se spécialise en

$$\frac{\texttt{t} \hookrightarrow \texttt{\{}l_1\texttt{ = fun }x_1\texttt{ -> }t_1\texttt{, ...\}}}{\begin{array}{c}\texttt{t(}l_i\texttt{ <- (fun y -> v))} \hookrightarrow \\ \texttt{\{}l_1\texttt{ = fun }x_1\texttt{ -> }t_1\texttt{, ..., }l_i\texttt{ = (fun y -> v), ...\}}\end{array}}$$

Pour forcer tous les champs d'un objet à être des fonctions, on peut modifier légèrement le langage des enregistrements, ce qui nous fait évoluer d'un langage d'enregistrements à un langage d'objets proprement dit. Le symbole {} lie désormais une variable dans chacun de ses arguments de rangs pairs — les termes —, le symbole . disparaît et est remplacé par le symbole #, le symbole <- lie désormais une variable dans son troisième argument.

Le terme {}(l_1, s_1 t_1, ..., l_n, s_n t_n) se note {l_1 = ςs_1 t_1, ..., l_n = ςs_n t_n}, le terme #(t,l) se note t#l et le terme <-(t,l,s u) se note t(l <- ς s u). Les règles de sémantique opérationnelle à grands pas se formulent alors ainsi

$$\frac{}{\texttt{\{}l_1\texttt{ = }\varsigma s_1\ t_1\texttt{, ...,}l_n\texttt{ = }\varsigma s_n\ t_n\texttt{\}} \hookrightarrow \texttt{\{}l_1\texttt{ = }\varsigma s_1\ t_1\texttt{, ...,}l_n\texttt{ = }\varsigma s_n\ t_n\texttt{\}}$$

$$\frac{\texttt{t} \hookrightarrow \texttt{\{}l_1\texttt{ = }\varsigma s_1 t_1\texttt{, ..., }l_n\texttt{ = }\varsigma s_n t_n\texttt{\}} \qquad (\texttt{t}/s_i)t_i \hookrightarrow \texttt{V}}{\texttt{t\#}l_i \hookrightarrow \texttt{V}}$$

$$\frac{\texttt{t} \hookrightarrow \texttt{\{}l_1\texttt{ = }\varsigma s_1 t_1\texttt{, ...\}}}{\begin{array}{c}\texttt{t(}l_i\texttt{ <- }\varsigma\texttt{s u)} \hookrightarrow \texttt{\{}l_1\texttt{ = }\varsigma s_1 t_1\texttt{, ..., }l_{i-1}\texttt{ = }\varsigma s_{i-1}\ t_{i-1}\texttt{,} \\ l_i\texttt{ = }\varsigma\texttt{ s u, }l_{i+1}\texttt{ = }\varsigma s_{i+1} t_{i+1}\texttt{, ...\}}\end{array}}$$

Exercice 8.7 *Écrire un interpréteur pour PCF avec des objets.*

Exercice 8.8 *(La liaison tardive) La valeur du terme*
`(({x = ς s 4, f = ς s fun y -> y + s#x} (x <- ς s 5))#f) 6`
est-elle `10` *ou bien* `11` *? Comparer ce résultat avec celui de l'exercice 2.8.*

8.2.3 Les objets et les références

La définition habituelle selon laquelle un objet a un état interne qui évolue au cours du temps, mêle les notions d'objets et de références, qui sont dissociées dans la notion d'objet fonctionnel présentée ci-dessus.

Dans un langage qui a des objets et des références, quand on transforme un champ non fonctionnel `a = u` en `a = fun x -> u`, l'interprétation de `fun x -> u`, ne produit plus les effets secondaires que produisait l'interprétation de `u`. C'est seulement quand on accédera à ce champ que se produiront ces effets secondaires. Ainsi, tout se passe comme pour la sémantique des enregistrements en appel par nom. Le terme

```
let x = {a = fun s -> ref 0}
in let inc = fun s -> (s#a := 1 + !(s#a))
in (inc x; !(x#a))
```