Un petit langage fonctionnel : FUN.

On se rapproche de notre but final en considérant un petit langage fonctionnel, nommé FUN.

On se donne l'ensemble des entiers relatifs $\mathbb{Z}$ et un ensemble infini de variables $\mathscr{V}$. L'ensemble des expressions de FUN, notées e, e' ou e_i, est défini par la grammaire suivante :

$$e ::= k \mid e_1 + e_2 \mid \underbrace{\texttt{fun}\, x \to e}_{\text{Fonction / Abstraction}} \mid \overbrace{e_1 \ e_2}^{\text{Application}} \mid x.$$

Note 1. On simplifie la notation par rapport à EA ou LEA : on ne souligne plus les entiers, on n'entoure plus les plus.

On notera de plus $e_1 \ e_2 \ e_3$ pour $(e_1 \ e_2) \ e_3$. Aussi, l'expression $\texttt{fun}\, x \ y \to e$ représentera l'expression $\texttt{fun}\, x \to (\texttt{fun}\, y \to e)$. On n'a pas le droit à plusieurs arguments pour une fonction, mais on applique la curryfication.

1 Sémantique opérationnelle « informellement ».

Exemple 1. Comment s'évalue $(\texttt{fun}\, x \to x + x)(7 + 7)$?

- ▷ D'une part, $7 + 7$ s'évalue en 14.
- ▷ D'autre part, $(\texttt{fun}\, x \to x + x)$ s'évalue en elle même.

- ▷ On procède à une substitution de $(x+x)[{}^{14}/_{x}]$ qui s'évalue en 28.

Exemple 2. Comment s'évalue l'expression

$$(\overbrace{(\texttt{fun}\, f \to \underbrace{(\texttt{fun}\, x \to x + (f\ \ x)))}_{B}}^{A}\ \underbrace{(\texttt{fun}\, y \to y + y)}_{C})\ 7\ ?$$

On commence par évaluer A et C qui s'évaluent en A et C respectivement. On continue en calculant la substitution

$$(\texttt{fun}\, x \to x + (f\ \ x))[{}^{\texttt{fun}\, y \to y + y}/_{f}],$$

ce qui donne

$$(\texttt{fun}\, x \to x + ((\texttt{fun}\, y \to y + y)\ \ x)).$$

Là, on **ne simplifie pas**, car c'est du code *dans* une fonction. On calcule ensuite la substitution

$$(x + ((\texttt{fun}\, y \to y + y)\ \ x))[{}^{7}/_{x}],$$

ce qui donne

$$7 + ((\texttt{fun}\, y \to y + y)\ \ 7).$$

On termine par la substitution

$$(y+y)[{}^{7}/_{y}] = 7 + 7.$$

On conclut que l'expression originelle s'évalue en 21.

Remarque 1. Dans FUN, le résultat d'un calcul (qu'on appellera *valeur*) n'est plus forcément un entier, ça peut aussi être une fonction.

L'ensemble des valeurs, notées v, est défini par la grammaire

$$v ::= k \mid \texttt{fun}\, x \to e.$$

Les fonctions sont des valeurs ! Et, le « contenu » la fonction n'est pas forcément une valeur.

On peut remarquer que l'ensemble des valeurs est un sous-ensemble des expressions de FUN.

2 Sémantique opérationnelle de FUN (version 1).

Définition 1. On définit l'ensemble des *variables libres* $\mathscr{V}\ell(e)$ d'une expression e par (on a 5 cas) :

- ▷ $\mathscr{V}\ell(x) = \{x\}$;
- ▷ $\mathscr{V}\ell(k) = \emptyset$;
- ▷ $\mathscr{V}\ell(e_1 + e_2) = \mathscr{V}\ell(e_1) \cup \mathscr{V}\ell(e_2)$;
- ▷ $\mathscr{V}\ell(e_1\ e_2) = \mathscr{V}\ell(e_1) \cup \mathscr{V}\ell(e_2)$;
- ▷ $\mathscr{V}\ell(\texttt{fun}\, x \to e) = \mathscr{V}\ell(e) \setminus \{x\}$.[1]

On dit que e est *close* si $\mathscr{V}\ell(e) = \emptyset$.

Définition 2. Pour $e \in$ FUN, $x \in \mathscr{V}$ et v une valeur **close**, on définit la *substitution* $e[{}^v/_x]$ de x par v dans e par :

- ▷ $k[{}^v/_x] = k$;
- ▷ $y[{}^v/_x] = \begin{cases} v & \text{si } x = y \\ y & \text{si } x \neq y \text{ ;} \end{cases}$
- ▷ $(\texttt{fun}\, y \to e)[{}^v/_x] = \begin{cases} \texttt{fun}\, y \to e & \text{si } x = y \\ \texttt{fun}\, y \to e[{}^e/_x] & \text{si } x \neq y; \end{cases}$
- ▷ $(e_1 + e_2)[{}^v/_x] = (e_1[{}^v/_x]) + (e_2[{}^v/_x])$;
- ▷ $(e_1\ e_2)[{}^v/_x] = (e_1[{}^v/_x])\ (e_2[{}^v/_x])$.

1. L'expression $\texttt{fun}\, x \to e$ est un *lieur* : x est liée dans e.

2.1 Grands pas pour FUN.

On définit la relation $\Downarrow$ sur couples (expression, valeur) par :

$$\frac{e_1 \Downarrow k_1 \qquad e_2 \Downarrow k_2}{e_1 + e_2 \Downarrow k}\;{\scriptstyle k = k_1 + k_2} \qquad \frac{}{v \Downarrow v}$$

$$\frac{e_1 \Downarrow \mathtt{fun}\, x \to e \qquad e_2 \Downarrow v_2 \qquad e[{}^{v_2}/_x] \Downarrow v}{e_1\ e_2 \Downarrow v.}$$

> **Remarque 2.** Certaines expressions ne s'évaluent pas :
>
> $$x \not\Downarrow \qquad \text{et} \qquad z + (\mathtt{fun}\, x \to x) \not\Downarrow$$
>
> par exemple.

2.2 Petits pas pour FUN.

On définit la relation $\to \subseteq \mathsf{FUN} * \mathsf{FUN}$ par :

$$\frac{}{k_1 + k_2 \to k}\;\mathscr{R}_{\mathrm{pk}}\ {\scriptstyle k = k_1 + k_2} \qquad \frac{}{(\mathtt{fun}\, x \to e)\ v \to e[{}^{v}/_x]}\;\mathscr{R}_\beta$$

$$\frac{e_2 \to e_2'}{e_1 + e_2 \to e_1 + e_2'}\;\mathscr{R}_{\mathrm{pd}} \qquad \frac{e_1 \to e_1'}{e_1 + k \to e_1' + k}\;\mathscr{R}_{\mathrm{pg}}$$

$$\frac{e_2 \to e_2'}{e_1\ e_2 \to e_1\ e_2'}\;\mathscr{R}_{\mathrm{ad}} \qquad \frac{e_1 \to e_1'}{e_1\ v \to e_1'\ v}\;\mathscr{R}_{\mathrm{ag}}.$$

> **Remarque 3.** Il existe des expressions que l'on ne peut pas réduire :
>
> 1. $k \not\to$;
> 2. $(\mathtt{fun}\, x \to x) \not\to$;
> 3. $e_1 + (\mathtt{fun}\, x \to x) \not\to$;
> 4. $3\ (5 + 7) \to 3\ 12 \not\to$.

Dans les cas 1. et 2., c'est cohérent : on ne peut pas réduire des valeurs.

Lemme 1. On a

$$e \Downarrow v \qquad \text{si, et seulement si,} \qquad e \to^\star v.$$

Remarque 4. Soit $e_0 = (\texttt{fun}\, x \to x\ \ x)\ (\texttt{fun}\, x \to x\ \ x)$. On remarque que $e_0 \to e_0$.

En FUN, il y a des divergences : il existe $(e_n)_{n\in\mathbb{N}}$ telle que l'on ait $e_n \to e_{n+1}$.

La fonction [2] définie par $\Downarrow$ est donc partielle.

Remarque 5 (Problème avec la substitution). On a la chaîne de réductions :

$$\begin{aligned} & ((\texttt{fun}\, y \to (\texttt{fun}\, x \to x + y))\ (x+7))\ 5 \\ (\star) \qquad \to\ & (\texttt{fun}\, x \to x + (x+7))\ 5 \\ \to\ & 5 + (5+7) \\ \to^\star\ & 17. \end{aligned}$$

Attention ! Ici, on a triché : on a substitué avec l'expression $x + 7$ mais ce n'est pas une valeur (dans la réduction $(\star)$) !

Mais, on a la chaîne de réductions

$$\begin{aligned} & (\texttt{fun}\, f \to (\texttt{fun}\, x \to (f\ \ 3) + x))\ (\texttt{fun}\, t \to x + 7)\ 5 \\ \to\ & (\texttt{fun}\, x \to ((\texttt{fun}\, t \to x + 7)\ 3) + x)\ 5 \\ \to\ & (\texttt{fun}\, x \to ((\texttt{fun}\, t \to x + 7)\ 3) + x)\ 5. \end{aligned}$$

Et là, c'est le drame, on a **capturé la variable libre**. D'où l'hypothèse de v close dans la substitution.

2. Pour indiquer cela, il faudrait démontrer que la relation $\Downarrow$ est déterministe.

Remarque 6. Les relations $\Downarrow$ et $\to$ sont définies sur des expressions **closes**. Et on a même $\to \subseteq \mathsf{FUN}_0 * \mathsf{FUN}_0$.[3]

Lemme 2. ▷ Si v est close et si $x \notin \mathscr{V}\ell(e)$ alors $e[{}^v/_x] = e$.
▷ Si v est close, $\mathscr{V}\ell(e[{}^v/_x]) = \mathscr{V}\ell(e) \setminus \{x\}$. □

Lemme 3. Si $e \in \mathsf{FUN}_0$ et $e \to e'$ alors $e' \in \mathsf{FUN}_0$.

Preuve. Montrons que, quelles que soient e et e', on a : si $e \to e'$ alors $(e \in \mathsf{FUN}_0) \implies (e' \in \mathsf{FUN}_0)$ On procède par induction sur la relation $e \to e'$. Il y a 6 cas :

1. Pour $\mathscr{R}_\beta$, on suppose $(\mathtt{fun}\, x \to e)\ v$ est close, alors
   - ▷ $(\mathtt{fun}\, x \to e)$ est close ;
   - ▷ v est close.

   On sait donc que $\mathscr{V}\ell(e) \subseteq \{x\}$, d'où par le lemme précédent, $\mathscr{V}\ell(e[{}^v/_x]) = \emptyset$ et donc $e[{}^v/_x]$ est close.

2–6. Pour les autres cas, on procède de la même manière.

□

Remarque 7. De même, si $e \Downarrow v$ où e est close, alors v est close.

Les relations $\Downarrow$ et $\to$ sont définies sur les expressions et les valeurs closes.

Définition 3 (Définition informelle de l'α-conversion)**.** On définit l'α-conversion, notée $e =_\alpha e'$: on a $\mathtt{fun}\, x \to e =_\alpha \mathtt{fun}\, y \to e'$ si, et seulement si, e' s'obtient en replaçant x par y dans e à condition que $y \notin \mathscr{V}\ell(e)$.[4]

On étend $e =_\alpha e'$ à toutes les expressions : « on peut faire ça

3. Il faudrait ici justifier que la réduction d'une formule close est close. C'est ce que nous allons justifier.

partout ».

Exemple 3 (*Les variables liées sont muettes.*). On a :

$$\begin{aligned}\texttt{fun}\, x \to x + z &=_\alpha \texttt{fun}\, y \to y + z\\ &=_\alpha \texttt{fun}\, t \to t + z\\ &\neq_\alpha \texttt{fun}\, z \to z + z.\end{aligned}$$

L'intuition est, quand on a fun $x \to e$ et qu'on a besoin de renommer la variable x, pour cela on prend $x' \notin \mathscr{V\ell}(e)$.

"Lemme" 1. Si $E_0 \subseteq \mathscr{V}$ est un ensemble fini de variables, alors il existe $z \notin E_0$ et $e' \in \mathsf{FUN}$ tel que $\texttt{fun}\, x \to e =_\alpha \texttt{fun}\, z \to e'$. □

Remarque 8 (Fondamental). En fait FUN désigne l'ensemble des expressions décrites par la grammaire initiale *quotientée* par α-conversion.

Remarque 9. On remarque que

$$(e =_\alpha e') \implies \mathscr{V\ell}(e) = \mathscr{V\ell}(e').$$

D'après le "lemme", on peut améliorer notre définition de la substitution.

Définition 4. Pour $e \in \mathsf{FUN}$, $x \in \mathscr{V}$ et v une valeur **close**, on définit la *substitution* $e[{}^v/_x]$ de x par v dans e par :

- ▷ $k[{}^v/_x] = k$;
- ▷ $y[{}^v/_x] = \begin{cases} v & \text{si } x = y\\ y & \text{si } x \neq y\end{cases}$;
- ▷ $(\texttt{fun}\, x \to e)[{}^v/_x] = (\texttt{fun}\, y \to e)[{}^v/_x]$ lorsque $x \neq y$;

4. C'est une « variable fraîche ».

- $(e_1 + e_2)[{}^v\!/\!_x] = (e_1[{}^v\!/\!_x]) + (e_2[{}^v\!/\!_x])$;
- $(e_1\ e_2)[{}^v\!/\!_x] = (e_1[{}^v\!/\!_x])\ (e_2[{}^v\!/\!_x])$.

3 Ajout des déclarations locales ($\mathsf{FUN} + \mathtt{let}$).

On ajoute les déclarations locales (comme pour $\mathsf{EA} \to \mathsf{LEA}$) à notre petit langage fonctionnel. Dans la grammaire des expressions de FUN, on ajoute :

$$e ::= \cdots \mid \mathtt{let}\, x \mathtt{=} e_1 \ \mathtt{in}\ e_2.$$

Ceci implique d'ajouter quelques éléments aux différentes opérations sur les expressions définies ci-avant :

- on définit $\mathscr{V\ell}(\mathtt{let}\, x \mathtt{=} e_1 \ \mathtt{in}\ e_2) = \mathscr{V\ell}(e_1) \cup (\mathscr{V\ell}(e_2) \setminus \{x\})$;
- on ne change pas les valeurs : une déclaration locale n'est pas une valeur ;
- on ajoute $\mathtt{let}\, x \mathtt{=} e_1 \ \mathtt{in}\ e_2 =_\alpha \mathtt{let}\, y \mathtt{=} e_1 \ \mathtt{in}\ e_2'$, où l'on remplace x par y dans e_2 pour obtenir e_2' ;
- pour la substitution, on pose lorsque $x \neq y$ (que l'on peut toujours supposer modulo α-conversion)
$$(\mathtt{let}\, y \mathtt{=} e_1 \ \mathtt{in}\ e_2)[{}^v\!/\!_x] = (\mathtt{let}\, y \mathtt{=} e_1[{}^v\!/\!_x] \ \mathtt{in}\ e_2[{}^v\!/\!_x]).$$
- pour la sémantique à grands pas, c'est comme pour LEA ;
- pour la sémantique à petits pas, on ajoute les deux règles :
$$\frac{}{\mathtt{let}\, x \mathtt{=} v \ \mathtt{in}\ e_2 \to e_2[{}^v\!/\!_x]}\ \mathscr{R}_{\mathrm{lv}}$$
et
$$\frac{e_1 \to e_1'}{\mathtt{let}\ x = e_1\ \mathtt{in}\ e_2 \to \mathtt{let}\ x = e_1'\ \mathtt{in}\ e_2}\ \mathscr{R}_{\mathrm{lg}}.$$
Attention ! On n'a pas de règle
$$\cancel{\frac{e_2 \to e_2'}{\mathtt{let}\ x = e_1\ \mathtt{in}\ e_2 \to \mathtt{let}\ x = e_1\ \mathtt{in}\ e_2'}\ \mathscr{R}_{\mathrm{ld}}},$$
on réduit d'abord l'expression e_1 jusqu'à une valeur, avant de passer à e_2.

Le langage que l'on construit s'appelle $\mathsf{FUN} + \mathtt{let}$.

3.1 Traduction de FUN + `let` vers FUN.

On définit une fonction qui, à toute expression de e dans FUN + `let` associe une expression notée $[\![e]\!]$ dans FUN (on supprime les expressions locales). L'expression $[\![e]\!]$ est définie par induction sur e. Il y a 6 cas :

- ▷ $[\![k]\!] = k$;
- ▷ $[\![x]\!] = x$;
- ▷ $[\![e_1 + e_2]\!] = [\![e]\!]_1 + [\![e]\!]_2$;
- ▷ $[\![e_1\ e_2]\!] = [\![e]\!]_1\ \ [\![e]\!]_2$;
- ▷ $[\![\texttt{fun}\, x \to e]\!] = \texttt{fun}\, x \to [\![e]\!]$;
- ▷ $[\![\texttt{let}\, x = e_1\ \texttt{in}\ e_2]\!] = (\texttt{fun}\, x \to [\![e_2]\!])\ \ [\![e_1]\!]$.

Lemme 4. Pour tout $e \in (\mathsf{FUN} + \texttt{let})$,

- ▷ $[\![e]\!]$ est une expression de FUN [5] ;
- ▷ on a $\mathscr{V\ell}([\![e]\!]) = \mathscr{V\ell}(e)$;
- ▷ $[\![e]\!]$ est une valeur *ssi* e est une valeur ;
- ▷ $[\![e[{}^{v}/_{x}]]\!] = [\![e]\!]\,[{}^{[\![v]\!]}/_{x}]$ [6]. □

Pour démontrer le lemme 4, on procède par induction sur e. C'est long et rébarbatif, mais la proposition ci-dessous est bien plus intéressante.

Proposition 1. Pour toutes expressions e, e' de FUN + `let`, si on a la réduction $e \to_{\mathsf{FUN+let}} e'$ alors $[\![e]\!] \to_{\mathsf{FUN}} [\![e']\!]$.

Preuve. On procède par induction sur $e \to e'$ dans FUN + `let`. Il y a 8 cas car il y a 8 règles d'inférences pour $\to$ dans FUN + `let`.

- ▷ *Cas* $\mathscr{R}_{\mathrm{lv}}$. Il faut montrer que $[\![\texttt{let}\, x = v\ \texttt{in}\ e_2]\!] \to_{\mathsf{FUN}} [\![e[{}^{v}/_{x}]]\!]$. Par définition, l'expression de droite vaut

$$(\texttt{fun}\, x \to [\![e]\!]_2)\ \ [\![v]\!] \xrightarrow{\mathscr{R}_\beta}_{\mathsf{FUN}} [\![e]\!]_2\,[{}^{[\![v]\!]}/_{x}],$$

5. *i.e.* $[\![e]\!]$ n'a pas de déclarations locales
6. On le prouve par induction sur e, c'est une induction à 6 cas

car $[\![v]\!]$ est une valeur par le lemme 4, ce qui justifie $\mathcal{R}_\beta$. De plus, encore par le lemme 4, on a l'égalité entre $[\![e]\!]_2\,[{}^{[\![v]\!]}/x] = [\![e[{}^v/x]]\!]$.

- ▷ *Cas* $\mathcal{R}_{\text{lg}}$. On sait que $e_1 \to e_1'$ et, par hypothèse d'induction, on a $[\![e_1]\!] \to [\![e_1']\!]$. Il faut montrer que

$$[\![\texttt{let}\, x \texttt{ = } e_1 \texttt{ in } e_2]\!] \to [\![\texttt{let}\, x \texttt{ = } e_1' \texttt{ in } e_2]\!]\,.$$

L'expression de droite vaut

$$(\texttt{fun}\, x \texttt{ → } [\![e_2]\!])\ \ [\![e_1]\!] \xrightarrow{\mathcal{R}_{\text{ad}}\ \&\ \text{hyp. ind.}} (\texttt{fun}\, x \texttt{ → } [\![e_2]\!])\ \ [\![e_1']\!]\,.$$

Et, par définition de $[\![\cdot]\!]$, on a l'égalité :

$$[\![\texttt{let}\, x \texttt{ = } e_1' \texttt{ in } e_2]\!] = (\texttt{fun}\, x \texttt{ → } [\![e_2]\!])\ \ [\![e_1']\!]\,.$$

- ▷ Les autres cas sont laissées en exercice.

□

Proposition 2. Si $[\![e]\!] \to [\![e']\!]$ alors $e \to e'$.

Preuve. La proposition ci-dessus est mal formulée pour être prouvée par induction, on la ré-écrit. On démontre, par induction sur la relation $f \to f'$ la propriété suivante :

> « quel que soit e, si $f = [\![e]\!]$ alors il existe e' une expression telle que $f' = [\![e']\!]$ et $e \to e'$ (dans FUN + `let`) »,

qu'on notera $\mathcal{P}(f, f')$.

Pour l'induction sur $f \to f'$, il y a 6 cas.

- ▷ *Cas de la règle* $\mathcal{R}_{\text{ad}}$. On suppose $f_2 \to f_2'$ et par hypothèse d'induction $\mathcal{P}(f_2, f_2')$. On doit montrer $\mathcal{P}(f_1\ f_2, f_1\ f_2')$. On suppose donc $[\![e]\!] = f_1\ f_2$. On a deux sous-cas.
  - – *1[er] sous-cas.* On suppose $e = e_1\ e_2$ et $[\![e_1]\!] = f_1 = f_2$. Par hypothèse d'induction et puisque $[\![e_2]\!] = f_2$, il

existe e_2' tel que $e_2 \to e_2'$ et $[\![e_2']\!] = f_2'$. De $e_2 \to e_2'$, on en déduit par $\mathscr{R}_{\mathrm{ad}}$ que $e_1\ e_2 \to e_1\ e_2'$. On pose $e' = e_1\ e_2'$ et on a bien $[\![e']\!] = [\![e_1]\!]\ [\![e_2']\!]$.

– *2ème sous-cas.* On suppose $e = \texttt{let}\,x\,\texttt{=}\,e_1\ \texttt{in}\ e_2$. Alors,

$$[\![e]\!] = \underbrace{(\texttt{fun}\,x \texttt{ -> } [\![e_2]\!])}_{f_1}\ \underbrace{[\![e_1]\!]}_{f_2}.$$

Par hypothèse d'induction, il existe e_1' tel que $e_1 \to e_1'$ et $[\![e_1']\!] = f_2'$. Posons $e' = (\texttt{let}\,x\,\texttt{=}\,e_1'\ \texttt{in}\ e_2)$. On doit vérifier $[\![e]\!] \to [\![e']\!]$ ce qui est vrai par $\mathscr{R}_{\mathrm{ad}}$ et que $[\![e']\!] = f_1\ f_2'$, ce qui est vrai par définition.

▷ *Cas de la règle $\mathscr{R}_{\mathrm{ag}}$.* On suppose $f_1 \to f_1'$ et l'hypothèse d'induction $\mathscr{P}(f_1, f_1')$. On doit vérifier que $\mathscr{P}(f_1\ v, f_1'\ v)$. On suppose $[\![e]\!] = f_1\ v$ et on a deux sous-cas.

– *1er sous-cas.* On suppose $e = e_1\ e_2$ et alors $[\![e]\!] = [\![e_1]\!]\ [\![e_2]\!]$ par le lemme 4 et parce que e_2 est une valeur (car $[\![e_2]\!] = v$). On raisonne comme pour la règle $\mathscr{R}_{\mathrm{ad}}$ dans le premier sous-cas, en appliquant $\mathscr{R}_{\mathrm{ag}}$.

– *2nd sous-cas.* On suppose $e = (\texttt{let}\,x\,\texttt{=}\,e_1\ \texttt{in}\ e_2)$ alors

$$[\![e]\!] = \underbrace{\texttt{fun}\,x \texttt{ -> } [\![e_2]\!]}_{f_1}\ \underbrace{[\![e_1]\!]}_{f_2}.$$

On vérifie aisément ce que l'on doit montrer.

▷ Les autres cas se font de la même manière (attention à $\mathscr{R}_\beta$).

□