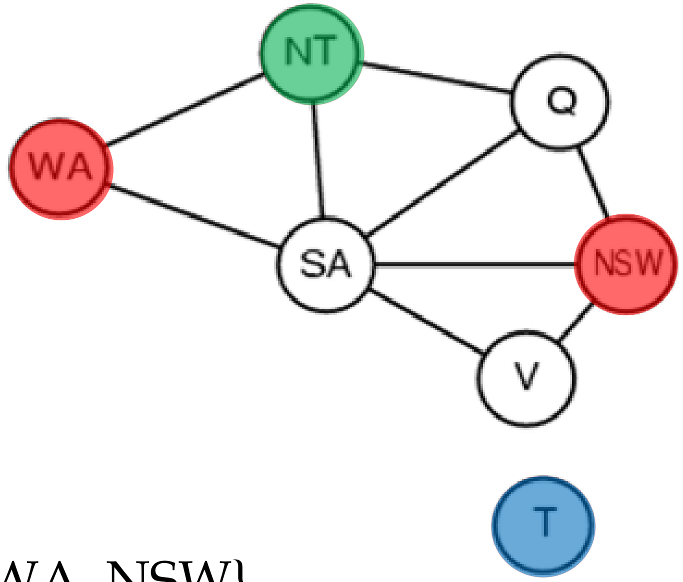
SA=blue   Conf(SA)={WA, NSW, NT}

Q=?   Conf(Q)={NSW, NT, SA}

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red — Conf(WA)={}

NSW=red — Conf(NSW)={}

T=blue — Conf(T)={}

NT=blue — Conf(NT)={WA, NSW}

Q=green — Conf(Q)={WA, NSW, NT}

SA=? — Conf(SA)={WA, NSW, NT, Q}

NT=green — Conf(NT)={WA, NSW}

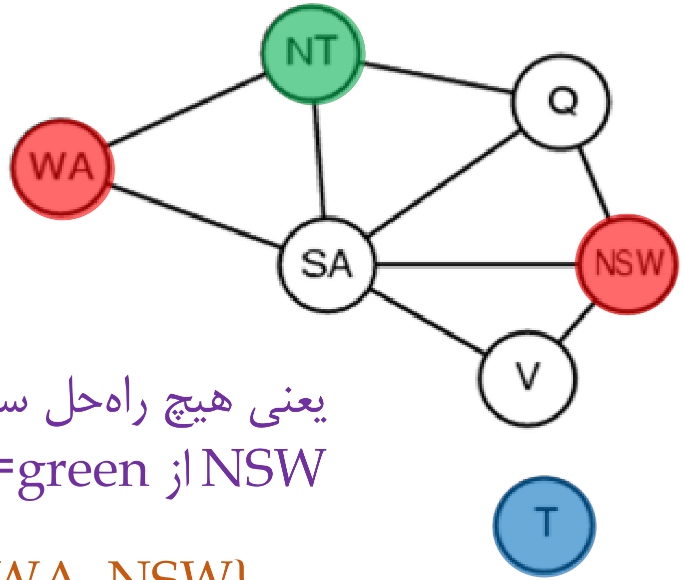
SA=blue — Conf(SA)={WA, NSW, NT}

Q=? — Conf(Q)={NSW, NT, SA}

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red Conf(WA)={}

↓

NSW=red Conf(NSW)={}

↓

T=blue Conf(T)={}

↓

NT=blue Conf(NT)={WA, NSW}

↓

Q=green Conf(Q)={WA, NSW, NT}

↓

SA=? Conf(SA)={WA, NSW, NT, Q}

یعنی هیچ راه‌حل سازگاری با انتساب‌های فعلی WA,
NSW از NT=green به بعد وجود ندارد.

NT=green Conf(NT)={WA, NSW}

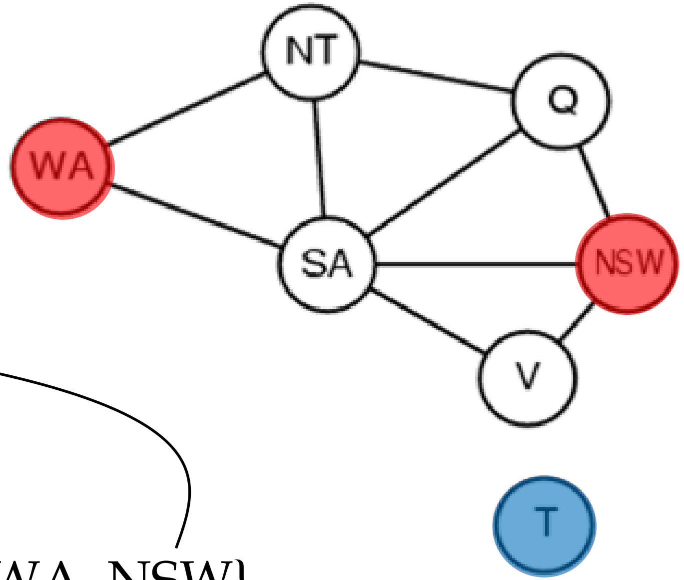
SA=blue Conf(SA)={WA, NSW, NT}

Q=? Conf(Q)={NSW, NT, SA}

پرش رو به عقب با هدایت برخورد

- مثال: ترتیب رنگ ناحیه‌ها را به صورت زیر فرض کنید

WA=red, NSW=red, T=blue, NT=blue, Q=green, SA=?



WA=red Conf(WA)={}

↓

NSW=red Conf(NSW)={} → Conf(NSW)={WA} امتحان می‌شود و ...

ارتباط میان NSW و WA کشف می‌شود، بنابراین NSW=green

↓

T=blue Conf(T)={}

↓

NT=blue Conf(NT)={WA, NSW} NT=green Conf(NT)={WA, NSW}

↓

Q=green Conf(Q)={WA, NSW, NT} SA=blue Conf(SA)={WA, NSW, NT}

↓

SA=? Conf(SA)={WA, NSW, NT, Q} Q=? Conf(Q)={NSW, NT, SA}

جستجوی محلی برای CSPها

حل CSPها با الگوریتم‌های جستجوی محلی

- در فرمول‌سازی CSP به‌صورت یک مسئله جستجو، مسیر اهمیتی ندارد بنابراین می‌توانیم از فرمول‌سازی حالت کامل استفاده کنیم.
  - حالت اولیه: انتساب مقادیر به تمام متغیرها (مثلا به‌صورت تصادفی)
  - اقدام‌ها: انتساب یک مقدار جدید به یکی از متغیرها
  - تابع هیوریستیک $h(s)$: تعداد محدودیت‌های نقض‌شده
  - کمینه سراسری: $h(s)=0$

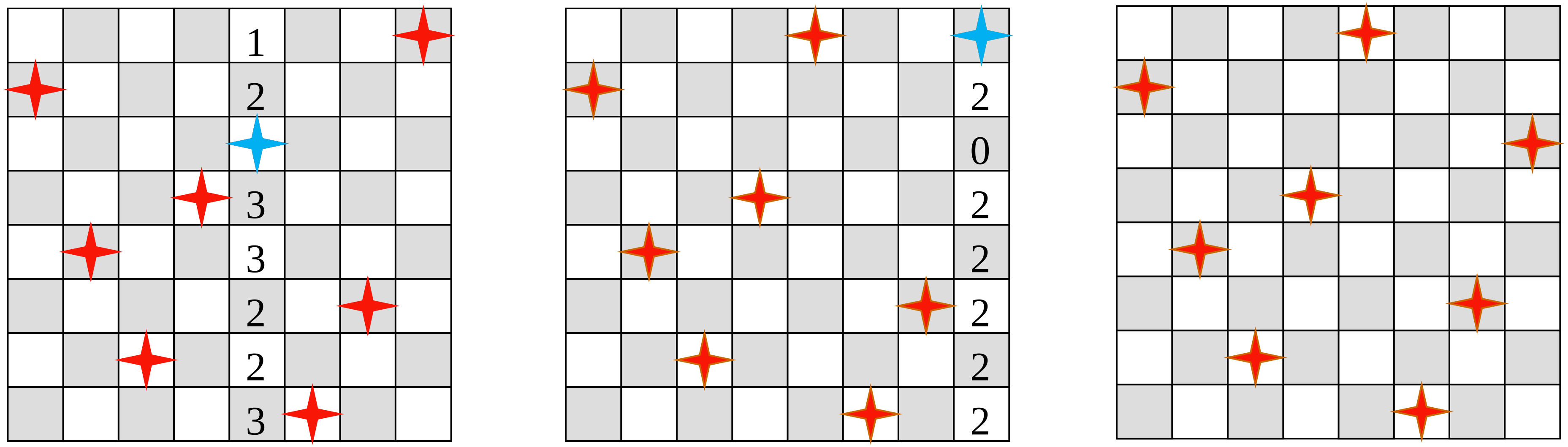
الگوریتم Min-Conflicts

- هیوریستیک min-conflicts
  - مقداری را انتخاب کن که کمترین تعداد محدودیت‌ها را نقض کند،
  - یعنی، تپه نوردی با هیوریستیک ” تعداد کل محدودیت‌های نقض شده“

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
   inputs: csp, a constraint satisfaction problem
           max_steps, the number of steps allowed before giving up

   current ← an initial complete assignment for csp
   for i = 1 to max_steps do
      if current is a solution for csp then return current
      var ← a randomly chosen conflicted variable from csp.VARIABLES
      value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
      set var = value in current
   return failure
```

حل ۸–وزیر با فرمول‌سازی کامل CSP

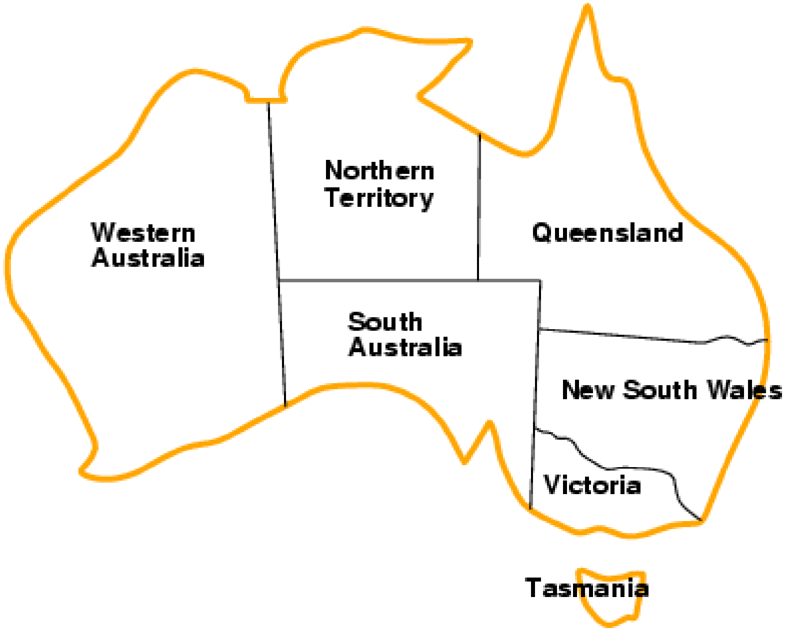


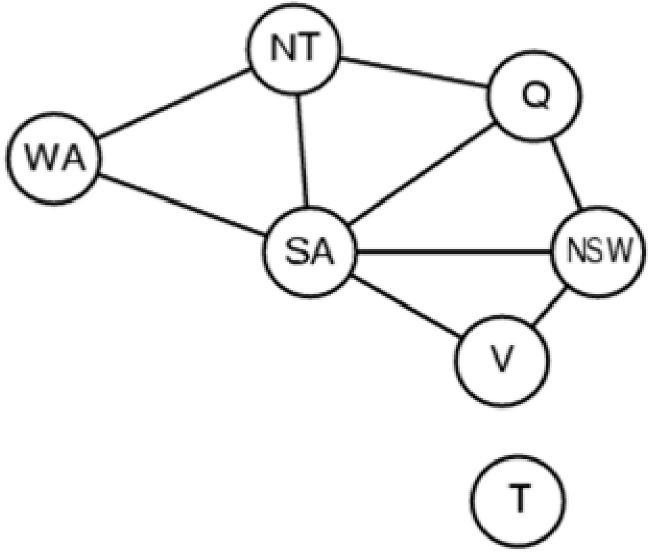
- پس از جای‌گذاری اولیه مهره‌ها، این روش حتی با در نظر گرفتن تعداد یک میلیون وزیر، مسئله را به طور متوسط در ۵۰ مرحله حل می‌کند.
- از آن‌جا که در مسئله n وزیر، حالت‌های هدف به طور متراکم در فضای حالت توزیع شده‌اند، روش جستجوی محلی درمورد آن بسیار مفید واقع می‌شود.
  - می‌توان گفت در نزدیکی هر حالت اولیه یک حالت هدف وجود دارد.

ساختار مسائل

ساختار مسائل

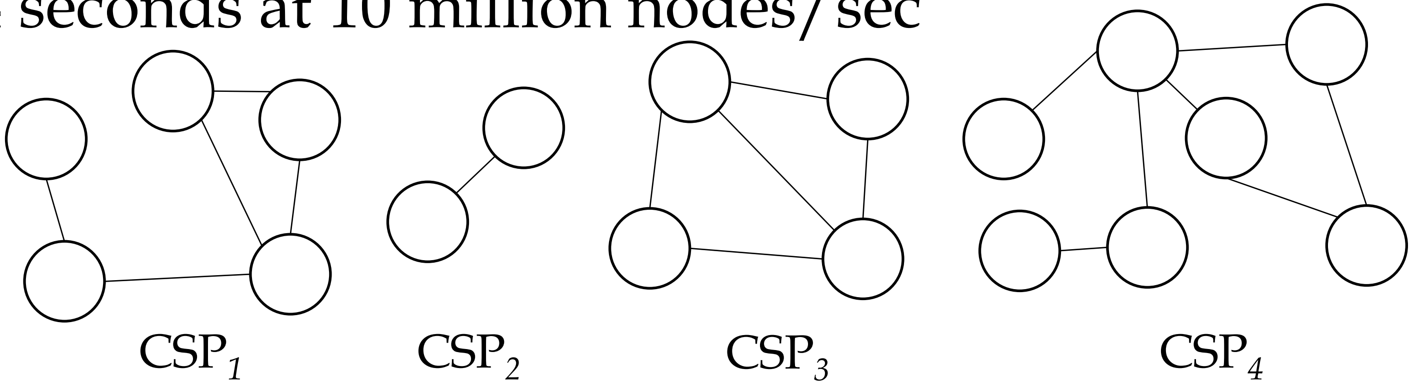
- در مسائل دنیای واقعی می‌توان مسائل را به چندین زیرمسئله شکست و هر زیرمسئله را به‌طور مستقل حل کرد و در نهایت آن‌ها را ترکیب نمود.
  - مثال: Tasmania و جزیره اصلی زیرمسائل مستقل هستند.
  - با توجه به مولفه‌های همبند در گراف محدودیت قابل شناسایی هستند.





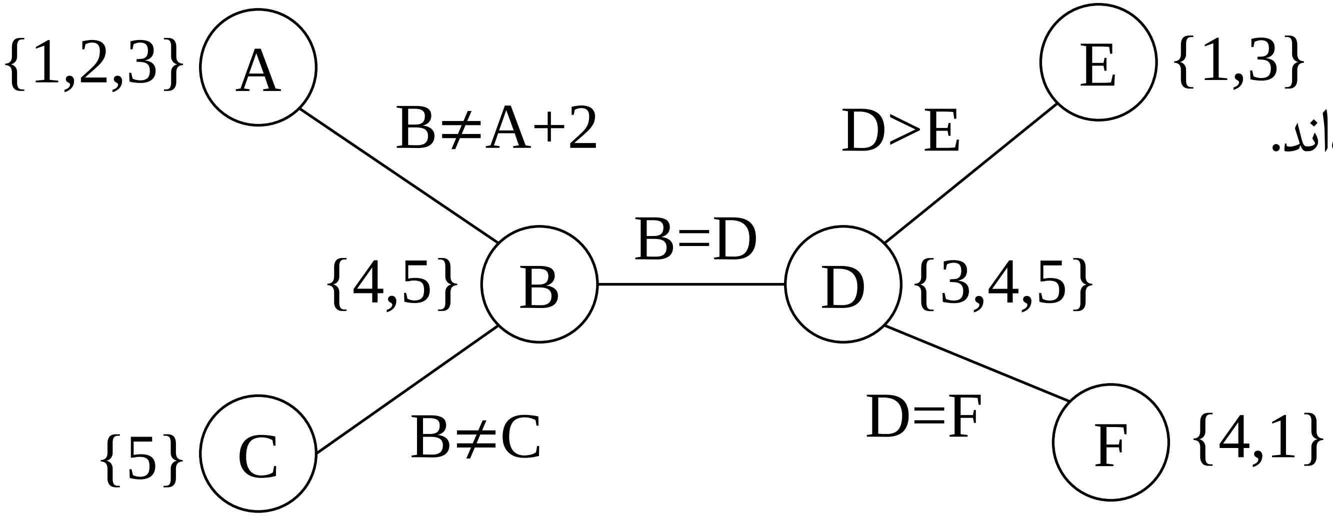
ساختار مسائل

- اگر هر مؤلفه همبند گراف را به‌عنوان جزئی از یک مسئله مجزا (CSP_i) درنظر بگیریم و برای هر یک از این مسائل انتساب S_i به‌عنوان راه‌حل منظور شود، درنهایت $\cup_i S_i$ راه‌حلی برای مسئله کلی $\cup_i CSP_i$ خواهد بود.
- فرض کنید هر زیرمسأله شامل c متغیر از مجموع n متغیر و d حداکثر سایز دامنه باشد.
  - هزینه راه حل در بدترین حالت خطی (برحسب n) و برابر $O((n/c) \cdot d^c)$ می‌باشد
  - مثال $n = 80, d = 2, c = 20$
- 2^{80} = 4 billion years at 10 million nodes/sec
- 4.2^{20} = 0.4 seconds at 10 million nodes/sec



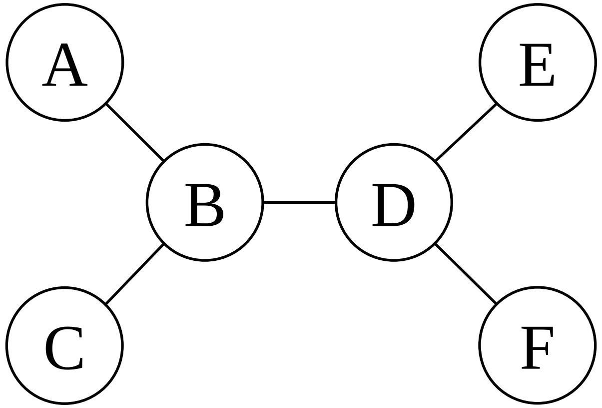
CSPهای دارای ساختار درختی

- مسائلی که بتوان آن‌ها را به زیرمسئله‌های کاملاً مجزا تقسیم کرد نادرند. در اغلب اوقات مسائل فرعی CSP به هم مرتبط‌اند و راحت‌ترین حالت زمانی است که این محدودیت‌ها به‌صورت یک درخت به هم مرتبط شده باشند.
- می‌خواهیم نشان دهیم چگونه می‌توان گراف‌های محدودیت کلی را به‌صورت گراف محدودیت درختی تبدیل کنیم و از این طریق به حل مسئله اصلی بپردازیم.
- CSP دارای ساختار درختی
  - هر جفت متغیر با یک مسیر به هم مرتبط شده‌اند.

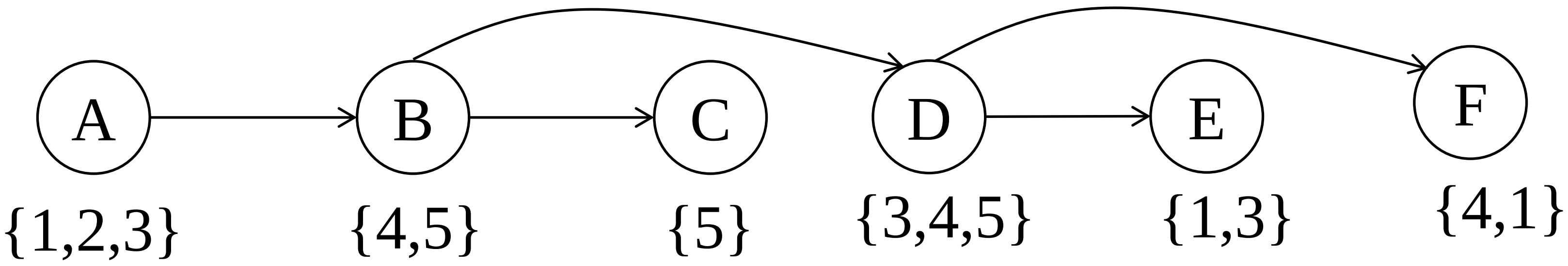


الگوریتمی برای CSPهای دارای ساختار درختی

- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)

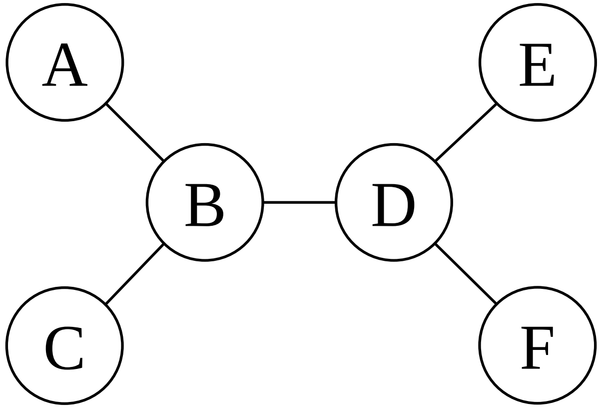


- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.

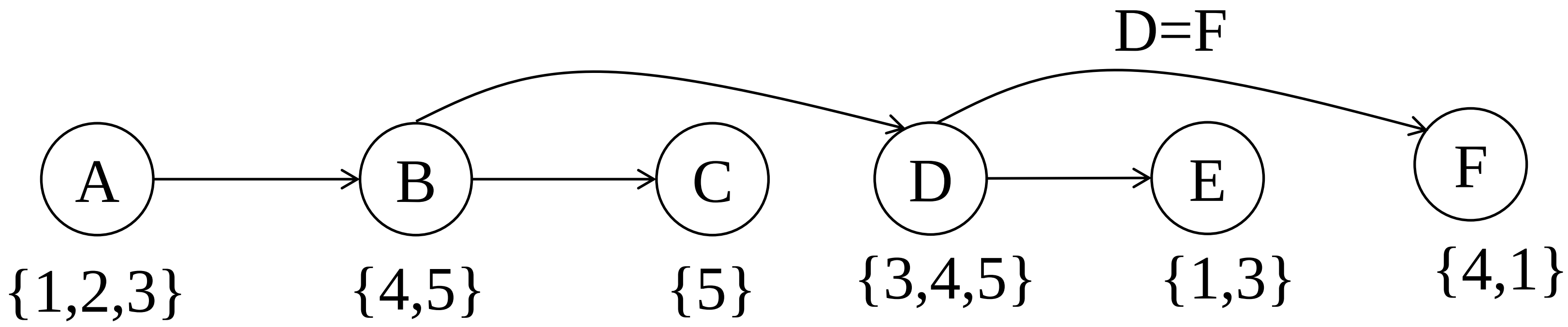


الگوریتمی برای CSPهای دارای ساختار درختی

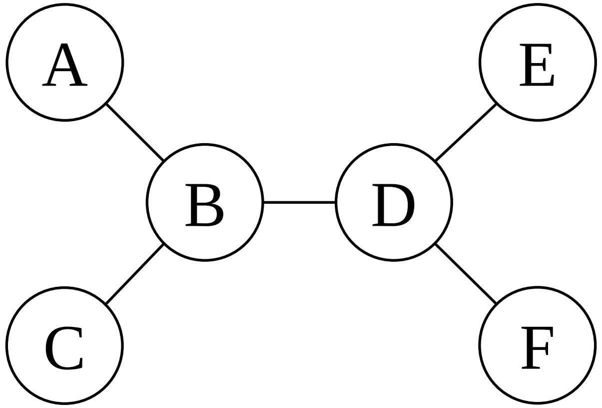
- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)



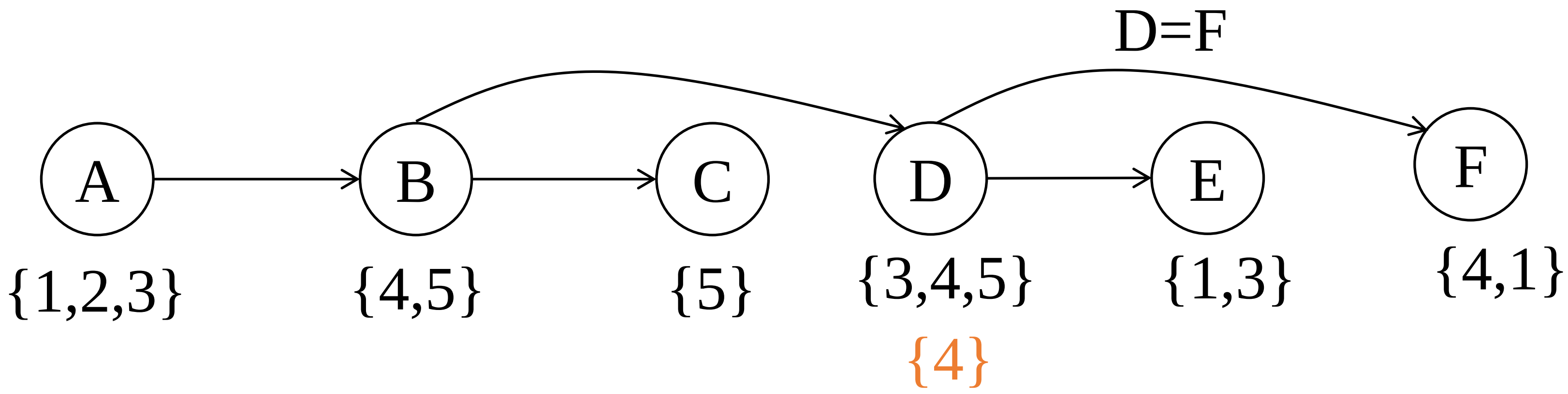
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.



الگوریتمی برای CSPهای دارای ساختار درختی

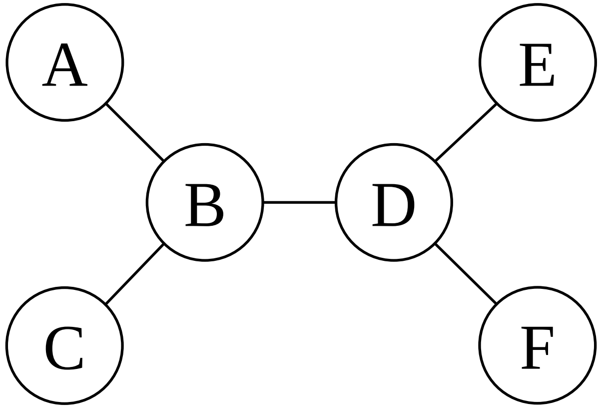


- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.

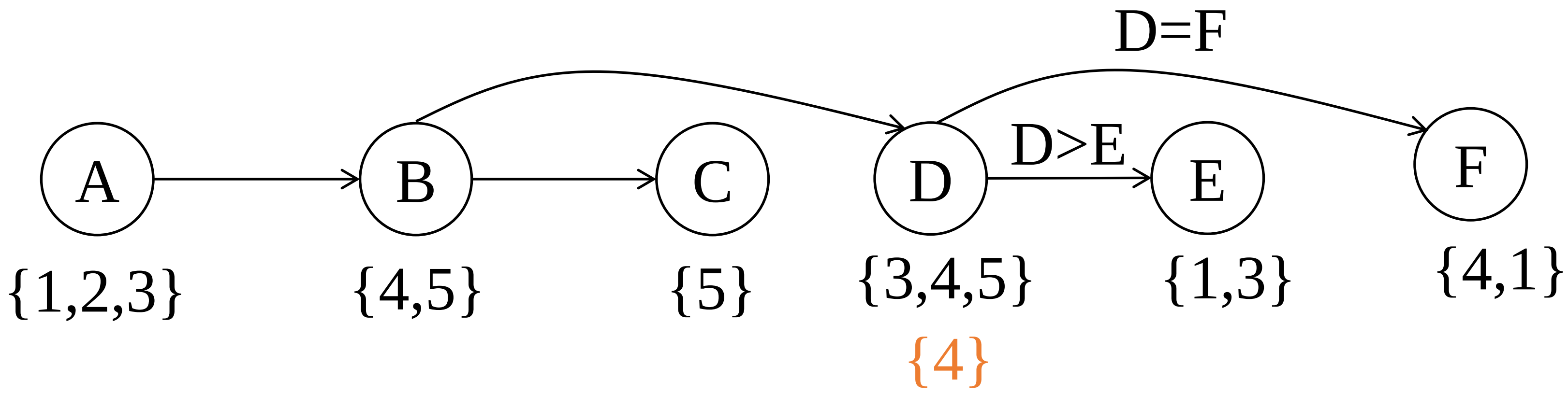


الگوریتمی برای CSPهای دارای ساختار درختی

- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)

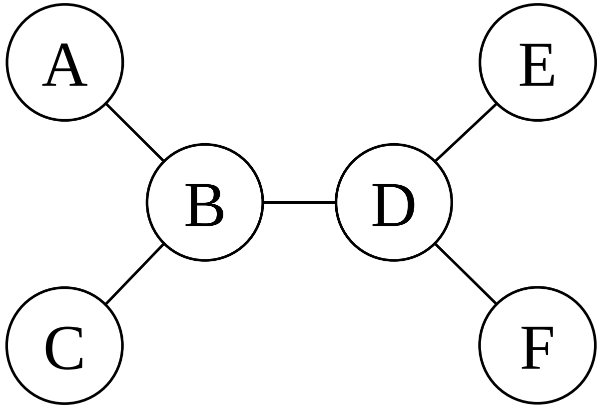


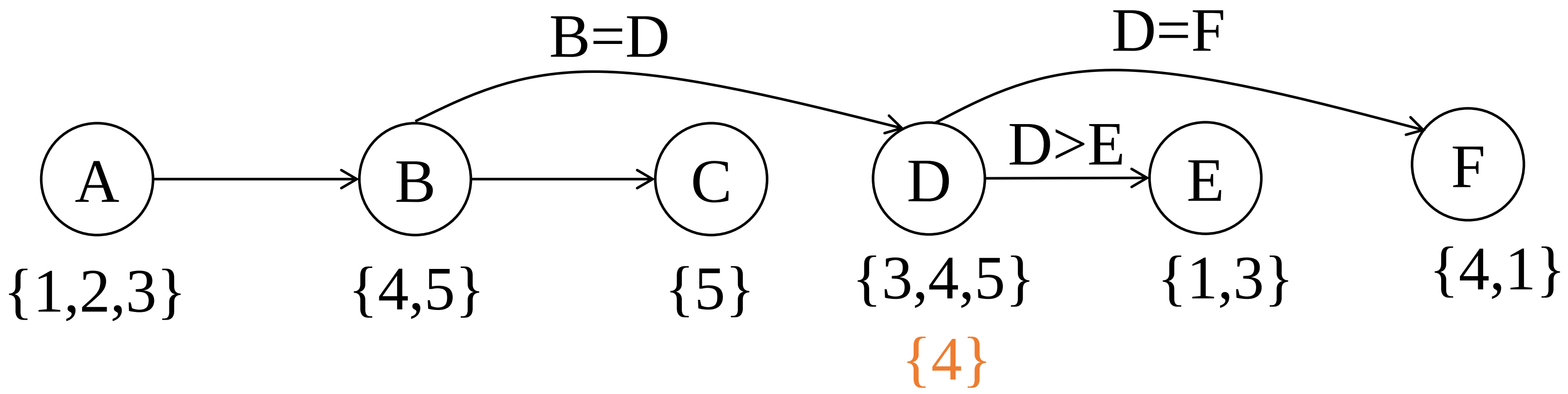
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.



الگوریتمی برای CSPهای دارای ساختار درختی

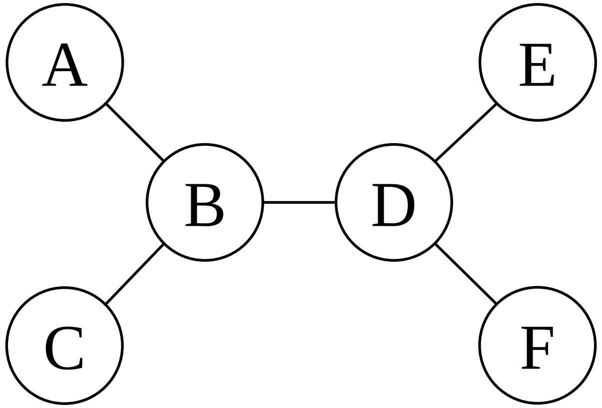
- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.



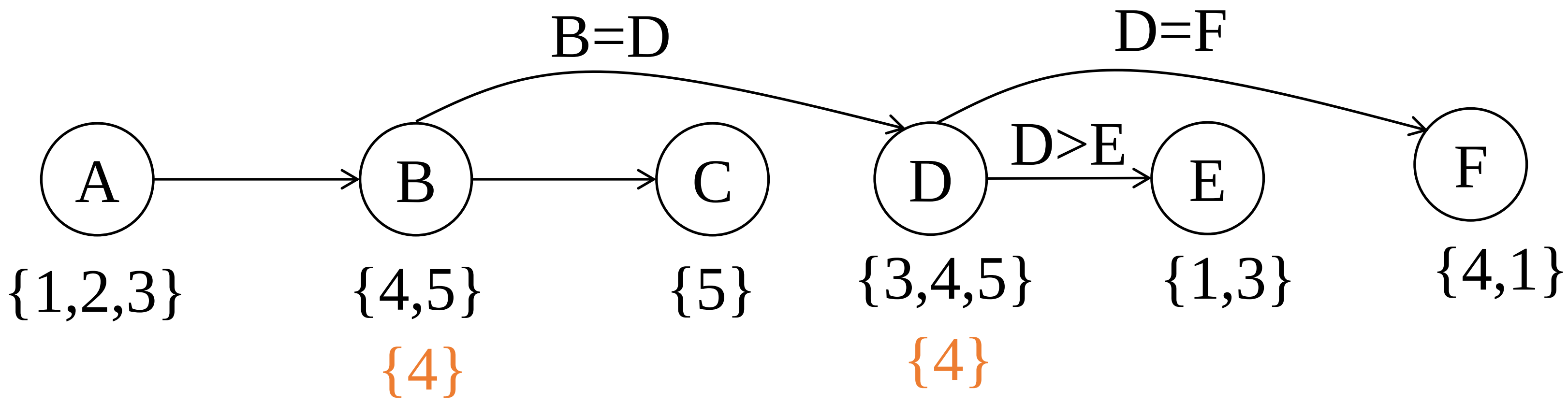


الگوریتمی برای CSPهای دارای ساختار درختی

- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)

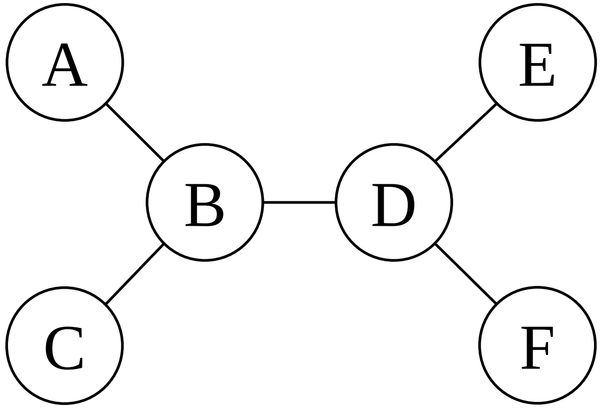


- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.

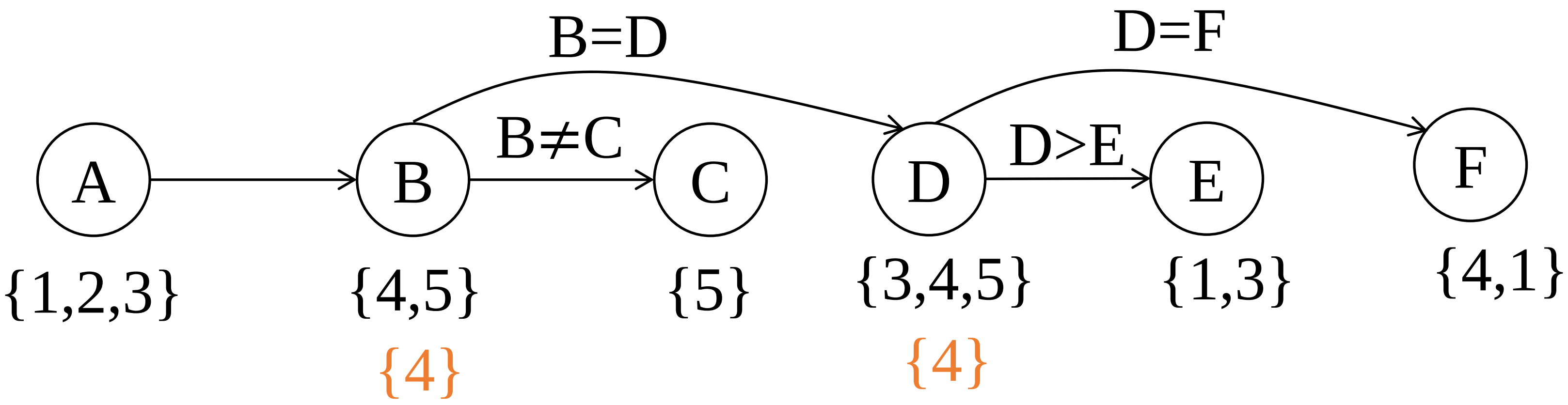


الگوریتمی برای CSPهای دارای ساختار درختی

- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)

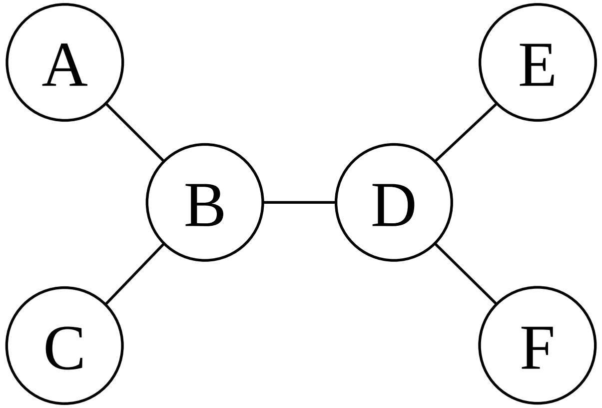


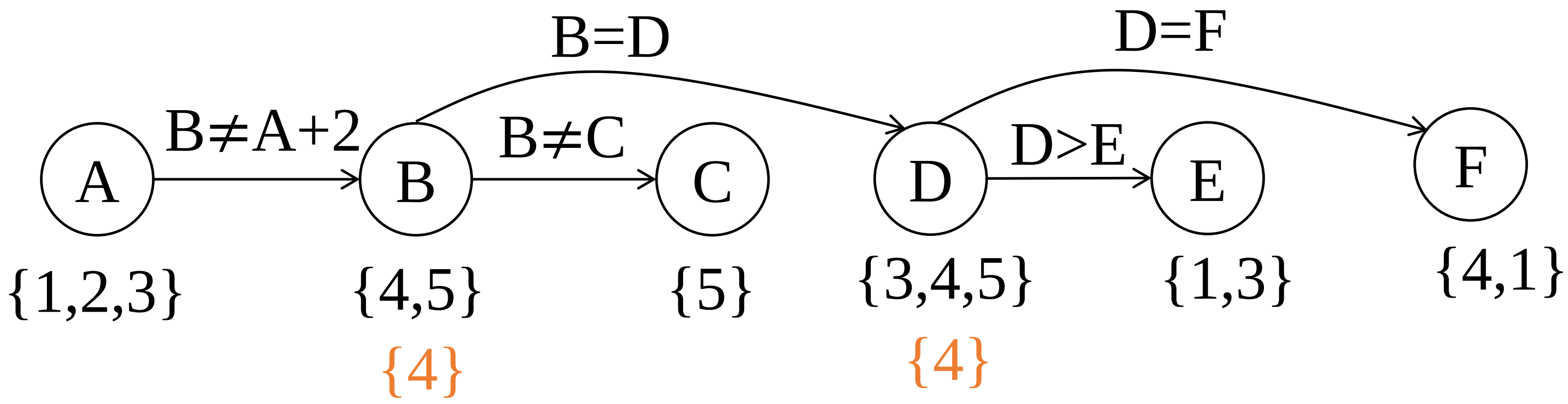
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.



الگوریتمی برای CSPهای دارای ساختار درختی

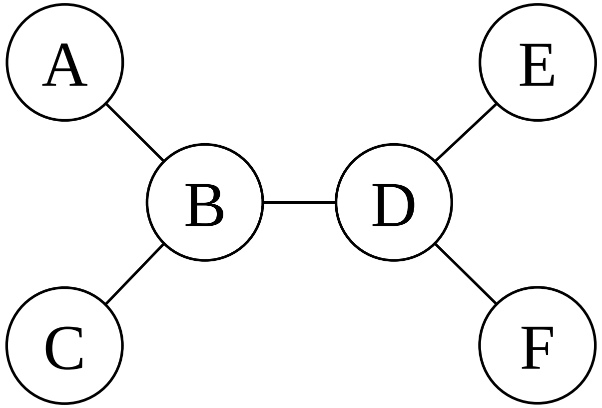
- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.



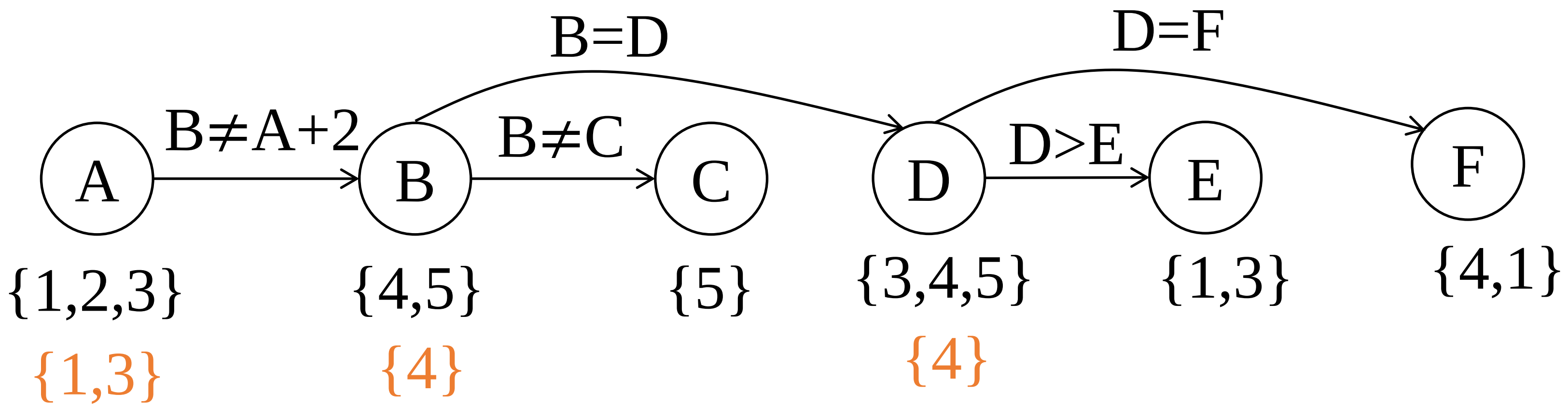


الگوریتمی برای CSPهای دارای ساختار درختی

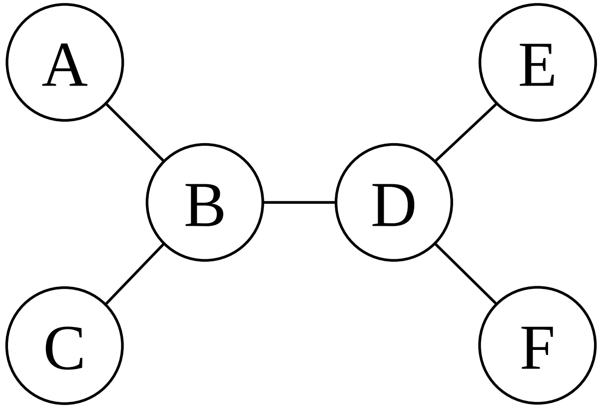
- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)



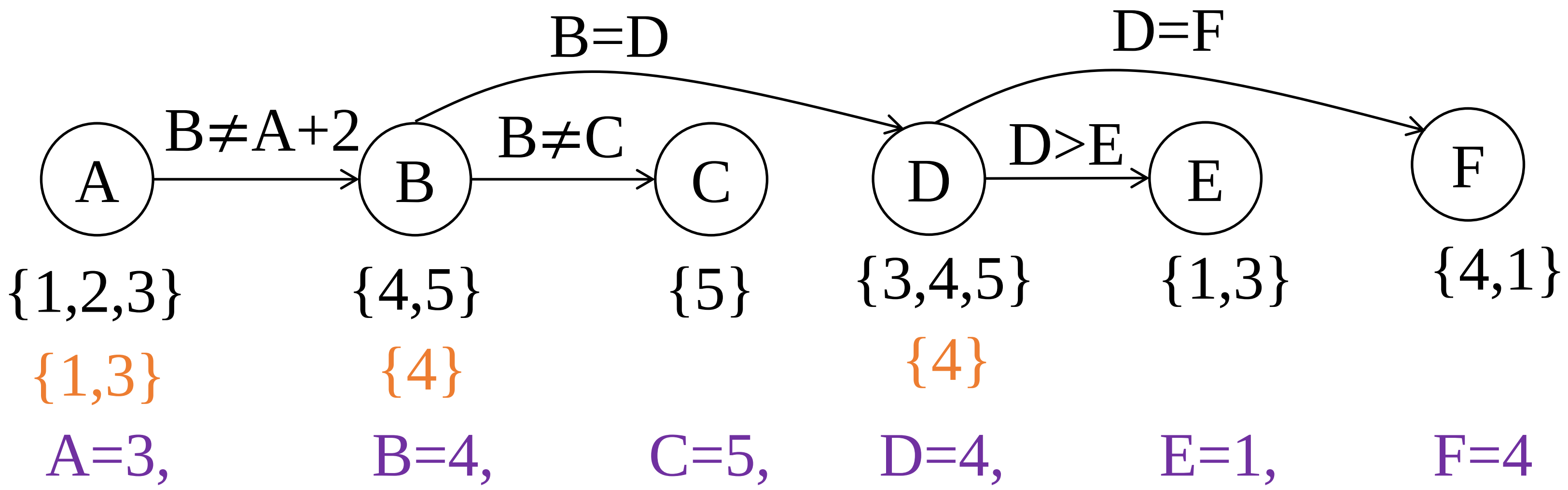
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.



الگوریتمی برای CSPهای دارای ساختار درختی



- یک متغیر را به‌عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگ‌ها به گونه‌ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.
  - یادآوری: مرتب‌سازی توپولوژیکی (Topological sort)
- بر روی هر یک از کمان‌های والد به فرزند عمل سازگاری کمان را انجام بده.
- برای هر متغیر مقداری سازگار با والدش انتخاب کن.



الگوریتمی برای CSPهای دارای ساختار درختی

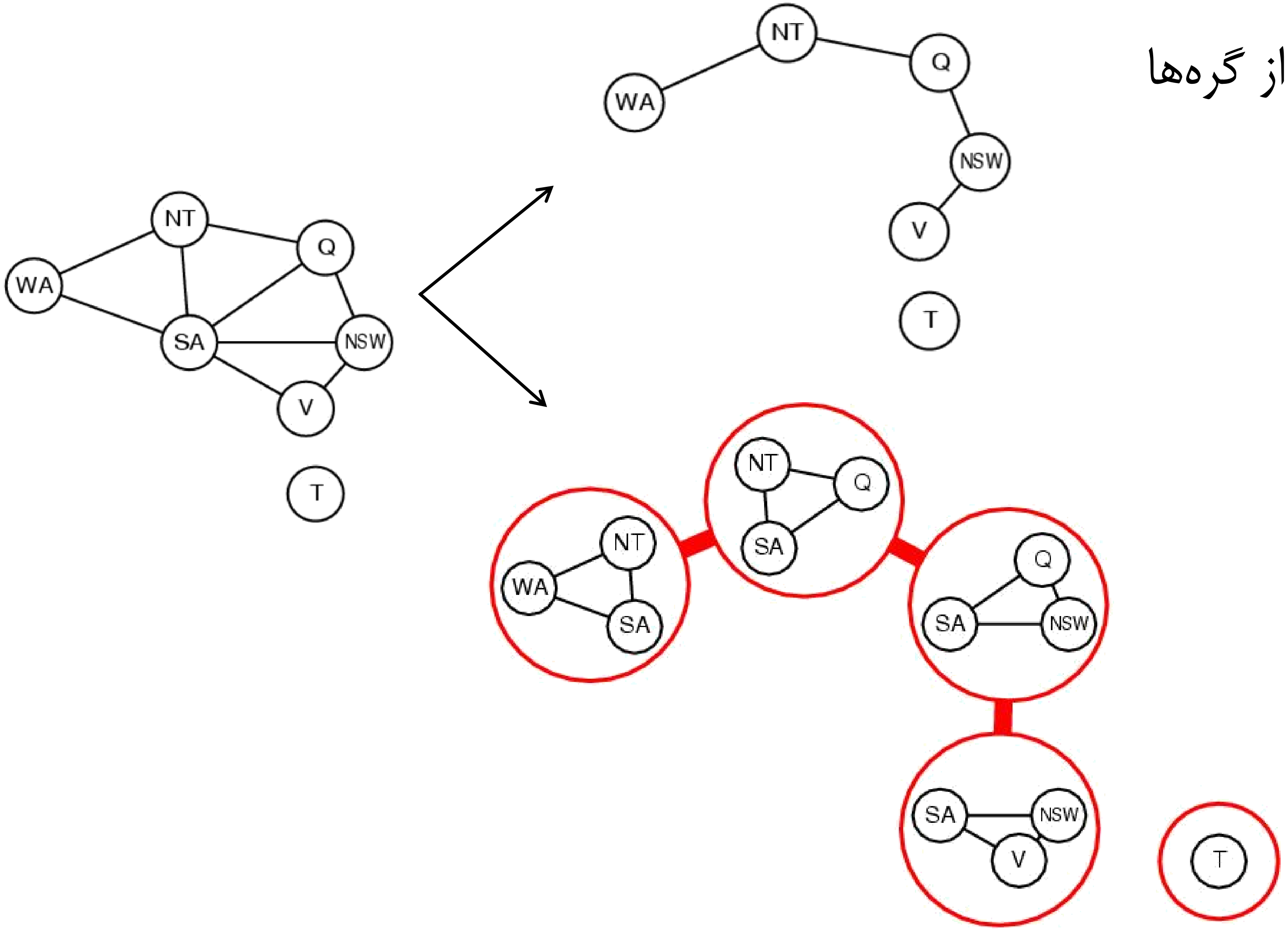
function TREE-CSP-SOLVER(*csp*) **returns** a solution, or failure
  inputs: *csp*, a CSP with components X, D, C

  $n \leftarrow$ number of variables in X
  $assignment \leftarrow$ an empty assignment
  $root \leftarrow$ any variable in X
  $X \leftarrow$ TOPOLOGICALSORT(X, $root$)
  for j = n **down to** 2 **do**
O((n-1)d²) MAKE-ARC-CONSISTENT(PARENT(X_j), X_j)
    if it cannot be made consistent **then return** *failure*
  for i = 1 **to** n **do**
O(nd) $assignment[X_i] \leftarrow$ any consistent value from D_i
    if there is no consistent value **then return** *failure*
  return *assignment*

تبدیل گراف محدودیت به حالت درختی

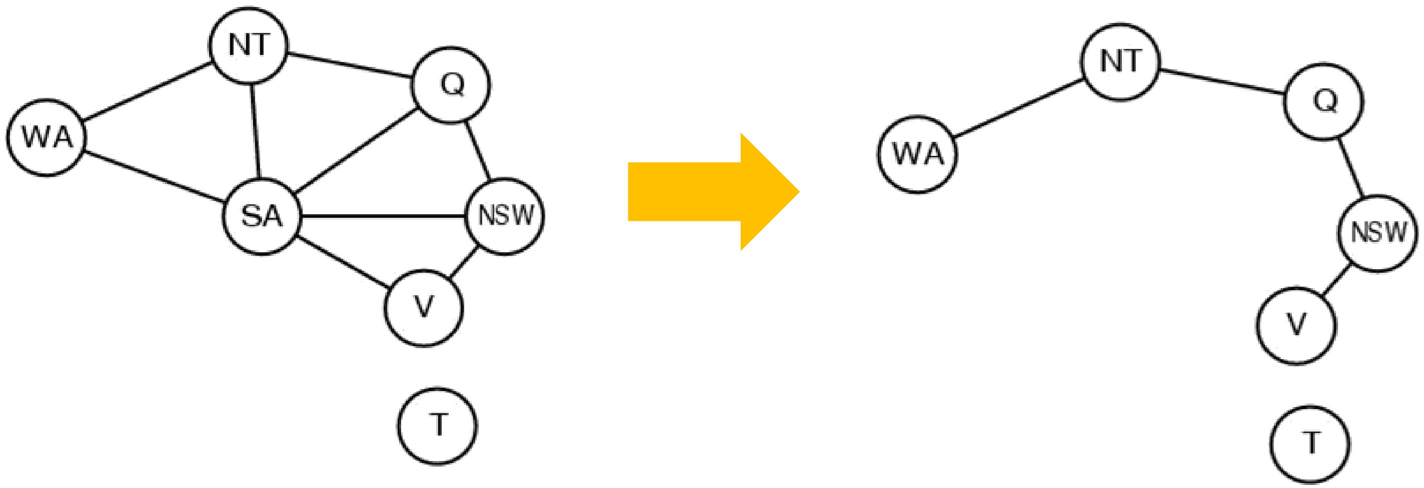
- راه‌کار اول: حذف یک سری از گره‌ها

- راه‌کار دوم: تجزیه درختی



راه‌کار اول: حذف یک سری از گره‌ها

- زیر مجموعه‌ای از متغیرها مانند S را پیدا کن به‌گونه‌ای که با حذف آن‌ها گراف محدودیت به یک درخت تبدیل گردد. (به این زیرمجموعه cycle cutset گویند.)
- برای هر انتساب سازگار ممکن به متغیرهای S که تمام محدودیت‌های S را ارضا می‌کند
  - مقادیر ناسازگار از دامنه‌های متغیرهای باقی‌مانده را حذف کن
  - اگر CSP باقی‌مانده دارای یک راه‌حل باشد آن راه‌حل را به همراه انتساب S برگردان.



راه‌کار اول: حذف یک سری از گره‌ها ...

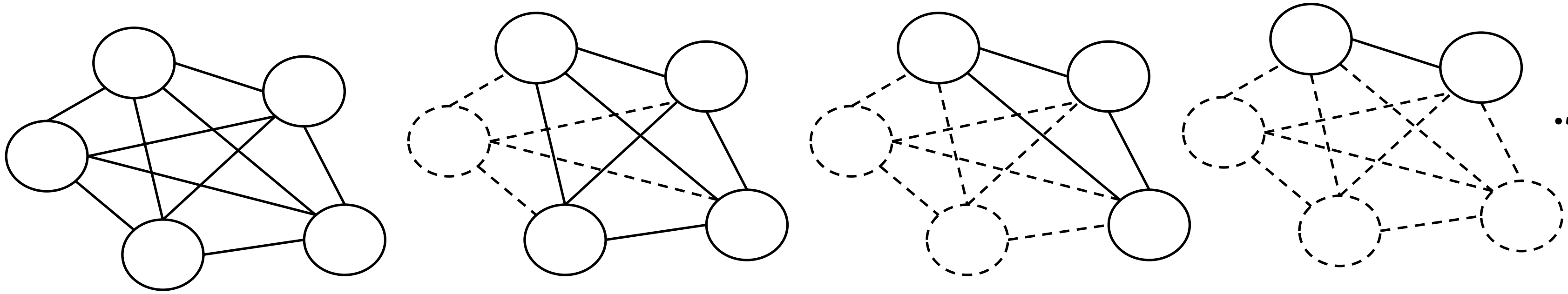
- پیچیدگی زمانی؟
  - سایز $c = $ cycle cutset
  - سایز حداکثر دامنه $= d$
  - تعداد متغیرها $= n$

$$O(d^c.(n-c)d^2)$$

تعداد حالات ممکن مجموعه cycle cuset

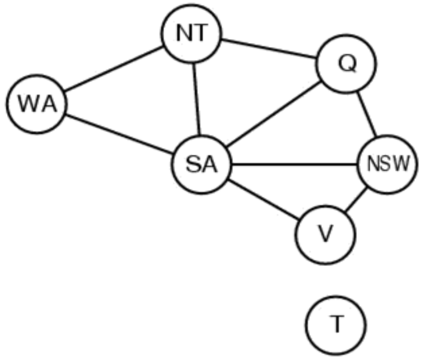
حل درخت با سایز n-c

- برای c کوچک سریع است
- c می‌تواند برابر با n-2 باشد.

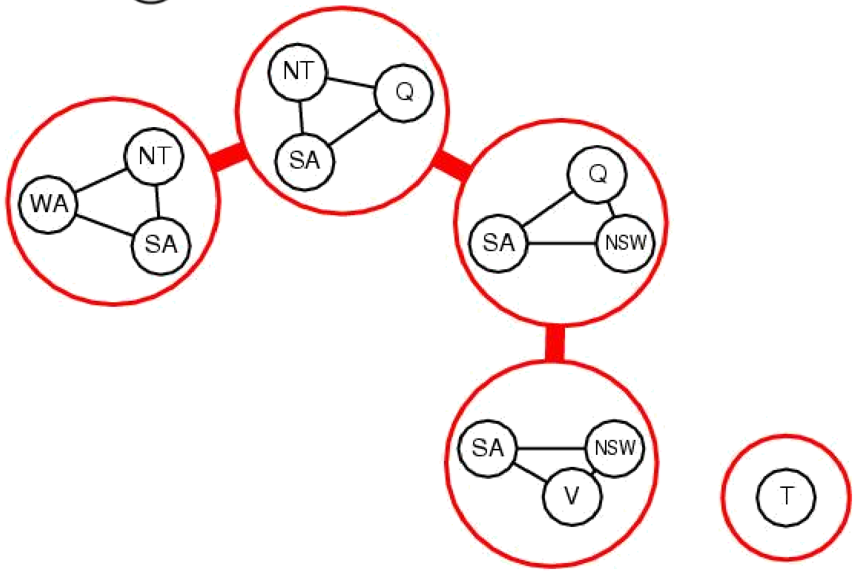


راه‌کار دوم: تجزیه درختی

- تجزیه‌ی درختی گراف محدودیت اصلی به مجموعه‌ای از زیرمسائل همبند.
  - هر زیرمسئله به‌طور مستقل حل می‌شود و سپس راه‌حل‌های به‌دست آمده با هم ترکیب می‌شوند.



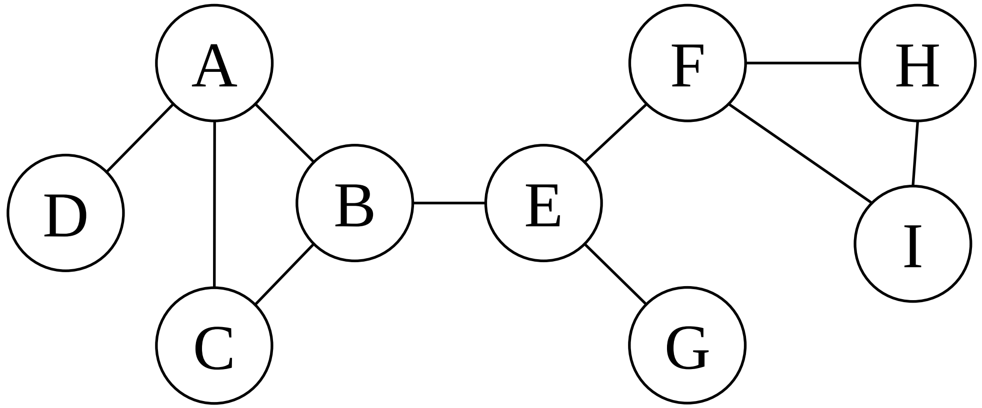
- شرایط تجزیه درختی:
  - هر متغیر در مسأله اصلی باید حداقل در یک زیرمسأله ظاهر شود.
  - اگر دو متغیر به وسیله محدودیتی در مسأله اصلی متصل شده باشند، آن‌گاه آن دو متغیر با هم (به همراه محدودیت) باید حداقل در یک زیر مسأله ظاهر شوند.
  - اگر متغیری در دو زیر مسأله در درخت ظاهر شده باشد، آن متغیر باید در تمامی زیر مسائلی که در مسیر اتصال آن دو زیرمسئله قرار دارند نیز آورده شود.



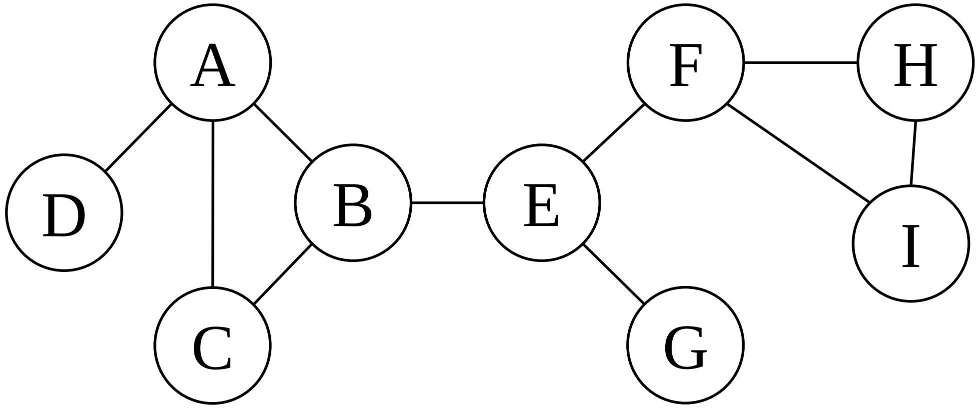
راه‌کار دوم: تجزیه درختی ...

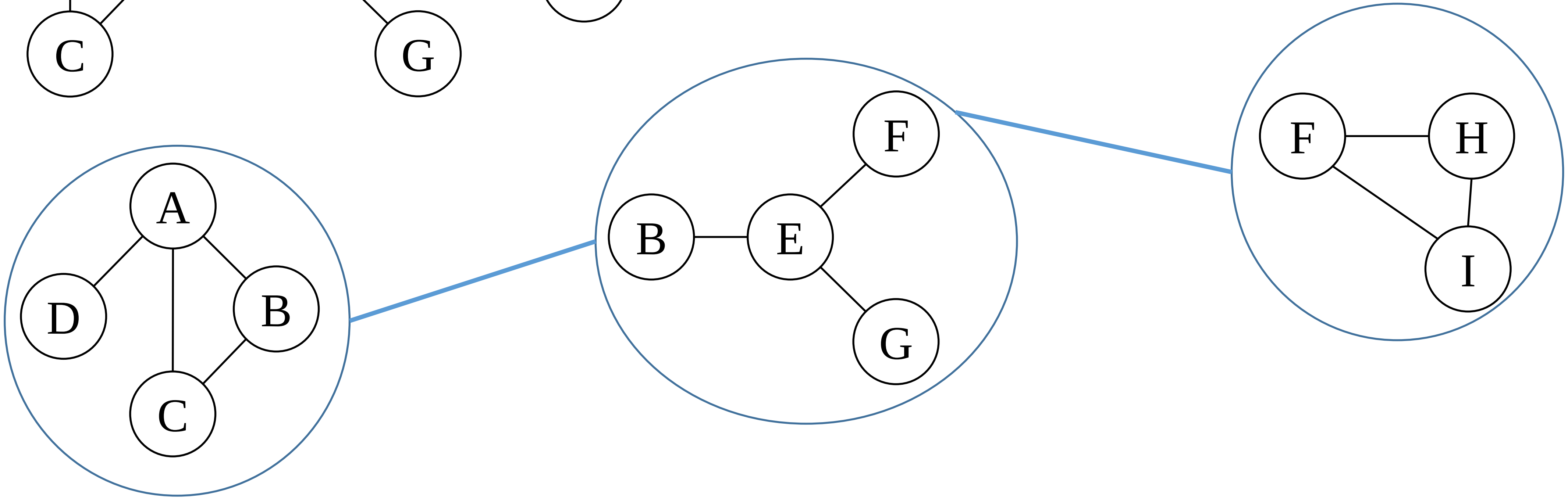
- حل تجزیه درختی:
  - هر زیرمسئله به صورت مجزا حل می‌شود.
    - اگر هر یک از زیر مسائل دارای راه حلی نباشد مسئله اصلی نیز هیچ راه‌حلی ندارد.
  - هر زیر مسئله را به صورت یک متغیر بزرگ (mega-variable) در نظر می‌گیریم که دامنه آن مجموعه تمام راه‌حل‌های ممکن آن زیر مسئله می‌باشد.
  - سپس محدودیت‌های میان زیرمسائل را با استفاده از الگوریتم کارای درخت حل می‌کنیم.
    - این محدودیت‌ها در حقیقت یکسان بودن مقادیر انتساب داده شده به متغیرهای مشترک میان متغیرهای بزرگ است.

راه‌کار دوم: تجزیه درختی ...

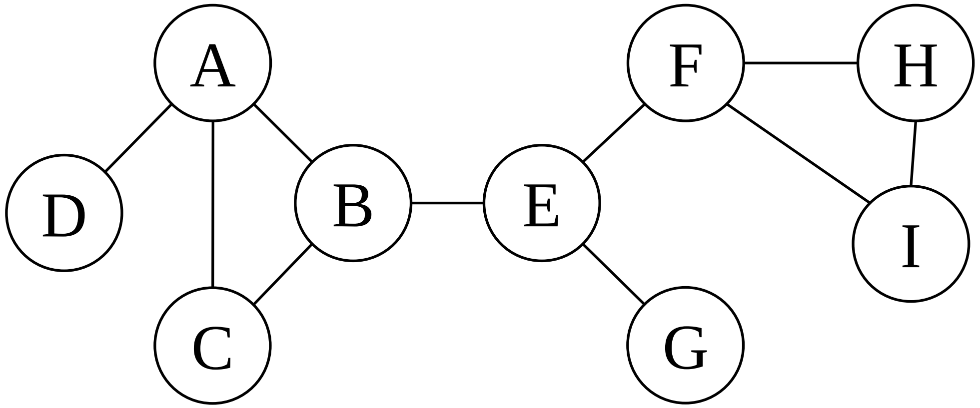


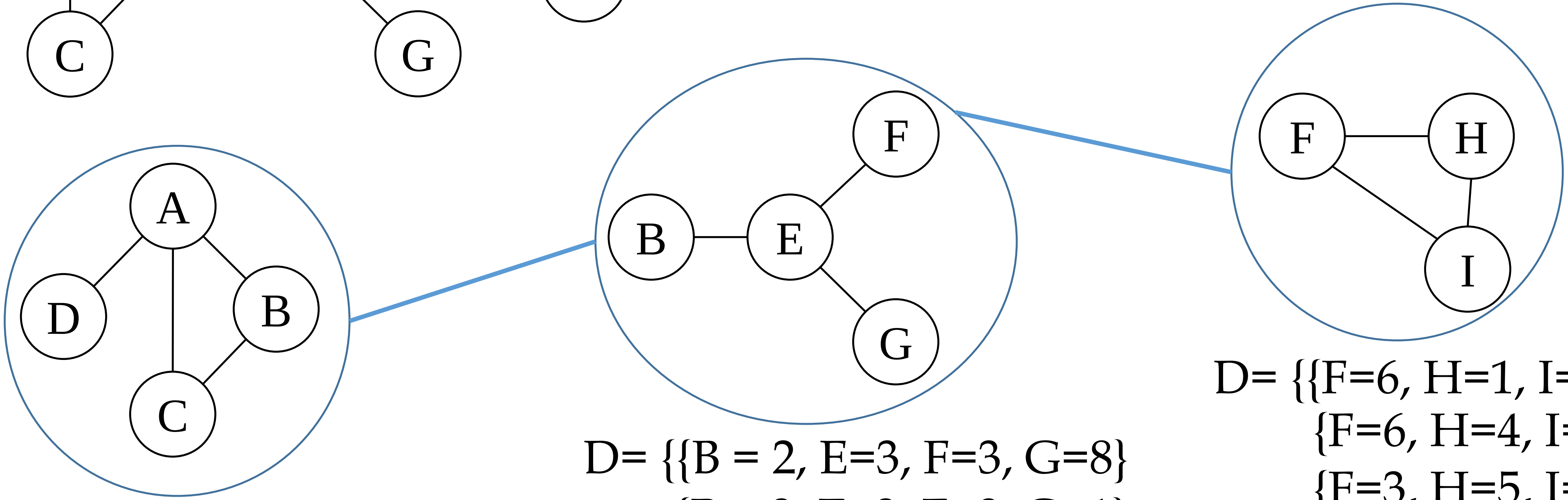
راه‌کار دوم: تجزیه درختی ...





راه‌کار دوم: تجزیه درختی ...



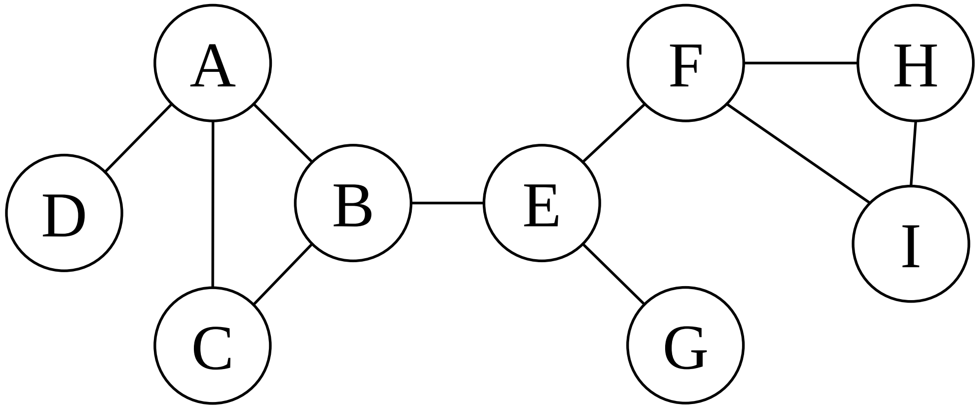


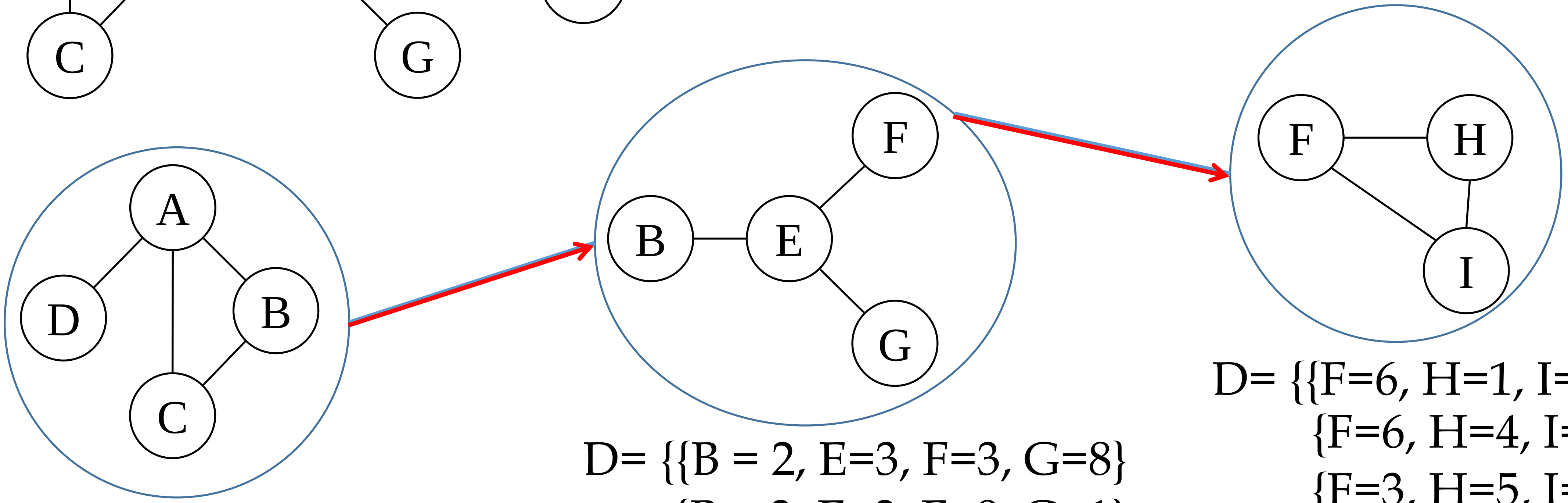
D= {{A = 2, B=1, C=3, D=1}
{A = 3, B=2, C=5, D=4}}

D= {{B = 2, E=3, F=3, G=8}
{B = 3, E=3, F=9, G=1}
{B = 2, E=4, F=6, G=7}}

D= {{F=6, H=1, I=2}
{F=6, H=4, I=1}
{F=3, H=5, I=7}}
{F=7, H=6, I=4}}

راه‌کار دوم: تجزیه درختی ...



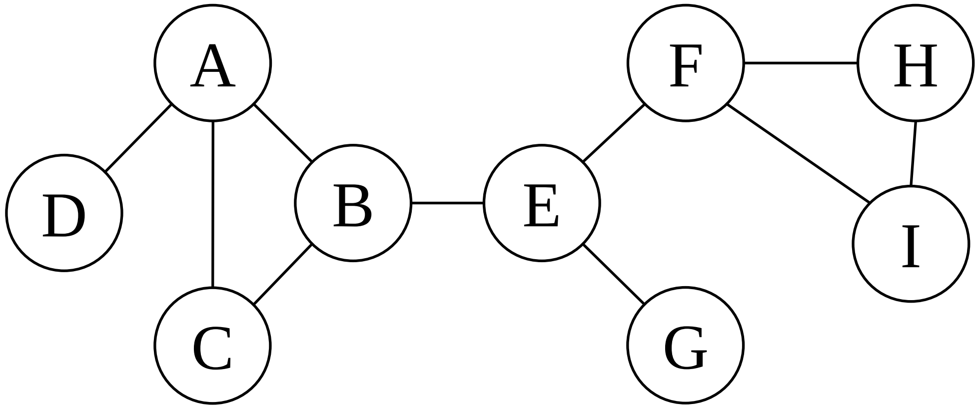


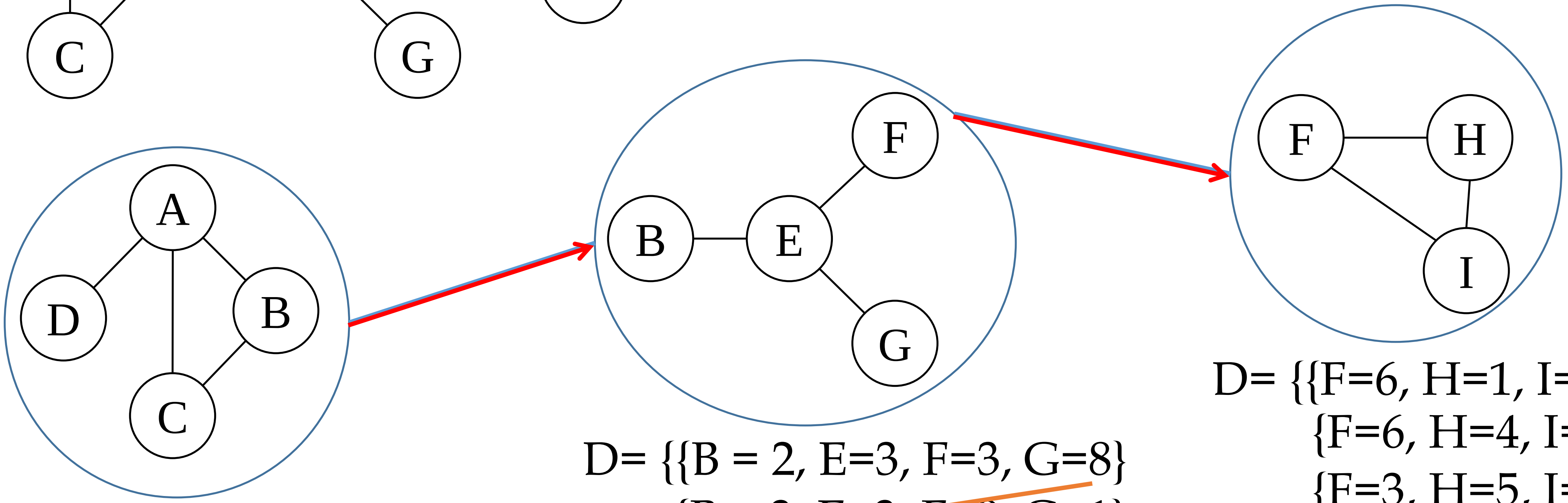
D= {{A = 2, B=1, C=3, D=1}
{A = 3, B=2, C=5, D=4}}

D= {{B = 2, E=3, F=3, G=8}
{B = 3, E=3, F=9, G=1}
{B = 2, E=4, F=6, G=7}}

D= {{F=6, H=1, I=2}
{F=6, H=4, I=1}
{F=3, H=5, I=7}}
{F=7, H=6, I=4}}

راه‌کار دوم: تجزیه درختی ...



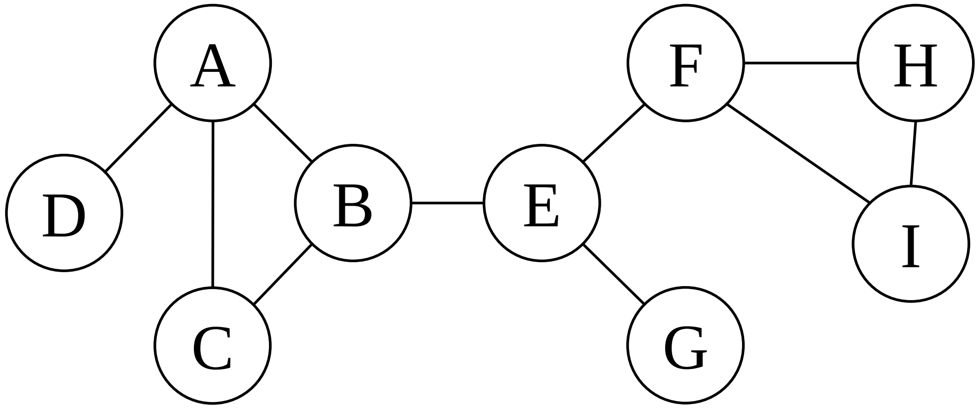


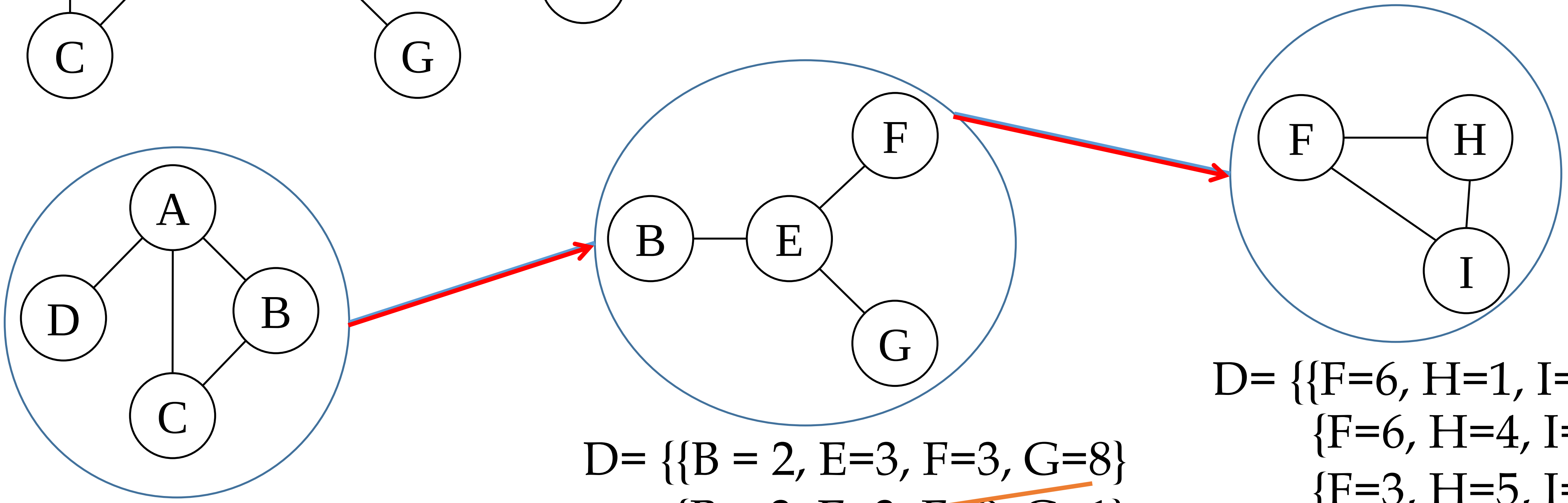
D= {{A = 2, B=1, C=3, D=1}
{A = 3, B=2, C=5, D=4}}

D= {{B = 2, E=3, F=3, G=8}
~~{B = 3, E=3, F=9, G=1}~~
{B = 2, E=4, F=6, G=7}}

D= {{F=6, H=1, I=2}
{F=6, H=4, I=1}
{F=3, H=5, I=7}}
{F=7, H=6, I=4}}

راه‌کار دوم: تجزیه درختی ...



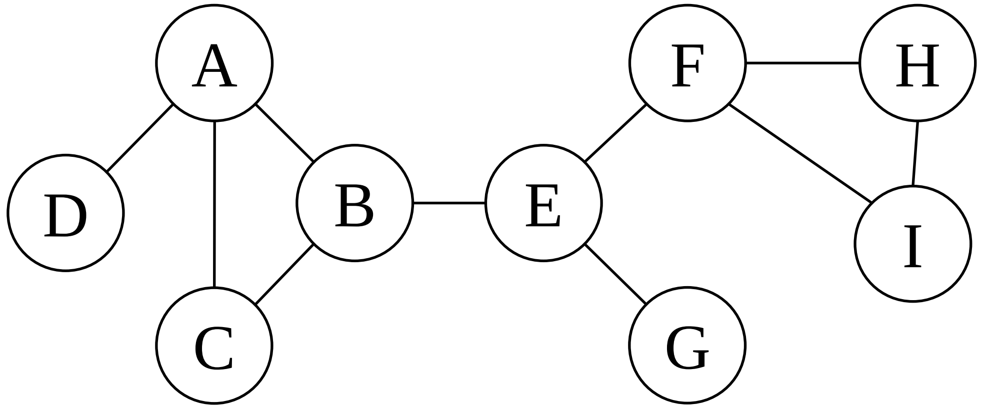


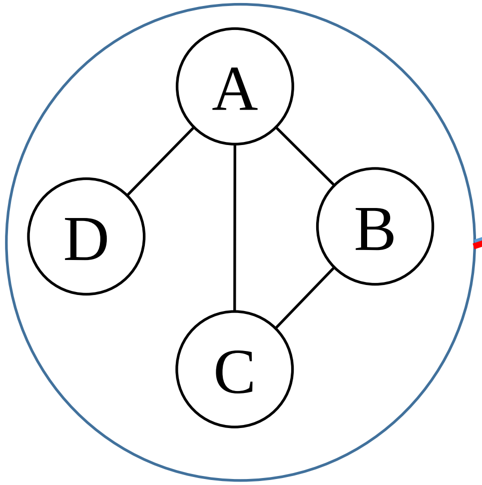
D= {~~{A = 2, B=1, C=3, D=1}~~
{A = 3, B=2, C=5, D=4}}

D= {{B = 2, E=3, F=3, G=8}
~~{B = 3, E=3, F=9, G=1}~~
{B = 2, E=4, F=6, G=7}}

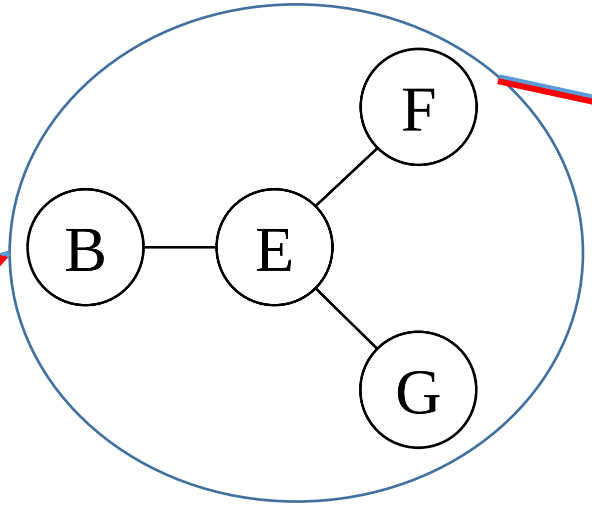
D= {{F=6, H=1, I=2}
{F=6, H=4, I=1}
{F=3, H=5, I=7}}
{F=7, H=6, I=4}}

راه‌کار دوم: تجزیه درختی ...

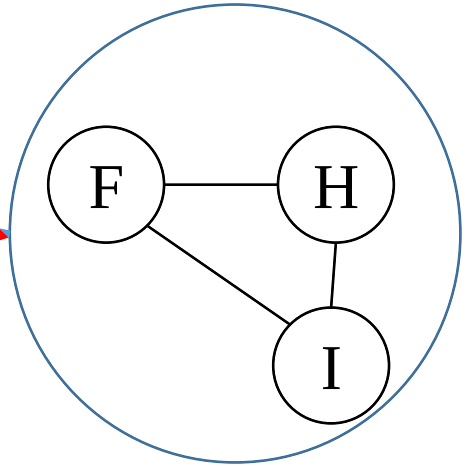




D= {~~{A = 2, B=1, C=3, D=1}~~
{A = 3, B=2, C=5, D=4}}



D= {{B = 2, E=3, F=3, G=8}
~~{B = 3, E=3, F=9, G=1}~~
{B = 2, E=4, F=6, G=7}}



D= {{F=6, H=1, I=2}
{F=6, H=4, I=1}
{F=3, H=5, I=7}}
{F=7, H=6, I=4}}

{A = 3, B=2, C=5, D=4, B = 2, E=4, F=6, G=7, F=6, H=4, I=1}

راه‌کار دوم: تجزیه درختی ...

- یک گراف محدودیت را می‌توان به روش‌های گوناگون مورد تجزیه‌ی درختی قرار داد.
  - هدف از تجزیه، کوچک کردن زیرمسئله تا حد ممکن است.

- عرض درخت = یک واحد کم‌تر از اندازه بزرگ‌ترین زیرمسئله موجود پس از تجزیه درختی
- عرض درخت در یک گراف محدودیت = w = حداقل عرض درخت تمامی تجزیه‌های درختی
- پیچیدگی زمانی؟ $O(nd^{w+1})$