En sémantique dénotationnelle également, on a le choix de définir une fonction partielle ⟦ ⟧, telle que ⟦`fix x:nat (x + 1)`⟧ ne soit pas défini, ou alors ajouter une valeur $\perp$ à ⟦`nat`⟧ telle que ⟦`fix x:nat (x + 1)`⟧ `=` $\perp$.

Si on ajoute une telle valeur $\perp$, quand on interprète un terme de la forme `t + u`, on commence par interpréter `u` et `t` et si l'un des deux termes boucle, alors c'est également le cas du terme `t + u`. La sémantique dénotationnelle d'un terme de la forme `t + u` se définit ainsi

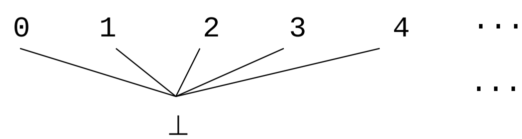
- ⟦`t + u`⟧ `=` ⟦`t`⟧ `+` ⟦`u`⟧ si ⟦`t`⟧ et ⟦`u`⟧ sont des entiers,
- ⟦`t + u`⟧ `=` $\perp$ si ⟦`t`⟧ `=` $\perp$ ou ⟦`u`⟧ `=` $\perp$.

On peut maintenant remarquer que la fonction ⟦`fun x:nat -> x + 1`⟧ qui n'avait pas de point fixe, quand ⟦`nat`⟧ ne contenait que les entiers en a désormais un : $\perp$, et que cette valeur est précisément celle que l'on veut attribuer comme sémantique au terme `fix x:nat (x + 1)`, qui boucle. La fonction ⟦`fun x:nat -> x`⟧, qui avait déjà plusieurs points fixes, en a un de plus $\perp$, et c'est ce point fixe là que l'on veut attribuer comme sémantique du terme `fix x:nat x`. La fonction ⟦`fun x:nat -> x + x`⟧ qui avait un point fixe unique `0` en a maintenant deux : `0` et $\perp$ et c'est ce second point fixe que l'on veut attribuer comme sémantique du terme `fix x:nat (x + x)`, qui boucle.

Toutes les fonctions que nous avions considérées ont maintenant des points fixes et quand elles en ont plusieurs, dont $\perp$, c'est ce dernier que l'on veut privilégier.

5.3.3 La relation d'ordre de Scott

Pour préciser cette idée, on définit la relation d'ordre suivante, la *relation d'ordre de Scott*, sur l'ensemble ⟦`nat`⟧



et, afin de privilégier $\perp$ sur les autres points fixes, on définit ⟦`fix x:nat t`⟧ comme le plus petit point fixe de la fonction ⟦`fun x:nat -> t`⟧. Reste à montrer que ce plus petit point fixe existe, et pour cela nous allons utiliser le théorème du point fixe. Pour pouvoir appliquer ce théorème, nous devons donc montrer que la relation d'ordre que nous avons définie sur ⟦`nat`⟧ est faiblement complète et que la sémantique d'un programme de type `nat -> nat` est toujours une fonction continue.

Plus généralement, nous allons construire pour chaque type `A`, un ensemble ⟦`A`⟧ muni d'une relation d'ordre faiblement complète et montrer que la sémantique d'un programme de type `A -> B` est une fonction continue de ⟦`A`⟧ dans ⟦`B`⟧.

Commençons par définir les ensembles ⟦`A`⟧. L'ensemble ⟦`nat`⟧ est par définition égal à $\mathbb{N} \cup \{\perp\}$, muni de la relation d'ordre ci-dessus. L'ensemble ⟦`A -> B`⟧ est par définition l'ensemble des fonctions continues de ⟦`A`⟧ dans ⟦`B`⟧ muni de la relation d'ordre `f` $\leq$ `g` si pour tout `x` dans ⟦`A`⟧, `f x` $\leq$ `g x`.