Types Intersection: *logique, sémantique, opérades*

Flavien BREUVART, Giulio MANZONETTO, Damiano MAZZA, Luc PELLISSIER, Domenico RUOPPOLO
(équipe LoVe)

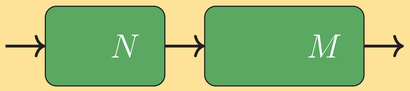
UNIVERSITÉ PARIS 13 — USPC Université Sorbonne Paris Cité

Un **langage** paradigmatique

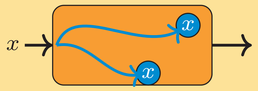
Le **λ-calcul** de Church : un système de réécriture de termes.

- Une syntaxe simple, fondée exclusivement sur deux idées :

- l'application MN



- l'abstraction de variable $\lambda x.M$



- Une dynamique de réécriture donnée par le remplacement:

$$(\lambda x.M)\ N \Rightarrow_\beta M\{N/x\}$$

- Un **modèle de calcul** : toute fonction calculable est représentable par des termes du système (*thèse de Church*).
- Une **théorie mathématique des fonctions exécutables** : idéale pour étudier la *chimie atomique de la programmation*.
- Le **noyau de tous les langages fonctionnels** (*OCaml*, *Haskell*, etc.) **et assistants de preuve** (*Coq*, *Agda*, etc.): indispensable pour leur conception, implementation, vérification.

Mais qu'est-ce qu'un **langage** ?

Notre point de vue issu de l'algèbre supérieure:

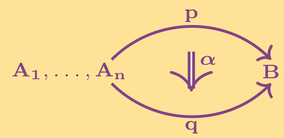
un **langage de programmation** $\mathcal{L}$ est une **2-opérade colorée**, ou **2-multicatégorie**, composée de :

objets, ou couleurs, $\mathbf{A}$, $\mathbf{B}$, ... ($\simeq$ types du langage)

pour tous objets $\mathbf{A_1}, \dots, \mathbf{A_n}, \mathbf{B}$ une catégorie $\mathcal{L}(A_1, \dots, A_n; B)$ avec :

– objets $\quad \mathbf{p} : \mathbf{A_1}, \dots, \mathbf{A_n} \longrightarrow \mathbf{B}$

aussi appelés 1-morphismes ($\simeq$ programmes du langage), qui composent *horizontalement* ($\simeq$ remplacement de termes)

– morphismes $\quad \mathbf{A_1}, \dots, \mathbf{A_n}$ 

aussi appelés 2-morphismes ($\simeq$ réductions de programmes), qui composent *verticalement* ($\simeq$ concatenation d'étapes d'exécution)

Des **typages** raffinés

Les termes du λ-calcul sont typables. Les **systèmes de types intersection** sont des mécanismes de typage très expressifs, fondés sur trois idées :

- le type flèche $A \to B$ pour les termes abstraits (fonctions)

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

- le sous-typage $A \leq B$, où le type A raffine le type B

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}$$

- l'intersection $A \wedge B$, où en particulier $A \wedge B \leq A$

$$\frac{\Gamma \vdash M : A \qquad \Delta \vdash M : B}{\Gamma, \Delta \vdash M : A \wedge B}$$

Des systèmes utiles pour plusieurs tâches :
- la **spécification** des comportements des programmes;
- la **caractérisation** facilement vérifiable de **propriétés dynamiques** de programmes, notamment leur **terminaison**;
- la **description** en forme syntaxique/logique de nombreuses **sémantiques mathématiques** du λ-calcul: modèles filtres, modèles de Krivine, modèles relationnels, ...

Des **typages** à la Grothendieck

Dans notre approche 2-opéradique, un **système de types** pour un langage $\mathcal{L}$ est un morphisme de 2-opérades colorées

$$\begin{array}{c} \mathcal{E} \\ \downarrow T \\ \mathcal{L} \end{array}$$

tel que les foncteurs sous-jacents

$$T_{A_1,\dots,A_n,B} : \mathcal{E}_{A_1,\dots,A_n,B} \longrightarrow \mathcal{L}_{A_1,\dots,A_n,B}$$

satisfont une notion faible de fibration. L'idée:

objet X de $\mathcal{E}$ $\simeq$ **raffinage** du type $T(X)$ de $\mathcal{L}$.

1-morphisme d de $\mathcal{E}$ $\simeq$ **derivation de typage** pour le programme $T(d)$ de $\mathcal{L}$.

Alternativement, grâce à une construction à la Grothendieck, T est un morphisme relaché

$$\mathcal{L} \xrightarrow{\tilde{T}} \mathcal{REL}$$

vers une bicatégorie $\mathcal{REL}$ de profoncteurs relationnels. L'idée:

$\tilde{T}(A)$ $\simeq$ *catégorie* des sous-typages de A

Des fondements **logiques**

- Le λ-calcul simplement typé Λ_{ST} est le typage fondé sur le seul type flèche. Il est également un système de preuve pour la logique implicative constructive, où $A \to B$ corresponde à l'implication (*isomorphisme de Curry-Howard*).

- La logique linéaire de Girard est une logique de **ressources** :

$A \multimap B$	exactement une copie de A produit B
$A \otimes A \multimap B$	deux copies de A produisent B
$!A \multimap B$	n'importe quel nombre de A produit B

- Λ_{ST} se traduit fidèlement dans la logique linéaire :

$$A \to B \rightsquigarrow\ !A \multimap B \quad \text{(traduction appel par nom)}$$
$$A \to B \rightsquigarrow\ !A \multimap !B \quad \text{(traduction appel par valeur)}$$

- Intuitivement, même les types $A \wedge B$ des systèmes avec intersection semblent liés à la logique linéaire :

$$\underbrace{A \wedge A \wedge \dots \wedge A}_{n} \to_{n\to\infty} \quad \begin{array}{c} A \wedge B \simeq A \otimes B \ ? \\ !A \ ? \end{array}$$

Mazza et Pellissier ont formalisé cette intuition dans un **contexte algébrique d'ordre supérieur** !

La **logique** linéaire au travail !

Le point de vue 2-opéradique clarifie la liaison entre types intersection et logique linéaire !

Mazza, Pellissier et Vial ont reconstruit la plupart des systèmes de types intersection comme instance d'une composition

$$\mathcal{L} \xrightarrow{G} \mathcal{L}_! \xrightarrow{Ap[\mathcal{D}]} \mathcal{REL}$$

L'idée:

$\mathcal{L}$ $\simeq$ 2-opérade du λ-calcul

$\mathcal{L}_!$ $\simeq$ 2-opérade de la logique linéaire

G $\simeq$ version opéradique d'une des traductions de Λ_{ST} dans la logique linéaire

le morphisme $Ap[\mathcal{D}]$ paramétré dans une opérade $\mathcal{D}$ de typages **polyadiques** (approximant la logique linéaire)

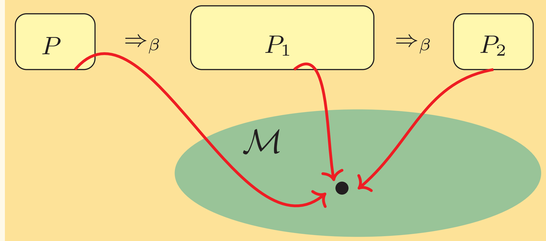
Dans ce cadre, les propriétés dynamiques capturées par les systèmes de types traditionnels deviennent des instances d'un seul **théorème fondamental**.

Référence principale

Mazza, Pellissier, Vial : **Polyadic Approximations, Fibrations and Intersection Types**, 2018, *Proceedings of the ACM on Programming Languages*

Une **sémantique** riche

La **sémantique dénotationnelle** interprète le λ-calcul dans d'autres espaces mathématiques $\mathcal{M}$ (structures algèbriques, topologiques, d'ordre, ...).
Les programmes β-convertibles y reçoivent la même dénotation : ils sont des différentes étapes d'une même exécution !



- L'équipe LoVe possède une vaste expértise dans l'études de modèles dénotationnels, souvent issus de logique linéaire, qui sont capables de représenter des aspects **quantitatifs** de l'exécution des programmes : nombre de ressources utilisées, temps d'exécution, probabilité de terminaison, ...
Les plus simples de ces modèles quantitatifs sont formalisables par systèmes de types intersection non idempotente

$$A \wedge A \quad \neq \quad A$$

L'idée: $\vdash P : A \Leftrightarrow$ la dénotation de P contient l'information A.

- Breuvart, Manzonetto et Ruoppolo ont exploité cette description par typage pour définir les propriétés d'**hyperimmunité** et **λ-König**, caractérisant tous les modèles relationnels qui engendrent certaines **equivalences observationnelles** (définitions de qu'est-ce ça veut dire que deux programmes ont le même comportement dans tout environnement d'évaluation).

Référence principale

Breuvart, Manzonetto, Ruoppolo : **Relational Graph Models at work**, 2018, *Logical Methods in Computer Science*

Vers une **sémantique** nouvelle ?

Travaux en cours

nous essayons maintenant d'explorer la **sémantique dénotationnelle** par le biais de l'approche opéradique !