2.2 Equivalence Observationnelle

Comme la sémantique opérationnelle ne donne pas vraiement de sémantique aux programmes sans entrées, il arrive souvent que elle ne soit pas adequate pour identifier programmes que calculent la même fonction. Par exemple, dans n'importe laquelle des sémantiques operationnelles déjà vues, on a que:

$$\texttt{fun x: A->(fun y: A->y) x} \Downarrow \texttt{fun x: A->(fun y: A->y) x}$$

$$\texttt{fun x: A->x} \Downarrow \texttt{fun x: A->x}$$

On a très envie de identifier `fun x: A->x` et `fun x: A->(fun y: A->y) x`, vu que les deux calculent bien la fonction identité, mais le problème est que, pour s'en apercevoir, il faut appliquer les deux programmes à un argument.

C'est bien cela l'idée qui est à la base de la notion d'equivalence observationnelle: on cherche à définir une notion d'équivalence de programmes qui identifie exactement ces (fragments de) programmes "qui font la même chose" du point de vue d'un observateur choisi.

Définition 2.2.1 (Contexte) *Un contexte avec un trou, noté $C[\,]$, est un (fragment de) programme où une variable spéciale distinguée $[\,]$ apparait au plus une fois. On note $C[M]$ le (fragment de) programme obtenu en remplaçant $[\,]$ par M textuellement (i.e. , sans faire les α-conversions comme dans la substitution du λ-calcul).*

Définition 2.2.2 (Equivalence observationnelle) *Fixée une sémantique opérationnelle S, une notion de observation O_S, et une notion de programme, deux fragments de programmes M et N sont dit equivalent observationnellement, par rapport à O_S et S, ssi:*

$$\forall C[]\ \textit{t.q. } C[M] \textit{ et } C[N] \textit{ sont programmes observables par } O_S$$

$$O_S(C[M]) \Longleftrightarrow O_S(C[N])$$

Dans ce cas, on écrit

$$M \approx_{O_S,S} N$$

Remarque 2.2.3 (Notions d'observation) *Il y a plusieurs notions d'observation que l'on trouve utilisées dans la literature. En voici qques exemples:*

- *$O_S(P)$ = le programme P termine avec la sémantique S*
- *$O_S(P)$ = le programme P de type de base donne comme résultat, avec la sémantique S, une constante donnée*

Remarque 2.2.4 *On va voir plus avant que, par rapport aux notions d'observation O precédentes, et toutes les sémantiques opérationnelles S déjà vues,*

$$\texttt{fun x: A->x} \quad \approx_{O,S} \quad \texttt{fun x: A->(fun y: A->y) x}$$

2.2.1 Rélations entre sémantiques opérationnelles

Si on a plusieurs sémantiques opérationnelles d'un langage (ou si on a deux langages dont un est inclus dans l'autre) on peut se poser plusieurs questions sur la relation entre les sémantiques. C'est pour cela que l'on introduit les notions suivantes:

Définition 2.2.5 *Soit L_1 un langage equipé d'une sémantique opérationnelle $\Downarrow_1$ et inclus dans un langage L_2 equipé d'une sémantique opérationnelle $\Downarrow_2$. On dit alors que:*

$\Downarrow_2$ **est adequate pour le calcul vis à vis de** $\Downarrow_1$ *si pour tout programme P (de L_1),*

$$O_{\Downarrow_1}(P) \Longleftrightarrow O_{\Downarrow_2}(P)$$

(On peut travailler aussi bien dans L_2 si on s'intéresse à observer un programme).

$\Downarrow_2$ **est equationnellement correcte vis à vis de** $\Downarrow_1$ *si*

$$M \approx_{O,\Downarrow_2} N \text{ implique } M \approx_{O,\Downarrow_1} N$$

(On peut travailler aussi bien dans L_2 si on s'intéresse à prouver l'equivalence observationnelle).

$\Downarrow_2$ **preserve l'abstraction de** $\Downarrow_1$ *si ell'est equationnellement correcte et en plus*

$$M \approx_{O,\Downarrow_1} N \text{ implique } M \approx_{O,\Downarrow_2} N$$

(Les deux sémantiques génèrent la même equivalence observationnelle sur L_1).

Si les deux langages coincident, les deux dernières proprietés decoulent de la première.

Remarque 2.2.6 *Dans le cas de TFun, toutes les sémantiques que l'on a vu sont bien equivalentes par rapport a tous ces critéres.*

- *par rapport à la terminaison: pas de difference parce-que tout programme termine*

- *si on observe des constantes sur les types de base: là, les valeurs coincident avec les formes normales dans le lambda calcul typé simple, et donc le resultats par nom ou valeur sont les mêmes, d'où le fait que $\Downarrow_v$ et $\Downarrow_n$ genèrent la même equivalence observationnelle si on observe les types de base (et de là on obtient les trois proprietés de plus en haut).*

2.3 Sémantique dénotationnelle de TFun dans Set

On rappelle que la sémantique dénotationnelle est la donnée de la syntaxe abstraite du langage (ce qu'on a déjà fait plus haut pour TFun), des domaines d'interprétation sémantique et des morphismes *compositionnelles* d'interprétation qui associent à chaque élément des différentes catégories syntaxiques un objet dans les domaines.

2.3.1 Domaines sémantiques

Nous allons interpréter TFun dans les ensembles, et plus spécifiquement dans une structure connue comme la "Full type hierarchy" sur certains ensembles de base.

Définition 2.3.1 (Fonctionnels de type fini) *La full type hierachy basée sur des ensembles non vides $B_1, \ldots, B_n$ pour les types de base est une famille*

$$\mathcal{F}_{B_1,\ldots,B_n} = (\mathcal{F}_t)_{t \in Types}$$

définie comme suit par induction sur les types:

- $\mathcal{F}_{\mathtt{bi}} = B_i$ *si* `bi` *est un type de base*
- $\mathcal{F}_{\mathtt{t1\text{->}t2}}$ = *toute fonction totale entre* $\mathcal{F}_{\mathtt{t1}}$ *et* $\mathcal{F}_{\mathtt{t2}}$
- $\mathcal{F}_{\mathtt{t1*t2}}$ = *le produit cartésien de* $\mathcal{F}_{\mathtt{t1}}$ *et* $\mathcal{F}_{\mathtt{t2}}$

2.3.2 Morphismes d'interprétation

On va maintenant donner notre sémantique sur $\mathcal{F}_{I\!N,\{true,false\}}$, que l'on va noter FTH.

T[[]]: Types $\rightarrow$ FTH

- $T[\![\mathtt{int}]\!] = I\!N$
- $T[\![\mathtt{bool}]\!] = \{true, false\}$
- $T[\![\mathtt{t1\ \text{->}\ t2}]\!]$ = l'ensemble des fonctions totales entre $T[\![\mathtt{t1}]\!]$ et $T[\![\mathtt{t2}]\!]$
- $T[\![Exp[\mathtt{t}]]\!] = \mathrm{T}[\![\mathtt{t}]\!]$
- $T[\![Id[\mathtt{t}]]\!] = \mathrm{T}[\![\mathtt{t}]\!]$

$G[\![]\!]$: Environnements → FTH L'interprétation d'un environnement de typage est les produit de l'interprétations de types des identificateurs qu'il déclare:

- $G[\![\Gamma]\!] = T[\![\texttt{t1}]\!] \times \cdots \times T[\![\texttt{tn}]\!]$ if $\Gamma = \texttt{x1:}\ Id[\texttt{t1}], \cdots, \texttt{xn:}\ Id[\texttt{tn}]$

$E[\![]\!]$: Expressions → FTH La donnée des morphismes d'interprétation sur les expressions est faite par induction sur la structure de la dérivation de typage[2]: à une assertion de la forme

$$\Gamma \vdash \texttt{e:}\ Exp[\texttt{t}]$$

on associe une fonction $E[\![\Gamma \vdash \texttt{e:}\ Exp[\texttt{t}]]\!]$ entre l'interprétation de Γ et l'interprétation du type $Exp[t]$. On notera $E[\![\Gamma \vdash \texttt{e:}\ Exp[\texttt{t}]]\!]_a$ pour la valeur de la fonction $E[\![\Gamma \vdash \texttt{e:}\ Exp[\texttt{t}]]\!]$ sur a.

- $E[\![\Gamma|-\texttt{i:}\ Exp[\texttt{int}]]\!]_e = i$ (pour i constante entière)
- $E[\![\Gamma|-\texttt{true:}\ Exp[\texttt{bool}]]\!]_e = true$,
- $E[\![\Gamma|-\texttt{false:}\ Exp[\texttt{bool}]]\!]_e = false$,
- $E[\![\Gamma|-\texttt{id:}\ Exp[\texttt{t}]]\!]_e = \pi_i^n(e)$ si id est le i-ème identificateur dans Γ (de longueur n)
- $E[\![\Gamma|-\texttt{fun x: t1->M:}\ Exp[\texttt{t1->t2}]]\!]_e = [a \mapsto f(e,a)]$
  où $f \in (G[\![\Gamma]\!] \times T[\![\texttt{t1}]\!] \to T[\![\texttt{t2}]\!])$ est $E[\![\Gamma, \texttt{x:}\ Id[\texttt{t1}]|-\texttt{M:}\ Exp[\texttt{t2}]]\!]$
- $E[\![\Gamma|-\texttt{M N:}\ Exp[\texttt{t}]]\!]_e = (E[\![\Gamma|-\texttt{M:}\ Exp[\texttt{t1->t}]]\!]_e)\ E[\![\Gamma|-\texttt{N:}\ Exp[\texttt{t1}]]\!]_e$

Remarque 2.3.2 (Projections) *Ici, on assume que π_i^n soit défini comme:*

$$\pi_i^n = \begin{cases} \pi_1^1 = id & \\ \pi_n^n = snd & n > 1 \\ \pi_i^n = \pi_{i-1}^{n-1} \circ fst & sinon \end{cases}$$

On peut aussi le définir comme on fait souvent en thèorie des catégories:

$$\pi_i^n = \begin{cases} \pi_n^n = snd & \\ \pi_i^n = \pi_{i-1}^{n-1} \circ fst & sinon \end{cases}$$

Mais dans ce cas, un environnement ne peut jamais être de taille plus petite que 1, *et on est forcé de poser:*

$$G[\![\Gamma]\!] = 1 \times T[\![\texttt{t1}]\!] \times \cdots \times T[\![\texttt{tn}]\!]$$

où 1 *est l'ensemble à un seul élément.*

[2] Ce qui est cohérent en raison de l'unicité de la dérivation.

Voici quelques relations typiques entre syntaxe et sémantique:

Lemme 2.3.3 (Substitution sémantique)

$E[\![\Gamma, \mathtt{x:\ t1}|-\mathtt{M:}\ \mathit{Exp}[\mathtt{t2}]]\!]_{(e,E[\![\Gamma|-\mathtt{N:}\ \mathit{Exp}[\mathtt{t1}]]\!]_e)} = E[\![\Gamma|-\mathtt{M[N/x]:}\ \mathit{Exp}[\mathtt{t2}]]\!]_e$

Preuve. Longue induction sur la structure de M. □

Lemme 2.3.4 (Indépendance sémantique) *Si e et e' coïncident sur les variables de Γ, alors*

$$E[\![\Gamma|-\mathtt{M:}\ \mathit{Exp}[\mathtt{t}]]\!]_e = E[\![\Gamma|-\mathtt{M:}\ \mathit{Exp}[\mathtt{t}]]\!]_{e'}$$

Notation 2.3.5 *Pour breveté, dans ce qui suit, on va souvent oublier le contexte de typage Γ, s'il n'est pas relevant dans le contexte. Aussi, on ne va pas toujours distinguer les morphismes d'interprétation: on écrira juste $[\![]\!]$.*

2.4 Relations entre sémantique opérationnelle et dénotationnelle

Il y a plusieurs choses que l'on veut pouvoir dire sur la relation entre une sémantique opérationnelle et une sémantique dénotationnelle d'un langage donné. En voici quelques unes:

Définition 2.4.1 *Soit L un langage équipé d'une sémantique opérationnelle $\Downarrow$, avec une équivalence observationnelle $\approx$, et d'une sémantique dénotationnelle $[\![]\!]$. On dit alors que:*

$[\![]\!]$ **est adéquate pour le calcul vis à vis de $\Downarrow$** *si:*

$$[\![\mathtt{M}]\!] = [\![\mathtt{v}]\!] \Rightarrow M \Downarrow v \quad v \text{ un valeur}$$

(on peut regarder les dénotations si on cherche juste le résultat)

$[\![]\!]$ **est correcte pour $\Downarrow$ vis à vis de $\approx$** *si:*

$$[\![\mathtt{M}]\!] = [\![\mathtt{N}]\!] \Rightarrow M \approx N$$

(on peut utiliser les dénotations pour prouver l'équivalence observationnelle)

$[\![]\!]$ **est complète pour $\Downarrow$ vis à vis de $\approx$** *si:*

$$M \approx N \Rightarrow [\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$$

(on peut utiliser les dénotations pour prouver la non-équivalence observationnelle)

$[\![ ]\!]$ **est complètement abstraite pour $\Downarrow$ vis à vis de $\approx$** *si ell'est correcte et complète.*
(on peut utiliser les dénotations à la place de l'équivalence observationnelle)

Remarque 2.4.2 *L'équivalence observationnelle est très difficile à établir à partir de la définition: il y a une quantification sur tous les contextes! C'est pour cela que une sémantique dénotationnelle complètement abstraite (ou même juste correcte), est très utile et recherchée. Par contre, la définition est utile pour prouver la non équivalence: il suffit de trouver un contexte départageant les deux (fragments de) programmes.*

Remarque 2.4.3 *L'abstraction complète n'est pas une propriété courante, mais par contre, si la définition de la sémantique dénotationnelle est donnée de façon compositionnelle, on a facilement la correction.*

Notation 2.4.4 *On écrit souvent*

$$\mathcal{M} \models M = N$$

à la place de $[\![\mathtt{M}]\!]=[\![\mathtt{N}]\!]$, *où* $\mathcal{M}$ *est le modèle utilisé pour la sémantique dénotationnelle.*

Proposition 2.4.5 (Consistance) *Dans TFun, si* $\Gamma|-\mathtt{M}\colon$ *Exp*`[t]` *et* $M \Downarrow_v v$, *alors* $[\![\mathtt{M}]\!] = [\![\mathtt{v}]\!]$.

Preuve. Induction sur la taille de $M \Downarrow_v v$. Voyons juste le cas de l'application:

$$\frac{e_1 \Downarrow_v \mathtt{fun}\ x : A\texttt{->}e_0 \qquad e_2 \Downarrow_v v_2 \qquad e_0[v_2/x] \Downarrow_v v_0}{e_1(e_2) \Downarrow_v v_0}$$

On sait par hypothèse d'induction que:

- $[\![e_1]\!] = [\![\mathtt{fun}\ x : A\texttt{->}e_0]\!]$
- $[\![e_2]\!] = [\![v_2]\!]$
- $[\![e_0[v_2/x]]\!] = [\![v_0]\!]$

Donc on peut dériver que:

$$\begin{array}{rcl}
[\![e_1(e_2)]\!](e) & = & ([\![e_1]\!](e))([\![e_2]\!](e)) \\
 & = & ([\![\mathtt{fun}\ x : A\texttt{->}e_0]\!](e))([\![v_2]\!](e)) \\
 & = & ([a \mapsto [\![e_0]\!](e,a)])([\![v_2]\!](e)) \\
 & = & [\![e_0]\!](e, ([\![v_2]\!](e))) \\
 & = & [\![e_0[v_2/x]]\!](e) \\
 & = & [\![v_0]\!](e)
\end{array}$$

et on a fini. $\square$

Exercice 2.4.6 *Compléter la preuve, et montrer le même résultat pour* $\Downarrow_n$ *.*

Lemme 2.4.7 (L'égalité dans $[\![]\!]$ est une *congruence*)
Si $[\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$*, alors* $[\![\mathtt{C[M]}]\!] = [\![\mathtt{C[N]}]\!]$ *pour tout contexte* $C[\,]$*.*

Preuve. Par induction sur la structure de $C[\,]$. Cela se passe bien en raison de la compositionalité de la définition de $[\![]\!]$. □

Corollaire 2.4.8 $[\![]\!]$ *est correcte vis à vis de* $\Downarrow$ *, soit en appel par nom que par valeur, si on observe les types de base.*

Preuve. Prouvons la contrapositive: soit un contexte qui départage M et N, donc tel que $C[M] \Downarrow v$, $C[N] \Downarrow v'$, et supposons $[\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$. Alors, pour tout contexte $C[\,]$, $[\![\mathtt{C[M]}]\!] = [\![\mathtt{C[N]}]\!]$.

Mais par la consistance, $[\![\mathtt{v}]\!] = [\![\mathtt{C[M]}]\!] = [\![\mathtt{C[N]}]\!] = [\![\mathtt{v'}]\!]$.

Or, sur les types de base, $v \equiv v' \Longleftrightarrow [\![\mathtt{v}]\!] = [\![\mathtt{v'}]\!]$, et on obtient une contradiction. □

Corollaire 2.4.9 *Si* $[\![\mathtt{M}]\!] = n$ *(n un booléen ou un entier), alors* $M \Downarrow n$*.*

Preuve. Parce-que alors M et n sont équivalents observationnellement, et étant de type de base, $M \Downarrow n$ est conséquence de la définition en prenant $C[\,]$ comme le contexte vide. □

Exercice 2.4.10 *Vérifier que l'équivalence suivante (règle η)*

```
fun x: A->x   ≈O,S   fun x: A->(fun y: A-> x y)
```

est valide si l'on observe les types de base.

Exercice 2.4.11 *Montrer que, si l'on observe les valeurs aux types supérieurs, η n'est pas valide.*

2.4.1 Conclusion

Dans le cas de TFun, on a donc prouvé que:

- $[\![]\!]$ est correcte vis à vis de l'observation des types de base
- $M \Downarrow_{v,n} n$ ssi $[\![\mathtt{M}]\!] = n$

($[\![]\!]$ est aussi correcte par rapport à la terminaison, mais pour des raison triviales pour TFun).

Exercice 2.4.12 *Montrez que* $[\![]\!]$ *n'est pas correcte si on observe des valeurs qui ne sont pas de base.*