Utilisation en Coq de l'opérateur de description

Pierre Castéran

LaBRI, UMR 5800, Université Bordeaux 1,
351, Cours de la Libération, 33405 Talence Cedex,
`pierre.casteran@labri.fr`,
`www.labri.fr/~casteran/`

Résumé

Dans cet article, nous nous intéressons à l'introduction de l'opérateur de description indéfinie : « epsilon d'Hilbert » dans les développements *Coq*. Il est connu que cette adjonction n'est pas sans conséquences : la logique devient « presque classique », et le caractère informatif de types de sorte `Type` perd sa signification. En nous appuyant sur des exemples : fonctions partielles, présentation axiomatique des ordinaux dénombrables, nous étudions comment articuler l'utilisation de cette construction avec le style constructif de *Coq*.

1. Introduction

Nous nous intéressons à certains aspects *périphériques* — mais pas forcément marginaux — de *Coq*[6, 4]. Les énoncés de théorèmes prouvés à l'aide de cet outil peuvent se trouver éloignés d'un langage mathématique usuel, ce qui peut rendre difficile la communication de ces résultats au « monde extérieur ». Il importe donc — au sein du même outil — de considérer l'écriture et la preuve d'énoncés dans un langage compréhensible par le plus grand nombre, tout en respectant les méthodes de développement propres à *Coq*.

Dans cet article, nous nous intéressons au gain d'expressivité obtenu en ajoutant à *Coq* l'opérateur de description indéfinie, dit « opérateur `epsilon` d'Hilbert ». Cet opérateur nous fournit un mécanisme linguistique de *présupposition existentielle*, en dissociant l'écriture d'un terme de celle des conditions d'existence de l'objet correspondant, comme dans le discours « Supposons $x > 0 \ldots \log x < x \ldots$ », où le terme « $\log x$ » est une abréviation de l'expression « un nombre y (s'il existe) tel que $e^y = x$ ».

Les avantages que nous obtenons sont, outre un gain de lisibilité dû à l'usage d'implicites, la possibilité de nommer (au niveau global) des objets dont l'existence est assurée par la preuve d'une proposition, et l'utilisation de fonctions partielles.

Les travaux de Hurkens[12], Geuvers[9], et Bell[3] montrent que l'adjonction de l'opérateur `epsilon` ne se fait pas sans conséquences lourdes : incompatibilité avec l'imprédicativité de `Set`, travail en logique quasi-classique (classique si l'on admet que l'opérateur `epsilon` est extensionnel), et perte de lisibilité de certains énoncés : un énoncé de la forme $\forall x,\ \{P\,x\} + \{\sim P\,x\}$ ne signifie plus que P est décidable, car il peut se dériver du tiers-exclu.

A l'aide d'exemples, nous nous efforcerons de montrer l'intérêt de travailler en *Coq* augmenté de la déclaration d'`epsilon`, en tenant compte bien sûr des réserves ci-dessus. La structure de l'article est la suivante :
- présentation des domaines illustrant cette étude, empruntés à l'arithmétique simple, puis à la théorie des ordinaux dénombrables,
- présentation de l'opérateur `epsilon` et ses dérivés, et de sa représentation en *Coq*,
- traduction en *Coq*+ϵ des raisonnements dans les domaines ci-dessus,

– relations entre définitions par epsilon et définitions effectives,
– conclusion, et perspectives

Les définitions et preuves présentées dans cet article — aussi bien en notation mathématique qu'en *Coq* — sont toutes extraites d'un développement [5] réalisé en *Coq* avec Evelyne Contéjean et Florian Hatat dans le cadre du projet A3PAT de l'Agence Nationale de la Recherche. Nous avons pris quelques libertés avec la notation usuelle de *Coq* pour rendre les énoncés plus lisibles, et omis tout ou partie des preuves pour n'en retenir que les étapes illustrant notre propos. Ces omissions sont signalées par des ellipses « `...` » dans les scripts de preuve.

2. Fragments d'un discours en langage mathématique

Nous présentons dans cette section quelques exemples utilisant des fonctions partielles et des définitions de constantes à partir de preuves d'existence. Les domaines considérés sont l'arithmétique de base et la théorie des ordinaux dénombrables.

2.1. Fonctions partielles de $\mathbb{N}$ dans $\mathbb{N}$

Nous illustrons la notion de fonction partielle par un exemple très simple : La fonction qui à tout $n \in \mathbb{N}$ associe l'entier naturel p (s'il existe) tel que $2^p = n$ (logarithme exact de base 2). Représenter en *Coq* une telle fonction peut se faire de deux façons différentes : l'approximation par une fonction totale, ou la définition d'une constante d'un type autre que `nat→nat`.

La définition suivante construit une fonction totale de type `nat→nat` (`half` étant une fonction totale calculant la partie entière de la moitié de tout nombre naturel).

```
Function total_log2 (n:nat) {wf lt n using lt_wf} : nat :=
match n with | 0 ⇒ 0
             | 1 ⇒ 0
             | p ⇒ S (total_log2 (half p))
end.
```

On remarque que l'égalité — non voulue — `total_log2 24 = 4` se prouve immédiatement. La fonction `total_log2` n'est donc qu'une approximation de l'objet mathématique que nous voulions représenter.

On peut certes représenter une fonction partielle à l'aide d'un type option, d'un produit dépendant ou par une relation binaire. Si ces représentations sont adaptées aux preuves et calcul en *Coq*, elles ne permettent pas l'écriture d'énoncés proches de la notation mathématique, comme par exemple « *si n est une puissance de 2, alors* $2^{(log_2\ n)} = n$ ».

La traduction de cet énoncé utilisant l'une ou l'autre de ces représentations entraîne un manque de lisibilité par rapport à la notation mathématique classique.

2.2. Ordinaux dénombrables

Dans son ouvrage *Proof Theory*[18], Kurt Schütte présente deux aspects, l'un axiomatique, l'autre constructif, de la notion d'ordinal. Ces deux volets constituent le chapitre 5 : « Ordinal Numbers and Ordinal Terms » de ce livre.

Le premier de ces aspects est une présentation non constructive de la théorie des ordinaux, sous la forme d'une caractérisation axiomatique de l'ensemble $\mathbb{O}$ des ordinaux finis ou dénombrables[1].

[1] Par la suite, le terme « dénombrable » sera pris au sens de « fini ou dénombrable ».

Cette présentation se fait à partir d'un petit nombre d'axiomes, dans le cadre d'une théorie naïve des ensembles (on dit qu'une partie X de $\mathbb{O}$ est *bornée* si elle admet un majorant strict dans $\mathbb{O}$) :

Ax1 L'ensemble $\mathbb{O}$ est bien ordonné pour la relation $<$,

Ax2 Tout sous-ensemble borné de $\mathbb{O}$ est dénombrable,

Ax3 Tout sous-ensemble dénombrable de $\mathbb{O}$ est borné.

A partir de ces axiomes, Schütte définit les notions classiques : ordinal 0, successeur, fonctions normales, addition, ordinaux additifs principaux et ordinaux critiques, existence et unicité de la forme normale de Cantor. Ces définitions permettent de déterminer le segment initial des ordinaux inférieurs à l'ordinal Γ_0 [14]. Ce développement de 13 pages est un très bel exemple de texte mathématique dans un style classique (avec un degré raisonnable de détails des preuves).

Schütte donne en second lieu une construction effective des ordinaux inférieurs à Γ_0. Cette construction se fait à partir de leur représentation en forme normale de Veblen : des termes finis, une relation d'ordre total et une notion de forme normale, toutes deux décidables. Les opérations comme le successeur, l'addition, l'exponentiation de base ω et les fonctions de Veblen sont définies par récurrence structurelle.

Cette étude présente un grand intérêt par la comparaison qu'elle permet entre développements constructifs et raisonnements mathématiques classiques. Il va de soi que ces deux aspects ne peuvent rester indépendants : les structures de données représentant les ordinaux comme ϵ_0 ou Γ_0, ainsi que les opérations sur ces données, doivent être validées par rapport à la définition mathématique.

2.3. Quelques détails de la construction axiomatique de Γ_0

Nous donnons ci-dessous quelques détails des définitions et preuves utiles à la compréhension de leur transcription en *Coq*. Schütte considère une définition « classique » : Un ensemble M est bien ordonné par un ordre total $\leq$ si tout sous-ensemble non vide de M admet un plus petit élément. Nous respecterons ce choix dans notre transcription en *Coq*.

2.3.1. Premiers résultats

Les axiomes de Schütte permettent de montrer que l'ensemble $\mathbb{O}$ n'est ni dénombrable ni borné. Un raisonnement en logique classique nous permet d'obtenir le schéma de récurrence transfinie (récurrence bien fondée sur $(\mathbb{O}, <)$). En appliquant `Ax3` à l'ensemble vide, nous prouvons que l'ensemble $\mathbb{O}$ est non vide. De façon générale, si X est une partie dénombrable de $\mathbb{O}$, l'ensemble de ses majorants est non vide, et admet donc un plus petit élément dans $\mathbb{O}$ que nous notons $\bigsqcup X$.

L'ordinal $\bigsqcup\emptyset$ permet de prouver la proposition $\exists!\, o \in \mathbb{O}, \forall\, \alpha \in \mathbb{O}, o \leq \alpha$, qui sert de définition à l'ordinal 0. Nous définissons également le successeur d'un ordinal α : comme l'ensemble $\mathbb{O}$ est non borné, il existe un ordinal β supérieur à α. L'ensemble des ordinaux supérieurs à α est donc non vide, et admet alors un plus petit élément (unique), que nous appelons $\mathrm{succ}(\alpha)$.

2.3.2. Segments et fonctions d'énumération

Un *segment initial* de $\mathbb{O}$ est un sous-ensemble A de $\mathbb{O}$ tel que si $\alpha < \beta$ et $\beta \in A$, alors $\alpha \in A$; un *segment propre* de $B \subseteq \mathbb{O}$ est un ensemble de la forme $B(\beta) = B \cap \{\alpha \in \mathbb{O} | \alpha < \beta\}$. On prouve que tout segment initial strictement inclus dans $\mathbb{O}$ est un segment propre de la forme $\mathbb{O}(\beta)$ pour un certain β.

Soit $B \subseteq \mathbb{O}$; une *fonction d'énumération*[2] pour B est une fonction strictement croissante, dont le domaine est un segment initial de $\mathbb{O}$, et l'image est B.

[2] *Ordering function* en anglais

On montre par récurrence transfinie que si f est une fonction d'énumération de B, de domaine A, alors $a \leq f(a)$ pour tout $a \in A$, et que tout sous ensemble $B \subseteq \mathbb{O}$ admet au plus une fonction d'énumération.

La preuve que tout $X \subseteq \mathbb{O}$ admet au moins une fonction d'énumération est plus complexe :

1. On montre le lemme suivant :

   > Soit B tel que tout segment propre de B admet une fonction d'énumération ; alors B admet lui même une fonction d'énumération.

   Cette fonction est construite point par point :

   (a) Soit $\beta \in X$, et $f_\beta : A_\beta \rightarrow B(\beta)$ une fonction d'énumération de $B(\beta)$. Comme f_β est bijective, A_β est dénombrable, et est donc un segment initial de la forme $\mathbb{O}(\gamma)$. On pose alors $g(\beta) = \gamma$.

   (b) On prouve ensuite que g est strictement croissante, et a pour image un segment initial de $\mathbb{O}$.

   (c) Il suffit alors de prouver que la bijection réciproque de g est une fonction d'énumération de B.

2. Pour montrer que tout sous-ensemble B de $\mathbb{O}$ admet au moins une fonction d'énumération, on prouve d'abord par une récurrence transfinie sur β que tout segment propre $B(\beta)$ admet une fonction d'énumération, puis on applique le lemme précédent.

Les résultats ci-dessus nous permettent de définir pour tout $B \subseteq \mathbb{O}$ et tout ordinal α, l'écriture « le α-ième élément de B » signifiant « l'image de α par l'unique fonction d'énumération de B ». Cette écriture n'a bien entendu de sens que si α appartient au domaine de cette fonction d'énumération.

2.3.3. Addition, ordinaux additifs principaux et critiques

Soient α et β deux ordinaux. On définit $\alpha + \beta$ comme le β-ième ordinal supérieur ou égal à alpha. Les propriétés des fonctions d'énumération permettent de déduire des propriétés de l'addition dans $\mathbb{O}$, parmi lesquelles l'associativité, la monotonie (stricte sur le second argument), et l'existence d'un unique ξ tel que $\alpha + \xi = \beta$ (si $\alpha \leq \beta$).

Un ordinal $\alpha > 0$ est *additif principal* si pour tout $\beta < \alpha$, on a $\beta + \alpha = \alpha$. On appelle ϕ_0 la fonction d'énumération de l'ensemble AP des additifs principaux. La notation ω^α est classiquement utilisée en lieu et place de $\phi_0(\alpha)$. Pour tout ordinal α, il existe une unique décomposition $\alpha = \omega^{\alpha_1} + \ldots \omega^{\alpha_n}$ avec $\alpha \geq \alpha_1 \geq \cdots \geq \alpha_n$ (forme normale de Cantor).

Terminons par la définition (quelque peu complexe) de l'ensemble $\mathrm{Cr}(\alpha)$ des ordinaux α-critiques :

1. $\mathrm{Cr}(0)$ est l'ensemble AP des ordinaux additifs principaux,
2. $\phi_\alpha : A_\alpha \rightarrow \mathrm{Cr}(\alpha)$ est la fonction d'énumération de $\mathrm{Cr}(\alpha)$,
3. Si $0 < \alpha$, $\mathrm{Cr}(\alpha)$ est l'ensemble des points fixes communs à toutes les ϕ_β, pour $\beta < \alpha$.

2.4. Structure des définitions précédentes

Les extraits présentés ci-dessus comportent plusieurs définitions d'objets à partir de preuves d'existence ; citons par exemple la définition du successeur de α, à partir de l'existence d'au moins un ordinal supérieur à α, la phrase « On pose alors $g(\beta) = \gamma$ » : où l'existence de γ provient de l'énumérabilité du domaine de f_β, et la locution « le α-ième ordinal de B », conséquence de l'existence et de l'unicité de la fonction d'énumération de B.

Remarquons que les deux derniers points utilisent des fonctions partielles. De même, l'expression $\bigsqcup X$ induit la présupposition « X est dénombrable ».

Cette théorie permet de définir des symboles comme « successeur », « le $\sqcup$-ième », $+$, etc. Ces symboles doivent bien sûr avoir une portée *globale* ; dans le cadre de l'implantation en *Coq* de cette théorie, ces symboles doivent pouvoir s'exporter vers d'autres modules.

3. Opérateurs de description

Les opérateurs de description ont été introduits en 1923 par David Hilbert, pour définir explicitement les quantificateurs existentiel et universel, et plus généralement représenter les symboles mathématiques à l'intérieur de systèmes déductifs formels. L'histoire de la création des opérateurs τ, ϵ et ι est présentée dans [20].

3.1. Descriptions indéfinies

Un epsilon-terme est un terme de la forme $\epsilon_x \,.\, A(x)$, où x est une variable et A un prédicat. L'interprétation voulue est « un x satisfaisant $A(x)$, s'il existe, sinon un objet arbitraire ». Cette interprétation est formalisée dans l'axiome dit « axiome transfini » : $A(y) \Rightarrow A(\epsilon_x \,.\, A(x))$.

Les quantificateurs peuvent être définis (en logique classique) par les équivalences ci-dessous ; cette représentation est notamment utilisée en HOL[10], Isabelle/HOL[16] et PVS[17].

$$\begin{array}{rcl} \exists x, A(x) & \Leftrightarrow & A(\epsilon_x \,.\, A(x)) \\ \forall x, A(x) & \Leftrightarrow & A(\epsilon_x \,.\, {\sim} A(x)) \end{array}$$

L'opérateur `epsilon` permet de dissocier la spécification d'un objet de sa preuve d'existence. En particulier il autorise la définition de fonctions partielles à partir de relations binaires. Par exemple, la locution « un majorant de » se traduit à l'aide de l'opérateur de description indéfinie : $\lambda\, X \,.\, \epsilon_m \,.\, (\forall\, x,\ x \in X \Rightarrow x \leq m)$.

La relation entre l'opérateur de description et la correspondance de Curry-Howard est étudiée dans l'article d'Abadi *et al.* [1]. Cet opérateur est aussi considéré avec intérêt dans la sémantique des langues naturelles, notamment pour la traduction des syntagmes nominaux indéfinis et des anaphores[19].

3.2. Descriptions définies

Un iota-terme s'écrit $\iota_x \,.\, A(x)$ et a pour signification « l'objet x tel que $A(x)$ — s'il existe et est bien unique —, sinon un objet arbitraire ».

En fait, l'opérateur `iota` de description définie se dérive d'`epsilon`, en posant $\iota_x \,.\, A(x) \ := \ \epsilon_x \,.\, (\mathrm{unique}(A))(x)$, où *unique* est la transformation $\lambda\, A \,.\, \lambda\, x \,.\, [A(x) \ \wedge \ (\forall\, y\,,\ A(y) \Rightarrow x = y)]$.

Les deux exemples ci-dessous sont des descriptions de fonctions partielles : le logarithme « exact »de base 2, et la borne supérieure d'un ensemble X :

$$\begin{array}{rcl} \log_2(n) & = & \iota_p \,.\, 2^p = n \\ \bigsqcup X & = & \iota_x \,.\, [(\forall y, y \in X \Rightarrow y \leq x) \ \wedge \ (\forall z, (\forall y, y \in X \Rightarrow y \leq z) \Rightarrow x \leq z)] \end{array}$$

4. L'opérateur epsilon en *Coq*

4.1. Définition

L'opérateur `epsilon` est défini dans le module `Logic.ClassicalEpsilon` de la bibliothèque standard de *Coq*. Ce module importe `Logic.Classical`, et admet donc le tiers exclu ; l'axiome suivant permet l'extraction d'un témoin à partir d'une preuve d'existence :

```
Axiom constructive_indefinite_description :
  ∀(A : Type)(P : A→Prop), (∃x, P x) → {x : A | P x }.
```

On en dérive les définitions suivantes, où `inhabited` A est une proposition signifiant que le type A est habité :

```
epsilon: ∀(A:Type)(P:A→Prop),inhabited A → A
epsilon_spec: ∀(A:Type)(i:inhabited A)(P:A→Prop), (∃x, P x) → P(epsilon i P)
```

Nous remarquons qu'un `epsilon`-terme est de la forme `epsilon` i P, le paramètre i permettant d'éviter la création d'un terme de type A dans le cas où A est un type vide. La valeur de ce paramètre est indifférente : on prouve en effet l'implication $(\exists x,\, P(x)) \rightarrow (\texttt{epsilon}\, i\, P = \texttt{epsilon}\, j\, P)$, où i et j sont deux preuves de `inhabited` A.

4.2. Conséquences de l'introduction en *Coq* de l'opérateur epsilon

La déclaration de `constructive_indefinite_description` est loin d'être anodine. Le module `Logic.ChoiceFacts` de la bibliothèque standard est consacré aux relations entre le choix et les opérateurs de description. En premier lieu, cet axiome implique, et est non contradictoire avec la propriété de « choix fonctionnel » :

```
∀(A B:Type)(R:A→B→Prop)
   (∀x : A, ∃y : B, R x y) → (∃f : A→B, ∀x : A, R x (f x)).
```

Par ailleurs, nous pouvons construire un terme de type `∀P Q : Prop, P∨Q → {P}+{Q}` ; En logique classique, nous pouvons en dériver $\forall P : \mathit{Prop},\ \{P\} + \{\sim P\}$. Depuis Hurkens[12] et Geuvers[9], nous savons que ce résultat est incohérent avec le Calcul des Constructions avec `Set` imprédicatif. Si l'on travaille avec l'option par défaut (`Set` prédicatif) de *Coq*, il faut néanmoins être conscient qu'un terme de type $\forall x,\ \{P\ x\} + \{\sim P\ x\}$ se dérive immédiatement à partir du tiers exclu, et n'indique plus forcément que P est décidable.

En dernier lieu, citons un résultat de J. Bell[3]. Si l'on déclare *en logique intuitionniste* une constante `epsilon` satisfaisant `epsilon_spec`, et possédant la propriété d'extensionnalité :

si P et Q sont logiquement équivalents, alors `epsilon` i P `=` `epsilon` i Q,

alors on peut prouver le tiers exclu, et retrouver ainsi la logique classique. L'opérateur de description est donc fortement lié à la logique classique.

4.3. La bibliothèque hilbert

A partir de la définition ci-dessus, nous avons développé quelques outils permettant de faciliter l'utilisation de l'opérateur de description. Ces outils sont essentiellement des opérateurs dérivés de `epsilon`, ainsi que des règles et tactiques d'élimination. Les exemples d'utilisation sont présentés en section 5, et les scripts complets dans [5].

4.3.1. Règle d'élimination pour `epsilon`

La règle d'élimination pour `epsilon` est une paraphrase du lemme `epsilon_spec` : Si Q est un prédicat sur A, nous pouvons déduire la proposition $Q($`epsilon` $i\ P)$ à partir des prémisses : $\exists\, x,\ P\,x$ et $\forall\, x, P\,x \rightarrow Q\ x$. La tactique `epsilon_e` t où t se réduit en un epsilon-terme revient à appliquer cette règle.

4.3.2. Description définie

L'opérateur `iota` est défini par `iota` $i\,P$ `:= epsilon` i `(unique` P`)`. La tactique `epsilon_e` s'applique alors aussi aux descriptions définies : le prédicat P est remplacé par `unique` P dans les buts engendrés.

4.3.3. Définitions à partir de preuves

Considérons les constructions linguistiques suivantes :

« Il existe un x tel que $P(x)$; soit a un tel élément »

« Il existe un unique x tel que $P(x)$; appelons le a »

Nous pouvons en donner un équivalent formel à l'aide de nouveaux opérateurs dérivés : Si H est une preuve de $\exists\, x : A,\ P\,x$, alors nous pouvons définir `select` H comme le terme `epsilon` $i\,H$, où i est une preuve que A est habité, prouvable à partir de H.

Nous définissons de la même manière `select_the` H si H est une preuve de $\exists!\, x : A,\ P\,x$ en utilisant l'opérateur `iota` au lieu d' `epsilon`.

Notons que si un terme t est défini à l'aide d'un de ces deux opérateurs, il contient une preuve d'existence. La tactique d'élimination `epsilon_e` est adaptée pour ne pas redemander cette preuve (voir la preuve de `le_zero` en section 5.1.1).

4.3.4. Relations binaires et fonctions partielles

Soient A et B deux types et i : inhabited B. Nous pouvons associer à toute relation R de A vers B la fonction qui à a associe le terme `epsilon` i `(fun` $b \Rightarrow R\,a\,b$`)`.

Soit D un prédicat sur A ; si nous disposons d'une preuve H de la proposition $\forall\, x, D\,x \rightarrow \exists\, y,\ R\,x\,y$, nous pouvons construire le terme `some_fun` $i\,H\ =_{def}$ `fun` $a \Rightarrow$ `epsilon` i `(fun` $b \Rightarrow$ `D a` $\wedge$ `R a b)`.

La règle d'élimination associée énonce que, si f est réductible en `some_fun` $i\,H$, alors la proposition $Q\,x\,(f\,x)$ est impliquée par les propositions $D\,x$ et $\forall\ y,\ D\,x \rightarrow R\,x\,y \rightarrow Q\,x\,y$.

Une construction similaire `the_fun` permet de traiter le cas où pour tout x vérifant $D\,x$, il existe un unique y tel que $R\,x\,y$. Cette construction et la tactique d'élimination associée `partial_fun_e` sont ilustrées section 5.1.2.

5. Présentation axiomatique des ordinaux en *Coq*

Nous montrons par quelques exemples comment les outils présentés plus haut ont permis de transcrire le texte mathématique de Schütte. Cette transcription se trouve dans le répertoire `schutte` de[5]. Ce développement est dans son état actuel assez long, mais il a pu s'obtenir par une traduction directe des arguments de Schütte, dont la structure est directement reflétée dans celle des définitions, preuves et sections du texte *Coq*.

5.1. Définitions de base

Nous utilisons le module `Ensembles` de la bibliothèque standard de *Coq*. Dans ce but, nous déclarons une constante `OT :Type`, et une structure de bon ordre `ON` sur le type `OT`, composée d'un ensemble $\mathbb{O}$ `: Ensemble OT`, d'une relation `le :OT→OT→Prop` d'ordre total sur $\mathbb{O}$.

Nous appelons `lt` l'ordre strict associé à `le`. Par la suite, nous utiliserons souvent les notations $\leq$ et $<$ en lieu et place de `le` et `lt`. La constante *Coq* `ordinal` est associée à l'ensemble $\mathbb{O}$; par la suite, nous emploierons indifféremment les écritures $\alpha \in \mathbb{O}$ et `ordinal` α.

La déclaration de `ON` traduit l'axiome `Ax1` de Schütte. Les deux axiomes suivants sont traduits directement en *Coq* à l'aide du module `Denumerable` écrit par Florian Hatat[5] :

```
Axiom Ax2 : ∀X: Ensemble OT, Included X ordinal →
                (∃a, ordinal a ∧ (∀y, X y → y < a)) →
                denumerable X.

Axiom Ax3 : ∀X : Ensemble OT,
               denumerable X → (Included X ordinal) →
               ∃a, ordinal a ∧ (∀y, X y → y < a).
```

Les diverses constructions utilisant les opérateurs de description dans ce développement nécessitent des lemmes affirmant que les types `OT`, `Ensemble OT` et `OT→OT` sont habités. Nous appelons respectivement ces lemmes `inh_OT`, `inh_Ensemble_OT` et `inh_OT_OT`. La preuve ci-dessous est une application de l'axiome `Ax3` :

```
Lemma inh_OT : inhabited OT.
Proof.
 case (AX3 (denumerable_empty OT)); firstorder.
Qed.
```

5.1.1. Définition de l'ordinal 0

L'ordinal 0 est défini à partir d'une preuve d'existence unique :

```
Definition zero_spec (z:OT) := ordinal z ∧ ∀α, ordinal α → z ≤ α.

Lemma zero_defined : ∃!o, zero_spec o.
Proof.
 exists (⨆ (Empty_set OT)). ...


Definition zero : OT := select_the zero_defined.

Lemma le_zero : ∀ a, ordinal a → zero ≤ a.
Proof.
 intros;epsilon_e zero.
```

a : OT
H : ordinal a
============================
$\forall\ b : OT,\ unique\ (\lambda\, o : OT.\, zero\_spec\ o)\ b \rightarrow b \leq a$

```
 unfold zero_spec; destruct 1; intuition.
Qed.
```

Nous remarquons que la définition de `zero` est issue d'une preuve d'existence, alors que le type de `zero` est de sorte `Type`. Cette définition, de portée globale, peut être utilisée dans n'importe quel module client.

5.1.2. Définition de la fonction successeur

La fonction successeur est définie comme une fonction partielle de domaine $\mathbb{O}$, associée à la relation définie par « `succ_spec` a b si b est le plus petit ordinal supérieur à a ». Dans la définition de `succ`, l'utilisation de `refine` engendre une obligation de preuve, dont la solution devient partie de la valeur de la constante `succ`.

```
Definition succ_spec (a:OT) := is_least_member ordinal lt (λz . a < z).

Definition succ : OT → OT.
 refine (the_fun _ ordinal succ_spec _); auto.

  ============================
   ∀α : OT, ordinal α → ∃! x, succ_spec α x
...
```

Le début de preuve ci-dessous montre comment prouver une proposition de la forme $Q\,\alpha\,(\texttt{succ}\,\alpha)$.

```
Lemma lt_succ_le : ∀ α β, α < β → succ α ≤ β.
Proof.
 intros α β H; partial_fun_e α (succ α).

  H : α < β
  ============================
   ordinal α

subgoal 2 is:
 ∀γ : OT, ordinal α → unique (succ_spec α) γ → γ ≤ β
...
```

5.2. Fonctions d'énumération

La définition de l'addition et de l'exponentiation de base ω repose sur l'existence d'une fonction d'énumération pour tout sous-ensemble $B \subseteq \mathbb{O}$ (voir la section 2.3.2). L'adaptation à *Coq* de la preuve de cette existence a posé quelques problèmes : en effet, considérons la phrase suivante : « Soit $\beta \in X$, et $f_\beta : A_\beta \rightarrow B(\beta)$ une fonction d'énumération de $B(\beta)$ ». Cette phrase déclare en fait *deux* symboles, dépendant de β : f_β est lié à une fonction de type `OT→OT`, et A_β est associé au domaine de la fonction (partielle) d'énumération de $B(\beta)$.

La traduction en *Coq* explicite ces deux définitions, à partir de la proposition « f est une fonction d'énumération de B, de domaine A », notée " `ordering_function` $f\ A\ B$ ". On commence par spécifier l'unique domaine d'une éventuelle fonction d'énumération du segment propre $B(\beta)$.

```
Definition ordering_segment(A B : Ensemble OT) : Prop
```

```
:= ∃f : OT → OT, ordering_function f A B.

Definition A_beta : Ensemble OT
:= iota inh_Ensemble_OT (fun E ⇒ ordering_segment E (proper_segment_of B β)).
```

La fonction d'énumération du segment propre $B(\beta)$ peut se définir à l'aide d'un epsilon-terme[3] :

```
Definition f_beta := OT → OT
:= epsilon inh_OT_OT
          (fun f ⇒ ordering_function f A_beta (proper_segment_of B β)).
```

Une fois prouvées par récurrence transfinie l'existence et l'unicité d'une fonction d'énumération pour tout $B \subseteq \mathbb{O}$, nous définissons deux symboles globaux pour désigner cette fonction partielle :

```
Definition the_ordering_segment (B:Ensemble OT) : Ensemble OT
:= iota inh_Ensemble_OT (fun x ⇒ ordering_segment x B).

Definition alpha_th (B:Ensemble OT) : OT → OT
:=  epsilon inh_OT_OT (fun f ⇒ ordering_function f the_ordering_segment B).

Theorem alpha_th_ok
     : ∀ B : Ensemble OT,
       Included B ordinal →
       ordering_function (alpha_th B) (the_ordering_segment B) B.
```

Cette preuve, bien qu'assez longue, est la traduction directe des arguments de Schütte, et en respecte la structure. Les epsilon- et iota-termes nous permettent de définir des symboles de sorte `Type` de portée globale, par élimination de termes de preuves d'existence, parfois obtenus par des arguments « classiques ». Par exemple la constante `alpha_th` est utilisée dans le module dédié à l'addition sur $\mathbb{O}$.

5.3. Addition

L'addition dans $\mathbb{O}$ est définie directement à l'aide de la fonctionnelle `alpha_th`.

```
Definition Greater α := funβ ⇒ α ≤ β.
Definition plus α := alpha_th (Greater α).
Notation "α + β" := (plus α β) : ord_scope.
```

On prouve que le domaine de `plus` α est tout l'ensemble $\mathbb{O}$. Le fait que `plus` α soit la fonction d'énumération de l'ensemble des ordinaux supérieurs ou égaux à α nous permet de prouver immédiatement les propriétés listées en 2.3.3.

Voici par exemple la preuve de $\beta \leq \alpha + \beta$. Le lemme technique `plus_elim` permet de remplacer un but de la forme $Q($`plus` $\alpha)$ par $Q(f)$ où f est une fonction d'énumération arbitraire de `Greater` α, de domaine $\mathbb{O}$. Dans ce cas précis, nous pouvons appliquer l'inégalité $\beta \leq f\,\beta$ (voir section 2.3.2) :

```
Lemma le_plus_r : ∀αβ, ordinal α → ordinal β → β ≤ α + β.
Proof.
```

[3]Nous n'utilisons pas de iota-terme, pour ne pas dépendre de l'extensionnalité des fonctions.

```
 intros α β Hα Hβ.
 pattern (plus α); apply plus_elim;auto.
```

Hα : ordinal α
Hβ : ordinal β
============================
$\forall\ f : OT \to OT,\ \ ordering\_function\ f\ ordinal\ (Greater\ \alpha) \to \beta \leq f\,\beta$
...

5.4. Soustraction

La soustraction dans $\mathbb{O}$ est définie par un iota-terme, à partir de l'addition :

```
Definition minus (α β :OT):= iota inh_OT (λγ. ordinal γ ∧ β + γ = α).
```

Un premier lemme nous donne une condition suffisante de définition de la soustraction :

```
Lemma minus_defined : ∀α β, α ≤ β → ∃!γ : OT, ordinal γ ∧ β + γ = α.
```

Dans la preuve suivante, l'élimination du terme α - β provoque deux obligations de preuves ; la première utilise l'inégalité $\beta \leq \alpha+\beta$, la seconde est une application directe du lemme `minus_defined`.

```
Lemma minus_le : ∀ α β, β ≤ α → α - β ≤ α.
Proof.
 intros α β H; epsilon_e (α-β).
```

H : $\beta \leq \alpha$
============================
$\forall\ \ \beta\ :\ OT,\ unique\ (\lambda\gamma\ :\ OT.\ ordinal\ \gamma \wedge \beta + \gamma = \alpha)\ \beta \to\ \ \beta \leq \alpha$

subgoal 2 is:
$\exists\,!\ \gamma\ :\ OT,\ ordinal\ \gamma \wedge \beta + \gamma = \alpha$
...

5.5. Ordinaux critiques

La transcription en *Coq* de la définition simultanée des ordinaux critiques et des fonctions ϕ_α de Veblen en section 2.3.3 a posé quelques problèmes. En effet, elle ne satisfait pas les conditions de positivité demandées par les définitions inductives.

Pour contourner cet obstacle, nous avons adapté les techniques de Balaa-Bertot[2] aux fonctions partielles, et défini la constante `critical` de type $\forall\alpha$, `ordinal` $\alpha \to$ `Ensemble OT` comme point-fixe de la fonctionnelle ci-dessous :

```
Definition critical_fun (α:OT)(Φ:∀γ, γ<α -> Ensemble OT)(_ : ordinal α)
: Ensemble OT
:= λ (β:OT) . ordinal β ∧
              ((α=0 ∧ AP β) ∨
               (0<α ∧ ∀(γ:OT)(H:γ<α), the_ordering_segment (Φ γ H) β  ∧
                                          alpha_th (Φ γ H) β = β)).
```

Les règles d'introduction et d'élimination du prédicat `critical` se déduisent du théorème de point fixe (module `prelude.Pfix` de [5]). Les symboles `Cr` et ϕ sont alors définis par projection :

```
Definition Cr(α : OT) : Ensemble OT := λβ.∃H: ordinal α, critical H β.
Definition phi(α : OT) : OT→OT := alpha_th (Cr α).
```

Nous pouvons alors énoncer et démontrer des théorèmes dans un vocabulaire proche de celui du texte mathématique, ce que ne permet pas le type de `critical` ; en voici trois exemples, le troisième permettant de s'assurer de l'existence d'ordinaux critiques.

```
Lemma phi0_well_named : ∀α, phi0 α = phi zero α.
Lemma Cr_incl : ∀α β, β≤α → Included (Cr α) (Cr β).
Theorem Cr_unbounded: ∀α, ordinal α → ∀β , ordinal β → ∃γ , γ ∈ Cr α ∧ β < γ.
Definition epsilon0 := phi (succ zero) zero.
Theorem epsilon_fixpoint : phi0 epsilon0 = epsilon0 (* ω^ε0 = ε0 *).
```

6. Définitions par `epsilon` et définitions effectives

Les objets « idéaux » représentés à l'aide de l'opérateur de description indéfinie et de ses dérivés se prêtent fort bien à l'énoncé clair et à la preuve de leurs propriétés mathématiques. Néanmoins, un terme construit à l'aide d'`epsilon` ne se prête pas à l'évaluation.

Dans les cas où l'on peut disposer d'une représentation effective d'un tel objet, une solution au problème de l'évaluation consiste en la preuve de lemmes de la forme $\forall x, P\,x \rightarrow f\,x = e\,x$, où P est un prédicat impliquant que $f(x)$ est défini.

Par exemple le logarithme de base 2 de la section 2.1 peut se définir à partir d'une relation binaire :

```
Inductive rlog2 : nat → nat → Prop :=
| rlog2_1 : rlog2 1 0
| rlog2_even : ∀n q, 0 < n → even n → rlog2 (n/2) q → rlog2 n (S q).

Definition log2 n = iota inhabited_nat (fun p → rlog2 n p).
```

Nous prouvons un lemme de correction de la fonction `total_log2` définie en section 2.1 ; un calcul de `log2` n peut alors s'effectuer par une réécriture suivie d'une réduction et d'une vérification que n vérifie bien les conditions de cette réécriture.

```
Lemma total_log2_ok : ∀n, is_a_power_of_2 n → log2 n =  total_log2 n.
...

Goal log2 64 = 6.
 rewrite total_log2_ok; [reflexivity | idtac]

 ====================
 is_a_power_of_two 64.
...
```

Le cas des ordinaux dénombrables est bien plus complexe. Une représentation effective de ces objets doit permettre d'effectuer des comparaisons et d'évaluer des expressions arithmétiques. Nous savons représenter de façon compacte les ordinaux inférieurs à ϵ_0 (forme normale de Cantor) et à Γ_0 (forme normale de Veblen).

Nous utilisons pour ϵ_0 la représentation que Manolios et Vroon ont développé pour ACL2 [13]. La représentation des ordinaux inférieurs à Γ_0 en est très largement inspirée : dans le type inductif suivant, le terme $\mathtt{cons}\,\alpha\,\beta\,n\,\gamma$ s'interprète comme $\psi_\alpha(\beta)\times(n+1)+\gamma$, où ψ_α est la variante sans point fixe de la fonction ϕ_α [18, 14].

```
Inductive T2 : Set :=  zero : T2 | cons : T2→T2→nat→T2→T2.
```

Sur ce type, nous définissons une notion d'ordre total décidable, un prédicat `nf` « être en forme normale » ; les fonctions successeur, $+$, ϕ_α sont définies de façon récursive primitive. Le schéma de récurrence transfinie est prouvé à l'aide d'un plongement dans l'ordre récursif des chemins avec statut [8], et en appliquant le théorème de fondation de cet ordre (preuve d'Évelyne Contéjean incluse dans [5]) :

```
Definition transfinite_induction :
 ∀ (P:T2 → Type),
   (∀x:T2, nf x → (∀ y:T2, nf y → y < x → P y) → P x) →
   ∀ a, nf a → P a.
```

L'arithmétique effective sur les ordinaux en forme normale de Veblen est de définition complexe et peu lisible. Par exemple, la définition inductive de l'ordre $<$ sur T2 ne comporte pas moins de sept constructeurs. La définition primitive récursive de l'addition, qui utilise une fonction de comparaison sur T2 est elle-même peu intuitive :

```
Fixpoint plus (t1 t2 : T2) {struct t1}:T2 :=
  match t1,t2 with
 |  zero, y  ⇒ y
 |  x, zero ⇒ x
 |  cons a b n c, cons a' b' n' c' ⇒
      (match compare (cons a b 0 zero)
                     (cons a' b' 0 zero)
       with | Lt ⇒ cons a' b' n' c'
            | Gt ⇒ cons a b n (c + (cons a' b' n' c'))
            | Eq ⇒ cons a b (S(n+n')) c'
       end)
 end
where "alpha + beta" := (plus alpha beta): g0_scope.
```

Il est important de prouver la correction de telles représentations, relativement aux définitions axiomatiques plus simples et plus faciles à appréhender (*c.f.* la définition de l'addition en section 2.3.3). Prouver la correction de ces fonctions revient à établir un isomorphisme strictement croissant entre les termes de T2 en forme normale et le segment des ordinaux inférieurs à Γ_0.

Dans l'état actuel de notre développement, nous avons construit une bijection strictement croissante entre les termes en forme normale de Cantor et les ordinaux de $\mathbb{O}$ inférieurs à ϵ_0. La correction des fonctions arithmétiques et l'extension à Γ_0 sont en cours de développement.

7. Conclusion et perspectives

L'introduction des opérateurs de description dans la logique de *Coq* permet d'exprimer simplement les fonctions partielles et de définir des constantes à partir de preuves d'existence, ces dernières pouvant être obtenues en logique classique. Une telle extension de *Coq* nous rapproche du langage mathématique usuel, tant pour les énoncés de théorèmes que pour les enchaînements de définitions et de preuves.

On ne peut bien sûr ignorer les perturbations que crée l'adjonction de ces opérateurs au Calcul des Constructions Inductives. Nous pensons cependant que ces inconvénients seraient surtout dûs à une mauvaise compréhension de cet apport, notamment l'interprétation des types `sig` et `sumbool`. Quel utilisateur naïf penserait avoir écrit en une ligne une procédure certifiée de décision de l'arrêt des machines de Turing ? Une telle illusion serait vite balayée :

```
Definition halting_dec : ∀n, {halts (machine n)} + {~halts (machine n)}.
Proof.
 intro n; apply or_to_sumbool; apply classic.
Defined.

Extraction halting_dec
```

Warning: You must realize axiom constructive_indefinite_description in the extracted code.

Le système de modules de *Coq* permet en outre de structurer une bibliothèque en un « noyau » écrit en *Coq* « pur », avec une interface écrite en langage mathématique usuel. Dans notre travail sur les ordinaux, la partie « effective » concernant les systèmes de notation permet d'extraire des algorithmes de calcul, alors que les modules utilisant les opérateurs de description fournissent un modèle reflétant la relation avec les bons ordres.

Ce travail peut être prolongé dans plusieurs directions :

En premier lieu, la création d'outils destinés à compenser les risques de confusion que fait naître une telle extension de *Coq* :

- Une commande appropriée doit permettre de vérifier si un terme est construit de façon *effective*. Ce test échouerait par exemple sur la définition `halting_dec` ci-dessus, mais permettrait de vérifier qu'une définition d'ϵ_0 comme limite de la suite $\langle 1, \omega, \omega^2, \ldots, \omega^i, \ldots \rangle$ est *construite effectivement* à partir de la notion de borne supérieure d'un ensemble dénombrable et des fonctions ϕ de Veblen.
- L'exemple de la section 6 suggère d'étendre la nouvelle commande `Function` de *Coq* en admettant des définitions de fonctions récursives partielles, par exemple avec un filtrage non exhaustif.
- Pierre Corbineau a défini pour *Coq* un langage déclaratif [7] permettant d'écrire les preuves de façon lisible, assez proche d'Isar[15]. Il serait intéressant d'étendre ce langage par des constructions de la forme « Soit x l'objet défini par le lemme L ». Les preuves de [5] sont encore des longues suites d'appels de tactiques, et ont besoin d'être réécrites dans un langage lisible.

En second lieu, le développement sur les ordinaux doit s'étendre, en établissant le morphisme injectif entre les systèmes de notations effectifs et les segments initiaux associés ; $\epsilon_0, \Gamma_0, \Lambda$, etc. On disposera alors d'un cadre général non-constructif et d'implantations particulières. Nous espérons que ce double aspect permettra de définir des tactiques de preuve par réflexion. Nous prévoyons également d'intégrer un troisième point de vue sur les ordinaux : Brian Huffman[11] a écrit pour *Isabelle/HOL* une théorie des ordinaux dénombrables, présentés comme des classes d'équivalence sur l'algèbre des termes construits à partir de 0 par les opérations de successeur et d'ω-limite. Il sera intéressant de formaliser en *Coq* la compatibilité entre les trois types de représentation.

Remerciements L'auteur remercie Hugo Herbelin pour les informations sur les opérateurs de description dans la théorie des types. Évelyne Contéjean m'a aidé dans l'utilisation de ses bibliothèques de preuves sur l'ordre récursif des chemins, et Florian Hatat a écrit les définitions et preuves sur les ensembles dénombrables.

Références

[1] Martin Abadi, Georges Gonthier and Benjamin Werner. Choice in Dynamic Linking. FOSSACS 2004, number 2987 in LNCS. Springer-Verlag (March 2004), 12-26.

[2] Antonia Balaa and Yves Bertot. Fix-point equations for well-founded recursion in type theory. In *Proceedings of TPHOLs'2000*, number 1869 in LNCS. Springer Verlag, 2000.

[3] John Bell. *A Logical Approach to Philosophy, Essays in Honour of Graham Solomon, David DeVidi and Tim Kenyon ed.*, chapter Choice Principles in Intuitionistic Set Theory. Springer, 2006.

[4] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art : The Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. An EATCS series. Springer Verlag, 2004.

[5] Pierre Castéran, Évelyne Contéjean, and Florian Hatat. Ordinal notations and the rpo. www.labri.fr/~casteran/Cantor/Kantor.tar.gz. Coq files for Coq V8.1.

[6] Coq Development Team. The *Coq* proof assistant. Documentation, system download. Contact : `http ://coq.inria.fr/`.

[7] Pierre Corbineau. Un langage déclaratif de preuve pour coq. Disponible à http ://www.cs.ru.nl/~corbinea/mmode.fr.html.

[8] Nachum Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science.* 17(3), 279-301, March 1982.

[9] Herman Geuvers. Inconsistency of classical logic in type theory, 2001. Disponible à www.cs.kun.nl/~herman/note.ps.gz.

[10] Michael Gordon and Tony Melham. *Introduction to HOL.* Cambridge University Press, 1993.

[11] Brian Huffman. A theory of countable ordinals in Isabelle/HOL. available online at the Archive of Formal Proofs : www.afp.sourceforge.net.

[12] A.J. Hurkens. A simplification of Girard's paradox. In *Proceedings of the 2nd international conference Typed Lambda-Calculi and Applications (TCLA'95)*, 1995.

[13] Panagiotis Manolios and Daron Vroon. Algorithms for ordinal arithmetics. In Franz Baader, editor, *CADE*, volume 2741 of *LNCS*, pages 243–257. Springer-Verlag, 2003.

[14] George Moser and Ingo Lepper. Why ordinals are good for you. In ESSLLI 2003 course notes, *15th European Summer School in Logic Language and Information*, Vienna 2003.

[15] Tobias Nipkow. Structured proofs in Isar/HOL. In *Types for Proofs and Programs (TYPES 2002)*, number 2646 in LNCS, 2002.

[16] Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL, A Proof Assistant for Higher-Order Logic.* Number 2283 in LNCS. Springer Verlag, 2002.

[17] Sam Owre, Sreeranga P. Rajan, John M. Rushby, Natarajan Shankar, and Mandayam K. Srivas. PVS : Combining specifications, proof checking and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification, CAV'96*, volume 1102 of *Lecture Notes in Computer Science*, pages 411–414, 1996.

[18] Kurt Schütte. *Proof Theory.* Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1977.

[19] Kurt von Heusinger. Choice functions and the anaphoric semantics of definite NPs. Technical report, University of Konstanz, 2002. Proceedings of the Workshop "Choice Functions and Natural Languages Semantics.

[20] Richard Zach. Hilbert's "Verunglueckter Beweis," the first epsilon theorem, and consistency proofs. http ://front.math.ucdavis.edu/math.LO/0204255.