Lambda-calcul
et sémantique des langages de programmation
Thibaut Balabonski @ *UPSay*
`https://www.lri.fr/~blsk/Semantique/`
Hiver 2021

Chapitre 2 - Stratégies de réduction

Graphe de réduction
Plusieurs réductions possibles pour un terme
Les différents chemins forment un graphe

$$
\begin{tikzcd}
 & (\lambda x.xx)\ ((\lambda y.y)\ a) \arrow[dl] \arrow[dr] & \\
((\lambda y.y)\ a)\ ((\lambda y.y)\ a) \arrow[d] \arrow[r] & & (\lambda x.xx)\ a \arrow[ddl] \\
a\ ((\lambda y.y)\ a) \arrow[dr] & ((\lambda y.y)\ a)\ a \arrow[d] & \\
 & aa &
\end{tikzcd}
$$

— question sur les chemins : certains sont-ils meilleurs ?
— questions sur le graphe : existence du résultat ? unicité ?

1 Normalisation

Forme normale
Une *forme normale* est un terme qui ne peut pas être réduit

Exemples
— x
— $\lambda x.xy$
— $x\ (\lambda y.y)\ (\lambda z.zx)$

Contre-exemples
— $(\lambda x.x)\ y$
— $x\ ((\lambda y.y)\ (\lambda z.zx))$

Si $t \to^* t'$ et t' est en forme normale, on dit que t' est une forme normale de t
C'est ce qu'on a considéré jusque là comme une notion intuitive de *résultat*

Terme sans forme normale

$$\begin{array}{rcl} \Omega & = & (\lambda x.xx)\ (\lambda x.xx) \\ & \rightarrow & (xx)\{x \leftarrow \lambda x.xx\} \\ & = & x\{x \leftarrow \lambda x.xx\}\ x\{x \leftarrow \lambda x.xx\} \\ & = & (\lambda x.xx)\ (\lambda x.xx) \\ & = & \Omega \end{array}$$

Autrement dit

— la réduction de Ω ne termine jamais

— Ω est un terme sans résultat

Mais que dire du terme suivant ?

$$(\lambda xy.y)\ \Omega\ z$$

Propriétés de normalisation

Un terme t est :

— *fortement normalisable* si toute séquence de réduction à partir de t aboutit à une forme normale

$$(\lambda xy.y)\ ((\lambda z.z)\ (\lambda z.z))$$

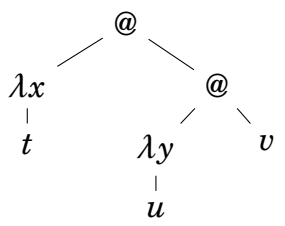
— *faiblement normalisable* s'il existe une séquence de réduction à partir de t qui aboutit à une forme normale

$$(\lambda xy.y)\ ((\lambda z.zz)\ (\lambda z.zz))$$

Note : la normalisation est un problème indécidable (voir chapitre λ-calculabilité)

2 Stratégies de réduction

Ordre de prise en compte des radicaux



Normal order : d'abord le radical le plus externe

— appliquer les fonctions sans chercher à réduire les arguments

Applicative order : d'abord le radical le plus interne

— normaliser les arguments avant de résoudre une application de fonction

Pour les radicaux disjoints : de gauche à droite

Exercice : comparaison normal / applicatif
Comparer les réductions en ordre normal et en ordre applicatif pour les termes suivants :

1. $(\lambda xy.x)\ z\ \Omega$
2. $(\lambda x.xx)\ ((\lambda y.y)\ z)$
3. $(\lambda x.x(\lambda y.y))\ (\lambda z.(\lambda a.aa)(z\ b))$

Dans chacun de ces cas, un autre ordre de réduction des radicaux permet-il une séquence plus courte ?
Correction

1. Ordre normal
$$\begin{array}{l} (\lambda xy.x)\ z\ \Omega \\ \rightarrow\ \ (\lambda y.z)\ \Omega \\ \rightarrow\ \ z \end{array}$$
Ordre applicatif
$$\begin{array}{l} (\lambda xy.x)\ z\ \Omega \\ \rightarrow\ \ (\lambda y.z)\ \Omega \\ \rightarrow\ \ (\lambda y.z)\ \Omega \\ \rightarrow\ \ \ldots \end{array}$$
La réduction en ordre normal est aussi courte que possible
2. Ordre normal
$$\begin{array}{l} (\lambda x.xx)\ ((\lambda y.y)\ z) \\ \rightarrow\ \ ((\lambda y.y)\ z)\ ((\lambda y.y)\ z) \\ \rightarrow\ \ z\ ((\lambda y.y)\ z) \\ \rightarrow\ \ zz \end{array}$$
Ordre applicatif
$$\begin{array}{l} (\lambda x.xx)\ ((\lambda y.y)\ z) \\ \rightarrow\ \ (\lambda x.xx)\ z \\ \rightarrow\ \ zz \end{array}$$
La réduction en ordre applicatif est aussi courte que possible
3. Ordre normal
$$\begin{array}{l} (\lambda x.x(\lambda y.y))\ (\lambda z.(\lambda a.aa)\ (z\ b)) \\ \rightarrow\ \ (\lambda z.(\lambda a.aa)(z\ b))\ (\lambda y.y) \\ \rightarrow\ \ (\lambda a.aa)\ ((\lambda y.y)\ b) \\ \rightarrow\ \ ((\lambda y.y)\ b)\ ((\lambda y.y)\ b) \\ \rightarrow\ \ b\ ((\lambda y.y)\ b) \\ \rightarrow\ \ bb \end{array}$$
Ordre applicatif
$$\begin{array}{l} (\lambda x.x(\lambda y.y))\ (\lambda z.(\lambda a.aa)(z\ b)) \\ \rightarrow\ \ (\lambda x.x(\lambda y.y))\ (\lambda z.(z\ b)\ (z\ b)) \\ \rightarrow\ \ (\lambda z.(z\ b)\ (z\ b))\ (\lambda y.y) \\ \rightarrow\ \ ((\lambda y.y)\ b)\ ((\lambda y.y)\ b) \\ \rightarrow\ \ b\ ((\lambda y.y)\ b) \\ \rightarrow\ \ bb \end{array}$$

Réduction la plus courte

$$
\begin{array}{ll}
 & (\lambda x.x(\lambda y.y))\ (\lambda z.(\lambda a.aa)\ (z\ b)) \\
\to & (\lambda z.(\lambda a.aa)(z\ b))\ (\lambda y.y) \\
\to & (\lambda a.aa)\ ((\lambda y.y)\ b) \\
\to & (\lambda a.aa)\ b \\
\to & bb
\end{array}
$$

Stratégie normalisante
Propriété de la stratégie *normal order*

— Si un terme t admet une forme normale alors la stratégie *normal order* atteint cette forme normale

(démonstration à venir dans un chapitre ultérieur)
Une telle stratégie est dite *normalisante*

3 Confluence

Confluences
Propriété du losange

$$
\begin{tikzcd}
 & t \arrow[dl] \arrow[dr] & \\
t_1 & & t_2
\end{tikzcd}
\quad \text{implique } \exists u \text{ tq} \quad
\begin{tikzcd}
t_1 \arrow[dr] & & t_2 \arrow[dl] \\
 & u &
\end{tikzcd}
$$

Confluence

$$
\begin{tikzcd}
 & t \arrow[dl, "*"'] \arrow[dr, "*"] & \\
t_1 & & t_2
\end{tikzcd}
\quad \text{implique } \exists u \text{ tq} \quad
\begin{tikzcd}
t_1 \arrow[dr, "*"] & & t_2 \arrow[dl, "*"'] \\
 & u &
\end{tikzcd}
$$

Le λ-calcul n'a pas la propriété du losange

$$
\begin{tikzcd}
 & (\lambda x.xx)\ ((\lambda y.y)\ a) \arrow[dl] \arrow[dr] & \\
(\lambda x.xx)\ a \arrow[dr] & & ((\lambda y.y)\ a)\ ((\lambda y.y)\ a) \arrow[dl] \\
 & aa &
\end{tikzcd}
$$

En revanche, le λ-calcul est confluent

Confluence du λ-calcul

1. On peut démontrer que le λ-calcul est *localement confluent*, c'est-à-dire

$$\begin{tikzcd} & t \arrow[dl] \arrow[dr] & \\ t_1 & & t_2 \end{tikzcd} \quad \text{implique } \exists u \text{ tq} \quad \begin{tikzcd} t_1 \arrow[dr, "*"] & & t_2 \arrow[dl, "*"'] \\ & u & \end{tikzcd}$$

2. Alors on ferme tout diagramme de la forme

$$\begin{tikzcd} & & & t \arrow[dl] \arrow[dr] & & & \\ & & u_1 \arrow[dl] & & v_1 \arrow[dr] & & \\ & u_2 \arrow[dl] & & & & v_2 \arrow[dr] & \\ \cdots & & & & & & \cdots \end{tikzcd}$$

par application répétée de la confluence locale

Contre-exemple : confluence locale n'implique pas confluence

$$\begin{tikzcd} a & b \arrow[l] \arrow[r, bend left] & c \arrow[l, bend left] \arrow[r] & d \end{tikzcd}$$

Cette relation est localement confluente, mais il est impossible de joindre la paire de réductions

$$\begin{tikzcd} & b \arrow[dl, "*"'] \arrow[dr, "*"] & \\ a & & d \end{tikzcd}$$

Pourquoi l'itération de la confluence locale ne fonctionne pas

$$\begin{tikzcd} & & & t \arrow[dl] \arrow[dr] & & & \\ & & u_1 \arrow[dl] \arrow[dr, "*"] & & v_1 \arrow[dl, "*"'] \arrow[dr] & & \\ & \cdots \arrow[dl] & & w & & \cdots \arrow[dr] & \\ u_k & & & & & & v_l \end{tikzcd}$$

Pas de garantie que les ouvertures

$$\begin{tikzcd} & u_1 \arrow[dl, "*"'] \arrow[dr, "*"] & \\ u_k & & w \end{tikzcd} \quad \text{et} \quad \begin{tikzcd} & v_1 \arrow[dl, "*"'] \arrow[dr, "*"] & \\ w & & v_l \end{tikzcd}$$

soient plus petites que celle d'origine !

Confluence du λ-calcul, *pour de vrai*
Preuve de Tait et Martin-Löf
Définir une relation $\rightrightarrows_\beta$ qui :

— est intermédiaire entre $\rightarrow_\beta$ et $\rightarrow_\beta^*$

— a la propriété du losange

Idée : réduction de plusieurs radicaux en parallèle, de sorte que, par exemple

$$((\lambda y.y)a)\ ((\lambda y.y)a) \quad \rightrightarrows_\beta \quad aa$$

Preuve de Tait et Martin-Löf : raisonnement

— $\rightrightarrows_\beta$ ayant la propriété du losange, on déduit que $\rightrightarrows_\beta^*$ a la propriété du losange

— de l'encadrement $\rightarrow_\beta \subseteq \rightrightarrows_\beta \subseteq \rightarrow_\beta^*$, on déduit $\rightrightarrows_\beta^* = \rightarrow_\beta^*$

— donc $\rightarrow_\beta^*$ a la propriété du losange

— d'où $\rightarrow_\beta$ est confluente

Définition de $\rightrightarrows_\beta$
Cas de base *réduction "identité" des variables*

$$\frac{}{x \rightrightarrows_\beta x}$$

Cas héréditaires *réduction parallèle des sous-termes*

$$\frac{t \rightrightarrows_\beta t'}{\lambda x.t \rightrightarrows_\beta \lambda x.t'} \qquad \frac{t_1 \rightrightarrows_\beta t'_1 \qquad t_2 \rightrightarrows_\beta t'_2}{t_1\ t_2 \rightrightarrows_\beta t'_1\ t'_2}$$

Radical déjà présent *réduction possible du β-radical et de ses sous-termes*

$$\frac{t \rightrightarrows_\beta t' \qquad u \rightrightarrows_\beta u'}{(\lambda x.t)\ u \rightrightarrows_\beta t'\{x \leftarrow u'\}}$$

Exemple de réduction parallèle

$$\frac{\dfrac{\dfrac{\dfrac{}{y \rightrightarrows_\beta y} \qquad \dfrac{\dfrac{}{z \rightrightarrows_\beta z}}{\lambda z.z \rightrightarrows_\beta \lambda z.z}}{(\lambda y.y)\ (\lambda z.z) \rightrightarrows_\beta \lambda z.z} \qquad \dfrac{}{x \rightrightarrows_\beta x}}{((\lambda y.y)\ (\lambda z.z))\ x \rightrightarrows_\beta (\lambda z.z)\ x} \qquad \dfrac{\dfrac{}{w \rightrightarrows_\beta w} \qquad \dfrac{}{a \rightrightarrows_\beta a}}{(\lambda w.w)a \rightrightarrows_\beta a}}{(\lambda x.((\lambda y.y)\ (\lambda z.z))\ x)\ ((\lambda w.w)\ a) \rightrightarrows_\beta (\lambda z.z)\ a}$$

Remarque : on ne réduit que des radicaux déjà présents *il peut donc rester de "nouveaux" radicaux dans le résultat*

Exercice : encadrement de $\rightrightarrows_\beta$
Montrer que

$$t \quad \rightrightarrows_\beta \quad t$$

Montrer que

$$\rightarrow_\beta \quad \subseteq \quad \rightrightarrows_\beta$$

Montrer que

$$\rightrightarrows_\beta \quad \subseteq \quad \rightarrow_\beta^*$$

Correction

— $t \rightrightarrows_\beta t$ par récurrence sur t

— $\rightarrow_\beta \subseteq \rightrightarrows_\beta$ par récurrence sur $\rightarrow_\beta$

— $\rightrightarrows_\beta \subseteq \rightarrow_\beta^*$ par récurrence sur $\rightrightarrows_\beta$

Exercice : méthode de Tait et Martin-Löf
Montrer que si une relation $\rightarrow$ a la propriété du losange, alors sa clôture réflexive transitive $\rightarrow^*$ a la propriété du losange.
Montrer que si deux relations $\rightarrow$ et $\rightrightarrows$ sont telles que

$$\rightarrow \quad \subseteq \quad \rightrightarrows \quad \subseteq \quad \rightarrow^*$$

alors leurs clôtures réflexives transitives $\rightrightarrows^*$ et $\rightarrow^*$ sont égales
Correction

— On suppose que $\rightarrow$ a la propriété du losange. Par récurrence sur la longueur de la séquence on démontre déjà que si $b \leftarrow a \rightarrow^n c$ alors il existe d tel que $d \rightarrow^n d \leftarrow c$. On en déduit (toujours par récurrence sur la longueur de l'une séquences) que si $b^k \leftarrow a \rightarrow^n c$ alors il existe d tel que $d \rightarrow^n d^k \leftarrow c$. D'où propriété du losange pour $\rightarrow^*$.

— De $\rightarrow \subseteq \rightrightarrows \subseteq \rightarrow^*$ on déduit $\rightarrow^* \subseteq \rightrightarrows^* \subseteq \rightarrow^{**}$. Remarque : $\rightarrow^{**} = \rightarrow^*$. On a donc $\rightarrow^* \subseteq \rightrightarrows^* \subseteq \rightarrow^*$, c'est-à-dire $\rightarrow^* = \rightrightarrows^*$.

Propriété du losange pour la réduction parallèle
Si $s \leftleftarrows t \rightrightarrows r$ alors il existe u tel que $s \rightrightarrows u \leftleftarrows r$

preuve par récurrence sur la dérivation de $t \rightrightarrows_\beta r$

— Cas $x \rightrightarrows x$. Alors $s = x$, on prend de même $u = x$

— Cas $\lambda x.t_0 \rightrightarrows \lambda x.r_0$ avec $t_0 \rightrightarrows r_0$. Alors $s = \lambda x.s_0$ avec $s_0 \leftleftarrows t_0$.
Par hypothèse de récurrence il existe u_0 tel que $s_0 \rightrightarrows u_0 \leftleftarrows r_0$.
Donc $\lambda x.s_0 \rightrightarrows \lambda x.u_0 \leftleftarrows \lambda x.r_0$

— Cas $t_1 t_2 \rightrightarrows r_1 r_2$ avec $t_1 \rightrightarrows r_1$ et $t_2 \rightrightarrows r_2$. On a deux possibilités pour $s \leftleftarrows t_1 t_2$.

  — si $s = s_1\, s_2$ avec $s_1 \leftleftarrows t_1$ et $s_2 \leftleftarrows t_2$
  alors par hypothèses de récurrence on trouve u_1 et u_2 tels que $s_1 \rightrightarrows u_1 \leftleftarrows r_1$ et $s_2 \rightrightarrows u_2 \leftleftarrows r_2$,
  d'où $s_1 s_2 \rightrightarrows u_1 u_2 \leftleftarrows r_1 r_2$

— si $s = s_1\{x \leftarrow s_2\}$ avec $t_1 = \lambda x.t'_1$ et $s_1 \Leftarrow t'_1$ et $s_2 \Leftarrow t_2$,
alors $r_1 = \lambda x.r'_1$ avec $t'_1 \Rightarrow r'_1$ et par hypothèses de récurrence on trouve u_1 et u_2 tels que $s_1 \Rightarrow u_1 \Leftarrow r'_1$ et $s_2 \Rightarrow u_2 \Leftarrow r_2$,
d'où $u_1\{x \leftarrow u_2\} \Leftarrow (\lambda x.r'_1)r_2$
et il reste à démontrer $s_1\{x \leftarrow s_2\} \Rightarrow u_1\{x \leftarrow u_2\}$
Lemme : si $a \Rightarrow_\beta a'$ et $b \Rightarrow_\beta b'$ alors $a\{x \leftarrow b\} \Rightarrow_\beta a'\{x \leftarrow b'\}$ prochainement sur votre écran

— Cas $(\lambda x.t_1)t_2 \Rightarrow r_1\{x \leftarrow r_2\}$ avec $t_1 \Rightarrow r_1$ et $t_2 \Rightarrow r_2$. On a deux possibilités pour $s \Leftarrow (\lambda x.t_1)t_2$.

— si $s = (\lambda x.s_1)s_2$ avec $s_1 \Leftarrow t_1$ et $s_2 \Leftarrow t_2$ on complète comme précédemment

— si $s = s_1\{x \leftarrow s_2\}$ avec $s_1 \Leftarrow t_1$ et $s_2 \Leftarrow t_2$
alors par hypothèses de récurrence on trouve u_1 et u_2 tels que $s_1 \Rightarrow u_1 \Leftarrow r_1$ et $s_2 \Rightarrow u_2 \Leftarrow r_2$,
Reste à démontrer que $s_1\{x \leftarrow s_2\} \Rightarrow u_1\{x \leftarrow u_2\} \Leftarrow r_1\{x \leftarrow r_2\}$
Même lemme

Lemme $a \Rightarrow_\beta a' \wedge b \Rightarrow_\beta b' \implies a\{x \leftarrow b\} \Rightarrow_\beta a'\{x \leftarrow b'\}$
Par récurrence sur la dérivation de $a \Rightarrow_\beta a'$

— Cas $y \Rightarrow y$.
Comparaison de x et y

— si $x = y$, alors $x\{x \leftarrow b\} = b \Rightarrow b' = x\{x \leftarrow b'\}$

— si $x \neq y$, alors $y\{x \leftarrow b\} = y \Rightarrow y = y\{x \leftarrow b'\}$

— Cas $\lambda y.a_0 \Rightarrow \lambda y.a'_0$ avec $a_0 \Rightarrow a'_0$.
Alors $(\lambda y.a_0)\{x \leftarrow b\} = \lambda y.(a_0\{x \leftarrow b\})$ et par hypothèse de récurrence $a_0\{x \leftarrow b\} \Rightarrow a'_0\{x \leftarrow b'\}$,
d'où $\lambda y.(a_0\{x \leftarrow b\}) \Rightarrow \lambda y.(a'_0\{x \leftarrow b'\}) = (\lambda x.a'_0)\{x \leftarrow b'\}$

— Cas $a_1a_2 \Rightarrow a'_1a'_2$ avec $a_1 \Rightarrow a'_1$ et $a_2 \Rightarrow a'_2$.
Alors $(a_1a_2)\{x \leftarrow b\} = (a_1\{x \leftarrow b\})(a_2\{x \leftarrow b\})$ et $(a'_1a'_2)\{x \leftarrow b'\} = (a'_1\{x \leftarrow b'\})(a'_2\{x \leftarrow b'\})$
et par hypothèse de récurrence $a_1\{x \leftarrow b\} \Rightarrow a'_1\{x \leftarrow b'\}$ et $a_2\{x \leftarrow b\} \Rightarrow a'_2\{x \leftarrow b'\}$,
d'où $(a_1a_2)\{x \leftarrow b\} \Rightarrow (a'_1a'_2)\{x \leftarrow b'\}$

— Cas $(\lambda y.a_1)a_2 \Rightarrow a'_1\{y \leftarrow a'_2\}$ avec $a_1 \Rightarrow a'_1$ et $a_2 \Rightarrow a'_2$.
Alors $((\lambda y.a_1)a_2)\{x \leftarrow b\} = (\lambda y.a_1\{x \leftarrow b\})(a_2\{x \leftarrow b\})$.
Or par hypothèses de récurrence $a_1\{x \leftarrow b\} \Rightarrow_\beta a'_1\{x \leftarrow b'\}$ et $a_2\{x \leftarrow b\} \Rightarrow_\beta a'_2\{x \leftarrow b'\}$,
d'où $(\lambda y.a_1\{x \leftarrow b\})(a_2\{x \leftarrow b\}) \Rightarrow (a'_1\{x \leftarrow b'\})\{y \leftarrow a'_2\{x \leftarrow b'\}\}$.
Par α-renommage on peut choisir $y \neq x$ et $y \notin \mathsf{fv}(b')$, donc par lemme de substitution $(a'_1\{x \leftarrow b'\})\{y \leftarrow a'_2\{x \leftarrow b'\}\} = (a'_1\{y \leftarrow a'_2\})\{x \leftarrow b'\}$.

Lemme de substitution
Si $x \neq y$ et $x \notin \mathsf{fv}(v)$ alors

$$t\{x \leftarrow u\}\{y \leftarrow v\} \quad = \quad t\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}$$

Preuve par récurrence sur t

— Cas d'une variable.
  — Cas $t = x$. Alors $x\{x \leftarrow u\}\{y \leftarrow v\} = u\{y \leftarrow v\}$ et $x\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\} = x\{x \leftarrow u\{y \leftarrow v\}\} = u\{y \leftarrow v\}$
  — Cas $t = y$. Alors $y\{x \leftarrow u\}\{y \leftarrow v\} = y\{y \leftarrow v\} = v$ et $y\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\} = v\{x \leftarrow u\{y \leftarrow v\}\} = v$
  — Cas $t = z$, sinon. Alors $z\{x \leftarrow u\}\{y \leftarrow v\} = z$ et $s\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\} = z$

— Cas d'une application $t_1\ t_2$. On suppose $t_1\{x \leftarrow u\}\{y \leftarrow v\} = t_1\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}$ et $t_2\{x \leftarrow u\}\{y \leftarrow v\} = t_2\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}$ Alors

$$\begin{array}{ll}
 & (t_1\ t_2)\{x \leftarrow u\}\{y \leftarrow v\} \\
= & (t_1\{x \leftarrow u\}\ t_2\{x \leftarrow u\})\{y \leftarrow v\} \\
= & t_1\{x \leftarrow u\}\{y \leftarrow v\}\ t_2\{x \leftarrow u\}\{y \leftarrow v\} \\
= & t_1\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}\ t_2\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\} \\
= & (t_1\{y \leftarrow v\}\ t_2\{y \leftarrow v\})\{x \leftarrow u\{y \leftarrow v\}\} \\
= & (t_1\ t_2)\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}
\end{array}$$

— Cas d'une abstraction $\lambda z.t$. On suppose $t\{x \leftarrow u\}\{y \leftarrow v\} = t\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}$ et par convention de Barendregt $z \neq x$ et $z \neq y$ et $z \notin \mathsf{fv}(u)$ et $z \notin \mathsf{fv}(v)$ (et $z \notin \mathsf{fv}(u\{y \leftarrow v\})$) Alors

$$\begin{array}{ll}
 & (\lambda z.t)\{x \leftarrow u\}\{y \leftarrow v\} \\
= & (\lambda z.(t\{x \leftarrow u\}))\{y \leftarrow v\} \\
= & \lambda z.(t\{x \leftarrow u\}\{y \leftarrow v\}) \\
= & \lambda z.(t\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}) \\
= & (\lambda z.(t\{y \leftarrow v\}))\{x \leftarrow u\{y \leftarrow v\}\} \\
= & (\lambda z.t)\{y \leftarrow v\}\{x \leftarrow u\{y \leftarrow v\}\}
\end{array}$$

Corollaire : théorème de Church-Rosser
Si

$$t_1 \quad =_\beta \quad t_2$$

alors il existe u tel que

$$t_1 \quad \rightarrow_\beta^* \quad u \qquad \text{et} \qquad t_2 \quad \rightarrow_\beta^* \quad u$$

Conséquences

— si t a une forme normale n, alors $t \rightarrow^* n$
— un terme ne peut avoir qu'une forme normale
— si deux formes normales n et m sont distinctes, alors $n \neq_\beta m$

4 Stratégies d'évaluation

Valeurs et formes normales
Les deux notions suivantes sont distinctes :

— valeur (en programmation fonctionnelle)

— forme normale (en λ-calcul)

La preuve avec la boucle interactive de caml

```
# let f = fun () -> 1/0 ;;
val f : unit -> int = <fun>

# f () ;;
Exception: Division_by_zero.
```

Valeurs en λ-calcul
En λ-calcul pur, une valeur est un terme de la forme

$$\lambda x.t$$

Chaque extension est susceptible d'apporter de nouvelles formes de valeurs

— constantes

— combinaisons de valeurs

— ...

Une grammaire des valeurs pour PCF

$$v \ ::= \ \lambda x.t \ | \ n \ | \ \mathsf{T} \ | \ \mathsf{F} \ | \ \langle v_1, v_2 \rangle$$

Évaluation
Évaluation

— réduire jusqu'à obtenir une valeur

À ne pas confondre avec *normalisation* : réduire jusqu'à obtenir une forme normale
Lors d'une évaluation, on ne réduit pas t dans $\lambda x.t$ On parle de *réduction faible*
On suppose souvent évaluer des termes *clos*, c'est-à-dire sans variables libres

Stratégies d'évaluation

```
let f x y z = x*x + y*y in
f (1+2) (3+4) (1/0)
```

Appel par valeur *call by value*

— évaluer les arguments avant d'appliquer une fonction

— `1+2`, `3+4`, `1/0` *Caml, C, Java, Python, et al.*

Appel par nom *call by name*

— passer les arguments non évalués, et les évaluer à la demande

— 1+2, 1+2, 3+4, 3+4 *macros*

Appel par nécessité *call by need*

— évaluer les arguments à la demande, et mémoriser le résultat

— 1+2, 3+4 *Haskell*

5 Sémantique opérationnelle

Sémantiques d'un langage de programmation
On veut donner un sens à un programme
Plusieurs styles

— sémantique opérationnelle *décrire les opérations du programme*

— sémantique dénotationnelle *associer le programme à un objet mathématique*

— sémantique axiomatique *associer le programme à des couples pré-conditions/post-conditions*

— sémantique de traces *décrire une séquence d'événements observables*

Sémantiques opérationnelles
Deux sous-styles
Opérationnelle à petits pas *sémantique à réduction*

— décrire chaque étape de calcul

— similaire à la β-réduction

Opérationnelle à grands pas *sémantique naturelle*

— décrire le résultat d'un programme

— similaire à un interprète

Dans les deux cas, reflète la stratégie d'évaluation

Appel par valeur, sémantique à petits pas : $\rightarrow_v$
Dans ces règles, v désigne exclusivement des valeurs

$$\frac{}{(\lambda x.t)\ v \ \rightarrow_v \ t\{x \leftarrow v\}}$$

$$\frac{t_1 \ \rightarrow_v \ t_1'}{t_1\ t_2 \ \rightarrow_v \ t_1'\ t_2} \qquad \frac{t_2 \ \rightarrow_v \ t_2'}{v_1\ t_2 \ \rightarrow_v \ v_1\ t_2'}$$

$\rightarrow_v$: relation binaire terme/terme, restriction de $\rightarrow_\beta$
$\{x \leftarrow v\}$ substitue des valeurs
Évaluation par enchaînement de petits pas : $\rightarrow_v^*$

Appel par nom, sémantique à petits pas : $\rightarrow_n$
On n'évalue jamais l'argument d'une application

$$\frac{}{(\lambda x.t_1)\ t_2\ \rightarrow_n\ t_1\{x \leftarrow t_2\}}$$

$$\frac{t_1\ \rightarrow_n\ t_1'}{t_1\ t_2\ \rightarrow_n\ t_1'\ t_2}$$

$\rightarrow_n$: relation binaire terme/terme, restriction de $\rightarrow_\beta$
$\{x \leftarrow t\}$ substitue des termes
Évaluation par enchaînement de petits pas : $\rightarrow_n^*$

Exercice : évaluation en sémantique à petits pas
Évaluer en appel par nom et en appel par valeur les termes

— $(\lambda xyz.zxy)\ ((\lambda a.a)v)\ (\lambda b.((\lambda c.c)v))$

— $(\lambda x.x(xv))\ ((\lambda yz.(\lambda w.w)z)\ v)$

Comparer les séquences de réduction Comparer les résultats obtenus
Correction

— Appel par nom

$$\begin{array}{l}(\lambda xyz.zxy)\ ((\lambda a.a)v)\ (\lambda b.((\lambda c.c)v))\\ \rightarrow\ \ (\lambda yz.z((\lambda a.a)v)y)\ (\lambda b.((\lambda c.c)v))\\ \rightarrow\ \ \lambda z.z((\lambda a.a)v)(\lambda b.((\lambda c.c)v))\end{array}$$

Appel par valeur

$$\begin{array}{ll}(\lambda xyz.zxy)\ ((\lambda a.a)v)\ (\lambda b.((\lambda c.c)v)) & \\ \rightarrow & (\lambda xyz.zxy)\ v\ (\lambda b.((\lambda c.c)v)) \rightarrow \\ (\lambda yz.zvy)\ (\lambda b.((\lambda c.c)v)) & \\ \rightarrow & \lambda z.zv(\lambda b.((\lambda c.c)v))\end{array}$$

Du fait que l'argument $(\lambda a.a)v$ n'est pas évalué en appel par nom avant d'être substitué dans le corps de la fonction, les deux valeurs obtenues ne sont pas égales, mais restent β-convertibles.

— Appel par nom

$$\begin{array}{l}(\lambda x.x(xv))\ ((\lambda yz.(\lambda w.w)z)\ v)\\ \rightarrow\ \ ((\lambda yz.(\lambda w.w)z)\ v)\ (((\lambda yz.(\lambda w.w)z)\ v)\ v)\\ \rightarrow\ \ (\lambda z.(\lambda w.w)z)\ (((\lambda yz.(\lambda w.w)z)\ v)\ v)\\ \rightarrow\ \ (\lambda w.w)\ (((\lambda yz.(\lambda w.w)z)\ v)\ v)\\ \rightarrow\ \ ((\lambda yz.(\lambda w.w)z)\ v)\ v)\\ \rightarrow\ \ (\lambda z.(\lambda w.w)z)\ v)\\ \rightarrow\ \ (\lambda w.w)\ v\\ \rightarrow\ \ v\end{array}$$

Appel par valeur

$$
\begin{array}{ll}
 & (\lambda x.x(xv))\ ((\lambda yz.(\lambda w.w)z)\ v) \\
\to & (\lambda x.x(xv))\ (\lambda z.(\lambda w.w)z) \\
\to & (\lambda z.(\lambda w.w)z)\ ((\lambda z.(\lambda w.w)z)\ v) \\
\to & (\lambda z.(\lambda w.w)z)\ ((\lambda w.w)v) \\
\to & (\lambda z.(\lambda w.w)z)\ v \\
\to & (\lambda w.w)v \\
\to & v
\end{array}
$$

L'ordre d'évaluation différent fait notamment que la radical $(\lambda y._)v$ n'est réduit qu'une seule fois en appel par valeur contre deux en appel par nom. Pour le reste on fait les mêmes opérations dans un ordre différent.

Appel par valeur, sémantique à grands pas : $\Downarrow_v$
On substitue l'argument évalué

$$\frac{}{\lambda x.t \ \Downarrow_v \ \lambda x.t}$$

$$\frac{t_1 \ \Downarrow_v \ \lambda x.t'_1 \qquad t_2 \ \Downarrow_v \ v_2 \qquad t'_1\{x \leftarrow v_2\} \ \Downarrow_v \ v}{t_1\ t_2 \ \Downarrow_v \ v}$$

$\Downarrow_v$: relation binaire terme/valeur $\{x \leftarrow v\}$: substitue des valeurs

Appel par nom, sémantique à grands pas : $\Downarrow_n$
On substitue l'argument non-évalué

$$\frac{}{\lambda x.t \ \Downarrow_n \ \lambda x.t}$$

$$\frac{t_1 \ \Downarrow_n \ \lambda x.t'_1 \qquad t'_1\{x \leftarrow t_2\} \ \Downarrow_n \ v}{t_1\ t_2 \ \Downarrow_n \ v}$$

$\Downarrow_n$: relation binaire terme/valeur $\{x \leftarrow t\}$: substitue des termes

Exercice : évaluation en sémantique à grands pas
Évaluer en appel par nom et en appel par valeur les termes suivants

— $(\lambda x.x(xv))\ ((\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b))$
— $(\lambda xy.x)\ v\ ((\lambda z.zz)(\lambda z.zz))$

Comparer les arbres de dérivation obtenus

Correction

— Appel par nom. Note : un même sous-arbre apparaît deux fois. Cela traduit deux évaluations d'un sous-terme dupliqué.

$$\dfrac{\dfrac{}{\lambda x.x(xv)\ \Downarrow_n\ \lambda x.x(xv)} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\lambda yz.z\ \Downarrow_n\ \lambda yz.z} \quad \dfrac{}{\lambda z.z\ \Downarrow_n\ \lambda z.z}}{(\lambda yz.z)\ (\lambda a.a)\ \Downarrow_n\ \lambda z.z} \quad \dfrac{}{\lambda b.b\ \Downarrow_n\ \lambda b.b}}{(\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b)\ \Downarrow_n\ \lambda b.b} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\lambda yz.z\ \Downarrow_n\ \lambda yz.z} \quad \dfrac{}{\lambda z.z\ \Downarrow_n\ \lambda z.z}}{(\lambda yz.z)\ (\lambda a.a)\ \Downarrow_n\ \lambda z.z} \quad \dfrac{}{\lambda b.b\ \Downarrow_n\ \lambda b.b}}{(\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b)\ \Downarrow_n\ \lambda b.b} \quad \dfrac{}{v\ \Downarrow_v\ v}}{((\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b))\ v\ \Downarrow_n\ v}}{((\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b))((\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b))\ v)\ \Downarrow_n\ v}}{(\lambda x.x(xv))\ ((\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b))\ \Downarrow_n\ v}$$

Appel par valeur. Pas de duplication ici.

$$\dfrac{\dfrac{}{\lambda x.x(xv)\ \Downarrow_v\ \lambda x.x(xv)} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\lambda yz.z\ \Downarrow_v\ \lambda yz.z} \quad \dfrac{}{\lambda a.a\ \Downarrow_v\ \lambda a.a} \quad \dfrac{}{\lambda z.z\ \Downarrow_v\ \lambda z.z}}{(\lambda yz.z)\ (\lambda a.a)\ \Downarrow_v\ \lambda z.z} \quad \dfrac{}{\lambda b.b\ \Downarrow_v\ \lambda b.b} \quad \dfrac{}{\lambda b.b\ \Downarrow_v\ \lambda b.b}}{(\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b)\ \Downarrow_v\ \lambda b.b} \quad \dfrac{\dfrac{}{\lambda b.b\ \Downarrow_v\ \lambda b.b} \quad \dfrac{\dfrac{\dfrac{}{\lambda b.b\ \Downarrow_v\ \lambda b.b} \quad \dfrac{}{v\ \Downarrow_v\ v} \quad \dfrac{}{v\ \Downarrow_v\ v}}{(\lambda b.b)v\ \Downarrow_v\ v} \quad \dfrac{}{v\ \Downarrow_v\ v}}{(\lambda b.b)((\lambda b.b)v)\ \Downarrow_v\ v}}{(\lambda x.x(xv))\ ((\lambda yz.z)\ (\lambda a.a)\ (\lambda b.b))\ \Downarrow_v\ v}$$

— Appel par nom. Ω est défaussé, tout se passe bien.

$$\dfrac{\dfrac{\dfrac{}{\lambda xy.x\ \Downarrow_n\ \lambda xy.x} \quad \dfrac{}{\lambda y.v\ \Downarrow_n\ \lambda y.v}}{(\lambda xy.x)\ v\ \Downarrow_n\ \lambda y.v} \quad \dfrac{}{v\ \Downarrow_n\ v}}{(\lambda xy.x)\ v\ \Omega\ \Downarrow_n\ v}$$

Appel par valeur. On n'arrive pas à construire un arbre fini : pour justifier $\Omega\ \Downarrow_v ??$ on aurait besoin d'une prémisse $\Omega\ \Downarrow_v ??$.

$$\dfrac{\dfrac{\dfrac{}{\lambda xy.x\ \Downarrow_v\ \lambda xy.x} \quad \dfrac{}{v\ \Downarrow_v\ v} \quad \dfrac{}{\lambda y.v\ \Downarrow_v\ \lambda y.v}}{(\lambda xy.x)\ v\ \Downarrow_v\ \lambda y.v} \quad \dfrac{\dfrac{\dfrac{}{\lambda z.zz\ \Downarrow_v\ \lambda z.zz} \quad \dfrac{}{\lambda z.zz\ \Downarrow_v\ \lambda z.zz} \quad \dfrac{\dots}{\Omega\ \Downarrow_v ??}}{\Omega\ \Downarrow_v ??} \quad \dfrac{}{v\{y \leftarrow ??\}\ \Downarrow_v ??}}{(\lambda xy.x)\ v\ \Omega\ \Downarrow_n ??}$$

Exercice : propriétés de la sémantique à grands pas
L'évaluation produit des valeurs

$$\textit{Si } t \Downarrow_v v \textit{ alors } v \textit{ est une valeur}$$
$$\textit{De plus, si } t \textit{ est clos alors } v \textit{ l'est également}$$

L'évaluation est déterministe

$$\textit{Si } t \Downarrow_v v \textit{ et } t \Downarrow_v v' \textit{ alors } v = v'$$

Correction

— Si $t \Downarrow_v v$ alors v est une valeur : par récurrence sur $t \Downarrow_v v$

— Également par récurrence sur $t \Downarrow_v v$: si $t \Downarrow_v v$ alors $\mathsf{fv}(v) \subseteq \mathsf{fv}(t)$, d'où t clos implique v clos.

— Si $t \Downarrow_v v$ et $t \Downarrow_v v'$ alors $v = v'$: par récurrence sur $t \Downarrow_v v$, en raisonnant à chaque cas sur les différentes formes que peut prendre $t \Downarrow_v v'$.

6 Interprétation

Sémantique appel par valeur à grands pas, en caml
Définition des valeurs

```
type valeur = VAbs of string * terme
```

Fonction d'évaluation

```
let rec eval = function
  | Var _ -> assert false
  | Abs(x, t) -> VAbs(x, t)
  | App(t1, t2) -> let VAbs(x,t1') = eval t1 in
                   let v2 = eval t2 in
                   eval (subst t1' x t2)
```

Un environnement pour éviter les substitutions
Définition des valeurs et des environnements

```
module Env = Map.Make(String)
type valeur = Fermeture of string * terme * env
and  env = valeur Env.t
```

Fonction d'évaluation

```
let rec eval t env = match t with
  | Var(x) -> Env.find x env
  | Abs(x, t) -> Fermeture(x, t, env)
  | App(t1, t2) ->
    let Fermeture(x, t1', env') = eval t1 env in
    let v2 = eval t2 env in
    eval t1' (Env.add x v2 env')
```

Sémantique à grands pas, avec un environnement
Environnements : Γ

— fonction des variables vers les valeurs

Valeurs : fermetures $\langle \lambda x.t, \Gamma \rangle$
Règles

$$\frac{\Gamma(x) = v}{x \Downarrow_v^\Gamma v} \qquad \frac{}{\lambda x.t \Downarrow_v^\Gamma \langle \lambda x.t, \Gamma \rangle}$$

$$\frac{t_1 \Downarrow_v^\Gamma \langle \lambda x.t_1', \Gamma' \rangle \qquad t_2 \Downarrow_v^\Gamma v_2 \qquad t_1' \Downarrow_v^{\Gamma', x:v_2} v}{t_1\ t_2 \Downarrow_v^\Gamma v}$$

7 Équivalence des sémantiques opérationnelles

Équivalence petits pas / grands pas
Les présentations à petits pas et à grands pas de la sémantique opérationnelle sont équivalentes

$$t \to_v^* v \quad \text{si et seulement si} \quad t \Downarrow_v v$$

et

$$t \to_n^* v \quad \text{si et seulement si} \quad t \Downarrow_n v$$

$t \Downarrow_v v$ **implique** $t \to_v^* v$
Preuve par récurrence sur la dérivation de $t \Downarrow_v v$

— Cas $\lambda x.t \Downarrow_v \lambda x.t$. On a bien $\lambda x.t \to_v^* \lambda x.t$ en zéro étapes

— Cas $t_1\ t_2 \Downarrow_v v$ avec $t_1 \Downarrow_v \lambda x.t_1'$ et $t_2 \Downarrow_v v_2$ et $t_1'\{x \leftarrow v_2\} \Downarrow_v v$ Alors

$$\begin{array}{lll} t_1\ t_2 & & \\ \to_v^* & (\lambda x.t_1')\ t_2 & \text{car } t_1 \to_v^* \lambda x.t_1' \text{ par HR} \\ \to_v^* & (\lambda x.t_1')\ v_2 & \text{car } t_2 \to_v^* v_2 \text{ par HR} \\ \to & t_1'\{x \leftarrow v_2\} & \\ \to_v^* & v & \text{par hypothèse de récurrence} \end{array}$$

Lemmes faciles :
si $a \to_v^* a'$ *alors* $ab \to_v^* a'b$
si $a \to_v^* a'$ *alors* $va \to_v^* va'$

$t \to_v^* v$ **implique** $t \Downarrow_v v$
Par récurrence sur la longueur de la séquence $t \to_v^* v$

— Cas 0 Alors $t = v$, et en particulier t est une valeur Donc on a $t = \lambda x.t_0$, et $\lambda x.t_0 \Downarrow_v \lambda x.t_0$

— Cas $n+1$, en supposant que pour toute séquence de réduction $t' \to_v^* v$ de longueur n on a bien $t' \Downarrow_v v$
Alors la séquence $t \to_v^* v$ de longueur n+1 se décompose en $t \to_v t' \to_v^* v$ où $t' \to_v^* v$ est de longueur n.
On a donc $t \to_v t' \Downarrow_v v$
On conclut avec un lemme

$$t \to_v t' \Downarrow_v v \quad \textit{implique} \quad t \Downarrow_v v$$

$t \to_v t' \Downarrow_v v$ **implique** $t \Downarrow_v v$
Par récurrence sur la dérivation de $t \to_v t'$

— Cas $t = (\lambda x.t_0)v_0 \to_v t_0\{x \leftarrow v_0\} = t'$
Par évaluation on a $\lambda x.t_0 \Downarrow_v \lambda x.t_0$ et $v_0 \Downarrow_v v_0$ et par hypothèse $t_0\{x \leftarrow v_0\} \Downarrow_v v_0$
donc $(\lambda x.t_0)v_0 \Downarrow_v v$

— Cas $t = t_1 t_2 \to_v t_1' t_2$ avec $t_1 \to_v t_1'$
Par hypothèse on a $t_1' t_2 \Downarrow_v v$ Alors par *inversion* on obtient $t_1' \Downarrow_v \lambda x.t_1''$ et $t_2 \Downarrow_v v_2$ et $t_1''\{x \leftarrow v_2\} \Downarrow_v v$
On a donc $t_1 \to_v t_1' \Downarrow_v \lambda x.t_1''$, d'où par hypothèse de récurrence $t_1 \Downarrow_v \lambda x.t_1''$
Finalement $t = t_1 t_2 \Downarrow_v v$

— *Cas* $t = v_1 t_2 \to_v v_1 t_2'$ *avec* $t_2 \to_v t_2'$ *similaire*

À propos des inversions
Un fait $t \Downarrow_v v$ ne peut être justifié que par un arbre de dérivation utilisant les règles d'inférence pour $\Downarrow_v$
L'*inversion* est un raisonnement par cas sur la dernière règle

— soit la dernière règle utilisée est celle des valeurs, et alors $t = \lambda x.t_0$ et $v = t$

— soit la dernière règle utilisée est celle des applications, et alors $t = t_1 t_2$ avec $t \Downarrow_v \lambda x.t_1$ et $t_2 \Downarrow_v v_2$ et $t_1\{x \leftarrow v_2\} \Downarrow_v v$

Adaptable à toute relation définie par des règles d'inférence

8 Extensions

Extensions : arithmétique
Nouvelle forme de valeurs

— constantes entières n

Nouvelles règles

$$\frac{}{n \Downarrow_v n} \qquad \frac{t_1 \Downarrow_v n_1 \qquad t_2 \Downarrow_v n_2 \qquad n_1 + n_2 = n}{t_1 \oplus t_2 \Downarrow_v n}$$

Extension arithmétique, en caml

```
type terme = ...
  | Cst of int
  | Add of terme * terme


type valeur = ...
  | VCst of int


let rec eval t env = ...
  | Cst n -> VCst n
  | Add(t1, t2) ->
    let VCst n1 = eval t1 env in
    let VCst n2 = eval t2 env in
    VCst(n1+n2)
```

Extensions : booléens

Nouvelle forme de valeurs

— constantes booléennes T et F

Nouvelles règles

$$\frac{}{\mathsf{T} \Downarrow_v \mathsf{T}} \qquad \frac{}{\mathsf{F} \Downarrow_v \mathsf{F}}$$

$$\frac{t \Downarrow_v 0}{\mathsf{isZero}(t) \Downarrow_v \mathsf{T}} \qquad \frac{t \Downarrow_v n \qquad n \neq 0}{\mathsf{isZero}(t) \Downarrow_v \mathsf{F}}$$

$$\frac{t_1 \Downarrow_v \mathsf{T} \qquad t_2 \Downarrow_v v_2}{\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \Downarrow_v v_2} \qquad \frac{t_1 \Downarrow_v \mathsf{F} \qquad t_3 \Downarrow_v v_3}{\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \Downarrow_v v_3}$$

Extensions : paires

Nouvelle forme de valeurs

— paires de valeurs $\langle v_1, v_2 \rangle$

Nouvelles règles

$$\frac{t_1 \Downarrow_v v_1 \qquad t_2 \Downarrow_v v_2}{\langle t_1, t_2 \rangle \Downarrow_v \langle v_1, v_2 \rangle}$$

$$\frac{t \Downarrow_v \langle v_1, v_2 \rangle}{\pi_1(t) \Downarrow_v v_1} \qquad \frac{t \Downarrow_v \langle v_1, v_2 \rangle}{\pi_2(t) \Downarrow_v v_2}$$

Extensions : point fixe
Nouvelle règle

$$\frac{t \Downarrow_v \lambda x.u \qquad u\{x \leftarrow \mathsf{Fix}(t)\} \Downarrow_v v}{\mathsf{Fix}(t) \Downarrow_v v}$$

Fait rare, on a une prémisse portant sur un terme $u\{x \leftarrow \mathsf{Fix}(t)\}$ qui n'est pas un sous-terme du terme $\mathsf{Fix}(t)$ présent en conclusion. C'est ce qui rend possible la non-terminaison

Exercice : extension listes chaînées
Définir une notion de valeur et des règles de sémantique à grands pas en appel par valeur pour

le λ-calcul étendu avec des listes chaînées

Rappel des termes de cette extension

$$\begin{array}{rcl} t & ::= & \ldots \\ & | & \mathsf{Nil} \\ & | & t_1 :: t_2 \\ & | & \mathsf{isNil}(t) \\ & | & \mathsf{hd}(t) \\ & | & \mathsf{tl}(t) \end{array}$$

Correction

— Nouvelles formes de valeurs : listes de valeurs Nil et $v_1 :: v_2$

— Nouvelles règles

$$\frac{}{\mathsf{Nil} \Downarrow_v \mathsf{Nil}} \qquad \frac{t_1 \Downarrow_v v_1 \qquad t_2 \Downarrow_v v_2}{t_1 :: t_2 \Downarrow_v v_1 :: v_2} \qquad \frac{t \Downarrow_v \mathsf{Nil}}{\mathsf{isNil}(t) \Downarrow_v \mathsf{T}} \qquad \frac{t \Downarrow_v v_1 :: v_2}{\mathsf{isNil}(t) \Downarrow_v \mathsf{F}}$$

$$\frac{t \Downarrow_v v_1 :: v_2}{\mathsf{hd}(t) \Downarrow_v v_1} \qquad \frac{t \Downarrow_v v_1 :: v_2}{\mathsf{tl}(t) \Downarrow_v v_2}$$

DM - *à rédiger et à me renvoyer avant le prochain cours*

1. Démontrer que le λ-calcul pur est *localement confluent* *sans utiliser le théorème de confluence*
2. (a) Définir une sémantique opérationnelle à petits pas pour le λ-calcul étendu avec des paires (en appel par valeur)

   (b) Tenter d'évaluer le terme suivant

   $$(\lambda x.\pi_1(\pi_2(x)))\ ((\lambda y.\langle y, (\lambda z.z)2\rangle)4)$$

   à l'aide des sémantiques à petits pas et à grands pas. Qu'est-ce qui différencie les deux tentatives ?