Option Informatique en Spé MP et MP*
Programmation Caml : le débit de l'eau

Placez au début de votre programme un commentaire indiquant votre nom, votre classe, la date, et le sujet du T.P. Rédigez les réponses aux questions autres que celles de pure programmation sur une feuille de papier séparée : vous n'êtes pas ici pour faire preuve de vos éventuels talents de dactylographe.

▶ Dupont et Dupond se sont perdus dans le désert et leur jeep est en panne. Il leur reste un bidon de huit litres d'eau, ainsi que deux bidons vides dont les capacités respectives sont de cinq et trois litres. Dupont pense qu'il faut marcher vers le nord, mais Dupond penche pour le sud. Ils décident donc de se séparer, mais ils veulent partager l'eau équitablement. Vous allez les aider !

▶ Associons aux trois bidons un triplet (a, b, c) qui indique le nombre de litres d'eau que chacun contient ; au départ, $a = 8$, $b = 0$ et $c = 0$. Une *opération élémentaire* consiste à verser de l'eau d'un bidon x dans un bidon y, jusqu'à ce que y soit plein ou que x soit vide. Par exemple, on peut passer de la situation initiale $i = (8, 0, 0)$ à la situation $(3, 5, 0)$ ou à la situation $(5, 0, 3)$ selon que l'on décide de remplir le deuxième ou le troisième bidon.

▶ Derrière notre problème de partage se profile donc un graphe : ses sommets sont les valeurs possibles de (a, b, c) ; ce graphe est orienté ; un arc mène de (a, b, c) à (a', b', c') si l'on peut passer de la première situation à la deuxième au moyen d'une opération élémentaire. Nous noterons ceci $(a, b, c) \longrightarrow (a', b', c')$.

Question 1 • Combien ce graphe compte-t-il de sommets ?

▶ Notons $s \xrightarrow{*} t$ s'il existe une suite $(e_k)_{0 \leqslant k \leqslant n}$ telle que $x_0 = s$, $x_n = t$ et $x_k \longrightarrow x_{k+1}$ pour $0 \leqslant k < n$.

Question 2 • Montrez que si $i \xrightarrow{*} t$, alors ou bien $t = i$, ou bien $t \longrightarrow i$, ou bien il existe un état t' tel que $t \longrightarrow t'$ et $t' \longrightarrow i$.

▶ Nous définissons les types suivants :

```
type bidon = { maxi : int ; actu : int } ;;
type état = Etat of bidon * bidon * bidon ;;
type graphe = { sommets : état list ; arcs : (état * état) list } ;;
```

Le champ `maxi` nous permet de traiter le problème avec des bidons de capacités quelconques. Nous nous limitons toutefois au cas de trois bidons.

Question 3 • Rédigez en Caml une fonction :

```
graphe_initial : int * int * int * int * int * int -> graphe
```

spécifiée comme suit : `graphe_initial(ma,ca,mb,cb,mc,cc)` crée le graphe (à un seul sommet et sans aucun arc) défini par les capacités (m_a, m_b, m_c) et les contenus initiaux (c_a, c_b, c_c) des trois bidons. Ainsi, le graphe associé à l'état initial i est obtenu avec `graphe_initial(8,8,5,0,3,0)`.

Question 4 • Rédigez en Caml une fonction :

```
opération_élémentaire : bidon * bidon -> bidon * bidon
```

spécifiée comme suit : `opération_élémentaire(x,y)` réalise l'opération élémentaire qui consiste à déverser le bidon x dans le bidon y.

Question 5 • Rédigez en Caml une fonction :

```
successeurs_état : état -> état list
```

spécifiée comme suit : `successeurs_état s` construit la liste des états t tels que $s \longrightarrow t$. Suggestion : définissez un type `position`, union de trois constructeurs constants, pour désigner les trois bidons.

▶ Nous voulons construire la liste des états t tels que $i \xrightarrow{*} t$. Pour ce faire, nous effectuons un *parcours en largeur* du graphe. Ceci consiste à gérer dynamiquement une partition de l'ensemble des sommets en trois parties :

- les sommets élus : ce sont ceux dont nous savons qu'ils appartiennent à t, et dont nous avons déterminé les successeurs ;
- les sommets éligibles : ce sont ceux dont nous savons qu'ils appartiennent à t, mais dont nous n'avons pas encore déterminé les successeurs ;
- les autres sommets : ce sont ceux dont nous n'avons pas encore eu connaissance ; parmi eux se trouvent les sommets inaccessibles depuis i.

Au départ, aucun sommet n'est élu, et i est le seul sommet éligible.

▶ L'ensemble des sommets éligibles est considéré comme une file d'attente. Lorsque cette file d'attente est vide, la liste est contruite ; sinon, on extrait un sommet de la file et on le range avec les élus ; on détermine ensuite ses successeurs avec la fonction `successeurs_état`, et on place dans la file d'attente ceux qui n'ont pas été déjà vus.

Question 6 • Rédigez en Caml une fonction :

```
élection : graphe -> état -> graphe * état list
```

spécifiée comme suit : `élection g t` rend un couple (g', r) où g' est le graphe déduit de g par élection du sommet t, et r est la liste des successeurs de t qui n'apparaissent pas dans g.

Question 7 • Rédigez en Caml une fonction :

```
construire_graphe : int * int * int * int * int * int -> graphe
```

spécifiée comme suit : `construire_graphe(ma,ca,mb,cb,mc,cc)` construit le graphe défini par les capacités (m_a, m_b, m_c) et l'état initial $i = (c_a, c_b, c_c)$.

Question 8 • Le partage équitable est-il possible ?

▶ Vous noterez que le champ `arcs` n'a pas servi. En fait, nous avons seulement construit la liste des sommets accessibles à partir de l'état initial, ce qui permet de répondre au *problème de décision.* Pour réaliser effectivement le partage, il faut disposer des arcs, afin de déterminer un chemin menant de i à l'état visé.

Question 9 • En fait, le graphe obtenu est petit, vous pouvez donc le dessiner. Déterminez alors un chemin menant de i à l'état visé.

Question 10 • Quelle(s) propriété(s) intéressante(s) possède ce graphe ?

Question 11 • Certains états ne sont pas accessibles en partant de i ; c'est le cas par exemple de $(5, 1, 2)$. Énumérez ces états inaccessibles ; quelle propriété ont-ils en commun ?

FIN