Autour des suites de Goodstein et de la terminaison des programmes

Corrigé

juillet 2002

Préambule

Ce sujet étudie des propriétés des arbres binaires et des mots. Il traite de représentation des ordinaux par des arbres binaires sans jamais les nommer. L'idée d'utiliser les ordinaux pour la preuve de correction totale (la terminaison) des programmes émerge chez Turing en 1947, c'est-à-dire dès la préhistoire de l'informatique (F. L. Morris et C. B. Jones An Early Program Proof by Alan Turing, *Annals of the History of Computing*, 6 (2) (1984) pp. 139-143). On pourra aussi lire F. Baader and T. Nipkow, *Term Rewriting and All That*, Cambridge University Press, (1998) où des généralisations des ordre étudiés ici sont utilisés pour prouver la terminaison de systèmes de récriture. Ces ordres peuvent aussi servir à prouver la terminaison des programmes.

Dans son livre Triangles de pensées *Édition Odile Jacob*, (2000), (p. 26) Alain Connes présente les suites de Goodstein (R.L. Goodstein, On the restricted ordinal theorem, *J. Symbolic Logic* 9 (1944) pp. 33-41). Pour lui, elles illustrent la fable du lièvre et de la tortue, comme le montrent les questions 2.6 et 3.9 où l'on voit que les suites évoluent extrêmement lentement vers 0. L'intérêt des suites de Goodstein est que leur terminaison ne peut pas être démontrée dans l'arithmétique de Peano (L. Kirby and J. Paris, *Accessible independence results in Peano arithmetic*, *Bull. London Math. Soc.* 14 (1982) pp. 285-93). Il faut donc utiliser des quantifications du second ordre, c'est-à-dire des quantifications sur des ensembles ou sur des suites, comme dans la question 3.5, ce qui ne pose pas de difficultés. Par ailleurs, le sujet étudie une représentation de très grands entiers naturels et l'implantation de l'addition et le multiplication dans cette représentation. Le résultat de la question 5.2 est connu sous le nom de théorème de Higman.

1 Introduction

1. Il faut montrer que $x \sqsupset y \ \& \ y \sqsupset x$ *implique* $x = y$ or on n'a jamais $x \sqsupset y \ \& \ y \sqsupset x$, car sinon on aurait par transitivité $x \sqsupset x$, ce qu'interdit l'irréflexivité ; d'où l'implication.
2. Supposons que les arbres sont définis par le type

   ```
   type arbre = o  | F of arbre*arbre
   ```

   Ainsi `o` est **O** et `F(A1,A2)` représente $\mathtt{A1} \cdot \mathtt{A2}$. Voici une version CAML :

```
let rec R A = function
     [] -> A |
     B :: l -> R (F(B,A)) l
```

Un programme itératif est :

R($\mathcal{A}$*, l) =*
  $\mathcal{B}$ *:=*$\mathcal{A}$
  tant que *l* $\neq$ *[ ]* ***faire***
    $\mathcal{B}$ *:=* $\mathcal{B}\cdot$ *hd(l)*
    l := tl(l)
  fait
  retourne($\mathcal{B}$*)*

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

1. La décomposition complète de 4 en base 3 est $3^{3^0}+3^0$ et celle de 83 est

$$3^{3^{3^0}+3^0}+3^0+3^0.$$

2. Montrons que la décomposition complète $\delta(n)$ de n s'exprime uniquement par addition, exponentiation et 0. (On suppose ici que $b \geq 2$). Cela se fait par récurrence généralisée[1] sur n. Si $n = 0$ alors la décomposition est 0 et c'est clair. Sinon supposons que $n > 0$ et que pour tout $m < n$ la décomposition $\delta(m)$ de m s'exprime uniquement par addition, exponentiation et 0. Alors la décomposition de n est $\delta(n) = b^{\delta(d)} + \delta(n - b^d)$. Par récurrence, puisque $d < n$ et $n - b^p < n$, on peut affirmer que $\delta(d)$ et $\delta(n-b^d)$ s'expriment uniquement par addition, exponentiation et 0 et donc aussi $\delta(n)$ qui est obtenue par une somme et une exponentiation à partir de ces deux décompositions.

3. Les arbres associés à 1, b, $b+1$ et b^b sont donnés par la figure 1.
   La fonction arbre se définit ainsi[2]
   $\mathsf{arbre}(0) := \mathbf{O}$
   $\mathsf{arbre}(n) := F(\mathsf{arbre}(k), \mathsf{arbre}(r)) \quad$ si $n > 0$ et $(k, r) = \mathsf{dec1}(b, n)$

4. Dans son étape principale, l'algorithme décompose par dec1 un naturel k en deux naturels m et r ; la décomposition de r, n'est pas calculée immédiatement et le nombre r ainsi que la liste de résultats intermédiaires déjà calculés sont mis de coté (sur *pile*) pour un calcul ultérieur. m est le nouveau naturel pour lequel on calcule la décomposition.
   - *liste_res* contient une liste d'arbres qui sont les arbres calculés au cours des décompositions dec1 déjà effectuées.
   - *courant* est un couple constitué d'un naturel dont il faut calculer la décomposition et d'une liste d'arbres du type de *liste_res*.

[1] C'est-à-dire que l'étape de récurrence consiste à prouver que si pour tous les $m < n$ on a $P(m)$ alors on a $P(n)$.

[2] Pour préciser la définition de l'énoncé nous affirmons que $\mathsf{dec1}(n) = (p, q)$ tels que $n = b^p + q < b^{p+1}$.

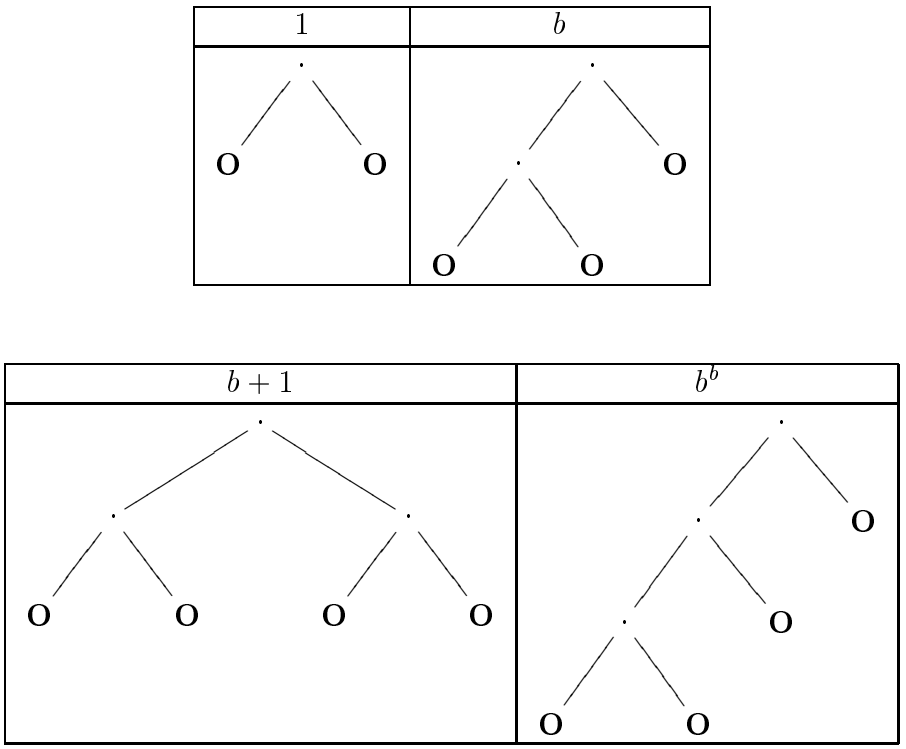


FIG. 1 – Les arbres correspondant aux nombres 1, b, $b+1$ et b^b.

– *pile* contient une liste de couples du même type que celui de *courant*.
Si n est un entier dont la décomposition complète s'écrit

$$\delta(b,n) = b^{\delta(b,n_1)} + \ldots + b^{\delta(b,n_k)}$$

avec $n_1 \geq \ldots \geq n_k$, au cours de l'exécution du programme, la pile passera successivement $k+1$ fois dans un état où elle ne contient qu'un élément. *liste_res* contiendra alors successivement pour chacun de ces passages :

$$[], [arbre(b,n_1)], \ldots [arbre(b,n_k), \ldots, arbre(b,n_1)]$$

Lorsque la pile est vide, *renv* construit à partir de cette liste l'arbre

$$arbre(b,n) = arbre(b,n_1) \cdot (arbre(b,n_2) \cdot \ldots \cdot arbre(b,n_k)) \ldots)$$

qui est alors le seul élément de *liste_res*.
Les sous-arbres de $arbre(b,n)$ qui sont successivement calculés correspondent à un parcours gauche de $arbre(b,n)$ (mais ce n'était pas demandé).
Plus précisément, calculons l'invariant de la boucle (à nouveau, ce n'était pas demandé).
Soit $courant_i = (k^i, L^i)$ le contenu de *courant* après i passages dans la boucle et $pile_i = [(k^i_1, L^i_1), \ldots, (k^i_{m_i}, L^i_{m_i})]$ le contenu de *pile* après i passages dans la boucle. Définissons encore par récurrence sur j la suite C^i_j :

$$\begin{array}{rcl} C^i_0 & = & R(\mathsf{arbre}(b,k^i), L^i) \\ C^i_{j+1} & = & R(\mathsf{arbre}(b,k^i_{j+1}), C^i_{j+1} :: L^i_{j+1}) \quad \text{Si } 0 \leq j \leq m_i - 2 \end{array}$$

```
arbre(b, n) =
1.      pile := [(0,[ ])]
2.      courant := (n,[ ])
3.      tant que pile ≠ [ ] faire
4.          k, liste_res := courant
5.          si k ≠ 0 alors (m,r) := dec1(b, k)
6.                          pile := (r, liste_res) :: pile ;
7.                          courant := (m, [ ]))
8.          sinon (q, l) := hd(pile)
9.                pile := tl(pile)
10.               courant := (q, renv(liste_res) :: l))
11.     fait
12. k, liste_res := courant
13. retourne(hd(liste_res))
```

FIG. 2 – L'algorithme de l'énoncé où les lignes ont été numérotées.

Montrons que $\mathsf{arbre}(b,n) = C^i_{m_i-1}$ pour tout i tel que la pile est non vide.

lorsque $i = 0$, $courant_0 = (n, [])$ et $pile_0 = [(0, [])]$. $C^0_0 = R(\mathsf{arbre}(b,n), []) = \mathsf{arbre}(b,n)$ et $m_0 = 1$ (lignes 1 et 2).

Supposons la propriété vraie au rang i et que $pile_{i+1}$ n'est pas vide. Deux cas se présentent alors (correspondant aux deux cas testés aux lignes 5 et 8) :

– si $k^i \neq 0$, alors soit $(m^{i+1}, r^{i+1}) = \mathsf{dec1}(b, k^{i+1})$ et

$$pile_{i+1} = [(r^{i+1}, L^i), (k^i_1, L^i_1), \ldots, (k^i_{m_i}, L^i_{m_i})]$$

(ligne 6 du programme)

$$courant_{i+1} = (m^{i+1}, [])$$

(ligne 7 d programme). Dans ce cas, $C^{i+1}_0 = R(\mathsf{arbre}(b, m^{i+1}), []) = \mathsf{arbre}(b, m^{i+1})$. Il en résulte que

$$\begin{aligned} C^{i+1}_1 &= R(\mathsf{arbre}(b, r^{i+1}), C^{i+1}_0 :: L^i) \\ &= R(\mathsf{arbre}(b, m^{i+1}) \cdot \mathsf{arbre}(b, r^{i+1}), L^i) \\ &= C^i_0 \end{aligned}$$

Il en résulte que $C^{i+1}_{m_{i+1}-1} = C^i_{m_i-1} = \mathsf{arbre}(b,n)$ par hypothèse de récurrence.

– si maintenant $k^i = 0$, alors soit $(q, l) = (k^i_1, L^i_1)$ (ligne 8),

$$pile_{i+1} = [(k^i_2, L^i_2), \ldots, (k^i_{m_i}, L^i_{m_i})]$$

(ligne 9) et

$$courant_{i+1} = (k^i_1, R(\mathbf{O}, L^i) :: L^i_1)$$

(ligne 10).

$$\begin{aligned} C_0^{i+1} &= R(\mathsf{arbre}(b, k_1^i), R(\mathbf{O}, L^i) :: L_1^i) \\ &= R(\mathsf{arbre}(b, k_1^i), R(\mathsf{arbre}(b, k^i), L^i) :: L_1^i) \\ &= C_1^i \end{aligned}$$

D'où $C_{m_{i+1}-1}^{i+1} = C_{m_i-1}^i = \mathsf{arbre}(b, n)$ par hypothèse de récurrence

Quand la pile devient vide, on passe dans la branche sinon de la boucle : $k^i = 0$, $q = 0$ et $l = []$. Il en résulte que $courant_{i+1} = (0, [R(\mathbf{O}, L^i)])$. Or $C_{m_i-1}^i = C_0^i = R(\mathsf{arbre}(b, k^i), L^i)$. Donc $\mathsf{arbre}(b, n) = R(\mathbf{O}, L^i)$ d'après ce que nous avons vu ci-dessus.

5. Une présentation par équation récursive.

$$\begin{aligned} \mathsf{erbra}(b, \mathbf{O}) &:= 0 \\ \mathsf{erbra}(b, \mathcal{A}_1 \cdot \mathcal{A}_2) &:= b^{\mathsf{erbra}(b, \mathcal{A}_1)} + \mathsf{erbra}(b, \mathcal{A}_2) \end{aligned}$$

Une présentation CAML (où `o` est $\mathbf{O}$ et `F(a1,a2)` représente $\mathcal{A}_1 \cdot \mathcal{A}_2$) est :

```
let rec erbra b a = match a with
      o -> 0 |
      F (a1,a2) -> (exp b (erbra b a1)) + (erbra b a2)
```

La complexité est linéaire, c'est-à-dire que $C(n) = O(n)$, en effet si on suppose que la complexité de l'algorithme exécuté sur un arbre de taille n est $C(n)$ et celle cumulée de la somme et de l'exponentiation est k alors

$$\begin{aligned} C(n+p) &= C(n) + C(p) + k \\ C(1) &= 0 \end{aligned}$$

où n est la taille de $\mathcal{A}_1$ et si p est la taille de $\mathcal{A}_2$. On peut montrer par récurrence que $C(n) = (n-1) \cdot k$. En effet,

$$\begin{aligned} C(n+p) &= C(n) + C(p) + k = (n-1) \cdot k + (p-1) \cdot k + k = (n+p-1) \cdot k \\ C(1) &= (1-1) \cdot k = 0 \end{aligned}$$

6. $g_3 = \mathsf{erbra}(3, \mathsf{arbre}(2, g_2)) - 1$ donc $g_2 = \mathsf{erbra}(2, \mathsf{arbre}(3, g_3 + 1))$. Si $g_3 = 84$, comme la décomposition complète de 84 en base 3 est

$$3^{3^{3^0}+3^0} + 3^{3^0},$$

$$g_2 = 2^{2^{2^0}+2^0} + 2^{2^0} = 8 + 2 = 10.$$

D'autre part, on a :

$$g_4 = 4^{4^{4^0}+4^0} + 4^0 + 4^0 - 1 = 1024 + 1 + 1 - 1 = 1025.$$

On a ensuite

$$g_5 = 5^{5^{5^0}+5^0}$$

et

$$g_6 = 6^{6^{6^0}+6^0} - 1 = 6^7 - 1 = 5\,6^6 + 5\,6^5 + 5\,6^4 + 5\,6^3 + 5\,6^2 + 5\,6 + 5.$$

$g_6 = 6^6 + 233279$ (le seul endroit où la calculatrice peut être utile). Montrons par récurrence sur k que, pour $6 \leq k < 233279 + 6$, $\mathsf{dec1}(k, g_k) = (k, h_k)$ avec $h_k \geq 233285 - k$.

Pour $k = 6$, c'est ce que nous venons de voir. Si c'est vrai pour k, alors

$$g_{k+1} = \mathsf{erbra}(k+1, \mathsf{arbre}(k, g_k)) - 1 \geq (k+1)^{k+1} + 233285 - k - 1$$

et, comme $233285 - k - 1 \geq 0$, $\mathsf{dec1}(k+1, g_{k+1}) = (k+1, h_{k+1})$ avec $h_{k+1} \geq 233285 - k - 1$.

Il en résulte en particulier que $g_{1000} \geq 1000^{1000}$ (et même que $g_{233285} \geq 233285^{233285}$).

7. Un algorithme direct est $\mathsf{arbre}(b, \mathsf{erbra}(b, m) + \mathsf{erbra}(b, n))$.

   Si l'on s'interdit l'addition sur les naturels, la difficulté est de gérer les retenues. On a besoin de définir l'égalité des arbres ainsi que la relation $\succ$ que nous notons `plus_grand` en CAML. [3]

```
let rec egal = function
           (o,o) -> true |
           (o,_) -> false |
           (_,o) -> false |
           (F(a1,a2),F(b1,b2)) -> (egal(a1,b1) || egal(a2,b2));;

let rec plus_grand = function
    o -> (function _ -> false) |
    F(a1,a2) -> function  o -> true |
                         F(b1,b2) -> if (egal(a1,b1))
                                     then (plus_grand a2 b2)
                                     else (plus_grand a1 b1);;
```

   Les arbres égaux le long de l'arête[4] sont regroupés pour former une liste de couples. Chaque couple (n, a) est lui-même formé d'un naturel n et d'un arbre a et sa présence dans la liste signifie que l'arbre a est répété n fois sur l'arête. Si l'arbre représente un nombre en base b, on a $0 < n < b$. On définit deux fonctions de type :

```
regroupe : arbre -> (int * arbre) list
degroupe : (int * arbre) list -> arbre

let rec regroupe = function
    o -> [] |
    F(a1,o) -> [1,a1] |
    F(a1,a2) -> let ar = (regroupe a2) in
                let (n,a) = (hd ar) in
```

[3] L'énoncé ne précisant pas explicitement qu'on ne pouvait pas utiliser la comparaison des grands entiers, une réponse réalisant `plus_grand` par $\mathsf{erbra}(3, \mathcal{A}) > \mathsf{erbra}(3, \mathcal{B})$ aurait été acceptée.

[4] Nous appelons *arête* la liste des arbres $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ d'un arbre $\mathcal{A}_1 \cdot (\mathcal{A}_2 \cdot (\ldots \cdot \mathcal{A}_n)\ldots)$.

```
                    if a1 = a then (n+1, a) :: (tl ar)
                    else (1,a1)::ar;;

    let rec degroupe = function
        [] -> o |
        (1,a) :: l ->  F(a, (degroupe l)) |
        (n,a) :: l ->  F(a, (degroupe ((n-1,a) :: l)));;
```

Nous définissons dans la figure 3 deux fonctions mutuellement récursives que nous appelons `addListe` et `somme`. `somme` réalise l'addition des arbres et `addListe` est une fonction auxiliaire qui réalise l'addition au niveau des listes de couples, obtenues par regroupement comme expliqué ci-dessus.

```
(* addListe:
   int -> (int * arbre) list -> (int * arbre) list -> (int * arbre) list *)

let rec addListe  base l = function
    [] -> l |
    ((n,a) :: l1) as l' -> match l with
        [] -> l' |
        (m ,b) :: l2 -> if (plus_grand a b)
                                then (n,a) :: (addListe  base l l1)
                        else if (plus_grand b a)
                                then (m,b) :: (addListe  base l2 l')
                        else if m+n < base
                                then (m+n,a) :: (addListe  base l2 l1)
                        else if ((m+n) mod base) = 0
                                then ((m+n) / base, (somme base a (F(o,o))))
                                     :: (addListe  base l2 l1)
                        else ((m+n) / base, (somme base a (F(o,o))))
                             :: ((m+n) mod base,a)
                             :: (addListe  base l2 l1)

(* somme: int -> arbre -> arbre -> arbre *)

and somme base a b = (degroupe (addListe base (regroupe a) (regroupe b)))
```

FIG. 3 – `addList` et `somme`

8. On remarque que, si $a_1 = \mathsf{arbre}(b, n_1) = a_{11} \cdot a_{21}$, $a_2 = \mathsf{arbre}(b, n_2) = a_{21} \cdot a_{22}$, $p_1 = \mathsf{erbra}(b, a_{11}), q_1 = \mathsf{erbra}(b, a_{12}), p_2 = \mathsf{erbra}(b, a_{21}), q_2 = \mathsf{erbra}(b, a_{22})$, alors

$$n_1 \cdot n_2 = b^{p_1+p_2} + q_1 \cdot b^{p_2} + q_2 \cdot b^{p_1} + q_1 \cdot q_2.$$

On en déduit le programme de la figure 4 (qui peut être optimisé) :

```
let rec prod base = function
     o -> (function a -> o) |
     F(a11,a12) -> function
          o -> o |
          F(a21, a22) -> let s1 = F(somme base a11 a21,o) and
                             s2 = prod base a12 (F(a21,o)) and
                             s3 = prod base a21 (F(a12,o)) and
                             s4 = prod base a12 a22 in
                    somme base (somme base s1 s2) (somme base s3 s4);;
```

FIG. 4 – Produit de deux arbres en base b

3 Un ordre sur les arbres

1. On montre que $\mathcal{A} \not\succ \mathcal{A}$ par récurrence structurelle sur $\mathcal{A}$: $\neg(\mathbf{O} \succ \mathbf{O})$ et si $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$ par hypothèse de récurrence, $\neg(\mathcal{A}_1 \succ \mathcal{A}_1)$ et $\neg(\mathcal{A}_2 \succ \mathcal{A}_2)$ donc $\neg(\mathcal{A} \succ \mathcal{A})$.
   On montre, par récurrence structurelle sur les triplets d'arbres [5] $(\mathcal{A}, \mathcal{A}', \mathcal{A}'')$ que, si $\mathcal{A} \succ \mathcal{A}'$ et $\mathcal{A}' \succ \mathcal{A}''$, alors $\mathcal{A} \succ \mathcal{A}''$.
   Par définition, pour tout arbre $\mathcal{A}$, $\mathbf{O} \not\succ \mathcal{A}$. Donc si l'un des arbres $\mathcal{A}, \mathcal{A}', \mathcal{A}''$ est $\mathbf{O}$, ce ne peut être que $\mathcal{A}''$. Dans ce cas $\mathcal{A} \succ \mathcal{A}''$ par définition.
   Sinon, $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$ et $\mathcal{A}' = \mathcal{A}'_1 \cdot \mathcal{A}'_2$ et $\mathcal{A}'' = \mathcal{A}''_1 \cdot \mathcal{A}''_2$.
   – Si $\mathcal{A}_1 \succ \mathcal{A}'_1$, alors soit $\mathcal{A}'_1 = \mathcal{A}''_1$, soit $\mathcal{A}'_1 \succ \mathcal{A}''_1$ donc $\mathcal{A}_1 \succ \mathcal{A}''_1$ (dans le deuxième cas par hypothèse de récurrence car $(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \rhd\rhd\rhd (\mathcal{A}_1, \mathcal{A}'_1, \mathcal{A}''_1)$), donc $\mathcal{A} \succ \mathcal{A}''$.
   – Si $\mathcal{A}_1 = \mathcal{A}'_1$ et $\mathcal{A}'_1 \succ \mathcal{A}''_1$ alors $\mathcal{A}_1 \succ \mathcal{A}''_1$, donc $\mathcal{A} \succ \mathcal{A}''$.
   – Si $\mathcal{A}_1 = \mathcal{A}'_1$ et $\mathcal{A}'_1 = \mathcal{A}''_1$ alors $\mathcal{A}_2 \succ \mathcal{A}'_2$ et $\mathcal{A}'_2 \succ \mathcal{A}''_2$ et par hypothèse de récurrence (car $(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \rhd\rhd\rhd (\mathcal{A}_2, \mathcal{A}'_2, \mathcal{A}''_2)$) $\mathcal{A}_2 \succ \mathcal{A}''_2$ et donc $\mathcal{A} \succ \mathcal{A}''$.
2. On peut prendre comme contrexemple $\mathcal{A}_0 = (\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}$ (mais aussi $\mathcal{A}_0 = (\mathcal{B} \cdot \mathcal{C}) \cdot \mathbf{O}$ avec $\mathcal{B}$ et $\mathcal{C}$ quelconques) et $\mathcal{A}_{i+1} = \mathbf{O} \cdot \mathcal{A}_i$. On montre par récurrence sur i que $\mathcal{A}_i \succ \mathcal{A}_{i+1}$
   – $\mathcal{A}_0 \succ \mathcal{A}_1 = \mathbf{O} \cdot ((\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O})$ par $(\mathbf{O} \cdot \mathbf{O}) \succ \mathbf{O}$
   – $\mathcal{A}_{i+1} \succ \mathcal{A}_{i+2}$ par $\mathbf{O} = \mathbf{O}$ et $\mathcal{A}_{i+1} \succ \mathcal{A}_i$ (hypothèse de récurrence).
3. Supposons qu'il existe une suite $>_A \times >_B$-décroissante $(a_i, b_i)_{i\in\mathbb{N}}$. Puisque $(A, >_A)$ est bien fondé, la suite $(a_i)_{i\in\mathbb{N}}$ est stationnaire à partir d'un certain rang j, donc la suite $(b_i)_{i\geq j}$ est $>_B$-décroissante. Contradiction.
4. Si $(\mathsf{A}^{dec}, >_{\mathsf{A}^*})$ est bien fondé alors, en particulier, $(\mathsf{A}, >_{\mathsf{A}^*})$ est bien fondé, où A représente par abus de notation l'ensemble des mots à une lettre. Il est facile de voir que $>_{\mathsf{A}^*}$ coïncide avec $>_{\mathsf{A}}$ sur ces mots à une lettre, d'où le résultat.
   Pour la réciproque, remarquons tout d'abord qu'une suite $>_{\mathsf{A}^*}$-décroissante $(\alpha_i)_{i\in\mathbb{N}}$ ne contient pas ε, elle est donc de la forme $(a_i\beta_i)_{i\in\mathbb{N}}$. Montrons que pour chaque suite $>_{\mathsf{A}^*}$-décroissante $(a_i\beta_i)_{i\in\mathbb{N}}$ il existe une suite $>_{\mathsf{A}^*}$-décroissante $(a'_i\beta'_i)_{i\in\mathbb{N}}$ telle que $a_0 >_{\mathsf{A}} a'_0$ ou bien telle que $a_0 \geq_{\mathsf{A}} a'_0$ et $\|\beta_0\| > \|\beta'_0\|$. En effet, dans la suite $(a_i\beta_i)_{i\in\mathbb{N}}$
   – ou bien il existe un j tel $a_0 >_{\mathsf{A}} a_j$ et la suite $(a'_i\beta'_i)_{i\in\mathbb{N}}$ est la suite $(a_i\beta_i)_{i\geq j}$,

[5] On peut aussi raisonner par induction sur *taille*$(\mathcal{A})$ + *taille*$(\mathcal{A}')$ + *taille*$(\mathcal{A}'')$.

– ou bien tous les a_i sont identiques. Dans ce cas, pour chaque $i \geq 0$, on a $\beta_i \neq \epsilon$. Posons $\beta_i = a'_i\beta'_i$. Puisque α_0 est un mot décroissant on a $a_0 \geq_{\mathsf{A}} a'_0 = a_1$ et $\|\beta_0\| > \|\beta'_0\|$.

Raisonnons maintenant par l'absurde. S'il existe au moins un suite infinie $>_{\mathsf{A}^*}$-décroissante dans A^*, alors il en existe une infinité construite par itération de la constructions ci-dessus. On a donc trouvé une suite $>_{\mathsf{A}} \times >$-décroissante dans $\mathsf{A} \times \mathbb{N}$, ce qui est contraire à la bonne fondation de $(\mathsf{A} \times \mathbb{N}, >_{\mathsf{A}} \times >)$ (qui résulte de la question précédente).

5. On définit le successeur ainsi :

$$\begin{aligned} s_\succ(\mathbf{O}) &= \mathbf{O} \cdot \mathbf{O} \\ s_\succ(\mathcal{A}_1 \cdot \mathcal{A}_2) &= \mathcal{A}_1 \cdot s_\succ(\mathcal{A}_2) \end{aligned}$$

On montre que, si $\mathcal{A} \in \mathbb{O}$, alors $s_\succ(\mathcal{A}) \in \mathbb{O}$ par récurrence structurelle sur $\mathcal{A}$. $s_\succ(\mathbf{O}) = \mathbf{O} \cdot \mathbf{O} \in \mathbb{O}$. De plus $s_\succ(\mathcal{A} \cdot \mathbf{O}) = \mathcal{A} \cdot (\mathbf{O} \cdot \mathbf{O}) \in \mathbb{O}$, parce que $\mathbf{O} \cdot \mathbf{O} \in \mathbb{O}$ et $\mathcal{A} \succeq \mathbf{O}$. et $s_\succ(\mathcal{A}_1 \cdot (\mathcal{A}_2 \cdot \mathcal{A}_3)) = \mathcal{A}_1 \cdot (\mathcal{A}_2 \cdot s_\succ(\mathcal{A}_3)) \in \mathbb{O}$ parce que, par hypothèse de récurrence, $\mathcal{A}_2 \cdot s_\succ(\mathcal{A}_3) = s_\succ(\mathcal{A}_2 \cdot \mathcal{A}_3) \in \mathbb{O}$ et parce que $\mathcal{A}_1 \succeq \mathcal{A}_2$ du fait que $\mathcal{A}_1 \cdot (\mathcal{A}_2 \cdot \mathcal{A}_3) \in \mathbb{O}$

Montrons que $s_\succ(\mathcal{A})$ est bien le successeur : on montre par récurrence structurelle sur $\mathcal{A}$ que si $\mathcal{B} \succ \mathcal{A}$ alors $\mathcal{B} \succeq s_\succ(\mathcal{A})$. Si $\mathcal{A} = \mathbf{O}$ alors $\mathcal{B} \succ \mathbf{O}$. Il en résulte que $\mathcal{B} = \mathcal{B}_1 \cdot \mathcal{B}_2$ et donc ou bien $\mathcal{B}_1 \neq \mathbf{O}$ et alors $\mathcal{B} \succ \mathbf{O} \cdot \mathbf{O}$, ou bien $\mathcal{B}_1 = \mathbf{O}$ et alors puisque $\mathcal{B}_2 \succeq \mathbf{O}$ on a $\mathcal{B} \succeq \mathbf{O} \cdot \mathbf{O}$.

Si $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$ alors $\mathcal{B} \succ \mathcal{A}$ entraine $\mathcal{B} = \mathcal{B}_1 \cdot \mathcal{B}_2$. Ou bien $\mathcal{B}_1 \succ \mathcal{A}_1$ et donc $\mathcal{B} \succ \mathcal{A}_1 \cdot s_\succ(\mathcal{A}_2) = s_\succ(\mathcal{A})$. Ou bien $\mathcal{B}_1 = \mathcal{A}_1$ et $\mathcal{B}_2 \succ \mathcal{A}_2$ et, par hypothèse de récurrence, on a $\mathcal{B}_2 \succeq s_\succ(\mathcal{A}_2)$ et donc $\mathcal{B} \succeq s_\succ(\mathcal{A})$.

6. On montre le résultat par récurrence structurelle sur $\mathcal{A}$. Si $\mathcal{A} = \mathbf{O}$, $\mathbb{O}_\mathcal{A}$ ne contient qu'un élément et le résultat est établi. Sinon, $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$, alors $(\mathbb{O}_\mathcal{A}, \succ)$ est isomorphe[6] à $(\mathbb{O}^{dec}_{\mathcal{A}_1}, \succ^*)$. Par hypothèse de récurrence, $(\mathbb{O}_{\mathcal{A}_1}, \succ)$ est bien fondé, donc $(\mathbb{O}^{dec}_{\mathcal{A}_1}, \succ^*)$ est bien fondé par la question 3.4 et donc $(\mathbb{O}_\mathcal{A}, \succ)$ est bien fondé.

7. Nous savons que pour tout $\mathcal{A}$, $(\mathbb{O}_\mathcal{A}, \succ)$ est bien fondé. Or toute suite $\succ$-décroissante $(\mathcal{A}_i)_{i\in\mathbb{N}}$ de $\mathbb{O}$ (si elle existe) est une suite $\succ$-décroissante de $(\mathbb{O}_{\mathcal{A}_0}, \succ)$ ce qui n'est pas possible, donc $(\mathbb{O}, \succ)$ n'a pas de suite $\succ$-décroissante et par conséquent $(\mathbb{O}, \succ)$ est bien fondé.

8. On montre que $\mathsf{arbre}(b, m) \in \mathbb{O}$ et $\forall n, n'. m \geq n > n' \Rightarrow \mathsf{arbre}(b, n) \succ \mathsf{arbre}(b, n')$ par récurrence sur m.

Si $m = 0$ alors $\mathsf{arbre}(b, m) = \mathbf{O}$ appartient à $\mathbb{O}$.

Soit $m \neq 0$ et $(p, r) = \mathsf{dec1}(m)$. Par hypothèse de récurrence, $\mathsf{arbre}(b, p) \in \mathbb{O}$, $\mathsf{arbre}(b, r) \in \mathbb{O}$ et pour tous n, n' tels que $m > n > n'$, $\mathsf{arbre}(b, n) \succ \mathsf{arbre}(b, n')$.

– Si $r = 0$ alors $\mathsf{arbre}(b, m) = \mathsf{arbre}(b, p) \cdot \mathbf{O}$, donc $\mathsf{arbre}(b, m) \in \mathbb{O}$
– Si $r \neq 0$ et $(q, s) = \mathsf{dec1}(r)$ alors l'exposant maximal q de r est inférieur ou égal à p. Donc d'une part, $\mathsf{arbre}(b, p) \in \mathbb{O}$, $\mathsf{arbre}(b, r) \in \mathbb{O}$ et

$$\mathsf{arbre}(b, m) \equiv \mathsf{arbre}(b, p) \cdot (\mathsf{arbre}(b, q) \cdot \mathsf{arbre}(b, s))$$

et $\mathsf{arbre}(b, p) \succeq \mathsf{arbre}(b, q)$ donc $\mathsf{arbre}(b, m) \in \mathbb{O}$.

[6] L'indication de l'énoncé parle de «mots ordonnées», il fallait lire «mots décroissants».

Soit maintenant $m = n > n'$ et $(p_m, r_m) = \mathsf{dec1}(m)$.
- Si $n' = 0$ alors $\mathsf{arbre}(b, m) \equiv \mathsf{arbre}(b, p_m) \cdot \mathsf{arbre}(b, r_m) \succ \mathbf{O} = \mathsf{arbre}(b, n')$.
- Si $n' > 0$ et $(p_{n'}, r_{n'}) = \mathsf{dec1}(n')$, alors si $p_m > p_{n'}$ on a $\mathsf{arbre}(b, p_m) \succ \mathsf{arbre}(b, p_{n'})$ et si $p_m = p_{n'}$ on a $r_m > r_{n'}$ et $\mathsf{arbre}(b, p_m) = \mathsf{arbre}(b, p_{n'})$ ainsi que $\mathsf{arbre}(b, r_m) \succ \mathsf{arbre}(b, r_{n'})$ par hypothèse de récurrence. Donc dans tous les cas $\mathsf{arbre}(b, m) \succ \mathsf{arbre}(b, n')$.

9. Clairement $\mathsf{arbre}(k+1, \mathsf{erbra}(k+1, \mathsf{arbre}(k, n))) = \mathsf{arbre}(k, n)$ donc

$$\mathsf{arbre}(k, n) \succ \mathsf{arbre}(k+1, \mathsf{erbra}(k+1, \mathsf{arbre}(k, n)) - 1)$$

et donc $\mathsf{arbre}(k, g_k) \succ \mathsf{arbre}(k+1, g_{k+1})$. Donc la suite des $\mathsf{arbre}(k, g_k)$ est finie. Or le seul cas où une suite de Goodstein s'arrête est quand elle vaut 0.

4 Un autre ordre sur les arbres

1. Montrons par récurrence sur la somme des tailles de $\mathcal{A}$ et $\mathcal{B}$ que, pour tous arbres $\mathcal{A}, \mathcal{B}$, $\mathcal{A} \blacktriangleright \mathcal{B}$ ou $\mathcal{B} \blacktriangleright \mathcal{A}$ ou $\mathcal{A} = \mathcal{B}$. Si la somme des tailles est 0, alors $\mathcal{A} = \mathcal{B} = \mathbf{O}$. Si l'un des deux arbres est $\mathbf{O}$ et pas l'autre, alors la propriété est aussi satisfaite. Si $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$ et $\mathcal{B} = \mathcal{B}_1 \cdot \mathcal{B}_2$ alors, par hypothèse de récurrence, on a ou bien $\mathcal{A}_1 \blacktriangleright \mathcal{B}_1$, ou bien $\mathcal{A}_1 = \mathcal{B}_1$, ou bien $\mathcal{B}_1 \blacktriangleright \mathcal{A}_1$.
   - Si $\mathcal{A}_1 \blacktriangleright \mathcal{B}_1$ alors, à nouveau par hypothèse de récurrence, $\mathcal{A} \blacktriangleright \mathcal{B}_2$ et $\mathcal{A} \blacktriangleright B$ par 1, ou bien $\mathcal{B}_2 \underline{\blacktriangleright} \mathcal{A}$ et dans ce cas $B \blacktriangleright A$ par 3..
   - Si $\mathcal{A}_1 = \mathcal{B}_1$ alors, par hypothèse de récurrence, $\mathcal{A}_2 \blacktriangleright \mathcal{B}_2$ ou bien $\mathcal{B}_2 \blacktriangleright \mathcal{A}_2$, ou bien $\mathcal{A}_2 = \mathcal{B}_2$. Dans chaque cas on conclut, soit à l'égalité, soit à ordonner les deux arbres, grâce à 2.
   - Le cas $\mathcal{B}_1 \blacktriangleright \mathcal{A}_1$ est symétrique du cas $\mathcal{A}_1 \blacktriangleright \mathcal{B}_1$.

   L'irréflexivité est une conséquence de la propriété $(*)$ démontrée dans la question 4.2 suivante.

   Montrons que $\blacktriangleright$ est transitif en raisonnant par récurrence structurelle généralisée aux triplets d'arbres. Soient $\mathcal{A}$, $\mathcal{A}'$ et $\mathcal{A}''$ trois arbres tels que $\mathcal{A} \blacktriangleright \mathcal{A}'$ et $\mathcal{A}' \blacktriangleright \mathcal{A}''$. Si $\mathcal{A}'' = \mathbf{O}$ clairement $\mathcal{A} \blacktriangleright \mathcal{A}''$. Supposons que $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$, $\mathcal{A}' = \mathcal{A}'_1 \cdot \mathcal{A}'_2$ et $\mathcal{A}'' = \mathcal{A}''_1 \cdot \mathcal{A}''_2$.
   - Si $\mathcal{A} \blacktriangleright \mathcal{A}'$ par 3., alors $\mathcal{A}_2 \blacktriangleright \mathcal{A}' \blacktriangleright \mathcal{A}''$ et par hypothèse de récurrence, car $(\mathcal{A}, \mathcal{A}', \mathcal{A}'') \rhd\!\rhd\!\rhd (\mathcal{A}_1, \mathcal{A}', \mathcal{A}'')$, $\mathcal{A}_2 \blacktriangleright \mathcal{A}''$ ou $\mathcal{A}_2 = \mathcal{A}' \blacktriangleright \mathcal{A}''$ et donc par 3., $\mathcal{A} \blacktriangleright \mathcal{A}''$.
   - Si $\mathcal{A} \blacktriangleright \mathcal{A}'$ (par 1. ou 2.) et $\mathcal{A}' \blacktriangleright \mathcal{A}''$ (par 1. et 2.), mais pas tous les deux par 2., on a $\mathcal{A}_1 \blacktriangleright \mathcal{A}'_1$ ou $\mathcal{A}_1 = \mathcal{A}'_1$ et $\mathcal{A}'_1 \blacktriangleright \mathcal{A}''_1$ ou $\mathcal{A}'_1 = \mathcal{A}''_1$ (sans $\mathcal{A}_1 = \mathcal{A}'_1 = \mathcal{A}''_1$), donc $\mathcal{A}_1 \blacktriangleright \mathcal{A}''_1$. D'autre part on a soit $\mathcal{A} \blacktriangleright \mathcal{A}' \blacktriangleright \mathcal{A}''_2$, soit $\mathcal{A} \blacktriangleright \mathcal{A}'_2 \blacktriangleright \mathcal{A}''_2$ et par hypothèse de récurrence $\mathcal{A} \blacktriangleright \mathcal{A}''_2$.
   - Si $\mathcal{A} \blacktriangleright \mathcal{A}'$ (par 2.) et $\mathcal{A}' \blacktriangleright \mathcal{A}''$ (par 2.), on a $\mathcal{A}_1 = \mathcal{A}'_1 = \mathcal{A}''_2$ et $\mathcal{A}_2 \blacktriangleright \mathcal{A}'_2 \blacktriangleright \mathcal{A}''_2$ et par hypothèse de récurrence $\mathcal{A}_2 \blacktriangleright \mathcal{A}''_2$ et 2. s'applique.
   - Si $\mathcal{A} \blacktriangleright \mathcal{A}'$ (par 1.) et $\mathcal{A}' \blacktriangleright \mathcal{A}''$ (par 3.), on a $\mathcal{A} \blacktriangleright \mathcal{A}'_2 \blacktriangleright \mathcal{A}''$ ou $\mathcal{A} \blacktriangleright \mathcal{A}'_2 = \mathcal{A}''$ et par hypothèse de récurrence $\mathcal{A} \blacktriangleright \mathcal{A}''$.
   - Si $\mathcal{A} \blacktriangleright \mathcal{A}'$ (par 2.) et $\mathcal{A}' \blacktriangleright \mathcal{A}''$ (par 3.), on a $\mathcal{A}_2 \blacktriangleright \mathcal{A}'_2 \blacktriangleright \mathcal{A}''$ ou $\mathcal{A}_2 \blacktriangleright \mathcal{A}'_2 = \mathcal{A}''$ et par hypothèse de récurrence $\mathcal{A}_2 \blacktriangleright \mathcal{A}''$, on peut appliquer 3. pour conclure que $\mathcal{A} \blacktriangleright \mathcal{A}''$.

2. On va montrer par récurrence structurelle sur $\mathcal{A}$ une propriété plus générale à savoir que

$$\text{si } \mathcal{B} \trianglerighteq \mathcal{A} \text{ alors } \neg(\mathcal{A} \blacktriangleright \mathcal{B}) \qquad (*)$$

   - Si $\mathcal{A} = \mathbf{O}$ c'est la définition.
   - Si $\mathcal{B} = \mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$, ni 1., ni 2. ne s'appliquent, d'autre part l'on a $\mathcal{B} \trianglerighteq \mathcal{A}_2$ donc $\neg(\mathcal{A}_2 \blacktriangleright \mathcal{B})$ par hypothèse de récurrence et 3. ne peut pas non plus s'appliquer.
   - Si $\mathcal{B} \triangleright \mathcal{A}$, on a soit $\mathcal{B}_1 \trianglerighteq \mathcal{A}$ ou $\mathcal{B}_2 \trianglerighteq \mathcal{A}$.
     - Si $\mathcal{B}_1 \trianglerighteq \mathcal{A}$, on n'a pas 1., parce qu'on a $\mathcal{B}_1 \triangleright \mathcal{A}_1$ donc $\neg(\mathcal{A}_1 \blacktriangleright \mathcal{B}_1)$. On n'a pas 2. parce qu'on n'a pas $\mathcal{A}_1 = \mathcal{B}_1$. On n'a pas 3., parce qu'on a $\mathcal{B} \triangleright \mathcal{A}_2$ donc ni $\mathcal{A}_2 \blacktriangleright \mathcal{B}$ ni $\mathcal{A}_2 = \mathcal{B}$.
     - Si $\mathcal{B}_2 \trianglerighteq \mathcal{A}$, on n'a pas 1., parce qu'on n'a pas $\mathcal{A} \blacktriangleright \mathcal{B}_2$, on n'a pas pas 2. parce qu'on a $\mathcal{B}_2 \triangleright \mathcal{A}_2$ donc $\neg(\mathcal{A}_2 \blacktriangleright \mathcal{B}_2)$. Et comme ci-dessus, on n'a pas 3., parce qu'on a $\mathcal{B} \triangleright \mathcal{A}_2$ donc ni $\mathcal{A}_2 \blacktriangleright \mathcal{B}$ ni $\mathcal{A}_2 = \mathcal{B}$.

   Comme $\blacktriangleright$ est irréflexif et total, on a $\mathcal{B} \trianglerighteq \mathcal{A}$ implique $\mathcal{B} \blacktriangleright \mathcal{A}$ ou $\mathcal{B} = \mathcal{A}$ et donc $\mathcal{B} \triangleright \mathcal{A}$ implique $\mathcal{B} \blacktriangleright \mathcal{A}$.

3.
```
let rec triangle = function
  o -> (function _ -> false) |
  F(a1,a2) ->
     (function  o -> true |
                F(b1,b2) -> ((triangle a1 b1) && (triangle (F(a1,a2)) b2)
                             || ((a1 = b1) && (triangle a2 b2))
                             || (triangle a2 (F(b1,b2)))
                             || (a2 = F(b1,b2))))
```

4. On peut montrer que $s_{\blacktriangleright}(\mathcal{A}) = \mathbf{O} \cdot \mathcal{A}$. $\mathbf{O} \cdot \mathcal{A} \blacktriangleright \mathcal{A}$ par 3 . Montrons maintenant par récurrence structurelle sur $\mathcal{A}$ que $\mathcal{B} \blacktriangleright \mathcal{A}$ entraine $\mathcal{B} \blacktriangleright \mathcal{A}$ ou $\mathcal{B} = \mathcal{A}$.
   Si $\mathcal{A} = \mathbf{O}$, alors $\mathbf{O} \cdot \mathcal{B}_2 \blacktriangleright \mathbf{O} \cdot \mathbf{O}$ ou $\mathbf{O} \cdot \mathcal{B}_2 = \mathbf{O} \cdot \mathbf{O}$.
   Si $\mathcal{B} \blacktriangleright \mathcal{A}$ et $\mathcal{A} \neq \mathbf{O}$, alors $\mathcal{B} = \mathcal{B}_1 \cdot \mathcal{B}_2$.
   - Si $\mathcal{B}_1 \blacktriangleright \mathbf{O}$ alors par 1. $\mathcal{B} \blacktriangleright \mathbf{O} \cdot \mathcal{A}$.
   - Si $\mathcal{B}_1 = \mathbf{O}$, on regarde les différents cas qui ont conduit à la preuve de $\mathcal{B} \blacktriangleright \mathcal{A}$.
     - par 2., c-à-d $\mathcal{A} = \mathbf{O} \cdot \mathcal{A}_2$ et $\mathcal{B}_2 \blacktriangleright \mathcal{A}_2$. Par hypothèse de récurrence, $\mathcal{B}_2 \blacktriangleright \mathbf{O} \cdot \mathcal{A}_2$ ou $\mathcal{B}_2 = \mathbf{O} \cdot \mathcal{A}_2$, donc ou bien par 1. $\mathcal{B} = \mathbf{O} \cdot \mathcal{B}_2 \blacktriangleright \mathbf{O} \cdot (\mathbf{O} \cdot \mathcal{A}_2) = \mathbf{O} \cdot \mathcal{A}$ ou bien $\mathcal{B} = \mathbf{O} \cdot \mathcal{A}$.
     - par 3., $\mathcal{B}_2 \blacktriangleright \mathcal{A}$ ou $\mathcal{B}_2 = \mathcal{A}$, donc ou bien par 1. $\mathcal{B} \blacktriangleright \mathbf{O} \cdot \mathcal{A}$, ou bien $\mathcal{B} = \mathbf{O} \cdot \mathcal{A}$.

5. Une suite $\blacktriangleright$-décroissante ne peut pas contenir $\mathbf{O}$. Soit $(\mathcal{A}_1^n \cdot \mathcal{A}_2^n)_{n\in\mathbb{N}}$ une suite $\blacktriangleright$-décroissante plus petite au sens de l'énoncé du problème. Aucune des inégalités

$$(\mathcal{A}_1^n \cdot \mathcal{A}_2^n) \blacktriangleright (\mathcal{A}_1^{n+1} \cdot \mathcal{A}_2^{n+1})$$

   n'est due à 3, car cela contredirait le fait que c'est la plus petite suite $\blacktriangleright$-décroissante. Les couples $(\mathcal{A}_1^n, \mathcal{A}_2^n)_{n\in\mathbb{N}}$ forment une suite $\blacktriangleright \times \blacktriangleright$-décroissante de $\mathbb{A}^2$. Ou bien $(\mathcal{A}_1^n)_{n\in\mathbb{N}}$ est stationnaire à partir de $j \in \mathbb{N}$ et $(\mathcal{A}_2^n)_{n>j}$ est $\blacktriangleright$-décroissante et donc

$$(\mathcal{A}_1^0 \cdot \mathcal{A}_2^0), ..., (\mathcal{A}_1^j \cdot \mathcal{A}_2^j), \mathcal{A}_2^{j+1}, \mathcal{A}_2^{j+2}, ...$$

   est une suite $\blacktriangleright$-décroissante plus petite. Ou bien l'on peut extraire de $(\mathcal{A}_1^n)_{n\in\mathbb{N}}$ une suite $(\mathcal{A}_1^{\theta(k)})_{k\in\mathbb{N}}$ $\blacktriangleright$-décroissante et alors la suite

$$(\mathcal{A}_1^0 \cdot \mathcal{A}_2^0), ..., (\mathcal{A}_1^{\theta(0)-1} \cdot \mathcal{A}_2^{\theta(0)-1}), \mathcal{A}_1^{\theta(0)}, \mathcal{A}_1^{\theta(1)}, ...$$

est une suite $\blacktriangleright$-décroissante plus petite.

6. Soient $\mathcal{A} \in \mathbb{O}$ et $\mathcal{B} \in \mathbb{O}$. On montre par récurrence structurelle généralisée aux paires d'arbres que $\mathcal{A} \succ \mathcal{B}$ entraine $A \blacktriangleright B$.

   Si $\mathcal{A} = \mathbf{O}$, la prémisse est fausse et donc l'implication est satisfaite. Si $\mathcal{A} \neq \mathbf{O}$ et $\mathcal{B} = \mathbf{O}$, alors l'implication résulte de la définition de $\blacktriangleright$. Supposons maintenant que $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$ et $\mathcal{B} = \mathcal{B}_1 \cdot \mathcal{B}_2$.

   En utilisant le fait que $\mathcal{A}$ et $\mathcal{B}$ sont ordonnés, on peut montrer que $\mathcal{A}_1 \succ \mathcal{B}_1$ implique $\mathcal{A} \succ \mathcal{B}_2$.

   – Si $\mathcal{A}_1 \succ \mathcal{B}_1$ on a $\mathcal{A} \succ \mathcal{B}_2$ et donc, par hypothèse de récurrence, $\mathcal{A}_1 \blacktriangleright \mathcal{B}_1$ et $\mathcal{A} \blacktriangleright \mathcal{B}_2$ et par 1. $\mathcal{A} \blacktriangleright \mathcal{B}$.

   – Si $\mathcal{A}_1 = \mathcal{B}_1$ et $\mathcal{A}_2 \succ \mathcal{B}_2$, on a, par hypothèse de récurrence, $\mathcal{A} \blacktriangleright \mathcal{B}_2$.

   On remarque que 3. ne s'applique jamais dans ce cas.

7. S'il existe une suite $\succ$-décroissante dans $\mathbb{O}$ alors la même suite est $\blacktriangleright$-décroissante dans $\mathbb{A}$ (question 6), ce qui n'est pas possible (question 5) donc $(\mathbb{O}, \succ)$ est bien fondé.

8. On montre d'abord que $\mathcal{A} \rhd \mathcal{B}$ implique $\mathcal{A} \succ \mathcal{B}$, lorsque $\mathcal{A}, \mathcal{B} \in \mathbb{O}$, par récurrence structurelle sur $\mathcal{A}$: lorsque $\mathcal{A} = \mathbf{O}$, il n'y a rien a prouver puisque la prémisse de l'implication est fausse. Si maintenant $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2$, alors $\mathcal{A} \succ \mathcal{A}_1$ car, ou bien $\mathcal{A}_1 = \mathbf{O}$, ou bien $\mathcal{A}_1 = \mathcal{A}_{11} \cdot \mathcal{A}_{12}$ et, par hypothèse de récurrence, $\mathcal{A}_1 \succ \mathcal{A}_{11}$. Alors, par définition de l'ordre, $\mathcal{A} \succ \mathcal{A}_1$. On a aussi $\mathcal{A} \succ \mathcal{A}_2$. En effet, ou bien $\mathcal{A}_2 = \mathbf{O}$, ou bien $\mathcal{A}_2 = \mathcal{A}_{21} \cdot \mathcal{A}_{22}$. Dans ce dernier cas, comme $\mathcal{A} \in \mathbb{O}$, $\mathcal{A}_1 \succeq \mathcal{A}_{21}$ et $\mathcal{A}_2 \succ \mathcal{A}_{22}$ par hypothèse de récurrence. Donc, par définition de $\succ$, $\mathcal{A} \succ \mathcal{A}_2$.

   Si maintenant $\mathcal{A} \rhd \mathcal{B}$, alors $\mathcal{A}_1 = \mathcal{B}$ ou $\mathcal{A}_1 \rhd \mathcal{B}$ ou $\mathcal{A}_2 \rhd \mathcal{B}$ ou $\mathcal{A}_2 = \mathcal{B}$.

   Dans les deux premier cas on clonclut que $\mathcal{A} \succ \mathcal{B}$ de $\mathcal{A} \succ \mathcal{A}_1$, de l'hypothèse de récurrence et de la transitivité de $\succ$ (question 3.1). Dans les deux derniers cas on conclut que $\mathcal{A} \succ \mathcal{B}$ de $\mathcal{A} \succ \mathcal{A}_2$, de l'hypothèse de récurrence et de la transitivité de $\succ$.

   a) Soit $E = \{s^n_{\succ}(\mathbf{O}) \mid n \in \mathbb{N}\} = \{s^n_{\blacktriangleright}(\mathbf{O}) \mid n \in \mathbb{N}\}$ (voir questions 3.5 et 4.4). E est l'ensemble des arbres t dont tout sous arbre est ou bien réduit à $\mathbf{O}$, ou bien a pour fils gauche $\mathbf{O}$.

   Montrons que $(\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O} = \sup_{\succ} E = \sup_{\underline{\blacktriangleright}} E$.

   Pour tout $t \in E$, $(\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O} \succ t$ puisque $(\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O} \notin E$ et de même $(\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O} \underline{\blacktriangleright} t$

   Si $t \notin E$, alors $t \succeq (\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}$ et $t \underline{\blacktriangleright} (\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}$. En effet, t a un sous-arbre u tel que $u = \mathcal{A}_1 \cdot \mathcal{A}_2$ et $\mathcal{A}_1 \neq \mathbf{O}$. D'après la question 4.2, $t \underline{\blacktriangleright} u$ et d'après ce que nous avons vu ci-dessus, $t \succeq u$. Il suffit donc de montrer le résultat lorsque le fils gauche de t n'est pas une feuille. Soit $t = t_1 \cdot t_2$. $t_1 \succeq s_{\succ}(0) = s_{\blacktriangleright}(0) = \mathbf{O} \cdot \mathbf{O}$. Par définition des ordres on a donc $t \succeq (\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}$ et $t \underline{\blacktriangleright} (\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}$.

   Il en résulte que $\psi((\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}) = (\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}$

   b) Soit $\mathcal{A}_n$ la suite d'arbres définie par : $\mathcal{A}_{n+1} = (s^n_{\succ}(\mathbf{O})) \cdot \mathbf{O}$.

   Montrons que $((\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}) \cdot \mathbf{O} = \sup_{\succ} \{\mathcal{A}_n \mid n \in \mathbb{N}\}$. Tout d'abord, $((\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O}) \cdot \mathbf{O} > \mathcal{A}_n$ pour tout n puisque $(\mathbf{O} \cdot \mathbf{O}) \cdot \mathbf{O} \succ s^n_{\succ}(\mathbf{O})$ pour tout n et par définition

de l'ordre. Supposons maintenant que $t \succ \mathcal{A}_n$ pour tout n et montrons que $t \succeq ((\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O}$. $t \neq \mathbf{O}$ et donc $t = t_1 \cdot t_2$. Par définition, $t_1 \succeq s^n_{\succ}(\mathbf{O})$ pour tout n et donc, d'après la question précédente, $t_1 \succeq (\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O}$. Par ailleurs $t_2 \succeq \mathbf{O}$, donc $t \succeq ((\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O}$ par définition de l'ordre.

Par ailleurs, montrons par récurrence sur n que $\psi(\mathcal{A}_n) = \mathcal{B}_n$ où $\mathcal{B}_1 = (\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O}$ et $\mathcal{B}_{n+1} = (\mathbf{O}\cdot\mathbf{O})\cdot\mathcal{B}_n$. Pour $n = 1$, c'est la question a). Supposons maintenant la propriété vraie pour n et montrons par récurrence sur k que $\psi(\mathcal{A}_{n,k}) = \mathcal{B}_{n,k}$ où $\mathcal{A}_{n,1} = \mathcal{A}_n$, $\mathcal{A}_{n,k+1} = s^n_{\succ}(\mathbf{O})\cdot\mathcal{A}_{n,k}$, $\mathcal{B}_{0,k} = s^k_{\blacktriangleright}(\mathbf{O})$, $\mathcal{B}_{n+1,k} = (\mathbf{O}\cdot\mathbf{O})\cdot\mathcal{B}_{n,k}$. Pour $k = 0$, cela vient de la première hypothèse de récurrence : $\mathcal{A}_{n,0} = \mathcal{A}_n$ et $\mathcal{B}_{n,0} = \mathcal{B}_n$.

Enfin, on montre que $\sup_{\blacktriangleright}\{\mathcal{B}_n \mid n \in \mathbb{N}\} = (\mathbf{O}\cdot(\mathbf{O}\cdot\mathbf{O}))\cdot\mathbf{O}$

Il en résulte que $\psi(((\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O}) = (\mathbf{O}\cdot(\mathbf{O}\cdot\mathbf{O}))\cdot\mathbf{O}$

c) Soit $\mathcal{C}_n = \mathcal{A}_n\cdot\mathbf{O}$.

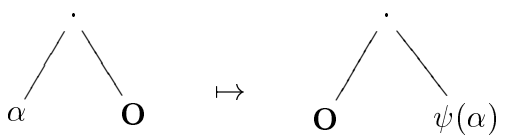
On montre que $(((\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O} = \sup_{\succ}\{\mathcal{C}_n \mid n \in \mathbb{N}\}$ puis que $\psi(\mathcal{C}_n) = \mathcal{A}_n$.

Enfin, $\sup_{\underline{\blacktriangleright}}\{\mathcal{A}_n \mid n \in \mathbb{N}\} = ((\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O}$.

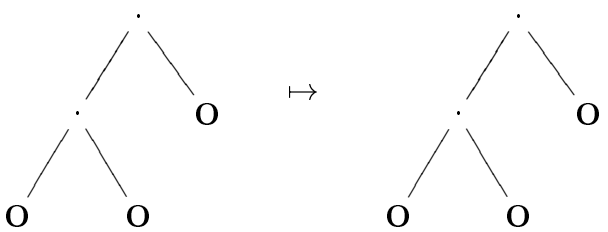
On en déduit $\psi((((\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O}) = ((\mathbf{O}\cdot\mathbf{O})\cdot\mathbf{O})\cdot\mathbf{O}$.

De manière plus synthétique et plus lisible, voici quelques valeurs de la fonction ψ, avec les notations de Cantor pour les ordinaux, pour ceux qui les connaissent.

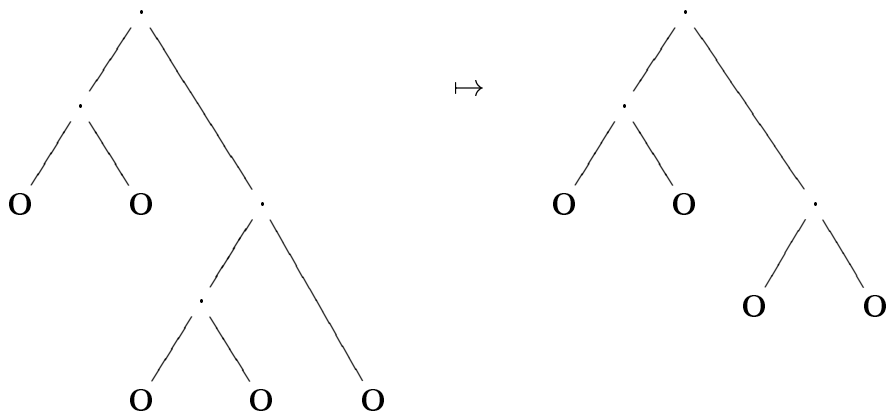
$$(\mathbb{O}, \succ) \longrightarrow (\mathbb{A}, \blacktriangleright)$$



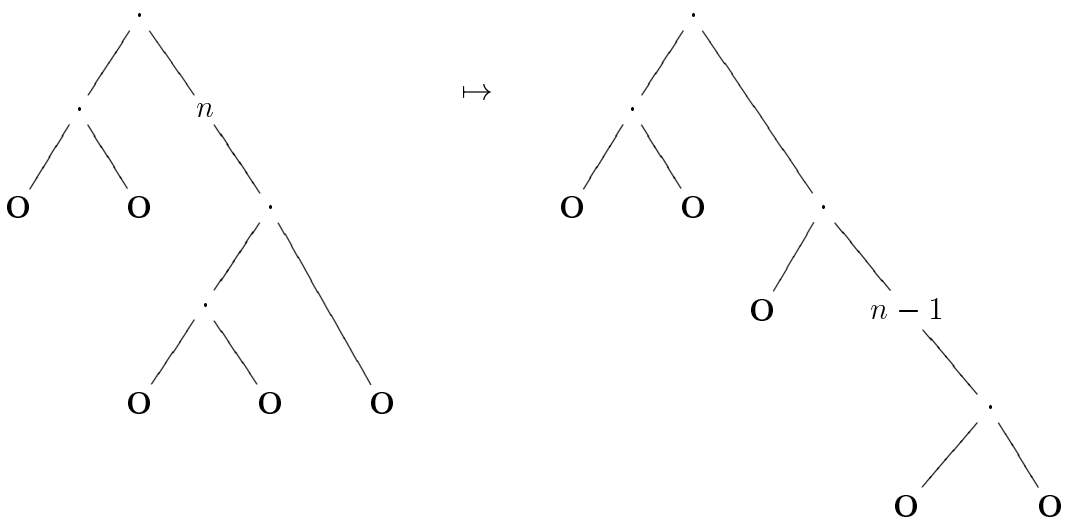
En passant à la limite les successeurs de $\mathbf{O}$ on obtient ω, quantité (a) :



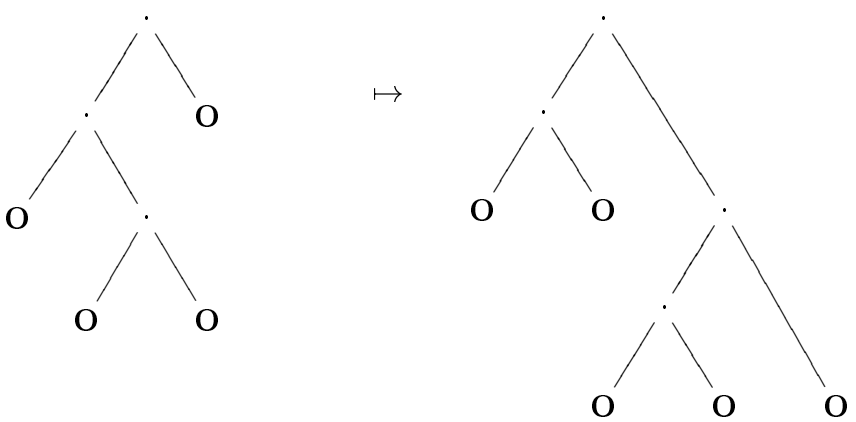
On prend les successeurs et on passe à la limite pour obtenir $\omega + \omega$:



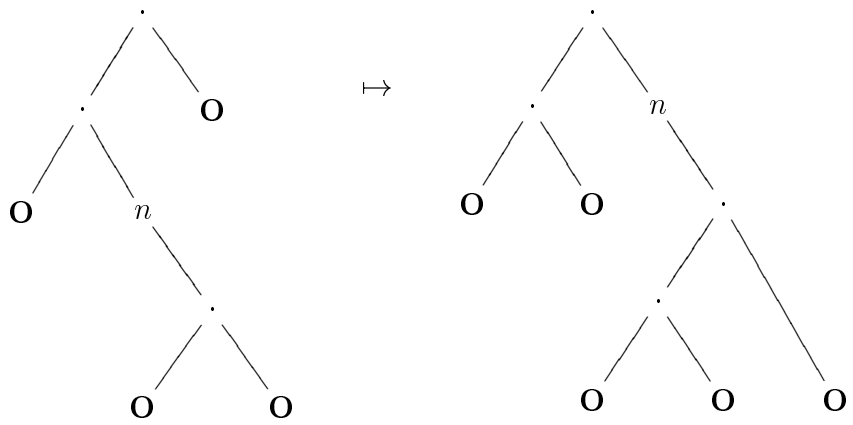
Puis ωn :



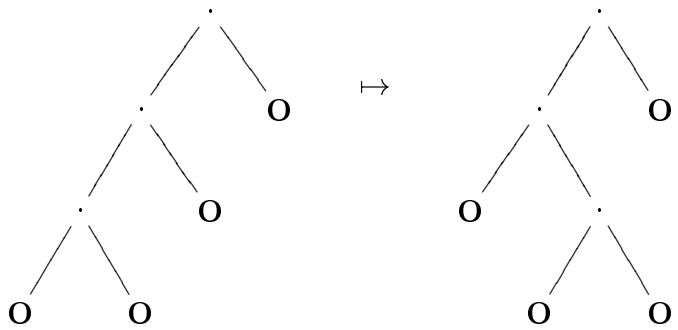
On passe à nouveau à la limite et l'on obtient ω^2 :



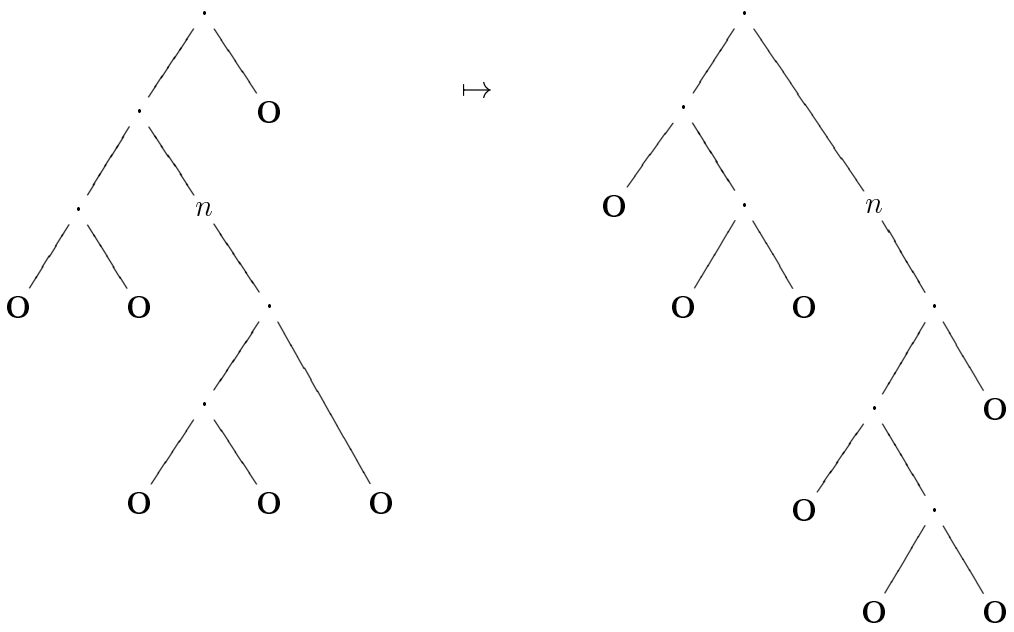
Puis ω^n :



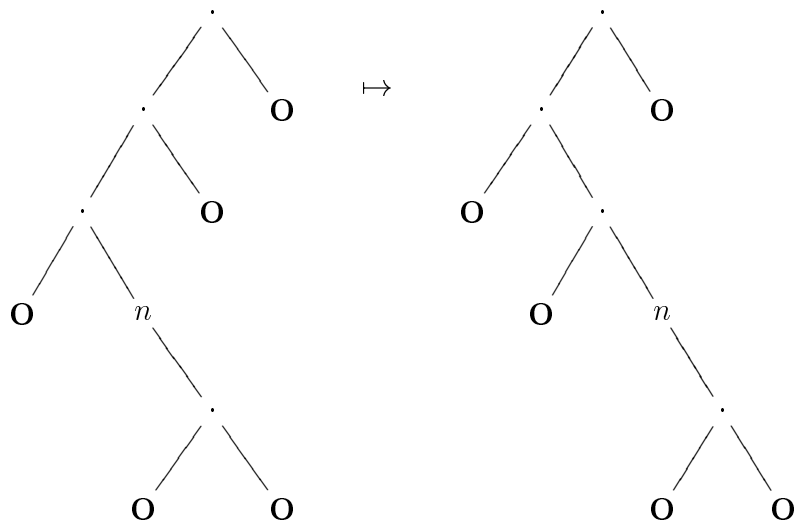
Puis ω^ω, quantité (b) :



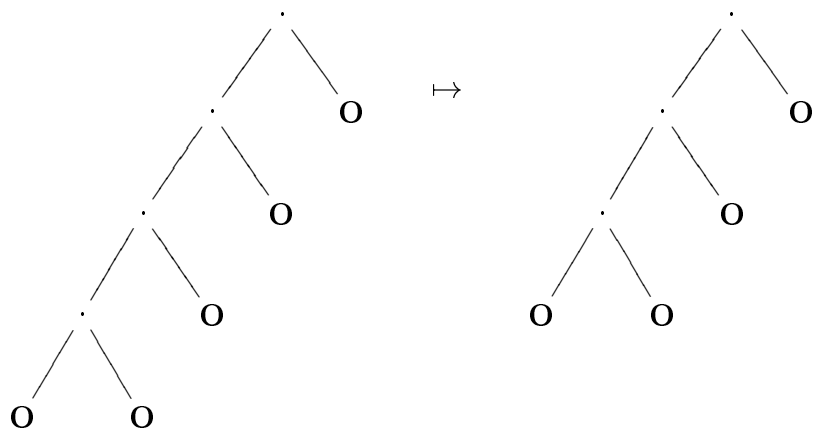
Puis $\omega^{\omega n}$:



Puis ω^{ω^n} :



Enfin ω^{ω^ω}, quantité (c) : :



5 Un ordre sur les mots

1. – $\boxed{a \Rightarrow b}$ Si dans a, on prend pour $\sqsupset$, l'ordre $>$ lui-même, on obtient que $(E, >)$ est bien fondé. Si $(E, >)$ contient une antichaîne infinie $a_0, \ldots, a_i, a_{i+1}, \ldots$, alors l'ordre défini par

$$x \sqsupset y \iff x > y \vee (\exists i, j)(i > j \wedge x = a_i \wedge y = a_j)$$

n'est pas bien fondé.

– $\boxed{b \Rightarrow a}$ S'il existe un ordre $\sqsupset$ qui contient $>$ et qui n'est pas bien fondé, alors il existe une suite $\sqsupset$ décroissante $a_0 \sqsupset a_1, \ldots, a_i \sqsupset a_{i+1}, \ldots$. Dans cette suite il existe un sous-suite $(a_{\varphi(i)})_{i\in\mathbb{N}}$ telle que $i > j \Rightarrow a_{\varphi(i)} \sqsupset a_{\varphi(j)} \wedge \neg(a_{\varphi(i)} > a_{\varphi(j)})$. $(a_{\varphi(i)})_{i\in\mathbb{N}}$ est un antichaîne infinie de $(E, >)$.

– $\boxed{c \Rightarrow b}$ Si dans toute suite $(x_i)_{i\in\mathbb{N}}$ on peut trouver i et j tels que $i < j$ et $x_i \leq x_j$ alors aucune suite n'est $>$-décroissante, ni une antichaîne.

– $\boxed{b \Rightarrow c}$ Soit $a_0, \ldots, a_i, a_{i+1}, \ldots$ une suite. Puisque $(E, >)$ est bien fondé aucune de ses sous-suites n'est $>$-décroissantes. Donc pour chaque i, on peut construire une liste $>$-décroissante de longueur maximum $a_{i_0} > \ldots > a_{i_j} > \ldots > a_{i_{g(i)}}$, qui commence en a_i c-à-d telle que $a_{i_0} = a_i$. Clairement ou bien

$$(\exists k)(k > g(i) \Rightarrow x_k \geq a_{i_{g(i)}}) \tag{1}$$

ou bien

$$((\forall k)k > g(i) \Rightarrow \neg(x_k \geq a_{i_{g(i)}}) \wedge \neg(a_{i_{g(i)}} > x_k)) \tag{2}$$

Si (1) est satisfait, alors la suite est bonne. Si (1) n'est satisfait pour aucun i, alors (2) est satisfait pour tout i, donc la suite des $a_{g(i)}$ est une antichaîne, ce qui est une contradiction.

– $\boxed{d \Rightarrow c}$ est clair.

– $\boxed{c \Rightarrow d}$ Soit $a_0, \ldots, a_i, a_{i+1}, \ldots$ une suite qui ne contient aucune sous-suite croissante. Considérons les listes croissantes maximales qu'il contient, ces sont des listes $a_k = a_{i_0} \leq \ldots \leq a_{i_j} \leq \ldots \leq a_{i_{g(k)}}$ telles que pour aucun $j > i_{g(k)}$ on ait $a_{i_{g(k)}} \leq a_j$. Il y a une infinité de telles listes commençant en $k_0, k_1, \ldots, k_n, \ldots$ (sinon ce serait en contradiction avec c). Dans la suite $(a_{g(k_j)})_{j\in\mathbb{N}}$ on peut trouver k_j et k_l tels que $k_j < k_l$ et $a_{g(k_j)} \leq a_{g(k_l)}$, donc au moins l'une des listes croissantes n'est pas maximales, ce qui est une contradiction.

2. Si $(\mathsf{A}^*, \geqslant)$ est un bel ordre alors clairement $(\mathsf{A}, >_\mathsf{A})$ est un bel ordre. En effet $\geqslant$ est un bel ordre sur les mots d'une lettre qui coïncide avec $>_\mathsf{A}$.

   Réciproquement on construit une plus petite suite mauvaise (c'est-à-dire qui n'est pas «bonne») $(\alpha_i)_{i\in\mathbb{N}}$ comme la suite mauvaise qui commence par le plus petit mot et telle que si la suite commence par $\alpha_0, \ldots, \alpha_n$ alors le mot α_{n+1} est le plus petit en $n+1$-ème position parmi les mots en $n+1$-ème position dans les suites mauvaises qui commencent par $\alpha_0, \ldots, \alpha_n$. Une suite mauvaise ne peut pas contenir ε. Donc la suite est de la forme $(a_i\beta_i)_{i\in\mathbb{N}}$. Comme $(\mathsf{A}, >_\mathsf{A})$ est un bel ordre, on peut extraire de la suite $(a_i)_{i\in\mathbb{N}}$ une sous-suite croissante $(a_{f(i)})_{i\in\mathbb{N}}$ telle que $a_{f(0)} \leq_\mathsf{A} a_{f(1)} \ldots a_{f(i)} \leq_\mathsf{A} a_{f(i+1)} \ldots$ La suite $(\beta_{f(i)})_{i\in\mathbb{N}}$ est bonne, sinon la suite $\alpha_0, \ldots \alpha_{f(0)-1}, \beta_{f(0)}, \beta_{f(1)}, \ldots \beta_{f(i)}, \ldots$ serait une suite mauvaise plus «petite» que $(\alpha_i)_{i\in\mathbb{N}}$. Donc de la suite $(\beta_{f(i)})_{i\in\mathbb{N}}$ on peut extraire une sous-suite croissante $\beta_{g(f(0))} \leq_\mathsf{A} \beta_{g(f(1))} \ldots \beta_{g(f(i))} \leq_\mathsf{A} \beta_{g(f(i+1))}$. La suite $(a_{g(f(i))}\beta_{g(f(i))})_{i\in\mathbb{N}}$ est une sous-suite croissante extraite de la suite $(\alpha_i)_{i\in\mathbb{N}}$. Contradiction. Donc $(\mathsf{A}^*, \geqslant)$ est un bel ordre.

Commentaires

Dans la question 1.1, on applique de façon essentielle le principe *ex falso sequitur quodlibet*, autrement dit «du faux on peut déduire ce qu'on veut».

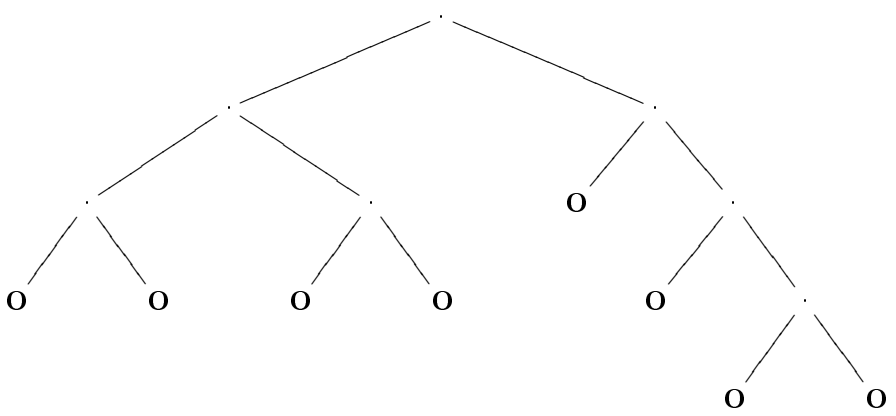
Pour la question 2.6, on peut faire mieux. On remarque que $5\,6^4+5\,6^3+5\,6^2+5\,6+5 = 6^5-1 = 7775 = \alpha$, donc on atteint $5.(\alpha+6)^{\alpha+6} + 5.(\alpha+6)^{\alpha+5}$, puis plus tard

$$5\ (5\ (\alpha+6)^{\alpha+5} + \alpha + 6)^{5\ (\alpha+6)^{\alpha+5}+\alpha+6}$$

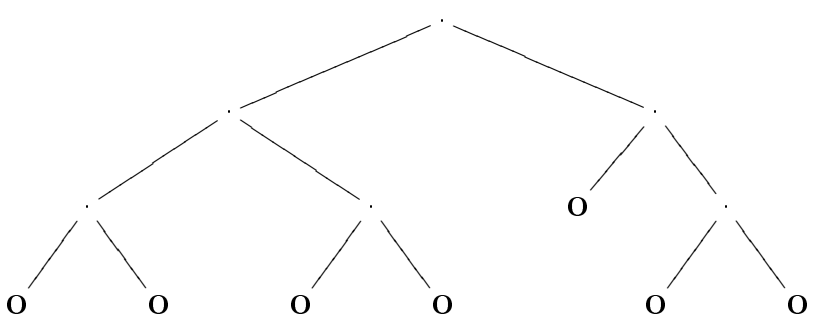
qui est plus grand que α^{α^α} et donc que $7000^{7000^{7000}}$.

Par les arbres correspondant à g_2, g_3, g_4, g_5, g_6, on peut observer le phénomène qui conduit à la décroissance :

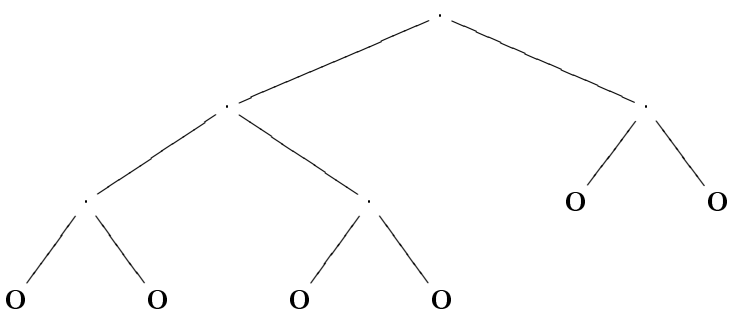
$g_2 =$



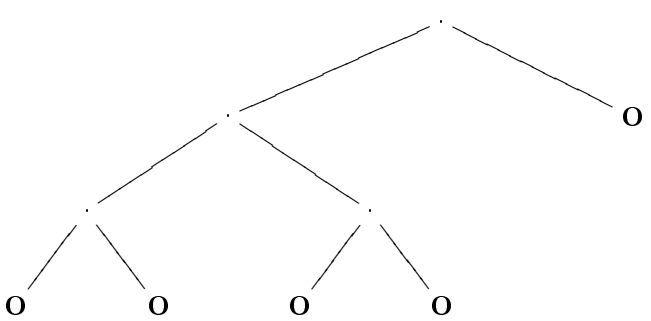
$g_3 =$



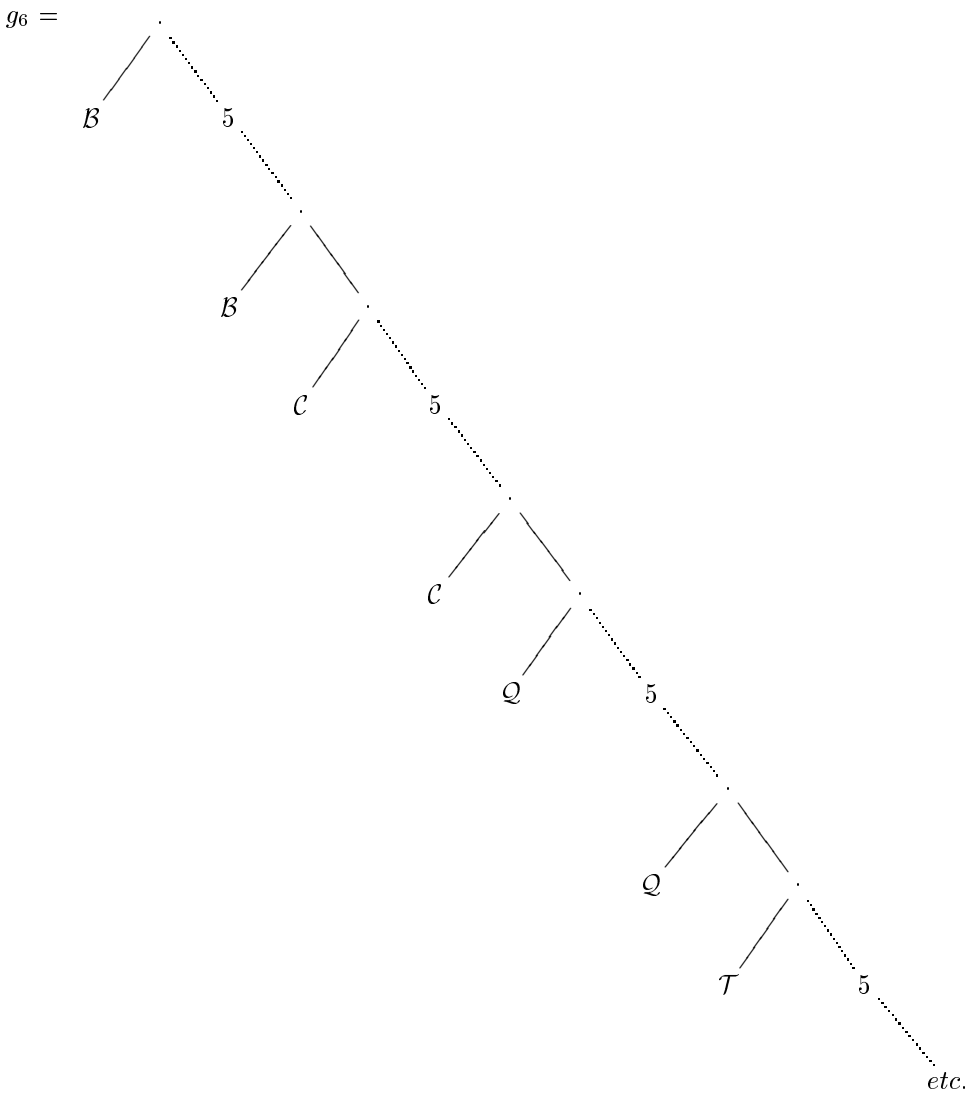
$g_4 =$



$g_5 =$

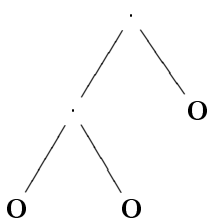


$g_6 =$

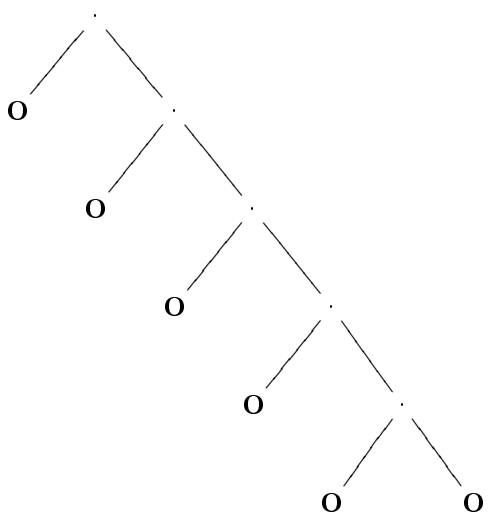


où

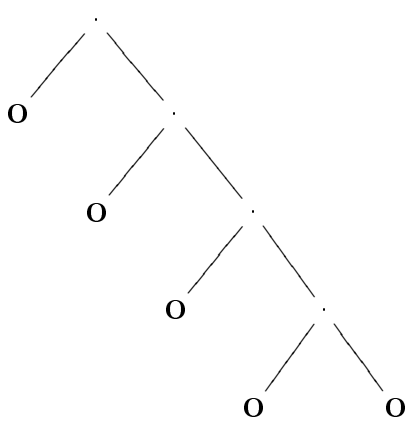
$\mathcal{B} =$



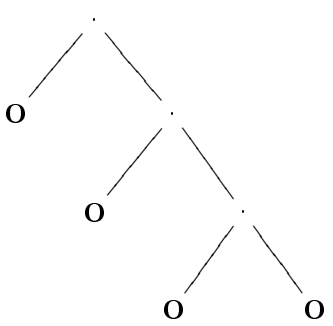
$\mathcal{C} =$



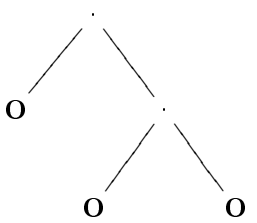
$\mathcal{Q} =$



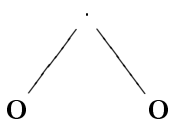
$\mathcal{T} =$



$\mathcal{D} =$



$\mathcal{U} =$



Dans g_6 le *etc.* correspond à 5 fois $\mathcal{D}$ et cinq fois $\mathcal{U}$. On voit que g_6 décroît quant à son sous-arbre droit et que g_6 est un arbre ordonné.

Dans la réciproque de la question 3.5, on peut appliquer un raisonnement plus direct. Reformulons la définition de $>_{\mathsf{A}^*}$,

– $a\alpha >_{\mathsf{A}^*} \varepsilon$ (où ε est le mot vide) **(1)**.

– $a\alpha >_{\mathsf{A}^*} b\beta$ si et seulement si $a >_{\mathsf{A}} b$ **(2)** ou bien $a = b$ et $\alpha >_{\mathsf{A}^*} \beta$ **(3)**.

Supposons que $(A, >_A)$ est bien fondé et soit $(\alpha_i)_{i\in\mathbb{N}}$ une suite $>_{\mathsf{A}^*}$-décroissante. Pour décroître de α_i à α_{i+1}, on peut soit utiliser une fois **(2)** avec donc $a_i^k = a_{i+1}^k$ pour $1 \leq k < j$ et $a_i^j > a_{i+1}^j$ (*principe de décroissance de lettre*), soit n'utiliser que plusieurs fois **(2)** et une fois **(1)**, auquel cas α_{i+1} est un préfixe strict de α_i et $\|\alpha_i\| > \|\alpha_{i+1}\|$ (*principe du préfixe*).

De deux choses l'une, dans la suite $>_{\mathsf{A}^*}$-décroissante $(\alpha_i)_{i\in\mathbb{N}}$

– ou bien on n'utilise que le *principe du préfixe* auquel cas on a une suite strictement de décroissante de naturels (la taille des α_i), ce qui n'est pas possible,
– ou bien on utilise une infinité de fois le *principe de décroissance de lettre*, ce qui permet de construire une suite $>_{\mathsf{A}}$-décroissante, de lettres. Chaque fois que l'on applique *principe de décroissance de lettre*, on choisit la lettre a_{i+1}^j correspondante ; grâce à la décroissante des mots, cette suite est $>_{\mathsf{A}}$-décroissante, là encore on aboutit à une contradiction.