A propos des modèles de réalisabilité de ZF

Jean-Louis Krivine

IRIF - Université Paris-Diderot

krivine@irif.fr

Marseille, juin 2018

Introduction

J'exposerai ici quelques résultats et quelques problèmes
en réalisabilité classique. Pas de résultats nouveaux.
La *réalisabilité classique* est une extension de la technique du forcing.
Elle permet, comme le forcing, de construire de nouveaux modèles de ZF.
Mais elle permet, de plus, d'associer des *programmes* aux preuves
(correspondance de Curry-Howard).
La structure de ces modèles de ZF est beaucoup plus complexe
que dans le cas du forcing, au point que celui-ci doive être considéré
comme *dégénéré*.
Voyons les nouveautés essentielles, qui obligent à changer complètement
de méthode pour étudier ces modèles.

Nouvelles propriétés

- *La différence majeure, qui concentre la plupart des difficultés :*

Pour montrer une propriété du modèle de réalisabilité
il faut *inventer un programme* qui réalise cette propriété
Dans le cas du forcing, il n'y a rien à inventer car il n'y a qu'un seul "programme" :
la plus grande condition notée $\mathbf{1}$.
Rien que pour obtenir les axiomes de ZF, cela oblige à passer par
une théorie intermédiaire sans extensionnalité, appelée $\mathrm{ZF}_{\varepsilon}$, plus forte que ZF
qui a deux notions d'appartenance $\in$ (faible) et ε (forte, non extensionnelle).

- *Les ordinaux ne sont pas conservés.*

Dans le cas du forcing, lorsque le modèle $\mathscr{M}$ de base est dénombrable
et bien fondé, le modèle de forcing $\mathscr{N}$ en est *une extension bien fondée*
avec les mêmes ordinaux.

Nouvelles propriétés

Ce n'est plus le cas en réalisabilité classique.
Le modèle de réalisabilité $\mathcal{N}$ n'est *jamais bien fondé*
et a même, la plupart du temps, des entiers non standard.

- *L'algèbre de Boole caractéristique.*

Le modèle de réalisabilité $\mathcal{N}$ contient une algèbre de Boole, notée $\mathbb{B}2$
qui opère sur $\mathcal{N}$ par une opération de produit distributive.
Cela donne à $\mathcal{N}$ une structure de *modèle booléen* sur $\mathbb{B}2$
qui est une *extension élémentaire du modèle de base*.
Cette algèbre de Boole est triviale - uniquement - dans le cas du forcing.
Elle possède un ultrafiltre canonique, qui est très important
pour comprendre la structure du modèle de réalisabilité.

Nouvelles propriétés

- *Le modèle de réalisabilité ne satisfait pas nécessairement AC*

contrairement au cas du forcing.

Actuellement, la situation est la suivante, en dehors du forcing : dans tous les cas où on peut décider, DC est vrai et AC est faux. Ces cas sont fort généraux.

Problèmes ouverts :

DC est-il toujours vrai ? Noter que cela implique l'existence d'un programme écrit en λ_c-calcul pur, qui réalise DC. Fort intéressant, mais étonnant.

AC est-il toujours faux ? (en dehors du cas du forcing, bien sûr).

Algèbre de réalisabilité

Structure *du premier ordre* analogue à une *algèbre combinatoire*
mais plus compliquée, parce qu'on parle non seulement de *programmes*
mais aussi d'*environnements*, qui sont des *piles*. On a donc trois ensembles :
Λ ensemble des termes, Π ensemble des piles, $\Lambda \star \Pi$ ensemble des processus.
Combinateurs élémentaires : B, C, I, K, W, cc et d'autres éventuellement.
Ce sont des constantes de termes. On a aussi les fonctions suivantes :
application : de Λ^2 dans Λ, notée $(\xi,\eta) \mapsto (\xi)\eta$; souvent abrégée en $\xi\eta$;
push : de $\Lambda \times \Pi$ dans Π, notée $(\xi,\pi) \mapsto \xi \cdot \pi$;
continuation : de Π dans Λ, notée $\pi \mapsto \mathsf{k}_\pi$;
processus : de $\Lambda \times \Pi$ dans $\Lambda \star \Pi$, notée $(\xi,\pi) \mapsto \xi \star \pi$.
On a une relation de préordre sur $\Lambda \star \Pi$, appelée *l'exécution* et notée $\succ$.
On a aussi un segment final de $\Lambda \star \Pi$, noté $\perp\!\!\!\perp$.

Algèbre de réalisabilité

On a enfin une partie QP de Λ : l'ensemble des *quasi-preuves*
qui contient les combinateurs et est clos par application.

Les axiomes pour le préordre d'exécution $\succ$ constituent une *machine* :

$\xi\eta \star \pi \succ \xi \star \eta \cdot \pi$	(empile)
$\mathsf{I} \star \xi \cdot \pi \succ \xi \star \pi$	(ne fait rien)
$\mathsf{K} \star \xi \cdot \eta \cdot \pi \succ \xi \star \pi$	(efface)
$\mathsf{W} \star \xi \cdot \eta \cdot \pi \succ \xi \star \eta \cdot \eta \cdot \pi$	(recopie)
$\mathsf{C} \star \xi \cdot \eta \cdot \zeta \cdot \pi \succ \xi \star \zeta \cdot \eta \cdot \pi$	(échange)
$\mathsf{B} \star \xi \cdot \eta \cdot \zeta \cdot \pi \succ \xi \star \eta\zeta \cdot \pi$	(applique)
$\mathsf{cc} \star \xi \cdot \pi \succ \xi \star \mathsf{k}_\pi \cdot \pi$	(sauve la pile)
$\mathsf{k}_\pi \star \xi \cdot \omega \succ \xi \star \pi$	(restaure la pile)

On peut alors définir $\lambda x\, t$ de façon à avoir la *réduction faible de tête* :

$\lambda x\, t \star \xi \cdot \pi \succ t[\xi / x] \star \pi$.

Algèbre de réalisabilité

Cela nécessite une traduction du λ-calcul en logique combinatoire
qui n'est pas la traduction usuelle. Pour $t \in \Lambda$, on définit d'abord $\lambda' x\, t \in \Lambda$:

1. $\lambda' x\, t = (\mathsf{K})\, t$ si t ne contient pas x.
2. $\lambda' x\, x = \mathsf{I}$.
3. $\lambda x(t)\, u = (\mathsf{C}\, \lambda' x\, t)\, u$ si u ne contient pas x.
4. $\lambda x(t)\, x = t$ si t ne contient pas x.
5. $\lambda x(t)\, x = (\mathsf{W})\, \lambda' x\, t$ (si t contient x).
6. $\lambda' x(t)(u)\, v = \lambda' x(((\mathsf{B})\, t)\, u)\, v$ (si $u\, v$ contient x).

Et on pose finalement $\lambda x\, t = \lambda' x(\mathsf{I})\, t$.

C'est la traduction la plus simple qui donne la réduction faible de tête.

Voilà pourquoi on en revient aux combinateurs originaux de Curry.

Modèles de ZF

Quel intérêt de compliquer ainsi la structure d'algèbre combinatoire ?

En informatique :

On représente bien mieux les programmes, en tenant compte de l'*environnement*.

En logique :

Grâce aux environnements et à l'instruction cc, on obtient le *tiers exclu* ;

on se libère enfin du carcan de la *logique intuitionniste*.

Cela permet de réaliser les axiomes de la théorie des ensembles

et d'obtenir des modèles de ZF, si l'algèbre satisfait l'*axiome de cohérence* :

Pour toute quasi-preuve θ, il existe une pile π telle que $\theta \star \pi \notin \perp\!\!\!\perp$.

Le forcing est le cas particulier où il existe une quasi-preuve θ

qui est un *ou parallèle*, c'est-à-dire :

$$\theta \star \xi \cdot \eta \cdot \pi \succ \xi \star \pi \text{ et } \theta \star \xi \cdot \eta \cdot \pi \succ \eta \star \pi$$

La notion de programme disparaît alors complètement.

Réalisabilité

Pour chaque formule close F de ZF, on définit :

une *valeur de fausseté* $\|F\| \subset \mathbf{\Pi}$ et une *valeur de vérité* $|F| \subset \mathbf{\Lambda}$.

$\xi \in |F|$ est écrit $\xi \Vdash F$ et se lit : ξ réalise F (ξ force F dans le cas du forcing).

Définition de $\|F\|$ et $|F|$ par récurrence sur F : $\xi \in |F| \Leftrightarrow (\forall \pi \in \|F\|)\, \xi \star \pi \in \perp\!\!\!\perp$;

$\|\bot\| = \mathbf{\Pi}$ et donc $\xi \Vdash \bot \Leftrightarrow \xi \star \pi \in \perp\!\!\!\perp$ pour toute pile π.

$\|\forall x\, F[x]\| = \bigcup_x \|F[x]\|$ et donc $\xi \Vdash \forall x\, F[x] \Leftrightarrow \forall x (\xi \Vdash F[x])$;

$\|F \to G\| = \{\xi \cdot \pi\, ;\, \xi \Vdash F, \pi \in \|G\|\}$ et donc :

$\xi \Vdash F \to G \Rightarrow \forall \eta (\eta \Vdash F \Rightarrow \xi\eta \Vdash G)$; $\forall \eta (\eta \Vdash F \Rightarrow \xi\eta \Vdash G) \Rightarrow \lambda x \xi x \Vdash F \to G$.

La définition pour les formules atomiques $x \in y$, $x = y$ est plus compliquée.

Comme dans le cas particulier du forcing, on part d'un modèle de ZFC,

ou même ZF + V = L, appelé *modèle de base* et on obtient

un nouveau modèle de ZF_ε : le *modèle de réalisabilité* (ou de *forcing*).

Formules atomiques

Il y a trois types de formules atomiques dans la théorie ZF_ε : $x \not\in y$, $x \subset y$, $x \not\in y$.

ε est l'appartenance forte (non extensionnelle).

$\subset, \in$ sont les symboles extensionnels d'inclusion et d'appartenance faible.

On définit $\|x \not\in y\| = \{\pi \in \mathbf{\Pi}\,;\, (x,\pi) \in y\}$;

$\|x \subset y\| = \|\forall z(z \not\in y \to z \not\in x)\|$; $\|x \not\in y\| = \|\forall z(z \subset x, x \subset z \to z \not\in y)\|$

(par induction sur les rangs de x, y).

Les axiomes de ZF_ε sont réalisés par des programmes (quasi-preuves)

très simples comme I ou Y.

Les axiomes de ZF par des quasi-preuves beaucoup plus compliquées

qui proviennent de ce qu'ils sont *conséquences de* ZF_ε.

Il serait donc très compliqué d'écrire directement ces programmes.

C'est pourquoi on est obligé d'introduire la théorie ZF_ε.

Le cas dégénéré du forcing

On a $\mathbf{\Lambda} = \mathbf{\Pi} = \mathbf{\Lambda} \star \mathbf{\Pi}$ qui est un semi-treillis avec plus grand élément $\mathbf{1}$.

Tous les combinateurs et toutes les quasi-preuves sont égaux à $\mathbf{1}$.

Toutes les opérations binaires sont confondues avec l'opération $\wedge$ de semi-treillis.

L'ordre d'exécution $\succ$ sur $\mathbf{\Lambda} \star \mathbf{\Pi}$ est l'ordre inverse du semi-treillis.

$\perp\!\!\!\perp$ est un segment initial et $\mathbf{\Lambda} \setminus \perp\!\!\!\perp$ est *l'ensemble des conditions*.

Deux conditions p, q sont donc *compatibles* ssi $p \wedge q \notin \perp\!\!\!\perp$.

Utilité comparée

Le forcing sert pratiquement toujours à montrer l'indépendance
de divers axiomes, les cas le plus célèbres étant AC et CH.
Si on montre que le modèle de forcing ne satisfait pas une formule Φ
cela ne peut être utile que *si le modèle de base satisfait Φ.*
La situation est très différente pour la réalisabilité classique :
toute information sur le modèle de réalisabilité est exploitable
(malheureusement elles sont bien plus difficiles à obtenir).
Si on montre que le nouveau modèle satisfait une formule Ψ
(par exemple AC, CH ou leurs négations !) on a un programme qui réalise Ψ,
qu'elle soit vraie ou non dans le modèle initial.
Et de plus, comme pour le forcing, on a montré la consistance de Ψ
ce qui est intéressant si Ψ est fausse dans le modèle de base.

Fonctionnelles

Tous les symboles de fonction définis dans le modèle $\mathcal{M}$ de base
se prolongent au modèle de réalisabilité $\mathcal{N}$.
Exemple important : $⟙X = X \times \mathbf{\Pi}$ qui se comporte comme *un type*.
Toutes les formules de Horn vraies dans $\mathcal{M}$ sont réalisées par I :
Si $\mathcal{M} \models (\forall x_1 \in X_1) \ldots (\forall x_n \in X_n)(f_1(\vec{x}) = g_1(\vec{x}), \ldots, f_k(\vec{x}) = g_k(\vec{x}) \to f(\vec{x}) = g(\vec{x}))$
alors $\mathsf{I} \Vdash \forall x_1^{⟙X_1} \ldots \forall x_n^{⟙X_n}(f_1(\vec{x}) = g_1(\vec{x}), \ldots, f_k(\vec{x}) = g_k(\vec{x}) \to f(\vec{x}) = g(\vec{x}))$.
Le quantificateur $\forall x^{⟙X}$ est défini par $\| \forall x^{⟙X} F[x] \| = \bigcup_{x \in X} \| F[x] \|$.
Il équivaut à $(\forall x \, \varepsilon \, ⟙X)$.
Une fonction $f : X \to Y$ dans le modèle de base se prolonge donc en
$f : ⟙X \to ⟙Y$ dans le modèle de réalisabilité.

L'algèbre de Boole $\daleth 2$

Les opérations triviales $\wedge, \vee, \neg$ sur $\{0,1\}$ définissent l'algèbre de Boole $\daleth 2$.
Le modèle de réalisabilité $\mathcal{N}$ est booléen sur $\daleth 2$: on définit αx pour $\alpha \varepsilon \daleth 2$
en posant dans $\mathcal{M}$: $0x = \emptyset$ et $1x = x$.
On a $\alpha\emptyset = 0x = \emptyset$; $(\alpha \vee \beta)x = (\alpha x) \sqcup (\beta x)$; $\alpha(\beta x) = (\alpha \wedge \beta)x$
où le symbole binaire $\sqcup$ est le prolongement de $\cup$
(attention, il ne représente pas la réunion dans $\mathcal{N}$).
Pour chaque formule $F[\vec{x}]$ de ZF avec paramètres, on définit la fonctionnelle $\langle F[\vec{x}] \rangle$
à valeurs dans $\daleth 2$. Le modèle booléen $\mathcal{N}$ est alors une extension élémentaire de $\mathcal{M}$.
(Dans le cas du forcing, on a $\mathcal{N} = \mathcal{M}$ et $\daleth 2$ est trivial).
Tout ultrafitre $\mathcal{D}$ sur $\daleth 2$ donne donc une extension élémentaire
du modèle $\mathcal{M}$ de base, notée $\mathcal{M}_{\mathcal{D}}$.

L'ultrafiltre canonique

Or, il existe toujours au moins un ultrafitre sur $\beth 2$, qui a des propriétés remarquables :
Il donne une extension élémentaire $\mathscr{M}_{\mathscr{D}}$ de $\mathscr{M}$ qui est *bien fondée*.
$\mathscr{M}_{\mathscr{D}}$ est donc un sous-modèle transitif du modèle de ZF associé à $\mathscr{N}$
(nous noterons ce modèle $\mathscr{N}_{\in}$). On en déduit :
Les constructibles de $\mathscr{N}_{\in}$ forment une extension élémentaire de ceux de $\mathscr{M}$.
(Tout cela est trivial dans le cas du forcing où on a $\mathscr{M}_{\mathscr{D}} = \mathscr{M}$).
Conclusion intéressante (cf. "Utilité comparée" ci-dessus) :
Toute propriété vraie dans $L^{\mathscr{M}}$ est réalisée par un programme (quasi-preuve) ;
en particulier toute formule arithmétique ou même Σ^1_2 ou Π^1_2 vraie.

Le générique

- Dans le cas du forcing, le nouveau modèle est caractérisé par *le générique*

qui est une partie $G = \{(p, p)\ ;\ p \in \mathbf{\Pi}\}$ de l'ensemble des conditions $\mathbf{\Lambda} = \mathbf{\Pi}$.
Il fournit la vérité dans le nouveau modèle, car on a :

$$\mathcal{N} \models \Phi \Leftrightarrow (\exists p \in G)\left(\mathcal{M} \models (p \Vdash \Phi)\right)$$

Le générique G est toujours un nouvel ensemble ; *il n'est donc jamais constructible*.

- En réalisabilité classique on définit le générique $G = \{(\pi, \pi)\ ;\ \pi \in \mathbf{\Pi}\}$.

On a trivialement $\|\Phi\| = \|\forall \pi^{\neg\mathbf{\Pi}}(\langle \pi \in \|\Phi\| \rangle = 1 \hookrightarrow \pi \notin G)\|$ et donc :

$$\mathcal{N} \models \left(\neg\Phi \Leftrightarrow (\exists \pi\, \varepsilon\, G)(\langle \pi \in \|\Phi\| \rangle = 1)\right)$$

pour toute formule close Φ de ZF_ε.
G est maintenant une partie de $\neg\mathbf{\Pi}$; mais il ne suffit plus à fournir la vérité dans $\mathcal{N}$.
D'ailleurs, dans le modèle de fil (voir plus loin)
le générique G est un ensemble d'entiers constructible et même récursif.

Deux modèles typiques

Parmi les innombrables modèles de forcing, on peut citer deux exemples très simples mais typiques de la méthode :

- Les conditions sont les fonctions $p : n \to 2$ avec $n \in \mathbb{N}$ (arbre binaire).

Le générique est appelé *réel de Cohen*.

On obtient ainsi la consistance de l'axiome : *Il existe un réel non constructible.*

Une variante donne *l'indépendance de l'hypothèse du continu.*

- Les conditions sont les fonctions $p : \rho \to \mathbb{R}$ où ρ est un ordinal dénombrable.

Le générique est une surjection de $\aleph_1$ sur $\mathbb{R}$.

On obtient ainsi *la consistance de l'hypothèse du continu*

(méthode alternative à celle de Gödel).

Deux modèles typiques

En réalisabilité classique, il y a aussi deux modèles typiques
dont la définition est particulièrement simple. Malheureusement,
les modèles eux-mêmes ne le sont pas, et on est très loin de connaître
toutes leurs propriétés.
Et même, en ce qui concerne le second, on n'en connaît aucune !
Il ne peut donc pas (pour le moment) donner un résultat de consistance.
Néanmoins, il est fort utile pour analyser les programmes en λ_c-calcul.

Le modèle de fil

L'idée est d'avoir un modèle cohérent avec $\perp\!\!\!\perp$ maximal.

Soient θ_i une énumération des quasi-preuves et π_i une suite de piles vides.

L'exécution de $\theta_i \star \pi_i$ donne le *fil numéro i*, noté $\Phi_i \subset \Lambda \star \Pi$.

On définit $\perp\!\!\!\perp$ en posant $\Lambda \star \Pi \setminus \perp\!\!\!\perp = \bigcup_i \Phi_i$. Autrement dit :

$\perp\!\!\!\perp$ est formé des processus qui n'apparaissent dans aucun fil.

Les quasi-preuves sont les termes qui ne contiennent aucune constante de pile.

L'intérêt de ce modèle est qu'on peut ajouter plein de nouvelles instructions.

Problème ouvert : que se passe-t-il si on n'en ajoute aucune ?

Pour le moment, *on ne sait absolument rien* sur le modèle de ZF associé.

Satisfait-il ou non $V = L$?

Si oui, on a un programme pour l'axiome de constructibilité !

Le modèle de fil

La situation change complètement avec une instruction de *signature* ou d'*horloge*.

• Signature. $\sigma \star \xi \cdot \eta \cdot \pi \succ \xi \star \underline{n_\eta} \cdot \pi$ où $\eta \mapsto n_\eta$ est une *injection* de Λ dans $\mathbb{N}$.

Noter qu'on ne suppose pas cette injection récursive. C'est alors un *oracle*.

• Horloge. $\hbar \star \xi \cdot \eta \cdot \pi \succ \xi \star \underline{n} \cdot \pi$ où n est :

soit le nombre de pas d'exécution depuis *le boot* ;

soit le nombre de fois où l'instruction $\hbar$ s'est exécutée.

Remarque : Cette instruction est très simple à implémenter pratiquement.
Mais il faut la définir dans le cadre des algèbres de réalisabilité
ce qui est moins simple. La même méthode permettra de définir
les instructions de *lecture et écriture en mémoire*
couramment affublées du nom étrange d'*effets de bord*.

Horloge, signature et choix dépendant

L'implémentation usuelle de l'instruction d'horloge ne rentre pas
dans le cadre des algèbres de réalisabilité. Voici la définiton correcte :
$\hbar \star \xi \cdot \eta \cdot \pi \succ \xi \star \underline{n} \cdot \pi$ où l'entier n est déterminé comme suit :
Soit i le numéro du fil courant, qui est déterminé par π.
On exécute $\theta_i \star \pi_i$ jusqu'à arriver à $\hbar \star \xi \cdot \eta \cdot \pi$.
n est alors le nombre de pas, ou bien le nombre d'exécutions de $\hbar$.
Noter que le fil peut boucler, c'est-à-dire passer deux fois par le même état.
Seule compte la première fois.
Les instructions d'horloge et de signature réalisent une forme forte du CD :
l'axiome du choix non extensionnel (ACNE) énoncé ainsi dans ZF_ε.

$$\forall u \exists f \{\mathrm{Func}[f], (\forall x \varepsilon u) \forall y (F[x,y] \to F[x, f(x)])\}$$

avec $\mathrm{Func}[f] \equiv \forall x \forall y \forall y' \big((x,y) \varepsilon f, (x,y') \varepsilon f \to y = y'\big)$
et $F[x, f(x)] \equiv \exists z \{(x,z) \varepsilon f, F[x,z]\}$.

Signature et négation de AC

L'instruction de signature est couramment utilisée :
fonctions de hachage, MD5, SHA1, etc.
Mais on lui demande l'impossible : par exemple, SHA1 devrait être
une injection de $2^{2^{64}}$ dans 2^{160} ce qui paraît difficile.
Notre instruction σ n'a pas ce problème : injection de $\mathbf{\Lambda}$ dans $\mathbb{N}$.
Elle permet de réaliser *avec CD* des formes très fortes (et nouvelles) de ¬AC,
par exemple une suite de parties infinies de $\mathbb{R}$ de cardinal strictement décroissant
ce qu'on ne sait pas faire avec le forcing.

Lecture et écriture en mémoire

On définit ces instructions, dont l'horloge est un cas particulier :

- Ecriture du terme η à l'adresse a (entier) : $\spadesuit \star \underline{a} \cdot \eta \cdot \xi \cdot \pi \succ \xi \star \pi$
- Lecture à l'adresse a : $\clubsuit \star \underline{a} \cdot \xi \cdot \pi \succ \xi \star \eta \cdot \pi$

où le terme η est obtenu en réexécutant le fil courant Φ_i
jusqu'à arriver au processus courant $\clubsuit \star \underline{a} \cdot \xi \cdot \pi$.
η est alors le terme qui suit $\underline{a}$ dans la *dernière* apparition de $\spadesuit \star \underline{a} \cdot \ldots$
S'il n'y en a pas, l'adresse a n'a pas été affectée. Arrêt sur erreur.
Bien entendu, l'implémentation concrète est complètement différente.
Mais la théorie montre ce fait intéressant et peut-être inattendu :
L'utilisation de la RAM n'est possible logiquement qu'à l'aide du boot.
Par exemple, elle est impossible dans les algèbres qui suivent.

Les algèbres BBC

Leur nom provient d'un article de Berardi, Bezem et Coquand.

Les piles sont de la forme $\pi = t_0 \cdot \ldots \cdot t_{n-1} \cdot \pi_0$ avec $t_i \in \Lambda$; π_0 est la *pile vide*.

On ajoute deux combinateurs κ et A dont les règles d'exécution sont :

$\kappa \star \pi \succ$ (stop)

$\mathsf{A} \star \xi \cdot \pi \succ \xi \star \pi_0$ (efface la pile)

On définit k_π par récurrence :

$\mathsf{k}_{\pi_0} = \mathsf{A}$; $\mathsf{k}_{t \cdot \pi} = \ell_t \mathsf{k}_\pi$, avec $\ell_t = ((\mathsf{C})(\mathsf{B})\mathsf{CB)}t$ c'est-à-dire $\lambda k \lambda x (k)(x)\, t$.

On a donc $\mathsf{k}_\pi = (\ell_{t_0}) \ldots (\ell_{t_{n-1}})\mathsf{A}$ c'est-à-dire $\lambda x (\mathsf{A})(x)\, t_0 \ldots t_{n-1}$.

$\Lambda \star \Pi = \Lambda \times \Pi$; $\xi \star \pi \in \perp\!\!\!\perp \Leftrightarrow (\exists \pi' \in \Pi)(\xi \star \pi \succ \kappa \star \pi')$.

Les quasi-preuves sont les termes qui ne contiennent pas κ

ce qui assure la cohérence. Noter que, par exemple, $\mathsf{A} = \mathsf{k}_{\pi_0}$ est une quasi-preuve.

Les algèbres BBC

J'appelle cette algèbre BBC_0. Elle correspond assez précisément
à ce qu'on appelle la programmation fonctionnelle en appel par nom.
Elle permet de montrer très simplement qu'à toute preuve dans ZF
d'un énoncé arithmétique est associé un programme
qui a des propriétés remarquables.
Mais, malgré sa simplicité et son intérêt informatique,
on ne sait *absolument rien* sur le modèle de ZF associé.
Peut-être satisfait-il V = L, ce qui serait fort intéressant
puisqu'on aurait *un programme pour l'axiome de constructibilité*.
Une variante de ce modèle va d'ailleurs donner
un programme pour l'hypothèse du continu.

Les algèbres BBC

Dans leur article, Berardi, Bezem et Coquand cherchent un programme pour l'axiome du choix dépendant. Pour cela, ils définissent ce qu'ils appellent *un langage de programmation* qui n'en est évidemment pas un.
Il lui correspond une algèbre de réalisabilité, que j'appelle BBC.
La différence (importante) avec BBC_0 est la suivante :
Pour chaque suite $t_i (i \in \mathbb{N})$ de quasi-preuves, on a un combinateur noté $\bigwedge_i t_i$ avec la règle d'exécution :

$$\bigwedge_i t_i \star \underline{n} \cdot \pi \succ t_n \star \pi \qquad \text{(oracle)}$$

Cette algèbre a donc la puissance du continu.
Elle a beaucoup de points communs avec mon deuxième exemple de forcing.

L'hypothèse du continu

Nos trois mousquetaires Be., Be. & Co. montrent qu'un λ-terme remarquable,
appelé *bar récursion*, réalise l'axiome du choix dépendant.
Mais, en poursuivant l'analogie avec le deuxième modèle de forcing,
on peut montrer que ce modèle de réalisabilité satisfait aussi l'axiome :
Tous les réels sont constructibles et donc *l'hypothèse du continu*.
Il y a donc un λ_c-terme qui réalise l'hypothèse du continu.
Je n'ai pas cherché jusqu'ici à l'écrire explicitement.
Il est, bien sûr, beaucoup plus compliqué que la bar récursion.
Il serait extrêmement intéressant de comprendre son exécution,
mais cette question n'est même pas résolue pour la bar récursion.

Variantes de BBC

L'anneau de Boole $\mathbb{2}$ de ces modèles de réalisabilité est infini et même sans atome.
On obtient des algèbres très intéressantes en ajoutant un combinateur γ
et la règle suivante pour $\perp\!\!\!\perp$ (proche du *ou parallèle* qui est interdit) :
Si deux des trois processus $\xi \star \pi, \eta \star \pi, \zeta \star \pi$ sont dans $\perp\!\!\!\perp$, alors $\gamma \star \xi \cdot \eta \cdot \zeta \cdot \pi \in \perp\!\!\!\perp$.
$\mathbb{2}$ est alors l'anneau de Boole à 4 éléments.
Le modèle de réalisabilité $\mathcal{N}$ est booléen sur cette algèbre et donc :

$$\mathcal{N} = \mathcal{M}_0 \times \mathcal{M}_1 \text{ avec } \mathcal{M}_0, \mathcal{M}_1 \succ \mathcal{M} \text{ et } \mathcal{M}_0 \simeq L^{\mathcal{N}}$$

A comparer avec le cas du forcing où $\mathbb{2}$ est trivial et $\mathcal{N} = \mathcal{M}$.
Plus généralement, Guillaume Geoffroy a introduit des combinateurs
qui donnent pour $\mathbb{2}$ tous les anneaux de Boole finis.

Références

S. Berardi, M. Bezem, T. Coquand *On the computational content of the axiom of choice.* J. Symb. Log. 63 (1998) p. 600-622.

U. Berger, P. Oliva *Modified bar recursion and classical dependent choice.* Proc. Logic Colloquium 2001 - Springer (2005) p. 89-107.

E. Engeler *Algebras and combinators.* Algebra Universalis, vol. 13, 1 (1981) p. 389-392.

J.-L. Krivine *Realizability algebras II : new models of ZF + DC.* Logical Methods in Comp. Sc., vol. 8, 1:10 (2012) p. 1-28.

J.-L. Krivine *Realizability algebras III: some examples.* Math. Struct. in Comp. Sc. vol. 28, 1 (2018) p. 45-76.

T. Streicher *A classical realizability model arising from a stable model of untyped λ-calculus.* To appear (2013).