Université de Paris

Rapport

Théorie homotopique des types

Présentation de l'article :

Focalisation and Classical Realisability

Pablo Donato

1 Contexte et motivations

On attribue souvent la découverte de la correspondance entre preuves et programmes à Curry et Howard, car ce sont les premiers à avoir vraiment mis au jour la similitude formelle, syntaxique de ces structures. Toutefois, l'interprétation BHK de la logique intuitionniste peut être vue comme une première réalisation que les preuves ont un contenu calculatoire. La réalisabilité de Kleene est une formalisation directe de cette idée, où les formules sont interprétées comme des ensembles de programmes partageant un comportement commun. Dans ce cadre, le lemme d'adéquation assure qu'à toute preuve d'une certaine formule A, on peut associer un programme qui *réalise* A, c'est-à-dire qui satisfait le comportement spécifié par A (techniquement qui appartient à l'interprétation de A). Si l'on choisit d'utiliser la logique comme système de types pour le langage de programmation sous-jacent, on obtient trivialement la converse du lemme d'adéquation, puisque les preuves et les programmes bien typés sont en bijection, conformément à la vision Curry-Howard. Toutefois, on peut arguer que le véritable atout de la réalisabilité est précisément de ne pas vérifier cette converse. En effet, en interprétant les preuves dans un modèle de calcul plus riche, on se donne l'opportunité de :

1. **Donner un sens à la logique**, comprise comme un système contrôlant le comportement des programmes. En effet, comment définir ce que signifie "bien se comporter" si l'on ne dispose pas d'exemples de mauvais comportements (par exemple la non-terminaison) ? On rejoint alors la philosophie de Girard concernant le fondement des lois logiques, telle qu'elle a pu être incarnée dans la Ludique [3]. Ce n'est d'ailleurs pas un hasard si G. Munch-Maccagnoni (à qui l'on réfèrera dans la suite par "MM") mentionne à plusieurs reprises la Ludique comme source de vocabulaire et d'inspiration. Le théorème de *complétude interne*, formulé originellement en Ludique et adapté en annexe de l'article au système L de l'auteur, est probablement le résultat technique le plus important dans cette direction : en plus de pouvoir interpréter les connecteurs logiques comme des opérations sur des "comportements" (à la fois dans le sens technique et intuitif), ces opérations satisfont une propriété de complétude, qui intuitivement garantit que les règles des connecteurs logiques permettent de typer tous les programmes qui réalisent les formules prouvées.

2. **Découvrir de nouvelles logiques**. C'est notamment ce qui s'est passé dans le cadre de la réalisabilité de Kleene : en essayant de réinterpréter les réalisateurs dans la logique, on a découvert qu'ils implémentaient, en plus des théorèmes de l'arithmétique de Heyting, deux principes qui n'étaient jusque-là pas considérés comme intuitionnistes – le principe de Markov et la thèse de Church étendue. Le même phénomène est à l'œuvre dans l'article de MM, lorsqu'il montre comment il est possible de définir en réalisabilité classique (avec système L) le comportement associé à une quantification universelle du second ordre non-standard, qui correspond en fait au polymorphisme à la ML obtenu via la *value restriction*.

2 Héritage et contributions

Comme il l'explique lui-même en conclusion, l'article de MM consiste en une synthèse de ces deux paradigmes de la constructivité logique que sont la correspondance de Curry-Howard, où les programmes sont vus comme une syntaxe de termes pour les preuves, et la réalisabilité, où les formules sont vues comme spécifications des programmes. Et comme nous l'avons mentionné, il s'agit ici de réalisabilité *classique* : l'objectif est de comprendre le contenu calculatoire des preuves de la logique classique, et non plus seulement intuitionniste.

2.1 Séquents, machines et réalisabilité classique

Du côté **Curry-Howard**, MM complète une lignée de travaux qui a commencé avec l'interprétation par Griffin [4] du raisonnement par l'absurde comme règle de typage de l'instruction `call/cc` de Scheme, et a culminé dans l'interprétation par Curien et Herbelin [2] du calcul des séquents classique LK de Gentzen comme système de type pour le $\bar{\lambda}\mu\tilde{\mu}$-calcul. Le système L que nous avons déjà évoqué n'est en fait qu'une nouvelle dénomination du $\bar{\lambda}\mu\tilde{\mu}$-calcul, choisie pour coller plus fidèlement à la tradition des systèmes de calcul des séquents à la Gentzen.

Du côté **réalisabilité**, MM étend à système L la réalisabilité classique de Krivine [5], qui a été conçu initialement pour les termes du λ-calcul étendu avec `call/cc`. Cette dernière se base sur une dualité entre les programmes et leur environnement/contexte d'évaluation, incarnée sous la forme de la machine abstraite de Krivine, qui est un système de réécriture implémentant l'évaluation en appel par nom des λ-termes. On peut alors définir une notion d'*observation* de l'interaction entre un programme et un environnement au sein de la machine, qui donne elle-même lieu à une notion d'*orthogonalité* : un programme et un environnement sont orthogonaux si l'on peut observer leur interaction. Typiquement, on peut définir les observations comme la clôture par anti-réduction de l'ensemble des formes normales, ce qui permet de dire qu'un programme et un environnement sont orthogonaux lorsque l'exécution du premier dans le second par la machine *termine*. Il n'est pas difficile d'étendre la notion d'orthogonalité à système L, qui hérite de la dualité intrinsèque aux côtés gauche et droite des séquents classiques via la négation involutive, et utilise une syntaxe à base de *commandes* qui reprend essentiellement la syntaxe des *processus* de la machine de Krivine.

2.2 Focalisation et stratégies d'évaluation

L'ingrédient final qui fait la spécificité du travail de MM est l'utilisation d'une stratégie d'évaluation **focalisante**, basée sur la notion de *polarité* issue de la logique linéaire. Historiquement, la focalisation a été découverte par Andreoli [1] dans le domaine de la programmation logique, alors qu'il cherchait à optimiser la procédure de recherche de preuves dans le calcul des séquents de la logique linéaire LL. L'idée est que la recherche peut être divisée en deux phases qui alternent l'une avec l'autre :

- La phase **asynchrone**, qui s'exécute dès que certaines formules dites *négatives* peuvent être décomposées dans le séquent courant, car il n'y a pas de risque de tomber sur des prémisses non-prouvables.
- La phase **synchrone**, qui s'exécute lorsque le séquent ne contient plus que des formules dites *positives* ou des atomes : il faut alors choisir une formule positive à décomposer. La découverte principale d'Andreoli est que cela peut être fait de manière complètement *focalisée*, c'est-à-dire qu'une fois la formule sélectionnée, il est possible de continuer à décomposer systématiquement ses sous-formules jusqu'à la prochaine phase asynchrone.

La propriété essentielle de cette procédure est qu'elle est complète vis-à-vis de la prouvabilité, c'est-à-dire qu'elle sera capable de trouver une preuve de n'importe quelle formule de LL qui est prouvable. De plus, elle exhibe une dualité intrinsèque aux formules/connecteurs de la logique, qui n'est pas limitée aux côtés droite/gauche des séquents. MM interprète cette dualité entre positifs et négatifs comme celle entre évaluations *stricte* et *paresseuse*, incarnée traditionnellement dans les stratégies en *appel par valeur* et *appel par nom*. Le fait d'avoir cette dualité au niveau des connecteurs, et donc des constructions du langage de programmation (appelé $\mathsf{L}_{\mathsf{foc}}$ pour "système L focalisé"), permet de mixer les deux modes d'évaluation au sein d'une unique stratégie, qui a comme propriété d'avoir pour formes normales des termes qui exhibent la même structure focalisée que les preuves d'Andreoli. De plus, la stratégie a le bon goût d'être *confluente*, au même titre que la procédure d'Andreoli déterminise drastiquement la recherche de preuves. Dit autrement, la focalisation permet d'avoir une sensibilité *locale* à l'ordre d'évaluation, et non *globale* comme c'est le cas en $\bar{\lambda}\mu\tilde{\mu}$-calcul, où c'est le choix d'une stratégie particulière qui permet de rendre le calcul confluent. MM utilise donc la logique linéaire pour typer les programmes de $\mathsf{L}_{\mathsf{foc}}$, en déconstruisant la conjonction classique $\wedge$ en une conjonction positive $\otimes$ et une conjonction négative $\&$, et la disjonction classique $\vee$ en une disjonction positive $\oplus$ et une disjonction négative $⅋$. Les connecteurs standards peuvent être retrouvés via un encodage inspiré de celui du système LC de Girard, donné en annexe de l'article.

2.3 Applications de la réalisabilité

Après avoir prouvé le lemme d'adéquation pour $\mathsf{L}_{\mathsf{foc}}$, MM conclut son article par une liste de résultats, qui exploitent ce lemme pour prouver diverses propriétés sur les programmes et les connecteurs de la logique. En particulier, il montre qu'il est possible d'obtenir des propriétés de constructivité telles que la *propriété de la disjonction* sur $\oplus$, et plus généralement que la réalisabilité classique permet d'exploiter la polarisation des formules pour prouver de la même façon ces propriétés de constructivité pour tous les connecteurs positifs. On en déduit une forme de **sécurité du typage**, qui se prouve de manière plus directe qu'avec les traditionnelles propriétés syntaxiques telles que la réduction du sujet. Un autre résultat surprenant est qu'en faisant abstraction du connecteur, on obtient gratuitement la **normalisation** de $\mathsf{L}_{\mathsf{foc}}$.

Bibliographie

[1] Jean-Marc Andreoli. 1992. Logic Programming with Focusing Proofs in Linear Logic. (1992).

[2] Pierre-Louis Curien et Hugo Herbelin. 2000. The Duality of Computation. *SIGPLAN Not.* 35, 9 (septembre 2000), 233-243. DOI :https://doi.org/10.1145/357766.351262

[3] Jean-Yves Girard. 2001. Locus Solum : From the Rules of Logic to the Logic of Rules. In *Proceedings of the 15th International Workshop on Computer Science Logic* (CSL '01), Springer-Verlag, Berlin, Heidelberg, 38.

[4] Timothy G. Griffin. 1989. A Formulae-as-Type Notion of Control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (POPL '90), Association for Computing Machinery, New York, NY, USA, 47-58. DOI :https://doi.org/10.1145/96709.96714

[5] Jean-Louis Krivine. 2009. Realizability in classical logic. *Panoramas et synthèses* 27, (2009), 197-229. Consulté à l'adresse https://hal.archives-ouvertes.fr/hal-00154500

[6] Guillaume Munch-Maccagnoni. 2009. Focalisation and Classical Realisability (version with appendices). In *18th EACSL Annual Conference on Computer Science Logic - CSL 09* (Lecture notes in computer science), Springer-Verlag, Coimbra, Portugal, 409-423. DOI :https://doi.org/10.1007/978-3-642-04027-6/_30