Licence STS, semestre 4 — 2015–16

Mathématiques pour l'Informatique 2 — 4 mai 2016

`http://www.lri.fr/~paulin/MathInfo2`

Examen session 1 - 21 mai 2015

L'examen dure 2 heures. L'énoncé est composé de 4 pages. Toutes les réponses devront être clairement justifiées.

Le seul document autorisé est une page A4 manuscrite recto-verso.

Les copies doivent être **cachetées**. Reporter le numéro d'anonymat sur les feuilles intercalaires et les numéroter à l'emplacement prévu.

Exercice 1 *(Ordre, 1 point).*

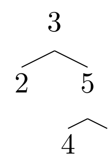
On se place sur l'ensemble $\mathbb{N} \times \mathbb{N}$ des couples d'entiers.

1. Donner un exemple d'ordre total sur cet ensemble.
2. Donner un exemple d'ordre partiel sur cet ensemble.

Exercice 2 *(Arbres, 5 points).*

On rappelle que la signature des arbres binaires contenant des entiers est formée d'une constante `leaf` et d'un symbole ternaire `node` d'arité $\texttt{tree} \times \mathbb{N} \times \texttt{tree} \to \texttt{tree}$.

Par exemple, l'arbre



est représenté par le terme node(node(leaf,2,leaf),3,node(node(leaf,4,leaf),5,leaf))

On définit de manière récursive sur la structure de l'arbre une fonction booléenne $\texttt{in}(n, t)$ qui renvoie `vrai` exactement si l'entier n se trouve dans l'arbre t. Les équations sont

$$\texttt{in}(n, \texttt{leaf}) = \texttt{faux} \qquad\qquad \texttt{in}(n, \texttt{node}(g, x, d)) = (x = n) \vee \texttt{in}(n, g) \vee \texttt{in}(n, d)$$

On s'intéresse aux arbres binaires de recherche, c'est-à-dire aux arbres qui vérifient que dans chaque sous-arbre de la forme $\texttt{node}(g, n, d)$, tous les entiers stockés dans le sous-arbre gauche g sont inférieurs ou égaux à n, et n est lui-même inférieur ou égal à tous les entiers stockés dans le sous-arbre droit d. On note $\texttt{abr}(t)$ la propriété qui est vraie lorsque t est un arbre binaire de recherche. Les deux propriétés suivantes sont vérifiées :

$$\texttt{abr}(\texttt{leaf})$$
$$\texttt{abr}(\texttt{node}(g, n, d)) \Leftrightarrow (\texttt{abr}(g) \wedge \texttt{abr}(d) \wedge \forall x, (\texttt{in}(x, g) \Rightarrow x \leq n) \wedge (\texttt{in}(x, d) \Rightarrow n \leq x))$$

1. Enoncer le principe de récurrence structurelle associé à la définition des arbres.
2. L'arbre donné en exemple est-il un arbre binaire de recherche ? donner deux autres configurations d'arbre qui contiennent exactement les mêmes entiers, l'une qui est une configuration d'arbre binaire de recherche et l'autre qui ne l'est pas.
3. Lorsque l'on recherche un élément n dans un arbre binaire de recherche de la forme $\texttt{node}(g, x, d)$ est-il utile de regarder à la fois si $\texttt{in}(n, g)$ et si $\texttt{in}(n, d)$? dans quel cas faut-il chercher dans g et dans quel cas dans d ?

4. Proposer une nouvelle définition de la fonction `in`, plus directe, qui recherche un élément dans un arbre binaire de recherche. On notera cette version `inabr`.
5. Montrer par récurrence sur t que $\forall n\, t,\ \texttt{abr}(t) \Rightarrow \texttt{inabr}(n,t) = \texttt{in}(n,t)$.
6. On veut maintenant tester efficacement si un arbre binaire est bien un arbre binaire de recherche. Pour cela on introduit une fonction plus puissante `verif` qui prend en argument deux entiers *min*, *max* et un arbre t et qui doit renvoyer vrai exactement lorsque t est un arbre binaire de recherche et que tous les entiers x dans t sont tels que $min \leq x \leq max$.

   Donner des équations récursives pour définir `verif`, mais attention : on n'utilisera dans les équations ni la fonction `abr` ni les fonctions `in` ou `inabr`, on pourra juste utiliser des comparaisons entre les entiers et la fonction `verif` elle-même.
7. Pour vérifier que votre fonction fait bien ce qui est demandé, montrer par récurrence sur t que $\forall m_1\, m_2\, t,\ \texttt{verif}(m_1, m_2, t) = \texttt{true} \Rightarrow \texttt{abr}(t) \wedge \forall x, \texttt{in}(x,t) \Rightarrow m_1 \leq x \leq m_2$.
8. Comment faut-il choisir *min* et *max* pour tester si t est un arbre binaire de recherche ?

Exercice 3 *(Problème, 16 points)*. Les questions sont largement indépendantes. On pourra admettre les résultats d'une question pour traiter les suivantes.

On considère un jeu dans lequel il y a 9 pions bicolores (une face noire et une face blanche) disposés sur un damier en carré (3 lignes de 3 pions). Le jeu consiste à chaque étape à choisir soit une ligne

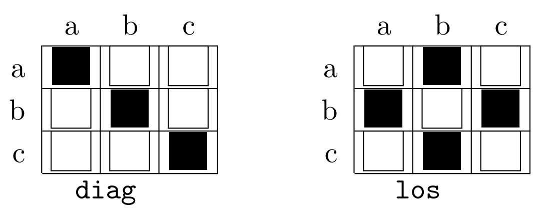


FIGURE 1 – Deux exemples de damier

soit une colonne et à retourner les 3 pions correspondants. Le problème qui nous intéresse est : étant donnés deux damiers, est-il possible de passer de l'un à l'autre par une suite de tels retournements ?

Pour modéliser ce jeu, on introduit un ensemble P de trois éléments notés $\{a, b, c\}$, qui permettent de repérer les lignes et les colonnes du jeu. Une case du damier est donc représentée par un couple $(x, y) \in P \times P$ dans lequel x représente la ligne et y la colonne. On décide de représenter la couleur noire par l'entier 0 et la couleur blanche par l'entier 1. Changer la couleur d'un pion revient à faire une opération notée `flip` définie par $\texttt{flip}(u) \stackrel{\text{def}}{=} 1 - u$.

Une configuration du jeu est représentée par une application qui à chaque case du damier associe la couleur du pion posé sur la case. L'ensemble des configurations est noté C et est défini comme l'ensemble $(P \times P) \to \{0, 1\}$.

La configuration du damier avec tous les pions blancs est notée `blanc`, elle est définie par l'équation $\forall xy \in P,\ \texttt{blanc}(x, y) = 1$

1. Donner le cardinal de l'ensemble des configurations C.
2. Représenter graphiquement le damier `dam` tel que

$$\forall xy \in P,\ x \neq \texttt{a} \Rightarrow \texttt{dam}(x, y) = 1 \quad \text{et} \quad \forall y \in P,\ \texttt{dam}(\texttt{a}, y) = 0$$

3. Donner les équations qui définissent le damier `diag` de la figure 1.

4. On introduit maintenant des applications qui permettent de retourner les pions soit sur une ligne, soit sur une colonne. L'application de retournement de la ligne d'indice $p \in P$ est notée $\mathtt{ligne}_p$. Étant donnée une configuration de damier $s \in C$, $\mathtt{ligne}_p(s)$ calcule une nouvelle configuration du damier dans laquelle les pions sur la ligne d'indice p ont été retournés (et ont donc changé de couleur). On introduit de même pour chaque position p, une application $\mathtt{col}_p \in C \to C$ telle que $\mathtt{col}_p(s)$ représente le damier obtenu à partir du damier de la configuration s en retournant les pions qui sont sur la colonne p. Dans la suite les applications $\mathtt{ligne}_p$ et $\mathtt{col}_p$ sont appelées des *retournements*.
   (a) Dessiner le damier correspondant à la configuration $\mathtt{col}_{\mathtt{c}}(\mathtt{ligne}_{\mathtt{a}}(\mathtt{blanc}))$.
   (b) Donner des équations qui caractérisent la couleur du pion de la configuration $\mathtt{col}_p(s)$ sur la case (x, y) en fonction de la couleur du pion sur la même case dans la configuration s. Faire de même pour $\mathtt{ligne}_p(s)$.
   Aide : Il faut donc relier $\mathtt{col}_p(s)(x, y)$ avec $s(x, y)$ en fonction de p, on pourra utiliser l'application `flip` de changement de couleur.
5. On peut représenter le jeu sous forme de graphe orienté dans lequel les sommets sont les configurations et on a une arête de s à s' si et seulement si on peut passer de s à s' par un unique retournement.
   (a) Montrer que s'il y a une arête de s à s' alors il y en a une de s' à s.
   (b) Dans toute la suite du problème, on considère le graphe non orienté correspondant (on met une arête entre s et s' si on peut passer de l'un à l'autre par un retournement). Quel est le degré des nœuds de ce graphe ?
6. Dans cette question, on montre que l'on peut limiter le nombre de suites de retournements à considérer. Il y a 6 retournements possibles (1 pour chaque ligne et 1 pour chaque colonne), on les note $r_1, r_2, r_3, r_4, r_5, r_6$. Chaque retournement r_i est une application de C dans C, on peut donc les composer. On notera $r_i \circ r_j$ l'application de C dans C qui commence par effectuer le retournement r_j puis le retournement r_i.
   (a) Que se passe-t-il si on applique deux fois le même retournement, c'est-à-dire que vaut $(r_i \circ r_i)(s)$?
   (b) L'ordre dans lequel on applique deux opérations de retournement différentes a-t-il une importance sur la configuration finale ? c'est-à-dire comparer $(r_i \circ r_j)(s)$ avec $(r_j \circ r_i)(s)$.
   (c) Que vaut $(r_1 \circ r_2 \circ r_3 \circ r_4 \circ r_5 \circ r_6)(s)$?
   (d) Déduire des questions précédentes que $(r_1 \circ r_2 \circ r_3 \circ r_4 \circ r_5)(s) = r_6(s)$.
   (e) Déduire des questions précédentes que si on peut, à partir d'une configuration s, arriver à une configuration s', alors on peut le faire en utilisant un sous-ensemble des 5 retournements $\{r_1, r_2, r_3, r_4, r_5\}$, chaque retournement étant utilisé une unique fois.
7. Soit une configuration s, on s'intéresse à la composante connexe de s dans le graphe qui est l'ensemble des configurations s' que l'on peut atteindre à partir de s par une suite de retournements.
   (a) En utilisant les résultats de la question précédente, montrer que la composante connexe de s contient au plus 2^5 configurations distinctes.
   (b) Peut-on, à partir d'une configuration s, arriver à n'importe quelle autre configuration s' du graphe ? Le graphe associé au jeu est-il connexe ?
   (c) En utilisant un argument de cardinalité, montrer que le graphe a au moins 2^4 composantes connexes.

8. On va maintenant définir formellement la relation $\texttt{jeu}(s, s')$ entre les configurations qui représente le fait que l'on peut passer de la configuration s à la configuration s' par une suite (éventuellement vide) de retournements. Cela correspond aussi à l'existence d'un chemin de s à s' dans le graphe associé au jeu.
   (a) Donner des règles d'inférence qui définissent la relation $\texttt{jeu}(s, s')$ en utilisant les applications $\texttt{ligne}_p$ et $\texttt{col}_p$.
   (b) En utilisant les règles précédentes, construire un arbre de dérivation de $\texttt{jeu}(\texttt{blanc}, \texttt{los})$ avec `los` le damier de la figure 1.
   (c) Enoncer le principe d'induction associé à la définition de jeu.
9. On introduit une application `paire` qui étant donnée une configuration s calcule la parité du nombre de coins blancs sur le damier. On pose $\texttt{paire}(s) = (s(\texttt{a}, \texttt{a}) + s(\texttt{a}, \texttt{c}) + s(\texttt{c}, \texttt{a}) + s(\texttt{c}, \texttt{c})) \bmod 2$, c'est-à-dire que $\texttt{paire}(s) = 0$ s'il y a un nombre pair de coins blancs sur le damier et vaut 1 sinon. Par exemple $\texttt{paire}(\texttt{diag}) = (0 + 1 + 1 + 0) \bmod 2 = 0$.
   (a) Montrer par induction sur la definition de `jeu` que $\texttt{jeu}(s, s') \Rightarrow \texttt{pair}(s) = \texttt{pair}(s')$
   (b) En déduire une configuration d qui ne peut pas être atteinte à partir du damier `blanc`.
10. On cherche maintenant à caractériser le nombre de composantes connexes dans le graphe.
   (a) Donner une manière à partir d'une configuration s quelconque d'atteindre par des retournements une configuration s' telle que tous les pions de la ligne `a` et de la colonne `a` de s' sont blancs.
   (b) Combien y a-t-il de configurations différentes pour lesquelles tous les pions de la ligne `a` et de la colonne `a` sont blancs ? en déduire un nombre maximal de composantes connexes dans le graphe, puis en utilisant le résultat de la question 7c le nombre exact.
11. En utilisant les résultats précédents, proposer une manière de décider pour deux configurations s et s' si elles sont ou non dans la même composante connexe.