Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique
Département d'Informatique
LMD Master 2ème Année ''Systèmes Informatiques Intelligents'' 2011/2012
Module ''Programmation par Contraintes''
Date : 14/01/2012

Corrigé de l'examen
<Corrigé préparé par le responsable du module, Mr ISLI>

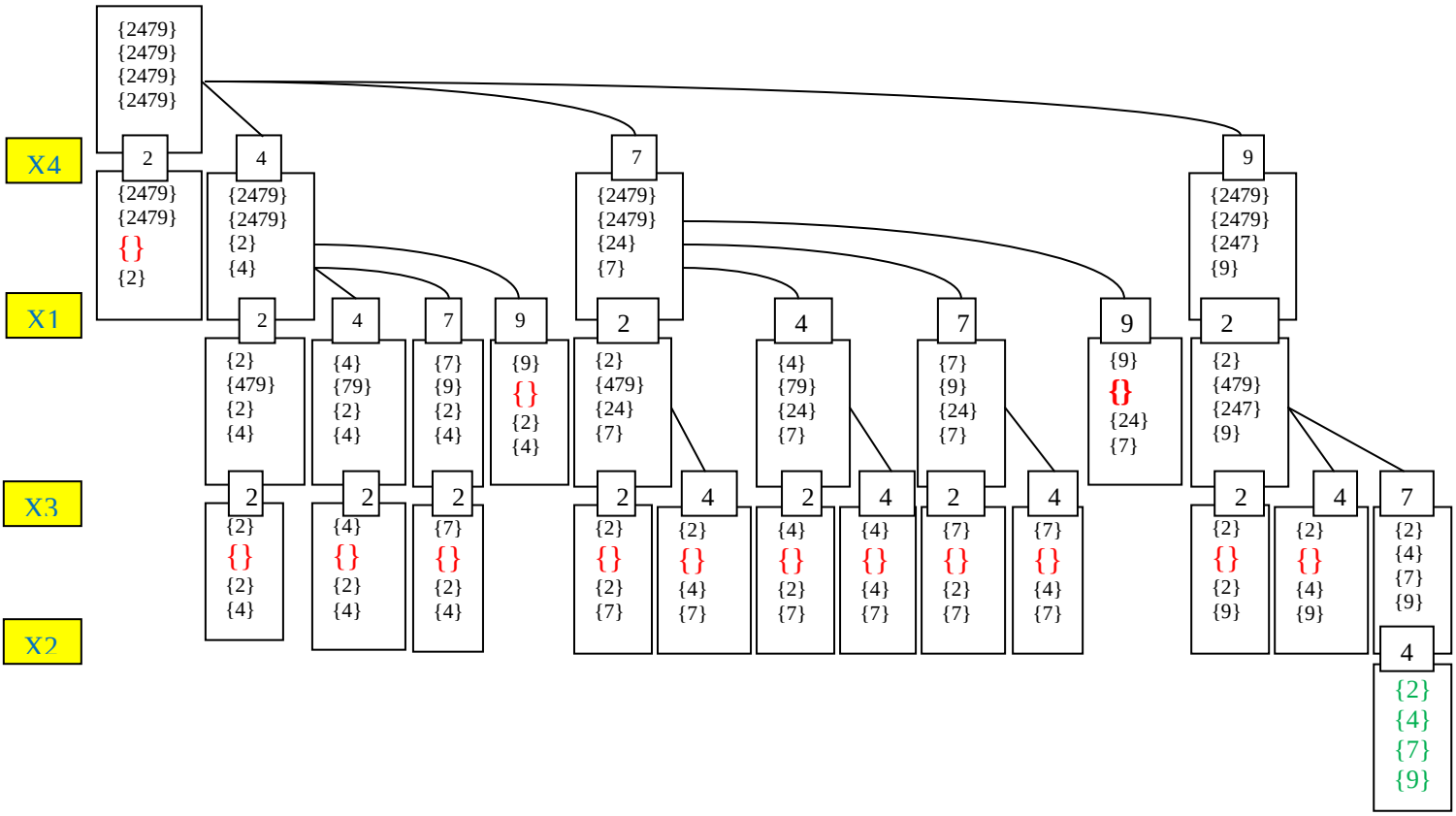
Exercice 1 :
On considère le CSP binaire discret P=(X,D,C) suivant :

- $X = \{X_1, X_2, X_3, X_4\}$
- $D(X_1)=D(X_2)=D(X_3)= D(X_4)=\{2,4,7,9\}$
- $C = \{c_1 : X_1<X_2, c_2 : X_2<X_3, c_3 : X_3<X_4\}$

Appliquer l'algorithme FC (Forward-Checking) à P en respectant l'ordre statique suivant d'instanciation des variables : X_4, X_1, X_3, X_2 ; et l'ordre suivant sur les valeurs du domaine commun des variables : 2, 4, 7, 9. Il vous est demandé de procéder de la façon suivante :

1. Donnez l'arbre de recherche correspondant en nommant lisiblement les différents nœuds.
2. Dressez un tableau donnant les domaines des différentes variables au niveau de chaque nœud de l'arbre de recherche.

Solution :



Exercice 2 :
On considère le CSP binaire discret P=(X,D,C) suivant :

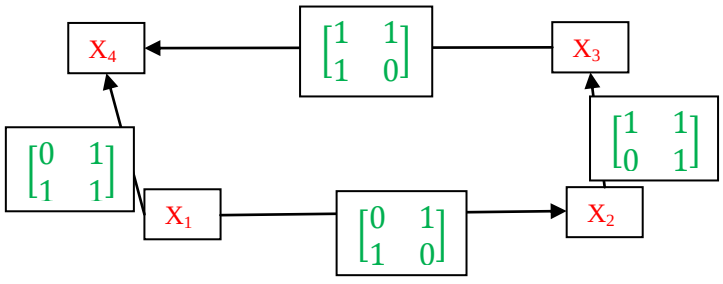
- $X = \{X_1, X_2, X_3, X_4\}$
- $D(X_1)=D(X_2)=\{a,b\}$; $D(X_3)=\{b,c\}$; $D(X_4)=\{a,c\}$
- $C = \{c_1 : X_1 \neq X_2, c_2 : X_2 \neq X_3, c_3 : X_1 \neq X_4, c_4 : X_3 \neq X_4 \}$

1. Appliquez l'algorithme de consistance d'arc AC3 à P.
2. Donnez une représentation graphique du CSP de départ P.
3. Quelle est la contrainte sur X_1 et X_3 inférée par composition des contraintes c_1 et c_2.
4. Quelle est la contrainte sur X_2 et X_4 inférée par composition des contraintes c_2 et c_4.

Solution :

1. L'algorithme AC3 appliqué à P initialiserait sa file comme suit : $F=\{(X_1,X_2),(X_2,X_1),(X_2,X_3),(X_3,X_2),(X_1,X_4),(X_4,X_1),(X_3,X_4),(X_4,X_3)\}$. L'algorithme, néanmoins, ne modifierait aucunement les domaines des variables pour la raison suivante : Pour toute paire (X_i,X_j) appartenant à F, pour tout élément V_i de $D(X_i)$, V_i a un correspondant dans $D(X_j)$: il existe $V_j \in D(X_j)$ tel que $V_i \neq V_j$ (l'unique contrainte portant sur X_i et X_j étant $X_i \neq X_j$). En d'autres termes, le CSP P est consistant d'arc.
2. Une représentation graphique de P (qui n'est pas unique) est le graphe orienté étiqueté $G_P=(V,E,l)$, dont les sommets sont les variables de P, et les arcs les paires (X_i,X_j) telles que i<j et il existe une contrainte de P entre X_i et X_j :

   - V=X
   - $E=\{(X_1,X_2),(X_1,X_4),(X_2,X_3),(X_3,X_4)\}$
   - L'étiquette d'un arc (X_i,X_j), l(Xi,Xj), est l'intersection des matrices $M_k(X_i,X_j)$ représentant les relations $R_k(X_i,X_j)$ associées aux contraintes c_k portant sur X_i et X_j
     - $l(X_1,X_2)=M_1(X_1,X_2)=\begin{bmatrix} & a & b \\ a & 0 & 1 \\ b & 1 & 0 \end{bmatrix}$
     - $l(X_2,X_3)=M_2(X_2,X_3)=\begin{bmatrix} & b & c \\ a & 1 & 1 \\ b & 0 & 1 \end{bmatrix}$
     - $l(X_1,X_4)=M_3(X_1,X_4)=\begin{bmatrix} & a & c \\ a & 0 & 1 \\ b & 1 & 1 \end{bmatrix}$
     - $l(X_3,X_4)=M_4(X_3,X_4)=\begin{bmatrix} & a & c \\ b & 1 & 1 \\ c & 1 & 0 \end{bmatrix}$



3. La contrainte sur X_1 et X_3 inférée par composition de c_1 et c_2 est la contrainte c_{13} dont la relation associée est représentée par la matrice $M_{13}(X_1,X_3)$ qui est la composition de $M_1(X_1,X_2)$ et $M_2(X_2,X_3)$:

   $M_{13}(X_1,X_3)= M_1(X_1,X_2)\circ M_2(X_2,X_3)=\begin{bmatrix} & a & b \\ a & 0 & 1 \\ b & 1 & 0 \end{bmatrix}\circ\begin{bmatrix} & b & c \\ a & 1 & 1 \\ b & 0 & 1 \end{bmatrix}=\begin{bmatrix} & b & c \\ a & 0 & 1 \\ b & 1 & 1 \end{bmatrix}$
4. La contrainte sur X_2 et X_4 inférée par composition de c_2 et c_4 est la contrainte c_{24} dont la relation associée est représentée par la matrice $M_{24}(X_2,X_4)$ qui est la composition de $M_2(X_2,X_3)$ et $M_4(X_3,X_4)$:

$M_{24}(X_2,X_4)= M_2(X_2,X_3)\circ M_4(X_3,X_4)= \begin{bmatrix} & a & b \\ a & 1 & 1 \\ b & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} & b & c \\ a & 1 & 1 \\ b & 1 & 0 \end{bmatrix} = \begin{bmatrix} & b & c \\ a & 1 & 1 \\ b & 1 & 0 \end{bmatrix}$

Exercice 3 :

On considère la scène spatiale composée des trois villes Alger, Béjaia et Oran, et d'un bâteau. On dispose de la description suivante de la scène à un instant t :

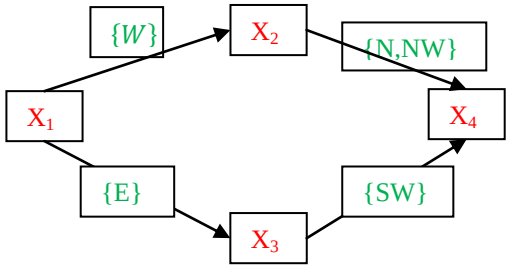
- Béjaia est à l'est d'Alger
- Oran est à l'ouest d'Alger
- Le bâteau est au sud ou au sud-est de Béjaia
- Le bâteau est au nord-est d'Oran

1. Représentez la description à l'aide d'un CSP qualitatif P=(X,C) de directions cardinales
2. Donnez une représentation graphique de P
3. Donnez une représentation matricielle de P
4. Expliquez le principe général de l'algorithme 'Générer Et Tester' (GET) sur un CSP qualitatif de directions cardinales. Justifiez la complétude de l'algorithme.
5. Le CSP P décrivant la scène spatiale en considération est-il consistant ? s'il ne l'est pas, montrez comment l'algorithme GET en détecte l'inconsistance ?
6. Expliquez le principe général de l'algorithme LookAhead sur un CSP qualitatif de directions cardinales. Justifiez la complétude de l'algorithme.
7. Appliquez l'algorithme LookAhead au CSP P décrivant la scène spatiale en considération.

Solution :

1) P=(X,C), avec $X=\{X_1,X_2,X_3,X_4\}$ et $C=\{c_1 : \{E\}(X_2,X_1), c_2 : \{W\}(X_3,X_1), c_3 : \{S,SE\}(X_4,X_2), c_4 : \{NE\}(X_4,X_3)\}$ (X_1, X_2, X_3 et X_4 représentent, respectivement, Alger, Béjaia, Oran et le bâteau)

2)



3) M_P=

	X1	X2	X3	X4
X1	{EQ}	{W}	{E}	?
X2	{E}	{EQ}	?	{N,NW}
X3	{W}	?	{EQ}	{SW}
X4	?	{S,SE}	{NE}	{EQ}

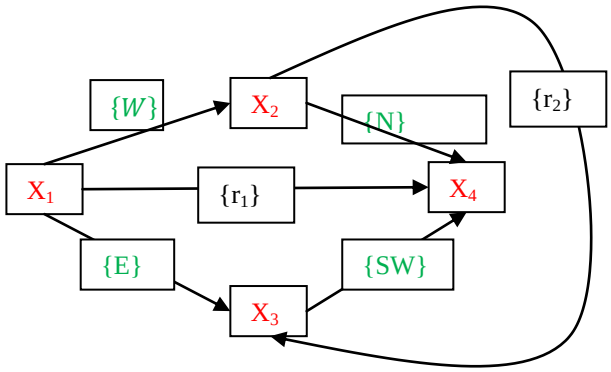
4) L'algorithme GET sur un CSP qualitatif de directions cardinales : pour plus de détails, voir cours. Nous en résumons ici les grandes lignes : GET parcourt tous les raffinements atomiques (un raffinement atomique est tel que chaque paire (X_i,X_j) de variables est reliée par une des neuf relations atomiques de l'algèbre des directions cardinales), jusqu'à éventuellement en rencontrer un pour lequel la consistance de chemin ne détecte pas la

relation vide : et si tel est le cas, c'est-à-dire s'il existe un raffinement atomique P' pour lequel la consistance de chemin ne détecte pas la relation vide, alors P' et le CSP de départ sont consistants, et l'algorithme GET prend fin avec une réponse positive au problème de consistance du CSP initial. Si le parcours de tous les raffinements atomiques prend fin sans en rencontrer un pour lequel la consistance de chemin ne détecte pas la relation d'inconsistance vide alors l'algorithme GET prend fin avec une réponse négative au problème de consistance du CSP initial. La complétude de l'algorithme GET sur un CSP qualitatif de directions cardinales se justifie par la complétude de la consistance de chemin pour les CSP atomiques de directions cardinales, qui est un résultat connu dans la littérature (voir, par exemple, les travaux de Ligozat).

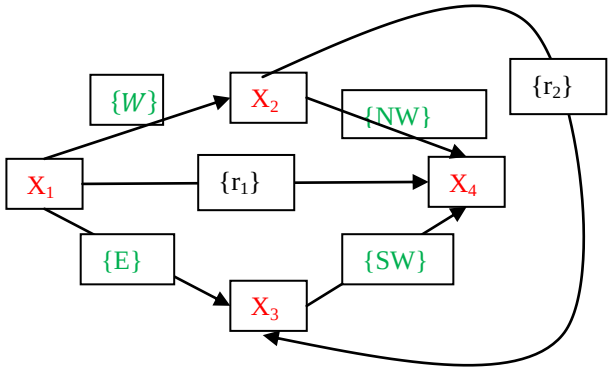
5) Le CSP décrivant la scène spatiale en considération est trivialement inconsistant : le bâteau ne peut pas être, à la fois, au nord-est d'Oran ; et au sud ou au sud-est de Béjaia (il est à noter que la composition des contraintes entre Oran et Alger, d'un côté, et entre Alger et Béjaia, de l'autre, impose la contrainte suivante sur Oran et Béjaia : Oran est l'ouest de Béjaia).

   L'algorithme GET appliqué à P considérera donc tous les raffinements atomiques possibles qui sont de l'une ou de l'autre des deux formes suivantes, où r1 et r2 sont tout simplement des relations atomiques (en tout, 9*9+9*9=162 raffinements atomiques possibles), jusqu'à éventuellement en rencontrer un pour lequel la consistance de chemin, par exemple l'algorithme PC2, ne détecte pas la relation d'inconsistance vide, ce qui prouverait la consistance du CSP de départ : mais vu que le CSP de départ est (trivialement) inconsistant, PC2 devrait, pour chacun des 162 raffinements atomiques, détecter la relation vide).

   Forme 1 de raffinement atomique : (X2,X4) étiqueté par {N}



Forme 2 de raffinement atomique : (X2,X4) étiqueté par {NW}



Pour un raffinement de la forme numéro 1, il faut noter ceci :

a) La contrainte sur X_1 et X_4 inférée par composition des contraintes sur X_1 et X_2, d'une part, et sur X_2 et X_4, d'autre part, est $R(X_1,X_4)$ avec $R=\{W\}°\{N\}=\{NW\}$
b) La contrainte toujours sur X_1 et X_4 inférée par composition des contraintes sur X_1 et X_3, d'une part, et sur X_3 et X_4, d'autre part, est $R'(X_1,X_4)$ avec $R'=\{E\}°\{SW\}=\{SW,S,SE\}$

L'intersection de R et R' est la relation vide : donc PC2 détectera l'inconsistance de chacun des 9*9 raffinements atomiques de la forme 1.

Pour un raffinement de la forme 2 :

a) La contrainte sur X_1 et X_4 inférée par composition des contraintes sur X_1 et X_2, d'une part, et sur X_2 et X_4, d'autre part, est $R(X_1,X_4)$ avec $R=\{W\}°\{NW\}=\{NW\}$
b) La contrainte toujours sur X_1 et X_4 inférée par composition des contraintes sur X_1 et X_3, d'une part, et sur X_3 et X_4, d'autre part, est $R'(X_1,X_4)$ avec $R'=\{E\}°\{SW\}=\{SW,S,SE\}$

L'intersection de R et R' est la relation vide : donc PC2 détectera aussi l'inconsistance de chacun des 9*9 raffinements atomiques de la forme 2.

6) L'algorithme LookAhead sur un CSP qualitatif de directions cardinales : pour plus de détails, voir cours. Nous en résumons ici les grandes lignes : l'algorithme applique un prétraitement par PC2 au CSP de départ. Si détection de la relation d'inconsistance vide alors le CSP de départ est inconsistant. Sinon, l'algorithme procède selon une politique instancier-puis-propager, qui consiste à instancier un arc disjonctif avec une des relations atomiques composant son étiquette, et à appliquer au CSP résultant un filtrage par PC2 ; si le filtrage détecte la relation vide, alors échec (deadend) : il faut essayer une autre relation atomique pour l'arc en cours d'instanciation ; si toutes les relations atomiques ont été essayées, il faut faire un retour arrière, remonter sur l'arc le plus récemment instancié ; si on remonte jusqu'à la racine et que toutes les possibilités ont déjà été essayées alors le CSP de départ est inconsistant. Si LookAhead réussit à instancier tous les arcs disjonctifs sans que les différents filtrages par PC2 ne détectent la relation vide alors le raffinement atomique résultant, et par conséquent le CSP de départ, est consistant. La complétude de l'algorithme LookAhead se justifie, ici aussi, par la complétude de l'algorithme de consistance de chemin PC2 pour les CSP atomiques de directions cardinales.

7) L'algorithme LookAhead appliqué au CSP P en considération commencera donc par appliquer à P un prétraitement par PC2 : le prétraitement détectera l'inconsistance de P de la façon suivante :
   a. La contrainte sur X_1 et X_4 inférée par composition des contraintes sur X_1 et X_2, d'une part, et sur X_2 et X_4, d'autre part, est $R(X_1,X_4)$ avec $R=\{W\}^\circ\{N,NW\}=\{NW\}$
   b. La contrainte toujours sur X_1 et X_4 inférée par composition des contraintes sur X_1 et X_3, d'une part, et sur X_3 et X_4, d'autre part, est $R'(X_1,X_4)$ avec $R'=\{E\}^\circ\{SW\}=\{SW,S,SE\}$

   L'intersection de R et R' est bien la relation vide !

Exercice 4 : Ecrire un programme Prolog de tri par sélection d'une liste.

Solution :

Rappel : algorithme de tri par sélection d'un tableau A

L'algorithme partitionne le tableau en deux parties : partie triée (gauche) et partie non triée (droite). Initialement, la partie triée est vide et la partie non triée est égale à tout le tableau. L'algorithme procède par augmentation d'une unité de la taille de la partie triée (et réduction d'une unité de la taille de la partie non triée), en ramenant le plus petit élément de la partie non triée en tête (à la première position) de cette même partie. A tout moment, tous les éléments de la partie triée sont plus petits que tous les éléments de la partie non triée. Quand la taille de la partie triée atteint n-1, le tableau est trié.

```
TRI-SELECTION(A)
 début
        pour i=1 à n-1 faire
                min= A[i]
                k=i
                pour j=i+1 à n faire
                        si min>A[j] alors
                                min=A[j]
                                k=j
                        finsi
                fait
                //permuter A[i] et A[k]
                temp=A[i]
                A[i]=A[k]
                A[k]=temp
        fait
 fin
```

Le programme Prolog :

```
tri_select([],[]).
tri_select([X],[X]).
tri_select(L1,[X|L2]) :-min_avec_ind(L1,X,I),permuter(L1,I,[X|L3]),tri_select(L3,L2).
//min_avec_ind(L,X,I) calcule le min de L dans X, et l'indice de la première occurrence du min dans I
min_avec_ind([X],X,1).
min_avec_ind([X|L],X,1) :-min_avec_ind(L,Y,k),X=<Y.
min_avec_ind([X|L],Y,I) :-J is I-1,min_avec_ind(L,Y,J).
//permuter(L1,I,L2) calcule dans L2 le résultat de la permutation dans L1 des 1er et Ième éléments
permuter(L,1,L).
permuter(L1,I,L2) :-deux_blocs(L1,I,[X|L3],[Y|L4]),append([Y|L3],[X|L4],L2).
//deux_blocs(L,I,L1,L2) : L1 de taille I-1 ; I≥2, ≤ à taille de L ; taille de L≥2
deux_blocs(L,I,L1,L2) :-J is I-1,length(L1,J),append(L1,L2,L).
```