Types de données et types fonctionnels en ludique

Alice Pavaux
LIPN, CNRS, UMR 7030 – Université Paris 13, Sorbonne Paris Cité

Logique et calcul

À l'origine, la **logique mathématique** s'intéresse aux fondements des mathématiques. Cette discipline a évolué au contact de l'informatique théorique, établissant en particulier des liens forts entre démonstrations formelles et modèles de calcul : c'est la **correspondance de Curry-Howard**.

Logique	Calcul
formules	**types**
preuves	**programmes**
normalisation	**exécution**

Dans le prolongement de cette correspondance, la **logique linéaire** s'intéresse à la gestion des ressources. C'est de cette logique que s'inspire la **ludique**, introduite par Girard [2].

Ludique (à la Terui [3])

La **ludique** est un système sur lequel la logique peut être reconstruite. Les **desseins**, objets de la ludique, sont des abstraction de preuves sans formules. Soient $\mathcal{A}$ une signature, $a, b, ... \in \mathcal{A}$ des noms munis chacun d'une arité.

Dessein positif | $\mathfrak{p} ::= \maltese \mid \Omega \mid \mathfrak{n}_0|\overline{a}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_{k_a}\rangle$
Dessein négatif | $\mathfrak{n} ::= x \mid \sum_{a\in\mathcal{A}} a(x_1^a, \dots, x_{k_a}^a).\mathfrak{p}_a$
tel que chaque variable apparaît libre au plus une fois.

L'**interaction** est la dynamique de la ludique. Elle consiste à éliminer les coupures, où une coupure est un lien établi entre deux desseins, l'un positif et l'autre négatif. Elle agit par la règle de réduction suivante :

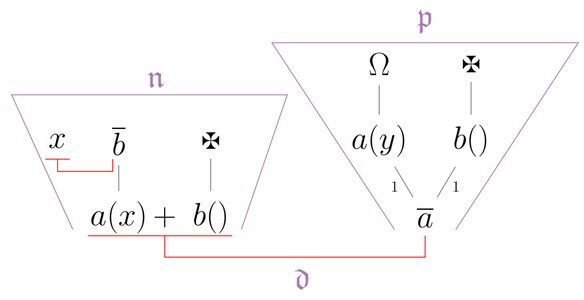
$$\sum_{a\in\mathcal{A}} a(x_1^a, \dots, x_{k_a}^a).\mathfrak{p}_a \mid \overline{b}\langle \mathfrak{n}_1, \dots, \mathfrak{n}_{k_b}\rangle \rightsquigarrow \mathfrak{p}_b[\mathfrak{n}_i/x_i^b]$$

Exemple : Soient $\mathfrak{n} = a(x).(x|\overline{b}\langle\rangle) + b().\maltese$ et $\mathfrak{p} = x_0|\overline{a}\langle a(y).\Omega + b().\maltese\rangle$. Posons $\mathfrak{d} = \mathfrak{p}[\mathfrak{n}/x_0]$, c'est-à-dire :

$$\mathfrak{d} = (a(x).(x|\overline{b}\langle\rangle) + b().\Omega) \mid \overline{a}\langle a(y).\Omega + b().\maltese\rangle$$

On a $\mathfrak{d} \rightsquigarrow (a(y).\Omega + b().\maltese) \mid \overline{b}\langle\rangle \rightsquigarrow \maltese$

Le dessein $\mathfrak{d}$ peut être représenté ainsi sous forme d'arbre :



Un nœud d'un tel arbre s'appelle une **action**.

Comportements et chemins visitables

Orthogonalité | $\mathfrak{p} \perp \mathfrak{n}$ si $\mathfrak{p}[\mathfrak{n}/x_0] \rightsquigarrow^* \maltese$

La relation d'**orthogonalité** entre deux desseins indique que l'interaction entre eux se passe bien.

Comportement | Ensemble de desseins $\mathbf{A}$ vérifiant $\mathbf{A}^{\perp\perp} = \mathbf{A}$

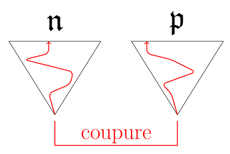
Les **comportements** sont les **types** de la ludique.

Le but de cette recherche est de caractériser les comportements qui correspondent aux types de données et de fonctions. Pour cela, la notion de **chemins visitables** d'un comportement [1] est indispensable.

Chemin visitable de A | Suite d'actions suivie au cours de l'interaction entre un dessein $\mathfrak{d} \in \mathbf{A}$ et un dessein $\mathfrak{e} \in \mathbf{A}^\perp$.

Si l'on reprend l'exemple précédent, on voit que $\mathfrak{p} \perp \mathfrak{n}$. De plus, on peut donner le chemin visitable qu'ils décrivent du côté de $\mathfrak{p}$:

$$\langle \mathfrak{p} \leftarrow \mathfrak{n} \rangle = \overline{a}\ \ b\ \ \maltese$$



Connecteurs de la ludique

Connecteurs appliqués sur des comportements pour en construire d'autres. Soient $\mathbf{A}, \mathbf{B}$ des comportements, soient $\uparrow, \overline{\uparrow}$ des noms d'actions, et posons $\downarrow = \overline{\uparrow}$, $\otimes = \overline{\overline{\uparrow}}$. On construit les comportements suivants :

$$\downarrow\mathbf{A} = \left\{ \begin{array}{c} \mathfrak{d} \\ \downarrow \end{array} \;\middle|\; \mathfrak{d} \in \mathbf{A} \right\}^{\perp\perp} \qquad \uparrow\mathbf{A} = \left\{ \begin{array}{c} \mathfrak{d} \\ \uparrow \end{array} \;\middle|\; \mathfrak{d} \in \mathbf{A} \right\}^{\perp\perp} \qquad \mathbf{A}\otimes\mathbf{B} = \left\{ \begin{array}{c} \mathfrak{d}\ \ \mathfrak{e} \\ \otimes \end{array} \;\middle|\; \mathfrak{d} \in \mathbf{A}, \mathfrak{e} \in \mathbf{B} \right\}^{\perp\perp}$$

$$\mathbf{A} \multimap \mathbf{B} = (\mathbf{A} \otimes \mathbf{B}^\perp)^\perp$$

La **complétude interne** est un résultat important sur les comportements construits avec connecteurs :

Théorème | La $\perp\perp$-clôture n'est pas nécessaire pour que $\downarrow\mathbf{A}$, $\uparrow\mathbf{A}$, $\mathbf{A} \otimes \mathbf{B}$ soient des comportements.

De cela, on peut déduire une caractérisation des chemins visitables de ces comportements.

- $V_{\mathbf{A}^\perp} = \widetilde{V_{\mathbf{A}}}$, où $\sim$ change la polarité des actions.
- $V_{\downarrow\mathbf{A}} = \downarrow V_{\mathbf{A}} \cup \{\maltese\}$ et $V_{\uparrow\mathbf{A}} = \uparrow V_{\mathbf{A}} \cup \{\epsilon\}$, où ϵ est le chemin vide.
- $V_{\mathbf{A}\otimes\mathbf{B}} \subseteq V_{\mathbf{A}} \sqcup V_{\mathbf{B}}$ où $\sqcup$ mélange les chemins ; $V_{\mathbf{A}\otimes\mathbf{B}} = V_{\mathbf{A}} \sqcup V_{\mathbf{B}}$ si $\mathbf{A}$ et $\mathbf{B}$ sont réguliers.
- $V_{\mathbf{A}\multimap\mathbf{B}} \subseteq \overbrace{V_{\mathbf{A}} \sqcup \widetilde{V_{\mathbf{B}}}}$; $V_{\mathbf{A}\multimap\mathbf{B}} = \overbrace{V_{\mathbf{A}} \sqcup \widetilde{V_{\mathbf{B}}}}$ si $\mathbf{A}$ et $\mathbf{B}$ sont réguliers.

Les comportements **réguliers** sont définis ci-dessous.

Correspondance type ↔ comportement, propriétés sur les types

Les types de données et de fonctions sont représentés par des comportements de la ludique, construits à l'aides des connecteurs ($\simeq$ *constructeurs* de types) suivants :

$$\mathbf{A}, \mathbf{B} ::= \mathcal{C}ste \mid \downarrow\uparrow\mathbf{A} \mid \mathbf{A} \otimes \mathbf{B} \mid \mathbf{A} \multimap \mathbf{B}$$

- Les constantes sont les types de bases.
- Si $\mathbf{C}$ est une constante, $\mathbf{C}$, $\downarrow\uparrow\mathbf{C}$, $\downarrow\uparrow\downarrow\uparrow\mathbf{C}$, ... représentent des types d'entiers finis
- $\mathbf{A} \otimes \mathbf{B}$ représente le type des paires (a, b) où $a : \mathbf{A}$ et $b : \mathbf{B}$
- $\mathbf{A} \multimap \mathbf{B}$ représente le type des fonctions de $\mathbf{A}$ dans $\mathbf{B}$

En combinant $\downarrow\uparrow$ et $\otimes$, on peut aussi représenter les listes, les arbres, ...

En plus de la représentation, on souhaiterait une **caractérisation**, c'est-à-dire avoir des propriétés simples à vérifier sur les comportements, permettant de prouver quelque chose du genre :

$\mathbf{A}$ est un comportement de données / fonctionnel **ssi** $\mathbf{A}$ satisfait [des propriétés ...]

Ce but n'est pas encore atteint. Mais sans avoir une caractérisation complète, on a déjà établi quelques conditions qui sont satisfaites par les comportements de données et de fonctions. En particulier, on veut qu'un type de données soit un comportement régulier et pur.

Régularité | Toute suite valide d'actions *utiles* de $\mathbf{A}$ est un chemin visitable de $\mathbf{A}$, et de même pour $\mathbf{A}^\perp$.
Pureté | Tout chemin visitable de $\mathbf{A}$ terminant par l'action $\maltese$ est extensible.

La **régularité** signifie que l'accès aux données n'est pas orienté. Les comportements réguliers correspondent exactement au fragment multiplicatif-additif de la logique linéaire.
La **pureté** signifie que les données sont complètes et pleinement accessibles: il n'y a pas de partie cachée, pas d'erreur.
On considère également une variante de la pureté :

Pureté* | Tout chemin visitable *bien parenthésé* de $\mathbf{A}$ terminant par $\maltese$ est extensible.

Le **bon parenthésage** signifie que l'on répond en premier à la dernière question posée.

Propriétés des données et des fonctions

On démontre que les bonnes propriétés définies plus haut sont conservées pour la majorité des nouveaux types construits avec les connecteurs :

Proposition
- Les constantes sont régulières et pures.
- Si $\mathbf{A}$ and $\mathbf{B}$ sont réguliers et purs, alors $\downarrow\uparrow\mathbf{A}$ et $\mathbf{A} \otimes \mathbf{B}$ le sont aussi.
- Si $\mathbf{A}$ and $\mathbf{B}$ sont réguliers, alors $\mathbf{A} \multimap \mathbf{B}$ l'est aussi.

Problème : $\mathbf{A}$ et $\mathbf{B}$ purs $\not\Rightarrow$ $\mathbf{A} \multimap \mathbf{B}$ pur.
En effet, dans les types de fonctions prenant en argument une fonction $(\mathbf{A} \multimap \mathbf{B}) \multimap \mathbf{C}$, certains chemins se terminent nécessairement par $\maltese$. Cependant, une propriété plus faible est vérifiée :

Proposition | Si $\mathbf{A}$ et $\mathbf{B}$ vérifient (**Pureté***) alors $\mathbf{A} \multimap \mathbf{B}$ aussi.

Interprétation

Pourquoi la pureté ne passe pas aux types fonctions ? Considérons une fonction de type

$$(\mathbf{A} \multimap \mathbf{B}) \multimap \mathbf{C}$$

Si la fonction répond un résultat dans $\mathbf{C}$ avant que toute l'information n'ait été obtenue sur le résultat dans $\mathbf{B}$ de la fonction passée en argument, alors l'information n'est pas complète dans le sens où l'on oublie de récupérer le résultat entier de $\mathbf{B}$. Ainsi, l'interaction produit un daimon $\maltese$.
Perspective : Donner une interprétation en terme d'exécution de **processus**.

Références

[1] C. Fouqueré et M. Quatrini : Incarnation in ludics and maximal cliques of paths. *Logical Methods in Computer Sciences*, 9(4), 2013.
[2] J.-Y. Girard : Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.
[3] K. Terui : Computational ludics. *Theor. Comput. Sci.*, 412(20): 2048–2071, 2011.