4
Le calcul comme une suite de petits pas

La définition de l'interpréteur, à la section 3.4.2, nous a donné une nouvelle manière d'aborder la notion de calcul, dans laquelle les programmes sont les expressions d'un langage $\mathcal{L}$. À partir d'un programme t et d'arguments p_1, ..., p_n, on construit un terme, qui est une expression d'un langage $\mathcal{L}'$, qui étend le langage $\mathcal{L}$. L'exécution des programmes est définie par une fonction calculable totale des termes de $\mathcal{L}'$ dans les termes de $\mathcal{L}'$. Cette fonction décrit un petit pas de calcul. On l'itère ensuite jusqu'à obtenir un entier, qui est le résultat du calcul, ou alors à l'infini, auquel cas le programme t ne termine pas sur les arguments $p_1, \ldots, p_n$. La notion de terminaison de la définition 3.12 rejoint ici l'acception courante : un programme qui ne termine pas est un programme qui calcule éternellement.

Les langages $\mathcal{L}$ et $\mathcal{L}'$ et la fonction qui décrit un petit pas de calcul définissent un langage de programmation et une fonction partielle f est dite *représentable* dans ce langage s'il existe un programme t tel que le terme formé du programme t et des entiers $p_1, \ldots, p_n$ se calcule en l'entier $f(p_1, \ldots, p_n)$ quand la fonction f est définie en p_1, ..., p_n et ne termine pas sinon. Il est facile de montrer que toutes les fonctions partielles représentables dans un tel langage sont calculables.

Dans certains langages de programmation, comme dans celui de la section 3.4.2, toutes les fonctions calculables sont représentables, on dit alors que ces langages sont *complets au sens de Turing*.

Entrent dans ce cadre les langages de programmation traditionnels comme Java, Caml, C, ... dont on peut décrire l'exécution des programmes comme une suite de petits pas.