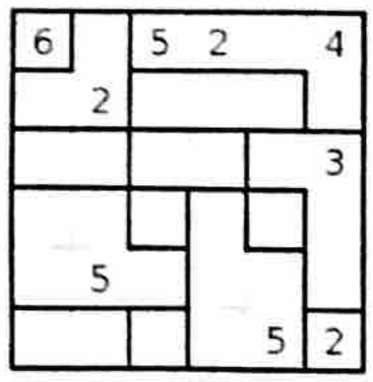
Université Paris Diderot - Sorbonne Paris Cité - Master 1 Informatique -
Programmation logique par contraintes
Examen du 9 janvier 2018 - Durée : 2 heures

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

Exercice 1 *Programmation* (**5 points**)
On considère le jeu suivant.



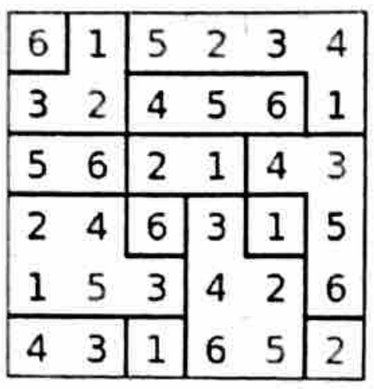


FIGURE 1 – Une grille et sa solution

Chaque case de la grille doit être remplie par un chiffre entre 1 et 6 de sorte que dans chaque colonne et chaque ligne il y ait tous les chiffres entre 1 et 6. En plus, les chiffres inscrits dans une zone doivent être tous différents et doivent former une suite de chiffres consécutifs (4-3-5-6 et 6-5-4-3 sont par exemple légales, par contre 2-3-4-6 est illégal).

— Écrivez un programme ECLiPSe CLP pour résoudre la grille de l'exemple.
Indications : Écrivez d'abord un prédicat auxiliaire pour décrire la dernière contrainte pour une zone donnée par une liste de variables. On peut entre autres utiliser le maximum et le minimum d'une liste ainsi que sa longueur. En effet, par exemple pour la liste `[4,3,5,6]` on a que le maximum moins le minimum est égal à 3 (la longueur de la liste moins un). On a la même chose pour `[6,5,4,3]` mais pas pour `[2,3,4,6]`. En ECLiPSe CLP (avec la librairie `ic`) on peut utiliser les fonctions `ic:min(L,Min)` (resp. `ic:max(L,Max)`) qui contraignent `Min` (resp. `Max`) à être le minimum (resp. le maximum) des variables (pas forcément instantiées) de la liste `L` ainsi que le prédicat `length(L,Length)` qui étant donné une liste `L` donne dans `Length` sa longueur. Par exemple `ic:min([4,3,5,6],Min)` donne `Min = 3`.

Exercice 2 *Consistance* (**5 points**)
On considère la contrainte $X+Y=3Z+1 \wedge Y+U \leq Z$ avec les domaines (donnés comme intervalles) $D(U) = [0...3]$, $D(X) = [4...6]$, $D(Y) = [2...4]$ et $D(Z) = [2...4]$.

— Rendez la contrainte **borne-consistante**.
— Est-ce que la contrainte a une solution ? Si oui, donnez-en une.

Exercice 3 *Simplex* (**7 points**)
On considère le problème suivant :
Minimiser $-X - Y$ par rapport à $X \geq 0, Y \geq 0$ et

$$\begin{aligned} 4 * Y &\leq X + 8 \\ 6 &\geq Y + 2 * X \end{aligned}$$

— Visualisez le problème en dessinant un plan (axes : X et Y) avec les contraintes. Soyez précis.
— Appliquez l'algorithme simplex (Il est simple d'obtenir une forme simplex de base).
— Pour quelles valeurs de X et Y le minimum est-il atteint ?
— Donnez une requête en ECLiPSeCLP permettant d'obtenir le minimum.
— Si on change la fonction objective (ici $-X - Y$) à minimiser, est-ce que le problème à toujours au moins une solution ? Justifiez.
— Donnez une fonction objective (à la place de $-X - Y$) à minimiser de sorte que le minimum soit atteint à plusieurs endroits.
— Dans le problème original on impose en plus que X et Y doivent être des entiers. Quel est le minimum dans ce cas ? Donnez la réponse à partir du dessin (sans calcul).

Exercice 4 *Programmation* (**5 points**)
Une machine à café a dans sa réserve P_2 pièces de 2 Euros, P_1 pièces de 1 Euro, P_{50} pièces de 50 centimes, P_{20} pièces de 20 centimes et P_{10} pièces de 10 centimes. L'utilisateur insère des pièces de monnaie pour un total de T centimes pour une boisson dont le prix est de P centimes (T et P sont des multiples de 10).

— Écrivez en ECLiPSe CLP un prédicat `monnaie(T,P,[P2,P1,P50,P20,P10],L)` qui étant donné des valeurs pour `T`, `P`, `P2`, etc. donne dans `L` une liste avec le nombre de pièces rendues par la machine. Cette liste `L` contient d'abord le nombre de pièces de 2 Euros, ensuite le nombre de pièces de 1 Euro, etc. Par exemple `monnaie(200,130,[3,4,5,2,3],L)` donne successivement `L = [0,0,0,2,3]`, `L = [0,0,1,0,2]` et `L = [0,0,1,1,0]`.

Pour pouvoir utiliser le prédicat d'une façon plus générale (avec un nombre variable de pièces de valeur différentes), on décide de donner en entrée aussi explicitement les valeurs des pièces en centimes. Une pièce de valeur P_v et le nombre N_v de cette pièce dans la réserve est représenté par $P_v : N_v$ et on donne en entrée au prédicat une liste de ces couples. En sortie, on produit une liste de couples de la forme $P_v : R_v$ qui signifie qu'on rend R_v pièces de la valeur P_v.

— Écrivez en ECLiPSe CLP le prédicat `monnaie2(T,P,LP,L)` où `T`, `P` sont comme pour `monnaie` tandis que `LP` est une liste de couples $P_v : N_v$ donnée. La taille de la liste `LP` n'est pas toujours la même ! Exemples d'utilisation : `monnaie2(200,130,[200:3,100:4,50:5,20:2,10:3],L)` donne successivement `L=[200:0,100:0,50:0,20:2,10:3]`, `L=[200:0,100:0,50:1,20:0,10:2]` ainsi que `L=[200:0,100:0,50:1,20:1,10:0]` tandis que `monnaie2(70,35,[5:3,10:4],L)` donne successivement `L=[5:1,10:3]` et `L=[5:3,10:2]`.