`C -> D`. Par hypothèse de récurrence, c'est un terme de la forme `fun`, et `t` est un radical, ce qui contredit l'hypothèse qu'il est irréductible. Si `t` est un opérateur arithmétique `t = u` $\otimes$ `v` alors `u` et `v` ont le type `nat`. Par hypothèse de récurrence, ce sont des constantes entières et `t` est un radical, ce qui contredit l'hypothèse qu'il est irréductible. Si `t` est un test `t = ifz u then v else w` alors `u` a le type `nat`. Par hypothèse de récurrence, c'est une constante entière et `t` est un radical, ce qui contredit l'hypothèse qu'il est irréductible.

Un terme `t` irréductible et clos est donc une constante entière ou un terme de la forme `fun`. S'il a le type `nat`, c'est une constante entière, s'il a le type `A -> B`, c'est un `fun`.

Si un terme clos bien typé se réduit sur un terme irréductible et clos, ce terme est lui-même bien typé et c'est donc une constante entière ou un terme de la forme `fun`.

### 5.2.2 En sémantique opérationnelle à grands pas

En sémantique opérationnelle à grands pas, cette propriété se formule un peu différemment. En effet, on peut montrer simplement que même dans le cas non typé, les seuls résultats que la sémantique peut attribuer à un terme sont des valeurs. En revanche, les règles de la sémantique opérationnelle de l'application, des opérateurs arithmétiques et du test peuvent être considérées comme incomplètes, car elles ne précisent pas comment calculer la valeur d'une application si la valeur de son membre gauche est une constante entière, la valeur d'un opérateur arithmétique si la valeur de l'un des arguments est un terme de la forme `fun`, ni la valeur d'un test si la valeur de son premier argument est de la forme `fun`. La propriété est ici que pour les termes typés ces règles sont complètes, c'est-à-dire que ces trois cas ne peuvent pas se produire.

On commence par montrer un *lemme de conservation des types par interprétation* selon lequel chaque fois qu'un terme clos `t` a le type `A` alors sa valeur, si elle existe, a également le type `A`. Ce lemme est le pendant du lemme de conservation des types par réduction de la sémantique opérationnelle à petits pas.

On montre ensuite, comme dans le cas de la sémantique opérationnelle à petits pas, 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`.

Comme on a vu que la valeur d'un terme est ou bien une constante entière ou bien un terme de la forme `fun`, on en déduit que la valeur d'un terme de type `nat` est une constante entière et que la valeur d'un terme de type `A -> B` est un terme de la forme `fun`. On en déduit que, lors de l'interprétation d'un terme bien typé, les membres gauches des applications s'interprètent toujours sur des termes de la forme `fun`, les arguments d'un opérateur arithmétique s'interprètent toujours sur des constantes entières et le premier argument d'un `ifz` s'interprète toujours sur une constante entière.

**Exercice 5.6** *(L'équivalence des sémantiques opérationnelles) Montrer que le calcul d'un terme typé a un résultat en sémantique opérationnelle à petits pas*