Une alternative est de retarder cette addition en considérant que le terme `{a = 3 + 4, b = 2}` est une valeur et qu'il s'interprète donc sur lui-même, mais qu'en revanche on doit interpréter le terme `3 + 4` à chaque fois que l'on accède au champ `a` de cette valeur. On peut appeler une telle sémantique *en appel par nom*, par opposition à la sémantique que nous avons donnée qui est *en appel par valeur*.

En appel par nom, les règles de sémantique opérationnelle deviennent

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

$$\frac{t \hookrightarrow \texttt{\{}l_1\texttt{ = }t_1\texttt{, ..., }l_n\texttt{ = }t_n\texttt{\}} \quad t_i \hookrightarrow V}{t.l_i \hookrightarrow V}$$

$$\frac{t \hookrightarrow \texttt{\{}l_1\texttt{ = }t_1\texttt{, ..., }l_n\texttt{ = }t_n\texttt{\}}}{t(l_i \texttt{ <- } u) \hookrightarrow \texttt{\{}l_1\texttt{ = }t_1\texttt{, ..., }l_{i-1}\texttt{ = }t_{i-1}\texttt{, }l_i\texttt{ = }u\texttt{, }l_{i+1}\texttt{ = }t_{i+1}\texttt{, ..., }l_n\texttt{ = }t_n\texttt{\}}}$$

Exercice 8.4 *Écrire un interpréteur pour PCF avec des enregistrements en appel par nom.*

En comparant ces deux sémantiques des enregistrements, on sera amené à faire à peu près les mêmes remarques que pour la sémantique des fonctions en appel par valeur et en appel par nom : l'interprétation de `let x = {a = fact 10, b = 4} in x.b` demande de calculer la factorielle de `10` en appel par valeur, mais pas en appel par nom. En revanche `let x = {a = fact 10, b = 4} in x.a + x.a` demande de calculer deux fois la factorielle de `10` en appel par nom, et une seule fois en appel par valeur. L'interprétation de `let x = {a = fix y y, b = 4} in x.b` boucle en appel par valeur, mais retourne `4` en appel par nom. Enfin, quand on a des références, les effets secondaires de l'interprétation d'un champ peuvent se produire un nombre arbitraire de fois selon le nombre de fois que l'on accède à ce champ — voir l'exercice 7.5.

Par exemple, si on construit un enregistrement `x`, qui contient un champ `a` qui est une référence sur un entier, dont la valeur initiale est `0`, et une fonction `inc` qui incrémente cet entier, puis un terme qui incrémente cet entier et retourne ensuite sa valeur, on aboutit au terme

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

Si on interprète ce terme en appel par valeur, on obtient `1`, ce qui est, sans doute, la sémantique espérée par les programmeurs. En appel par nom en revanche, comme on accède trois fois lors de l'interprétation de ce terme, au champ `a` de l'enregistrement `x` on interprète trois fois le terme `ref 0`, ce qui crée trois références qui partent toutes les trois de la valeur `0`. La troisième référence, créée par l'interprétation du terme `!(x.a)` n'est jamais affectée, et l'interprétation de ce terme en appel par nom produit donc la valeur `0`.