Programmation Par Contraintes

Cours 2 - Arc-Consistance et autres amusettes

David Savourey

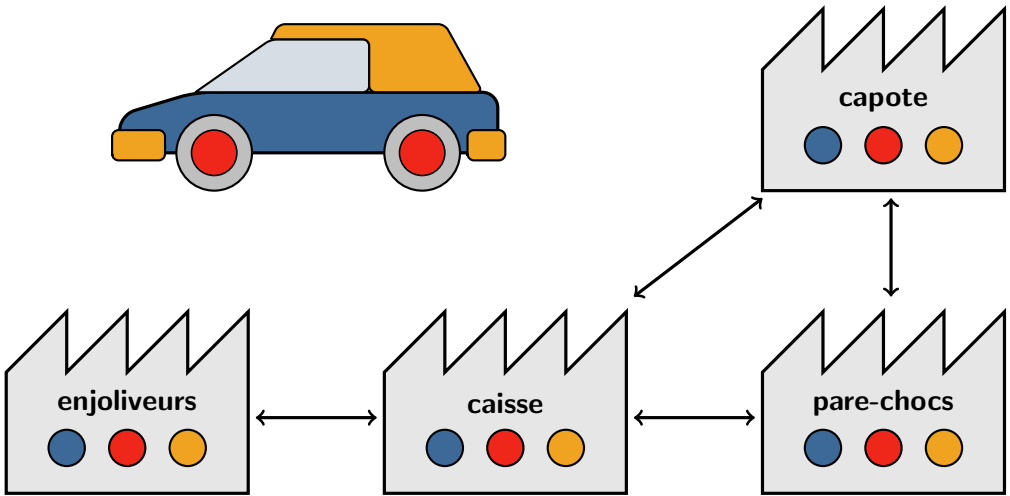
CNRS, École Polytechnique

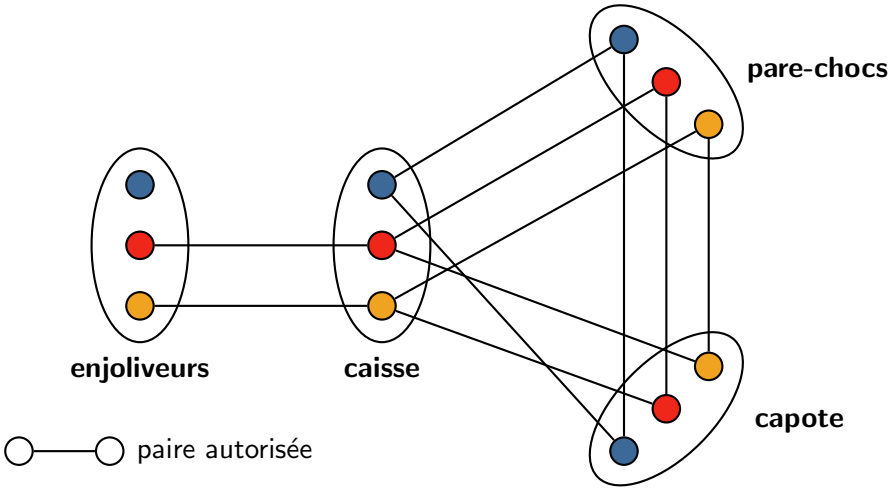
inspiré des cours de Philippe Baptiste, Ruslan Sadykov et de la thèse d'Hadrien Cambazard

Sommaire

Arc-Consistance

Un exemple trivial




pare-chocs
enjoliveurs
caisse
capote
paire autorisée

Résolution naïve : Generate & Test

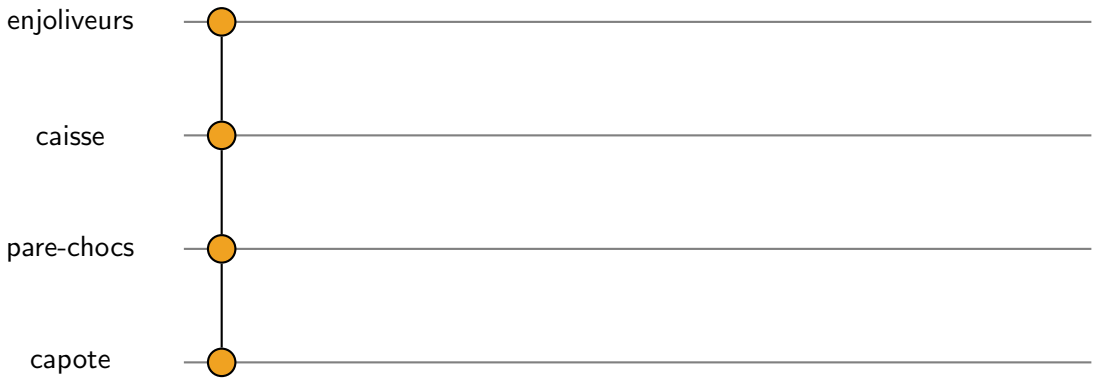
enjoliveurs

caisse

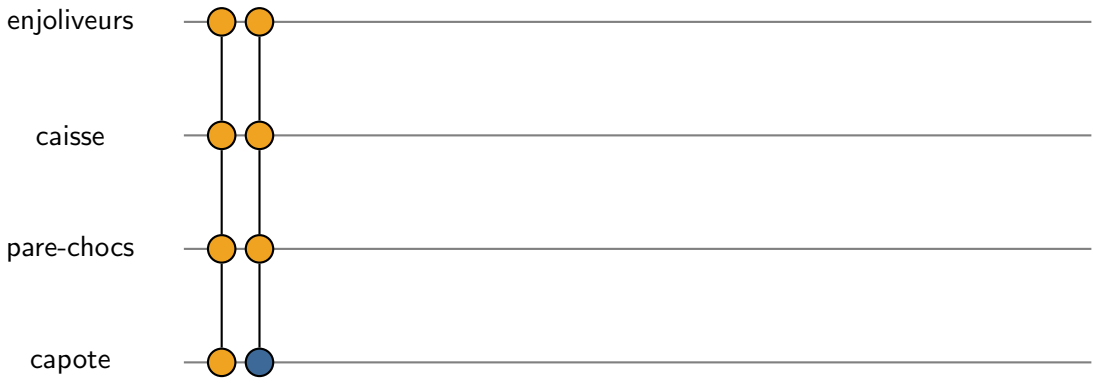
pare-chocs

capote

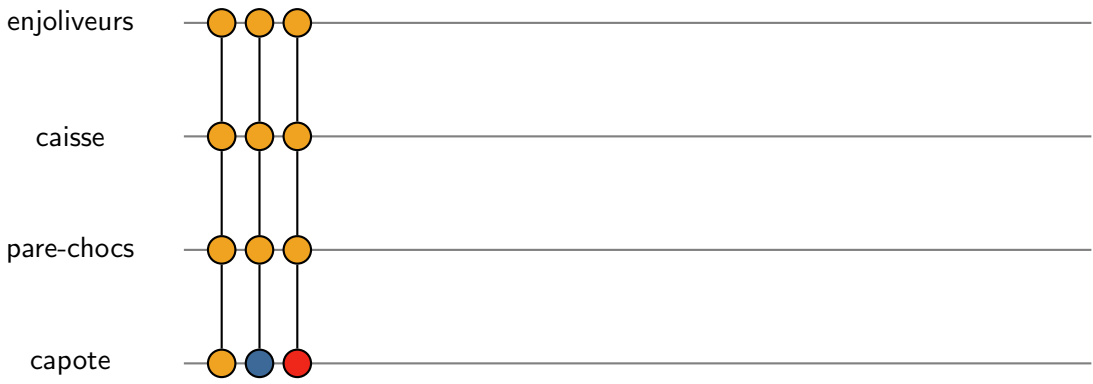
Résolution naïve : Generate & Test



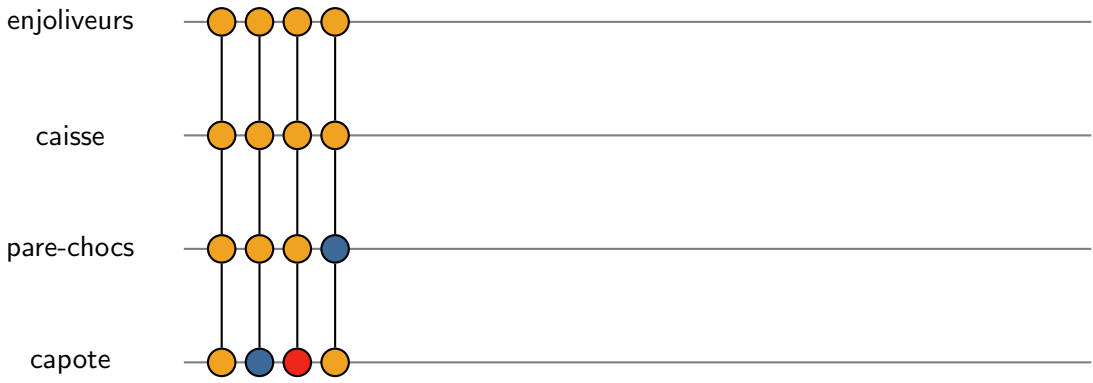
Résolution naïve : Generate & Test

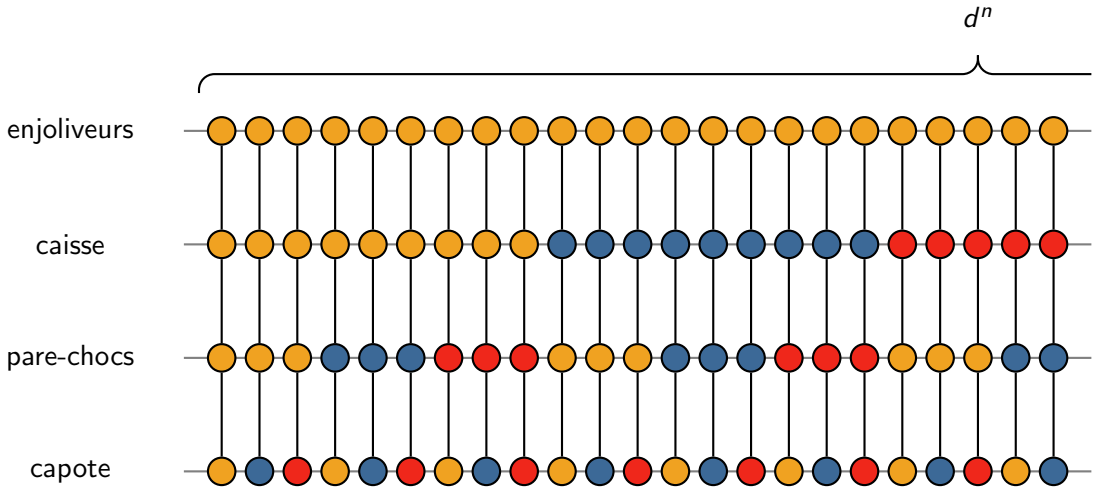


Résolution naïve : Generate & Test



Résolution naïve : Generate & Test



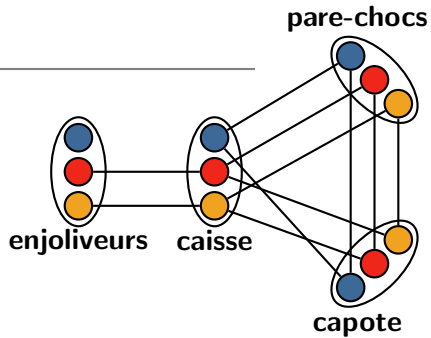

d^n
enjoliveurs
caisse
pare-chocs
capote

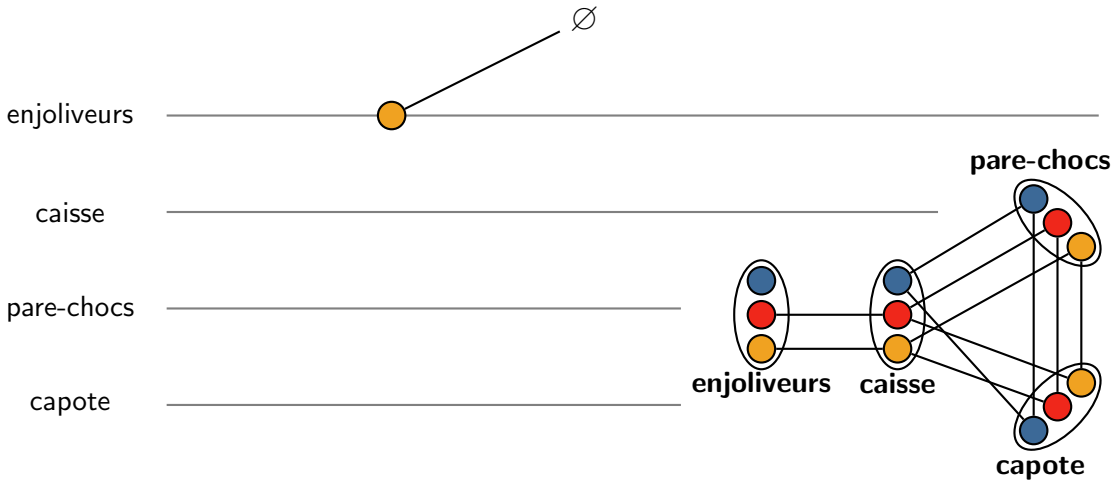
enjoliveurs

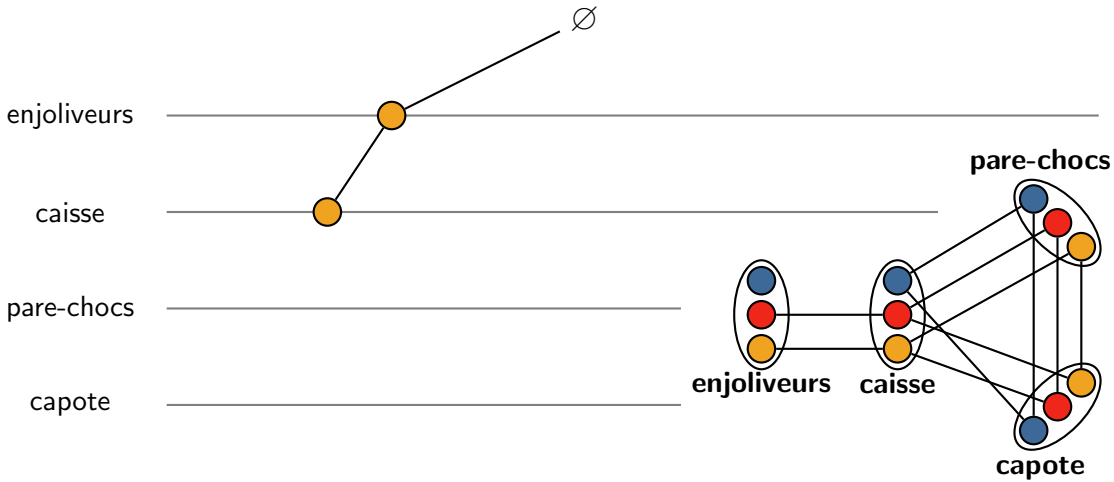
caisse

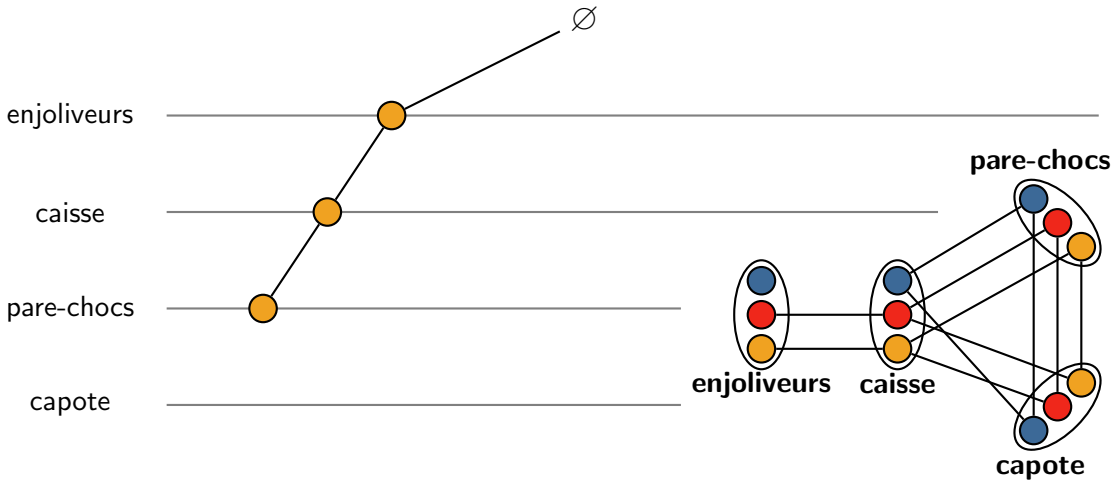
pare-chocs

capote

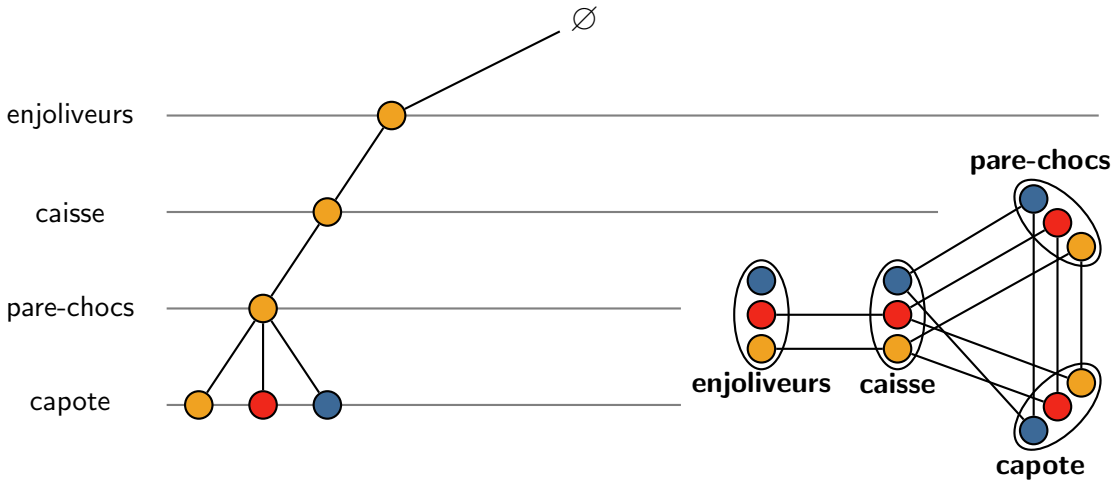


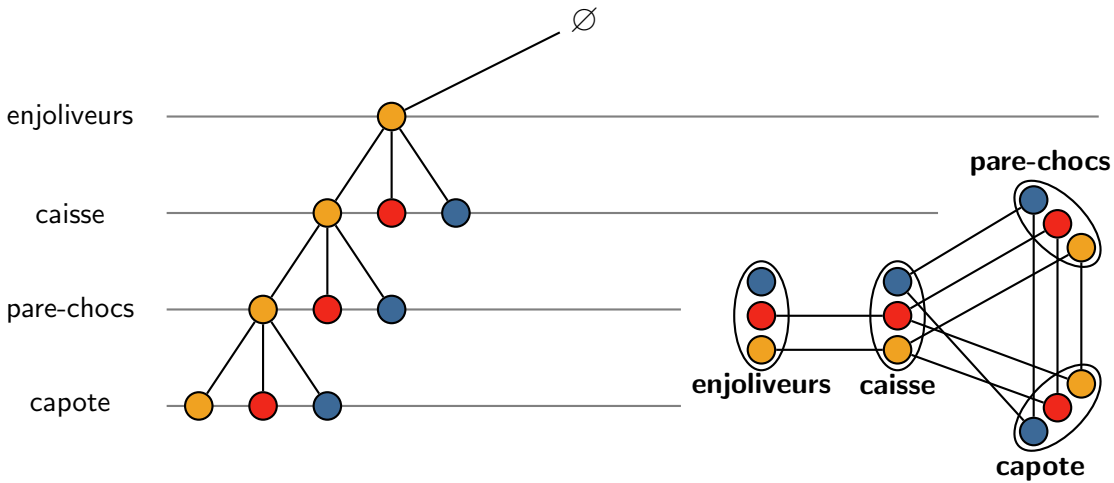

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

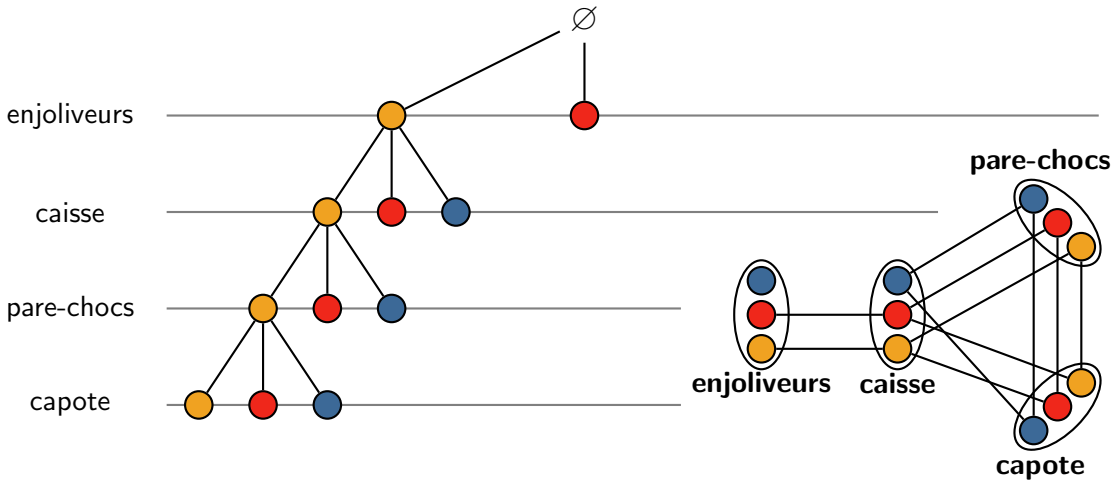

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

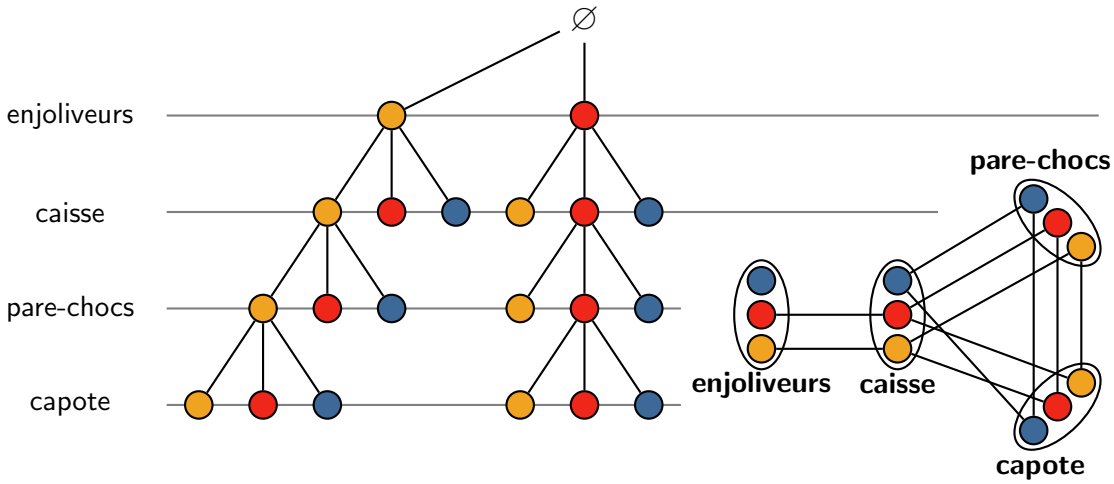
Mieux : Backtrack



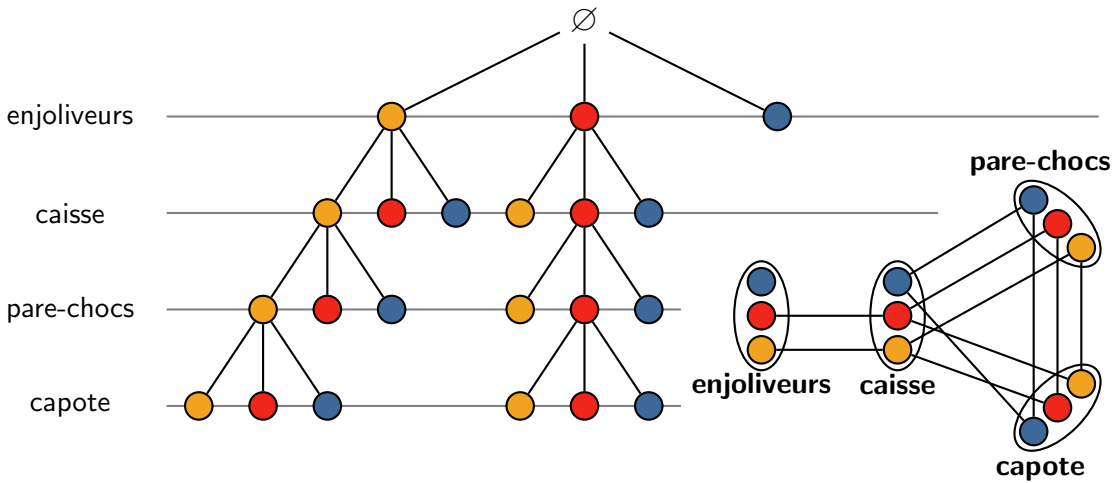

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

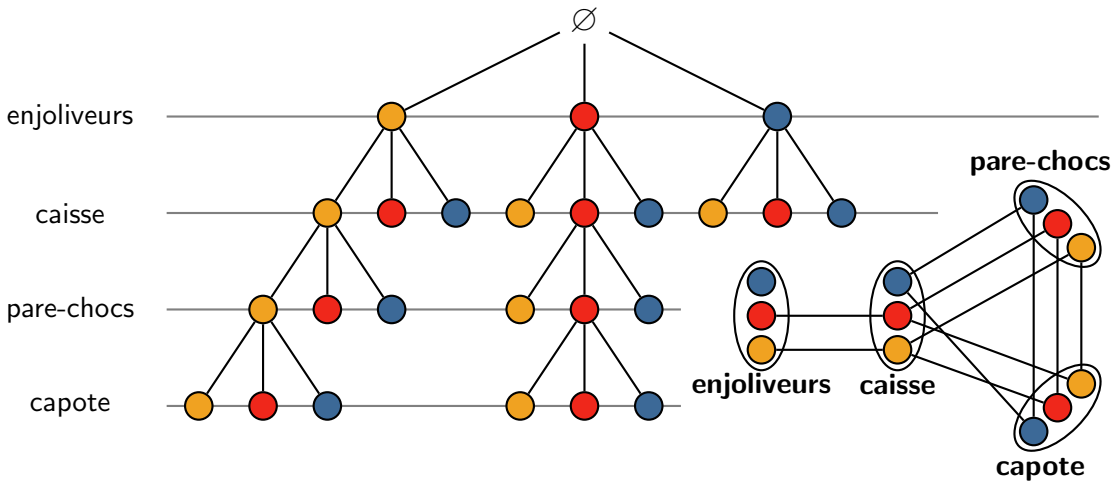

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

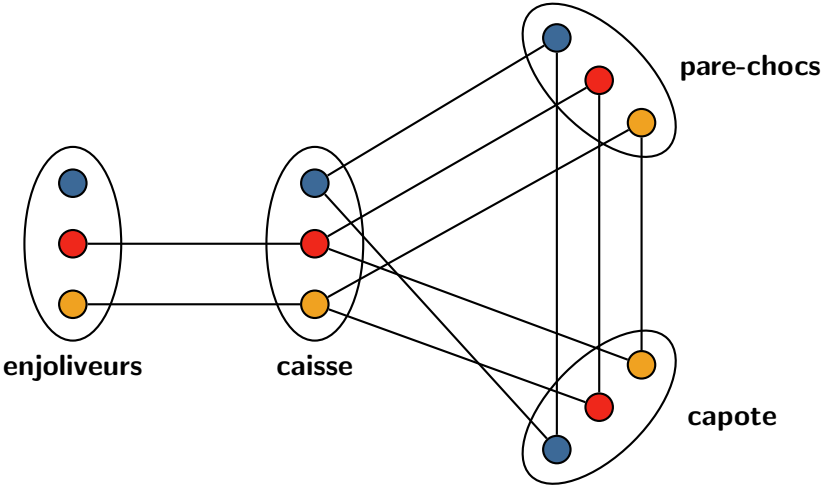
Mieux : Backtrack

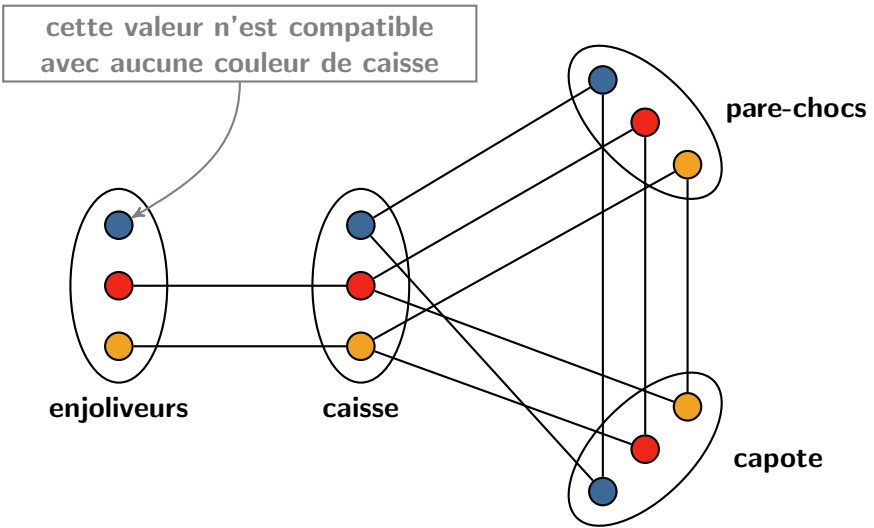


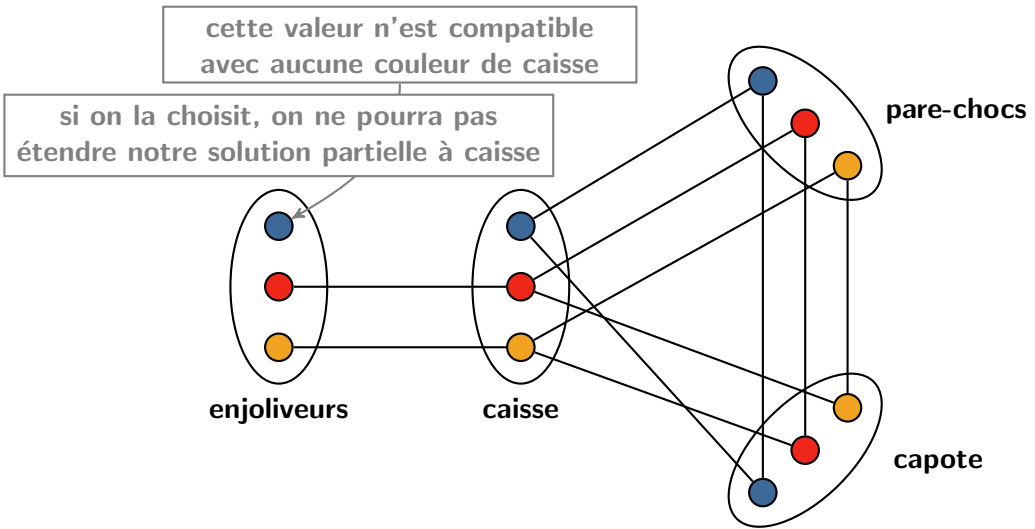
Mieux : Backtrack

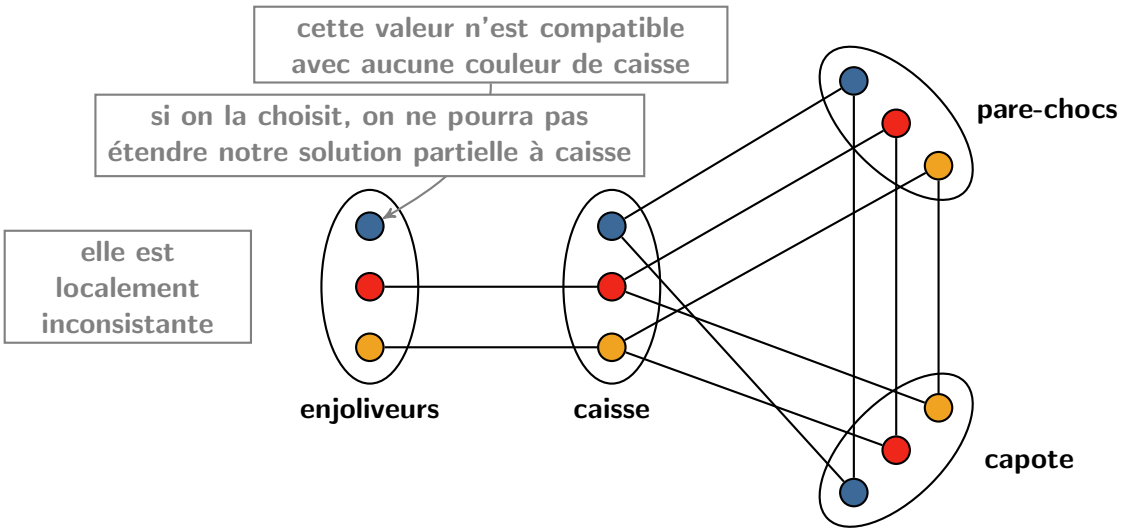


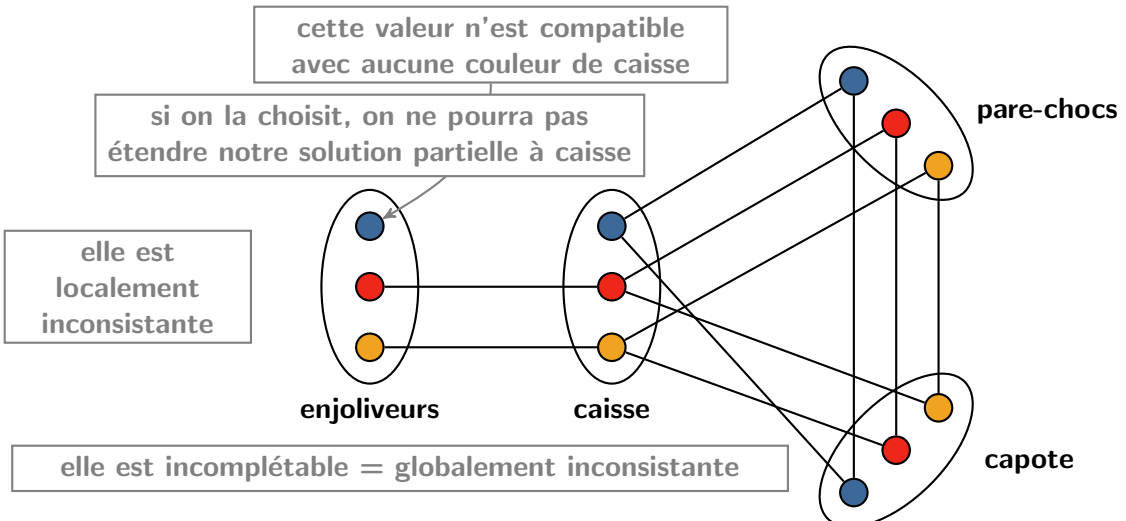

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

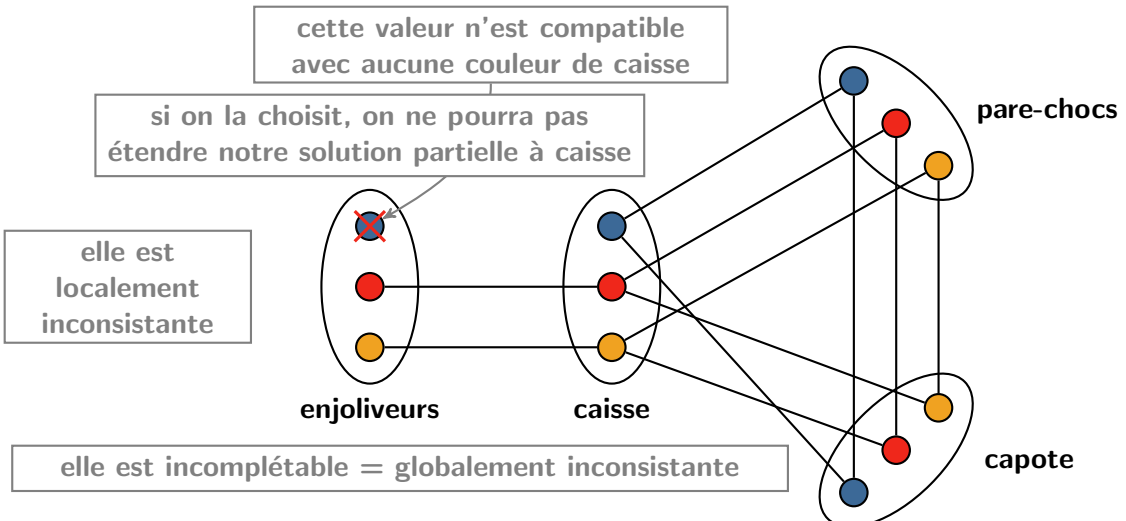

pare-chocs
enjoliveurs
caisse
capote

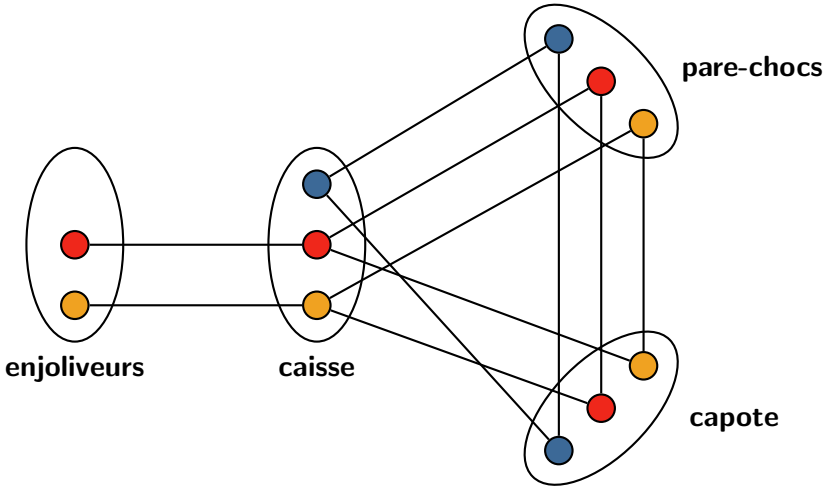

cette valeur n'est compatible
avec aucune couleur de caisse
pare-chocs
enjoliveurs
caisse
capote

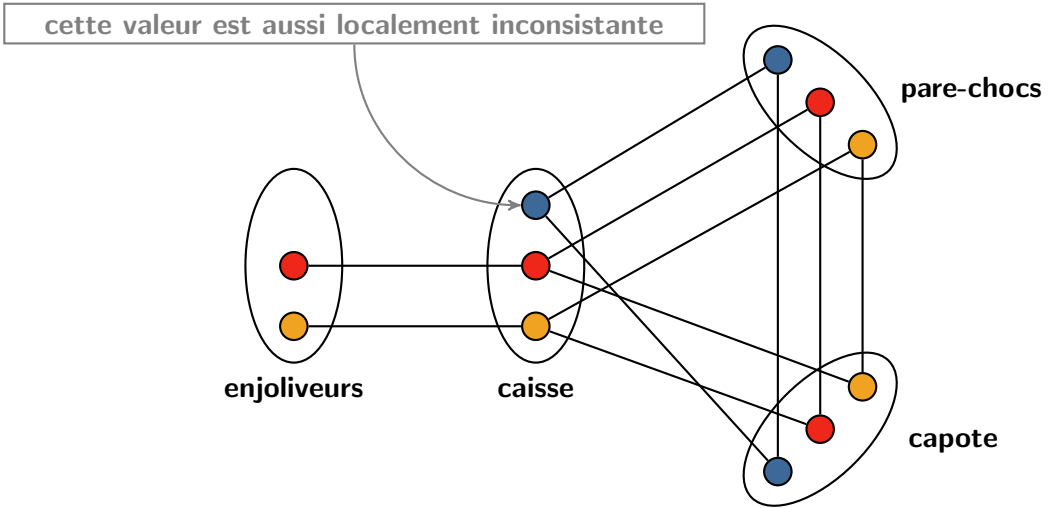

cette valeur n'est compatible
avec aucune couleur de caisse
si on la choisit, on ne pourra pas
étendre notre solution partielle à caisse
pare-chocs
enjoliveurs
caisse
capote

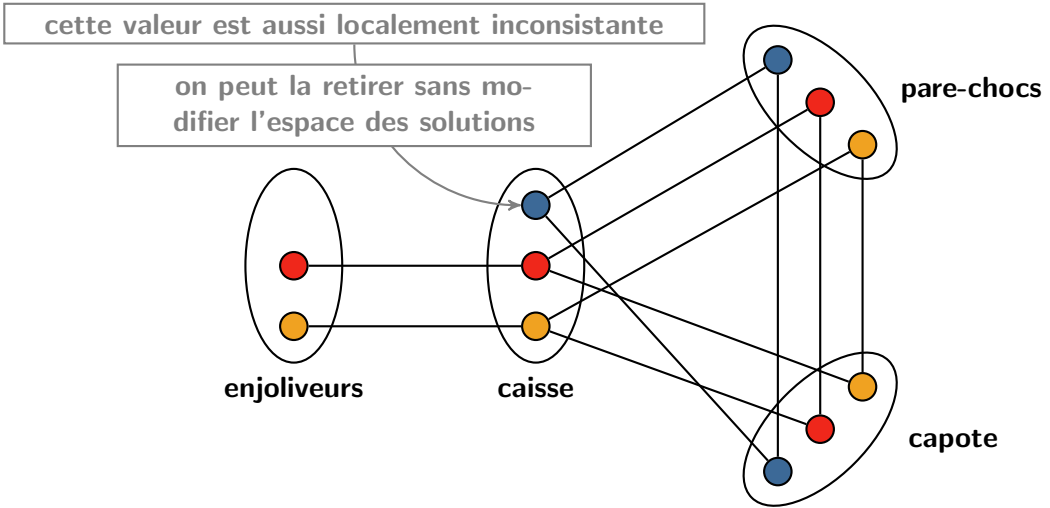

cette valeur n'est compatible avec aucune couleur de caisse
si on la choisit, on ne pourra pas étendre notre solution partielle à caisse
elle est localement inconsistante
pare-chocs
enjoliveurs
caisse
capote

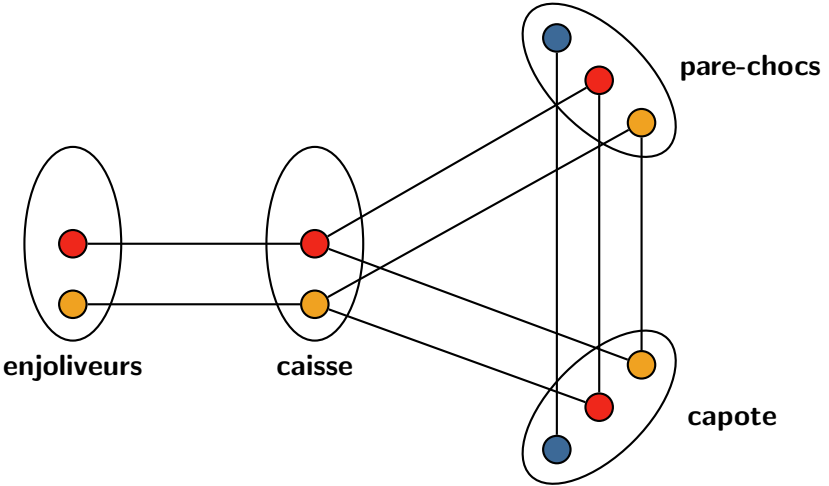

cette valeur n'est compatible avec aucune couleur de caisse
si on la choisit, on ne pourra pas étendre notre solution partielle à caisse
elle est localement inconsistante
pare-chocs
enjoliveurs
caisse
capote
elle est incomplétable = globalement inconsistante

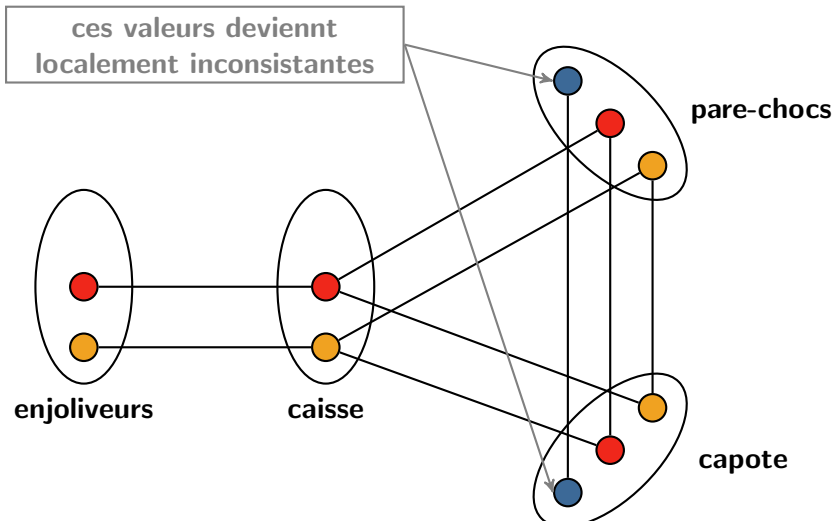

cette valeur n'est compatible avec aucune couleur de caisse
si on la choisit, on ne pourra pas étendre notre solution partielle à caisse
elle est localement inconsistante
pare-chocs
enjoliveurs
caisse
capote
elle est incomplétable = globalement inconsistante

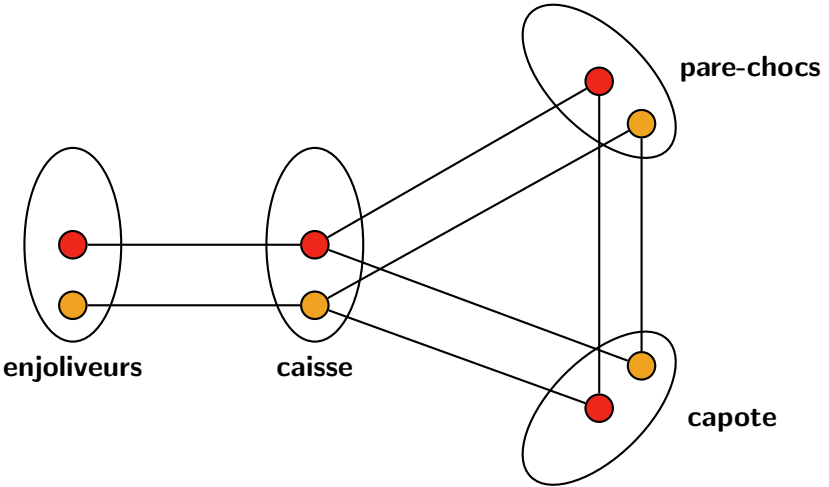

pare-chocs
enjoliveurs
caisse
capote


cette valeur est aussi localement inconsistante
pare-chocs
enjoliveurs
caisse
capote

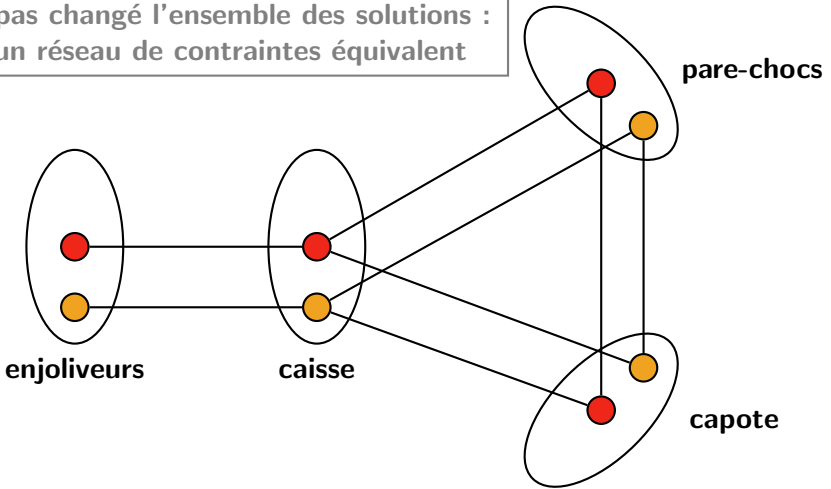

cette valeur est aussi localement inconsistante
on peut la retirer sans modifier l'espace des solutions
pare-chocs
enjoliveurs
caisse
capote


pare-chocs
enjoliveurs
caisse
capote


ces valeurs deviennt
localement inconsistantes
pare-chocs
enjoliveurs
caisse
capote

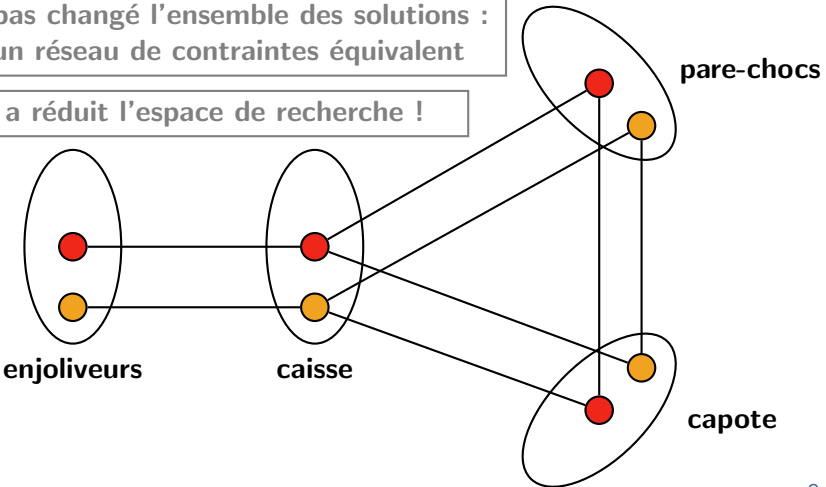

pare-chocs
enjoliveurs
caisse
capote

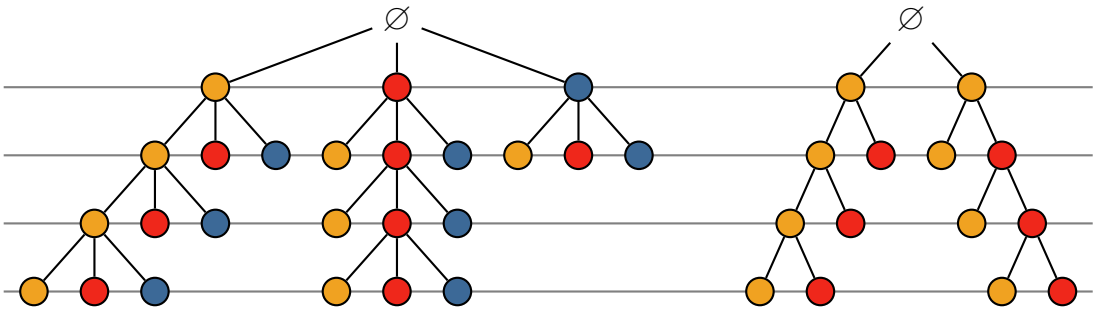
on n'a pas changé l'ensemble des solutions :
on a un réseau de contraintes équivalent

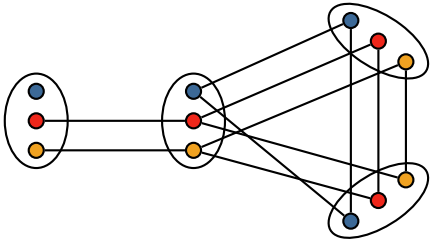


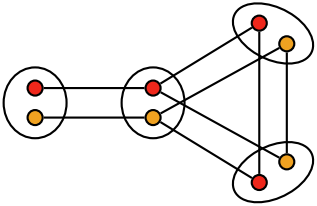
on n'a pas changé l'ensemble des solutions :
on a un réseau de contraintes équivalent

on a réduit l'espace de recherche !









- consistance globale = recherche de solutions
- consistance locale : on résout des sous-problèmes de taille fixée

Attention, dans la suite on considèrera les contraintes de manière directionnelle : une contrainte portant sur les variables x et y générera les deux contraintes $C_{x,y}$ et $C_{y,x}$.

- Nous venons de réaliser la fermeture arc-consistante de notre CSP
- La valeur a de la variable x est arc-consistante ssi elle possède au moins une valeur compatible (support) dans chaque domaine voisin.
- $\langle x, a \rangle$ est arc-consistante ssi $\forall C_{x,y} \exists b \in D_y$ tq. $C_{x,y}(a, b)$
- $\langle x, a \rangle$ est arc-inconsistante ssi $\exists C_{x,y} \forall b \in D_y$ tq. $\neg C_{x,y}(a, b)$
- une contrainte est arc-consistante si toutes les valeurs de ses variables sont arc-consistantes.
- un CSP est un arc-consistant si toutes ses contraintes sont arc-consistantes.

Intérêt de l'arc-consistance

- réduit la combinatoire
- ne touche pas aux solutions valides
- il existe des algorithmes polynomiaux pour propager l'arc-consistance dans un CSP.

Algos d'AC

- AC-1
- AC-3
- AC-4
- AC-6, AC-2000
- tous font la même chose, mais plus ou moins vite.

On notera :

- n : le nombre de variables
- e : le nombre de contraintes
- d : la taille du plus grand des domaines

Algorithme : AC1

Données : Un CSP $\langle X, D, C \rangle$

$term \leftarrow FAUX$;

tant que $term = FAUX$ **faire**

  $term \leftarrow VRAI$;

  pour chaque contrainte $C_{x,y} \in C$ **faire**

    Vérifier que chaque valeur pour la variable x est supportée par une valeur de la variable y ;

    si une valeur v **n'a pas de support alors**

      $D_x \leftarrow D_x \backslash \{v\}$;

      $term \leftarrow FAUX$;

Algorithme : AC1

Données : Un CSP $\langle X, D, C \rangle$

$term \leftarrow FAUX$;

tant que $term = FAUX$ **faire**

  $term \leftarrow VRAI$;

  pour chaque contrainte $C_{x,y} \in C$ **faire**

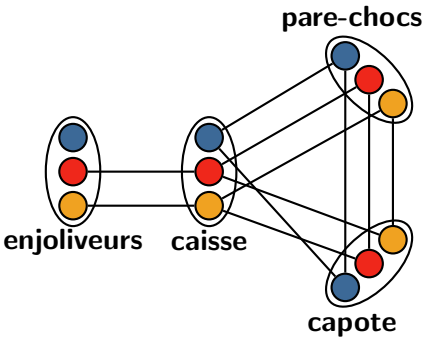
    Vérifier que chaque valeur pour la variable x est supportée par une valeur de la variable y ;

    si une valeur v **n'a pas de support alors**

      $D_x \leftarrow D_x \backslash \{v\}$;

      $term \leftarrow FAUX$;

ordre
enj ← cai
cai ← enj
cai ← p-c
p-c ← cai
p-c ← cap
cap ← p-c
cai ← cap
cap ← cai



- tant qu'on bouge quelque chose, on refait tout
- Complexité temporelle : ?

- tant qu'on bouge quelque chose, on refait tout
- Complexité temporelle : $O(ned^3)$

Algorithme : AC3

Données : Un CSP $\langle X, D, C \rangle$

$aTester \leftarrow \{\}$;

pour chaque contrainte $C_{x,y} \in C$ **faire**

  $aTester \leftarrow aTester \cup \{(x, y)\}$;

tant que aTester non vide faire

  Enlever un couple (x, y) de $aTester$;

  Vérifier que chaque valeur pour la variable x est supportée par une valeur de la variable y ;

  si une valeur v **n'a pas de support alors**

    $D_x \leftarrow D_x \backslash \{v\}$;

    pour chaque contrainte $C_{z,x} \in C$ **avec** $z \neq y$ **faire**

      $aTester \leftarrow aTester \cup \{(z, x)\}$;

Algorithme : AC3

Données : Un CSP $\langle X, D, C \rangle$

$aTester \leftarrow \{\}$;

pour chaque contrainte $C_{x,y} \in C$ **faire**

  $aTester \leftarrow aTester \cup \{(x, y)\}$;

tant que aTester non vide faire

  Enlever un couple (x, y) de $aTester$;

  Vérifier que chaque valeur pour la variable x est supportée par une valeur de la variable y ;

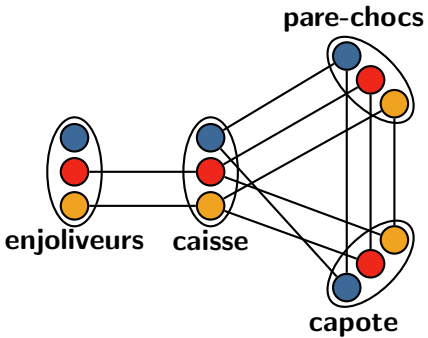
  si une valeur v **n'a pas de support alors**

    $D_x \leftarrow D_x \backslash \{v\}$;

    pour chaque contrainte $C_{z,x} \in C$ **avec** $z \neq y$ **faire**

      $aTester \leftarrow aTester \cup \{(z, x)\}$;

ordre
enj ← cai
cai ← enj
cai ← p-c
p-c ← cai
p-c ← cap
cap ← p-c
cai ← cap
cap ← cai



- quand on supprime une valeur à la variable x, on ne réexamine que les variables liées à x par une contrainte
- complexité temporelle : ?

- quand on supprime une valeur à la variable x, on ne réexamine que les variables liées à x par une contrainte
- complexité temporelle : $O(ed^3)$

Algorithme : initAC4

Données : Un CSP $\langle X, D, C \rangle$

$Q \leftarrow \{\}$; $S \leftarrow \{\}$;

pour chaque contrainte $C_{x,y} \in C$ **faire**

  pour chaque $a \in D_x$ **faire**

    $total \leftarrow 0$;

    pour chaque $b \in D_y$ **faire**

      si $(a, b) \in C_{x,y}$ **alors**

        $total \leftarrow total + 1$;

        $S(\langle y, b \rangle) \leftarrow S(\langle y, b \rangle) \cup \{\langle x, a \rangle\}$;

    $Count(x, y, a) \leftarrow total$;

    si $Count(x, y, a) = 0$ **alors**

      $D_x \leftarrow D_x \backslash \{a\}$;

      $Q \leftarrow Q \cup \{\langle x, a \rangle\}$;

Retourner Q ;

Algorithme : AC4

$Q \leftarrow initAC4()$;
tant que Q **non vide faire**
    Enlever un élément $\langle y, b \rangle$ de Q ;
    pour chaque $\langle x, a \rangle \in S(\langle y, b \rangle)$ **faire**
        $Count(x, y, a) \leftarrow Count(x, y, a) - 1$;
        si $Count(x, y, a) = 0$ **et** $a \in D_x$ **alors**
            $D_x \leftarrow D_x \backslash \{a\}$;
            $Q \leftarrow Q \cup \{\langle x, a \rangle\}$;

Algorithme : initAC4

Données : Un CSP $\langle X, D, C \rangle$
$Q \leftarrow \{\}$; $S \leftarrow \{\}$;
pour chaque contrainte $C_{x,y} \in C$ **faire**
  pour chaque $a \in D_x$ **faire**
    $total \leftarrow 0$;
    pour chaque $b \in D_y$ **faire**
      si $(a, b) \in C_{x,y}$ **alors**
        $total \leftarrow total + 1$;
        $S(\langle y, b \rangle) \leftarrow S(\langle y, b \rangle) \cup \{\langle x, a \rangle\}$;
    $Count(x, y, a) \leftarrow total$;
    si $Count(x, y, a) = 0$ **alors**
      $D_x \leftarrow D_x \backslash \{a\}$;
      $Q \leftarrow Q \cup \{\langle x, a \rangle\}$;
Retourner Q ;

Algorithme : AC4

$Q \leftarrow initAC4()$;
tant que Q **non vide faire**
  Enlever un élément $\langle y, b \rangle$ de Q ;
  pour chaque $\langle x, a \rangle \in S(\langle y, b \rangle)$ **faire**
    $Count(x, y, a) \leftarrow Count(x, y, a) - 1$;
    si $Count(x, y, a) = 0$ **et** $a \in D_x$ **alors**
      $D_x \leftarrow D_x \backslash \{a\}$;
      $Q \leftarrow Q \cup \{\langle x, a \rangle\}$;

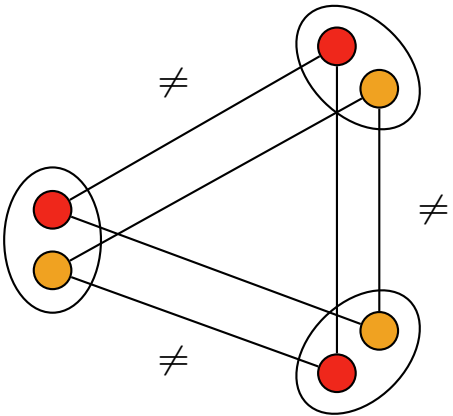
ordre
enj ← cai
cai ← enj
cai ← p-c
p-c ← cai
p-c ← cap
cap ← p-c
cai ← cap
cap ← cai

- on mémorise tous les supports des couples $\langle x, a \rangle$.
- on supprime une valeur a quand elle n'a plus de support.
- on propage à tous les couples $\langle y, b \rangle$ que $\langle x, a \rangle$ supportait.
- complexité temporelle : ?

- on mémorise tous les supports des couples $\langle x, a \rangle$.
- on supprime une valeur a quand elle n'a plus de support.
- on propage à tous les couples $\langle y, b \rangle$ que $\langle x, a \rangle$ supportait.
- complexité temporelle : $O(ed^2)$

- il existe encore d'autres algos : AC-6, AC-2000, etc.
- AC-4 plus dur à implémenter qu'AC-1 et AC-3
- AC-3 et AC-4 sont les plus utilisés

L'arc-consistance est une consistance locale. Un CSP qui est arc-consistant peut être globalement inconsistant !



Algorithmes prospectifs

- *Look-Ahead* in English
- idée : anticiper certaines prises de décisions dans le backtrack
- version light : Forward-Checking
- version lourde : Maintain-Arc-Consistency

Dès qu'une variable x est instanciée à la valeur a, on filtre toutes les valeurs incompatibles avec $\langle x, a \rangle$ **mais on ne propage pas plus**.

Algorithme : ForwardChecking

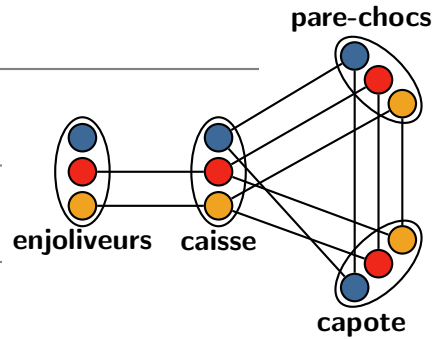
Données : Une instantiation partielle I où on vient de fixer $\langle x, a \rangle$
pour chaque variable $y \notin I$ **tq** $\exists C_{x,y}$ **faire**
  pour chaque valeur $b \in D_y$ **faire**
    si $\neg C_{x,y}(a, b)$ **alors**
      $D_y \leftarrow D_y \backslash \{b\}$;

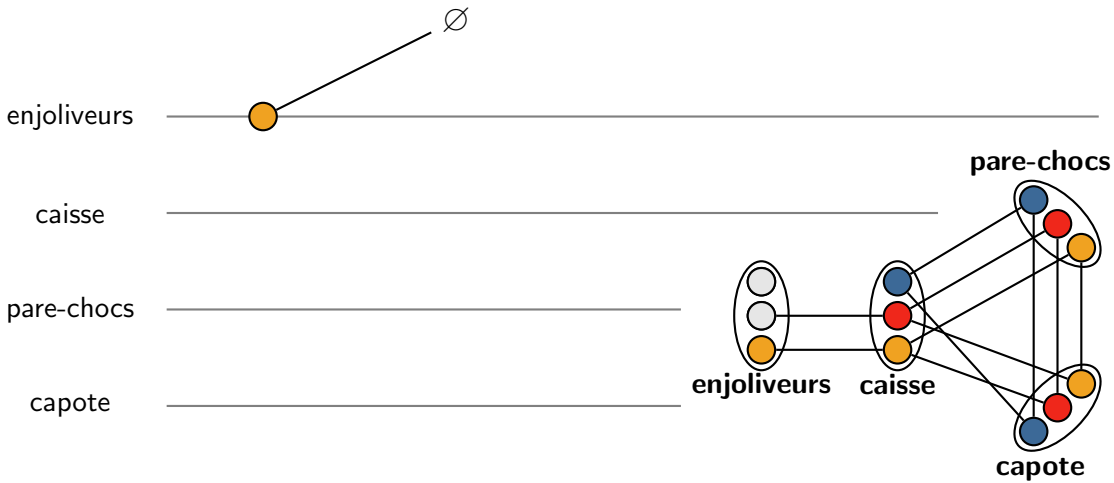
enjoliveurs

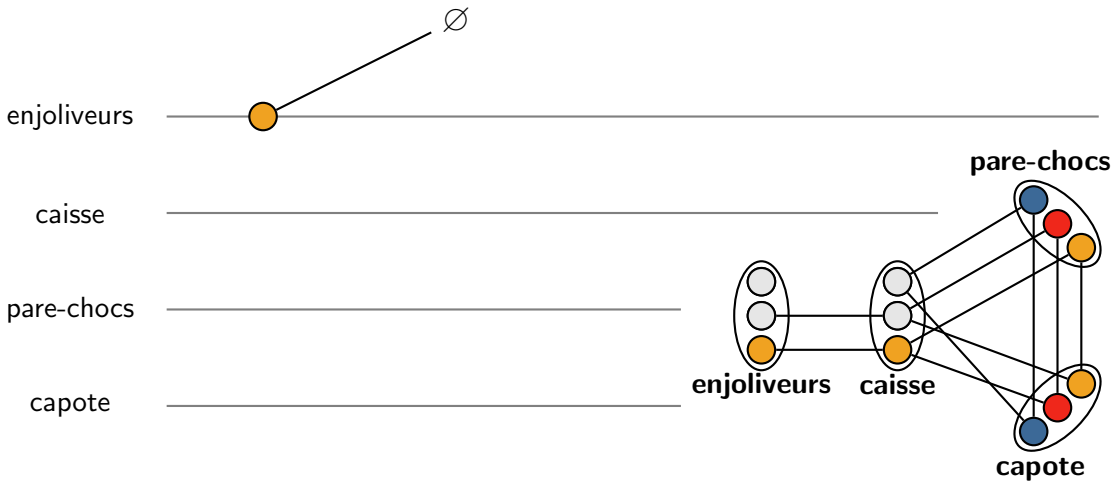
caisse

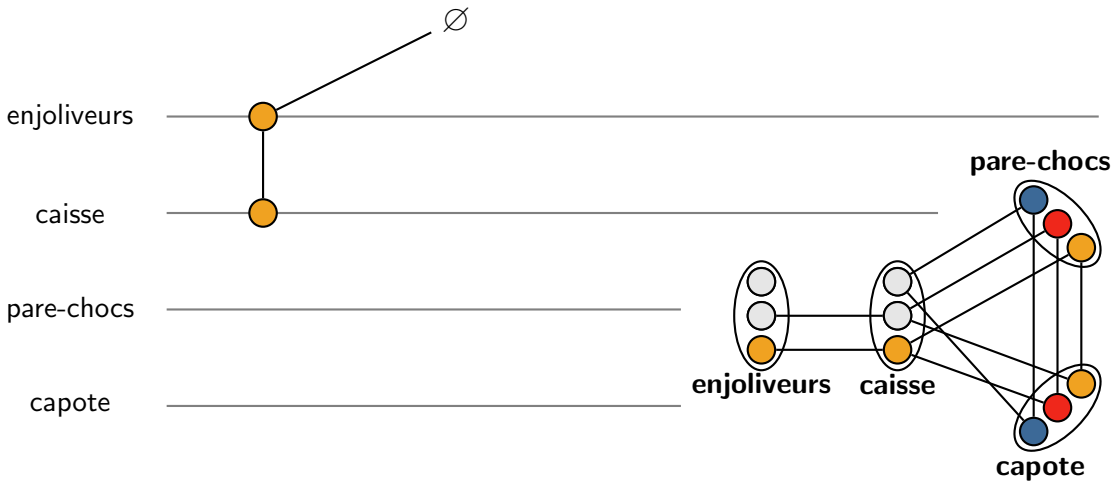
pare-chocs

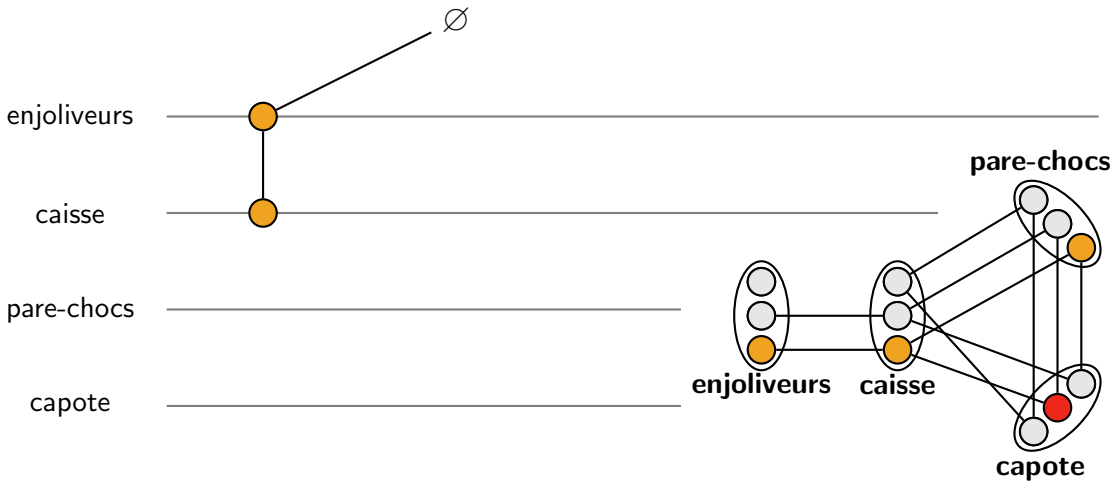
capote

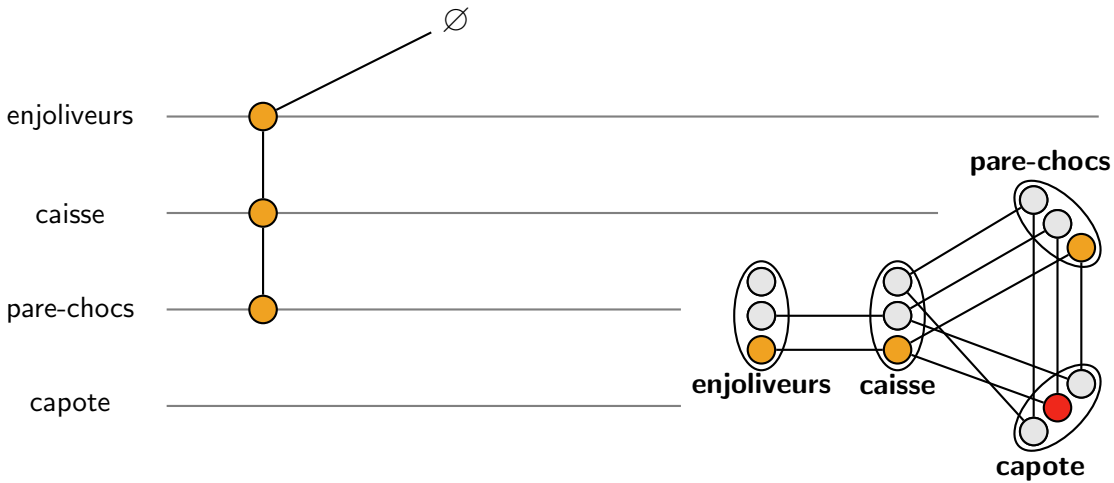


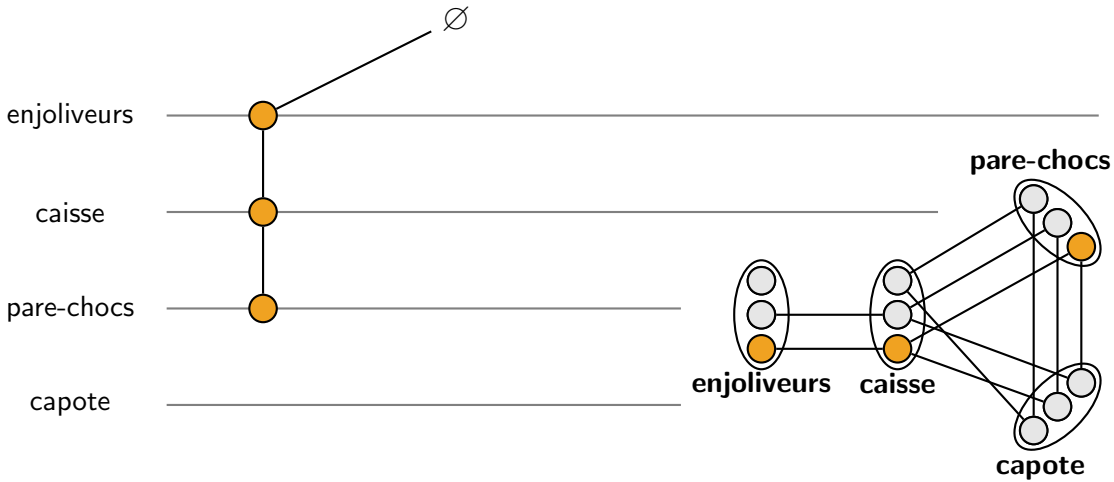

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

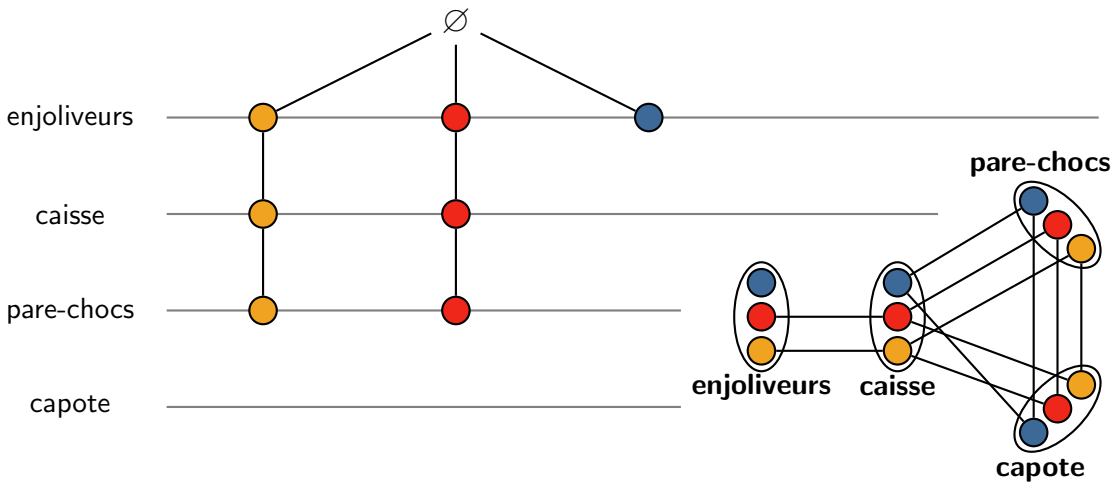

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

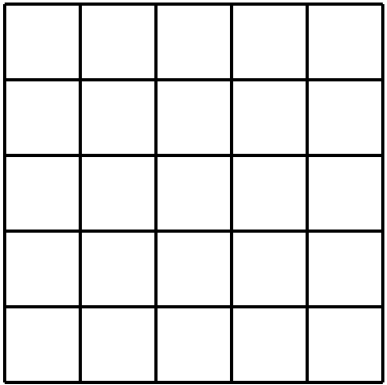

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

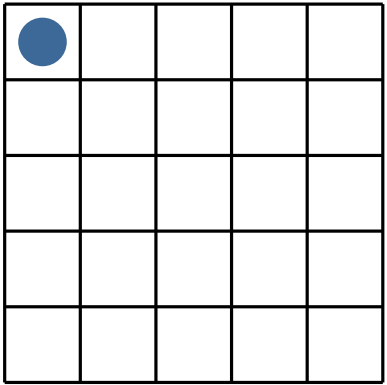

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

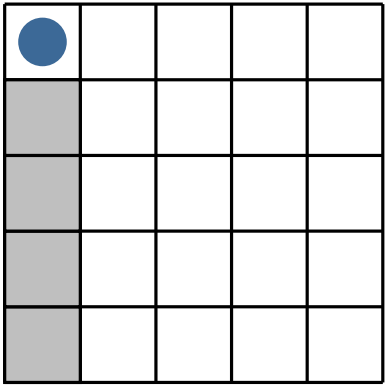
Exemple de FC sur les n-Reines



Exemple de FC sur les n-Reines

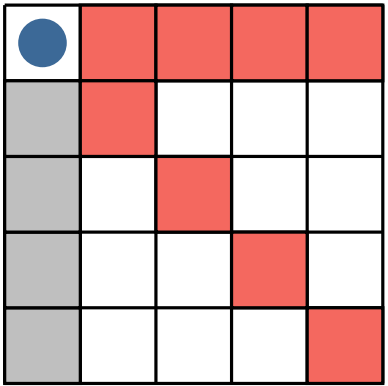


Exemple de FC sur les n-Reines



: éliminé par branchement

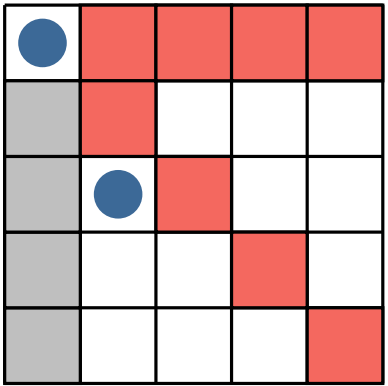
Exemple de FC sur les n-Reines



: éliminé par branchement

: éliminé par forward-checking

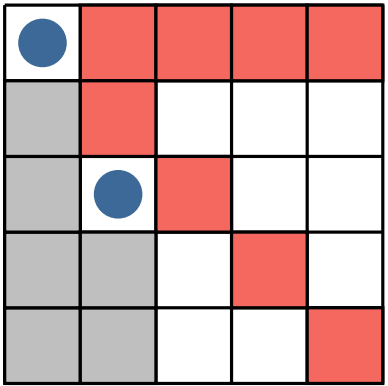
Exemple de FC sur les n-Reines



: éliminé par branchement

: éliminé par forward-checking

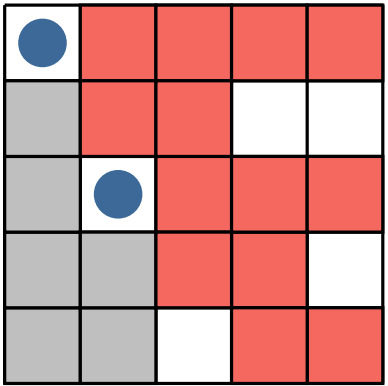
Exemple de FC sur les n-Reines



: éliminé par branchement

: éliminé par forward-checking

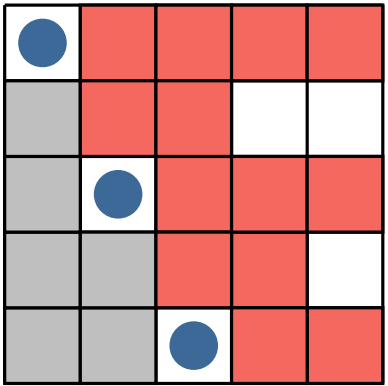
Exemple de FC sur les n-Reines



: éliminé par branchement

: éliminé par forward-checking

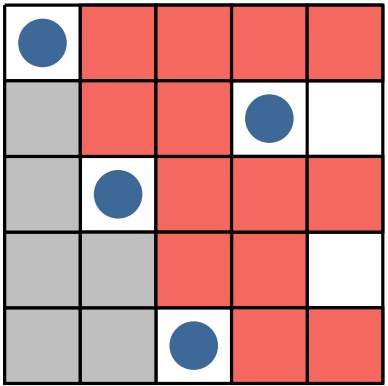
Exemple de FC sur les n-Reines



: éliminé par branchement

: éliminé par forward-checking

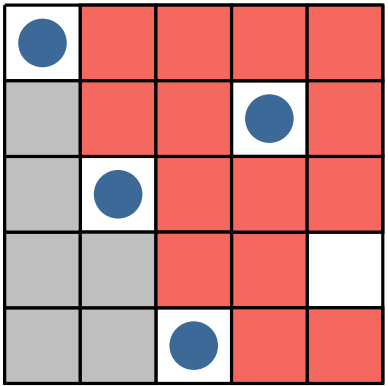
Exemple de FC sur les n-Reines



: éliminé par branchement

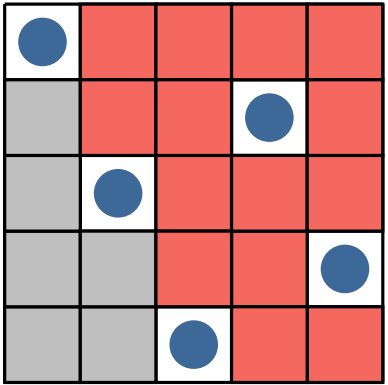
: éliminé par forward-checking

Exemple de FC sur les n-Reines



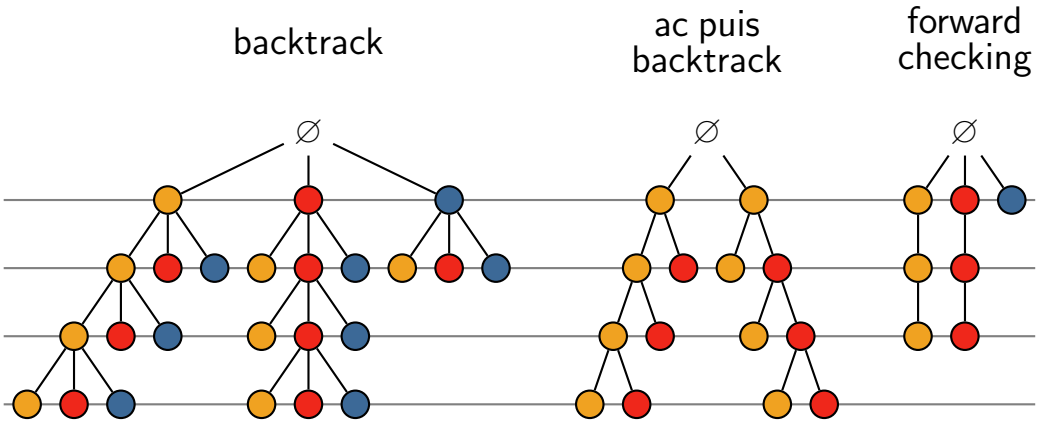
: éliminé par branchement

: éliminé par forward-checking



: éliminé par branchement

: éliminé par forward-checking


backtrack
ac puis backtrack
forward checking
∅
∅
∅

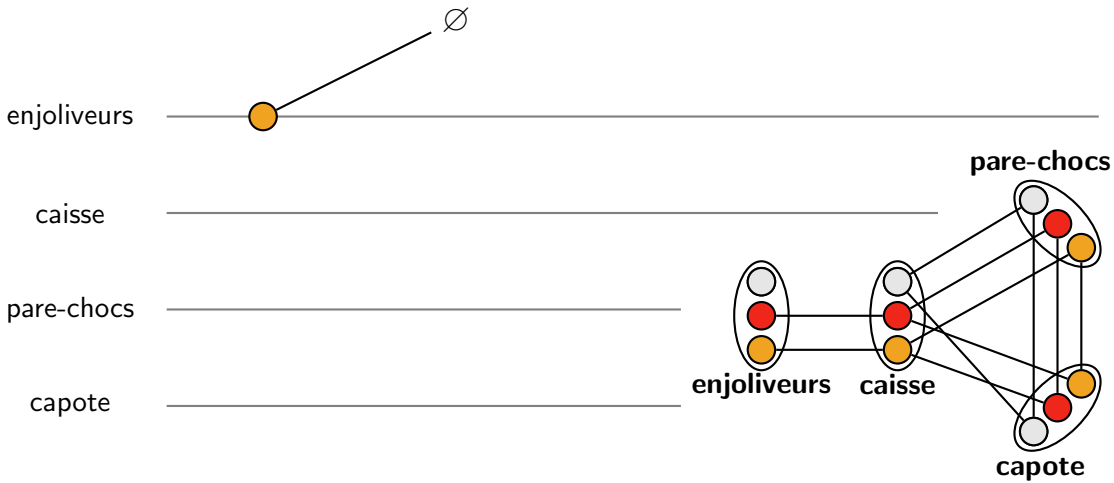
- après chaque instantiation, on réalise la fermeture arc-consistante
- s'appelle aussi *Full Look Ahead*

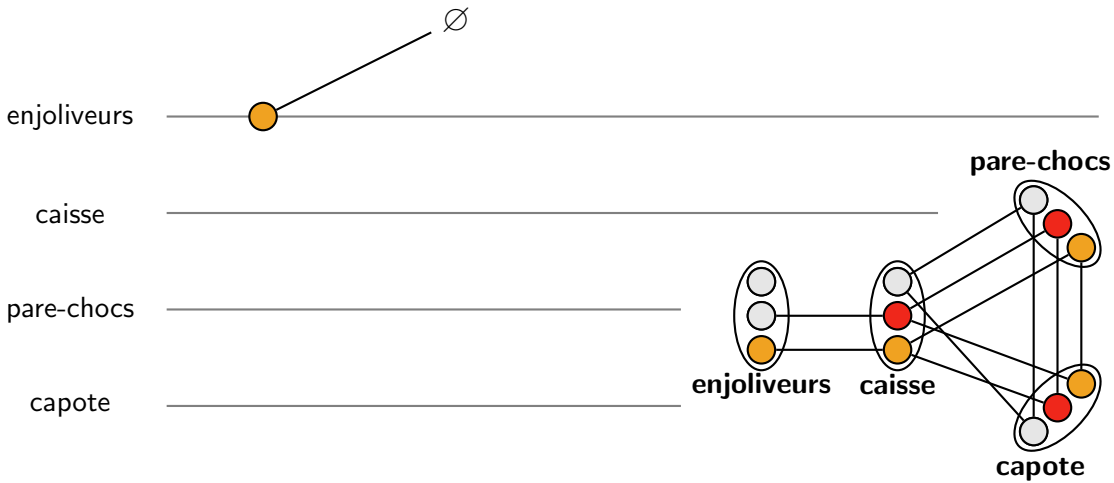
Algorithme : MaintainArcConsistency

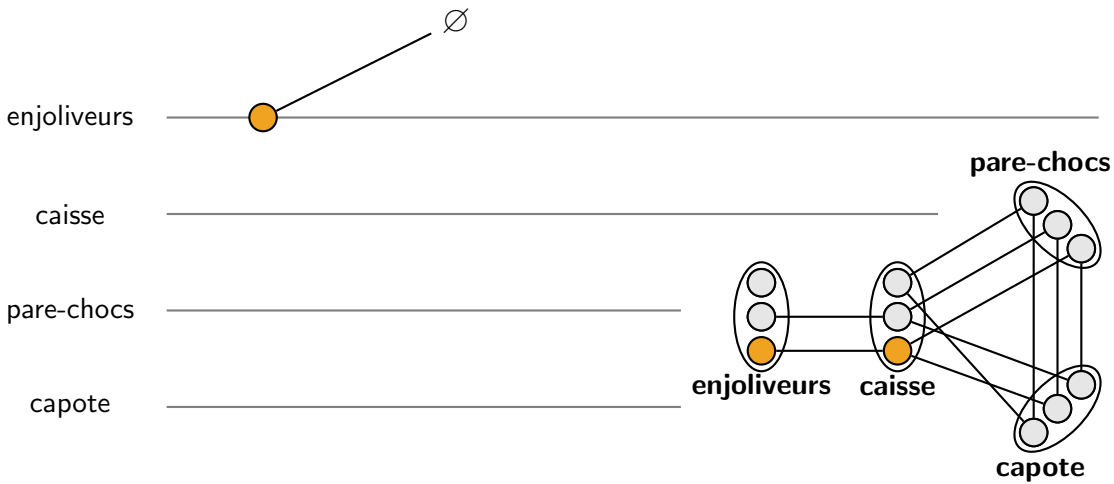
Données : Une instantiation partielle I où on vient de fixer $\langle x, a \rangle$
Appliquer AC-4 ;

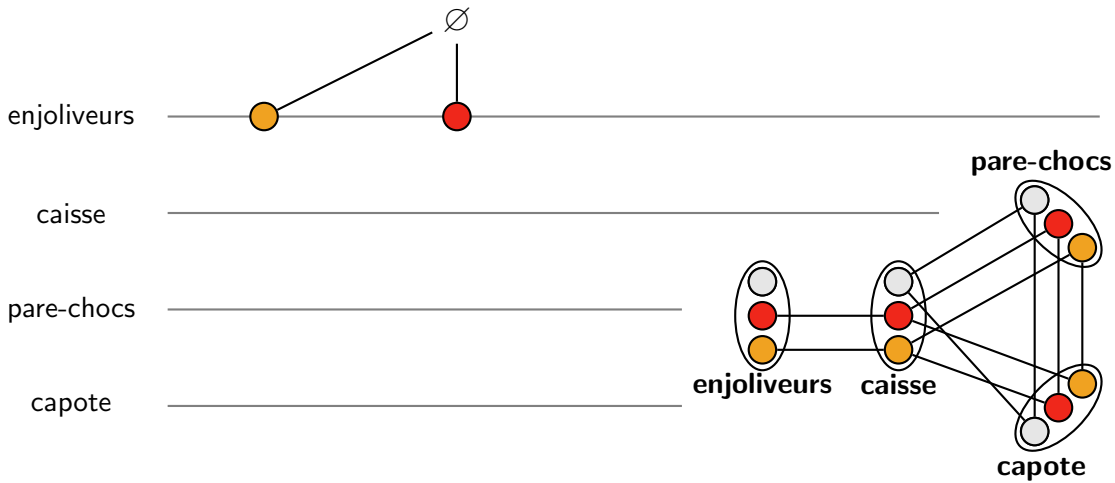

enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

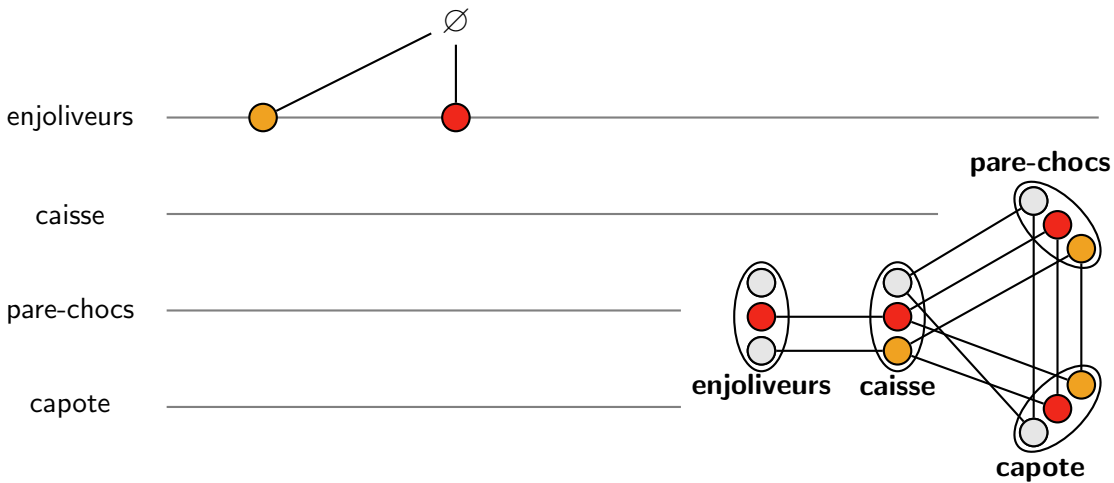

enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

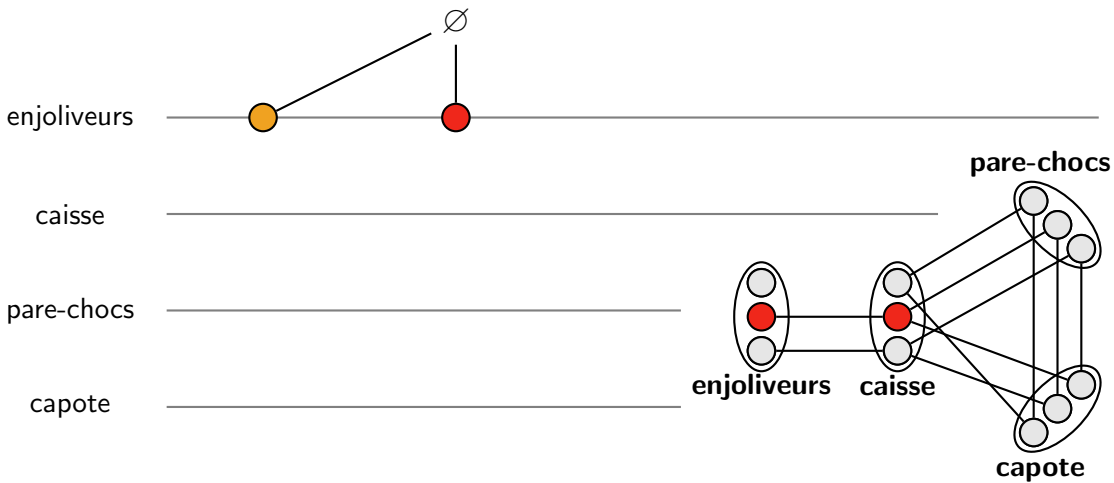

∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote


∅
enjoliveurs
caisse
pare-chocs
capote
pare-chocs
enjoliveurs
caisse
capote

- le nombre de nœuds visités diminue quand on augmente la puissance du raisonnement
- il n'en va pas nécessairement de même pour le temps de calcul !
- quand un problème est simple (sur contraint ou sous contraint), le temps de calcul le plus faible sera généralement atteint par le raisonnement le plus simple

Conclusion sur la consistance locale

- la consistance locale doit accélerer la recherche de solutions
- il faut comparer le temps qu'elle coûte au temps d'exploration qu'elle évite
- en pratique, souvent Forward Checking

Stratégies de branchement

Données : Une instantiation partielle i
si i **viole une contrainte alors**
  Retourner *FAUX* ;
si i **est complète alors**
  Retourner *VRAI* ;
Choisir une variable x non instanciée ;
pour chaque valeur v **dans** D_x **faire**
  $j \leftarrow i \cup \langle x, v \rangle$;
  si *Backtrack*(j) **alors**
    Retourner *VRAI* ;
Retourner *FAUX* ;

- dans le backtrack, tout n'est pas précisé
- ordre des variables
- ordre des valeurs
- choisir ces ordres définit une stratégie de recherche
- on parle aussi d'heuristiques
- **impacte fortement le temps de résolution**
- ordre des variables plus d'impact que l'ordre des valeurs
- 2 types : statique ou dynamique

- principe : aller le plus vite vers une contradiction.
- variable de domaine minimal
- variable la plus contrainte
- instancier en dernier les variables non liées entre elles

- principe : choisir les valeurs les plus prometteuses
- valeur la plus supportée
- valeur qui induit le moins de filtrage
- valeur qui conduit à des sous-problèmes faciles
- parfois, séparer le domaine en deux

Au delà de l'arc-consistance

On se situe toujours dans le cadre de CSP binaires.

- Nœud-Consistance : les contraintes unaires sont satisfaites par toutes les valeurs des domaines
- Arc-Consistance : déjà vu.
- Chemin-Consistance : le couple de variable (x, y) est chemin-consistant avec la variable z ssi $\forall (a, b) \in D_x \times D_y$ tel que $C_{x,y}(a, b)$, $\exists c \in D_z$ tel que $C_{x,z}(a, c)$ et $C_{y,z}(b, c)$.

On se situe toujours dans le cadre de CSP binaires.

- généralisation des cas précédents : nœud =1, arc=2, chemin=3
- Un CSP est k-consistant ssi toute instantiation partielle de $k-1$ variables consistante peut-être étendue à une instantiation partielle de k variables, en ajoutant n'importe quelle variable non instanciée.
- Complexité en $O(n^k d^k)$
- n-consistance = consistance globale = résolution du CSP

- on peut étendre la définition de l'arc-consistance aux CSP n-aires.
- Pour toute contrainte n-aire, pour toute variable impliquée dans cette contrainte, pour chaque valeur dans le domaine de cette variable, il existe des valeurs pour les autres variables de la contraintes telles que la contrainte est satisfaite.
- ça s'appelle l'**arc-consistance généralisée**.
- ça s'appelle aussi l'**hyper-arc consistance**.

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