Chapitre 1

Modèles de calcul

AVERTISSEMENT : version préliminaire des notes du cours fondamental 2, susceptible de corrections et d'ajouts.

L'objet de cette partie est de préciser les notions d'effectivité en mathématiques en proposant plusieurs modélisations de la notion de fonction et d'ensemble calculables. On va introduire essentiellement trois modèles différents, à savoir :

— la définition de Kleene des fonctions calculables : les fonctions μ-récursives ;
— les fonctions calculables par machines à registres ;
— les fonctions calculables par machines de Turing.

On montre ensuite que ces trois approches de la calculabilité (il en existe bien d'autres) qui mettent en jeux, a priori, des notions de ressources différentes mènent à des notions équivalentes de fonction calculable. Les notes se poursuivent ensuite par une introduction aux résultats de base de la théorie de la calculabilité.

La première approche présentée est celle des fonctions μ-récursives. Les objets de référence sont les fonctions sur les entiers naturels. On considère dans ce cadre qu'une fonction est calculable si elle appartient à un certain ensemble de base (fonctions constantes, successeur, ...) ou si elle peut être obtenue à partir de l'ensemble de fonctions de bases par composition, définition par récurrence ou d'autres schémas que l'on détaillera. Ces fonctions sont intuitivement calculables, mais, contrairement aux définitions que l'on verra ensuite, le calcul reste implicite. Cette approche élégante, fournit un intermédiaire commode pour montrer que les diverses notions de fonctions calculables sont équivalentes. Mais la formalisation du calcul s'avère très vite indispensable pour prolonger l'étude. Il s'avère que n'importe quel modèle de calcul convient et conduit aux mêmes résultats, pourvu qu'il soit suffisamment riche.

Notation. Dans la suite, un vecteur de paramètres sera alternativement désigné par $(x_1, ..., x_p)$ ou par $\overline{x}$ suivant les situations. On parle d'*arité* pour désigner le nombre de paramètres des fonctions et relations.

1.1 Fonctions récursives primitives

Les fonctions récursives primitives sont essentiellement les fonctions qui se calculent par récurrence sur un argument entier, et les composées de celles-ci. Elles ont été introduites dans les années 1920, et les mathématiciens se sont rendu compte assez vite qu'elles ne pouvaient représenter toutes les fonctions calculables (Ackermann, 1926), même si « beaucoup » de fonctions calculables « usuelles » sur les entiers sont récursives primitives.

Définition 1.1.1 L'ensemble des *fonctions récursives primitives* est le plus petit sous-ensemble de l'ensemble des fonctions à plusieurs arguments entiers à valeurs entières ($\bigcup_{p\in\mathbb{N}^*} \mathbb{N}^{\mathbb{N}^p}$) qui satisfait les conditions suivantes :

i. il contient la fonction *nulle* $\lambda x.0 : \mathbb{N} \to \mathbb{N}$, la fonction *successeur* $\mathrm{s} : \mathbb{N} \to \mathbb{N}$ et les *projections* p_i^k : $\mathbb{N}^k \to \mathbb{N}$ $(1 \leq i \leq k)$, définies par $p_k^i(x_1, \ldots, x_k) = x_i$;

ii. il est clos par le *schéma de composition* :
si $h : \mathbb{N}^p \to \mathbb{N}$, et $g_1, \ldots, g_p : \mathbb{N}^n \to \mathbb{N}$ sont récursives primitives, alors $f : \mathbb{N}^n \to \mathbb{N}$ définie par $f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_p(x_1, \ldots, x_n))$ est récursive primitive.

iii. il est clos par le *schéma de récurrence primitive* :
si $g : \mathbb{N}^p \to \mathbb{N}$ et $h : \mathbb{N}^{p+2} \to \mathbb{N}$ sont récursives primitives, alors $f : \mathbb{N}^{p+1} \to \mathbb{N}$ est récursive primitive, f définie par :

$$f(a_1, \ldots, a_p, 0) = g(a_1, \ldots, a_p)$$
$$f(a_1, \ldots, a_p, x+1) = h(a_1, \ldots, a_p, x, f(a_1, \ldots, a_p, x)) .$$

On parlera de *définition récursive primitive* d'une fonction, pour une définition de la fonction qui utilise ces trois clauses.

Un prédicat P sur $\mathbb{N}^p$ (resp. un sous-ensemble E de $\mathbb{N}^p$) est un *prédicat récursif primitif* (resp. *un sous-ensemble récursif primitif*), quand sa fonction caractéristique est récursive primitive.

On rappelle que la fonction caractéristique d'un ensemble est définie par $\chi_A(\overline{x}) = 1$ si $\overline{x} \in A$ et $\chi_A(\overline{x}) = 0$ sinon ; la fonction caractéristique d'un prédicat est celle de l'ensemble des uples pour lesquels le prédicat est vrai.

Plus généralement On dira d'un sous-ensemble de $\bigcup_{p\in\mathbb{N}^*} \mathbb{N}^{\mathbb{N}^p}$ qu'il est *clos par opérations récursives primitives* s'il satisfait les trois clauses **i**, **ii** et **iii** ci-dessus, sans être nécessairement le plus petit. Intuitivement, l'ensemble de toutes les fonctions calculables, doit être clos par opérations récursives primitives. Cela sera démontré quand nous aurons une caractérisation satisfaisante de la notion de fonction calculable.

1.1.1 Exemples de fonctions récursives primitives

Fonctions constantes Pour $n \in \mathbb{N}$, on note s^n la fonction successeur composée n fois. La fonction constante $c_n : \mathbb{N} \to \mathbb{N}$ telle que $c_n(x) = n$ se définit par $c_n(x) = \mathrm{s}^n(\lambda x.0(x))$, en utilisant donc n fois le schéma de composition.

Addition, Multiplication, exponentielle Une définition par récurrence de l'addition est :

$$x + 0 = x \text{ et } x + (y+1) = (x+y) + 1 .$$

et cette définition est récursive primitive. De façon plus explicite, la fonction $+ : \mathbb{N}^2 \to \mathbb{N}$ est définie par

$$+(x, 0) = p^1_1(x) \text{ et } +(x, y+1) = \mathrm{s}(p^3_3(x, y, +(x, y))) .$$

L'addition est bien obtenue à partir des fonctions initiales p^1_1, p^3_1, p^3_3, s, d'une occurrence du schéma de composition, et d'une occurrence du schéma de récurrence primitive.

De même la fonction multiplication $\times : \mathbb{N}^2 \to \mathbb{N}$, définie par récurrence par

$$x \cdot 0 = 0 \text{ et } x \cdot (y+1) = x + x \cdot y$$

est aussi récursive primitive :

$$\times(x, 0) = 0 \text{ et } \times(x, y+1) = +(p^3_1(x, y, +(x, y)), p^3_3(x, y, \times(x, y)))$$

ainsi que la fonction exponentielle définie par

$$x^0 = 0 \text{ et } x^{y+1} = x^y \cdot x .$$

Définitions par récurrence primitive de fonctions à un argument Le schéma de récurrence primitive donné en définition ne permet pas de définir directement des fonctions à un argument. Cependant, on montre facilement qu'il s'étend par :

iii$_0$ Si $b \in \mathbb{N}$, $h : \mathbb{N}^2 \to \mathbb{N}$ est récursive primitive, alors f est récursive primitive, $f : \mathbb{N} \to \mathbb{N}$ définie par

$$f(0) = b$$
$$f(x+1) = h(x, f(x)) .$$

En effet il suffit de définir par récurrence primitive une fonction auxiliaire $aux : \mathbb{N}^2 \to \mathbb{N}$, avec un second argument inutile :

$$aux(a,0) = c_b(a) \text{ et } aux(a,x+1) = h(p_2^3(a,x,f(x)), p_1^3(a,x,f(x))) \; ; f(x) = aux(p_1^1(x), p_1^1(x)).$$

Par exemple la fonction factorielle définie par

$$0! = 1 \text{ et } (x+1)! = (x+1)\cdot x!$$

est récursive primitive en suivant **iii$_0$**. Ce schéma est utilisé au paragraphe suivant pour les fonctions de signe et le prédécesseur.

Signe, prédécesseur, soustraction Les deux fonctions de signe sg et $\overline{\text{sg}}$ qui suivent sont récursives primitives

$$\text{sg}(0) = 0 \text{ et } \text{sg}(x+1) = 1 \; ; \; \overline{\text{sg}}(0) = 1 \text{ et } \overline{\text{sg}}(x+1) = 0$$

($\text{sg}(x+1) = c_1(p_1^2(x, \text{sg}(x))$).
Les fonctions récursives primitives sont partout définies. On définit un prédécesseur nul en 0 et une soustraction tronquée :

$$\text{pred}(0) = 0 \text{ et } \text{pred}(x+1) = x \; ; \; x \dot{-} 0 = x \text{ et } x \dot{-} (y+1) = \text{pred}(x \dot{-} y).$$

avec $\text{pred}(0) = 0$ et $\text{pred}(x+1) = p_1^2(x, \text{pred}(a,x))$.

Dès que l'on a une définition par récurrence sur un argument à partir de fonctions récursives primitives, la fonction obtenue est récursive primitive. Cette restriction aux récurrences à un argument ne peut être levée en toute généralité : on verra qu'il existe des fonctions comme la fonction d'Ackermann (voir 1.2.2 page 13) qui se définissent par récurrence double et ne sont pas récursives primitives. Ainsi une définition naturelle de la fonction $\dot{-}$ par récurrence double est

$$\dot{-}(x+1, y+1) = \dot{-}(x,y), \; \dot{-}(x,0) = x \text{ et } \dot{-}(0,y) = 0 \, .$$

cette définition induit manifestement un calcul, d'ailleurs plutôt plus naturel et plus efficace, de la fonction $\dot{-}$, mais elle ne met pas en évidence que cette fonction est récursive primitive (voir cependant l'exercice 8 page 11).

Comparaison Les prédicats de comparaison $\leq, \geq, <, >, =, \neq$ sont récursifs primitifs, c'est-à-dire que leurs fonctions caractérisques le sont. En effet : $\chi_>(x,y) = \text{sg}(x \dot{-} y)$, $\chi_\leq(x,y) = \overline{\text{sg}}(x \dot{-} y)$, et donc $\chi_=(x,y) = \overline{\text{sg}}(x \dot{-} y) \cdot \overline{\text{sg}}(y \dot{-} x)$, $\chi_\neq(x,y) = \overline{\text{sg}}(\chi_=(x,y))$.

1.1.2 Propriétés de clôtures

Au delà de la définition, l'ensemble des fonctions récursives primitives, et plus généralement tout ensemble clos par opérations récursives primitives, satisfait un grand nombre de propriétés de clôture. On a déjà vu la définition par récurrence primitive des fonctions à un argument. On en détaille quelques autres ci-dessous.

Définition par itération On peut ne pas utiliser tous les arguments dans la récurrence. Par exemple l'addition et la multiplication utilisent le schéma de définition par itération qui est de la forme $f(\overline{x}, 0) = g(\overline{x})$ et $f(\overline{x}, y+1) = h(\overline{x}, f(\overline{x}, y))$. On montre facilement que les fonctions définies ainsi sont récursives primitives.

Sommes et produits bornés Si f de $\mathbb{N}^{p+1} \to \mathbb{N}$ est une fonction récursive primitive, les fonctions g et h de $\mathbb{N}^{p+1} \to \mathbb{N}$ définies par

$$g(x_1, \ldots, x_p, y) = \sum_{i=0}^{y} f(x_1, \ldots, x_p, i) \quad \text{et} \quad h(x_1, \ldots, x_p, y) = \prod_{i=0}^{y} f(x_1, \ldots, x_p, i)$$

sont récursives primitives. En effet (pour la somme) :

$$g(\overline{x}, 0) = f(\overline{x}, 0) \text{ et } g(\overline{x}, y+1) = f(\overline{x}, y+1) + g(\overline{x}, y).$$

(Le schéma de récurrence utilisé est celui de la définition, ce n'est pas une définition par itération).

Opérations logiques L'ensemble des prédicats récursifs primitifs d'arité quelconque est clos par opération booléennes (conjonction, disjonction, négation). Ainsi, si A et B sont des prédicats récursifs primitifs, alors $P(\overline{x},\overline{y}) \wedge Q(\overline{x},\overline{z})$ (pensez à $\chi_P(\overline{x},\overline{y}) \cdot \chi_Q(\overline{x},\overline{z})$) est récursif primitif. La fonction $\overline{\text{sg}}$ permet d'obtenir la négation, et donc la disjonction.

Opérations ensemblistes Les résultats précédents se traduisent immédiatement de façon ensembliste : la classe des sous-ensembles récursifs primitifs de $\mathbb{N}^p$, p fixé, est close par intersection, réunion et passage au complémentaire. La classe des ensembles récursifs primitifs est close par produit cartésien.

Définition par cas Soient f et g deux fonctions récursives primitives et A un ensemble récursif primitif alors la fonction h suivante est récursive primitive :

$$\begin{array}{ll} h(\overline{x}) & = f(\overline{x}) \text{ si } \overline{x} \in A \\ h(\overline{x}) & = g(\overline{x}) \text{ sinon.} \end{array}$$

Cela se voit simplement en remarquant que $h(\overline{x}) = f(\overline{x}) \cdot \chi_A(\overline{x}) + f(\overline{x}) \cdot \chi_{\mathbb{N}^k \setminus A}(\overline{x})$. Plus généralement, si $A_1, ..., A_n$ sont des ensembles récursifs primitifs deux à deux disjoints et $f_1, ..., f_n, g$ des fonctions récursives primitives alors la fonction h suivante est récursive primitive :

$$\begin{array}{ll} h(\overline{x}) & = f_1(\overline{x}) \text{ si } \overline{x} \in A_1 \\ h(\overline{x}) & = f_2(\overline{x}) \text{ si } \overline{x} \in A_2 \\ \vdots & \vdots \\ h(\overline{x}) & = f_n(\overline{x}) \text{ si } \overline{x} \in A_n \\ h(\overline{x}) & = g(\overline{x}) \text{ si } \overline{x} \notin A_1 \cup \cdots \cup A_n. \end{array}$$

Minimisation bornée Soit h une fonction récursive primitive. Une fonction f est obtenue par schéma de *minimisation bornée* à partir de h si elle est définie par :

$$\begin{array}{lcll} f(x_1,\ldots,x_p,y) & = & \text{le plus petit entier } t \leq y \text{ tel que } h(x_1,\ldots,x_p,t) = 0 & \text{s'il existe un tel entier,} \\ f(x_1,\ldots,x_p,y) & = & 0 \text{ s'il n'existe pas de tel entier.} & \end{array}$$

On note l'opération de minimisation bornée :

$$f(\overline{x},y) = \mu t \leq y.[h(\overline{x},t) = 0] \ .$$

La fonction f ainsi obtenue est récursive primitive. En effet :

$$\begin{array}{lcl} f(\overline{x},0) & = & 0 \\ f(\overline{x},y+1) & = & \text{sg}\big(h(\overline{x},0)\big) \cdot \big(f(\overline{x},y) + \overline{\text{sg}}\big(f(\overline{x},y)\big) \cdot \overline{\text{sg}}\big(h(\overline{x},y+1)\big) \cdot (y+1)\big) \end{array}$$

Par composition, si k est récursive primitive, alors f définie par $f(\overline{x}) = \mu t \leq k(\overline{x}).[h(\overline{x},t) = 0]$ est aussi récursive primitive.

Quantifications bornées Si P est un prédicat récursif primitif alors les deux prédicats P_e et P_q définis comme suit sont récursifs primitifs :

$$\begin{array}{l} P_e\, x_1 \ldots x_p\, y \equiv \exists z \leq y\, P\, x_1 \ldots x_p\, y \\ P_q\, x_1 \ldots x_p\, y \equiv \forall z \leq y\, P\, x_1 \ldots x_p\, y \ . \end{array}$$

En effet

$$\begin{array}{l} \chi_{P_e}(\overline{x},y) = \text{sg}(\sum_{z=0}^{y} \chi_P(\overline{x},z)) \\ \chi_{P_q}(\overline{x},y) = \prod_{z=0}^{y} \chi_P(\overline{x},z) \qquad . \end{array}$$

Par composition avec les fonctions de projections, les prédicats (dépendants des mêmes variables que P)

$$\begin{array}{l} \exists z \leq x_i\, P\, x_1 \ldots x_p \\ \forall z \leq x_i\, P\, x_1 \ldots x_p \ . \end{array}$$

sont aussi récursifs primitifs. Plus généralement, on montre par composition que les prédicats

$$\exists z \leq f(\overline{x})\ P\overline{x}$$
$$\forall z \leq f(\overline{x})\ P\overline{x}$$

sont récursifs primitifs dès que le quantificateur est borné par une fonction récursive primitive f.

Les propriétés de clôture de ce paragraphe se généralisent évidemment à tout ensemble de fonctions arithmétiques clos par opérations récursives primitives, puisque seul cette partie de la définition des fonctions récursives primitives a été utilisée.

Exercice 1 Montrer que les sous-ensembles finis et cofinis des $\mathbb{N}^k$, $k \in \mathbb{N}$, sont récursifs primitifs.

Exercice 2 On a vu que l'ensemble des prédicats récursifs primitifs d'arité quelconque est clos sous les opérations booléennes (conjonction, disjonction, négation), (voir page 6). Détailler la démonstration et en déduire que la classe des ensembles récursifs primitifs est close par réunion, intersection, produit cartésien et passage au complémentaire.

1.1.3 Prédicats définissables au premier ordre par quantification bornée

On considère ici un sous-ensemble de prédicats récursifs primitifs qui contient la plupart des prédicats arithmétiques naturels. Appelons $\mathscr{R}$ le plus petit ensemble contenant les prédicats d'addition, de multiplication et clos par opérations booléennes et quantification bornée par un polynôme. En d'autres termes un prédicat $R(x_1, ..., x_k)$ est dans $\mathscr{R}$ s'il est définissable par une formule du premier ordre sur la signature $\{+, \times\}$ et dont toutes les quantifications sont bornées par un polynôme en $x_1, ..., x_k$.

Tout prédicat de $\mathscr{R}$ est récursif primitif (au vu de ce que l'on a déjà prouvé). La classe $\mathscr{R}$ nous fournit un moyen simple (par une définition logique) de montrer que certains prédicats sont récursifs primitifs. Par exemple, un nombre p est premier s'il satisfait :

$$p \text{ est premier } \equiv p \geq 2 \wedge \forall x \leq p\, \forall y \leq p\ \big(x \cdot y = p \rightarrow \big(x = 1 \vee x = p\big)\big)\ .$$

De même, x divise y, noté $x|y$ s'écrit : $x|y \equiv \exists z \leq y\ x \cdot z = y$. De façon générale, la plupart des prédicats arithmétiques naturels sont dans $\mathscr{R}$.

Enfin, en utilisant la clôture par minimisation bornée et par récurrence primitive on montre à partir de l'exemple précédent que la fonction $\mathrm{p} : \mathbb{N} \rightarrow \mathbb{N}$ qui à n associe $\mathrm{p}(n)$ (noté p_n) le $n+1$-ème nombre premier est bien récursive primitive ($\mathrm{p}_0 = 2$, $\mathrm{p}_1 = 3$, ...). Il suffit de remarquer que le $n+1$-ème nombre premier est forcément borné par $\mathrm{p}_n! + 1$ (la factorielle est récursive primitive). La définition de p se fait alors par récurrence :

$$\mathrm{p}(0) = 2 \text{ et } \mathrm{p}(n+1) = \mu x \leq \mathrm{p}(n)! + 1.[x \text{ est premier } \wedge\, x > \mathrm{p}(n)].$$

Exercice 3 (division euclidienne) Montrer que que le prédicat de divisibilité $|$ est récursif primitif, et que les fonctions quotient : $\mathbb{N}^2 \rightarrow \mathbb{N}$ (quotient de la division de n par p), reste : $\mathbb{N}^2 \rightarrow \mathbb{N}$ (reste de la division de n par p) sont des fonctions récursives primitives.

1.1.4 Premiers codages

Les paramètres des fonctions récursives primitives sont des entiers, et ce sera encore le cas pour les diverses notions de fonctions calculables que nous allons étudier. Il est cependant bien évident que la notion de calcul dépasse de loin ce cadre restreint. Dans un sens, un modèle de calcul général doit pouvoir prendre en entrée des objets finis de natures très différentes : nombres autres qu'entiers, mots sur un alphabet fini, structures algébriques finies, graphes, hypergraphes, arbres,... La notion de calcul a intuitivement un sens dans tous ces contextes. De même, clairement, les fonctions calculables doivent pouvoir, à travers une représentation finie (les programmes qui les calculent) être considérées comme des paramètres valides d'autres fonctions calculables.

Dans cette section, on va montrer comment représenter à l'aide d'entiers (on dira souvent « coder »), tout d'abord les couples et n-uplets, puis les listes — c'est-à-dire les suites finies — d'entiers. Ces codages se manipulent par des fonctions récursives primitives, c'est-à-dire que les fonctions usuelles,

par exemple sur les listes (longueur, i-ème élément, sous-liste, etc) sont représentées par des fonctions récursives primitives. De plus ces codages permettent d'établir de nouvelles propriétés de clôture pour les fonctions récursives primitives, définition par récurrence en fonction d'un entier strictement plus petit (qui n'est pas nécessairement le prédécesseur) par exemple. La méthode utilisée pour les listes se généralise à des structures plus complexes comme les arbres étiquetés, ce que l'on verra dans la partie 3.2.

La bijection de Cantor

On appelle α la bijection de Cantor $\mathbb{N}\times\mathbb{N}$ dans $\mathbb{N}$, définie par :

$$\alpha(n,p) = (\sum_{i=0}^{n+p} i) + p = \frac{(n+p+1)(n+p)}{2} + p$$

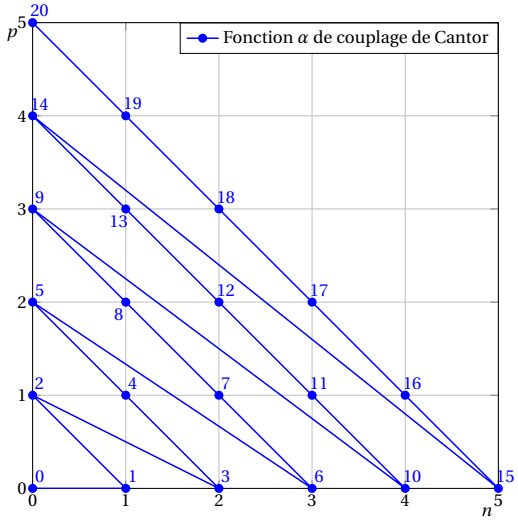
On vérifie facilement que α est bijective, strictement croissante pour ses deux entrées n et p et qu'elle est récursive primitive. Ona également $n \le \alpha(n,p)$ et $p \le \alpha(n,p)$.

L'application étant bijective, on peut définir deux projections π_1 et π_2 vérifiant pour tout entier c et tout couple d'entiers (n,p) :

$$\begin{aligned}&\alpha(\pi_1(c),\pi_2(c)) = c,\\ &\pi_1(\alpha(n,p)) = n\\ &\pi_2(\alpha(n,p)) = p\end{aligned}$$

qui sont également récursives primitives. En effet :

$$\begin{aligned}&\pi_1(x) = \mu z \le x.[\exists t \le x\ \alpha(z,t) = x]\\ &\pi_2(x) = \mu t \le x.[\exists z \le x\ \alpha(z,t) = x].\end{aligned}$$



On peut alors définir par récurrence sur $k \ge 1$ les fonctions $\alpha_k : \mathbb{N}^k \to \mathbb{N}$ par :

$$\begin{aligned}&\alpha_1(n) = n\\ &\alpha_{k+1}(n_1,\dots,n_{k+1}) = \alpha_2(n_1,\alpha_k(n_2,\dots,n_{k+1}))\quad (\text{ en particulier } \alpha_2 = \alpha)\ .\end{aligned}$$

À nouveau on démontre par récurrence sur k que chaque α_k, $k \in \mathbb{N}^*$, est une bijection. Les projections correspondantes, notées π_i^k pour $1 \le i \le k$ sont définies par :

$$\text{pour } 1 \le i < k,\ \pi_i^k = \pi_1 \circ \underbrace{\pi_2 \circ \cdots \circ \pi_2}_{i-1}\ ;\quad \pi_k^k = \underbrace{\pi_2 \circ \cdots \circ \pi_2}_{k-1}\quad (\text{en particulier } \pi_1^2 = \pi_1 \text{ et } \pi_2^2 = \pi_2).$$

Chacune des fonctions α_k, de même que les projections π_i^k, sont récursives primitives par composition. On pose $\alpha_k(x_1,\dots,x_k) = \langle x_1,\dots,x_k\rangle$ et on utilisera le plus souvent cette dernière écriture.

Cette famille de fonctions ne permet pas, telle quelle, de définir une bijection de l'ensemble des suites finies vers $\mathbb{N}$ puisque précisement chacune de ces fonctions est bijective. Par exemple : $\alpha_3(1,0,0) = \alpha_2(1,\alpha_2(0,0)) = \alpha_2(1,0)$. On va modifier un tout petit peu l'approche pour obtenir un codage bijectif des suites finies, les listes de l'informatique.

Un codage bijectif des suites finies

Pour coder les suites finies d'entiers de taille arbitraire, ou listes d'entiers, on utilise à nouveau le codage des couples.