

RÉFLEXION SUR LA NOTION DE CALCUL ET D'ALGORITHME

Le Continu algorithmique

Marc Jambon

Université de la Réunion

Résumé. – Une formalisation satisfaisante de la notion de fonction calculable ne peut se faire que dans un environnement “logique intuitionniste”. On propose dans ce contexte, différents types d’algorithmes : élémentaires, primitifs récursifs et d’autres illustrés par des documents “mathematica” en annexe ; on discute de leur adéquation. On dispose alors d’une bonne base pour définir un Continu algorithmique dans lequel on saura démontrer le théorème de l’éventail sans aucun autre axiome.

Abstract. – A satisfactory formalism of notion of computable function may be only with “intuitionistic logic”. It is proposed in this context, some types of algorithms : elementary, primitive recursive and others illustrated by “mathematica” documents in annexe ; it is discussed on their adequacy. Then, an appropriate basis is available to define an algorithmic Continuum in which we are able to prove the fan theorem whith no other axiom.

Introduction

Le mot *calcul* vient du latin *calculus* caillou. En effet, une tradition très ancienne rapporte qu’un berger comptait ses moutons à l’aide d’un sac de caillou. Par là même, il savait mettre un caillou de plus dans son sac : il avait inventé l’application *successeur* ; il savait enlever un caillou du sac, pourvu que ce soit possible : il avait inventé l’application *prédécesseur* ; il savait répéter quelque chose autant de fois que le contenu de son sac de caillou : il avait inventé ce qui allait devenir la *primitive récursivité*. Si l’on ajoute que son objectif était de comparer le contenu de son sac de caillou à celui de son troupeau, il savait ramener ce test d’égalité à un test entre un sac et un troupeau dont l’un a un contenu vide. À partir de ces considérations, on introduira en III les *applications primitives récursives*.

Le mot, à connotation plus savante, *algorithme* tire son étymologie de Al-Khwarizmi, médecin arabe (780 850) dont l’œuvre consiste, entre autres, en un exposé sur la numération de position apparue en Inde dès le II^{ème} siècle de notre ère. Cette numération plus élaborée, facilite grandement les “quatre opérations” en nombres entiers naturels, c’est même sa raison d’être, elle contient en elle même, de façon cachée, les puissances de 10 et l’addition répétée des chiffres multipliés par les dites puissances successives de 10. C’est ce qui conduira, au paragraphe II, à la notion très récente (deuxième moitié du XX^e siècle) de *fonction élémentaire* due au mathématicien Kalmar.

Un objectif plus ambitieux qui m’a profondément motivé est la mise en place d’un *Continu algorithmique* : penser à des nombres réels engendrés par des suites algorithmiques.

Première partie

Calcul et algorithme

I Logique et fondements

1. Nécessité de la logique intuitionniste

Supposons qu’on dispose d’une notion formelle de fonctions calculables, qu’on appelle fonctions *formellement calculables* qui contienne la fonction nulle, la fonction constante égale à 1.

Soit maintenant n’importe quel problème irrésolu \mathcal{P} , par exemple la conjecture de Goldbach [7]. Définissons la fonction f ainsi : si \mathcal{P} est vrai, f est la fonction nulle, si \mathcal{P} est faux, f est la fonction constante égale à 1 ; voici une fonction qui ne paraît guère calculable au sens intuitif, en effet son calcul équivaldrait à résoudre le problème irrésolu \mathcal{P} . Pourtant, en logique classique, cette fonction apparaît comme *formellement calculable*, en effet, si \mathcal{P} est vrai, f est la fonction nulle, elle est

formellement calculable, si \mathcal{P} est faux, c'est-à-dire $\neg \mathcal{P}$ est vrai, f est la fonction 1, elle est *formellement calculable*.

$\mathcal{P} \vee \neg \mathcal{P} \Rightarrow f$ *formellement calculable*

car

$\mathcal{P} \Rightarrow f$ *calculable* $\wedge \neg \mathcal{P} \Rightarrow f$ *formellement calculable*

Comme le tiers exclu est accepté en logique classique : f *formellement calculable* est vrai.

Il n'y a aucune contradiction, mais un paradoxe au sens que la notion de fonction *formellement calculable* ne répond pas à l'objectif souhaité, à savoir, traduire le calculable au sens intuitif.

Pour échapper à ce paradoxe, il est indispensable d'utiliser une logique sans tiers exclu. Je me référerai donc dans la suite à la *logique intuitionniste* la plus élémentaire telle que proposée dans [7].

2. Génération inductive

Il est peu usuel en mathématiques classiques de faire appel à des *générations inductives*, la génération des nombres entiers naturels à l'aide de 0, *élément de base*, et de successeur *schéma de construction* en est un exemple. L'axiome du *raisonnement par récurrence* traduit formellement que tous les entiers naturels sont engendrés par le procédé ci-dessus décrit. Un autre exemple est proposé en [6]. L'axiome du *raisonnement par induction* n'est en fait qu'une extension du raisonnement par récurrence à toutes les générations inductives.

Axiome du raisonnement par induction. Si une propriété est vraie sur les objets de base d'une génération inductive et qu'elle se transfère par les schémas de construction, alors elle est vraie sur tous les objets engendrés par la génération inductive.

Exemple : voir démonstration du théorème 5 à la fin du paragraphe III.

3. \mathbb{N} et ensembles produits \mathbb{N}^p

3.1. Ensembles supports

On ne se référera pas à une théorie générale des ensembles type Zermelo-Frankel inadaptée à la logique intuitionniste.

On se contentera de l'ensemble \mathbb{N} et de ses produits qu'on appellera *ensembles supports*. On notera \mathbb{N}^p l'ensemble produit de \mathbb{N} , p fois par lui-même où $p \in \mathbb{N}^*$, de même \mathbb{N}^q , \mathbb{N}^r avec $q \in \mathbb{N}^*$, $r \in \mathbb{N}^*$...

3.2. Vocabulaire des applications

On pourra parler d'*application* de \mathbb{N}^p dans \mathbb{N}^q pour une correspondance qui à un élément de \mathbb{N}^p fait correspondre un élément unique de \mathbb{N}^q compatible avec l'égalité ; lorsque f est une telle application : on écrit cet élément $f(x_1, \dots, x_p)$, on dit aussi que f est une application à p arguments. On parlera de *fonction* pour une application de \mathbb{N}^p dans \mathbb{N} .

Exceptionnellement, on pourra utiliser la notation \mathbb{N}^0 et parler d'application de \mathbb{N}^0 dans \mathbb{N}^p pour désigner une *application sans argument* qui se confond avec un élément de l'ensemble support d'arrivée.

Dans les autres cas, on ne sait pas, au départ, s'il existe de telles applications (ou fonctions) et on ne dispose pas d'ensemble d'applications de \mathbb{N}^p dans \mathbb{N}^q . Les axiomes de génération inductive des fonctions *élémentaires* en II et des applications *primitives récursives* en III vont combler cette lacune.

L'égalité $f = g$ entre applications f et g de \mathbb{N}^p dans \mathbb{N}^q sera définie à l'aide du quantificateur universel \forall :

$$\forall x \in \mathbb{N}^p \quad f(x) = g(x).$$

On ne sait prouver que cette égalité vérifie le tiers exclu, aussi on parlera, lorsqu'on saura les engendrer, d'*espèce* d'applications [*élémentaires* respectivement *primitives récursives*] de \mathbb{N}^p dans \mathbb{N}^q .

II Fonctions et algorithmes élémentaires

1. Nombres entiers naturels

On suppose connu les nombres entiers naturels à partir d'une axiomatique riche.

- La structure $(\mathbb{N}, +, \times, \leq)$ et toutes les propriétés usuelles : notamment 0, 1 éléments neutres de l'addition, respectivement de la multiplication...

- L'axiome du *raisonnement par récurrence* [élémentaire].

Soit $\mathcal{P}(n)$ une proposition [égalité entre 0 et une fonction élémentaire (ci-dessous, voir aussi 7)] ou même un prédicat [i.e. avec en plus des quantificateurs] avec la variable libre n , $n \in \mathbb{N}$: si $\mathcal{P}(0)$ et $\forall n (\mathcal{P}(n) \Rightarrow \mathcal{P}(n+1))$ alors $\forall n \mathcal{P}(n)$

- La *soustraction positive* $a \dot{-} b$ soustraction usuelle pour $b \leq a$, prolongée par 0, de telle sorte qu'elle est partout définie. On peut même retrouver \leq à l'aide de $\dot{-}$ et de l'**égalité à 0**, en effet :

$$a \leq b \Leftrightarrow a \dot{-} b = 0$$

$\dot{-}$ peut être directement formalisé par les axiomes :

$$a = b \Leftrightarrow (a \dot{-} b) + (b \dot{-} a) = 0$$

$$(a + b) \dot{-} b = a$$

$$a \dot{-} (a + b) = 0$$

$$(a \dot{-} b) \dot{-} c = a \dot{-} (b + c)$$

Remarque concernant l'axiomatique.

- L'axiomatique peut trouver sa justification expérimentale en considérant \mathbb{N} comme ensemble des cardinaux finis, l'addition traduisant le cardinal de la réunion disjointe, la multiplication traduisant le cardinal d'un produit d'ensembles finis.

• Cette même axiomatique trouve aussi bien sa justification en considérant \mathbb{N} comme ensemble des mots de l'alphabet fini $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ interprétés comme des nombres entiers écrits en base "dix" avec addition, soustraction, multiplication découlant des règles apprises à l'école primaire. Les puissances de "dix" sont un cas particulier de l'utilisation du symbole Π , multiplication répétée (ci-dessous), la somme des chiffres de rang k multipliés par 10^k , est un cas particulier de l'utilisation du symbole Σ , addition répétée (ci-dessous) cf.8 ci-dessous. Ainsi la représentation des entiers naturels en base dix suggère d'accepter soustraction, addition, multiplication, Σ et Π . Ceci va nous conduire à la définition des *fonctions élémentaires* qu'on trouve dans la littérature classique en [11] ou [12], elle est reprise ci-dessous sous forme de génération inductive dans un environnement logique intuitionniste.

2. Axiomes de génération inductive des algorithmes élémentaires

2.1. Algorithmes et fonctions élémentaires

Fonctions de base	
Fonction <i>nulle</i> [i.e. prenant la seule valeur 0] de \mathbb{N} dans \mathbb{N} ou de \mathbb{N}^0 dans \mathbb{N}	
Fonction partout égale à 1 de \mathbb{N} dans \mathbb{N} ou même de \mathbb{N}^0 dans \mathbb{N}	
<i>Addition</i> $+$ de \mathbb{N}^2 dans \mathbb{N}	
<i>Multiplication</i> \times de \mathbb{N}^2 dans \mathbb{N}	
<i>Soustraction positive</i> $\dot{-}$ de \mathbb{N}^2 dans \mathbb{N}	
<i>Projections</i> de \mathbb{N}^p sur chacune de ses coordonnées (y compris l'identité de \mathbb{N} dans \mathbb{N}).	
Schémas de constructions	
Substitution	
Si, à un argument d'une fonction g à q arguments, $q \in \mathbb{N}^*$, on substitue une fonction f , à p arguments, $p \in \mathbb{N}$, on obtient une nouvelle fonction à $q + p - 1$ arguments. C'est la composition ordinaire dans le cas où g a un seul argument, on la note alors $g \circ f$.	
Somme pour $0 \leq k < n$ et produit pour $0 \leq k < n$	
Si u est une fonction définie sur $\mathbb{N} \times \mathbb{N}^p$, où $p \in \mathbb{N}$, $\sum_{k < n} u(k, x)$ et $\prod_{k < n} u(k, x)$ sont des fonctions de n et de x définies sur $\mathbb{N} \times \mathbb{N}^p$ et vérifiant :	
$\sum_{k < 0} u(k, x) = 0$	$\prod_{k < 0} u(k, x) = 1$
$\sum_{k < n+1} u(k, x) = \sum_{k < n} u(k, x) + u(n, x)$	$\prod_{k < n+1} u(k, x) = \prod_{k < n} u(k, x) \times u(n, x)$

2.2. Algorithmes et applications élémentaires, c'est-à-dire à valeur dans \mathbb{N}^q , $q \in \mathbb{N}^*$, (au lieu de \mathbb{N}).

On reprend le tableau ci-dessus avec le schéma de construction supplémentaire *application à valeur dans un produit*. De plus, on peut remplacer le schéma de substitution par le schéma de *composition*.

Application à valeur dans un produit

Si f est une application de \mathbb{N}^p dans \mathbb{N}^q , si g est une application de \mathbb{N}^q dans \mathbb{N}^r , alors $x \mapsto (f(x), g(x))$ est une application de \mathbb{N}^p dans \mathbb{N}^{q+r} .

Composition (remplace le schéma de substitution)

Si f est une application de \mathbb{N}^p dans \mathbb{N}^q et g une application de \mathbb{N}^q dans \mathbb{N}^r , alors $g \circ f$ est l'application de \mathbb{N}^p dans \mathbb{N}^r définie par : $x \mapsto g(f(x))$.

Les schémas de constructions sont compatibles avec l'égalité des fonctions ou applications (I.3.2).

2.3. Distinction entre algorithme et fonction ou application

Une *fonction* ou une *application* se distingue d'un *algorithme* par l'égalité. Deux algorithmes égaux sont engendrés de la même façon selon le procédé de génération inductive ci-dessus. Deux algorithmes distincts peuvent donner naissance à la même application selon l'égalité définie en fin du paragraphe I.

Exemple : fonction signum

On utilise ici et dans tous les paragraphes suivants le symbole \equiv comme égalité de définition.

On propose une première définition notée sg de signum par composition de la soustraction positive :

$$sg(x) \equiv 1 \dot{-} (1 \dot{-} x)$$

On en propose une autre notée $sign$ à l'aide de soustraction positive, fonction nulle et Π :

$$zéro(k) \equiv 0$$

$$sign(x) \equiv 1 \dot{-} \Pi_{k < x} zéro(k)$$

sg et $sign$ sont des algorithmes différents mais donnent naissance à la même fonction parce que : $sg(0) = 0 = sign(0)$ et $\forall x \in \mathbb{N}^* \quad sg(x) = 1 = sign(x)$.

3. Théorème 1

Tout tronc de suite admet un prolongement en une fonction élémentaire.

Démonstration.

Soit $\{c_k\}$, $0 \leq k < l$, avec $k \in \mathbb{N}$, $l \in \mathbb{N}$, $c_k \in \mathbb{N}$, un tronc de suite de longueur l . La suite u , ci-dessous définie, fournit une solution.

$$u(n) \equiv \sum_{k < l} c_k \times ((n+1) \dot{-} k) \times ((k+1) \dot{-} n)$$

Remarque.

Le résultat ne dépend pas de l , le symbole Σ n'est utilisé ici qu'à titre de commodité, il ne s'agit en fait, pour $3 \leq l$, que de l'utilisation du schéma de substitution répété $(l-2)$ fois.

4. Minimum et maximum**4.1. min, max à deux arguments**

$$x \in \mathbb{N}, y \in \mathbb{N} \quad \min(x, y) \equiv x \dot{-} (x \dot{-} y)$$

$$x \in \mathbb{N}, y \in \mathbb{N} \quad \max(x, y) \equiv x + (y \dot{-} x)$$

4.2. Min, Max agissant sur une suite

u étant une fonction définie sur $\mathbb{N} \times \mathbb{N}^p$ avec $p \in \mathbb{N}$.

$$k \in \mathbb{N}, n \in \mathbb{N}, x \in \mathbb{N}^p \quad \text{Min}_{k \leq n} (u(k, x)) \equiv u(0, x) \dot{-} \sum_{k < n} (u(k, x) \dot{-} u(k+1, x))$$

$$k \in \mathbb{N}, n \in \mathbb{N}, x \in \mathbb{N}^p \quad \text{Max}_{k \leq n} (u(k, x)) \equiv u(0, x) + \sum_{k < n} (u(k+1, x) \dot{-} u(k, x))$$

4.3. min, max agissant sur un multiplé d'entiers

$p \in \mathbb{N}^*$

$$x \in \mathbb{N}^p \quad \min(x) \equiv \text{Min}_{k \leq p-1} (\text{pr}_{k+1}(x))$$

$$x \in \mathbb{N}^p \quad \max(x) \equiv \text{Max}_{k \leq p-1} (\text{pr}_{k+1}(x))$$

$\text{pr}_1, \dots, \text{pr}_p$ désignent les projections de \mathbb{N}^p sur chacune de ses coordonnées numérotées de 1 à p .

Même remarque qu'à l'issue du théorème 1 du paragraphe 3 ci-dessus.

5. Puissance exposant

$$a \in \mathbb{N}, x \in \mathbb{N} \quad a^x \equiv \Pi_{0 < x} a$$

On notera aussi :

$$\exp_a(x) \equiv a^x$$

Cas particulier avec $a := 2 \equiv 1 + 1$ où $:=$ est le symbole d'affectation.

$$\exp_2(x) \equiv 2^x$$

6. “Plus petit entier majoré tel que” ou minimisation bornée

6.1. Minimisation bornée

Étant donné A fonction [élémentaire] définie, pour un certain $p \in \mathbb{N}$ sur $\mathbb{N}^p \times \mathbb{N}$, le *plus petit entier y , majoré par z* s'il existe, tel que : $A(x, y) = 0$, est donné par :

$$\gamma(x, z) \equiv \Sigma_{j < z+1} \Pi_{k < j+1} \text{sg}(A(x, k))$$

$\gamma(x, z)$ a un sens et vaut $z + 1$ même si, pour un certain x , la condition $A(x, y) = 0$ n'est vérifiée pour aucun $y \leq z$, dans ce dernier cas, l'interprétation « le plus petit entier tel que... » disparaît.

6.2. Quotient de la division euclidienne

Le quotient de la division euclidienne de a par b en est un cas particulier avec

$$A((a, b), q) := (a + 1) \dot{-} b(q + 1) \text{ et } z := a$$

soit :

$$\text{quotél}(a, b) \equiv \Sigma_{j < a+1} \Pi_{q < j+1} \text{sg}((a + 1) \dot{-} b(q + 1))$$

On vérifie que : $\text{quotél}(a, 0) = a + 1$

On définit aussi :

$$\text{restél}(a, b) \equiv a \dot{-} b \times q$$

restél est toujours défini, mais l'inégalité de la division euclidienne :

$$\text{restél}(a, b) < b$$

n'est vérifiée que pour $0 < b$.

6.3. Racine carrée entière entière approchée par défaut

$$\text{racdéfél}(a) \equiv \Sigma_{j < a+1} \Pi_{q < j+1} \text{sg}((a + 1) \dot{-} (q + 1)^2)$$

7. Calcul propositionnel “élémentaire”

7.1. Calcul propositionnel

L'axiomatique de \mathbb{N} suppose aussi inégalité large et égalité, cette dernière se ramène à l'égalité à 0 par la méthode de notre berger. Si on accepte que “ $x = 0$ ” est à valeur dans $\{\text{Faux}, \text{Vrai}\}$, il vérifie par là même le tiers exclu. Le calcul élémentaire ramène aussi l'inégalité large et l'égalité entre nombres entiers naturels à l'égalité à 0 (II.1).

En remarquant que :

$$x = 0 \wedge y = 0 \Leftrightarrow x + y = 0$$

$$x = 0 \vee y = 0 \Leftrightarrow x \times y = 0$$

$$x = 0 \Rightarrow y = 0 \Leftrightarrow (1 \dot{-} x) \times y = 0$$

le calcul propositionnel restreint aux égalités et inégalités entre fonctions élémentaires s'exprime, sans l'aide des connecteurs \wedge , \vee , \Rightarrow , par simple référence à l'égalité à 0. Le tiers exclu s'en trouve ainsi vérifié.

7.2. Quantificateurs finis (le second membre peut servir de définition)

$$\forall x \in \{0, \dots, n-1\} (\alpha(x) = 0) \Leftrightarrow \Sigma_{x < n} \alpha(x) = 0$$

$$\exists x \in \{0, \dots, n-1\} (\alpha(x) = 0) \Leftrightarrow \Pi_{x < n} \alpha(x) = 0$$

Les quantificateurs finis font ainsi partie du calcul élémentaire, le tiers exclu reste vérifié tant qu'on n'utilise que des quantificateurs finis.

8. Représentation des entiers naturels en base dix ou autre base

Il s'agit de relier entre eux, d'une part les mots écrits dans l'alphabet fini $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ dont les éléments sont appelés lettres ou plus précisément ici *chiffres* et d'autres part les nombres entiers naturels obéissant aux axiomes ci-dessus (cf. 1 et 2).

8.1. Définitions

0 et 1 sont les éléments de \mathbb{N} , issus des axiomes ci-dessus ; $2 \equiv 1 + 1$, $3 \equiv 2 + 1$, $4 \equiv 3 + 1$, $5 \equiv 4 + 1$, $6 \equiv 5 + 1$, $7 \equiv 6 + 1$, $8 \equiv 7 + 1$, $9 \equiv 8 + 1$, $10 \equiv 9 + 1$.

8.2. Application mot \mapsto nombre entier naturel notée \mathcal{N}

Axiome 1. À tout mot m correspond sa longueur, nombre entier naturel noté $\text{length}(m)$ et pour chaque $k \in \mathbb{N}$, $0 \leq k < \text{length}(m)$, on sait faire correspondre : $c_k(m)$ lettre de rang k à partir de la droite de m .

Ainsi au mot m correspond le nombre entier naturel :

$$\mathcal{N}(m) \equiv \sum_{k < \text{length}(m)} c_k(m) \times 10^k$$

8.3. Application nombre entier naturel \mapsto mot notée \mathcal{W}

On associe, à tout entier naturel n , dans un premier temps, son “nombre de chiffres” $\text{nch}(n)$: le plus petit entier l majoré par n tel que $n < 10^l$ et, dans un deuxième temps, la suite de chiffres indexée par k : $\text{restél}(\text{quotél}(n, 10^k), 10)$, la restriction de cette suite à $0 \leq k < \text{nch}(n)$ “lue de droite à gauche” engendre le mot $\mathcal{W}(n)$ de longueur $\text{nch}(n)$.

Axiome 2. Avec les notations précédentes, pour $0 \leq k < \text{nch}(n)$:

$$c_k(\mathcal{W}(n)) = \text{restél}(\text{quotél}(n, 10^k), 10).$$

Ainsi, au mot $\mathcal{W}(n)$ correspond $\mathcal{N}(\mathcal{W}(n)) = n$

Exemple : 021, $\text{length}(021) = 3$, $c_0(021) = 1$, $c_1(021) = 2$, $c_2(021) = 0$

$$\mathcal{N}(021) \equiv 1 + 2 \times 10 + 0 \times 10^2$$

Remarque aussi que le nombre de chiffres de $\mathcal{N}(m)$ est inférieur ou égal à $\text{length}(m)$ et peut être strictement inférieur, exemple donné ci-dessus de 021.

$\mathcal{N} \circ \mathcal{W}$ est l'identité sur \mathbb{N} traduit que \mathcal{N} est surjective (non injective) et que \mathcal{W} est injective (non surjective). On a aussi une bijection entre \mathbb{N} et l'image par \mathcal{W} de \mathbb{N} . Dès lors qu'on a décidé de représenter les entiers en base 10, on confond n , élément de \mathbb{N}^* avec : $\mathcal{W}(n)$. On prendra garde que $\mathcal{W}(0)$ est le mot vide, et qu'il est d'usage de représenter le nombre 0 par le chiffre 0 [et non le mot vide], et on a bien quand même $\mathcal{N}(0) = 0$. Ainsi la représentation a l'avantage d'être lisible, dans tous les cas, pour les sens de “l'homo-sapiens”.

Le lecteur pourra généraliser la représentation dans n'importe quelle base b : nombre entier naturel supérieur ou égal à 2.

9. Conclusion du paragraphe II

Comme l'a remarqué, à juste titre au milieu du XXe siècle, le mathématicien Kalmar, les fonctions élémentaires suffisent à engendrer toutes les fonctions usuelles de l'arithmétique, du calcul propositionnel “élémentaire” (ci-dessus en 7) et même de l'analyse. Le théorème 4 en III.6 et la deuxième partie “Le continu algorithmique” devraient nous renforcer dans cette conviction.

III Algorithmes, fonctions et applications primitives récursives ou 1-récursives

1. Nombres entiers naturels

On suppose connu les nombres entiers naturels à partir d'une axiomatique pauvre, à savoir : les axiomes de Péano.

- $0 \in \mathbb{N}$
- sc est une application de \mathbb{N} dans \mathbb{N}
- 0 n'est pas un successeur
- Pour $x \in \mathbb{N}$, $y \in \mathbb{N}$, $x = y \Leftrightarrow sc(x) = sc(y)$
- Axiome du *raisonnement par récurrence* [primitif récursif].

Soit $\mathcal{P}(n)$ une proposition [égalité entre 0 et une fonction primitive récursive (ci-dessous)] ou même un prédicat [i.e. avec en plus des quantificateurs] avec la variable libre n , $n \in \mathbb{N}$: si $\mathcal{P}(0)$ et $\forall n (\mathcal{P}(n) \Rightarrow \mathcal{P}(sc(n)))$ alors $\forall n \mathcal{P}(n)$.

On trouve la définition des fonctions *primitives récursives* dans la littérature classique [11] ou [12], elle est reprise ci-dessous sous forme de génération inductive dans un environnement logique intuitionniste.

2. Axiomes de génération inductive des *algorithmes* et *fonctions primitifs récursifs*

Fonctions de base
Fonction <i>nulle</i> [i.e. prenant la seule valeur 0] de \mathbb{N} dans \mathbb{N} ou de \mathbb{N}^0 dans \mathbb{N} Fonction <i>successeur</i> de \mathbb{N} dans \mathbb{N} notée <i>sc</i> <i>Projections</i> de \mathbb{N}^p sur chacune de ses coordonnées
Schémas de constructions
Substitution Si, à un argument d'une fonction f à p arguments, $p \in \mathbb{N}^*$, on substitue une fonction g , à q arguments, $q \in \mathbb{N}$, on obtient une nouvelle fonction à $p + q - 1$ arguments. C'est la composition ordinaire dans le cas où f a un seul argument, on la note alors $g \circ f$.
Primitive récursivité Si f est une application de $\mathbb{N}^p \times \mathbb{N} \times \mathbb{N}$ dans \mathbb{N} et $y \in \mathbb{N}$, alors les équations : $g(x, y, 0) = y$ $g(x, y, \text{sc}(n)) = f(x, n, g(x, y, n))$ définissent une unique application g de $\mathbb{N}^p \times \mathbb{N} \times \mathbb{N}$ dans \mathbb{N} .

Exemple : La fonction *prédécesseur* notée *pd* est définie par le schéma de primitive récursivité :

$$\text{pd}(0) = 0$$

$$\text{pd}(\text{sc}(n)) = n$$

3. Axiomes de génération inductive des *applications primitives récursives*

Applications de base
Fonction <i>nulle</i> [i.e. prenant la seule valeur 0] de \mathbb{N} dans \mathbb{N} ou de \mathbb{N}^0 dans \mathbb{N} Fonction <i>successeur</i> de \mathbb{N} dans \mathbb{N} notée <i>sc</i> <i>Projections</i> définies sur un produit d'ensembles tels $\mathbb{N}^p \times \mathbb{N}^q$ sur chacune de leur composante \mathbb{N}^p ou \mathbb{N}^q
Schémas de constructions
Composition Si f est une application de \mathbb{N}^p dans \mathbb{N}^q et g une application de \mathbb{N}^q dans \mathbb{N}^r , alors $g \circ f$ est l'application de \mathbb{N}^p dans \mathbb{N}^r définie par : $x \mapsto g(f(x))$.
Application à valeur dans un produit Si f est une application de \mathbb{N}^p dans \mathbb{N}^q , et g une application de \mathbb{N}^p dans \mathbb{N}^r , alors (f, g) est l'application de \mathbb{N}^p dans \mathbb{N}^{q+r} définie par $x \mapsto (f(x), g(x))$.
Itération pure (cas particulier de primitive récursivité) Si f est une application de \mathbb{N}^p dans \mathbb{N}^p , la composition de f , n fois par elle-même, engendre une application de $\mathbb{N} \times \mathbb{N}^p$ dans \mathbb{N}^p : $(n, x) \mapsto f^{[n]}(x)$ vérifiant les équations : $f^{[0]}(x) = x$ $f^{[n+1]}(x) = f^{[n]}(f(x))$

Remarques

$$0 = \text{pd}^{[n]}(n)$$

On peut ainsi remplacer, dans les fonctions de base, l'application nulle de \mathbb{N} dans \mathbb{N} par l'application *prédécesseur*. C'est de cette façon que le berger de notre introduction vide son sac !

On distingue *algorithme* d'une part et *fonction* ou *application* d'autre part comme au paragraphe précédent (II.2.3).

4. Théorème 2.

Toute fonction élémentaire est primitive réursive.

Démonstration. Il suffit de voir, d'une part, que addition, soustraction positive et multiplication sont engendrées par primitive récursivité à partir de 0 et sc et d'autre part que $\Sigma_{k < n}$ et $\Pi_{k < n}$ sont des cas particuliers de primitive récursivité.

Il est facile de trouver des algorithmes primitifs récursifs, non élémentaires, mais ce n'est pas pour autant qu'ils donneront naissance à des fonctions non élémentaires.

Exemple du **quotient de la division euclidienne** défini par les équations :

$$\text{quot1réc}(0, b) = 0$$

$$\text{quot1réc}(\text{sc}(a), b) = \text{quot1réc}(a, b) + \text{sg}((\text{sc}(a) + 1) \dot{-} b \times (\text{quot1réc}(a, b) + 1))$$

On obtient ainsi :

$$\text{quot1réc}(a, 0) = a$$

et dans tous les cas :

$$\text{quot1réc}(a, b) = \text{quotél}(a, b) \dot{-} (1 \dot{-} b)$$

qui est donc bien une fonction élémentaire.

Il est relativement difficile d'engendrer des fonctions primitives récursives, non élémentaires à cause du théorème 4, il en existe néanmoins à l'aide du théorème 5.

5. Théorème 3

Une application primitive réursive a ses coordonnées qui sont des fonctions primitives récursives et réciproquement.

Démonstration. Dans le sens direct, on utilise une bijection élémentaire de \mathbb{N}^2 sur \mathbb{N} , voir [5] exercice 2.2 page 24. Pour la réciproque, il y a lieu de remarquer que la primitive récursivité s'exprime en termes d'itération pure avec plusieurs coordonnées ; on utilise aussi les projections coordonnées, et le schéma "application à valeur dans un produit".

6. Théorème 4

Si un schéma de primitive récursivité est effectué à l'aide de fonctions élémentaires et si la fonction engendrée est majorée par une fonction élémentaire, alors elle est élémentaire.

Démonstration en [11] ou [12].

7. Exemple de fonction primitive réursive non élémentaire

L'exemple s'appuie sur le théorème qui suit.

7.1. Théorème 5

Pour chaque fonction élémentaire, f à p arguments avec $p \geq 1$, il existe un nombre entier naturel k tel que pour tout $x \in \mathbb{N}^p : f(x) \leq \exp_2^{[k]}(\max(x))$.

On renvoie à II.4.2 pour la définition de max.

La démonstration du théorème 5 est intéressante en ce sens qu'elle fournit un exemple de démonstration par induction (on en trouve des variantes dans [11] ou [12]).

7.2 Initialisation

La propriété est vérifiée pour les fonctions élémentaires de base.

- Fonctions nulle et 1 à un argument x :

$$0 \leq \exp_2^{[0]}(x) = x$$

$$1 \leq \exp_2^{[1]}(x) = 2^x$$

- Addition et multiplication

$$x + y \leq 2 \times \max(x, y) \leq 2^{\max(x, y)} = \exp_2^{[1]}(\max(x, y))$$

$$x \times y \leq 2^x \times 2^y \leq 2^{x+y} = \exp_2(x + y) \leq \exp_2(\exp_2(\max(x, y))) = \exp_2^{[2]}(\max(x, y))$$

- $q^{\text{ème}}$ projection pr_q de \mathbb{N}^p sur \mathbb{N}

$$\text{pr}_q(x) \leq \max(x) = \exp_2^{[0]}(\max(x))$$

7.3. Transfert de la propriété par les schémas de construction

- Schéma de substitution

Avec les notations de II.2.1, il existe des entiers k et l tels que :

$$f(x) \leq \exp_2^{[k]}(x)$$

$$g(y) \leq \exp_2^{[l]}(y)$$

La nouvelle fonction h de z est alors majorée :

$$h(z) \leq \exp_2^{[k+l]}(z)$$

Dans le cas de composition, z et x se confondent, $h = g \circ f$ et la démonstration est plus évidente.

- Sommation Σ

Supposons, toujours avec les notations de II. 2. 1, qu'il existe un entier q tel que :

$$u(k, x) \leq \exp_2^{[q]}(\max(k, x))$$

Remarque. $\max(k, x)$ est le max du paragraphe II.4.3 appliqué au couple (k, x) élément de \mathbb{N}^{p+1} , pour $p = 1$, c'est aussi le max de II.4.1.

$$\sum_{k < n} u(k, x) \leq n \times \exp_2(\max(n, x)) \leq \exp_2^{[2]}(\max(n, \exp_2^{[q]}(\max(n, x)))) \leq \exp_2^{[2+q]}(\max(n, x))$$

- Produit Π

$$\prod_{k < n} u(k, x) \leq \prod_{k < n} \exp_2^{[q]}(\max(n, x)) \leq \exp_2^{[\sum_{k < n} (\exp_2^{[q-1]}(\max(n, x))]} \leq \exp_2^{[2+(q-1)]}(\max(n, x)) \leq \exp_2^{[3+(q-1)]}(\max(n, x))$$

7.4. Corollaire

La fonction $n \mapsto \exp_2^{[n]}(1)$ est primitive réursive, non élémentaire.

Supposons le contraire, il existerait $k \in \mathbb{N}$ tel que :

$$\exp_2^{[n]}(1) \leq \exp_2^{[k]}(n)$$

faisons $n := 2k + 1$, il vient :

$$\exp_2^{[k+k+1]}(1) \leq \exp_2^{[k]}(2k + 1)$$

par contraposée de la stricte croissance de $\exp_2^{[k]}$:

$$\exp_2^{[k+1]} \leq 2k + 1$$

qui est faux pour tout k , on le prouve par récurrence sur k en initialisant la propriété pour $k := 0$, $k := 1$.

IV Machines algorithmiques primitives récursives

On construit inductivement de telles machines en suivant III.3 et la remarque qui suit immédiatement.

Le sac de caillou de notre berger de l'Antiquité fournit un excellent support d'information pour les entiers naturels. Sur ce sac, on écrit un mot à une ou plusieurs lettres x, n, \dots pour l'identifier, ce sac est supposé extensible à la demande, notre berger avait, par là même, inventé la notion de variable entière, il n'était pas loin d'avoir inventé les machines primitives récursives ! Je me suis permis de lui donner un petit coup de pouce ci-dessous.

1. Machines algorithmiques de base

“Mettre un caillou supplémentaire dans un sac” réalise la machine algorithmique qui calcule la fonction successeur sur un **même** support d'information.

“Enlever un caillou d'un sac, s'il y en a, ou ne rien enlever, s'il n'y en a pas” réalise la machine algorithmique qui calcule la fonction prédécesseur sur un **même** support d'information.

“Sélectionner un sac ou un multiplet de sacs, parmi plusieurs, en l'identifiant par le ou les mots qui sont écrits dessus” réalise une machine algorithmique qui calcule la fonction projection.

2. Constructions de nouvelles machines algorithmiques

Si on dispose, d'une part, d'une machine algorithmique qui calcule une application f sur un même support d'information (un sac ou un multiplet de sacs), ce qui signifie qu'elle substitue $f(x)$ à x dans ce support d'information, d'autre part d'un sac n qui sert de compteur, “faire fonctionner la machine f autant de fois qu'il y a de cailloux dans le sac n ” réalise le schéma de construction itération pure.

Lorsque f est la fonction prédécesseur, la machine dont on vient de parler, avec le même sac qui sert de compteur et de support d'information pour f , réalise la fonction nulle. C'est bien de cette façon que notre berger vide son sac !

Lorsque f est la fonction successeur, la machine, dont on vient de parler, initialisée à 0, recopie le contenu de n dans x . C'est encore, de cette façon, que notre berger recopie le contenu de son troupeau dans son sac ! Cette machine algorithmique nous permet de conserver l'information initiale contenue

dans un sac avant de lui appliquer la fonction successeur, prédécesseur ou autre. Avec plusieurs recopiage, on dispose ainsi d'une machine qui réalise la fonction successeur ou prédécesseur, ou toute autre application d'un support d'information x sur un support d'information distinct y , l'information contenue dans x est alors conservée.

Si on dispose d'une machine qui calcule f à partir d'un support d'information x sur un autre support d'information y et d'une machine qui calcule g à partir du support d'information y sur un autre support z , la "mise en série" de ces machines calcule la fonction $g \circ f$ du support x sur le support z .

De plus, si x et z sont des multipléts de p sacs (avec le même p), par recopiage de z sur x , on a réalisé une machine qui calcule $g \circ f$ sur un même support d'information, ceci peut-être utile, en vue de lui appliquer une itération pure.

Enfin si on dispose d'une machine qui calcule f et g , chacune sur deux supports différents, toutes deux définies sur \mathbb{N}^p , en recopiant les arguments de f et de g sur un même support d'information x , et en "mettant les machines en parallèles", on dispose d'une machine qui calcule $x \mapsto (f(x), g(x))$.

3. Conclusion et remarques

3.1. Toutes les applications de base et les schémas de constructions sont réalisés, on est ainsi capable de construire une machine qui calcule une application primitive récursive donnée ou un nombre fini d'applications primitives récursives. On ne sait pas construire, par ce procédé, une machine qui calculerait toutes les applications primitives récursives.

De telles machines sont, bien entendu, susceptibles de calculer les fonctions élémentaires.

3.2. Le fonctionnement d'une machine primitive récursive est automatiquement **fini**, les calculs répétés se faisant toujours selon un compteur, "autant de fois qu'il y a de caillou dans un sac".

Pour calculer une application primitive récursive f donnée, une machine est **finie** dans sa conception, seuls les supports d'informations, d'entrée, de sortie et éventuellement de recopiage, sont supposés extensibles à la demande. Le nombre de ces supports d'informations est fixé, il ne dépend que de f et non du contenu des arguments "sacs placés en entrée".

3.3. Le lecteur pourra remplacer, pour donner une connotation plus actuelle à ces machines, "sac de caillou" par "ruban avec une origine inextensible et une extrémité extensible à la demande". Le ruban initialement vierge, accepte de recevoir un seul type de symbole autre que le blanc noté conventionnellement $|$, ces symboles sont astreints à être écrits de façon **consécutif** à partir de l'origine. Clairement, les symboles $|$ remplacent les cailloux.

3.4. Il est certain qu'on peut diminuer considérablement la longueur des rubans, par exemple en accolant quatre rubans du type précédent, les nouvelles cases sont des multipléts de 4 cases [de ruban simple], ce qui permet d'écrire 16 types de symboles, on conserve un symbole "blanc", par exemple "4 blancs", Il reste 15 symboles dont dix suffisent à écrire les lettres de l'alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Comme précédemment, les symboles, autres que le blanc, sont astreints à être écrits de manière **consécutif** à partir de l'origine et en nombre fini, ce qui permet d'écrire des mots dans ce même alphabet ; les rubans sont extensibles à la demande à leur extrémité autre que l'origine.

3.5. Dans le cas précédent, il est possible de maîtriser, au moins dans le cas élémentaire, la longueur des rubans au sens que le nombre de chiffres [en base dix] nécessaire pour écrire le résultat s'exprime par un algorithme plus simple [i.e. qui vient avant dans les schémas de construction] avec des entiers plus petits.

	Majorant du nombre de chiffres en entrée	Majorant du nombre de chiffres en sortie
Addition	p, p	$p + 1$
Soustraction positive	p, q	p
Multiplication	p, q	$p + q$
$\Sigma_{k < n} u(k, x)$	(nombres de chiffres de $u(x, k) \leq p$)	$p + (\text{nombre de chiffres de } n)$
$\Pi_{k < n} u(k, x)$	(nombres de chiffres de $u(x, k) \leq p$)	$p \times n$
composition	(nombre de chiffres de $f(x) \leq \varphi$ (nombre de chiffres de x) (nombre de chiffres de $g(y) \leq \gamma$ (nombre de chiffres de x))	(nombre de chiffres de $g \circ f(x)$) $\leq \gamma \circ \varphi$ (nombre de chiffres de x)

V Fonctionnelles

1. Axiomes de génération inductive des fonctionnelles élémentaires, primitives récursives

Les fonctionnelles élémentaires, respectivement primitives récursives de base sont les schémas de construction des applications élémentaires. L'une des variables fonctionnelles qui figure dans ces schémas peut être fixée.

Les schémas de construction sont les compositions entre fonctionnelles, pourvu qu'elles aient un sens.

Exemples

Outre les fonctionnelles de base, Min, Max, minimisation bornée, quantificateur existentiel fini, quantificateur universel fini déjà cités précédemment, sont en fait des fonctionnelles élémentaires.

Remarque

De même que toute fonction élémentaire est primitive récursive, toute fonctionnelle élémentaire est primitive récursive.

Égalité entre fonctionnelles

Elle est définie comme en I.3.2 à l'aide d'un quantificateur universel, lequel agit sur une variable fonctionnelle. Les fonctionnelles élémentaires comme les fonctionnelles primitives récursives sont compatibles avec l'égalité des applications (raisonner par induction).

Applications entre espèces d'applications

Les applications entre espèces d'applications sont par définitions les fonctionnelles destinées à agir sur une application de l'espèce de départ. Une autre façon de l'interpréter est de dire qu'une application Φ agissant sur une application α de l'espèce de départ est construite à partir des fonctions de base [élémentaires, respectivement primitives récursives] et de α qui joue le même rôle qu'une fonction de base, à l'aide des schémas de constructions [élémentaires, respectivement primitifs récursifs].

2. Seconde récursivité ou récursivité d'ordre 2

La composition des fonctionnelles répétée n fois, lorsqu'elle a un sens, n'est pas un schéma de construction primitif récursif, elle va nous conduire à la notion de *seconde récursivité*.

2.1. Axiomes de génération inductive des applications et fonctionnelles 2-récurrentes

Applications de base
Les applications primitives récurrentes de base
Fonctionnelles de base
Les fonctionnelles primitives récurrentes de base
Schéma de construction de seconde récurrentivité
<p>Ce schéma engendre, par composition d'une fonctionnelle n fois par elle même, une nouvelle application et une nouvelle fonctionnelle.</p> <p>Soient :</p> <p>α une application de \mathbb{N}^p dans \mathbb{N}^q.</p> <p>Φ une fonctionnelle qui à une application de \mathbb{N}^p dans \mathbb{N}^q fait correspondre une application de \mathbb{N}^p dans \mathbb{N}^q.</p> <p>L'application de $\mathbb{N} \times \mathbb{N}^p$ dans $\mathbb{N}^q : (n, x) \mapsto (\Phi^{n!}(\alpha))(x)$ est une nouvelle application 2-récurrente.</p> <p>La fonctionnelle qui à α fait correspondre : $\{(n, x) \mapsto (\Phi^{n!}(\alpha))(x)\}$ est une nouvelle fonctionnelle 2-récurrente.</p>
Schémas de construction de nouvelles applications
Les trois schémas de constructions primitifs récurrents
Schéma de construction de nouvelles fonctionnelles
La composée de deux fonctionnelles, pourvu qu'elle ait un sens, est une fonctionnelle.

2.2. Remarques

Clairement les applications primitives récurrentes sont 2-récurrentes, une application engendrée par un tel processus 2-récurrent peut très bien être primitive récurrente ou même élémentaire, voir, ci-dessous : exemple des coefficients du binôme. Il existe des applications 2-récurrentes non primitives récurrentes, voir ci-dessous : exemples de la fonction d'Ackermann et d'Ackermann-bis.

3. Exemples

3.1. Coefficients du binôme

Avec les notations ci-dessus, sauf x remplacé par k :

$$\alpha(k) := 1 \text{ } _ k$$

Φ est la fonctionnelle qui à une suite u , application de \mathbb{N} dans \mathbb{N} , fait correspondre la nouvelle suite :

$$k \mapsto u(k) + \text{sg}(k) \times u(k \text{ } _ 1)$$

$$C(n, k) \equiv (\Phi^{n!}(\alpha))(k)$$

$$C(0, 0) = 1, \text{ pour } 1 \leq k : C(0, k) = 0$$

$$C(1, 0) = 1, C(1, 1) = 1, \text{ pour } 2 \leq k : C(1, k) = 0$$

$$C(2, 0) = 1, C(2, 1) = 2, C(2, 2) = 1, \text{ pour } 3 \leq k : C(2, k) = 0 \dots$$

On reconnaît bien, pour $k \leq n$, les coefficients du binôme donnés par le "triangle de Pascal".

Comme chacun sait, on a aussi, après avoir défini la fonction élémentaire factorielle :

$$\text{fact}(n) \equiv \prod_{k < n} (k + 1)$$

$$C(n, k) = \text{quotél}(\text{fact}(n), \text{fact}(k) \times \text{fact}(n \text{ } _ k) \times \text{sg}((n + 1) \text{ } _ k)$$

ce qui prouve que C est élémentaire.

3.2. Fonction d'Ackermann

Avec les notations ci-dessus :

$$\alpha(x) := 1 + x$$

Φ est la fonctionnelle qui à une suite u , application de \mathbb{N} dans \mathbb{N} , fait correspondre la nouvelle suite : $k \mapsto u^{k+1}(1)$.

La fonction d'Ackermann est alors définie par :

$$\text{Ac}(n, x) \equiv (\Phi^{n!}(\alpha))(x)$$

On montre dans [12], d'une manière comparable à ce qui a été fait en III.7, que la fonction d'Ackermann n'est pas primitive récurrente.

3.3. Fonction d'Ackermann-bis

La fonction d'Ackermann-bis qu'on rencontre, dans certains ouvrages, encore sous le nom de fonction d'Ackermann, est une fonction à 3 arguments $(n, x, y) \mapsto \text{Ab}(n, x, y)$ qui calcule :

pour $n := 0$, $1 + y$
 pour $n := 1$, $x + y$
 pour $n := 2$, $x \times y$
 pour $n := 3$, $x^y \dots$

• On interprète donc $(x, y) \mapsto \text{Ab}(n, x, y)$ comme une suite d'applications qu'on initialise [$n := 0$] par :

$$\alpha(x, y) := (0, 1 + y)$$

• $h(k, x)$ est une fonction auxiliaire qui servira à initialiser [$y := 0$] : $y \mapsto \text{Ab}(k + 1, x, y)$, ainsi, h est définie par les équations :

$$\text{initialisation de l'addition : } h(0, x) = x + 0 = x,$$

$$\text{initialisation de la multiplication : } h(1, x) = x \times 0 = 0$$

$$\text{initialisation de puissance exposant et au delà, pour } n \geq 2 : h(n, x) = 1$$

d'où : $h(n, x) \equiv (1 \dot{-} n) \times x + \text{sg}(n \dot{-} 1)$

• Φ est la fonctionnelle qui au couple d'un entier naturel n et d'une application u de \mathbb{N}^2 dans \mathbb{N} , [lequel couple peut s'interpréter comme application, dépendant du paramètre, n de \mathbb{N}^2 dans \mathbb{N}^2 : $(x, y) \mapsto (n, u(x, y))$] fait correspondre le couple $(n + 1, v(x, y))$ [lequel couple peut s'interpréter comme application de \mathbb{N}^2 dans \mathbb{N}^2 : $(x, y) \mapsto (n + 1, v(x, y))$] où v est défini par le schéma de primitive récursivité :

$$v(x, 0) = h(n ; x)$$

$$v(x, y + 1) = u(x, v(x, y))$$

La fonction d'Ackermann-bis est alors définie par :

$$\text{Ab}(n, x, y) \equiv \text{pr}_{2 \circ}(\Phi^{n!}(\alpha))(x, y)$$

Comme la fonction d'Ackermann, la fonction d'Ackermann-bis n'est pas primitive récursive.

4. Machine algorithmique pour la seconde récursivité ?

Je ne connais aucune machine algorithmique, répondant aux mêmes critères que ceux donnés en IV.3.2, susceptible de calculer la fonction d'Ackermann ou d'Ackermann-bis. Toutefois, la restriction de la fonction d'Ackermann, d'Ackermann-bis ou de toute application 2-récursive à un ensemble fini se calcule sur une machine primitive récursive (la restriction pour le premier argument suffit pour Ackermann et Ackermann-bis). Il faudrait une suite [infinie] de machines primitives récursives emboîtées pour les calculer. Ainsi la seconde récursivité engendre une notion d'algorithme dans un sens bien plus faible que la primitive récursivité.

VI Autres récursivités

On rencontre d'autres récursivités dans la littérature classique, en particulier la μ -récursivité, elle est inspirée par la minimisation bornée du II.6, mais on supprime la borne. Il est ainsi proposé un nouveau schéma de construction dit *schéma de minimisation non bornée*.

1 μ -récursivité

1.1. Définition

Étant donné A fonction [élémentaire ou primitive récursive] définie, pour un certain $p \in \mathbb{N}$, sur $\mathbb{N}^p \times \mathbb{N}$, le plus petit entier y , s'il existe, tel que : $A(x, y) = 0$ est une nouvelle fonction de x dite μ -récursive.

Le seul ennui est que pour chaque x , la recherche par une suite de tests pour $y := 0, y := 1 \dots$ peut très bien ne jamais se terminer. Les langages algorithmiques ou plutôt "dits algorithmiques" usuels

acceptent ce type de recherche par le biais d'instructions du type "Si non $A(x, y) = 0$ aller à une instruction antérieure", "If ...go to..." ou "Tant que non $A(x, y) = 0$ faire $y := y + 1$ ", "While ...". Du fait de la terminaison non garantie, j'accepterai dans le meilleur des cas de qualifier ces instructions de programme et je les appellerai dans la suite "*programme μ -récursif*".

Si, de plus, on adjoint une démonstration mathématique qui prouve l'existence pour certains x d'un y tel que $A(x, y) = 0$, alors je veux bien qualifier le **couple** du programme μ -récursif et de la démonstration mathématique d'algorithme.

1.2. Exemple

Le quotient de la division euclidienne de a par b peut-être programmé par :

$q := 0$; Tant que $b(q + 1) \leq a$, $q := q + 1$.

On prouve facilement que, pour $1 \leq b$, le quotient de la division euclidienne existe et est inférieur ou égal à a , mais, ce programme tourne sans fin pour $b := 0$.

2. Inutilité de la μ -récursivité

Qui dit démonstration mathématique dit démonstration s'appuyant sur une logique. En logique classique, on s'enfoncé un peu plus dans le paradoxe signalé en I.1. Si l'on veut garder un aspect tant soit peu "calculable" à ce nouveau type d'algorithme, il faut bien s'appuyer sur la logique intuitionniste. Je supposerais donc qu'on puisse tester par une méthode déjà connue (i.e. n'utilisant pas la μ -récursivité) les x pour lesquels il existe y tel que $A(x, y) = 0$. Ceci signifie qu'on connaît le sous-ensemble de définition de la nouvelle fonction par sa fonction caractéristique (qui vaut 1 sur l'ensemble de définition et 0 ailleurs), cette fonction résultant d'un concept d'algorithme déjà connu est élémentaire, primitive récursive ou même 2-récursive. En multipliant le test par cette fonction, on a un prolongement immédiat par 0 en dehors de son "ancien ensemble de définition". Je poursuis le raisonnement en supposant donc que pour tout x , il existe y tel que $A(x, y) = 0$, je l'exprime par le prédicat avec la variable libre x :

$\exists y A(x, y) = 0$

Comment a pu être introduit ce quantificateur existentiel ?

Il n'y a aucun axiome avec quantificateur existentiel dans les axiomes tels que je les ai présentés en II.1 ou III.1.

Pour introduire ce quantificateur, il a donc bien fallu utiliser \exists -introduction, c'est-à-dire exhiber un certain nombre entier dépendant de x sous la forme $A(x, g(x))$, la seule façon de donner un sens à ce $g(x)$ dans le cadre de la théorie élémentaire, primitive récursive, (éventuellement 2-récursive) est de dire que g est une fonction élémentaire, primitive récursive, (éventuellement 2-récursive) de x ; on n'en connaît pas d'autres ! Ce $g(x)$ étant connu, le schéma de minimisation non borné devient un schéma de minimisation borné avec $g(x)$ comme borne. Ainsi le couple (minimisation non bornée, démonstration intuitionniste (cf. 1.2)), ne permet pas de sortir de l'élémentaire en théorie élémentaire, du primitif récursif en théorie primitive récursive, on n'atteint même pas le 2-récursif par ce procédé ! En tout état de cause, il n'y a pas lieu d'introduire la μ -récursivité.

Il est décevant de constater que les mathématiciens classiques considèrent comme acquis le fait que cette μ -récursivité traduise la "bonne" formalisation de la notion de "fonction calculable", et ce depuis les travaux des logiciens Gödel (1906 1978), Church (1903 1995), qui sont considérés comme les fondateurs. Les arguments à l'appui sont les nombreuses caractérisations équivalentes qui ont été développées, notamment la notion de "fonctions calculables sur machine de Turing", sur "machine à registre", du " λ -calcul". C'est ce qu'on trouve essentiellement développé en [9], [11] et [12] et dans tous les ouvrages classiques traitant de la calculabilité. Les informaticiens s'en sont emparé ; les programmes du secondaire ont suivi, c'est ainsi qu'on trouve au programme actuellement en vigueur, 2ème page, dans la rubrique "mathématiques et informatique", "boucles, test, récursivité", "l'élève devra mettre en œuvre, notamment sur sa calculatrice, les notions de boucle et de test".

3. Récursivité partielle et générale

On parle de *récursivité partielle* pour les fonctions non partout définies et de *récursivité générale* pour les fonctions partout définies. Il est impossible de distinguer récursivité partielle et récursivité

générale sans faire appel au tiers exclu ! Noter au passage que les fonctions élémentaires, primitives récursives, 2-récursives sont automatiquement partout définies.

Certains mathématiciens ont quelques scrupules à accepter la récursivité partielle comme calculable, du fait que, même en logique classique, la fonction caractéristique de l'ensemble de définition n'est qualifiable d'aucune récursivité, mais acceptent la récursivité générale.

Je n'ai jamais vu et n'ai jamais pu obtenir de personne un seul exemple de fonction définie par un programme μ -récursif, partout définie, non primitive récursive. À défaut de fournir un tel exemple, l'argument pour faire accepter la récursivité générale consiste usuellement à accepter comme axiome [caché] qu'il existe une solution unique du système d'équations à l'inconnue fonctionnelle α :

$$\alpha(0, x) = 1 + x$$

$$\alpha(n + 1, 0) = \alpha(n, 1)$$

$$\alpha(n + 1, x + 1) = \alpha(n, \alpha(n + 1, x))$$

dite fonction d'Ackermann. La fonction Ac (que je propose en V.3.2) vérifie bien ces équations mais, à ma connaissance, la fonction d'Ackermann n'est jamais présentée à l'aide de la 2-récursivité nulle part formalisée. Il est ensuite prouvé que cette fonction est calculable sur machine de Turing, partout définie, donc générale récursive par les équivalences précédemment démontrées (issues de ces mêmes ouvrages). En comparant avec ce que j'ai développé en 2 ci-dessus, il est certain que les démonstrations ne sont pas intuitionnistes et ne le seront jamais. Signalons quand même que la démonstration, déjà signalée, du fait que la fonction d'Ackermann n'est pas primitive récursive est acceptable dans le cadre de mon axiomatique réduite.

S.C. Kleene, dans le cadre de sa formalisation de l'intuitionnisme [10], considère que les "fonctions calculables" qui conviennent sont les fonctions générales récursives ! D'autres écoles mathématiques se disant constructives, tout en acceptant la logique classique, ont développé des théories basées sur la générale récursivité, notamment l'école russe de Markov, Sanin [13]. Pour sa part, le constructiviste Bishop [1] refuse tout recours à la récursivité, on commence à comprendre pourquoi !

Deuxième partie

Le continu algorithmique

I Déploiement élémentaire, primitif récursif

1. Rappel de la définition simplifiée de déploiement

On rappelle les définitions d'un déploiement et d'un déploiement finitaire telles qu'elles figurent dans [8].

Un déploiement est défini comme le triplet de trois notions.

- **Suite d'ensembles**, E_n ($n \in \mathbb{N}$) sous-ensembles d'un même ensemble à prendre parmi \mathbb{N} , \mathbb{N}^2 , \mathbb{Z} , \mathbb{Q} ... (les objets de l'ensemble ont une représentation finie).

- **Relation** entre élément de E_n et élément de E_{n+1} : un élément de E_{n+1} en relation avec un élément de E_n est dit *descendant immédiat* de cet élément, un élément E_n en relation avec un élément de E_{n+1} est dit *ascendant immédiat* de cet élément.

Étant donné un élément de E_n , on peut tester si un élément de E_{n+1} est ou n'est pas descendant immédiat de cet élément de E_n .

Pour $n \in \mathbb{N}$, tout élément de E_n a un descendant immédiat dans E_{n+1} .

Pour tout $n \in \mathbb{N}^*$, tout élément de E_n a un ascendant immédiat dans E_{n-1} .

On prolonge les définitions en *descendant* et *ascendant* par réflexivité et transitivité.

- **Suites admissibles**. Suites (u_n) , $u_n \in E_n$ telle que u_{n+1} est un descendant immédiat de u_n . De plus le mot *suite* doit être compris comme *suite de choix*.

Une *suite de choix* (u_n) dans un déploiement est, comme son nom semble l'indiquer, une suite « construite » terme à terme : u_0 est choisi librement dans E_0 , puis u_1 dans E_1 ... On peut penser à une suite de *pile ou face*, de *lancers de dés*, mais aussi de décisions successives d'un être intelligent, un algorithme gérant la suite est également accepté. Une suite de choix est, de par sa définition même,

toujours inachevée. Le premier principe de Brouwer est un axiome destiné à formaliser le concept de suite de choix, c'est ainsi que je l'interprète.

Un déploiement dans lequel tous les ensembles E_n sont finis est dit *finitaire*.

2. Déploiement élémentaire, 1-récurif

On remplace « suite de choix » par « suite gérée par un algorithme » (qui est un cas particulier accepté ci-dessus) élémentaire, respectivement 1-récurif, même 2-récurif. Dans la suite, je ne proposerai que des énoncés élémentaires, respectivement 1-récurif, mais le lecteur pourra les reprendre en termes 2-récurifs.

L'objectif de ce qui suit est de mettre en évidence que la définition de déploiement, éventuellement finitaire, s'insère parfaitement dans le contexte des fonctions élémentaires, 1-récurives dans lequel on prouvera alors le théorème de l'éventail.

2.1. Expression élémentaire, 1-récurive, de la définition de déploiement

Chaque ensemble E_n est supposé sous-ensemble élémentaire, respectivement 1-récurif de \mathbb{N} , \mathbb{N}^2 , \mathbb{Z} , \mathbb{Q} ; on remplace si besoin \mathbb{N}^2 , \mathbb{Z} , \mathbb{Q} par \mathbb{N} , en utilisant une bijection élémentaire de chacun de ces ensembles avec \mathbb{N} , on donne explicitement une telle bijection de \mathbb{N}^2 sur \mathbb{N} à l'exercice 2.2 du chapitre 2 de [5]. On est ainsi ramené à des sous-ensembles finis de \mathbb{N} . La relation "ascendant descendant immédiat" s'exprime alors à l'aide d'une application élémentaire, respectivement 1-récurive de \mathbb{N}^3 dans \mathbb{N} $ad(n, x, y)$ telle que :

$$x \in E_n \wedge y \in E_{n+1} \wedge \ll y \text{ est descendant immédiat de } x \gg \Leftrightarrow ad(n, x, y) = 0.$$

Appelons $\mathcal{A}_e(\mathbb{N})$, $\mathcal{A}_{pr}(\mathbb{N})$, l'espèce des applications élémentaires, respectivement 1-récurives de \mathbb{N} dans \mathbb{N} . Pour dire que les suites admissibles d'un déploiement élémentaire sont dans $\mathcal{A}_e(\mathbb{N})$, on dira aussi qu'il est *sous-espèce* de $\mathcal{A}_e(\mathbb{N})$; pour dire que les suites admissibles d'un déploiement 1-récurif sont dans $\mathcal{A}_{pr}(\mathbb{N})$, on dira aussi qu'il est *sous-espèce* de $\mathcal{A}_{pr}(\mathbb{N})$.

2.2. Expression élémentaire, 1-récurive de la définition de déploiement finitaire

Outre ce qui précède, chaque ensemble E_n est fini, avec au moins un élément, de cardinal $e(n) + 1$, la fonction $e(n)$ est, bien entendu, supposée élémentaire, respectivement 1-récurive. À l'aide de la bijection de définition de E_n fini, on se ramène à : $x \in E_n \Leftrightarrow 0 \leq x \leq e(n)$. « Pour $n \in \mathbb{N}$, tout élément x de E_n a un descendant immédiat dans E_{n+1} » se traduit en termes de quantificateur existentiel fini : $\prod_{y < e(n)+1} ad(n, x, y) = 0$. « Pour $n \in \mathbb{N}$, tout élément y de E_{n+1} a un ascendant immédiat dans E_n » se traduit en termes de quantificateur existentiel fini : $\prod_{x < e(n)+1} ad(n, x, y) = 0$.

Les réductions précédentes s'obtiennent très facilement dans le cadre de la définition du Continu [8] qui sera alors dit élémentaire, respectivement 1-récurif et des intervalles fermés bornés du Continu qui sont alors qualifiés d'élémentaires, respectivement de 1-récurif.

2.3. Sous-espèce élémentaire, 1-récurive

Étant donnée une fonctionnelle élémentaire, respectivement 1-récurive, définie sur $\mathcal{A}_e(\mathbb{N})$, les suites qui annulent cette fonctionnelle décrivent, par définition, une *sous-espèce élémentaire* de $\mathcal{A}_e(\mathbb{N})$, respectivement une *sous-espèce 1-récurive* de $\mathcal{A}_{pr}(\mathbb{N})$. En particulier un déploiement élémentaire dont les suites admissibles (u_n) sont dans $\mathcal{A}_e(\mathbb{N})$ est *sous-espèce* de $\mathcal{A}_e(\mathbb{N})$; un déploiement 1-récurif dont les suites admissibles sont dans $\mathcal{A}_{pr}(\mathbb{N})$ est *sous-espèce* de $\mathcal{A}_{pr}(\mathbb{N})$. Dans les deux cas, la fonctionnelle à annuler est : $u \mapsto \{n \mapsto ad(n, u_n, u_{n+1})\}$, c'est-à-dire : $\forall n \ ad(n, u_n, u_{n+1}) = 0$.

2.4. Relation élémentaire, primitive récurive

Une *relation élémentaire*, respectivement *1-récurive* entre deux suites d'une même espèce est obtenue en annulant une fonctionnelle élémentaire, respectivement 1-récurive agissant sur ces suites, il y a, par là même, un quantificateur universel dans la définition et une telle relation n'a aucune raison de vérifier le tiers exclu. Une telle relation peut être qualifiée d'*ordre*, d'*équivalence* au sens classique usuel.

II Théorème de l'éventail

1. Lemme 1 de prolongement d'une application initialement définie sur un déploiement finitaire

1.1. Énoncé du lemme

Soit \mathcal{D} un déploiement finitaire élémentaire, sous-espèce de $\mathcal{A}_\varepsilon(\mathbb{N})$, respectivement primitif récursif, sous-espèce de $\mathcal{A}_{pr}(\mathbb{N})$. Toute application de \mathcal{D} dans $\mathcal{A}_\varepsilon(\mathbb{N})$, respectivement $\mathcal{A}_{pr}(\mathbb{N})$, admet un prolongement à $\mathcal{A}_\varepsilon(\mathbb{N})$, respectivement à $\mathcal{A}_{pr}(\mathbb{N})$.

Plan de la démonstration. On construit, une fonctionnelle élémentaire respectivement primitive récursive Ψ qui est une surjection de $\mathcal{A}_\varepsilon(\mathbb{N})$, respectivement $\mathcal{A}_{pr}(\mathbb{N})$ dans \mathcal{D} . La restriction de cette surjection à \mathcal{D} est l'identité. Étant donné une application de \mathcal{D} dans $\mathcal{A}_\varepsilon(\mathbb{N})$, respectivement $\mathcal{A}_{pr}(\mathbb{N})$, la composition à droite par Ψ fournit le prolongement cherché.

Démonstration dans le cas primitif récursif. \mathcal{D} est défini comme précédemment par la suite majorante (e_n) et la relation ascendant descendant $ad(n, x, y) = 0$. Partons d'une suite (u_n) de $\mathcal{A}_{pr}(\mathbb{N})$, on lui fait correspondre dans un premier temps, la suite v_n définie par $\min(u_n, e_n)$, dans un deuxième temps la suite w_n définie ainsi :

$$w_0 = v_0$$

w_{n+1} est un entier, le plus proche possible, de v_{n+1} de telle sorte que $ad(n, w_n, w_{n+1}) = 0$

Plus précisément, soit k_n le plus petit entier k majoré par e_n tel que :

$$v_{n+1} + k_n \text{ ou } v_{n+1} - k_n \text{ est descendant immédiat de } w_n$$

(on a effectué une minimisation bornée avec existence garantie par définition de déploiement).

Si $v_{n+1} - k_n$ est descendant immédiat de w_n , alors : $w_{n+1} = v_{n+1} - k_n$, sinon, c'est $v_{n+1} + k_n$ qui est descendant immédiat de w_n , et $w_{n+1} = v_{n+1} + k_n$.

Les équations : $w_0 = v_0$ et la fonction élémentaire qui fait passer de w_n à w_{n+1} définissent une suite primitive récursive admissible.

Clairement, si v_{n+1} est déjà descendant immédiat de w_n , il est conservé. Ainsi, si la suite (u_n) est admissible, la fonctionnelle primitive récursive qui fait passer de (u_n) à (w_n) conserve (u_n) , c'est cette fonctionnelle qu'on appelle Ψ .

Démonstration dans le cas élémentaire. Dans le cas d'un déploiement élémentaire \mathcal{D} , on reprend, mot pour mot la démonstration précédente, on remarque que tous les calculs sont élémentaires, de plus, le théorème 3 du paragraphe II.5 permet de conclure que (w_n) est élémentaire parce que la suite admissible (w_n) est majorée par la suite élémentaire (e_n) . La fonctionnelle qui fait passer de (u_n) à (w_n) est ainsi élémentaire.

1.2. Corollaire

Toute application de \mathcal{D} dans $\mathcal{A}_\varepsilon(\mathbb{N})$, $\mathcal{A}_{pr}(\mathbb{N})$ est la restriction d'une application de $\mathcal{A}_\varepsilon(\mathbb{N})$, respectivement $\mathcal{A}_{pr}(\mathbb{N})$ dans $\mathcal{A}_\varepsilon(\mathbb{N})$, respectivement $\mathcal{A}_{pr}(\mathbb{N})$, autrement dit, il s'agit d'une fonctionnelle (cf. 1^{ère} partie V. 1).

2. Ordre de \mathbb{N}^p

2.1. Définition de l'ordre dans \mathbb{N}^p

On prolonge l'inégalité de \mathbb{N} à \mathbb{N}^p , $\mathbb{N}^q \dots$ coordonnée à coordonnée, on conserve une relation d'ordre, on ne perd que l'ordre total.

2.2. min, max, Min, Max dans \mathbb{N}^p

On prolonge, de même, min, max, Min, Max coordonnée à coordonnée, ainsi : $\max((x, y), (x', y')) \equiv (\max(x, x'), \max(y, y'))$

$$\text{Pour } k \in \mathbb{N} : \text{Max}_{k \leq n}(u(k, x), v(k, x)) \equiv (\text{Max}_{k \leq n}(u(k, x)), \text{Max}_{k \leq n}(v(k, x)))$$

2.3 Autres lemmes

Tous les lemmes qui suivent ont une démonstration immédiate.

Lemme 2. Soit φ application [élémentaire respectivement primitive récursive] de \mathbb{N} dans \mathbb{N} , alors, $\text{Max}_{k \leq n} \varphi(k)$ est croissante, [élémentaire, respectivement primitive récursive] et majore φ .

Remarque. Le résultat s'applique aussi bien pour une application de \mathbb{N}^p dans \mathbb{N}^q , mais ce n'est pas vraiment indispensable pour la suite.

Lemme 3. Le max, pourvu qu'il ait un sens, de deux applications [élémentaires, respectivement primitives récursives] croissantes est une application [élémentaire, respectivement primitive récursive] croissante.

Lemme 4. La composée, pourvu qu'elle ait un sens, de deux applications [élémentaires respectivement primitives récursives] croissantes est une application [élémentaire, respectivement primitive récursive] croissante.

3. Théorème algorithmique de l'éventail

3.1. Énoncé du théorème

Étant donné une application Φ [fonctionnelle élémentaire, respectivement primitive récursive] d'un déploiement finitaire dans un déploiement, il existe deux applications [élémentaires, respectivement primitives récursives] de \mathbb{N} dans \mathbb{N} , F et φ telles que pour toute suite amissible α , et tout entier naturel n , $(\Phi(\alpha))(n) \leq F(n)$ et $(\Phi(\alpha))(n)$ ne dépend que du tronc de suite de longueur $\varphi(n) : (\alpha(k))_{k < \varphi(n)}$.

Dans tout ce qui suit, l'application F sera appelée *majorante*, l'application $\varphi(x)$ ou par abus de langage *nombre de Brouwer*.

3.2. Transformation de l'énoncé

Compte tenu du lemme 1 de prolongement et du lemme 2, on propose de démontrer par induction l'énoncé plus fort ci-dessous avec des applications F et φ croissantes.

Étant donné un déploiement finitaire \mathcal{D} , [élémentaire, respectivement primitif récursif], dont les suites admissibles α sont définies par la seule majoration $\alpha(n) \leq e(n)$ et une fonctionnelle Φ [élémentaire, respectivement primitive récursive] à valeur dans $\mathcal{A}_\varepsilon(\mathbb{N}^p, \mathbb{N}^q)$, respectivement $\mathcal{A}_{pr}(\mathbb{N}^p, \mathbb{N}^q)$, il existe deux applications croissantes [élémentaires, respectivement primitives récursives] : F de \mathbb{N}^p dans \mathbb{N}^q [dite encore *majorante*] et φ de \mathbb{N}^p dans \mathbb{N} telles que pour toute suite α de \mathcal{D} et tout élément x de \mathbb{N}^p , $(\Phi(\alpha))(x) \leq F(x)$ et $(\Phi(\alpha))(x)$ ne dépend que du tronc de suite de longueur $\varphi(x) : (\alpha(k))_{k < \varphi(x)}$.

Comme on l'a déjà dit en 1.2, Φ est construite à partir des fonctions de base et de α considérée comme fonction de base.

Comme au théorème 5 de la première partie III.7, la démonstration construit inductivement les applications F et φ [en composant des schémas de constructions élémentaires respectivement primitifs récursifs] à l'aide de la suite majorante $e(n)$ définissant le déploiement finitaire \mathcal{D} .

3.3. Initialisation

Φ est l'identité agissant :

- soit sur une fonction de base autre que α , φ est nulle et la fonction majorante F est la fonction, de base elle-même éventuellement majorée par une fonction croissante (à l'exception de la soustraction positive, toutes les fonctions de bases sont croissantes ; $x \div y$ est majoré par x)

- soit sur α , φ est alors la fonction successeur et la fonction majorante est obtenue en majorant la fonction e définissant le déploiement finitaire \mathcal{D} par une fonction croissante selon le lemme 2.

3.4. Transfert de la propriété par les schémas de construction communs à élémentaire et primitif récursif.

Hypothèse d'induction. On suppose qu'il existe d'une part F et φ croissantes, indépendantes de α telles que f soit majorée par F et que $f(x)$ ne dépende que du tronc de suite $\alpha(k)_{k < \varphi(x)}$ d'autre part G et γ croissantes, indépendantes de α telles que g soit majorée par G et que $g(x)$ ne dépende que du tronc de suite $\alpha(k)_{k < \varphi(x)}$

- Application à valeur dans un produit : $x \mapsto (f(x), g(x))$

Majorante. $(F(x), G(x))$ convient et est croissante d'après la définition de l'ordre.

Nombre de Brouwer. $\max(\varphi(x), \gamma(x))$ convient et est encore croissante (lemme 3)

- Composition d'applications : $x \mapsto g \circ f(x)$

Majorante. $G \circ F(x)$ convient comme majorante et est croissante (lemme 4).

Nombre de Brouwer. $\max(\varphi(x), \gamma \circ F(x))$ convient comme nombre de Brouwer et est encore croissante (lemmes 3 et 4).

En effet, par hypothèse d'induction, $f(x)$ ne dépend que des $\varphi(x)$ premiers termes de α . Supposons maintenant $f(x)$ connue, toujours par hypothèse d'induction, $g(f(x))$ ne dépend que des $\gamma(f(x))$ premiers termes de α , comme γ est croissante et f majorée par F , $g(f(x))$ ne dépend a fortiori que des $\gamma(F(x))$ premiers termes de α . En conclusion, $\max(\varphi(x), \gamma \circ F(x))$ convient comme nombre de Brouwer.

3.5. Transfert de la propriété par le schéma d'itération.

Hypothèse d'induction. On suppose qu'il existe F et φ croissantes, indépendantes de α telles que f soit majorée par F et que $f(x)$ ne dépende que du tronc de suite $\alpha(k)_{k < \varphi(x)}$. On commence par remplacer F par $\max(x, F(x))$ encore croissante conformément au lemme 3, ainsi : $x \leq F(x), f(x) \leq F(x)$.

Majorante. En utilisant 3.4, $F^{[n]}(x)$ fournit une application majorante de $f^{[n]}(x)$, c'est une application croissante de (n, x) parce que F est croissante et $x \leq F(x)$.

Nombre de Brouwer. Pour $f^{[2]}$, conformément à 3.4, $\max(\varphi(x), \varphi(F(x)))$ convient (cf. 3.4 composition), comme φ est croissante et $x \leq F(x)$, $\varphi(F(x))$ convient aussi. En raisonnant par récurrence sur $n > 0$, pour $f^{[n]}(x)$, $\varphi(F^{[n-1]}(x))$ convient comme nombre de Brouwer. C'est encore vrai pour $n := 0$ en prenant $\text{sg}(x) \times \varphi(F^{[n-1]}(x))$. Cette nouvelle application est croissante du couple (n, x) , comme ci-dessus pour majorante.

3.6. Transfert de la propriété par Σ et Π .

C'est encore plus facile et laissé au lecteur !

4. Corollaires du théorème de l'éventail

Le *Continu algorithmique élémentaire*, respectivement *primitif récursif* est défini comme en [8], mais à partir d'un déploiement élémentaire, respectivement primitif récursif.

4.1 Théorème de continuité

Toute application du Continu algorithmique [élémentaire, respectivement primitif récursif] est uniformément continue sur les intervalles fermés bornés. De plus l'application module de continuité est élémentaire, respectivement primitive récursive.

Revoir la démonstration donnée en [8].

Il est remarquable de constater alors que la continuité est obtenue sans aucun axiome spécifique, ni principes de Brouwer, ni axiome de l'éventail.

4.2 Tronc de suite et réduction au premier ordre

Ce paragraphe a pour objet de préparer à l'énoncé de 4.3.

On note dans ce qui suit $\langle \alpha, l \rangle$ le tronc de suite $(\alpha(k))_{k < l}$

Un tronc de suite de \mathbb{N} peut être représenté par un nombre entier naturel et ce, de bien des manières [11] ou [12], sans même l'hypothèse tronc de suite d'un déploiement finitaire. Dans le cas qui nous intéresse ici, réalisons une bijection élémentaire entre les troncs de suites admissibles α définies par la seule condition de majoration $\alpha(n) \leq e(n)$ et \mathbb{N}^2 ("extension" de la représentation en base dix des entiers naturels du paragraphe II. 8).

$$\langle \alpha, l \rangle \mapsto (\sum_{k < l} \alpha(k) \times \prod_{j < k} (e(j) + 1), l)$$

$$(n, l) \mapsto (\text{restél}(\text{quotél}(n, \prod_{j < k} (e(j) + 1)), e(k) + 1))_{k < l}$$

On peut composer, si on le souhaite, avec une bijection élémentaire de \mathbb{N}^2 sur \mathbb{N} , mais, c'est en fait inutile pour la réduction au premier ordre qui suit.

Ainsi les applications "tronc de suite \mapsto tronc de suite" s'interprètent comme de simples applications entre éléments de \mathbb{N}^2 .

4.3. Théorème de réduction au premier ordre d'une application d'un déploiement finitaire dans un déploiement.

Il faut entendre par *réduction au premier ordre*, le remplacement d'une fonctionnelle élémentaire, respectivement primitive récursive par une application élémentaire, respectivement primitive récursive

qui à un tronc de suite fait correspondre un tronc de suite, c'est à dire, moyennant la bijection précédente, une application élémentaire, respectivement primitive récursive de \mathbb{N}^2 dans \mathbb{N}^2 .

Étant donné une application Φ [fonctionnelle élémentaire, respectivement primitive récursive] d'un déploiement finitaire \mathcal{D} dans un déploiement, il existe une fonction croissante [élémentaire, respectivement primitive récursive] $\tilde{\varphi}$ et une application [élémentaire, respectivement primitive récursive] f qui, au tronc de suite amissible $\langle \alpha, l \rangle$ fait correspondre le tronc de suite : $f(\langle \alpha, l \rangle) \equiv \langle \Phi(\alpha), \tilde{\varphi}(l) \rangle$.

Construction de $\tilde{\varphi}$

On part de la fonctionnelle Φ restreinte à \mathcal{D} , on lui fait correspondre l'application nombre de Brouwer φ qu'on suppose croissante (lemme 3). Il s'agit de faire correspondre à un tronc de suite de longueur l un autre tronc de suite de longueur $\tilde{\varphi}(l)$. Soit t le plus petit entier naturel, s'il existe, inférieur ou égal à l , tel que $\varphi(l \dot{-} t) \leq l$, c'est une minimisation bornée (II, 6). Il y a lieu de remarquer que pour $l < \varphi(0)$, un tel entier n'existe pas et que par l'algorithme de la minimisation bornée, t prend alors la valeur $l + 1$; pour $\varphi(0) \leq l$, un tel entier existe parce que $\varphi(l \dot{-} l) \leq l$. On définit alors, dans tous les cas, la longueur du tronc de suite image : $\tilde{\varphi}(l) \equiv (l + 1) \dot{-} t$.

Remarque

Pour $l < \varphi(0)$, la longueur du tronc de suite image est nulle, il s'agit du tronc de suite vide. Pour $\varphi(0) \leq l$, l'entier $n := l \dot{-} t = l - t$ (parce que $t \leq l$) est le plus grand entier $n \leq l$ tel que $\varphi(n) \leq l$ et $\tilde{\varphi}(l) = n + 1$.

Distinguons deux cas pour le prouver :

- $0 = t$, donc $n = l$, la propriété est alors évidente.
- $0 < t$, donc $n < l$,

par définition de t : $\varphi(l - t) \leq l < \varphi(l - (t - 1))$, donc : $\varphi(n) \leq l < \varphi(n + 1)$.

Ceci prouve aussi la croissance de $\tilde{\varphi}$.

Proposition

La suite $(\tilde{\varphi}(l))$ tend vers l'infini lorsque l tend vers l'infini.

Ainsi, la nouvelle application f engendre bien toute la suite image $\Phi(\alpha)$. C'est même de cette façon que sont construits directement tous les exemples usuels d'applications d'un intervalle fermé borné ou d'un produit d'intervalles fermés bornés du Continu dans le Continu [8].

Démonstration de la proposition

Soit un entier donné, s'il est nul, $\tilde{\varphi}(0) \geq 0$, s'il est strictement positif, il s'écrit $n_1 + 1$, soit alors : $l := \max(n_1, \varphi(n_1))$. Nous sommes dans le cas : $\varphi(0) \leq l$ parce que : $\varphi(0) \leq \varphi(n_1) \leq l$, par définition de $\tilde{\varphi}$ et remarque qui suit : $\tilde{\varphi}(l) = n + 1$ avec n le plus grand entier tel que $n \leq l$ et $\varphi(n) \leq l$. Comme par définition de l : $\varphi(n_1) \leq l$ et $n_1 \leq l$, donc $n_1 \leq n$ et $\tilde{\varphi}(l) \geq n_1 + 1$ c.q.f.d.

5. Analyse élémentaire

Certains procédés de l'analyse classique ne sont pas acceptables en analyse intuitionniste, notamment : borne supérieure d'une partie non vide majorée de \mathbb{R} , suite croissante majorée convergente, extraction d'une suite convergente d'une suite bornée, théorème des valeurs intermédiaires, voir [1] et [8]. Au contraire les procédés suivants, non seulement, sont acceptables en analyse intuitionniste, mais ne sortent pas du continu élémentaire.

- Régulateur de convergence d'une suite convergente [6].
- Série convergente et produit infini convergent, à partir de Σ , Π et de suite convergente (point précédent).

- Module de continuité d'une application d'un intervalle fermé borné du Continu [élémentaire] dans le Continu [élémentaire], voir théorème de continuité en 4.1 et [8].

- Intégrale de Riemann sur un intervalle fermé borné d'une fonction définie sur cet intervalle, automatiquement uniformément continue.

- Suite itérative : u_0 donné, $u_{n+1} = f(u_n)$ d'éléments du Continu algorithmique. Lorsque la suite est convergente et que toutes les données sont élémentaires, alors la limite est encore dans le Continu

élémentaire. C'est nettement moins évident, en effet la définition conduit à $f^{nl}(u_0)$ qui fait appel à la 2-récursivité, mais la suite étant convergente, elle reste dans un intervalle fermé borné représenté par un déploiement finitaire, la réduction au premier ordre de 4.1 nous ramène à la primitive récursivité, de plus, le théorème 4 de III.5 permet de conclure que la nouvelle suite est élémentaire.

En fait, je ne connais aucun procédé de l'analyse, acceptable en logique intuitionniste, qui sorte du Continu élémentaire.

6. Conclusion

Le Continu élémentaire est largement suffisant pour faire de l'analyse intuitionniste dans un contexte purement mathématique. Si on préfère un Continu plus physique, le Continu engendré par les suites de choix, formalisées par l'axiome de l'éventail, trouve précisément, par le théorème algorithmique de l'éventail, un argument supplémentaire à sa justification [8].

Références bibliographiques

- [1] Erett BISHOP (1967) *Foundations of Constructive Analysis*. Mac Graw Hill.
- [2] Michael DUMMETT (1977) *Elements of Intuitionism*. Clarendon Press, Oxford.
- [3] Arend HEYTING (1971) *Intuitionism, an Introduction*. North Holland Publishing Company.
- [4] Marc JAMBON (2000) *Un paradoxe issu de la récursivité*. <http://www.univ-reunion.fr/~jambon>
- [5] Marc JAMBON (1999) *Notions et techniques de base*. Ellipses.
- [6] Marc JAMBON (1995) *Limites à partir des suites de référence*. Expressions n°6, mai 1995.
- [7] Marc JAMBON (2001) *Géométrie avec ou sans tiers exclu ?* Expressions n°18, octobre 2001.
- [8] Marc JAMBON (2003) *La droite graduée*. Expressions n°21, mai 2003.
- [9] Stephen Cole KLEENE (1952) *Introduction to Metamathematics*. North-Holland.
- [10] Stephen Cole KLEENE (1965) *The Foundations of Intuitionistic Mathematics*. North-Holland Publishing Company.
- [11] Maurice MARGENSTERN (1989) *Langage Pascal et logique du 1^{er} ordre*. Masson.
- [12] Helmut SCHWICHTENBERG (1997) *Recursion Theory*. Sommersemester 1997. Mathematisches Institut. Ludwig-Maximilians-Universität München.
- [13] Nikolaj Aleksandrovic SANIN (1968) *Constructive Real Numbers and Function Spaces*. Volume 21. Translations of Mathematical Monographs.