λ-calcul

Un langage simple

Syntaxe du λ-calcul

$$
\begin{array}{rcll}
e & ::= & \mathrm{x} & \text{référence de variable} \\
 & | & \lambda \mathrm{x}.\ e' & \lambda\text{-expression} \\
 & | & e'\ e'' & \text{appel de fonction}
\end{array}
$$

Conventions syntaxiques:

- Le contenu du corps d'une λ-expression inclut tout ce qui suit le point à moins qu'une parenthèse fermante ne vienne le terminer.
- L'opérateur "application de fonction" est associatif à gauche. I.e., $e'\ e''\ e''' \equiv (e'\ e'')\ e'''$.

Définitions

Définition d'une variable

La *définition* d'une variable 'x' (en une expression e_l) est la plus proche expression $e_{l'}$ qui englobe e_l telle que $e_{l'} = (\lambda_{l'}\mathrm{x}.\ e_{l''})$.

On dit alors que 'x' (en l'expression e_l) est liée.

e_l n'a pas à être une référence à 'x' pour qu'on puisse parler de la définition de 'x'.

Il faut mentionner l'expression d'où on recherche la définition d'une variable car, en général, plus d'une variable de nom 'x' peuvent apparaître dans la même expression.

Ex:

$$
\begin{array}{l}
({}_1(\lambda_2\mathrm{x}.\ (\lambda_3\mathrm{y}.\\
\quad ({}_4({}_5\mathrm{x}_6\ \mathrm{y}_7)\ (\lambda_8\mathrm{y}.\ \mathrm{y}_9))))\\
\ \mathrm{x}_{10})
\end{array}
$$

$$
\begin{array}{lcl}
\text{déf. de '}\mathrm{x}\text{' en } e_l & = & l', \quad \text{si } e_{l'} = (\lambda_{l'}\mathrm{x}.\ e_l) \\
\text{déf. de '}\mathrm{x}\text{' en } e_l & = & \text{déf. de '}\mathrm{x}\text{' en } e_{l'}, \quad \text{si } e_{l'} = ({}_{l'}e_l\ e_{l''}) \\
\text{déf. de '}\mathrm{x}\text{' en } e_l & = & \text{déf. de '}\mathrm{x}\text{' en } e_{l'}, \quad \text{si } e_{l'} = ({}_{l'}e_{l''}\ e_l)
\end{array}
$$

Définitions

Portée d'une définition

La *portée* de la définition $(\lambda_l \mathrm{x}.\ e_{l''})$ d'une variable 'x' est l'ensemble des expressions $e_{l'}$ telles que la définition de 'x' en $e_{l'}$ est e_l.

Variable libre

Une variable 'x', à l'intérieur d'un λ-terme e_0, est libre en une expression e_l si, en e_l, 'x' n'a pas de définition.

On dit aussi que 'x' est une variable libre de e_0.

On peut calculer l'ensemble des variables libres d'un λ-terme à l'aide de FV.

$$\begin{aligned} \mathrm{FV}(\mathrm{x}) &= \{\mathrm{x}\} \\ \mathrm{FV}(\lambda \mathrm{x}.\ e) &= \mathrm{FV}(e) - \{\mathrm{x}\} \\ \mathrm{FV}(e\ e') &= \mathrm{FV}(e) \cup \mathrm{FV}(e') \end{aligned}$$

Définitions

Substitution de variable

La *substitution* d'une variable par une valeur dans un λ-terme consiste à remplacer les apparitions de la variable qui sont dans la portée de la substitution par la valeur. En particulier, la valeur peut être un λ-terme.

On dénote par $e[\mathrm{x} \mapsto v]$ la substitution de 'x' par v dans e.

$$\frac{}{\mathrm{x}[\mathrm{x}{\mapsto}v]{=}v} \qquad \frac{\mathrm{x}{\not\equiv}\mathrm{y}}{\mathrm{y}[\mathrm{x}{\mapsto}v]{=}\mathrm{y}}$$

$$\frac{}{(\lambda \mathrm{x}.\ e)[\mathrm{x}{\mapsto}v]{=}\lambda \mathrm{x}.\ e} \qquad \frac{\mathrm{x}{\not\equiv}\mathrm{y} \quad \mathrm{y}{\notin}\mathrm{FV}(v)}{(\lambda \mathrm{y}.\ e)[\mathrm{x}{\mapsto}v]{=}\lambda \mathrm{y}.\ (e[\mathrm{x}{\mapsto}v])}$$

$$\frac{}{(e\ e')[\mathrm{x}{\mapsto}v]{=}(e[\mathrm{x}{\mapsto}v])\ (e'[\mathrm{x}{\mapsto}v])}$$

Sémantique du λ-calcul

Nous allons donner la *sémantique concrète* du λ-calcul.

On dit aussi que l'on décrit les règles d'*évaluation* (concrète) des expressions ou l'*interprétation concrète* du λ-calcul.

En fait, nous allons donner *trois* sémantiques concrètes différentes au λ-calcul: une pour l'évaluation dans un ordre quelconque, une pour l'évaluation paresseuse, une pour l'évaluation stricte.

La méthodologie employée pour spécifier la sémantique est celle de la sémantique *par petits pas* (*small-step semantics*) basée sur les *contextes d'évaluation*.

Chaque petit pas est effectué à l'aide d'une *réduction*.

Réductions

En λ-calcul, l'évaluation se fait à l'aide de réductions. Les deux réductions dont nous aurons besoin sont l'α-*réduction*, qui est l'équivalent du renommage de variable, et la β-*réduction*, qui est l'équivalent de l'appel de fonction.

L'α-réduction consiste à effectuer la réécriture suivante:

$$\lambda \mathrm{x}.\ e \overset{\alpha}{\mapsto} \lambda \mathrm{y}.\ (e[\mathrm{x} \mapsto \mathrm{y}])$$

quelque part dans le λ-terme global.

La β-réduction consiste à effectuer la réécriture suivante:

$$(\lambda \mathrm{x}.\ e)\ e' \overset{\beta}{\mapsto} e[\mathrm{x} \mapsto e']$$

quelque part dans le λ-terme global.

On considère que seule la β-réduction effectue un calcul réel. On appelle une expression sujette à réduction, comme $(\lambda \mathrm{x}.\ e)\ e'$, *expression réductible* ou *redex*.

La notion de "*quelque part*" est décidément floue.

Contextes d'évaluation

Un contexte d'évaluation représente un "reste du programme", au sens syntaxique, dans lequel il y a un emplacement distingué servant à contenir une expression.

$$C ::= [\![\cdot]\!] \mid \lambda \mathrm{x}.\ C \mid C\ e \mid e\ C$$

où e est une expression (ou un λ-terme).

Ex:

$$\begin{array}{rcl} (\lambda \mathrm{x}.\ \mathrm{x}\ \mathrm{x})\ (\lambda \mathrm{z}.\ \mathrm{y}) & \equiv & C_1[\![\mathrm{x}\ \mathrm{x}]\!] \text{ où } C_1 = (\lambda \mathrm{x}.\ [\![\cdot]\!])\ (\lambda \mathrm{z}.\ \mathrm{y}) \\ & \equiv & C_2[\![\lambda \mathrm{z}.\ \mathrm{y}]\!] \text{ où } C_2 = (\lambda \mathrm{x}.\ \mathrm{x}\ \mathrm{x})\ [\![\cdot]\!] \\ & \equiv & \ldots \end{array}$$

À l'aide des contextes, on peut exprimer formellement les α- et β-réductions sur un λ-terme entier:

- α-réduction: $C[\![\lambda \mathrm{x}.\ e]\!] \overset{\alpha}{\mapsto} C[\![\lambda \mathrm{y}.\ (e[\mathrm{x} \mapsto \mathrm{y}])]\!]$
- β-réduction: $C[\![(\lambda \mathrm{x}.\ e)\ e']\!] \overset{\beta}{\mapsto} C[\![e[\mathrm{x} \mapsto e']]\!]$

i.e. on n'a plus besoin de parler du "*quelque part*". Ceci est la 1ère sémantique.

Stratégies d'évaluation

La *stratégie d'évaluation* est la façon de choisir les emplacements où sont effectuées les β-réductions.

En utilisant les contextes d'évaluation C, on permet une évaluation dans un ordre quelconque. Ex:

$$\begin{array}{l}((\lambda \mathrm{c}.\ ((\lambda \mathrm{d}.\ \mathrm{d})\ (\lambda \mathrm{e}.\ \mathrm{e}))\ \mathrm{c})\ ((\lambda \mathrm{a}.\ \mathrm{a})\ \mathrm{b})) \\ \quad \#2 \quad \#1 \qquad\qquad\qquad\quad \#3\end{array}$$

1. On permet d'évaluer le corps des fonctions avant même qu'elles ne soient appelées.
2. On invoque une fonction sur un argument pas encore évalué.
3. Une fois passé, l'argument est évalué.

Au fait, qu'est-ce que c'est qu'être *évalué*? Ici, on considère qu'une λ-expression ou une variable pas encore substituée est une *valeur*.

Stratégies d'évaluation

L'évaluation en *ordre normal* est la stratégie d'évaluation qui correspond à la façon d'évaluer des langages paresseux:

- le corps d'une fonction n'est pas évalué avant que la fonction ne soit appelée;
- l'argument passé lors d'un appel n'a pas à être évalué.

On peut décrire précisément cette stratégie d'évaluation en définissant les contextes et les réductions ainsi:

$$\begin{array}{rcl} C^{\mathrm{n}} & ::= & [\![\cdot]\!] \mid C^{\mathrm{n}}\ e \\ C[\![\lambda \mathrm{x}.\ e]\!] & \overset{\alpha}{\mapsto} & C[\![\lambda \mathrm{y}.\ (e[\mathrm{x} \mapsto \mathrm{y}])]\!] \\ C^{\mathrm{n}}[\![(\lambda \mathrm{x}.\ e)\ e']\!] & \overset{\beta}{\mapsto} & C^{\mathrm{n}}[\![e[\mathrm{x} \mapsto e']]\!] \end{array}$$

Notez que l'α-réduction est toujours définie à l'aide des contextes C.

Ceci est la 2ème sémantique.

Stratégies d'évaluation

L'évaluation en *ordre applicatif* est la stratégie d'évaluation qui correspond à la façon d'évaluer des langages stricts:

- le corps d'une fonction n'est pas évalué avant que la fonction ne soit appelée;
- l'argument passé lors d'un appel doit avoir été évalué au préalable.

On peut décrire précisément cette stratégie d'évaluation en définissant les contextes, les valeurs et les réductions ainsi:

$$
\begin{array}{rcl}
C^{\mathrm{a}} & ::= & [\![\cdot]\!] \mid C^{\mathrm{a}}\ e \mid v\ C^{\mathrm{a}} \\
v & ::= & \mathrm{x} \mid \lambda \mathrm{x}.\ e \\
C[\![\lambda \mathrm{x}.\ e]\!] & \overset{\alpha}{\mapsto} & C[\![\lambda \mathrm{y}.\ (e[\mathrm{x} \mapsto \mathrm{y}])]\!] \\
C^{\mathrm{a}}[\![(\lambda \mathrm{x}.\ e)\ v]\!] & \overset{\beta}{\mapsto} & C^{\mathrm{a}}[\![e[\mathrm{x} \mapsto v]]\!]
\end{array}
$$

En outre, ces définitions forcent une évaluation de gauche à droite.

Ceci est la 3ème sémantique.

Stratégies d'évaluation

Exemple: expression qui est évaluée différemment selon que l'ordre normal ou l'ordre applicatif est utilisé.

$$((\lambda z.\ (\lambda x.\ x))\ ((\lambda f.\ (f\ f))\ (\lambda g.\ (g\ g))))$$