Dans le cas où l'expression `t` *est une variable* `x`*, on a* Σ`(x := u,e,m) = m + (`Θ`(x,e,m) = `Θ`(u,e,m)) = m + (e(x) = `Θ`(u,e,m))` *et on retrouve donc la définition de la fonction* Σ *pour l'affectation de Java.*

La définition de la fonction Σ *de C n'est pas très différente de celle de la fonction* Σ *de Java. La principale différence est dans le cas des déclarations de variables*

$$\Sigma(\texttt{\{T x = t ; q\},e,m}) = (\Sigma(\texttt{q,e+(x=r),m + (r = }\Theta\texttt{(t,e,m))}))_{|\texttt{Ref}-\{\texttt{r}\}}$$

où `r` *est une référence quelconque qui n'apparaît pas dans* `e` *et* `m` *et la notation* $\texttt{m}_{|\texttt{Ref}-\{\texttt{r}\}}$ *désigne la mémoire* `m` *dans laquelle on a supprimé le couple de la forme* `r = v` *s'il en existe un. Ainsi, si l'on exécute l'instruction* `{int x = 4 ; p} q` *dans un état* `e, m`*, on exécute l'instruction* `p` *dans l'état* `e + (x = r), m + (r = 4)` *en C comme en Java. En revanche, on exécute l'instruction* `q` *dans l'état* `e, m + (r = 4)` *en Java et dans l'état* `e, m` *en C.*

Comme, dans l'environnement `e`*, il n'y a pas de variable qui permette d'accéder à la référence* `r`*, le couple* `r = 4` *ne sert à rien et qu'il reste dans la mémoire, comme en Java ou en Caml, ou qu'il soit supprimé, comme en C, ne change rien. Cependant, nous verrons, à l'exercice 2.17, que ce choix de C est à l'origine de difficultés, quand le langage contient d'autres constructions.*

Exercice 1.6

Le *test incomplet* permet de former une instruction à partir d'une expression booléenne et d'une instruction. Cette instruction s'écrit `if (b) p`. La valeur de la fonction Σ pour une telle instruction se définit ainsi. Si Θ`(b,e,m) = true` alors

$$\Sigma\texttt{(if (b) p,e,m) = }\Sigma\texttt{(p,e,m)}$$

Si Θ`(b,e,m) = false` alors

$$\Sigma\texttt{(if (b) p,e,m) = m}$$

Montrer que l'on peut définir cette construction à partir du test complet et de l'instruction `skip ;`.

Exercice 1.7

La boucle `do` permet de former une instruction à partir d'une expression booléenne et d'une instruction. Cette instruction s'écrit `do p while (b)`. C'est une abréviation pour l'instruction `{p while (b) p}`. Donner une définition directe de la fonction Σ pour cette construction.