Semantique et types

Strategies d'évaluation: etant donne une fonction, comment evaluer les operateurs?

-> appel par nom: lors de l'application de la fonction, substitution des parametres, puis evaluation de ces parametres uniquement quand c'est necessaire. probleme: l'évaluation peut se faire plusieurs fois pour un parametre

-> appel par valeur: evaluation des parametres avant substitution, des l'appel de la fonction, pour remplacer par les valeurs. probleme: un parametre peut être evalue inutilement

→ solution possible: utilisation d'un environnement

-> Strategie paresseuse: n'évalues les parametres que si c'est necessaire, et dans ce cas n'évaluer le parametre qu'une seule fois

principe:
- conserver dans l'environnement les parametres non evalues
- les evaluer la premiere fois qu'on en a besoin
- Utiliser la valeur deja calculee le reste du temps

Modele theorique des langages de programmation fonctionnelle:

λ-calcul, formalisme permettant de decrire la notion de calculabilité

notion centrale du λ-calcul: la fonction, où $f(x):t$ est noté $\lambda x.t$

On manipule des termes, de trois formes

x ⟼ variables

$\lambda x.t$ ⟼ fonctions

$t_1\ t_2$ ⟼ application de fonctions

β-egalité entre termes: $(\lambda x.t)\ u = t\{x \leftarrow u\}$ (substitution) (aussi β-reduction)

(« s'interprète comme » de la règle)

substitution:
- $x\{x \leftarrow u\} = u$
- $y\{x \leftarrow u\} = y \quad x \neq y$
- $(t_1\ t_2)\{x \leftarrow u\} = t_1\{x \leftarrow u\}\ t_2\{x \leftarrow u\}$
- $(\lambda x.t)\{x \leftarrow u\} = \lambda x.t$
- $(\lambda y.t)\{x \leftarrow u\} = \lambda y.(t\{x \leftarrow u\})$ « y ne doit pas appartenir aux variables libres de u

variable libre d'un terme (vl d'un terme)
- $vl(x) = \{x\}$
- $vl(t_1\ t_2) = vl(t_1) \cup vl(t_2)$
- $vl(\lambda x.t) = vl(t) - \{x\}$

en realité, les substitutions d'un terme est la substitution des occurences libres de ce terme.

convention d'écriture de Barendregt: ne pas donner le meme nom à deux variables (en particulier, ne pas donner le meme nom à des variables libres ou liees)

Operation de renommage $\lambda x.t = \lambda y.(t\{x \cdot y\})$, a condition que la variable y soit « fraiche »

Structural Operational Semantics

Si $t \to t'$ alors $tu \to t'u$

Si $u \to u'$ alors $tu \to tu'$

Si $t \to t'$ alors $\lambda x.t \to \lambda x.t'$

Système d'inférence associé

$\frac{t \to t'}{tu \to t'u}$ ① $\quad \frac{u \to u'}{tu \to tu'}$ ② $\quad \frac{t \to t'}{\lambda x.t \to \lambda x.t'}$ ③

$\frac{}{(\lambda x.t)u \to t\{x \leftarrow u\}}$ ④ (axiome)

système d'inférence ensemble de règles de la forme $\frac{P_1 \cdots P_n}{C}$ = "si toutes les prémisses P_i sont vérifiées alors la conclusion C est vérifiée

Prop : si $T \to T'$ alors $vl(T) \subseteq vl(T')$

→ Preuve par récurrence sur la dérivation $T \to T'$

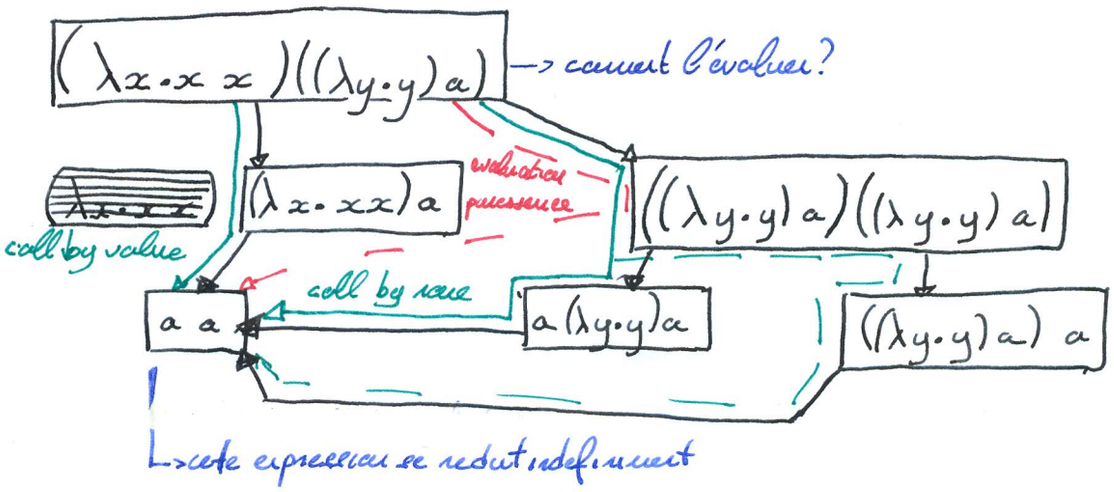
- ① $\frac{t \to t'}{tu \to t'u}$ (T = tu, T' = t'u)
  - Par hypothèse de récurrence, $vl(t') \subseteq vl(t)$
  - $vl(T') = vl(t'u) \cup vl(u)$ $\quad vl(t) \cup vl(u) = vl(T)$
- ② $\frac{u \to u'}{tu \to tu'}$
  - Par hypothèse de récurrence, $vl(u') \subseteq vl(u)$
  - $vl(tu) = vl(t) \cup vl(u) \subseteq vl(t) \cup vl(u') = vl(tu')$
- ③ $\frac{t \to t'}{\lambda x.t \to \lambda x.t'}$
  - même principe que dans les deux autres preuves
- ④ $\frac{}{(\lambda x.t)u \to t\{x \leftarrow u\}}$
  - $vl(t\{x \leftarrow u\}) \subseteq vl((\lambda x.t)u)$
    $vl(\lambda x.t) \cup vl(u)$
    $vl(t) \setminus \{x\} \cup vl(u)$

Lemme : $vl(t\{x \leftarrow u\}) \subseteq vl(t) \setminus \{x\} \cup vl(u)$: preuve par récurrence sur la définition de la substitution

① $vl(x\{x \leftarrow u\}) = vl(u) \subseteq (vl(x) \setminus \{x\}) \cup vl(u)$

② $vl(y\{x \leftarrow v\}) = vl(y) \subseteq vl(y) \setminus \{x\} \cup vl(v)$

⑤ $vl(t_1\, t_2)\{x \leftarrow u\} = vl(t_1\{x \leftarrow u\}\ t_2\{x \leftarrow u\})$



- Un terme peut ne pas avoir de résultat, ou avoir un unique résultat

Terminaison/Normalisation :

→ Terminaison forte : tous les chemins sont finis

→ Terminaison faible : existence d'un chemin qui s'arrête

Confluence « Tous les chemins se rejoignent en fait »
Implique l'unicité du résultat

Encore précis : t se réduit en un certain nombre d'étapes en u, et se réduit en un même nombre d'étapes en v, alors il existe un w tel que ce dernier est rejoignable à la fois par u et par v.

$\twoheadrightarrow$ réflexo-transitive de la relation $\longmapsto$

Confluence forte (diamant)

si $t \longmapsto u$ (en une étape) alors $\exists w$, $u \longmapsto w$ et $v \longmapsto w$ en une seule étape
$t \longmapsto v$

Réduction Parallèle $\longmapsto\!\!\!\!\!\!\rightarrow$

On s'autorise à effectuer plusieurs réductions en même temps

(P1) $\dfrac{t \longmapsto\!\!\!\!\!\!\rightarrow t' \quad u \longmapsto\!\!\!\!\!\!\rightarrow u'}{tu \longmapsto\!\!\!\!\!\!\rightarrow t'u'}$ (P2) $\dfrac{t \longmapsto\!\!\!\!\!\!\rightarrow t'}{\lambda x.t \longmapsto\!\!\!\!\!\!\rightarrow \lambda x.t'}$ (P3) $\dfrac{}{t \longmapsto\!\!\!\!\!\!\rightarrow t}$

(P4) $(\lambda x.t)u \longmapsto\!\!\!\!\!\!\rightarrow t\{x \leftarrow u\}$

Prop $\longmapsto = \underset{①}{\subseteq} \longmapsto\!\!\!\!\!\!\rightarrow \underset{②}{\subseteq} \longmapsto^*$ i.e est

① Pour tous t, t', si $t \longmapsto t'$ alors $t \longmapsto\!\!\!\!\!\!\rightarrow t'$
② Pour tous t, t' si $t \longmapsto\!\!\!\!\!\!\rightarrow t'$ alors $t \longmapsto^* t'$.

- Preuve du ① par récurrence sur $T \longmapsto T'$

- **règle 1**: HR: $t \longmapsto\!\!\!\!\!\!\rightarrow t'$.
  On est dans le cas $tu \to t'u$ avec $t \to t'$, et on cherche à montrer $tu \longmapsto\!\!\!\!\!\!\rightarrow t'u$

hypothèse de récurrence $\dfrac{t \longmapsto\!\!\!\!\!\!\rightarrow t' \quad \overline{u \longmapsto\!\!\!\!\!\!\rightarrow u}\ P3}{tu \longmapsto\!\!\!\!\!\!\rightarrow t'u}\ P1$

- **règle 2** même principe
- **règle 3** Trivial
- **règle 4** Trivial car aucune : dans le cas $(\lambda x.t)u \longmapsto t\{x \leftarrow u\}$
  on a directement $\overline{(\lambda x.t)u \longmapsto\!\!\!\!\!\!\rightarrow t\{x \leftarrow u\}}$

Preuve du ② Par récurrence sur $T \longmapsto\!\!\!\!\!\!\rightarrow T'$

- cas $tu \longmapsto\!\!\!\!\!\!\rightarrow t'u'$, avec la règle P1 appliquée et avec $\begin{cases} t \longmapsto\!\!\!\!\!\!\rightarrow t' \\ u \longmapsto\!\!\!\!\!\!\rightarrow u' \end{cases}$ $\boxed{\begin{matrix} t \longmapsto^* t' \\ u \longmapsto^* u' \end{matrix}}$ hypothèse de récurrence

objectif :
$tu \longmapsto^* t'u'$
$tu \longmapsto^* t'u \longmapsto t'u'$

Lemme si $t \longmapsto^* t'$ alors $tu \longmapsto^* t'u$, démonstration par récurrence sur le nombre d'étapes de $t \longmapsto t^*$ avec la règle A)

$(\lambda x.t)\,V \longrightarrow t\{x=V\}$

$\dfrac{v \longrightarrow v'}{tv \longrightarrow tv'}$ $\qquad$ $\dfrac{t \longrightarrow t'}{tv \longrightarrow t'v}$

V: les valeurs sont les fonctions $\lambda x.t$

→ Sémantique à petits pas (ou sémantique à réductions)
exécution d'un programme comme une séquence de petites étapes $t \longrightarrow t'$

$\lambda x.t \Rightarrow \lambda x.t$

$\dfrac{t \Rightarrow \lambda x.b \quad v \Rightarrow V \quad b\{x \leftarrow V\} = w}{tu \Rightarrow w}$

→ Sémantique à grands pas (ou sémantique naturelle)
exécution du programme vue comme la valeur produite, note $t \Rightarrow V$

Prop $t \longrightarrow^* V$ ssi $t \Rightarrow V$ (équivalence entre les deux modèles sémantiques)

Soit un langage auquel l'on ajoute des valeurs qui ne sont pas des fonctions (des entiers dans ce cas).

Sémantique à petits pas

$(\lambda x.x\,1)\,2 \longrightarrow 2\,1$
$\longrightarrow$ erreur/crash

Sémantique à grands pas → ajout d'une axiome

$\dfrac{}{n \Rightarrow n}$

$\dfrac{\lambda x.x\,1 \Rightarrow \lambda x.x1 \quad 2 \Rightarrow 2 \quad 2\,1 = ?}{(\lambda x.x\,1)\,2 \Rightarrow ?}$

→ On a une erreur ou un crash explicite dans de l'évaluation avec la sémantique à petits pas, tandis que l'évaluation avec la sémantique à grands pas ne fait que boucler indéfiniment

Démonstration de l'équivalence entre les deux sémantiques

— SGP → SPP : on veut montrer que $T \Rightarrow V$ implique $T \longrightarrow^* V$
Preuve par récurrence sur $T \Rightarrow V$

① Si $T = \lambda x.t$, on a $\lambda x.t \Rightarrow \lambda x.t$: on a donc $T \longrightarrow^* V$ en 0 étapes

② Soit $tu \Rightarrow w$
avec $\begin{cases} t \longrightarrow^* \lambda x.b \\ v \longrightarrow^* V \text{ (hypothèse de récurrence)} \\ b\{x \leftarrow v\} \longrightarrow^* w \end{cases}$

$tu \longrightarrow^* tV \longrightarrow^* (\lambda x.b)V \longrightarrow^* b\{x \leftarrow v\} \longrightarrow^* w$

— SPP → SGP plus difficile, il faut utiliser deux lemmes
• $V \Rightarrow V$ (trivial)
• $t \longrightarrow t' \Rightarrow V$ implique $t \Rightarrow V$ (se démontre par récurrence sur $t \longrightarrow t'$)

Extension du langage : ajout de paires

→ $t ::= x$
$\mid \lambda x.t$
$\mid t\,u$
$\mid (t,u)$
$\mid \mathrm{fst}(t)$
$\mid \mathrm{snd}(t)$
} ajout de règles pour ces trois constructions

Nouvelles règles

Sémantique à petits pas	Sémantique à grands pas
$\dfrac{t1 \longrightarrow t'1}{(t1,V) \longrightarrow (t'1,V)}$	$\dfrac{t1 \Rightarrow V1 \quad t2 \Rightarrow V2}{(t1,t2) \Rightarrow (V1,V2)}$
$\dfrac{}{\mathrm{fst}(v_1,v_2) \longrightarrow v_1}$ axiome même chose pour snd(t)	$\dfrac{t \Rightarrow (v_1,v_2)}{\mathrm{fst}(t) \Rightarrow v_1}$ même chose pour snd
$\dfrac{t2 \longrightarrow t'2}{(t1,t2) \longrightarrow (t1,t'2)}$	