Master d'Informatique Fondamentale
École Normale Supérieure de Lyon

Théorie de la démonstration
Chantal KELLER

Computational types from a logical perspective

P.N. Benton, *G.M.* Bierman, *V.C.V. De* Paiva

Enseignants : Tom Hirschowitz
Philippe Audebaud

12 juin 2008

Introduction

Au début des années 1990, Moggi a introduit le "computational λ-calculus" (ou $\lambda\mathrm{ML}_T$) : à partir du λ-calcul simplement typé comprenant la conjonction et le pattern matching, il propose d'ajouter un constructeur de calcul T. L'un des objectifs est de permettre des notions impératives, comme les exceptions, dans un langage fonctionnel. Plusieurs années auparavant, Curry [1] puis récemment, Fairtlough et Mendler [2] ont présenté deux systèmes de logique s'avérant être très proches de $\lambda\mathrm{ML}_T$ pour la correspondance de Curry-Howard.

Ici, Benton et al. se proposent de partir de $\lambda\mathrm{ML}_T$ et de sa logique correspondante, nommée CL-logique, pour en faire une synthèse très complète. En particulier, ils introduisent un constructeur équivalent à T, noté $\diamond$, permettant de définir cette logique comme intermédiaire entre la logique intuitionniste et la logique linéaire.

Après une rapide description de $\lambda\mathrm{ML}_T$, nous reviendrons sur les présentations complémentaires de la CL-logique que sont la déduction naturelle et le calcul des séquents, puis prouverons la normalisation forte de cette logique. Nous évoquerons également les règles de réduction η que l'on peut définir, et enfin donnerons un modèle de la CL-logique basé sur les modèles de Kripke.

Remarque Pour définir $\lambda\mathrm{ML}_T$, Moggi donne une présentation en théorie des catégories. Dans l'article, Benton et al. évoquent très rapidement le modèle des catégories, mais nous n'en parlerons pas ici.

Table des matières

1 $\lambda\mathbf{ML}_T$

$\lambda\mathrm{ML}_T$ est formé à partir du λ-calcul simplement typé, comprenant conjonction et pattern matching, et auquel on a rajouté les deux règles suivantes :

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \mathrm{val}(e) : TA} \qquad \frac{\Gamma \vdash e : TA \qquad \Gamma, x : A \vdash f : TB}{\Gamma \vdash \mathrm{let}\ x \Leftarrow e\ \mathrm{in}\ f : TB}$$

Les règles de β-réduction pour $\lambda\mathrm{ML}_T$ sont donc :

$$\begin{aligned}
(\lambda x.u)v &\to_\beta u[v/x] && (1)\\
\mathrm{fst}(u,v) &\to_\beta u && (2)\\
\mathrm{snd}(u,v) &\to_\beta v && (3)\\
\mathrm{case\ inl}(e)\mathrm{of\ inl}(x) \to f \mid \mathrm{inr}(y) \to g &\to_\beta f[e/x] && (4)\\
\mathrm{case\ inr}(e)\mathrm{of\ inl}(x) \to f \mid \mathrm{inr}(y) \to g &\to_\beta g[e/y] && (5)\\
\mathrm{let}\ x \Leftarrow \mathrm{val}(u)\ \mathrm{in}\ v &\to_\beta v[u/x] && (6)
\end{aligned}$$

et on ajoute les axiomes suivants :

$$\begin{aligned}
\mathrm{let}\ x \Leftarrow (\mathrm{val}(e))\ \mathrm{in}\ f &= f[e/x] && (7)\\
\mathrm{let}\ x \Leftarrow e\ \mathrm{in}\ (\mathrm{val}(x)) &= e && (8)\\
\mathrm{let}\ x' \Leftarrow (\mathrm{let}\ x \Leftarrow e\ \mathrm{in}\ f)\ \mathrm{in}\ g &= \mathrm{let}\ x \Leftarrow e\ \mathrm{in}\ (\mathrm{let}\ x' \Leftarrow f\ \mathrm{in}\ g) && (9)
\end{aligned}$$

ainsi que des règles η, comme on le verra au paragraphe 5. L'axiome (8) sert à définir le cas de base, et nous reviendrons au paragraphe 3.2 sur l'axiome (9).

À partir de ces règles, il est très simple de montrer le lemme de substitution :

Lemme 1 (Substitution)

$$\frac{\Gamma \vdash e : A \qquad \Gamma, x : A \vdash f : B}{\Gamma \vdash f[e/x] : B}$$

Démonstration On a l'arbre de preuve suivant :

$$\frac{\dfrac{\Gamma \vdash e : A}{\Gamma \vdash \mathrm{val}(e) : TA} \qquad \Gamma, x : A \vdash f : B}{\Gamma \vdash \mathrm{let}\ x \Leftarrow (\mathrm{val}(e))\ \mathrm{in}\ f : B}$$

et l'axiome 7 clos la preuve. □

$\lambda\mathrm{ML}_T$ vérifie donc également la réduction du sujet pour $\to_\beta$:

Proposition 1 (Réduction du sujet) *Si* $\Gamma \vdash e : A$ *et* $e \to_\beta e'$ *alors* $\Gamma \vdash e' : A$.

ce qui se montre aisément par induction sur l'arbre de preuve de $\Gamma \vdash e : A$ à l'aide du lemme précédent.

L'une des idées originelles de Moggi était de pouvoir définir des notions de programmation impérative. Un exemple est celui des exceptions, que l'on peut définir par induction de la sorte :

$$\begin{array}{rcl} T(A) & \to & 1 + A \\ \mathrm{val}(e) & \to & \mathrm{inr}_{1+A}(e) \\ (\mathrm{let}\ x \Leftarrow e\ \mathrm{in}\ f) & \to & \mathrm{case}\ e\ \mathrm{of}\ \mathrm{inl}(*) \to\ \mathrm{inl}_{1+A}(*) \mid \mathrm{inr}(x) \to f \end{array}$$

L'utilisation du pattern matching semble naturelle puisqu'elle permet de distinguer les cas où aucune exception n'est levée (cas de "droite") des cas où une exception est levée (cas de "gauche"). Cette définition sera de nouveau utilisée au paragraphe 3.3.1 pour montrer la normalisation forte de $\lambda\mathrm{ML}_T$.

2 CL-logique

On utilise la correspondance de Curry-Howard "à l'envers" afin de pouvoir définir un système de logique à partir de $\lambda\mathrm{ML}_T$. Pour cela, on utilise les constructeurs de la logiques équivalents à ceux de la programmation, en utilisant notamment $\supset$ pour $\to$. De plus, on doit définir un nouveau constructeur équivalent à T, que l'on notera $\Diamond$.

2.1 Déduction naturelle

De même que pour définir $\lambda\mathrm{ML}_T$, on a les axiomes et les règles habituelles de déduction naturelle, ainsi que les deux règles suivantes :

$$\frac{\Gamma \vdash A}{\Gamma \vdash \Diamond A}\ \Diamond_{\mathcal{I}} \qquad\qquad \frac{\Gamma \vdash \Diamond A \qquad \Gamma, A \vdash \Diamond B}{\Gamma \vdash \Diamond B}\ \Diamond_{\mathcal{E}}$$

2.2 Calcul des séquents

On a également deux règles s'ajoutant aux axiomes et aux règles du calcul des séquents en logique intuitionniste, règles que l'on peut déduire des règles écrites en déduction naturelle :

$$\frac{\Gamma, A \vdash \Diamond B}{\Gamma, \Diamond A \vdash \Diamond B}\ \Diamond_{\mathcal{L}} \qquad\qquad \frac{\Gamma \vdash A}{\Gamma \vdash \Diamond A}\ \Diamond_{\mathcal{R}}$$

2.3 Équivalence entre les deux présentations

Les deux présentations (déduction naturelle et calcul des séquents) sont équivalentes, c'est-à-dire :

Proposition 2

$$\Gamma \vdash_N A \ \Leftrightarrow\ \Gamma \vdash_S A$$

Démonstration Pour les règles autres que les deux nouvelles règles, le résultat est connu.
Montrons que $\Gamma \vdash_S A \Rightarrow \Gamma \vdash_N A$ pour ces deux nouvelles règles.

- Clairement, $\diamond_{\mathcal{R}} \Rightarrow \diamond_{\mathcal{I}}$.
- Cet arbre de preuve montre que $\diamond_{\mathcal{L}} \Rightarrow \diamond_{\mathcal{E}}$:

$$\dfrac{\Gamma \vdash \diamond A \qquad \dfrac{\Gamma, A \vdash \diamond B}{\Gamma, \diamond A \vdash \diamond B}\,\diamond_{\mathcal{L}}}{\Gamma \vdash \diamond B}\,Cut$$

On peut démontrer de même que $\Gamma \vdash_N A \Rightarrow \Gamma \vdash_S A$. □

3 Normalisation et règles de réduction

Nous allons maintenant nous intéresser à la normalisation de la CL-logique en déduction naturelle, encore appelée élimination des coupures si on se place en calcul des séquents, puis aux règles de réduction en $\lambda \mathrm{ML}_T$. Cette étude est intéressante car elle permet de définir une notion d'égalité entre les preuves.

3.1 Normalisation en déduction naturelle

L'étude de la logique intuitionniste permet d'affirmer que l'on a la normalisation pour les règles habituelles [3]. Montrons-la lorsqu'on a une coupure pour $\diamond$, c'est-à-dire lorsque, dans un arbre de preuve, la règle $\diamond_{\mathcal{I}}$ est immédiatement suivie de la règle $\diamond_{\mathcal{E}}$. On est alors nécessairement dans le cas suivant :

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ A\end{array}}{\diamond A}\,\diamond_{\mathcal{I}} \qquad \diamond B}{\diamond B}\,\diamond_{\mathcal{E}}$$

où le premier $\diamond B$ est obtenu à partir de preuves de A. On peut donc éliminer la coupure pour obtenir :

$$\begin{array}{c}\vdots\\ [A]\\ \vdots\\ \diamond B\end{array}$$

On constate que cette réduction est similaire à la β-réduction (6).

Un autre cas qui peut apparaître lors de la normalisation est le suivant :

$$\dfrac{\dfrac{\diamond A \qquad \diamond B}{\diamond B}\,\diamond_{\mathcal{E}} \qquad \diamond C}{\diamond C}\,\diamond_{\mathcal{E}} \quad \overset{?}{\Longleftrightarrow} \quad \dfrac{\diamond A \qquad \dfrac{\diamond B \qquad \diamond C}{\diamond C}\,\diamond_{\mathcal{E}}}{\diamond C}\,\diamond_{\mathcal{E}}$$

Pour régler ce problème, il suffit d'orienter la flèche : c'est la commutation. Dans l'article, on choisit de l'orienter de la première règle vers la deuxième. Cette règle de réduction correspond alors à la réduction c, expliquée au paragraphe suivant 3.2.

3.2 Règles de réduction

Pour avoir la normalisation de $\lambda\mathrm{ML}_T$, on ajoute la règle suivante :

$$\text{let } x' \Leftarrow (\text{let } x \Leftarrow e \text{ in } f) \text{ in } g \quad \rightarrow_c \quad \text{let } x \Leftarrow e \text{ in } (\text{let } x' \Leftarrow f \text{ in } g)$$

Cette règle est en fait l'orientation de l'axiome (9), et elle correspond au λ-lifting. De plus, la propriété de réduction du sujet est conservée :

Proposition 3 (Réduction du sujet) *Si* $\Gamma \vdash e : A$ *et* $e \rightarrow_{\beta c} e'$ *alors* $\Gamma \vdash e' : A$.

3.3 Normalisation forte et confluence

3.3.1 Normalisation forte

Nous allons maintenant montrer la normalisation forte de $\lambda\mathrm{ML}_T$ avec la règle de réduction $\rightarrow_{\beta c}$. Pour cela, nous allons utiliser une variante de la méthode de réduction de Tait.

Étant donnés deux langages $\mathcal{L}_1$ et $\mathcal{L}_2$ munis chacun d'une règle de réduction $\rightarrow_1$ et $\rightarrow_2$, on cherche à montrer la normalisation forte de $\mathcal{L}_1$ connaissant celle de $\mathcal{L}_2$. Pour cela, il suffit de trouver une traduction :

$$^\circ : \mathcal{L}_1 \rightarrow \mathcal{L}_2$$

telle que

$$\forall (e, f) \in \mathcal{L}_1 \times \mathcal{L}_1,\ e \rightarrow_1 f \Rightarrow e^\circ \rightarrow_2^+ f^\circ \tag{10}$$

c'est-à-dire une traduction conservant la réduction.

Dans notre cas, nous pouvons choisir :

- $\mathcal{L}_1 = \lambda\mathrm{ML}_T$, $\rightarrow_1 = \rightarrow_{\beta c}$;
- $\mathcal{L}_2 = \lambda$-calcul simplement typé, $\rightarrow_2 = \rightarrow_{\beta c}$.

En effet, on sait [4] que le λ-calcul simplement typé est fortement normalisant. De plus, on peut trouver une traduction $^\circ$ assez simple vérifiant la propriété (10), définie par récurrence de la sorte : $^\circ$ ne change rien sur les éléments du λ-calcul simplement typé, et pour les constructeurs nouveaux :

- sur les types : $(TA)^\circ = 1 + A^\circ$;
- sur les termes :

$$\begin{aligned} (\mathrm{val}(e))^\circ &= \mathrm{inr}(e^\circ) \\ (\text{let } x \Leftarrow e \text{ in } f)^\circ &= \text{case } e^\circ \text{ of } \mathrm{inl}(z) \rightarrow \mathrm{inl}(z) \mid \mathrm{inr}(x) \rightarrow f^\circ \end{aligned}$$

Il est aisé de montrer les propriétés voulues de cette traduction.

On peut ici remarquer un parallèle avec la construction des exceptions au paragraphe 1.

3.3.2 Confluence

Maintenant que nous connaissons la normalisation forte de $\lambda\mathrm{ML}_T$, le lemme de Newmann affirme qu'il suffit de prouver la confluence faible de $\lambda\mathrm{ML}_T$ pour en montrer la confluence. De plus, le théorème des paires critiques assure que $\lambda\mathrm{ML}_T$ est faiblement confluent si et seulement il est faiblement confluent sur ses paires critiques.

Ici, les paires critiques apparaissent lorsqu'on a deux redex possibles dans une expression. Leur étude est donc similaire à celle des paires critiques du λ-calcul simplement typé.

4 Règles η

Dans la partie précédente, nous avons constaté qu'à chaque règle de normalisation de la CL-logique (élimination des coupures, commutation) correspondait une règle de réduction de $\lambda\mathrm{ML}_T$ (β, c). De même, nous pouvons caractériser des règles d'égalité η en $\lambda\mathrm{ML}_T$ en définissant des égalités entre arbres de preuve en CL-logique. Là encore, le but est de pouvoir réduire les preuves pour avoir une notion d'égalité entre elle.

Nous allons expliquer le fonctionnement de ces égalités sur l'exemple du constructeur T, puisqu'il est moins connu. Le principe est d'appliquer une règle droite (c'est-à-dire une règle d'introduction) pour le constructeur T à l'identité (l'axiome), immédiatement suivie d'une règle gauche (c'est-à-dire une règle d'élimination) pour ce même constructeur. On a donc :

$$\dfrac{\dfrac{\overline{z : TA, x : A \vdash x : A}}{z : TA, x : A \vdash \mathrm{val}(x) : TA}\, T_{\mathcal{R}} \qquad \overline{z : TA \vdash z : TA}}{z : TA \vdash \mathrm{let}\ x \Leftarrow z\ \mathrm{in}\ \mathrm{val}(x) : TA}\, T_{\mathcal{L}}$$

En réduisant cette dérivation à l'identité :

$$\overline{z : TA \vdash z : TA}$$

et en comparant les deux, on obtient la règle de réduction η correspondante en $\lambda\mathrm{ML}_T$:

$$\mathrm{let}\ x \Leftarrow z\ \mathrm{in}\ \mathrm{val}(x) \to_\eta z$$

5 Modèles de Kripke

Les modèles de Kripke permettent de montrer la consistance et la complétude de la CL-logique. On les définit ainsi :

Définition 1 (Modèle de Kripke) Un modèle de Kripke est un quintuplet $(W, V, \leqslant, R, \models)$ où :

- W est un ensemble non vide (ensemble des mondes possibles) ;
- $V : \mathcal{E} \to \mathcal{P}(W)$ où $\mathcal{E}$ est l'ensemble des variables propositionnelles (ensemble des mondes ou p est satisfaite) ;
- R et $\leqslant$ sont des ordres partiels sur W (relations d'accessibilité) ;

– $\models$ est une relation entre mondes et formules

qui vérifie, pour tout $w \in W$:

– $w \models p$ si et seulement si $w \in V(p)$;
– $x \models \top$;
– $w \models \bot$ si et seulement si $\forall p,\ w \models p$;
– $w \models A \wedge B$ si et seulement si $w \models A$ et $w \models B$;
– $w \models A \vee B$ si et seulement si $w \models A$ ou $w \models B$;
– $w \models A \supset B$ si et seulement si $\forall v \geqslant w,\ v \models A$ implique $v \models B$;
– $w \models \Diamond A$ si et seulement si $\forall v \geqslant w,\ \exists u,\ vRu$ et $u \models A$

et :

– si $w \models p$ et $w \leqslant v$ alors $v \models p$ (on dit que V est dirigée) ;
– si $w \models A$ et wRv alors $v \models A$.

Proposition 4 (Consistance et complétude) *$\vdash A$ si et seulement si $\forall w,\ w \models A$ (que l'on note $\models A$).*

Démonstration On va uniquement montrer la consistance, c'est-à-dire :

$$\text{si } \vdash A \text{ alors } \models A$$

Pour cela, on raisonne par induction sur l'arbre de preuve de $\Gamma \vdash A$, et on ne considère que le cas où la dernière règle est $\Diamond_{\mathcal{I}}$ ou $\Diamond_{\mathcal{E}}$, les autres cas étant connus. On se restreint également au cas (plus simple) où $R = \leqslant$.

On utilise la notation suivante : si $\Gamma = \phi_1, \ldots, \phi_n$,

$$\Gamma \models A \Leftrightarrow (\forall w \in W,\ w \models \phi_1, \ldots, w \models p_n \Rightarrow w \models A)$$

– Si le nœud de la racine est $\Diamond_{\mathcal{I}}$, alors le jugement est $\Gamma \vdash \Diamond A$. Par hypothèse d'induction, $\Gamma \models A$.
  Soit w tel que $w \models \phi_1, \ldots, w \models \phi_n$. Alors $w \models A$. Soit $v \geqslant w$. Par direction de V, $v \models A$. Comme $v \geqslant v$ par symétrie, $w \models \Diamond A$.
  Donc $\Gamma \models \Diamond A$.
– Si le nœud de la racine est $\Diamond_{\mathcal{E}}$, alors le jugement est $\Gamma \vdash \Diamond B$. Par hypothèse d'induction, $\Gamma \models \Diamond A$ et $\Gamma, A \models \Diamond B$.
  Soit w tel que $w \models \phi_1, \ldots, w \models \phi_n$. Comme $\Gamma \models \Diamond A$, $w \models \Diamond A$. Soit $v \geqslant w$. Comme $w \models \Diamond A$, $\exists u,\ v \leqslant u$ et $u \models A$. Or $u \geqslant w$ par transitivité, donc $u \models \phi_1, \ldots, u \models \phi_n$. Comme $u \models A$ et $\Gamma, A \models \Diamond B$, $u \models \Diamond B$. Donc $w \models \Diamond B$.
  Donc $\Gamma \models \Diamond B$. □

6 Remarques

L'article se veut une synthèse de $\lambda\mathrm{ML}_T$ et de la CL-logique, et est donc très concis. Cette concision est nécessaire pour développer une présentation aussi complète de ce système de logique, et souvent efficace (en effet, on constate que les preuves passées sous silence dans l'article ne sont pas difficiles, comme vu dans le rapport), compte tenu également du fait que la plupart des preuves sont déjà connues.

Cependant, il aurait parfois mérité plus de développement, comme par exemple la partie impérative. Celle qui concerne $\lambda\mathrm{ML}_T$ est très courte et aurait nécessité plus d'interprétation, et il n'y a aucune application de cette vision impérative pour la CL-logique.

De plus, les auteurs passent très rapidement sur la remarque faite en 1952 à Curry [1] concernant la possibilité de prouver $\Diamond A, \Diamond B \vdash \Diamond(A \wedge B)$. Cette formule, bien que ne prouvant pas l'inconsistance de la CL-logique (heureusement, car les modèles de Kripke en montrent la consistance !), peut introduire des formules intuitivement fausses. Cependant, l'explication donnée est peu satisfaisante...

Conclusion et perspectives

Benton et. al partent d'un système de calcul introduit "naturellement" en particulier pour donner des aspects impératifs à un langage de programmation fonctionnel. À partir de cela, ils définissent une logique consistante et complète, et font la synthèse de toutes ses propriétés connues. Ils obtiennent ainsi une logique intermédiaire entre la logique intuitionniste et la logique linéaire.

Cette logique a de nombreuses applications très variées, comme l'étude d'extensions de la logique linéaire, la vérification de hardware [2], ou encore d'autres extensions du λ-calcul simplement typé [5].

Références

[1] H.B. Curry : The elimination theorem when modality is present. *Journal of symbolic logic*, pages 249–265, 1952.

[2] M. Fairtlough et M. Mendler : An intuitionistic modal logic with applications to the formal verification of hardware. *Proceedings of conference on computer science logic*, 1995.

[3] J.Y. Girard, Y. Lafont et P. Taylor : Proofs and types. *Cambridge tracts in theoretical computer science*, 1989.

[4] D. Prawitz : Ideas and results in proof theory. *Proceedings of second scandinavian logic symposium*, pages 253–307, 1971.

[5] S. Lindley et I. Stark : Reducibility and TT-lifting for computation types. *Proceedings of TLCA*, 2005.

[6] P.N. Benton, G.M. Bierman et V.C.V. De Paiva : Computational types from a logical perspective. *Journal of functionnal programming*, 1993.