Exercice 5.4 *On étend PCF avec les constructions pour les listes d'entiers de l'exercice 3.14 et on ajoute un type* `natlist` *pour ces listes. Donner les règles de typage de cette extension de PCF. Écrire un vérificateur de types pour cette extension de PCF.*

Exercice 5.5 *On étend PCF avec les constructions pour les arbres de l'exercice 3.15 et un type* `nattree` *pour ces arbres. Donner les règles de typage de cette extension de PCF. Écrire un vérificateur de types pour cette extension de PCF.*

5.2 L'absence d'erreurs à l'exécution

On peut maintenant montrer que l'interprétation d'un terme bien typé ne produit pas d'erreurs de type à l'exécution. Selon que l'on décrit l'exécution par une sémantique opérationnelle à petits pas ou à grands pas, la démonstration est un peu différente.

5.2.1 En sémantique opérationnelle à petits pas

En sémantique opérationnelle à petits pas, cette propriété se formule comme le fait que le résultat du calcul d'un terme bien typé et clos, s'il existe, est toujours une valeur, c'est-à-dire une constante entière ou un terme réduit et clos de la forme `fun x -> t`, mais que ce ne peut pas être un terme bloqué : un terme de la forme $\mathtt{V_1}\ \mathtt{V_2}$, où $\mathtt{V_1}$ et $\mathtt{V_2}$ sont des termes irréductibles et clos et $\mathtt{V_1}$ n'est pas de la forme `fun x -> t`, un terme de la forme $\mathtt{V_1} \otimes \mathtt{V_2}$, où $\mathtt{V_1}$ et $\mathtt{V_2}$ sont des termes irréductibles et clos qui ne sont pas tous les deux des constantes entières, ou un terme de la forme $\mathtt{ifz}\ \mathtt{V_1}\ \mathtt{then}\ \mathtt{V_2}\ \mathtt{else}\ \mathtt{V_3}$ où $\mathtt{V_1}$, $\mathtt{V_2}$ et $\mathtt{V_3}$ sont des termes irréductibles et clos et $\mathtt{V_1}$ n'est pas une constante entière.

Le premier lemme, que nous ne démontrerons pas ici, est le *lemme de conservation des types par réduction* selon lequel chaque fois qu'un terme clos `t` a le type `A` et se réduit en une étape sur un terme `u` ($\mathtt{t} \triangleright \mathtt{u}$), alors `u` a également le type `A`. On en déduit que chaque fois qu'un terme clos `t` a le type `A` et se réduit en un nombre quelconque d'étapes sur un terme `u` ($\mathtt{t} \triangleright^* \mathtt{u}$), alors `u` a également le type `A`.

On démontre ensuite qu'un terme de la forme `fun` ne peut pas avoir le type `nat` et, de même, qu'une constante entière ne peut pas avoir un type de la forme `A -> B`. Cela demande une simple récurrence structurelle sur la relation de typage.

On montre ensuite qu'un terme `t` irréductible et clos de type `nat` est une constante entière et qu'un terme `t` irréductible et clos de type `A -> B` a la forme `fun`. Pour cela, on fait la démonstration suivante par récurrence sur la structure du terme `t`.

Comme `t` est un terme clos, ce ne peut pas être une variable. Comme il est irréductible, ce ne peut être ni un `fix` ni un `let`.

On montre que `t` ne peut pas être une application, un opérateur arithmétique ou un test. Si `t` est une application `t = u v` alors `u` a un type de la forme