La réalisabilité classique et l'exemple du modèle des threads

Guillaume Geoffroy
Sous la direction d'Alexandre Miquel

Introduction au domaine de recherche

École Normale Supérieure 17 octobre 2014

Introduction

Qu'est-ce que la réalisabilité classique ?

La réalisabilité [1] établit une relation entre des formules logiques et des programmes informatiques [2]. Elle constitue donc un aspect de la correspondance preuves-programmes, qui fait correspondre d'un côté les preuves formelles avec les programmes informatiques, et de l'autre les formules logiques avec les types.

Plus précisément, la réalisabilité consiste à interpréter chaque formule logique ϕ comme une règle sur le comportement des programmes. On dit alors de chaque programme respectant cette règle qu'il *réalise* ϕ.

Pour l'intuition, une formule *réalisée* par au moins un programme est en un certain sens « vraie », au même titre que, en théorie des modèles, une formule est considérée comme vraie lorsque sa valeur de vérité vaut *vrai*. De plus, un programme qui réalise ϕ peut être vu comme une « preuve » de ϕ [3].

Sous sa forme initiale, telle que décrite par Stephen Cole Kleene dans [Kle45], la réalisabilité interprète des formules de l'arithmétique du premier ordre. Cette interprétation respecte les principes des mathématiques *constructives*. Par exemple, pour réaliser $\forall m \exists n\ \phi(m,n)$, un programme doit prendre en entrée un entier quelconque m, et renvoyer un entier n et un programme qui réalise $\phi(m,n)$ [4]. Comme l'illustre cet exemple, dans cette version de la réalisabilité, on peut en général extraire d'un réaliseur un contenu calculatoire intéressant [5].

Néanmoins, la réalisabilité de Kleene est, dès le départ, une sémantique de l'arithmétique de Heyting. Autrement-dit, de l'arithmétique du premier ordre *intuitionniste* [6]. Elle ne premet donc pas d'interpréter le raisonnement classique, qu'emploient les mathématiques habituelles. De fait, le principe du tiers exclus est incompatible avec les principes des mathématiques constructives.

La réalisabilité classique est une reformulation de la réalisabilité de Kleene pour la rendre compatible avec la logique classique. Le point clef est la découverte par Timothy Griffin (présentée dans [Gri90]) du fait que le raisonnement classique, du côté « logique », peut s'interpréter du côté « programmation » par des *opérateurs de contrôle*, tels que l'opérateur *call/cc* (call-with-current-continuation) du langage SCHEME. En réalisabilité classique, l'extraction d'un contenu calculatoire intéressant à partir des réaliseurs reste possible, mais seulement dans certain cas, et par des méthodes plus complexes qu'en réalisabilité intuitionniste.

Intérêts de la réalisabilité classique

La réalisabilité fournit une sémantique à la logique intuitionniste. L'un des intérêts de disposer d'une sémantique est que'elle permet de prouver des résultats d'indépendance. Pour la logique classique, des sémantiques existaient déjà avant la réalisabilité classique. L'exemple le plus simple est donné par la théorie des modèles : cette sémantique, dite de Tarski, consiste à attribuer à chaque formule une valeur de vérité

1. Qu'elle soit intuitionniste ou classique.
2. Modélisés, par exemple, par des machines de Turing, ou encore par des termes du λ-calcul.
3. En réalité, une preuve formelle de ϕ contient plus d'informations qu'un réaliseur. De fait, de façon calculable, on peut vérifier si un objet donné est bien une preuve correcte de ϕ, mais pas si un programme donné réalise ϕ. En effet, il est impossible de prévoir à l'avance le comportement d'un programme.
4. Il s'agit de la propriété de Church.
5. Ici, d'un réaliseur de $\forall m \exists n\ \phi(m,n)$, on extrait la fonction qui calcule n en fonction de m.
6. C'est-à-dire que toutes les formules démontrables dans l'arithmétique de Heyting sont réalisées. En particuiler, toutes les tautologies intuitionnistes sont réalisées.

vrai ou *faux*. Le forcing, introduit par Paul Joseph Cohen dans [Coh63] pour montrer l'indépendance de l'hypothèse du continu dans ZFC, fournit un autre exemple, très riche. Dès lors, que peut apporter la réalisabilité à la logique classique ? Autrement-dit, quels avantages la réalisabilité possède-t-elle sur le forcing, qui justifient que l'on l'adapte à la logique classique ?

Deux avantages se détachent. Ils proviennent tous les deux du fait que la réalisabilité interprète les preuves, alors que le forcing, comme la théorie des modèles, n'interprète que la prouvabilité (ou la « vérité »).

Premièrement, grâce à la réalisabilité classique, on peut extraire un contenu calculatoire des preuves classiques. Par exemple, dans [Miq11a], Alexandre Miquel décrit des techniques d'extraction de témoin pour des formules existentielles. En particulier, il est possible, grâce à la réalisabilité classique, de donner une interprétation calculatoire des preuves par forcing : une méthode générale est décrite dans [Miq11b], et appliquée à un exemple concret dans [Rie14]. De façon remarquable, le forcing est interprété par des notions utilisées depuis lontemps en informatique pratique : la distinction entre exécution en mode protégé et en mode réel, et l'emploi d'appels systèmes pour passer de l'un à l'autre. Par exemple, dans l'exemple décrit dans [Rie14], le forcing joue le rôle d'un ordonnanceur [1].

Deuxièmement, la réalisabilité fait apparaître une subtilité que le forcing ignore : prouver une conjonction et prouver une quantification universelle sont deux activités différentes. En effet, pour prouver une conjonction, il suffit de prouver chaque facteur séparément, alors que pour prouver une quantification universelle, il faut fournir une unique preuve qui convienne dans tous les cas. Le forcing ne tient pas compte de cette distinction : la conjonction et la quantification universelle sont toutes les deux interprétées par une intersection [2]. En réalisabilité, la quantifiaction universelle est toujours interprétée par une intersection, mais la conjonction est interprétée par un produit cartésien. Pour cette raison, la réalisabilité classique permet d'obtenir des modèles que l'on ne pourrait pas obtenir par forcing. C'est ainsi que, dans [Kri12a], Jean Louis Krivine définit le modèle des threads. Il s'agit d'un modèle particulier de la théorie des ensembles, aux propriétés bien étranges : par exemple, l'ensemble des cardinaux en dessous de $\mathbb{R}$, muni de son ordre naturel, contient une copie de $\mathbb{Q}$ [3]. Pour résumer :

Tarski, forcing	**Réalisabilité**
Interprètent la prouvabilité	Interprète les preuves (contenu calculatoire)
$\wedge = \forall = \cap$	$\wedge = \times$ $\forall = \cap$

Sommaire

La première section décrit le langage que l'on utilise ici pour formaliser la notion de « programme » : le λ_c-*calcul*.

La deuxième présente la théorie de la réalisabilité classique en arithmétique du premier ordre. On y définit la notion de *modèle de réalisabilité*, l'*algèbre de Boole caractéristique* $\nabla 2$, et le *modèle des threads*, dont on donne quelques propriétés.

1. En Anglais, *scheduler*.
2. Finie dans le cas de la conjonction, infinie dans le cas de la quantification universelle.
3. En particulier, l'axiome du choix et l'hypothèse du continu sont tous les deux archi-faux !

La troisième section donne un résultat d'Alexandre Miquel sur la réalisation des clauses de Horn, ainsi qu'une extension aux clauses quelconques, que j'ai obtenue pendant mon stage.

La dernière section évoque l'adaptation de la réalisabilité classique à la théorie des ensembles, et donne en particulier les propriétés les plus frappantes du modèle des threads.

Table des matières

I Le langage des programmes : le λ_c-calcul

Il faut choisir une façon de formaliser la notion intuitive de « programme informatique ». Ici, on emploiera le langage du λ_c-calcul, que décrit la présente section.

I.1 Syntaxe du λ_c-calcul

Le λ_c-calcul distigue deux catégories syntaxiques : les λ_c-*termes* (notation : $t, u, v, ...$) qui représentent les programmes, et les piles (notation : $\pi, \pi', \varpi, ...$), qui représentent les contextes d'évaluation. Il est nécessaire de représenter les contextes d'évaluation, car, comme annoncé dans l'introduction, le raisonnement classique est interprété par des opérateurs de contrôle, c'est-à-dire des instructions qui manipulent le contexte. Les termes sont des termes du λ-calcul pur [1] enrichis par une instruction supplémentaire, l'opérateur de contôle cc (call with current continuation), et par une *constante de continuation* k_π pour chaque pile π. Les piles sont des listes de termes clos terminées par un *fond de pile* α_n :

1. Voir par exemple le cours de *Lambda-calcul et langages fonctionnels* de Jean Goubault-Larrecq, disponible à l'adresse http ://www.lsv.ens-cachan.fr/ goubault/Lambda/lambda.pdf, pour une introduction au λ-calcul.

$$
\begin{array}{llcll}
(Termes) & t & := & x & (x \text{ variable}) \\
 & & | & \lambda x.\, t & \\
 & & | & t\, u & \\
 & & | & \mathrm{cc} & \\
 & & | & k_\pi & (\pi \text{ pile}) \\
(Piles) & \pi & := & t.\pi & (t \text{ terme clos}) \\
 & & | & \alpha_n & (n \text{ entier naturel})
\end{array}
$$

I.2 Évaluation

Les λ_c-termes sont destinés à s'évaluer en face d'une pile : on appelle processus tout couple $t \star \pi$ formé d'un terme clos t et d'une pile π [1]. Dans ce cadre, π représente la liste de tous les *arguments* passés au programme t [2]. Les processus sont *évalués* selon les règles suivantes :

$$
\begin{array}{llcll}
\text{(Push)} & t\, u \star \pi & \succ_s & t \star u.\pi & \\
\text{(Grab)} & \lambda x.\, t \star u.\pi & \succ_s & t\{x := u\} \star \pi & \text{(Substitution de } u \text{ pour } x \text{ dans } t) \\
\text{(Save)} & \mathrm{cc} \star t.\pi & \succ_s & t \star k_\pi.\pi & \\
\text{(Restore)} & k_\pi \star t.\varpi & \succ_s & t \star \pi &
\end{array}
$$

La relation $\succ_s$ représente l'évaluation en une étape, et la relation $\succ$ représente l'évaluation en un nombre d'étapes quelconque, éventuellement nul.

Les règles (Push) et (Grab) effectuent la β-réduction faible de tête [3] et interprètent le raisonnement *intuitionniste*. Les règles (Save) et (Restore) permettent au programme de manipuler son contexte d'exécution, et interprètent le raisonnement *classique*. Ceci sera détaillé à la section II.

On note Λ l'ensemble des λ_c-termes clos, Π l'ensemble des piles, et $\Lambda \star \Pi$ l'ensemble des processus.

À titre d'exemple, on peut définir une classe de termes particuliers : les entiers de Krivine. L'entier de Krivine $\underline{0}$ est le terme $\lambda f.\lambda x.\, x$, le terme « successeur » σ est défini comme $\lambda n.\lambda f.\lambda x.\, f\,(n\, f\, x)$, et pour tout entier naturel n, l'entier de Krivine $\underline{n}$ est le terme $\sigma^n\, \underline{0}$. Ainsi, l'entier de Krivine $\underline{n}$ représente un programme qui prend en entrée un autre programme (vu comme une fonction) et le compose n fois [4]. Les entiers de Krivine servent à coder les entiers par des λ_c-termes.

Il est possible d'étendre le λ_c-calcul avec d'autres instructions. En particulier, l'instruction *quote* du LISP [5] permet d'interpréter l'axiome des choix dépendants [6]. Le cadre des algèbres de réalisabilité, défini dans [Kri11] permet d'avoir un jeu d'instructions « ouvert », en axiomatisant l'évaluation plutôt que de la définir.

II Modèles de réalisabilité de l'arithmétique du premier ordre

On va présenter la théorie de la réalisabilité dans le cadre de l'arithmétique du premier ordre. C'est-à-dire que l'on va définir comment interpréter les formules du langage de l'arithmétique du premier ordre

1. Ainsi, $t \star \pi$ n'est rien d'autre qu'une notation pour (t, π).
2. C'est-à-dire que le processus $t \star u_1 ... u_m.\alpha_n$ représente l'exécution du programme t avec les arguments u_1, ..., u_m.
3. Voir [Kri09]
4. Cet encodage des entiers se comporte donc comme l'encodage de Church, et il lui est β-équivalent. Néanmoins, en réalisabilité classique, on préfère l'encodage de Krivine à celui de Church pour des raisons techniques.
5. L'instruction *quote* prend en entrée un programme et renvoie son code-source.
6. Voir par exemple [Kri12a].

dans des *modèles de réalisabilité classique* (de l'arithmétique du premier ordre).

Remarque 1 (Nature des modèles de réalisabilité classique)**.** Les modèles de réalisabilité classique ne sont pas des modèles de Tarski, c'est-à-dire des modèles au sens usuel. En particulier, pour un modèle de réalisabilité donné, on n'a pas accès, depuis l'extérieur, à l'ensemble de ses *individus*. On a seulement accès à sa *théorie*, qui n'est d'ailleurs pas forcément complète, et à la fraction de ses individus constituée des *entiers standard*. Ce point marque une différence majeure entre les modèles de réalisabilité classique et aussi bien les modèles de Tarski que les modèles de forcing. Il constitue également l'une des difficultés de la réalisailité classique [1].

Tout d'abord, précisons le langage des formules.

II.1 Le langage de l'arithmétique du premier ordre

On considère un langage du premier ordre [2] $\mathcal{L}$ dont la signature σ contient au moins un symbole de constante n pour chaque entier naturel n, un symbole de fonction k-aire f pour chaque fonction k-aire *récursive primitive* f (et en particulier un symbole s pour la fonction « successeur » s), le symoble de différence « $\neq$ » (symbole de relation binaire) et un symbole de relation unaire $N(x)$, qui signifie « x est une entier ». Le langage $\mathcal{L}$ peut également contenir d'autres symboles de relation et d'autres symboles de fonction.

Le symbole N est d'une importance capitale. En effet, contrairement aux modèles de Tarski, les modèle de réalisabilité peuvent contenir des individus qui ne sont *pas* des entiers, pas même des entiers non standard.

Pour les symboles logiques, on se restreint à l'implication ($\to$), la quantification universelle ($\forall$), le faux ($\bot$) et le vrai ($\top$), qui forment un jeu de connecteurs complet. Les autres connecteurs sont définis de la manière suivante :

$$\begin{aligned}
\phi \wedge \psi &\equiv (\phi \to \psi \to \bot) \to \bot \\
\phi \vee \psi &\equiv (\phi \to \bot) \to (\psi \to \bot) \to \bot \\
\neg\phi &\equiv \phi \to \bot \\
\phi \leftrightarrow \psi &\equiv (\phi \to \psi) \wedge (\psi \to \phi) \\
\exists x\, \phi &\equiv (\forall x\ \ \phi \to \bot) \to \bot
\end{aligned}$$

Le symbole d'égalité « = » est défini par négation : $(\alpha = \beta) \equiv (\alpha \neq \beta) \to \bot$.

Il y a une raison à ces détours surprenants. En effet, les quatre connecteurs $\to$, $\forall$, $\bot$ et $\top$ ont des interprétations naturelles en réalisabilité classique, plus que les autres connecteurs. De même, l'interprétation de la différence est plus simple que celle de l'égalité, d'où l'idée de prendre la première comme notion primitive au lieu de la seconde.

On peut désigner par $\phi(x_1, ..., x_n)$ une formule sans variable libre autre que x_1, ..., x_n, et alors on note $\phi(\alpha_1, ..., \alpha_n)$ la substitution de chaque α_i pour le x_i correspondant dans ϕ. On emploie des conventions analogues pour les termes du premier ordre.

1. Le même phénomène est aussi présent en réalisabilité intuitionniste.
2. On considère des langages non égalitaires, c'est-à-dire que le symbole « = » ne fait pas a priori partie du langage. S'il est présent, il fait partie de la signature.

II.2 L'architecture des modèles de réalisabilité classique

Un modèle de réalisabilité classique est principalement défini par un paramètre : son *pôle* $\perp\!\!\!\perp$. Il s'agit d'un sous-ensemble de $\Lambda \star \Pi$ qui est *saturé*, c'est-à-dire que pour toute paire de processus p et p', si p' est dans $\perp\!\!\!\perp$ et que $p \succ p'$, alors p est dans $\perp\!\!\!\perp$.

La notion centrale que l'on cherche à définir est celle de *réalisation* d'une *formule* par un λ_c*-terme*. Intuitivement, dire que t réalise ϕ (notation $t \Vdash \phi$) signifie que t est *garant* de ϕ. Pour fixer les idées, on va d'abord définir formellement cette notion. Puis, dans la sous-section suivante, on expliquera les intuitions derrière cette définition.

Comme dans les modèles de Tarski, les termes du premier ordre sont inteprétés par des entiers : à chaque terme clos α, on associe une *valeur* α^*. Il suffit pour cela d'interpréter chaque symbole de constante c par un entier c^* et chaque symbole de fonction k-aire f par une fonction $f^* : \mathbb{N}^k \to \mathbb{N}$. Bien entendu, pour chaque entier n, la constante n s'interprète par l'entier n, et pour chaque fonction récursive primitive f, le symbole f s'interprète par la fonction f. Si la signature contient d'autres symboles de fonction, leur interprétation fait partie des paramètres du modèle de réalisabilité classique.

En revanche, là où, dans les modèles de Tarski, les formules sont interprétées par *vrai* ou par *faux*, dans les modèles de réalisabilité classique, les formules sont interprétées par des *valeurs de fausseté* et des *valeurs de vérité*.

Une valeur de fausseté est simplement une partie de Π. Pour chaque valeur de fausseté S, on définit son *orthogonal* $S^{\perp\!\!\!\perp} \equiv \{ t \in \Lambda : \forall \pi \in S \; t \star \pi \in \perp\!\!\!\perp \}$. Une valeur de vérité est une partie de Λ qui est l'othogonal d'une certaine valeur de fausseté. Pour chaque partie T de Λ, on définit également son orthogonal $T^{\perp\!\!\!\perp} \equiv \{ \pi \in \Pi : \forall t \in T \; t \star \pi \in \perp\!\!\!\perp \}$. Ainsi, les valeurs de vérités sont les parties de Λ qui sont *saturées*, c'est-à-dire égales à leur bi-orthogonal. On verra en II.3 pourquoi l'on s'intéresse aux parties saturées en particulier.

Ainsi, à chaque formule close ϕ, on associe une valeur de fausseté $\|\phi\| \in \mathcal{P}(\Pi)$, définie par induction sur la structure de ϕ, et une valeur de vérité $|\phi| \equiv \|\phi\|^{\perp\!\!\!\perp}$. Par conséquent, chaque symbole de relation k-aire R doit être interprété par une fonction $R^* : \mathbb{N}^k \to \mathcal{P}(\Pi)$. Comme on va le voir, les interprétations des symboles $\neq$ et N sont imposées. Ensuite, il faut supposer que l'on dispose pour chaque fonction k-aire $Z : \mathbb{N}^k \to \mathcal{P}(\Pi)$ d'un symbole de relation k-aire Z, dont l'interprétation est la fonction Z. S'il y a d'autres symboles de relation, leur interprétation fait partie des paramètres du modèle de réalisabilité classique.

Définition 2. Valeur de fausseté d'une formule

$$
\begin{aligned}
\|\top\| &\equiv \emptyset \\
\|\bot\| &\equiv \Pi \\
\|\forall x \;\; \phi(x)\| &\equiv \bigcup_{n \in \mathbb{N}} \|\phi(n)\| \\
\|\phi \to \psi\| &\equiv \{ t.\pi : t \in |\phi|, \pi \in \|\psi\| \} \\
\|R(\alpha_1, ..., \alpha_n)\| &\equiv R^*({\alpha_1}^*, ..., {\alpha_n}^*)
\end{aligned}
$$

En particulier,

$$
\begin{aligned}
\|\alpha \neq \beta\| &\equiv \begin{cases} \|\top\| \text{ si } \alpha^* \neq \beta^* \\ \|\bot\| \text{ si } \alpha^* = \beta^* \end{cases} \\
\|N(\alpha)\| &\equiv \bigcup_{Z : \mathbb{N} \to \mathcal{P}(\Pi)} \|(\forall x \;\; Z(x) \to Z(s(x))) \to Z(0) \to Z(\alpha)\|
\end{aligned}
$$

On expliquera dans la sous-section suivante l'intuition derrière l'interprétation de chaque symbole logique.

Définition 3. On dit qu'un λ_c-terme *t réalise* une formule close ϕ si $t \in |\phi|$. On le note $t \Vdash \phi$.

Rappelons qu'intuitivement, dire que t réalise ϕ signifie t est garant de ϕ.

Il faut mainteant définir quelles formules sont « vraies » dans le modèle de réalisabilité. Le terme exact est *réalisées*. Le critère « avoir au moins un réaliseur » ne convient pas. En effet, dès que le pôle $\perp\!\!\!\perp$ est non-vide, toute formule close, y compris $\bot$, a au moins un réaliseur. Ceci est d'ailleurs une des différences fondamentale entre réalisabilité intuitonniste et réalisabilité classique : on ne peut pas accepter n'importe quel réaliseur comme garant [1].

On dit qu'un terme clos est une *quasi-preuve* s'il ne contient aucune constante de continuation k_π. On dit qu'une formule ϕ est *réalisée* [2] si elle est réalisée par au moins une quasi-preuve.

Définition 4. La *théorie du modèle de réalisabilité* est la théorie du premier ordre $\mathcal{T}^{\perp\!\!\!\perp}$ constituée de toutes les formules closes qui sont *réalisées*.

II.3 Intuitions

L'intuition derrière cette construction est la suivante : l'intéraction entre une λ_c-terme et une pile (c'est-à-dire l'évaluation) représente un *dialogue argumentatif*. Pour chaque formule ϕ, sa valeur de fausseté $\|\phi\|$ représente l'ensemble des *objections* que l'on peut lui opposer. Un λ_c-terme *t réfute* une objection π lorsque $t \star \pi$ est dans $\perp\!\!\!\perp$. Ainsi, *t réalise* ϕ lorsqu'il réfute toutes les objections que l'on peut opposer à ϕ.

Étant donné une formule ϕ, son interprétation directe est sa valeur de fausseté $\|\phi\|$. Sa valeur de vérité $|\phi|$, qui est l'objet que l'on visait à construire, n'est définie qu'indirectement, à partir de la valeur de fausseté. Ce détour a pour but de s'assurer que $|\phi|$ est bien égal à son bi-orthogonal, puisque $|\phi|$ est déjà l'orthogonal de $\|\phi\|$.

Pourquoi faut-il que les valeurs de vérité soient égales à leur bi-orthogonal ? Reprenons l'analogie de l'argumentation. L'opération « orthogonal » représente un changement de point de vue entre le *défenseur*, qui cherche à prouver ϕ, et son *opposant*, qui cherche à la réfuter. D'un point de vue logique, ce changement de point de vue se traduit par une négation, puisque l'opposant défend en quelque sorte $\neg\phi$. De sorte que le bi-orthogonal représente la double négation. Or on cherche à interpréter la logique classique, où le raisonnement par l'absurde est valide. C'est pourquoi la valeur de vérité d'une formule doit être égale à son bi-orthogonal.

On peut comprendre l'interprétation des symboles logiques de la façon suivante. N'importe quel terme réalise $\top$. Un terme qui réalise $\bot$ réalise n'importe quelle autre formule [3]. Un terme réalise $\forall x\ \phi(x)$ lorsqu'il réalise $\phi(x)$ pour toutes les valeurs possibles de x. Enfin, un terme réalise $\phi \to \psi$ s'il prend en entrée un réaliseur de ϕ [4] et fournit un réaliseur de ψ.

L'interprétation du symbole $\neq$ n'a rien de choquant : si m est effectivement différent de m, alors la valeur de vérité de $m \neq n$ est celle de $\top$, qui représente le *vrai*, et sinon, c'est celle de $\bot$, qui représente le *faux*. Rappelons toutefois que le symbole $=$ est défini par $(\alpha = \beta) \equiv (\alpha \neq \beta) \to \bot$. Par conséquent, si m est effectivement égal à n, alors la valeur de vérité de $m = n$ est celle de $\bot \to \bot$, qui est une

1. Certains font des chèques en bois !
2. Intuitivement, qu'elle est vraie dans le modèle de réalisabilité.
3. C'est le principe de l'explosion : *ex falso quod libet*.
4. Rappelons que la pile représente la liste des arguments passés au programme.

façon *différente* de représenter le *vrai*, et sinon, c'est celle de $\top \to \bot$, qui est une façon *différente* de représenter le *faux*. Autrement-dit, on distigue deux façons de représenter le couple *vrai / faux*, selon que l'on évalue une formule *positive* (c'est-à-dire de la forme $\alpha = \beta$) ou *négative* (c'est-à-dire de la forme $\alpha \neq \beta$). Cette distinction est d'une importance capitale : elle permet d'expliquer une bonne partie des propriétés étranges des modèles de réalisabilité.

Passons maintenant au symbole N. Comme indiqué plus haut, il peut exister dans le modèle de réalisabilité des individus *non-entiers*. C'est une façon de dire que même si $\forall x\ \ \phi(x) \to \phi(s(x))$ et $\phi(0)$ sont réalisées, il n'y a aucune raison pour que $\forall x\ \ \phi(x)$ le soit. En effet, si t réalise $\forall x\ \ \phi(x) \to \phi(s(x))$ et u réalise $\phi(0)$, cela veut dire que pour tout n, on a un terme $t^n\, u$ qui réalise $\phi(n)$, mais a priori, on ne dispose pas d'un unique terme qui réalise $\phi(n)$ pour tout n. Ainsi, le schéma d'axiomes de récurrence, sous sa forme absolue $\forall y\ \ (\forall x\ \ \phi(x) \to \phi(s(x))) \to \phi(0) \to \phi(y)$, n'est a priori pas réalisé. On cherche donc simplement à réaliser sa version relativisée aux entiers, c'est-à-dire au prédicat $N : \forall y\ \ N(y) \to (\forall x\ \ N(x) \to \phi(x) \to \phi(s(x))) \to \phi(0) \to \phi(y)$. L'interprétation de N est définie précisément de manière à permettre ceci.

Remarque 5. Pour tout entier n, l'exemple canonique de réaliseur de $N(n)$ est l'entier de Krivine $\underline{n}$ [1].

II.4 Résultats d'adéquation

Malgré leurs propriétés parfois étranges, dont on donnera quelques exemples plus loin, les modèles de réalisabilité sont bien des modèles acceptables de l'arithmétique, comme l'illustrent les résultats suivants.

Theorème 6 (Théorème d'adéquation)**.** *Pour toutes formules $\phi_1, ..., \phi_n, \psi$ telles que $\phi_1, ..., \phi_n \vdash \psi$, il existe un λ_c-terme t tel que pour tous $u_1, ..., u_n$ réalisant respectivement $\phi_1, ..., \phi_n$, $t\{x_1 := u_1, ..., x_n := u_n\}$ réalise ϕ.*

Autrement-dit, toutes les règles de déduction du raisonnement classique sont admissibles [2].

Démonstration. Donnons l'exemple des règles du *modus ponens* et du raisonnement par l'absurde.

Pour le modus ponens, si t réalise $\phi \to \psi$ et u réalise ϕ, alors pour toute pile $\pi \in \|\psi\|$, $t\, u \star \pi \succ t \star u.\pi \in \bot$, donc $t\, u \star \pi \in \Vdash$. Autrement-dit, $t\, u$ réalise ψ. Ceci est conforme à l'intuition que l'on avait donné de la définition de $\|\phi \to \psi\|$.

Pour le raisonnement par l'absurde, on peut montrer que pour toute formule close ψ, l'instruction cc réalise la formule $((\phi \to \bot) \to \bot) \to \phi$ [3]. En effet, soient $t \in |(\phi \to \bot) \to \bot|$ et $\pi \in \|\phi\|$. Alors $\mathrm{cc} \star t.\pi \succ t \star k_\pi.\pi$. Or k_π réalise $\phi \to \bot$ [4], car pour tout $u \in |\phi|$, pour toute pile π', $k_\pi \star u.\pi' \succ u \star \pi \in \Vdash$. Donc $\mathrm{cc} \star t.\pi \in \Vdash$. □

Le raisonnement purement intuitionniste est interprété par la partie « λ-calcul pur » du λ_c-calcul, constituée des constructions « abstraction » et « application » et des règles (Push) et (Grab). Le raisonnement classique est interprété par la constante cc (qui réalise le raisonnement par l'absurde : $((\phi \to \bot) \to \bot) \to \phi$) et par les règles correspondantes (Save) et (Restore).

1. Le n^{e} entier de Church fonctionne tout aussi bien.
2. En réalité, ce sont même les règles de typage des termes par déduction naturelle qui sont admissibles. Voir par exemple [Miq11b].
3. En réalité, cc réalise même la *Loi de Pierce* : pour toutes formules ϕ et ψ, cc réalise $((\phi \to \psi) \to \phi) \to \phi$.
4. Et, puisque k_π est le terme qui représente la pile $\pi \in \|\phi\|$, on retrouve cette idée que passer d'une valeur de fausseté à son orthogonal représente une négation.

Remarque 7. En réalité, le fait que $\neg\neg\phi$ soit équivalent à ϕ provient en premier lieu du fait que les valeurs de vérité sont égales à leur bi-orthogonal (qui représente la double négation). Le terme cc permet simplement d'*internaliser* ce fait.

Theorème 8. *Tous les axiomes de l'arithmétique de Peano du premier ordre, relativisés à N, sont réalisés.*

Ainsi, $\mathcal{T}^{\perp\!\!\!\perp}$ est toujours une extension de l'arithmétique de Peano.

On s'intéresse bien sûr aux cas où $\mathcal{T}^{\perp\!\!\!\perp}$ est cohérente, c'est-à-dire au cas où $\perp$ n'est pas réalisé. On dit alors que le modèle de réalisabilité est cohérent. On peut le voir directement sur $\perp\!\!\!\perp$:

Theorème 9. *Le modèle de réalisabilité est cohérent si et seulement si pour toute quasi-preuve t, il existe au moins une pile π telle que $t \star \pi \notin \perp\!\!\!\perp$. On dit d'un tel $\perp\!\!\!\perp$ qu'il est cohérent.*

Pour obtenir des modèles de réalisabilité classique, il faut donc chercher des ensembles saturés cohérents, pour servir de pôle.

II.5 Le modèle dégénéré

L'exemple le plus simple d'ensemble saturé et cohérent est l'ensemble vide. Lorsque l'on choisit le *pôle vide*, le modèle de réalisabilité n'est autre que le modèle standard de l'arithmétique, c'est-à-dire $\mathbb{N}$:

Theorème 10. *Supposons $\perp\!\!\!\perp = \emptyset$. Alors, pour toute formule close ϕ, ϕ est réalisée si et seulement si elle est vraie (dans $\mathbb{N}$).*

II.6 Le modèle des *threads*

Pour obtenir un modèle nouveau, on cherche donc à s'éloigner le plus possible du pôle vide : on part de l'ensemble $\Lambda \star \Pi$, et l'on essaye d'en enlever juste assez pour le rendre cohérent.

Pour ce faire, il faut choisir pour chaque quasi-preuve t une pile π, et enlever $t\star\pi$ [1] et tous les processus qui suivent dans l'évaluation [2].

On considère donc une énumération $(t_n)_{n\in\mathbb{N}}$ des quasi-preuves. Pour tout n, on appelle n^{e} thread l'ensemble de processus $\Theta_n \equiv \{ u \star \pi \in \Lambda \star \Pi : t_n \star \alpha_n \succ u \star \pi \}$. Ainsi, le n^{e} thread représente l'histoire de l'évaluation du processus $t_n \star \alpha_n$, c'est-à-dire la suite d'états successifs que l'on atteint en l'évaluant : $t_n \star \alpha_n \succ_{\text{s}} u \star \pi \succ_{\text{s}} u' \star \pi' \succ_{\text{s}} \ldots \succ_{\text{s}} u'' \star \pi'' \succ_{\text{s}} \ldots$.

Le *pôle des threads* est l'ensemble $\perp\!\!\!\perp_\theta \equiv \Lambda\star\Pi \setminus \bigcup_{n\in\mathbb{N}} \Theta_n$, et le modèle de réalisabilité associé s'appelle le *modèle des threads*.

Le point important, qui donne ses propriétés étranges au modèle des threads, c'est que l'évaluation est presque *bidéterministe* [3]. Plus précisément, c'est la combinaison d'une part de la structure « linéaire » des threads [4], qui provient du *déterminisme* de l'évaluation, et d'autre part du fait que deux threads ne peuvent jamais se rejoindre [5].

1. Pour la cohérence.
2. Pour la saturation.
3. C'est à dire à la fois déterministe et réversible. Bien entendu, elle n'est pas vraiment réversible, mais tout se passe presque comme si elle l'était.
4. Pour être exact, chaque thread prend l'une des trois formes suivantes :
   - Une ligne infinie : le programme ne termine jamais mais ne boucle pas non plus,
   - Une ligne finie : le programme termine ou « plante » en un temps fini,
   - Une ligne finie terminée par une boucle : le programme répète les mêmes opérations à l'infini.
5. En effet, le fond de pile d'un processus ne peut pas changer. Or ce fond de pile indique le numéro du thread.

II.7 L'algèbre de Boole caractéristique $\nabla 2$

L'un des objets étonnants des modèles de réalisabilité, et en particulier du modèle des threads, est leur algèbre de Boole carctéristique $\nabla 2$. Pour la définir, on ajoute au langage un symole de relation unaire $\nabla 2$, que l'on interprète de la façon suivante :

Définition 11 (Algèbre de Boole caractérsitique $\nabla 2$)**.** Pour tout entier n, on pose :

$$\nabla 2^*(n) = \|\nabla 2(n)\| \equiv \begin{cases} \|\bot \to \bot\| \text{ si } n = 0 \text{ ou } n = 1 \\ \|\top \to \bot\| \text{ sinon} \end{cases} .$$

Cette construction fonctionne dans n'importe quel modèle de réalisabilité, mais elle a un intérêt particulier dans le modèle des threads.

L'ensemble $\{0, 1\}$ muni des opérations $\wedge$, $\vee$ et $\neg$ forme une algèbre de Boole. Chacune de ces opérations donne un symbole de fonction [1], et a donc un pendant dans le modèle de réalisabilité.

Theorème 12. *Dans le modèle de réalisabilité,* $\nabla 2$ *muni des opérations* $\wedge$*,* $\vee$ *et* $\neg$ *est une algèbre de Boole, d'élément minimum* 0 *et d'élément maximum* 1*. C'est-à-dire que tous les axiomes des algèbre de Boole, exprimés avec les symboles* 0*,* 1*,* $\wedge$*,* $\vee$ *et* $\neg$*, et relativisés à* $\nabla 2$*, sont réalisés.*

Contrairement à ce que l'on pourrait croire, $\nabla 2$ peut contenir d'autres éléments que 0 et 1. Le problème est le suivant. On peut produire facilement une quasi-preuve t_0 réalisant $0 = 0 \vee 0 = 1$ et une quasi-preuve t_1 réalisant $1 = 0 \vee 1 = 1$. En revanche on peut montrer que, pour réaliser $\forall x \ \ \nabla 2(x) \to (x = 0 \vee x = 1)$, il faudrait disposer d'une même quasi-preuve t réalisant à la fois $0 = 0 \vee 0 = 1$ et $1 = 0 \vee 1 = 1$: il n'y a pas de raison qu'un tel objet existe. De fait, dans le modèle des threads, ce n'est pas le cas.

II.8 Propriétés du modèle des threads

Rappelons que, puisque le pôle des threads est cohérent, le modèle des threads est bien un modèle cohérent de l'arithmétique [2].

Voici quelques propriétés remarquables de ce modèle :

- Il possède des entiers non-standard,
- Il possède une infinité d'éléments qui ne sont pas entiers.
- Son algèbre de Boole caractéristique est *sans atome*. En particulier, elle n'est pas finie. Pourtant, elle ne contient que deux entiers : 0 et 1.

III Formules de Horn et formules clausales

On appelle *littéral positif* toute formule de la forme $\alpha = \beta$ et *littéral négatif* toute formule de la forme $\alpha \neq \beta$. On appelle *clause* toute disjonction de littéraux et *clause de Horn* toute clause contenant au plus un littéral positif. Dans les mathématiques courantes, de nombreauses notions fondamentales sont définies par des clauses de Horn. Un exemple est la notion d'ordre. D'autres sont définies par des clauses qui ne sont pas de Horn. Par exemple, la notion d'ordre total.

1. À condition de l'étendre arbitrairement à tout $\mathbb{N}$.
2. Du moins si l'on se restreint aux entiers, c'est-à-dire aux individus qui vérifient le prédicat N.

Theorème 13 (Clauses de Horn)**.** *Soit $H(\bar{x})$ une clause de Horn. Quel que soit le modèle de réalisabilité classique, si $\mathbb{N} \models \forall \bar{x}\ \ H(\bar{x})$, alors $\forall \bar{x}\ \ H(\bar{x})$ est réalisée ; sinon, $(\forall \bar{x}\ \ H(\bar{x})) \to \bot$ est réalisée.*

Theorème 14 (Clauses quelconques)**.** *Soient $P_1, ..., P_m$ des littéraux positifs ($m \geq 1$) et $N_1, ..., N_n$ des littéraux négatifs ($n \geq 0$). Soit $\bar{x}$ la liste des variables apparaissant dans ces littéraux. Considérons la formule clausale $\phi = \bar{x}\ \ \bigvee_{i=1}^{m} M_i \vee \bigvee_{j=1}^{n} N_j$. S'il existe un i tel que $\forall \bar{x}\ \ P_i \vee \bigvee_{j=1}^{n} N_j$ soit vraie dans $\mathbb{N}$, alors ϕ est réalisée dans tout modèle de réalisabilité classique. Sinon, $\neg\phi$ est réalisée dans le modèle des threads.*

IV Le modèle des threads en théorie des ensembles

On peut effectuer une construction analogue en théorie des ensembles. On définit ainsi une notion de modèle de réalisabilité de ZF, la théorie des ensembles usuelle. On définit de la même façon le modèle des threads.

Les détails de la construction sont plus compliqués que dans le cas de l'arithmétique, en particulier à cause du problème de l'axiome d'extensionnalité, qui se pose également en forcing. On peut les trouver dans [Kri11] et [Kri12a].

Pour une étude générale de la réalisabilité classique en théorie des ensembles, et du modèle des threads en particuiler, on peut voir [Kri11], [Kri12a], [Kri12b], [Kri14].

Le modèle des threads est donc un modèle de ZF. Il possède toutes les propriétés listées en II.8, mais aussi d'autres, plus marquantes encore :

- À condition de rajouter l'instruction *quote* mentionnée plus haut, l'axiome des choix dépendants est vrai,
- $\nabla 2$ n'est pas équipotent à $\nabla 2 \times \nabla 2$. Or $\nabla 2$ est infinie. Par conséquent, d'une part $\nabla 2$ est indénombrable, et d'autre part l'axiome du choix est faux,
- $\nabla 2$ se plonge dans $\mathbb{R}$, donc d'une part l'hypothèse du continu est fausse, et d'autre part, il n'existe pas de bon ordre sur $\mathbb{R}$,
- Il existe une application strictement croissante de $\nabla 2$ [1] dans l'ensemble des cardinaux infinis sous $\mathbb{R}$. Or $\nabla 2$ est une algèbre de Boole sans atomes, donc l'ensemble des cardinaux infinis sous $\mathbb{R}$ doit être plutôt complexe. En particulier, il contient une copie de $\mathbb{Q}$.

1. Muni de l'ordre qui provient de sa structure d'algèbre de Boole.

Références

[Coh63] Paul J. Cohen. The Independence of the Continuum Hypothesis. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 38 (2), pages 1143–1148. National Academy of Sciences, December 1963. http://www.jstor.org/stable/71858.

[Gri90] Timothy G. Griffin. A formulae-as-types notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 47–58. ACM, 1990.

[Kle45] Stephen C. Kleene. On the interpretation of intuitionistic number theory. *Journal of symbolic logic*, 10 :109–124, 1945.

[Kri09] J.-L. Krivine. Realizability in classical logic. In *Interactive models of computation and program behaviour*, volume 27 of *Panoramas et synthèses*. Société Mathématique de France, 2009. http://hal.archives-ouvertes.fr/hal-00154500.

[Kri11] J.-L. Krivine. Realizability algebras : a program to well order $\mathbb{R}$. *Logical Methods in Computer Science*, 7(3 :02) :1–47, 2011.

[Kri12a] J.-L. Krivine. Realizability algebras II : new models of ZF + DC. *Logical Methods in Computer Science*, 8(1 :10) :1–28, 2012.

[Kri12b] J.-L. Krivine. Realizability algebras III : some examples. http://arxiv.org/abs/1210.5065 (to appear), 2012.

[Kri14] J.-L. Krivine. On the structure of classical realizability models of ZF. http://arxiv.org/abs/1408.1868 (to appear), 2014.

[Miq11a] Alexandre Miquel. Existential witness extraction in classical realizability and via a negative translation. *Logical Methods in Computer Science*, 7(2), 2011.

[Miq11b] Alexandre Miquel. Forcing as a program transformation. In *Proceedings of the Twenty-Sixth Annual IEEE Symposium on Logic in Computer Science (LICS 2011)*, pages 197–206. IEEE Computer Society Press, June 2011.

[Rie14] Lionel Rieg. *On Forcing and Classical Realizability*. Theses, Ecole normale supérieure de lyon - ENS LYON, June 2014. https://tel.archives-ouvertes.fr/tel-01061442.