erreurs évoquées ci-dessus et de montrer que l'on peut définir simplement une sémantique dénotationnelle pour ce langage.

5.1 Les types

En mathématiques, n'importe quel ensemble peut être domaine de définition d'une fonction. Par exemple, on peut définir une fonction `m` de $2\mathbb{N}$ dans $\mathbb{N}$ qui à un nombre pair associe sa moitié. De ce fait, pour savoir si l'expression `m (3 + (4 + 1))` est bien formée ou non, c'est-à-dire si l'argument réel de la fonction est dans son domaine de définition, il faut déterminer si `3 + (4 + 1)` est un nombre pair ou non. Déterminer si un objet quelconque est dans un ensemble quelconque est un problème indécidable, et, de ce fait, la correction d'un terme est également un problème indécidable. Par ailleurs, pour savoir si le terme `ifz t then u else v` déclenche une erreur ou non, nous avons uniquement besoin de savoir si `t` s'interprète sur un entier ou sur un terme de la forme `fun`, la parité de cet entier importe peu.

Ces deux remarques amènent à restreindre la classe des ensembles que l'on peut utiliser comme domaine de définition d'une fonction. Un tel ensemble est appelé un *type*.

5.1.1 Le langage PCF avec des types

En PCF les types sont définis inductivement par

- `nat` — c'est-à-dire $\mathbb{N}$ — est un type,
- si `A` et `B` sont des types alors `A -> B` — c'est-à-dire l'ensemble des fonctions de `A` vers `B` — est un type.

Les types peuvent être exprimés dans un langage formé d'une constante `nat` et d'un symbole `->` à deux arguments ne liant pas de variables. Un tel terme est également appelé un *type*.

Quand on écrit une fonction en PCF, `fun x -> t`, il faut maintenant indiquer le type de la variable `x`. Ainsi, on notera `fun x:nat -> x` l'identité sur les entiers et `fun x:(nat -> nat) -> x` l'identité sur les fonctions des entiers dans les entiers. Plus généralement, le symbole `fun` prend maintenant deux arguments, le premier est un type et le second un terme, et il lie une variable dans le second. Le langage PCF avec des types est donc un langage à deux sortes d'objets : les termes et les types et l'arité du symbole `fun` est `((type), (terme, terme), terme)`. Les symboles `fix` et `let` doivent également indiquer le type de la variable qu'ils lient.

Le langage PCF typé contient donc

- un symbole de terme `fun` à deux arguments, dont le premier est un type et le second un terme, liant une variable de terme dans son second argument,
- un symbole de terme α à deux arguments, dont les deux arguments sont des termes, ne liant pas de variables dans ses arguments,
- une infinité de constantes de terme pour les entiers,