Lambda-calcul
et sémantique des langages de programmation

Thibaut Balabonski @ *UPSay*, Hiver 2021
https://www.lri.fr/~blsk/Semantique/

Examen - 3h - tous documents autorisés - les exercices sont indépendants - les minutages sont donnés à titre indicatif

Exercice 1. Lambda-calcul pur (*30-40 minutes*)

1. Montrer que $(\lambda y.(\lambda x.t))\ u\ =_\beta\ \lambda x.((\lambda y.t)u)$, en précisant le cas échéant la condition qui doit être respectée.
2. On définit la taille $|t|$ d'un λ-terme t par les équations suivantes :

$$\begin{array}{rcl} |x| & = & 1 \\ |\lambda x.t| & = & 1 + |t| \\ |t_1\ t_2| & = & 1 + |t_1| + |t_2| \end{array}$$

   Démontrer que pour tout terme t on a $|\mathsf{fv}(t)| \leqslant |t|$.
3. On rappelle que $\Theta\ =\ (\lambda xy.y(xxy))\ (\lambda xy.y(xxy))$ et que $f \circ g\ =\ \lambda x.f(g(x))$. Montrer que $g(\Theta(f \circ g))$ est un point fixe de $g \circ f$.
4. Construire un λ-terme K^∞ tel que pour tout t, $\mathsf{K}^\infty\ t\ =_\beta\ \mathsf{K}^\infty$.
5. On rappelle que $\mathsf{I} = \lambda x.x$. Dessiner le graphe de réduction du terme $t\ =\ (\lambda x.\mathsf{I}xx)\ (\lambda x.\mathsf{I}xx)$.

Exercice 2. Itérations (*30-40 minutes*)

1. Dans le codage de Church, le nombre entier (positif ou nul) n est codé par le λ-terme

$$[n] \quad \equiv \quad \lambda fx.f(f(\ldots f(x)\ldots))$$

   qui itère n fois la fonction f donnée en premier paramètre. On rappelle que l'on sait construire les λ-termes suivants :

S	tel que $\mathsf{S}\ [n] \to^*_\beta\ [n+1]$
$\langle t_1, t_2\rangle$	représentant la paire formée par les termes t_1 et t_2
$\pi_i(t)$	extrayant le i-ème élément de la paire t

   On pose

$$\begin{array}{rcl} t & \equiv & \lambda p.\langle \pi_2(p), \mathsf{S}\ \pi_2(p)\rangle \\ u & \equiv & \langle [0], [0]\rangle \\ v & \equiv & \lambda n.\pi_1(n\ t\ u) \end{array}$$

   Quel est le résultat de l'application suivante ?

$$v\ [n]$$

2. On propose de représenter une liste par sa fonction fold (correspondant à fold_left en Caml, ou reduce dans certains autres langages). Ainsi, une liste $[x, y, z]$ sera représentée par le λ-terme

$$\lambda fb.f\ x\ (f\ y\ (f\ z\ b))$$

   qui prend en paramètres une fonction f et un terme de base b, et calcule l'application itérée de f aux éléments de la liste en partant de la valeur de base b. Donner des λ-termes réalisant les opérations primitives suivantes :
   (a) la liste vide,
   (b) l'ajout d'un élément en tête d'une liste,
   (c) l'ajout d'un élément à la fin d'une liste,
   (d) le test de vacuité d'une liste,
   (e) l'extraction de la tête de la liste (c'est-à-dire, son premier élément),
   (f) l'extraction de la queue de la liste (c'est-à-dire, la liste formée de tous les éléments après le premier).

   Indication : vous pouvez vous inspirer de ce qui a été vu pour les entiers de Church, dans le cours ou à la question précédente.

Exercice 3. Sous-typage (*30-40 minutes*)

On considère l'ensemble des types définis par la grammaire suivante.

$$\begin{array}{rcl} \sigma, \tau & ::= & \mathsf{bool} \\ & | & \mathsf{int} \\ & | & \tau \to \tau \end{array}$$

On dit que σ est un sous-type de τ lorsque toute valeur de type σ peut être utilisée à une position où l'on attendait une valeur de type τ. Cette relation de sous-typage est notée $\sigma <: \tau$ et est définie par les règles d'inférence suivantes.

$$\frac{}{\mathsf{bool} <: \mathsf{int}}\ \textsc{Cast} \qquad \frac{}{\mathsf{bool} <: \mathsf{bool}}\ \textsc{RBool} \qquad \frac{}{\mathsf{int} <: \mathsf{int}}\ \textsc{RInt} \qquad \frac{\tau_1 <: \sigma_1 \qquad \sigma_2 <: \tau_2}{\sigma_1 \to \sigma_2 <: \tau_1 \to \tau_2}\ \textsc{Fun}$$

1. Pouvez-vous donner une intuition expliquant la règle Fun ?
2. Parmi les relations de sous-typage suivantes, lesquelles sont valides ? Justifier en donnant une dérivation ou en expliquant le problème.
   (a) $\mathsf{int} <: \mathsf{bool} \to \mathsf{int}$
   (b) $\mathsf{bool} \to \mathsf{bool} <: \mathsf{bool} \to \mathsf{int}$
   (c) $\mathsf{bool} \to \mathsf{int} <: \mathsf{int} \to \mathsf{int}$
   (d) $(\mathsf{bool} \to \mathsf{int}) \to \mathsf{bool} <: (\mathsf{int} \to \mathsf{int}) \to \mathsf{int}$
3. Montrer que pour tout type τ on peut dériver $\tau <: \tau$.
4. On souhaite démontrer que pour tous types σ, τ et ρ, s'il existe des dérivations des relations $\sigma <: \tau$ et $\tau <: \rho$, alors il existe une dérivation de $\sigma <: \rho$. On propose de procéder par récurrence sur la somme des tailles des dérivations de $\sigma <: \tau$ et $\tau <: \rho$.
   (a) Préciser l'énoncé de récurrence.
   (b) Démontrer le résultat.
   (c) À quel endroit aurait bloqué la preuve si l'on avait tenté une démonstration par récurrence sur la dérivation de $\sigma <: \tau$?

Exercice 4. Sûreté pour la sémantique à grands pas (*60-80 minutes*)

On considère un λ-calcul étendu avec des paires.

$$\begin{array}{rcl} t & ::= & x \\ & | & \lambda x.t \\ & | & t\ t \\ & | & \langle t, t \rangle \\ & | & \pi_1(t) \\ & | & \pi_2(t) \end{array}$$

Dans la sémantique à grands pas, on note $t \Downarrow v$ lorsque l'évaluation de t aboutit à la valeur v. Voici des règles d'inférence pour cette relation d'évaluation.

$$\frac{}{\lambda x.t \Downarrow \lambda x.t}\ \textsc{Abs} \qquad \frac{t \Downarrow \lambda x.t' \qquad t'\{x \leftarrow u\} \Downarrow v}{t\ u \Downarrow v}\ \textsc{App}$$

$$\frac{t_1 \Downarrow v_1 \qquad t_2 \Downarrow v_2}{\langle t_1, t_2 \rangle \Downarrow \langle v_1, v_2 \rangle}\ \textsc{Pair} \qquad \frac{t \Downarrow \langle v_1, v_2 \rangle}{\pi_i(t) \Downarrow v_i}\ \textsc{Proj}$$

1. À quelle stratégie de réduction correspond cette sémantique ?
2. On rappelle que $\Omega = (\lambda x.xx)\ (\lambda x.xx)$. Comment sont évalués les termes suivants ?
   (a) $\pi_1((\lambda xy.\langle x, y \rangle)\ (\lambda a.a)\ ((\lambda b.b)(\lambda c.c)))$
   (b) $\pi_1(\langle \lambda x.x, \Omega \rangle)$

(c) $\pi_2(\lambda x.x)$

(d) $\pi_2(\lambda x.\Omega)$

3. Supposons le temps d'une question que l'on remplace les constructions des paires et des projections par les encodages classiques suivants n'utilisant que le λ-calcul pur :

$$\begin{array}{rcl} \langle t_1, t_2\rangle & \equiv & \lambda x.x\ t_1\ t_2 \\ \pi_1(t) & \equiv & t\ (\lambda xy.x) \\ \pi_2(t) & \equiv & t\ (\lambda xy.y) \end{array}$$

Comparer la sémantique obtenue avec celle donnée par les règles d'inférence d'origine. *Indication : vous pouvez considérer comme exemples les termes de la question précédente.*

On associe à notre λ-calcul étendu les règles de typage suivantes.

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \textsc{TyVar} \qquad \frac{\Gamma, x : \sigma \vdash t : \tau}{\Gamma \vdash \lambda x.t : \sigma \to \tau}\ \textsc{TyAbs} \qquad \frac{\Gamma \vdash t : \sigma \to \tau \qquad \Gamma \vdash u : \sigma}{\Gamma \vdash t\ u : \tau}\ \textsc{TyApp}$$

$$\frac{\Gamma \vdash t_1 : \tau_1 \qquad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \langle t_1, t_2\rangle : \tau_1 \times \tau_2}\ \textsc{TyPair} \qquad \frac{\Gamma \vdash t : \tau_1 \times \tau_2}{\Gamma \vdash \pi_i(t) : \tau_i}\ \textsc{TyProj}$$

4. Comment se typent les termes de la question 2 ?
5. Démontrer la propriété suivante de préservation du typage.

$$\text{Si} \quad \Gamma \vdash t : \tau \quad \text{et} \quad t \Downarrow v \quad \text{alors} \quad \Gamma \vdash v : \tau.$$

On propose d'étendre notre sémantique à grands pas avec une notion d'erreur. Ainsi on notera $t \Downarrow \mathsf{err}$ lorsque l'évaluation de t ne peut aboutir à cause d'une opération incohérente (en revanche, quand on écrit $t \Downarrow v$ on sous-entend toujours $v \neq \mathsf{err}$). On introduit ainsi par exemple la règle suivante, indiquant comme erronée l'application d'une paire.

$$\frac{t \Downarrow \langle v_1, v_2\rangle}{t\ u \Downarrow \mathsf{err}}\ \textsc{AppPairErr}$$

Outre cette règle de base, on inclut les règles suivantes pour faire en sorte que toute erreur trouvée lors de l'évaluation d'un terme soit remontée à la racine.

$$\frac{t \Downarrow \mathsf{err}}{t\ u \Downarrow \mathsf{err}}\ \textsc{AppErr1} \qquad \frac{t \Downarrow \lambda x.t' \qquad t'\{x \leftarrow u\} \Downarrow \mathsf{err}}{t\ u \Downarrow \mathsf{err}}\ \textsc{AppErr2}$$

$$\frac{t_1 \Downarrow \mathsf{err}}{\langle t_1, t_2\rangle \Downarrow \mathsf{err}}\ \textsc{PairErr1} \qquad \frac{t_1 \Downarrow v_1 \qquad t_2 \Downarrow \mathsf{err}}{\langle t_1, t_2\rangle \Downarrow \mathsf{err}}\ \textsc{PairErr2} \qquad \frac{t \Downarrow \mathsf{err}}{\pi_i(t) \Downarrow \mathsf{err}}\ \textsc{ProjErr}$$

6. Il manque une règle pour tenir compte de toutes les erreurs possibles. Laquelle ?
7. Parmi les termes de la question 2, lesquels changent de sémantique avec cette extension ?
8. Qu'est-ce qui changerait si la règle PairErr2 était remplacée par la variante suivante ?

$$\frac{t_2 \Downarrow \mathsf{err}}{\langle t_1, t_2\rangle \Downarrow \mathsf{err}}\ \textsc{PairErr2Alt}$$

Vous pouvez présenter l'exemple d'un terme dont la sémantique serait différente.

9. Montrer le théorème suivant de sûreté.

$$\text{Si} \quad \Gamma \vdash t : \tau \quad \text{alors il est impossible que} \quad t \Downarrow \mathsf{err}.$$