ULC 214 J. 2034

SESSION 2002

Filière MP (groupes M/MP/MPI)

(Epreuve commune aux ENS de Paris, Lyon et Cachan)

Filières MP et PC (groupe I)

(Epreuve commune aux ENS de Paris et Lyon)

INFORMATIQUE

Durée : 4 heures

L'usage de calculatrices électroniques de poche à alimentation autonome, non imprimantes et sans document d'accompagnement, est autorisé. Cependant, une seule calculatrice à la fois est admise sur la table ou le poste de travail, et aucun échange n'est autorisé entre les candidats.

Tournez la page S.V.P.

Ordres pour la terminaison des programmes et suites de Goodstein

Les différentes parties du problème sont largement indépendantes les unes des autres. Plus précisément, la partie 1 introduit les concepts; à l'exception des questions 8 et 9, la partie 3 est indépendante de la partie 2. Les premières questions (1 à 5) de la partie 4 ne font référence ni à la partie 2, ni à la partie 3. La partie 5 peut être résolue indépendamment des autres.

1 Introduction

En informatique, la terminaison des programmes (c'est-à-dire le fait qu'un programme se termine pour tout valeur d'entrée qu'il peut prendre) est une propriété fondamentale que l'on cherche à démontrer. Dans ce problème, nous allons étudier des relations que l'on appelle «bien fondées» et qui permettent de garantir la terminaison des programmes. En fait, les relations bien fondées que nous examinerons sont puissantes et nous permettront d'aborder un problème de logique intéressant : les suites de Goodstein. Pour cela nous avons besoin de représenter les naturels par des arbres, ce qui nous conduira à de l'arithmétique sur ces représentations.

Ordres Une relation R sur E est *irréflexive* si elle vérifie $(\forall x \in E)\ \neg(x\ R\ x)$. Un *ordre strict* est une relation irréflexive et transitive. Si $\sqsupset$ est un ordre strict, $\sqsupseteq$ est la relation $\sqsupset \cup =$, c'est-à-dire la relation telle que $x \sqsupseteq y$ si et seulement si $x \sqsupset y$ ou $x = y$. Un ordre strict est *total* si pour tout $(x, y) \in E \times E$ on a $x \sqsupset y \vee y \sqsupseteq x$.

Une suite $(x_i)_{i\in\mathbb{N}}$ d'éléments de E telle que $x_0 \sqsupset x_1 \ldots x_n \sqsupset x_{n+1} \ldots$ est dite $\sqsupset$-décroissante. Une relation $\sqsupset$ *est bien fondée* sur E (on dit aussi que $(E, \sqsupset)$ est bien fondé) s'il n'existe pas de suite infinie $\sqsupset$-décroissante d'éléments de E.

S'il existe, le *minimum* d'un ensemble A pour un ordre strict $\sqsupset$ est l'élément $min_{\sqsupset}\ A$ tel que

$$min_{\sqsupset}\ A \in A \quad \& \quad (\forall a \in A)\ a \sqsupseteq min_{\sqsupset}\ A.$$

Si elle existe, la *borne supérieure* d'un ensemble A pour un ordre $\sqsupset$ est

$$sup_{\sqsupset}\ A = min_{\sqsupset}\{x \in E \mid (\forall a \in A)\ x \sqsupseteq a\}.$$

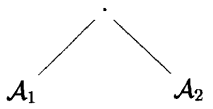
Le *successeur* $s_{\sqsupset}(e)$ d'un élément $e \in E$ est $min_{\sqsupset}\{x \in E \mid x \sqsupset e\}$ si ce minimum existe.

Listes Une *liste* de E est une structure de données informatique qui implante une suite finie d'éléments de E. Si ses éléments sont dans l'ordre $e_1, \ldots, e_n$, elle s'écrit $[e_1; \ldots; e_n]$. On construit les listes à partir de la liste vide $[\,]$ par ajout d'éléments en tête de liste. Le premier élément d'une liste non vide l s'écrit $\mathsf{hd}(l)$, la liste obtenue à partir de l par suppression de son premier élément s'écrit $\mathsf{tl}(l)$. L'ajout en tête de l de l'élément a s'écrit $a :: l$. On a les relations suivantes :

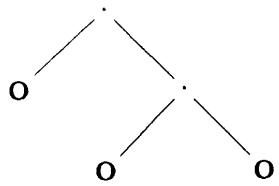
$$\begin{aligned} \mathsf{hd}(a :: l) &= a \\ \mathsf{tl}(a :: l) &= l \end{aligned}$$

tandis que $\mathsf{hd}([\,])$ et $\mathsf{tl}([\,])$ ne sont pas définis.

Arbres binaires Un arbre binaire est soit $\mathbf{O}$, soit $\mathcal{A}_1 \cdot \mathcal{A}_2$ où $\mathcal{A}_1$ et $\mathcal{A}_2$ sont eux-mêmes des arbres binaires. Dans la suite du problème, nous dirons simplement «arbre». Un arbre peut aussi être dessiné



Ainsi l'arbre $\mathbf{O} \cdot (\mathbf{O} \cdot \mathbf{O})$ se dessine :



Les arbres forment l'ensemble $\mathbb{A}$. La *taille* d'un arbre est le nombre d'occurrences de l'opérateur «$\cdot$» qu'il contient. Nous admettrons le principe d'*induction structurelle* sur $\mathbb{A}$:

$$P(\mathbf{O}) \;\&\; (\forall(\mathcal{A}_1, \mathcal{A}_2) \in \mathbb{A} \times \mathbb{A})[P(\mathcal{A}_1) \;\&\; P(\mathcal{A}_2) \Rightarrow P(\mathcal{A}_1 \cdot \mathcal{A}_2)] \quad \Rightarrow \quad (\forall \mathcal{A} \in \mathbb{A}) P(\mathcal{A})$$

qui permet de prouver par récurrence une propriété P pour tous les arbres.

La relation *sur-arbre strict* $\rhd$ est définie par

- $\mathcal{A}_1 \cdot \mathcal{A}_2 \rhd \mathbf{O}$,
- $\mathcal{A}_1 \cdot \mathcal{A}_2 \rhd \mathcal{B}$ si et seulement si $\mathcal{A}_1 = \mathcal{B}$ ou $\mathcal{A}_2 = \mathcal{B}$ ou $\mathcal{A}_1 \rhd \mathcal{B}$ ou $\mathcal{A}_2 \rhd \mathcal{B}$.

On admettra que $\rhd$ est un ordre strict.

Nous généralisons le principe d'induction structurelle aux triplets d'arbres en définissant sur $\mathbb{A}^3$ l'ordre $\rhd\rhd\rhd$ qui dit que $(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \rhd\rhd\rhd (\mathcal{B}, \mathcal{B}', \mathcal{B}'')$ si et seulement si $\mathcal{A} \trianglerighteq \mathcal{B}$, $\mathcal{A}' \trianglerighteq \mathcal{B}'$ et $\mathcal{A}'' \trianglerighteq \mathcal{B}''$ et l'une au moins des inégalités est stricte. Ce principe s'énonce :

$$\begin{gathered} (\forall(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \in \mathbb{A}^3) \\ \{(\forall(\mathcal{B}, \mathcal{B}', \mathcal{B}'') \in \mathbb{A}^3)[(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \rhd\rhd\rhd (\mathcal{B}, \mathcal{B}', \mathcal{B}'') \Rightarrow P(\mathcal{B}, \mathcal{B}', \mathcal{B}'')]\} \Rightarrow P(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \\ \Rightarrow \\ (\forall(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \in \mathbb{A}^3) P(\mathcal{A}, \mathcal{A}', \mathcal{A}''). \end{gathered}$$

Tournez la page S.V.P.

Remarque : Si dans une preuve par induction structurelle, le prédicat P contient plusieurs occurrences d'arbres, le candidat veillera à indiquer clairement sur quelle variable il fait son induction.

Présentation des algorithmes Quand un algorithme est demandé, le candidat n'est pas supposé produire un programme complet, mais un texte suffisamment et clairement documenté pour qu'un programme puisse être extrait sans difficulté. Le candidat pourra aussi bien utiliser un style proche de celui de la figure 1 qu'un style proche du langage de programmation CAML. Les correcteurs attacheront de l'importance au fait que les candidats aient abordé les questions algorithmiques.

1. Montrer qu'un ordre strict est antisymétrique.
2. On considère la fonction $R(\mathcal{A}, l)$ qui prend un arbre $\mathcal{A}$ et une liste l d'arbres $[\mathcal{A}_1; \ldots; \mathcal{A}_n]$ et qui retourne
   – $\mathcal{A}$ si la liste l est vide
   – et $R((\mathcal{A}_1 \cdot \mathcal{A}), [\mathcal{A}_2; \ldots; \mathcal{A}_n])$ si la liste n'est pas vide.
   Formellement,

$$\begin{aligned} R(\mathcal{A}, [\,]) &= \mathcal{A} \\ R(\mathcal{A}, \mathcal{B} :: l) &= R(\mathcal{B} \cdot \mathcal{A}, l) \end{aligned}$$

   Dans la suite, nous utiliserons la fonction $\mathsf{renv}(l)$ définie par $R(\mathbf{O}, l)$.
   Donner un algorithme qui calcule la fonction R.

2 Décomposition complète d'un nombre dans une base et suites de Goodstein

Étant donné un naturel b appelé *base*, l'*exposant maximal* d'un naturel n est le nombre p tel que $b^p \leq n < b^{p+1}$ tandis que la *décomposition complète* de n dans la base b est
– soit 0 si $n = 0$,
– soit la somme de b^d et de d', où d est la décomposition complète de l'exposant maximal p de n dans la base b, et où d' est la décomposition complète de $n - b^p$.

Ainsi la décomposition complète de 83 dans la base 3 est obtenue à partir de

$$3^d + 3^0 + 3^0$$

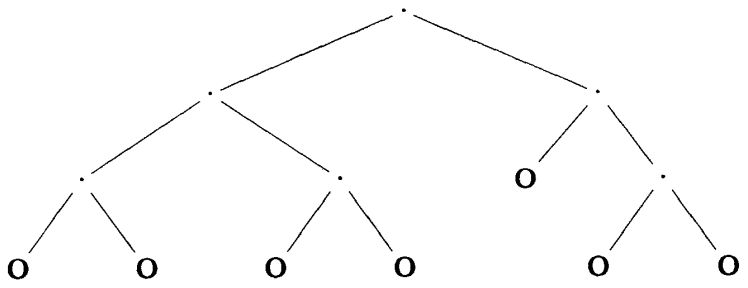
où d est la décomposition complète de 4.

1. Donner la décomposition complète de 4 et celle de 83.
2. Étant donné une base b, montrer que la décomposition complète d'un naturel permet d'exprimer ce naturel à l'aide de l'addition, de l'exponentiation et de 0 et des ces opérations seulement.

3. On associe un arbre à la décomposition complète d'un naturel de la façon suivante :

$$\begin{aligned} \varphi_b(0) &= \mathbf{O} \\ \varphi_b(b^d + d') &= \varphi_b(d) \cdot \varphi_b(d') \end{aligned}$$

Pour le naturel 83 et la base 3, on obtient



Quel est l'arbre associé respectivement à 1, à b, à $b+1$ et à b^b dans toutes les bases ? Soit **dec1** la fonction qui prend un couple de naturels b (une base) et n (un naturel) et retourne un couple de naturels (p, q) tel que

$$n = b^p + q.$$

Donner la définition formelle (à la manière de la fonction R de la question 1.1) de la fonction **arbre** qui associe directement à (b, n) l'arbre donné par φ_b.

4. Dans l'algorithme de la figure 1 qui calcule la fonction **arbre** quelle est la nature (ou le

```
arbre(b, n) =
    pile := [(0,[ ])]
    courant := (n,[ ])
    tant que pile ≠ [ ] faire
        k, liste_res := courant
        si k ≠ 0 alors (m,r) := dec1(b, k)
                       pile := (r, liste_res) :: pile;
                       courant := (m, [ ]))
         sinon (q, l) := hd(pile)
               pile := tl(pile)
               courant := (q, renv(liste_res) :: l))
    fait
k, liste_res := courant
retourne(hd(liste_res))
```

FIG. 1 –

type) des objets contenus dans *pile*, *courant* et *liste_res* ? Que contient *liste_res* dans les étapes de l'algorithme et quand l'algorithme se termine ?

Tournez la page S.V.P.

5. On considère la fonction **erbra** qui prend un couple formé d'une base b et d'un arbre $\mathcal{A}$ et donne le naturel qui correspond à l'arbre $\mathcal{A}$ dans la base b; autrement dit on a $\mathsf{erbra}(b, \mathsf{arbre}(b, n)) = n$. Donner un algorithme récursif qui calcule la fonction **erbra**. Quelle est la complexité de cet algorithme en fonction de la taille de l'arbre, si l'on suppose que la somme et l'exponentiation se font en temps constant ?
6. Les *suites de Goodstein* commencent au rang 2. Si $g_k \neq 0$, g_{k+1} est défini à partir de g_k ainsi :
$$g_{k+1} = \mathsf{erbra}(k+1, \mathsf{arbre}(k, g_k)) - 1.$$
Si $g_3 = 83$, quels sont g_2 et g_4 ? Montrer que pour la suite où $g_3 = 83$ il existe un k tel $g_k > 1000^{1000}$.
7. Donner un algorithme directement dérivé des algorithmes précédents qui prend deux arbres $\mathsf{arbre}(b, m)$ et $\mathsf{arbre}(b, n)$ et construit l'arbre $\mathsf{arbre}(b, m+n)$. Cet algorithme pourra utiliser directement l'addition sur les naturels. Nous appellerons l'opération que cet algorithme implante l'*addition des arbres en base b*. Donner ensuite un algorithme qui permet d'additionner des arbres en base b en n'utilisant que des additions sur des nombres inférieurs à b.
8. Donner un algorithme qui prend deux arbres $\mathsf{arbre}(b, m)$ et $\mathsf{arbre}(b, n)$ et construit l'arbre $\mathsf{arbre}(b, m \cdot n)$ où $m \cdot n$ est le produit des naturels m et n et qui n'utilise que des opérations sur des nombres inférieurs à b.

3 Un ordre sur les arbres

La relation $\succ$ est définie par
- $\mathcal{A}_1 \cdot \mathcal{A}_2 \succ \mathbf{O}$,
- $\mathcal{A}_1 \cdot \mathcal{A}_2 \succ \mathcal{A}'_1 \cdot \mathcal{A}'_2$ si et seulement si
  - ou bien $\mathcal{A}_1 \succ \mathcal{A}'_1$
  - ou bien $\mathcal{A}_1 = \mathcal{A}'_1$ et $\mathcal{A}_2 \succ \mathcal{A}'_2$

1. Montrer que $\succ$ est un ordre strict.
2. Montrer par un contrexemple que $(\mathbb{A}, \succ)$ n'est pas bien fondé.
3. Si $(A, >_A)$ et $(B, >_B)$ sont des ordres stricts bien fondés, on définit $(A \times B, >_A \times >_B)$ par
$$(a, b) >_A \times >_B (a', b') \text{ si } a >_A a' \text{ ou } a = a' \text{ et } b >_B b'.$$
Montrer que $(A \times B, >_A \times >_B)$ est un ordre strict bien fondé.
4. Soit A un alphabet muni d'une relation d'ordre $>_{\mathsf{A}}$. Un mot $a_1 \dots a_n$ sur A^* est *décroissant* si pour $1 \leq i < n$ on a $a_i \geq_{\mathsf{A}} a_{i+1}$. Les mots décroissants forment l'ensemble A^{dec}.
L'ordre du dictionnaire sur les mots est défini par
- $a\alpha >_{\mathsf{A}^*} \varepsilon$ (où ε est le mot vide).
- $a\alpha >_{\mathsf{A}^*} b\beta$ si et seulement si $a >_{\mathsf{A}} b$ ou bien $a = b$ et $\alpha >_{\mathsf{A}^*} \beta$.
Montrer que $(\mathsf{A}^{dec}, >_{\mathsf{A}^*})$ est bien fondé si et seulement si $(\mathsf{A}, >_{\mathsf{A}})$ est bien fondé.

5. Un arbre $\mathcal{A}_1 \cdot (\mathcal{A}_2 \cdot \mathcal{A}_3)$ est *ordonné* si $\mathcal{A}_1$ et $\mathcal{A}_2 \cdot \mathcal{A}_3$ sont ordonnés et si $\mathcal{A}_1 \succeq \mathcal{A}_2$. De plus, $\mathbf{O}$ est ordonné et $\mathcal{A}_1 \cdot \mathbf{O}$ est ordonné si $\mathcal{A}_1$ est ordonné. Les arbres ordonnés forment l'ensemble $\mathbb{O}$. Montrer que pour tout arbre $\mathcal{A}$ de $\mathbb{O}$ le successeur $s_\succ(\mathcal{A})$ existe.
6. $\mathcal{A}$ étant un arbre, montrer par induction structurelle que la relation $\succ$ est bien fondée sur l'ensemble $\mathbb{O}_\mathcal{A} = \{\mathcal{B} \in \mathbb{O} \mid \mathcal{A} \succeq \mathcal{B}\}$. *Indication* : on montrera que l'ensemble ordonné $(\mathbb{O}_\mathcal{A}, \succ)$ est isomorphe à un ensemble, ordonné par un ordre du dictionnaire, de mots ordonnés.
7. Conclure de la question 6 que $\succ$ est bien fondée sur $\mathbb{O}$.
8. Montrer pour tous les naturels b, n et m, que d'une part $\mathsf{arbre}(b, n) \in \mathbb{O}$ et que d'autre part, $m > n$ implique $\mathsf{arbre}(b, m) \succ \mathsf{arbre}(b, n)$.
9. Montrer que toute suite de Goodstein atteint 0, autrement dit pour toute suite de Goodstein $g_2, \ldots, g_k, \ldots$, il existe un naturel n, tel que $g_n = 0$.

4 Un autre ordre sur les arbres

On définit sur $\mathbb{A}$ une relation $\blacktriangleright$ par

$$\mathcal{A} \blacktriangleright \mathcal{B} \qquad \text{si et seulement si}$$

- ou bien $\mathcal{A} \neq \mathbf{O}$ et $\mathcal{B} = \mathbf{O}$,
- ou bien $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$ et $\mathcal{B} = \mathcal{B}_1 \cdot \mathcal{B}_2$ et l'une des trois conditions suivantes est satisfaite :
  1. $\mathcal{A}_1 \blacktriangleright \mathcal{B}_1$ et $\mathcal{A} \blacktriangleright \mathcal{B}_2$,
  2. $\mathcal{A}_1 = \mathcal{B}_1$ et $\mathcal{A}_2 \blacktriangleright \mathcal{B}_2$,
  3. $\mathcal{A}_2 \blacktriangleright \mathcal{B}$ ou $\mathcal{A}_2 = \mathcal{B}$.

1. Montrer que $\blacktriangleright$ est un ordre strict total sur $\mathbb{A}$.
2. Montrer que $\mathcal{A} \rhd \mathcal{B}$ implique $\mathcal{A} \blacktriangleright \mathcal{B}$. Autrement dit que si $\mathcal{A}$ est un sur-arbre strict de $\mathcal{B}$ alors $\mathcal{A} \blacktriangleright \mathcal{B}$.
3. Donner un algorithme qui étant donnés deux arbres $\mathcal{A}$ et $\mathcal{B}$ retourne **vrai** si $\mathcal{A} \blacktriangleright \mathcal{B}$ et **faux** sinon.
4. Montrer que pour tout élément de $\mathbb{A}$ le successeur existe. Peut-on exprimer $s_\blacktriangleright(\mathcal{A})$ à partir de $\mathcal{A}$, «$\cdot$» et $\mathbf{O}$?
5. Pour prouver que $\blacktriangleright$ est bien fondée sur $\mathbb{A}$, on va raisonner par l'absurde. S'il existe une suite infinie $\blacktriangleright$-décroissante, il en existe une que nous notons $(\mathcal{A}_i)_{i\in\mathbb{N}}$ et qui est plus petite que les autres au sens suivant :
   - $\mathcal{A}_0$ est un plus petit arbre *par la taille*, qui commence une suite infinie $\blacktriangleright$-décroissante.
   - Si l'on suppose $\mathcal{A}_0, \ldots, \mathcal{A}_i$ construits, $\mathcal{A}_{i+1}$ est un plus petit arbre *par la taille* en $(i+1)$-ème position pour une suite infinie $\blacktriangleright$-décroissante qui commence par $\mathcal{A}_0, \ldots, \mathcal{A}_i$.

   Montrer que cette suite ne contient pas $\mathbf{O}$. Montrer qu'on peut construire une suite infinie décroissante plus petite que la suite $(\mathcal{A}_i)_{i\in\mathbb{N}}$, d'où une contradiction.

Tournez la page S.V.P.

6. Montrer que l'identité est une application croissante de $(\mathbb{O}, \succ)$ vers $(\mathbb{A}, \blacktriangleright)$.
7. Déduire de la question précédente que la bonne fondation de $(\mathbb{A}, \blacktriangleright)$ implique la bonne fondation de $(\mathbb{O}, \succ)$. *On a ainsi une nouvelle démonstration de la bonne fondation de* $(\mathbb{O}, \succ)$.
8. On considère la fonction croissante $\psi : (\mathbb{O}, \succ) \longrightarrow (\mathbb{A}, \blacktriangleright)$ qui satisfait les conditions suivantes :

$$\begin{aligned} \psi(\mathbf{O}) &= \mathbf{O} \\ \psi(s_{\succ}(\mathcal{A})) &= s_{\blacktriangleright}(\psi(\mathcal{A})) \\ \psi(sup_{\succ}\ M) &= sup_{\blacktriangleright}\ \psi(M). \end{aligned}$$

La dernière identité doit se lire
« si $sup_{\succ}\ M$ existe, alors $sup_{\blacktriangleright}\ \psi(M)$ existe et $\psi(sup_{\succ}\ M) = sup_{\blacktriangleright}\ \psi(M)$ ».
Que valent les quantités suivantes ?

(a) $\psi((\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O})$?
(b) $\psi(((\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}) \cdot \mathbf{O})$?
(c) $\psi((((\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}) \cdot \mathbf{O}) \cdot \mathbf{O})$?

5 Un ordre sur les mots

Dans cette partie, on va étudier une catégorie d'ordres bien fondés qui possèdent certaines propriétés ainsi qu'un ordre sur les mots.

Une suite $(x_i)_{i\in\mathbb{N}}$ dans laquelle il existe i et j tels que $i < j$ et $x_i \leq x_j$ est dite *bonne*.

Une *sous-suite* de $(x_i)_{i\in\mathbb{N}}$ est donnée par une application $\varphi : \mathbb{N} \to \mathbb{N}$ croissante, autrement dit la sous-suite est celle des $(x_{\varphi(i)})_{i\in\mathbb{N}}$. Une suite $(y_i)_{i\in\mathbb{N}}$ est *faiblement croissante* si $i < j \quad \Rightarrow \quad y_i \leq y_j$. Une *antichaîne* de $(E, >)$ est un sous ensemble A de E tel que pour tout $(x, y) \in A^2$ on a $x \geq y \quad \Rightarrow \quad x = y$.

1. Montrer que les quatre propriétés suivantes sur un ordre strict $(E, >)$ sont équivalentes.
   (a) Tout ordre strict $\sqsupset$ qui satisfait $(\forall (x, y) \in E^2)\ (x > y \Rightarrow x \sqsupset y)$ est bien fondé.
   (b) $(E, >)$ est bien fondé et sans antichaîne infinie.
   (c) Toute suite est bonne,
   (d) De toute suite $(x_i)_{i\in\mathbb{N}}$ on peut extraire une sous-suite faiblement croissante.

   Un ordre qui satisfait ces conditions équivalentes est dit un*bel ordre.*
2. On définit l'ordre $\gg$ sur A^* qui étend l'ordre $>_{\mathsf{A}}$ sur A.
   – $\alpha \gg \varepsilon$,
   – $a \geq_{\mathsf{A}} b$ et $\alpha \gg \beta$ impliquent $a\alpha \gg b\beta$,
   – $\alpha \geqq \beta$ implique $a\alpha \gg \beta$.

   Montrer que $(\mathsf{A}^*, \gg)$ est un bel ordre si et seulement si $(\mathsf{A}, >_{\mathsf{A}})$ est un bel ordre. *Indication* : On pourra utiliser une technique de démonstration qui s'inspire de celle de la question 4.5.