Le théorème de cardinalité

Michaël Monerau

Décembre 2007

Plan

Résumé

Savoir si un ensemble est récursif, c'est-à-dire, savoir s'il est *décidable par une machine de Turing*, est le coeur même de la théorie de la décidabilité. C'est pourquoi tout critère permettant d'arriver à une telle conclusion est intéressant.

Ici, nous allons aborder le théorème de cardinalité. Il fait partie de ces critères très fins qui permettent de conclure à la récursivité d'un ensemble sous certaines hypothèses. Nous commencerons par poser le cadre de l'étude, puis nous énoncerons et prouverons le théorème de cardinalité.

1 Oracles et décidabilité

1.1 Oracles et objectifs de l'étude

Un ***oracle*** est une machine de Turing qui permet de décider un certain problème en une étape de calcul. On peut voir une telle machine comme une « boîte noire » qui décide instantanément si les entrées qu'on lui donne vérifient ou non le problème en question. Un oracle renvoie donc toujours 0 ou 1.

L'optique dans toute la suite de cette étude est de mesurer la décidabilité d'une partie A de $\mathbb{N}$ selon le nombre de requêtes qu'il est nécessaire de faire à un oracle B fixé pour répondre à des questions sur A.

On peut avoir l'intuition que cette vision sera fructueuse. En effet, si on autorise un nombre très faible de requêtes, cela implique qu'il faut peu d'informations pour répondre aux questions qu'on se pose sur A, et donc sa structure est certainement simple (*ie.* A est récursif). Au contraire, si on est obligé de faire beaucoup de requêtes à l'oracle, c'est que A n'est certainement pas facile à comprendre, et donc non récursif. La suite va consister à formaliser clairement et précisément cette idée.

1.2 Le théorème de non-accélération (nonspeedup)

En 1986, Beigel a montré dans [Bei87] le théorème suivant :

[1.A] Théorème ***(Non-accélération - nonspeedup, 1987)***

Soit A une partie de $\mathbb{N}$. Soit la fonction :

$$f_n^A \ : \left| \begin{array}{ccc} \mathbb{N}^{2^n} & \longrightarrow & \{0,1\}^{2^n} \\ (x_1, \ldots, x_{2^n}) & \longmapsto & (\chi_A(x_1), \ldots, \chi_A(x_{2^n})) \end{array} \right.$$

Si f_n^A peut être calculée avec au plus n requêtes à un oracle B, alors A est récursif.

On peut voir le résultat de ce théorème sous une forme plus imagée : 2^n requêtes à un oracle non récursif A ne peuvent pas être décidées en faisant n appels à un certain oracle B (même si les appels s'adaptent aux données d'entrées).

C'est bien dans l'idée de ce qu'on a pressenti : si A est *trop compliqué*, il faut beaucoup de requêtes à un oracle pour répondre à une question le concernant.

1.3 La conjecture de cardinalité

Après avoir démontré ce théorème, Beigel a conjecturé un résultat plus fort. Plutôt que de demander le calcul des 2^n valeurs de la fonction indicatrice, on pourrait restreindre l'hypothèse simplement au calcul du nombre d'éléments parmi $(x_1, \ldots, x_{2^n})$ qui sont dans A.

On note $\#A$ le cardinal d'un ensemble A, et on définit la fonction $\#_{2^n}^A$ ainsi :

$$\#_{2^n}^A \ : \left| \begin{array}{ccc} \mathbb{N}^{2^n} & \longrightarrow & \{0 \ldots 2^n\} \\ (x_1, \ldots, x_{2^n}) & \longmapsto & \#\{i \in \mathbb{N} \mid x_i \in A\} \end{array} \right.$$

Et on exprime alors la :

[1.B] Conjecture ***(de la cardinalité, 1987)***

Soit A une partie de $\mathbb{N}$.
Si $\#_{2^n}^A$ peut être calculée en faisant au plus n requêtes à un oracle B, alors A est récursif.

Comme nous allons le voir, cette conjecture est vraie. Nous nous baserons sur la démonstration donnée par M. Kummer dans [Kum92].

2 Le théorème de cardinalité

Dans [Kum92], M. Kummer ne prouve pas directement la conjecture [1.B]. En effet, il prouve le *théorème de cardinalité*, duquel on déduit la conjecture. Nous allons donc énoncer ce théorème, expliquer comment la conjecture s'y ramène, et nous le prouverons ensuite.

2.1 Enoncé du théorème

De manière abrupte, le caractère « nombre borné de requêtes à un oracle » n'est pas utilisable directement. Il faut voir cette condition de manière plus fonctionnelle et plus souple. Pour cela, on pose la :

[2.A] Définition ***(fonction n-énumérable)***

On dit qu'une fonction $f : \mathbb{N}^p \longrightarrow \mathbb{N}$ *est n-énumérable s'il existe une machine de Turing M telle que, sur chaque entrée* $(x_1, \ldots, x_p)$*, M écrit sur sa bande de sortie au plus n entiers tels que l'un d'eux au moins est* $f(x_1, \ldots, x_p)$*. M ne termine pas forcément, mais elle écrit nécessairement* $f(x_1, \ldots, x_p)$ *au bout d'un certain temps.*

Cette notion de *n-énumérabilité* traduit l'idée d'un calcul approché : on obtient un ensemble dans lequel il y a la réponse de la fonction, mais on ne sait pas *a priori* quel est cet élément. C'est bien plus souple que la récursivité pure où on oblige à connaître la valeur *exacte* de la fonction (une fonction récursive est m-énumérable pour tout $m \geq 1$).

Le théorème de cardinalité de Kummer s'exprime alors ainsi :

[2.B] Théorème ***(de cardinalité, 1992)***

Soient A une partie de $\mathbb{N}$ *et* $m \geq 1$*.*
Si $\#_m^A$ *est m-énumérable, alors A est récursif.*

La force de ce théorème est d'exprimer que même en demandant à $\#_m^A$ d'être « moins » que récursive, on aboutit toujours à la récursivité de A.

2.2 Fonctions calculées par requêtes bornées et fonctions partielles récursives

Afin de lier le théorème de cardinalité et la conjecture de Beigel, nous devons voir les fonctions à requêtes bornées sous une lumière plus formelle.

[2.C] Proposition ***(Lien avec les fonctions partielles récursives)***

Soient $p \geq 1$*, et une fonction* $f : \mathbb{N}^p \longrightarrow \mathbb{N}$*. On suppose que f peut être calculée avec au plus n requêtes à un oracle B.*
Alors il existe un ensemble S de fonctions partielles récursives tel que $\#S \leq 2^n$ *et :*

$$\forall x \in \mathbb{N}^p, \quad \exists h \in S \quad tq \quad f(x) = h(x)$$

Preuve de [2.C].

Supposons que M^B soit une machine de Turing comme dans les hypothèses, c'est-à-dire que M^B calcule f en faisant au plus n appels à l'oracle B. On rappelle que l'oracle B répond à chaque requête par 0 ou 1.

L'exécution de M^B sur un certain x de $\mathbb{N}^p$ génère une suite de réponses de B que l'on peut représenter comme un mot sur $\{0, 1\}$ de longueur n, qu'on appellera une *chaîne de réponse* (on rajoute des 0 à la fin s'il y a eu moins de n appels à B). Clairement, il y a 2^n chaînes de réponses différentes puisqu'il y a 2^n mots de longueur n sur $\{0, 1\}$.

L'idée est de créer une fonction par chaîne de réponse r. Chacune d'elle a pour tâche de simuler M^B comme si les requêtes à B retournaient toujours r. On note h_r la fonction ainsi obtenue. On pose alors l'ensemble :

$$S = \{h_r \quad | \quad r \in \{0,1\}^n\}$$

Par définition, on a bien $\#S \leq 2^n$.

Pour x fixé dans $\mathbb{N}^p$, le calcul de M^B sur x conduit à une chaîne de réponses $r(x)$ de B. Alors, $h_{r(x)}(x) = M^B(x) = f(x)$. Donc il existe bien h dans S qui coïncide avec f en x.

Reste à voir que quelque soit r, h_r est une fonction partielle récursive. Soit x un élément de $\mathbb{N}^p$ et r une chaîne de réponse.

- Si $r \neq r(x)$: rien ne garantit que le calcul $h_r(x)$ ne termine puisque ce calcul n'a alors plus rien à voir avec une quelconque exécution de M^B. Cependant, lorsqu'il termine, $h_r(x)$ est obtenu par machine de Turing, donc de manière récursive.
- Si $r = r(x)$: $h_r(x)$ est bien sûr obtenu de manière récursive comme exécution de M^B.

Donc h_r est une fonction partielle récursive. □

2.3 Preuve de la conjecture

Dans cette partie, nous acceptons le théorème de cardinalité [2.B] et avons pour but de montrer que la conjecture [1.B] en découle.

Supposons que $\#_{2^n}^A$ soit calculable avec au plus n requêtes à un oracle B. Alors, d'après [2.C], on dispose d'un ensemble S de fonctions partielles récursives de cardinal au plus 2^n et tel que :

$$\forall\,(x_1,\ldots,x_{2^n}) \in \mathbb{N}^{2^n}, \quad \exists h \in S, \quad \#_{2^n}^A\,(x_1,\ldots,x_{2^n}) = h\,(x_1,\ldots,x_{2^n})$$

À l'aide de S, nous allons montrer que $\#_{2^n}^A$ est 2^n-énumérable, ce qui prouvera la conjecture après application de [2.B].

On définit une machine de Turing M qui exécute en parallèle toutes les fonctions h de S. Dès qu'une fonction h retourne un résultat, il est écrit sur la bande de sortie (s'il n'y figure pas déjà).

Prenons alors x dans $\mathbb{N}^p$. L'exécution de M sur x aboutit à l'écriture de moins de 2^n valeurs (car $\#S \leq 2^n$), parmi lesquelles figure $f(x)$ par [2.C]. Donc $\#_{2^n}^A$ est 2^n-énumérable. Par [2.B], on obtient que A est récursif ce qui prouve la conjecture.

Notons qu'il est possible que l'exécution de M ne termine pas, mais ce n'est pas un problème puisque la définition [2.A] ne le requiert pas. □

3 Preuve du théorème de cardinalité

La démonstration est basée sur une étude assez fine des propriétés de certains arbres binaires.

Dans un premier temps, nous allons montrer des lemmes purement formels sur des arbres binaires. Puis nous verrons ensuite comment le problème de la récursivité de A s'exprime lui-même sous la forme de tels arbres. Enfin, nous conclurons en appliquant ces lemmes à notre situation.

3.1 Arbres binaires, branches récursives et rang

3.1.1 Codage

On décide de coder les arbres binaires finis ou non sur l'alphabet $\{0,1\}$.

Pour cela, on commence par coder un noeud de la manière suivante : en partant de la racine, on suit le chemin vers le noeud et on code 0 à chaque bifurcation à gauche, 1 à chaque bifurcation à droite (s'il n'y a qu'un seul fils à la bifurcation, on note 0). La racine est le mot vide ε. Ainsi, un noeud est bien codé par un mot sur $\{0,1\}$. L'exemple qui suit illustre bien cette définition :

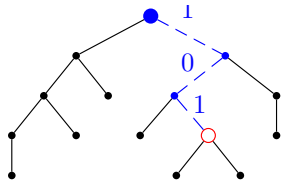

Figure 1: Le codage du noeud creux en rouge est 101 (chemin pointillé bleu)

On définit maintenant le codage d'un arbre simplement comme l'ensemble des codages de tous ses noeuds. Ainsi, le codage de l'arbre de la figure 1 est :

$$\{\underbrace{0, 00, 01, 000, 001, 0000}_{\text{sous-arbre gauche}}, \underbrace{1, 10, 11, 100, 101, 110, 1010, 1011}_{\text{sous-arbre droit}}\}$$

.

On remarque que pour qu'un ensemble de mots finis sur $\{0,1\}$ soit un arbre, il suffit qu'il soit clos par préfixe.

Notre définition des arbres englobe les arbres de hauteur infinie : on peut très bien avoir un arbre où la taille des mots n'est pas bornée. Par exemple $\{0^k | k \in \mathbb{N}\}$ (au sens de « 0 répété k fois ») est un arbre de hauteur infinie (chaque noeud a un fils).

Pour gérer le cas des arbres infinis, on appelle ***branche*** d'un arbre T un mot b sur $\{0,1\}$ tel que tout préfixe fini b soit dans T. Cette définition étend simplement, dans le formalisme défini plus haut, l'idée intuitive de noeud à l'infini dans un arbre infini. En effet, une branche de longueur infinie est un chemin infini dans un arbre infini (qui n'a pas le droit de remonter).

3.1.2 Branches récursives

Comme on vient de le voir, une branche infinie est un mot infini sur le langage $\{0,1\}$ et on peut donc la voir comme la fonction indicatrice d'une certaine partie X de $\mathbb{N}$ (si la branche est finie, on peut la compléter avec des 0 en un mot infini). On pose alors la définition suivante (où $b[i]$ représente le i-ème caractère du mot b) :

[3.A] Définition ***(branche récursive)***

Soient $T \subset \{0,1\}^*$ un arbre et b une branche de T. On note :

$$X = \{i \in \mathbb{N} \quad | \quad b[i] = 1\} \subset \mathbb{N}$$

On dit que b est une branche récursive de T si X est un ensemble récursif.

La preuve s'articule autour de cette notion : nous allons montrer que A est l'ensemble sous-jacent d'une branche récursive d'un arbre bien choisi.

3.1.3 Rang d'un arbre binaire

Afin d'établir notre théorème, il nous faut avoir des moyens de caractériser certains arbres binaires particuliers sur lesquels on peut déduire des résultats qui nous seront utiles.

Lorsqu'on travaille avec des arbres infinis, il n'est plus question de mesurer leur taille à partir de leur hauteur. Cependant, l'idée de mesurer s'ils sont grands ou non n'est pas vouée à l'échec : il suffit de passer par la notion de *plongement*.

3.1.3.1 Plongements

Dans la suite, on notera B_n l'arbre binaire complet de hauteur n ($2^{n+1} - 1$ sommets).

Informellement, on dit que B_n ***se plonge dans*** un arbre T si l'arbre T contient une « copie » de B_n. Puis on définit le ***rang*** de T comme le maximum des n tels que B_n se plonge dans T. C'est ce *rang* qui mesure en quelque sorte la taille de T, sa « complexité ».

Passons à la formalisation de ce point de vue.

Dans tout ce qui suit, on note par " . " la concaténation de deux mots.

[3.B] Définition ***(plongement)***

Soient T un arbre et $n \geq 1$.
On dit que $f : B_n \longrightarrow T$ est un plongement de B_n dans T si pour tout mot w de taille au plus $n-1$:

- *$f(w).0$ est préfixe de $f(w.0)$*
- *$f(w).1$ est préfixe de $f(w.1)$*

De plus, si un mot e de T est préfixe de $f(\varepsilon)$, on dit que f est un plongement par dessus *e.*

Sans surprise, on pose alors la :

[3.C] Définition ***(plongeabilité)***

On dit que B_n se plonge dans un arbre T (resp. par dessus $e \in T$) s'il existe un plongement de B_n dans T (resp. par dessus e).

Il faut voir que cette définition est très intuitive. En effet, elle traduit le fait qu'on a une correspondance entre des noeuds de B_n et de T telle que lorsqu'on avance à gauche (resp. à droite) dans B_n, le chemin correspondant dans T a aussi avancé vers la gauche (resp. droite), mais possiblement de plusieurs étages. On comprend donc bien pourquoi on dit que T contient une copie de B_n.

Exerçons notre intuition sur un exemple :

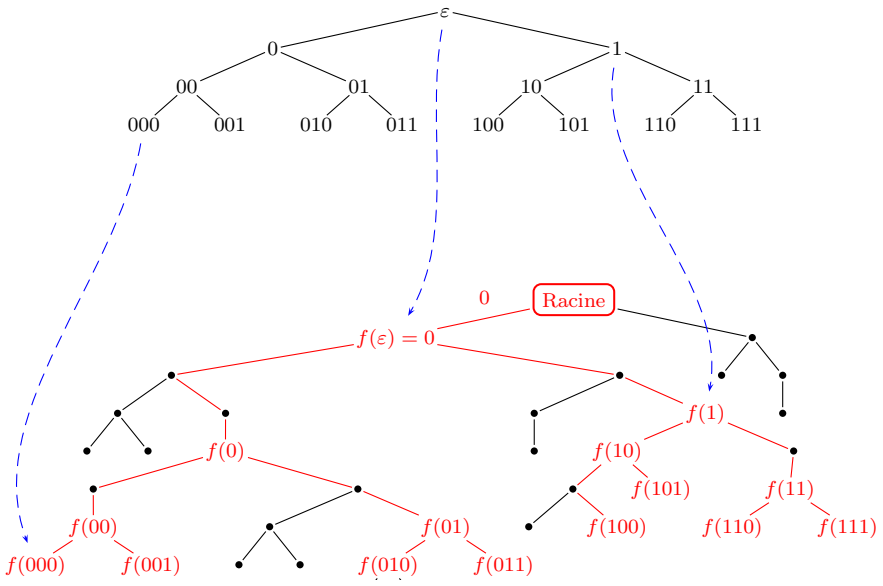

Figure 2: Un plongement de B_3 dans un arbre binaire, par dessus le noeud 0

3.1.3.2 Rang

[3.D] Définition *(Rang)*

Soit T un arbre. On définit le rang (fini ou non) de T par :

$$\mathsf{rg}\, T \quad = \quad \sup_{n\geq 1} \{n \in \mathbb{N}^* \mid B_n \text{ est plongeable dans } T\}$$

Cette définition recouvre bien l'idée qu'on a de mesurer la densité de l'arbre. Par exemple, disons que l'arbre précédent possède une branche infinie pour devenir :

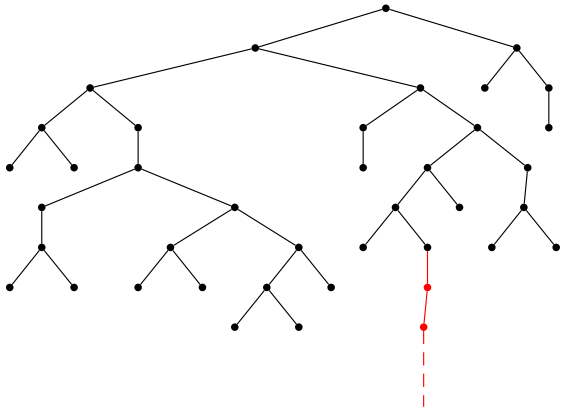

Figure 3: Cet arbre binaire infini est de rang 3

Comme on l'a vu ci-dessus, B_3 se plonge dans cet arbre. Cependant, il est visiblement clair que B_4 ne s'y plonge pas. Et on peut considérer que cet arbre n'est en effet pas fondamentalement plus compliqué que celui la figure 2.

3.2 Quelques résultats formels

Afin de clarifier la structure de la preuve, nous montrons d'abord les lemmes qui lui sont nécessaires, indépendamment du contexte où ils seront utilisés.

3.2.1 Arbres de rang fini

[3.E] Proposition *(Branches récursives et rang fini)*

Soit T un arbre de rang fini. On suppose de plus qu'il est récursivement énumérable, c'est-à-dire que c'est un ensemble récursivement énumérable vu comme langage sur $\{0,1\}$. Alors chacune des branches de T est récursive.

Preuve de [3.E].

Soit b une branche de T. Bien sûr, on suppose qu'elle est infinie sinon elle est clairement récursive. On pose

$$k_0 = \mathsf{sup}\,\{\, n \in \mathbb{N} \mid B_n \text{ se plonge dans } T \text{ par dessus tout préfixe de } b\,\}$$

k_0 est bien défini puisqu'on a par définition la relation $k_0 \leq \mathsf{rg}\, T < \infty$. On choisit alors un noeud e_0 préfixe de b tel que B_{k_0+1} ne se plonge pas dans T par dessus e_0 (son existence est claire par la définition de k_0). On dispose alors du :

[3.F] Lemme

Soit f un plongement de B_{k_0} dans T par dessus e_0. Alors $f(\varepsilon)$ est nécessairement préfixe de b.

Acceptons provisoirement ce lemme pour terminer la preuve de la proposition.

L'algorithme suivant permet d'obtenir $b[i]$ pour $i \in \mathbb{N}$ ce qui montre que t est récursive (notation : $|w|$ est la longueur du mot w) :

- **Si** $i \leq |e_0|$: $b[i]$ peut être obtenu par accès à une table finie précalculée.
- **Sinon** $i > |e_0|$: on énumère T et on cherche un plongement f de B_{k_0} par dessus e_0 tel que $|f(\varepsilon)| \geq i$. D'après le lemme [3.F], on sait que $b[i] = f(\varepsilon)[i]$ puisque $f(\varepsilon)$ est préfixe de b.

  Il reste tout de même à justifier le fait qu'on puisse trouver ce plongement. Or, c'est clair : on sait par hypothèse qu'il existe, et le trouver revient à choisir un nombre fini de branches finies dans b infinie. Donc en attendant assez longtemps, l'énumération de T nous fournira les branches voulues. Un algorithme naïf peut alors les trouver. Cet algorithme termine bien en temps fini car on exécute la recherche de plongement après chaque nouveau mot énuméré par T jusqu'à ce qu'un plongement soit trouvé.

Il reste maintenant à faire la :

Preuve de [3.F].

Supposons qu'on dispose de $f : B_{k_0} \longrightarrow T$ plongement par dessus e_0 et tel que $f(\varepsilon)$ ne soit pas préfixe de b.

Soit $d \in b$ le préfixe commun maximal de $f(\varepsilon)$ et de b. Par hypothèse, e_0 est préfixe de $f(\varepsilon)$, et e_0 est préfixe de b. Donc e_0 est préfixe de d.

Or, par définition de k_0, B_{k_0} se plonge dans T par dessus $d.c_d$, où c_d dénote le caractère suivant d dans b (existe car b est infinie). Par ailleurs, B_{k_0} se plonge également dans T par dessus $d.(1 - c_d)$ puisque $d.(1 - c_d)$ est préfixe de $f(\varepsilon)$.

On en déduit que B_{k_0+1} se plonge dans T par dessus d. Mais e_0 est préfixe de d, donc il existe un plongement de B_{k_0+1} dans T par dessus e_0. Cela contredit la définition de e_0 et montre donc le lemme.

La figure qui suit a pour vocation de clarifier le raisonnement naturel, mais un petit peu technique, qu'on vient d'utiliser.

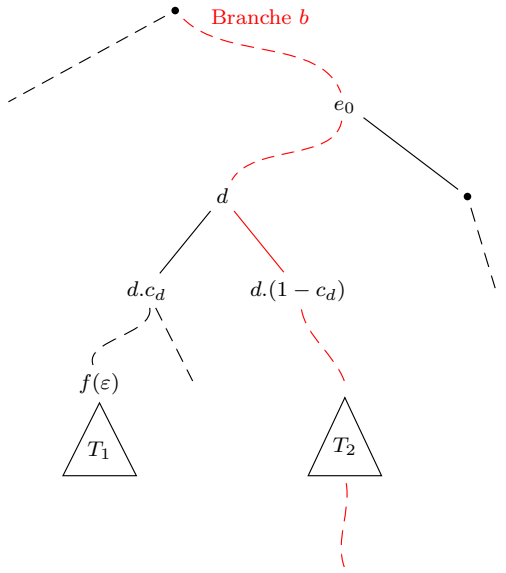

Figure 4: Illustration de la preuve de [3.F] (B_{k_0} se plonge dans T_1 et T_2)

□

3.2.2 2-coloriages

On appelle 2-coloriage de T l'assignation d'une couleur à chaque noeud de T, 0 ou 1 par exemple. On dispose d'un lemme purement combinatoire :

[3.G] Proposition ***(2-coloriage)***
Soit $k \geq 1$. Pour tout 2-coloriage de B_{2k}, il existe un plongement f de B_k dans B_{2k} tel que tous les noeuds de $f(B_k)$ soient de la même couleur.

Nous accepterons ce résultat qui paraît assez intuitif ; sa preuve nous éloignerait trop du sujet.

3.2.3 Lemme de choix

Introduisons une fonction $h : \mathbb{N}^{*2} \longrightarrow \mathbb{N}$ définie par récurrence comme suit :

$$\begin{cases} h(n, 2n-1) & = & 0 & \text{pour} & n \in \mathbb{N}^* \\ h(n, i-1) & = & 2(h(n,i)+1) & \text{pour} & n \in \mathbb{N}^* \text{ et } 1 \leq i \leq 2n-1 \\ h(n,j) & = & 0 & \text{sinon} & \end{cases}$$

Enfin, on pose $k(n) = h(n,0) = 4^n - 2$. On énonce alors la :

[3.H] Proposition ***(Choix)***
Soit $n \geq 1$. Soit T un arbre tel que $B_{k(n)}$ se plonge dans T. Alors il existe des noeuds $(t_1, \ldots, t_{n+1})$ de T, des entiers $(x_1, \ldots, x_n)$, et $b \in \{0,1\}$ tels que :

$$\forall j \in [\![1, n+1]\!],\ \forall i \in [\![1,n]\!],\quad \begin{cases} t_j(x_i) = 1-b & si \quad i < j \\ t_j(x_i) = b & si \quad i \geq j \end{cases}$$

En particulier, $\{ \sum_{i=1}^{n} t_j(x_i) \mid 1 \leq j \leq n+1 \} = [\![0, n]\!]$.

Preuve de [3.H].

Afin de rendre plus clair le résultat de l'énoncé, faisons un petit dessin. La conclusion de la proposition peut-être schématisée comme suit :

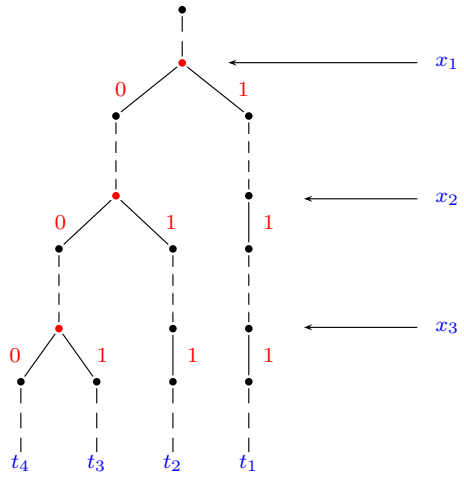

Figure 5: Objectif de construction de la preuve

Pour ceci, nous allons raisonner assez finement sur l'arbre T. Nous allons faire une récurrence à n fixé sur i, pour i de 1 à $2n-1$. Nous avons des renseignements sur $B_{h(n,0)} = B_{k(n)}$, et nous construisons par récurrence des mots w_i et s_i de T, un indicateur b_i qui vaut 0 ou 1, et surtout un plongement $f_i : B_{h(n,i)} \longrightarrow T$. À partir de ces constructions, nous obtiendrons facilement les objets de la proposition.

Le principe est de formaliser la construction esquissée dans la figure 5.

On sait par hypothèse que $B_{k(n)}$ se plonge dans T. Notons f_0 un tel plongement. Ensuite, on avance dans l'arbre $B_{k(n)}$ en faisant à chaque étape une bifurcation « décisive » : elle laisse à sa droite un chemin qui sera destiné à ne contenir plus que des 1, et la construction bifurque à gauche pour avoir un 0 et continuer plus loin. On voit bien qu'à la fin du processus, on arrive à une situation comme dans la figure 5.

Fixons i dans $[\![1, 2n-1]\!]$. Par hypothèse de récurrence au rang i, on dispose d'un plongement $f_{i-1} : B_{h(n,i)} \longrightarrow T$. Soit s une feuille de $B_{h(n,i-1)}$ telle que $f_{i-1}(s)$ soit de taille maximale, et $s_i = f_{i-1}(s)$. De par sa maximalité, s_i induit un 2-coloriage de $B_{h(n,i-1)}$: tout noeud interne e est colorié avec le nombre $s_i[f_{i-1}(e)]$, et chaque feuille est coloriée arbitrairement par 0.

D'après [3.G] et la définition de h, on sait qu'il existe un plongement $g_{i-1} : B_{h(n,i)+1} \longrightarrow B_{h(n,i-1)}$ tel que $g_{i-1}(B_{h(n,i)+1})$ soit monochromatique.

Posons maintenant $w_i = f_{i-1}(g_{i-1}(\varepsilon))$, et $b_i = s_i(|w_i|)$.

Enfin, on pose $f_i(u) = f_{i-1}(g_{i-1}((1-b_i).u))$ pour $|u| \leq h(n,i)$.

Schématisons la construction qu'on vient de faire :

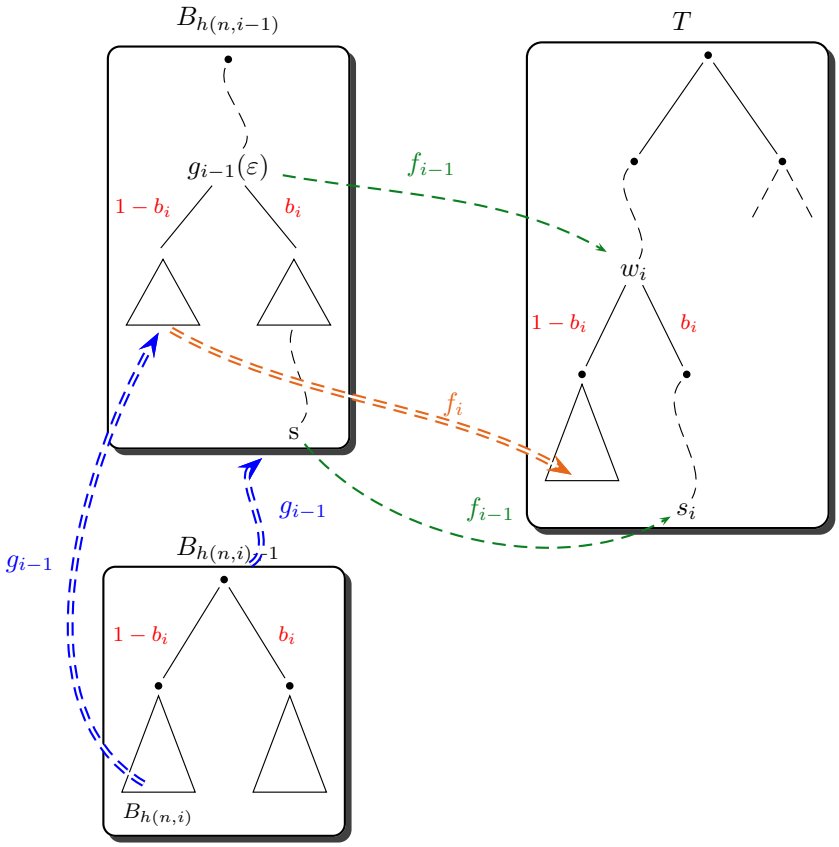

Figure 6: Illustration du procédé de construction par récurrence de la preuve

Quelques propriétés suivent immédiatement des définitions et se comprennent bien sur la figure qui précède (notation : $u \sqsubseteq v$ signifie que u est préfixe de v) :

1. $\mathbf{f_i(B_{h(n,i)}) \subset f_{i-1}(B_{h(n,i-1)})}$:

$$f_i(B_{h(n,i)}) \;=\; f_{i-1}(\underbrace{g_{i-1}(\overbrace{(1-b_i).B_{h(n,i)}}^{\in B_{h(n,i)+1}})}_{\in B_{h(n,i-1)}})$$

2. $\mathbf{w_i.(1-b_i) \sqsubseteq w_{i+1}}$:

$$\begin{array}{rcl} f_{i-1}(g_{i-1}(\varepsilon)).(1-b_i) & \sqsubseteq & f_{i-1}(\underbrace{g(\varepsilon).(1-b_i)}_{\sqsubseteq g(1-b_i)}) \\ & \sqsubseteq & f_{i-1}(g_{i-1}(1-b_i)) \\ & \sqsubseteq & f_i(\varepsilon) \\ & \sqsubseteq & w_{i+1} \end{array}$$

3. $\mathbf{w_i.(1-b_i) \sqsubseteq s_{i+1}}$:

   Cela vient juste du fait que $w_{i+1} \sqsubseteq s_{i+1}$, et 2.

4. $\mathbf{\forall i \geq j,\ s_j(|w_i|) = b_j}$:

   À l'étape j, on restreint le choix des noeuds suivants selon une coloration dictée par s_j. Il est donc naturel que les noeuds suivants choisis dans la construction soient toujours de cette couleur, qui est b_j.

Avec tout cela en poche, concluons.

On a une famille $(b_i)_{1 \leq i \leq 2n-1}$ d'éléments de $\{0,1\}$, donc d'après le lemme des tiroirs, il existe $b \in \{0,1\}$ et n indices $1 \leq i_1 < \ldots < i_n \leq 2n-1$ tels que $b_{i_m} = s_{i_m}(|w_{i_m}|) = b$.

Alors on pose $t_m = s_{i_m}$ et $x_m = |w_{i_m}|$ pour $1 \leq m \leq n$, et $t_{n+1} = w_{i_n}.(1-b)$.

Ceci fait, les équations voulues dans l'énoncé de la proposition découlent directement de la définition des éléments et des remarques 1, 2, 3 et 4 ci-dessus.

□

3.3 La preuve

Ici, nous utilisons tous les résultats introduits jusqu'ici pour prouver le théorème de cardinalité.

On se munit donc d'un ensemble A et on suppose que $\#_m^A$ m-énumérable pour un certain $m \geq 1$. On note φ cette m-énumération. Pour $(x_1, \ldots, x_m)$ fixés, on notera $\varphi(x_1, \ldots, x_m)$ l'ensemble des valeurs énumérées par φ sur $(x_1, \ldots, x_m)$.

Nous allons montrer que χ_A est une branche d'un arbre binaire récursivement énumérable de rang fini. Ainsi, par application de [3.E], on obtiendra que A est un ensemble récursif.

Soit l'ensemble de mots sur $\{0,1\}$:

$$T_A \;=\; \{t \in \{0,1\}^* \mid \forall\, (x_1, \ldots, x_m)\,,\ 1 \leq x_1 < \ldots < x_m \leq |t|,\ \sum_{i=1}^{m} t[x_i] \in \varphi\,(x_1, \ldots, x_m)\}$$

T_A est visiblement clos par préfixe, donc c'est un arbre binaire. De plus, T_A est récursivement énumérable car étant donné un mot t, on dispose d'un algorithme évident répondant *oui* si le mot est dans T_A (il peut boucler si $t \notin T_A$) : sur chaque famille $(x_1, \ldots, x_m)$, on réalise le test de la condition. Si t est dans T_A, alors au bout d'un certain temps, pour chaque $(x_1, \ldots, x_m)$, $\sum_{i=1}^{m} t[x_i]$ est énuméré par φ et on passe à la famille $(x_1, \ldots, x_m)$ suivante, jusqu'à les avoir toutes faites (il y en a un nombre fini car les x_i sont bornés par $|t|$).

De plus, si on voit χ_A comme un mot infini sur $\{0,1\}$ (*ie.* $\chi_A[i] = \chi_A(i)$), on a :

$$\begin{aligned} \forall\,(x_1,\ldots,x_m)\,,\ 1 \leq x_1 < \ldots < x_m \leq |t|,\ \sum_{i=1}^{m} \chi_A[x_i] &= \#_m^A(x_1,\ldots,x_m) \\ &\in \varphi\,(x_1,\ldots,x_m) \end{aligned}$$

Donc χ_A est une branche de l'arbre récursivement énumérable T_A. Reste à voir que T_A est de rang fini et on obtiendra que χ_A est récursive.

Supposons que $\mathsf{rg}\,T_A \geq k(m)$. Alors par le lemme du choix [3.H] :

$$\exists\,(t_1,\ldots,t_{m+1}) \in T_A,\ 1 \leq x_1 < \ldots < x_m,\ \left\{ \sum_{i=1}^{m} t_j[x_i] \quad | \quad j \in [\![1, m+1]\!] \right\} \subset [\![0, m]\!]$$

Mais alors, ceci implique que $[\![0,m]\!] \subset \varphi\,(x_1,\ldots,x_m)$. C'est absurde car φ est un m-énumérateur, donc son image est formée d'au plus m éléments.

Donc $\mathsf{rg}\,T_A < k(m)$, et en particulier $\mathsf{rg}\,T_A$ est fini. Il s'ensuit que χ_A est récursive par [3.E] comme on l'a déjà vu, et donc que A est récursif. Le théorème est montré. □

Bibliographie

[Bei87] Beigel. *Query-limited reducibilities.* PhD thesis, Université de Stanford, 1987.

[Car07] O. Carton. Cours langages formels, calculabilité et complexité. ENS, 2007.

[Kum92] M. Kummer. Proof of beigel's cardinality conjecture. *The journal of symbolic logic*, 57(2):pp. 677–681, Juin 1992.

[Odi89] P. Odifreddi. *Classical recursion theory*, volume 125 of *Studies in Logic and the foundations of mathematics.* North-Holland, Amsterdam, 1989.