Interpréteur LOGO en OCaml

Maxence Ahlouche

Table des matières

1 Présentation du projet

Le but de ce projet était de programmer, en monôme, les analyseurs syntaxiques et sémantiques d'un interpréteur du langage LOGO, le reste des fonctions nécessaires étant fourni. Ce rapport présente la démarche de conception et de programmation qui a été suivie pour ce projet.

Ce programme a été réalisé, pour les parties non fournies par les professeurs, en programmation purement fonctionnelle, à l'aide du langage OCaml. Il a pour objectif de simuler le comportement d'une tortue LOGO, qui servait à initier à la programmation de manière ludique.

Le langage LOGO tel que nous l'avons étudié ne comporte que quatre instructions élémentaires :

- JUMP, qui permet de déplacer la tortue sans dessiner ;
- MOVE, qui déplace la tortue tout en dessinant ;
- ROTATE, qui permet de diriger la tortue ;
- COLOR, qui change la couleur du crayon.

Il offre également plusieurs structures de contrôle :

- IF (...) THEN (...) ELSE (...), qui permet d'exécuter certaines instructions seulement si une certaine condition est vérifiée ;
- REPEAT (...), qui permet de répéter un bloc d'instructions autant de fois que souhaité ;
- DEF (...), la définition de procédures, qui introduit notamment la possibilité de dessiner facilement des fractales.

Les fonctions développées dans le cadre du projet transforment la liste de mots du programme LOGO (déjà parsée) en une liste de commandes interprétables par la fonction d'affichage.

2 Le programme

Cette section couvre les différentes étapes de réalisation de ce projet, selon leur ordre chronologique.

2.1 Conception

L'étape de conception a été très courte, du fait que la structure du programme était donnée avec le sujet. Elle consistait principalement en la définition des types nécessaires au passage des informations entre les analyseurs syntaxique et sémantique. J'ai donc commencé par définir mon type `programme`, qui était composé d'une liste de `procedure` et d'un arbre de `sous_programme`. Comme j'avais l'intention de produire un résultat, même basique, le plus tôt possible, je ne me suis pas occupé de définir le type `procedure`, et me suis contenté de le déclarer. Un `sous_programme` peut être soit une instruction élémentaire (Move, Jump, Rotate ou Color), soit une structure de contrôle :

- If : il est alors composé d'une condition, d'un premier bloc d'`instruction` qui sera exécuté si la condition est vérifiée, et un second bloc d'`instruction` qui sera exécuté dans le cas contraire.
- Repeat : celui-ci est formé d'une expression indiquant le nombre de fois que le bloc d'instructions devra être exécuté, et du bloc d'`instruction` en question.
- Call : le nom de la fonction appelée et la liste des valeurs des paramètres passés à cette fonction forment ce `sous_programme`.

2.2 Développement

2.2.1 L'analyseur syntaxique

2.2.2 L'analyseur sémantique

2.3 Tests