Nécessité faite loi

Alexis Saurin & Pierre-Marie Pédrot

10 janvier 2014

JFLAs 2014

Plan de l'exposé

- Survol : l'appel par nécessité
- La réduction linéaire de tête
- Une construction (plus canonique) pas à pas de l'appel par nécessité
- Conclusion

Pour s'échauffer

Une tension entre appel par nom et appel par valeur

$$\Delta = \lambda x.(x)x, I = \lambda y.y$$

Calculs inutiles en appel par valeurs :

$$M = (\lambda x.I)(\Delta)\Delta \rightarrow_{CBN} I$$

$$M = (\lambda x.I)(\Delta)\Delta \rightarrow_{CBV} M \rightarrow_{CBV} M \rightarrow_{CBV} \dots$$

Calculs redondants en appel par nom :

$$N = (\Delta)(I)I \rightarrow_{CBN} (I)I(I)I \rightarrow_{CBN} (I)(I)I \rightarrow_{CBN} (I)I \rightarrow_{CBN} I$$

$$N = (\Delta)(I)I \rightarrow_{CBV} (\Delta)I \rightarrow_{CBN} (I)I \rightarrow_{CBN} I$$

L'évaluation paresseuse, informellement

- John Reynolds : *Ideally, one would like to have one's cake and eat it too : to postpone evaluating an expression (...) until it is clear that its value is really needed, but also to avoid repeated evaluation.*
- Watt & Findlay : *We evaluate the actual parameter when the argument is first needed ; then we store the argument for use whenever it is subsequently needed.*
- Danvy & al. : *Demand-driven computation and memoization of intermediate results.*

L'évaluation paresseuse, informellement

- John Reynolds : *Ideally, one would like to have one's cake and eat it too : to postpone evaluating an expression (...) until it is clear that its value is really needed, but also to avoid repeated evaluation.*
- Watt & Findlay : *We evaluate the actual parameter when the argument is first needed ; then we store the argument for use whenever it is subsequently needed.*
- Danvy & al. : *Demand-driven computation and memoization of intermediate results.*

Une optimisation de l'appel par nom

- Induit la même équivalence observationnelle que l'appel par nom.
- Mais optimise l'appel par nom.
- À voir également comme variante de l'appel par valeur qui ne substitue les valeurs que de manière paresseuse.

Pourtant...

- Plusieurs versions de l'appel par nécessité, comme machines ou comme calcul.
- L'implémentation du calcul n'est pas évidente et les machines usuelles ont recours à un tas pour stocker les valeurs intermédiaires.
- Problématique du contrôle : call-by-need diffère de l'appel par nom en présence de contrôle.

Objectifs de l'exposé

L'appel par nécessité est une réduction linéaire de tête faible en appel par valeur avec partage de clôture

- Faire comprendre call-by-need, la réduction linéaire de tête et les liens qu'ils entretiennent ;
- Reconstruire un λ-calcul en appel par nécessité à partir de la réduction linéaire de tête ;
- Au cours de cette reconstruction, faire apparaître trois éléments caractéristiques de call-by-need :
  - calcul à la demande ;
  - mémoisation des résultats intermédiaires ;
  - partage de clôtures ;

Exemple de calcul par nécessité

(Système d'Ariola-Felleisen, JFP 97)

$$
\begin{array}{lll}
N & = & (\Delta)(I)I \\
 & = & (\lambda x.(x)x)\ (I)I \\
 & \rightarrow_{deref} & (\lambda x.(x)x)\ (\lambda y.I)\ I \\
 & \rightarrow_{assoc} & (\lambda y.(\lambda x.(x)x)\ I)\ I \\
 & \rightarrow_{deref} & (\lambda y.(\lambda x.(\lambda z.z)\ x)\ I)\ I \\
 & \rightarrow_{deref} & (\lambda y.(\lambda x.(\lambda z.z)\ I)\ I)\ I \\
 & \rightarrow_{deref} & (\lambda y.(\lambda x.(\lambda z.\blacksquare)\ I)\ I)\ I \\
 & \rightarrow^{\star}_{gc} & \blacksquare
\end{array}
$$

Différences entre appel par valeur et par nécessité

En appel par nécessité, on a :

- substitution linéaire des valeurs ;
- on réduit des radicaux qui ne sont pas (encore) des radicaux β ;
- on a “moins” de β-radicaux disponibles car contrainte de nécessité sur le contexte d'évaluation.

Réduction linéaire de tête

La machine de Krivine

Clôtures	c	$::=$	(t, σ)
Environnements	σ	$::=$	$\emptyset \mid \sigma + (x := c)$
Piles	π	$::=$	$\varepsilon \mid c \cdot \pi$
Processus	p	$::=$	$\langle c \mid \pi \rangle$

PUSH	$\langle ((t)\, u, \sigma) \mid \pi \rangle$	$\to$	$\langle (t, \sigma) \mid (u, \sigma) \cdot \pi \rangle$
POP	$\langle (\lambda x.\, t, \sigma) \mid c \cdot \pi \rangle$	$\to$	$\langle (t, \sigma + (x := c)) \mid \pi \rangle$
GRAB	$\langle (x, \sigma + (x := c)) \mid \pi \rangle$	$\to$	$\langle c \mid \pi \rangle$
GARBAGE	$\langle (x, \sigma + (y := c)) \mid \pi \rangle$	$\to$	$\langle (x, \sigma) \mid \pi \rangle$

La machine de Krivine

Clôtures	c	$::=$	(t, σ)
Environnements	σ	$::=$	$\emptyset \mid \sigma + (x := c)$
Piles	π	$::=$	$\varepsilon \mid c \cdot \pi$
Processus	p	$::=$	$\langle c \mid \pi \rangle$

Push	$\langle ((t)\, u, \sigma) \mid \pi \rangle$	$\to$	$\langle (t, \sigma) \mid (u, \sigma) \cdot \pi \rangle$
Pop	$\langle (\lambda x.\, t, \sigma) \mid c \cdot \pi \rangle$	$\to$	$\langle (t, \sigma + (x := c)) \mid \pi \rangle$
Grab	$\langle (x, \sigma + (x := c)) \mid \pi \rangle$	$\to$	$\langle c \mid \pi \rangle$
Garbage	$\langle (x, \sigma + (y := c)) \mid \pi \rangle$	$\to$	$\langle (x, \sigma) \mid \pi \rangle$

Est-ce vraiment la réduction de tête faible ?

La machine de Krivine

Clôtures	c	$::=$	(t, σ)
Environnements	σ	$::=$	$\emptyset \mid \sigma + (x := c)$
Piles	π	$::=$	$\varepsilon \mid c \cdot \pi$
Processus	p	$::=$	$\langle c \mid \pi \rangle$

PUSH	$\langle ((t)\, u, \sigma) \mid \pi \rangle$	$\to$	$\langle (t, \sigma) \mid (u, \sigma) \cdot \pi \rangle$
POP	$\langle (\lambda x.\, t, \sigma) \mid c \cdot \pi \rangle$	$\to$	$\langle (t, \sigma + (x := c)) \mid \pi \rangle$
GRAB	$\langle (x, \sigma + (x := c)) \mid \pi \rangle$	$\to$	$\langle c \mid \pi \rangle$
GARBAGE	$\langle (x, \sigma + (y := c)) \mid \pi \rangle$	$\to$	$\langle (x, \sigma) \mid \pi \rangle$

Est-ce vraiment la réduction de tête faible ?

Simuler n'est pas implémenter.

La σ-équivalence

(Danos & Regnier, ≈ 1990)

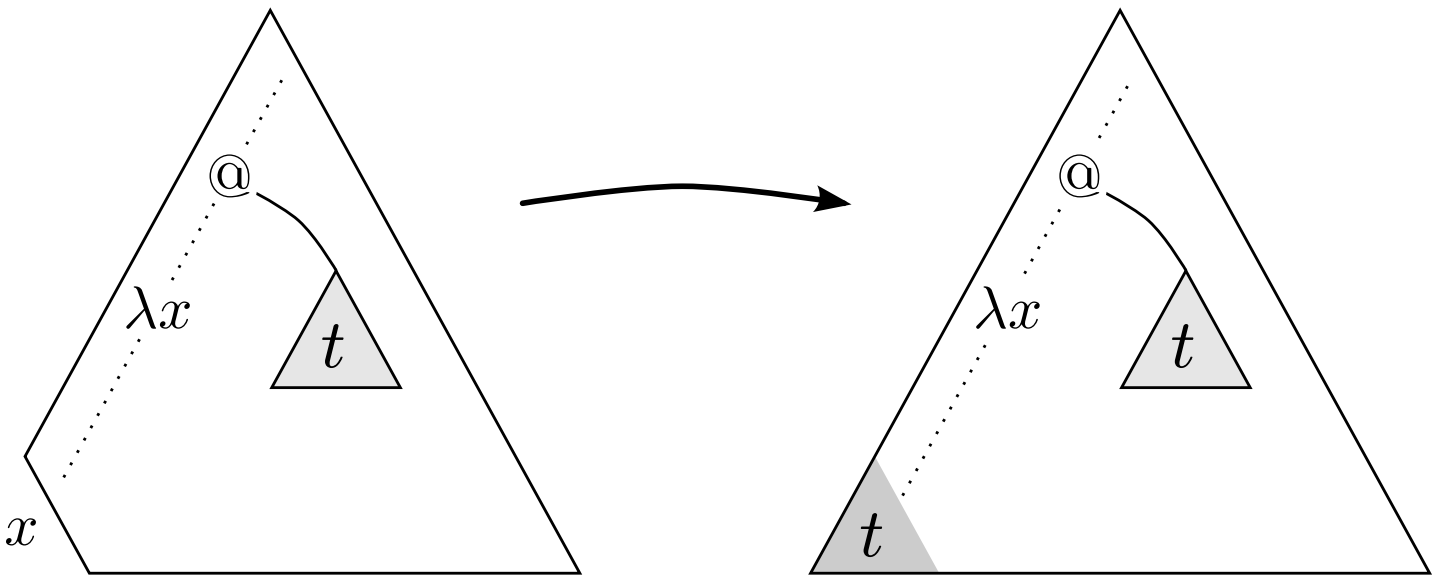
Une relation qui capture ce type de comportement.

$$\begin{array}{lcl} (\lambda x.(M)N_2)N_1 & =_\sigma & (\lambda x.M)N_1\ N_2 \\ (\lambda x_1.\lambda x_2.M)N & =_\sigma & \lambda x_2.(\lambda x_1.M)N \end{array}$$

- $\rightsquigarrow$ Inspirée par les commutations dans les réseaux MELL
- $\rightsquigarrow$ Saute des radicaux ignorés par la KAM

Réduction linéaire de tête

(Danos & Regnier, ≈ 1990)



Réduction linéaire de tête

Définition (Regnier 1990, Danos & Regnier, 2004)

- *λ de tête*, $\lambda_h(M)$: $\lambda_h(\lambda x.M) = \lambda x$; $\lambda_h((\lambda x.M)NN_1 \dots N_k) = \lambda_h((M)N_1 \dots N_k)$.
- *Radical premier* : $(\lambda x, N) \in pr(T)$ si T a un sous-terme $(M)N$ tel que $\lambda_h(M) = \lambda x$.
- *Réduction linéaire de tête* : on dit que M se réduit linéairement de tête sur M' s'il existe un radical premier $(\lambda x, N)$ tel que λx lie la variable hoc de M et si M' est obtenu à partir de M en remplaçant la variable hoc par N.

À σ-équivalence près, la réduction linéaire de tête est la réduction de tête usuelle, linéarisée.

Réduction linéaire de tête

Exemple

$$
\begin{array}{lll}
N & = & (\Delta)(I)I \\
 & = & (\lambda x.(x)x)(I)I \\
 & \to & (\lambda x.(\lambda x'.x')Ix)(I)I \\
 & \to & (\lambda x.(\lambda x'.\lambda x''.x'')Ix)(I)I \\
 & \to & (\lambda x.(\lambda x'.\lambda x''.x)Ix)(I)I \\
 & \to & (\lambda x.(\lambda x'.\lambda x''.(\lambda x'''.x''')I)Ix)(I)I \\
 & \to & (\lambda x.(\lambda x'.\lambda x''.(\lambda x'''.I)I)Ix)(I)I \\
 & \to_{gc}^{\star} & I
\end{array}
$$

Différences entre appel par nom et réduction linéaire de tête

En réduction linéaire de tête, on a :

- substitution **linéaire** des arguments
- on réduit des radicaux qui ne sont pas (encore) des radicaux β (grâce aux radicaux premiers et à la σ-équivalence) ;
- on a “moins” de β-radicaux disponibles car on ne réduit que le radical premier correspondant à la variable hoc.

La réduction linéaire de tête comme un calcul

L'identification des radicaux premiers peut se faire au moyen d'une grammaire de contextes :

$$\mathscr{C} ::= \Box \mid (\mathscr{C}[\lambda x.\mathscr{C}])M$$

λ_{lhr} :

terme	M	$::=$	$x \mid \lambda x.M \mid (M)M$
contexte de clôtures	$\mathscr{C}$	$::=$	$\Box \mid (\mathscr{C}[\lambda x.\mathscr{C}])M$
contexte d'évaluation gauche	E	$::=$	$\Box \mid (E)M \mid \lambda x.E$

$$(\beta_{lhr}) \quad E[(\mathscr{C}[\lambda x.E'[x]])M] \rightarrow E[(\mathscr{C}[\lambda x.E'[M]])M]$$

Construction pas à pas de l'appel par nécessité

Vers call-by-need (I)

Réduction linéaire de tête faible

On peut définir une notion de réduction linéaire de tête faible, qui ne réduit pas sous les λ, en restreignant les contextes d'évaluation de λ_{lhr} :

$$E ::= \Box \mid (E)M \mid \cancel{\lambda x.M}$$

Vers call-by-need (I)

Réduction linéaire de tête faible

On peut définir une notion de réduction linéaire de tête faible, qui ne réduit pas sous les λ, en restreignant les contextes d'évaluation de λ_{lhr} :

$$E ::= \Box \mid (E)M \mid \cancel{\lambda x.M}$$

Trop restrictif : il faut reformuler β_{lhr}.

- Autoriser des abstractions dans E_1 et E_2 si impliquées dans des radicaux premiers :

$$E_1[\mathscr{C}_1[(\mathscr{C}[\lambda x.\mathscr{C}_2[E_2[x]]])M]]$$

- $\mathscr{C}$ insuffisant : radicaux premiers entre lieurs dans E_2 et arguments dans E_1

Vers call-by-need (I)

Réduction linéaire de tête faible

$\mathscr{C}^{\lambda}$ et $\mathscr{C}^{@}$:

$$\begin{array}{rcl} \mathscr{C}^{@} & ::= & \Box \mid (\mathscr{C}[\mathscr{C}^{@}])M \\ \mathscr{C}^{\lambda} & ::= & \Box \mid \mathscr{C}[\lambda x.\mathscr{C}^{\lambda}] \end{array}$$

($\cong$ Applications et lieurs à clôtures près)

Ce qui donne la nouvelle réduction :

$$E[\mathscr{C}^{@}[(\mathscr{C}[\lambda x.\mathscr{C}^{\lambda}[E'[x]]])M]] \rightarrow_{\beta_{wlhr}} E[\mathscr{C}^{@}[(\mathscr{C}[\lambda x.\mathscr{C}^{\lambda}[E'[M]]])M]]$$

à condition que $\mathscr{C}^{@}[\mathscr{C}^{\lambda}]$ soit un contexte de clôtures.

Vers call-by-need (II)

Réduction linéaire de tête par valeur

On force *whlr* à ne substituer que des valeurs, obtenant λ_{wlhr^v} :

valeur	V	$::=$	$\lambda x.M$
cont. clôtures	$\mathscr{C}$	$::=$	$\Box \mid (\mathscr{C}[\lambda x.\mathscr{C}])M$
valeur close	W	$::=$	$\mathscr{C}[V]$
$\mathscr{C}^{@}$	$\mathscr{C}^{@}$	$::=$	$\Box \mid (\mathscr{C}[\mathscr{C}^{@}])M$
$\mathscr{C}^{\lambda}$	$\mathscr{C}^{\lambda}$	$::=$	$\Box \mid \mathscr{C}[\lambda x.\mathscr{C}^{\lambda}]$
contexte d'évaluation	E	$::=$	$\Box \mid (E)M \mid (\mathscr{C}[\lambda x.E'[x]])E$

Et la nouvelle réduction est :

$$E[\mathscr{C}^{@}[(\mathscr{C}[\lambda x.\mathscr{C}^{\lambda}[E'[x]]])W]] \rightarrow_{\beta^{v}_{wlhr}} E[\mathscr{C}^{@}[(\mathscr{C}[\lambda x.\mathscr{C}^{\lambda}[E'[W]]])W]]$$

à condition que $\mathscr{C}^{@}[\mathscr{C}^{\lambda}]$ soit un contexte de clôtures.

Vers call-by-need (III)

Réduction linéaire de tête faible par valeur avec partage de clôtures

- Dernière transformation sur le calcul : partage
- Le calcul précédent substitue des termes $\mathscr{C}[\lambda x.M]$
- D'où duplication des calculs dans $\mathscr{C}$

Vers call-by-need (III)

Réduction linéaire de tête faible par valeur avec partage de clôtures

- Dernière transformation sur le calcul : partage
- Le calcul précédent substitue des termes $\mathscr{C}[\lambda x.M]$
- D'où duplication des calculs dans $\mathscr{C}$

On ajoute une commutation de partage de contexte, $\lambda_{wlhr}{}^{vs}$:

$$\begin{array}{rl} & E[\mathscr{C}^{@}[(\mathscr{C}[\lambda x.\mathscr{C}^{\lambda}[E'[x]]])\mathscr{C}'[V]]] \\ \rightarrow_{\beta^{vs}_{wlhr}} & E[\mathscr{C}^{@}[\mathscr{C}'[(\mathscr{C}[\lambda x.\mathscr{C}^{\lambda}[E'[V]]])V]]] \end{array}$$

à condition que $\mathscr{C}^{@}[\mathscr{C}^{\lambda}]$ soit un contexte de clôtures.

Vers call-by-need (III)

Réduction linéaire de tête faible par valeur avec partage de clôtures

Le calcul ainsi obtenu est précisément le call-by-need de Felleisen et Chang, avec substitution linéaire.

La réduction linéaire de tête faible par valeur avec partage de clôtures est un λ-calcul en appel par nécessité.

Un autre chemin possible :

On aurait pu commencer par intégrer les règles de la σ-équivalence dans le calcul de la réduction linéaire de tête, on arrive alors au calcul d'Ariola et Felleisen.

Conclusion

- Présentation de la réduction linéaire de tête comme un calcul ;
- Dérivation canonique d'un call-by-need à partir de la réduction linéaire de tête :
  La réduction linéaire de tête faible qui ne substitue que des valeurs (et fait du partage de clôtures) est un calcul call-by-need
- Les règles de gestion des clôtures en call-by-need sont directement liées à la σ-équivalence.
  La règle $(\lambda x_1.\lambda x_2.M)N =_\sigma \lambda x_2.(\lambda x_1.M)N$ pourrait permettre de faire plus de partage de clôtures que les calculs actuels pour call-by-need tout en les rapprochant des réseaux.
- Lien avec le λ-caclul structurel ;
- Approfondir les liens entre LL et call-by-need ...

Merci !

« Paressons en toute chose,
hormis en aimant et en buvant,
hormis en paressant. »

D. Lessing