Université Paris 7 - Master 1 Informatique - Programmation logique par contraintes

Examen du 7 janvier 2011 - Durée : 2 heures

Informations : Tous les documents sont autorisés. Le barème est donné à titre indicatif et peut être modifié.

Exercice 1 (**4 points**)

Considérez le problème suivant :
Minimiser $-X - 2 * Y$ par rapport à
$-3 * X + 2 * Y \leq 2 \wedge$
$-X + 3 * Y \leq 6 \wedge$
$X + Y \leq 5 \wedge$
$X \geq 0 \wedge Y \geq 0$

- Visualiser le problème en dessinant l'espace des solutions possibles (un polygone) déterminé par les contraintes.
- Donner les détails de la résolution de ce problème par la méthode simplex. **Indication :** Pour cela, il faut entre autres mettre le problème en forme simplex, trouver une solution de base, etc.
- Pour quelles valeurs de X et Y le minimum est-il atteint ?
- Si on exige que X et Y soient entières, quel est le minimum ? Pour quelles valeurs de X et Y est-il atteint ? Répondre à ces deux questions sans justification.

Exercice 2 (**4 points**)

On considère la contrainte $2 * X + 3 * Y = 5 * Z \ \wedge \ 2 * X + 1 = Y \ \wedge \ X > 3$ sur les entiers avec les domaines $D(X) = [1 \dots 9] = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $D(Y) = [3 \dots 15]$ et $D(Z) = [6 \dots 10]$. Les points suivants sont à faire **indépendamment**.

- Rendez la contrainte nœud-consistante. Donnez uniquement les nouveaux domaines.
- Rendez la contrainte nœud **et** arc-consistante. Donnez uniquement les nouveaux domaines.
- Rendez la contrainte borne-consistante. **Détaillez.**
- Quelle est l'unique solution de la contrainte ?

Exercice 3 (**4 points**)

Pour un $n \geq 2$, on considère la contrainte *alldifferent*$(x_1, \dots, x_n)$ qui signifie $\bigwedge_{1 \leq i,j \leq n \wedge i \neq j} x_i \neq x_j$. On peut appliquer la borne consistance à ce type de contrainte (qu'on considère comme une contrainte simple). Rappel de la borne consistance pour ce cas : On note $D_i \subseteq \mathbb{N}$ le domaine fini de chaque variable x_i. On note $min(D_i)$ (resp. $max(D_j)$) le minimum (resp. maximum) de chaque domaine. Une contrainte simple est borne-consistante, ssi pour chaque variable et le minimum (resp. maximum) de son domaine, on peut trouver des valeurs pour les autres variables entre les valeurs min et max de leurs domaines telles que la contrainte est satisfaite. Les valeurs pour les autres variables ne font pas forcément partie de leurs domaines ! (voir exemple 2 ci-dessous) Formellement,

- Une contrainte simple c avec n variables est borne-consistante, si pour chaque variable x_i dans $variables(c)$
  - ils existent pour chaque j avec $1 \leq j \leq n$ et $j \neq i$, des valeurs entières d_j avec
    - $min(D_j) \leq d_j \leq max(D_j)$
    - $\{x_1 \leftarrow d_1, \dots, x_{i-1} \leftarrow d_{i-1}, x_i \leftarrow min(D_i), x_{i+1} \leftarrow d_{i+1}, \dots x_n \leftarrow d_n\}$ est une solution de c
  - et ils existent pour chaque j avec $1 \leq j \leq n$ et $j \neq i$, des valeurs entières d_j avec
    - $min(D_j) \leq d_j \leq max(D_j)$
    - $\{x_1 \leftarrow d_1, \dots, x_{i-1} \leftarrow d_{i-1}, x_i \leftarrow max(D_i), x_{i+1} \leftarrow d_{i+1}, \dots x_n \leftarrow d_n\}$ est une solution de c

Exemples :

1. La contrainte `alldifferent`(x_1, x_2, x_3) avec $D_1 = \{1, 2\}$, $D_2 = \{1\}$, $D_3 = \{1, 2, 3\}$ n'est pas borne-consistante (car pour $x_1 = 1$ on ne peut pas trouver de valeur pour x_2 qui satisfait la contrainte).
2. La contrainte `alldifferent`(x_1, x_2, x_3) avec $D_1 = \{1, 3\}$, $D_2 = \{1, 3\}$, $D_3 = \{1, 3\}$ est borne-consistante, car on a le droit d'utiliser 2 !

3. La contrainte `alldifferent`(x_1, x_2, x_3) avec $D_1 = \{1, 3\}$, $D_2 = \{2\}$, $D_3 = \{1, 2, 3\}$ est aussi borne-consistante.

On considère maintenant la contrainte `alldifferent`$(x_1, x_2, x_3, x_4, x_5, x_6)$ avec domaines $D_1 = \{3, 4\}$, $D_2 = \{2, 3, 4\}$, $D_3 = \{3, 4\}$, $D_4 = \{2, 3, 4, 5\}$, $D_5 = \{3, 4, 5, 6\}$, $D_6 = \{1, 2, 3, 4, 5, 6\}$.

- Réduire les domaines en utilisant la borne-consistance.
- Donner un exemple d'une contrainte `alldifferent` avec domaines de variables qui est borne-consistante mais qui n'a pas de solution.
- La contrainte `alldifferent`(x_1, x_2, x_3, x_4) avec $D_1 = \{1, 4\}$, $D_2 = \{2, 3\}$, $D_3 = \{1, 2, 3, 4\}$, $D_4 = \{2, 3\}$ est borne-consistante, mais on peut réduire le domaine avec un autre type de consistance. Proposer une notion de consistance qui permet d'éliminer au moins 2 du domaine de D_3. Préférer une notion qui n'est pas équivalente à la satisfaisabilité de la contrainte.

Exercice 4 (**4 points**)

On considère le problème de générer un emploi de temps. Il y a 7 cours et 4 créneaux. Un créneaux peut accueillir au maximum 2 cours. Il y a les contraintes suivantes :

- Cours 4 doit être avant cours 6.
- Cours 5 doit être avant cours 7.
- Cours 6 doit être avant cours 2.
- Cours 1 ne peut pas être en parallèle avec les cours $2, 3, 4$ et 7.
- Cours 2 ne peut pas être en parallèle avec les cours 3 et 6.
- Cours 3 ne peut pas être en parallèle avec les cours $4, 5$ et 6.
- Cours 4 ne peut pas être en parallèle avec les cours 5 et 6.
- Cours 5 ne peut pas être en parallèle avec le cours 7.

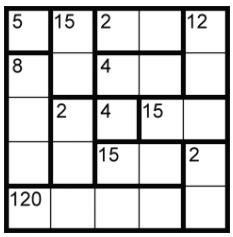
Donner un programme GPROLOG pour ce problème. Décrire votre façon de modéliser le problème (Quelles sont les variables ? Que signifient leurs valeurs ? etc.). Indication : le prédicat GPROLOG prédéfini `fd_cardinality(N,L,N1)` est vrai, ssi le nombre de contraintes vraies dans `L` est entre `N` et `N1`. **Bonus :** Une solution qui est générale et qui peut être utilisée pour d'autres problème du même genre.

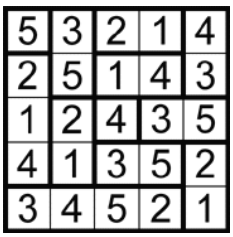
Exercice 5 (**6 points**)

Inshi no heya est un jeu logique japonais. Il est joué sur une grille rectangulaire de cellules qui est séparée en rectangles. Une dimension de chaque rectangle est 1 tandis que l'autre dimension varie. Chaque rectangle est placé horizontalement ou verticalement et contient un nombre. Le but est de remplir toutes les cellules avec des chiffres de 1 à 9 de sorte que :

- Si on multiplie tous les chiffres de chaque rectangle on obtient le nombre indiqué dans le rectangle
- Aucun chiffre apparaît plusieurs fois dans une colonne.
- Aucun chiffre apparaît plusieurs fois dans une ligne.

Un exemple d'une grille d'origine et une solution sont donné ci-dessous :



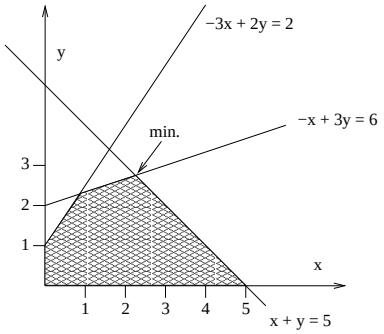


- Donner un programme en GPROLOG qui résout le Inshi no heya *de l'exemple.*
- (A faire s'il vous reste du temps) Donner un programme en GPROLOG qui résout un Inshi no heya *quelconque.* Décrire votre façon de modéliser le problème. Décrire notamment comment représenter les entrées (taille, position, etc. des rectangles, etc.). On peut supposer que l'entrée est correcte (les rectangles couvrent tout, etc.). Rappel : `length(L,N)` où `L` est une variable et `N` une valeur crée une liste `L` de `N` variables.

Correction :

Exercice 1

Visualisation :



On met le problème en forme simplex :

Minimiser $-X - 2 * Y$ par rapport à
$-3 * X + 2 * Y + U = 2 \wedge$
$-X + 3 * Y + V = 6 \wedge$
$X + Y + W = 5 \wedge$
$X, Y, U, V, W \geq 0$

On obtient une solution de base facilement :
Minimiser $-X - 2 * Y$ par rapport à
$U = 2 + 3 * X - 2 * Y \wedge$
$V = 6 + X - 3 * Y \wedge$
$W = 5 - X - Y \wedge$
$X, Y, U, V, W \geq 0$

On choisit X où Y maintenant ? La solution de base correspond au point $X = 0$, $Y = 0$. Pour atteindre plus rapidement le minimum il faut faire en sorte que X devienne 5. Donc on choisit X et on pivote la troisième équation. On obtient :
Minimiser $-5 - Y + W$ par rapport à
$U = 17 - 5 * Y - 3 * W \wedge$
$V = 11 - W - 4 * Y \wedge$
$X = 5 - W - Y \wedge$
$X, Y, U, V, W \geq 0$

Maintenant, il reste Y avec un coefficient négatif. Quelle équation choisir ? On compare 17/5, 11/4 et 5/1 et prend le plus petit. Donc la deuxième équation. Cela donne :
Minimiser $-31/4 + (5/4) * W + (1/4) * V$ par rapport à
$U = 13/4 - (7/4) * W + (5/4) * V \wedge$
$Y = 11/4 - (1/4) * W - (1/4) * V \wedge$
$X = 9/4 - (3/4) * W + (1/4) * V \wedge$
$X, Y, U, V, W \geq 0$

et c'est en forme résolue. Le minimum est donc -7.75 qui est atteint pour $X = 2.25$ et $Y = 2.75$ ce qu'on peut vérifier sur le dessin.

Si on exige que X et Y soient entières, le minimum est 7 atteint pour $X = 3$ et $Y = 2$.

Exercice 2

- nœud-consistance : $D(X) = [4 \dots 9]$, $D(Y) = [3 \dots 15]$, $D(Z) = [6 \dots 10]$
- nœud et arc-consistance : $D(X) = [4 \dots 7]$, $D(Y) = \{9, 11, 13, 15]$, $D(Z) = [6 \dots 10]$
- borne consistance : D'abord $X \geq 3$ implique que $D(X) = [4 \dots 9]$. Ensuite nous considérons la contrainte $Y = 2 * X + 1$. Nous avons $9 = 2 * min(X) + 1 \leq Y \leq 2 * max(X) + 1 = 19$. D'où $D(Y) = [9 \dots 15]$. Ensuite $2 * X + 1 = Y$ est équivalent à $X = (Y - 1)/2$. Donc $4 =$

$(min(Y)-1)/2 \leq X \leq (max(Y)-1)/2 = 7$. D'où $D(X) = [4 \dots 7]$. Ensuite nous considérons la contrainte $2*X+3*Y = 5*Z$. Elle est équivalente à $Z = (2*X+3*Y)/5$ ainsi qu'à $Y = (5*Z-2*X)/3$ et à $X = (5*Z-3*Y)/2$. Donc $7 = (2*min(X)+3*min(Y))/5 \leq Z \leq (2*max(X)+3*max(Y))/5 = 59/5$. D'où $D(Z) = [7 \dots 10]$. Aussi $9 = (5*min(Z)-2*max(X))/3 \leq Y \leq (5*max(Z)-2*min(X))/3 = 14$. D'où $D(Y) = [9 \dots 14]$. Aussi $-9/2 = (5*min(Z)-3*max(Y))/2 \leq X \leq (5*max(Z)-3*min(Y))/2 = 23/2$. Donc pas de changement du domaine de X. On vérifie encore une fois avec $Z = (2*X+3*Y)/5$ qui ne fait plus changer les domaines. On considère une deuxième fois la contrainte $Y = 2*X+1$. Le domaine de Y ne change d'abord pas, car celui de X n'a pas changé. Par contre le domaine de X change. On a $4 = (min(Y)-1)/2 \leq X \leq (max(Y)-1)/2 = 13/2$. D'où $D(X) = [4 \dots 6]$. Ensuite $9 = 2*min(X)+1 \leq Y \leq 2*max(X)+1 = 13$. Donc $D(Y) = [9 \dots 13]$. La contrainte $2*X+3*Y = 5*Z$ ne change plus les domaines. Donc le résultat final est $D(X) = [4 \dots 6]$, $D(Y) = [9 \dots 13]$ et $D(Z) = [7 \dots 10]$.

- L'unique solution est : $X = 4$, $Y = 9$, $Z = 7$.

Exercice 3

- D'abord on enlève 4 et ensuite 3 du domaine de D_2. Ensuite, on enlève successivement 2, 3 et 4 du domaine de D_4. Ensuite, on enlève 3, 4 et 5 du domaine de D_5 et finalement on enlève tout sauf 1 de D_6. Résultat final : $D_1 = \{3,4\}$, $D_2 = \{2\}$, $D_3 = \{3,4\}$, $D_4 = \{5\}$, $D_5 = \{6\}$, $D_6 = \{1\}$.
- Exemple 2 de l'exercice
- Au lieu de juste regarder les bornes de chaque variable on regarde toutes les valeurs du domaine. Par exemple pour $x_3 = 2$, il n'y a pas de valeurs pour x_2 et x_4 entre les deux bornes qui rendent la contrainte satisfaisable.

Exercice 4
Solution spécifique :

```
emploi(L) :- L = [C1,C2,C3,C4,C5,C6,C7],
             fd_domain(L,1,4),
             C4 #< C6, C5 #< C7, C6 #< C2,
             C1 #\= C2, C1 #\= C3, C1 #\= C4, C1 #\= C7,
             C2 #\= C3, C2 #\= C6,
             C3 #\= C4, C3 #\= C5, C3 #\= C6,
             C4 #\= C5, C4 #\= C6,
             C5 #\= C7,
             fd_cardinality(1,[C1#=1,C2#=1,C3#=1,C4#=1,C5#=1,C6#=1,C7#=1],2),
             fd_cardinality(1,[C1#=2,C2#=2,C3#=2,C4#=2,C5#=2,C6#=2,C7#=2],2),
             fd_cardinality(1,[C1#=3,C2#=3,C3#=3,C4#=3,C5#=3,C6#=3,C7#=3],2),
             fd_cardinality(1,[C1#=4,C2#=4,C3#=4,C4#=4,C5#=4,C6#=4,C7#=4],2),
             fd_labeling(L).
```

Solution générale :

```
prec([],_).
prec([(X,Y)|L1],L) :- nth(X,L,A), nth(Y,L,B), A #< B, prec(L1,L).

parallel([],_).
parallel([(X,L2)|L1],L) :- nth(X,L,A), para(A,L2,L), parallel(L1,L).

para(_,[],_).
para(A,[Y|L1],L) :- nth(Y,L,B), A #\= B, para(A,L1,L).

nombrecours(_,0,_) :- !.
nombrecours(L1,Nbcreneaux,Maxcours) :-
    cours(L1,Nbcreneaux,Maxcours),
    N1 is Nbcreneaux-1,
```

```
    nombrecours(L1,N1,Maxcours).

cours(L1,Nbcreneaux,Maxcours) :- listecontrainte(L1,Nbcreneaux,Maxcours,L),
                                 fd_cardinality(0,L,Maxcours).

listecontrainte([],_,_,[]).
listecontrainte([X|L1],Nbcreneaux,Maxcours,[X #= Nbcreneaux|L]) :-
                          listecontrainte(L1,Nbcreneaux,Maxcours,L).

emploi(Nbcreneaux,Nbcours,Maxcours,Prec,Parallel,L1) :-
    length(L1,Nbcours),
    fd_domain(L1,1,Nbcreneaux),
    prec(Prec,L1),
    parallel(Parallel,L1),
    nombrecours(L1,Nbcreneaux,Maxcours),
    fd_labeling(L1).

% Example:
% emploi(4,7,2,[(4,6),(5,7),(6,2)],[(1,[2,3,4,7]),(2,[3,6]),(3,[4,5,6]),
                                    (4,[5,6]),(5,[7])],L1).
```

La dernière ligne montre comment utiliser le prédicat `emploi`. On donne le nombre de créneaux, le nombre de cours, le nombre de cours max. par créneaux, la liste des contraintes de préséance et la liste des contraintes de simultanéité (données comme des paires cours et liste de cours qui ne peuvent pas être en parallèle et le dernier argument contiendra le résultat.

Exercice 5

Solution pour l'exemple :

```
inshi(L) :- L = [X11,X21,X31,X41,X51,
                 X12,X22,X32,X42,X52,
                 X13,X23,X33,X43,X53,
                 X14,X24,X34,X44,X54,
                 X15,X25,X35,X45,X55],
             fd_domain(L,1,9),
             fd_all_different([X11,X12,X13,X14,X15]),
             fd_all_different([X21,X22,X23,X24,X25]),
             fd_all_different([X31,X32,X33,X34,X35]),
             fd_all_different([X41,X42,X43,X44,X45]),
             fd_all_different([X51,X52,X53,X54,X55]),
             fd_all_different([X11,X21,X31,X41,X51]),
             fd_all_different([X12,X22,X32,X42,X52]),
             fd_all_different([X13,X23,X33,X43,X53]),
             fd_all_different([X14,X24,X34,X44,X54]),
             fd_all_different([X15,X25,X35,X45,X55]),
             X11 #= 5, X31*X41 #= 2, X32*X42 #= 4, X33 #= 4,
             X43*X53 #= 15, X34*X44 #= 15, X15*X25*X35*X45 #= 120,
             X21*X22 #= 15, X51*X52 #= 12, X12*X13*X14 #= 8,
             X23*X24 #= 2, X54*X55 #= 2,
             fd_labeling(L).
```

Solution générale :

```
% inshi(Width,Height,Horizontal,Vertical,L)
% Horizontal is the list of horizontal rectangles, given as 4-tuples with
% starting point coordinates, size, the number inside
% Vertical is the list of vertical rectangles (starting point is top)
```

```
% Example:
% inshi(5,5,[(1,1,1,5),(3,1,2,2),(3,2,2,4),(3,3,1,4),(4,3,2,15),
%            (3,4,2,15),(1,5,4,120)],
%           [(2,1,2,15),(5,1,2,12),(2,3,2,2),(5,4,2,2)],L).

extract(L,L,0,[]) :- !.
extract([X|L],L2,Width,[X|L1]) :- W is Width-1, extract(L,L2,W,L1).


% generates all_different constraints for rows
rows([],_) :- !.
rows(Vars,Width) :-
    extract(Vars,Vars1,Width,L), fd_all_different(L), rows(Vars1,Width).

extract1([],[],_,_,[]) :- !.
extract1([X|L],L1,1,W,[X|L2]) :- !,extract1(L,L1,2,W,L2).
extract1([X|L],[X|L1],W,Width,L2) :- W =< Width, !,W1 is W+1,
        extract1(L,L1,W1,Width,L2).
extract1(L,L1,W,Width,L2) :- W > Width, extract1(L,L1,1,Width,L2).

% generates all_different constraints for columns
columns([],_) :- !.
columns(Vars,Width) :-
    extract1(Vars,Vars1,1,Width,L), fd_all_different(L), W is Width-1,
    columns(Vars1,W).


% generates constraints for rectangles (vertical or horizontal depending on Mode
rectangles(_,_,[],_).
rectangles(Mode,Vars,[(X,Y,Size,Number)|L],Width) :-
            rect(Mode,Vars,X,Y,Size,Number,Width),
            rectangles(Mode,Vars,L,Width).

rect(Mode,Vars,X,Y,Size,Number,Width) :-
            getvariables(Mode,Vars,X,Y,Size,Width,LV),
            mult(LV,Number).

getvariables(_,_,_,_,0,_,[]) :- !.
getvariables(ver,Vars,X,Y,Size,Width,[E|L1]) :-
            S is Size - 1, Y1 is Y+ 1,
            element(Vars,X,Y,Width,E),
            getvariables(ver,Vars,X,Y1,S,Width,L1).
getvariables(hor,Vars,X,Y,Size,Width,[E|L1]) :-
            S is Size - 1, X1 is X+1,
            element(Vars,X,Y,Width,E),
            getvariables(hor,Vars,X1,Y,S,Width,L1).

element(L,X,Y,Width,E) :- N is (Y-1)*Width+X, nth(N,L,E).

mult(LV,Number) :- term(LV,T), T #= Number.

term([X],X).
term([X|L],X*T) :- term(L,T).

printline(L,0,L).
printline([X|L],W,L1) :- write(X),W1 is W-1,printline(L,W1,L1).
```

```
show([],_).
show(Vars,Width) :- printline(Vars,Width,Vars1), nl, show(Vars1,Width).

inshi(Width,Height,Horizontal,Vertical,Vars) :-
            NVars #= Width*Height,
            length(Vars,NVars),
            fd_domain(Vars,1,5),
            rows(Vars,Width),
    columns(Vars,Width),
            rectangles(hor,Vars,Horizontal,Width),
            rectangles(ver,Vars,Vertical,Width),
            fd_labeling(Vars),
            show(Vars,Width).
```