Introduction au meta-programming

Quelques concepts de base

Sommaire

Introduction
Qu'est-ce que le meta-programming

- Capacité d'un programme à se modifier lui-même
  - Introspection
  - Réflexion
  - Génération de code
- Programmes qui créent des programmes
  - Programmes en tant que structures de données

Introduction
Avantages du meta-programming

- Élimination du code "boilerplate"
  - Certains programmes ont du code répétitif
  - Possibilité de générer automatiquement ce code
- Algorithmes complexes
  - Un algorithme peut se modifier lui-même selon la situation
  - Piste d'intelligence artificielle abandonnée dans les années 80
- Langage évolutif
  - Possibilité d'ajouter de nouvelles fonctionnalités à un langage existant
  - Une librairie peut modifier le langage (e.g. ajouter la notion d'Aspects)
  - Pas besoin d'outil de troisième parti (e.g. AspectJ)

Introduction
Inconvénients du meta-programming

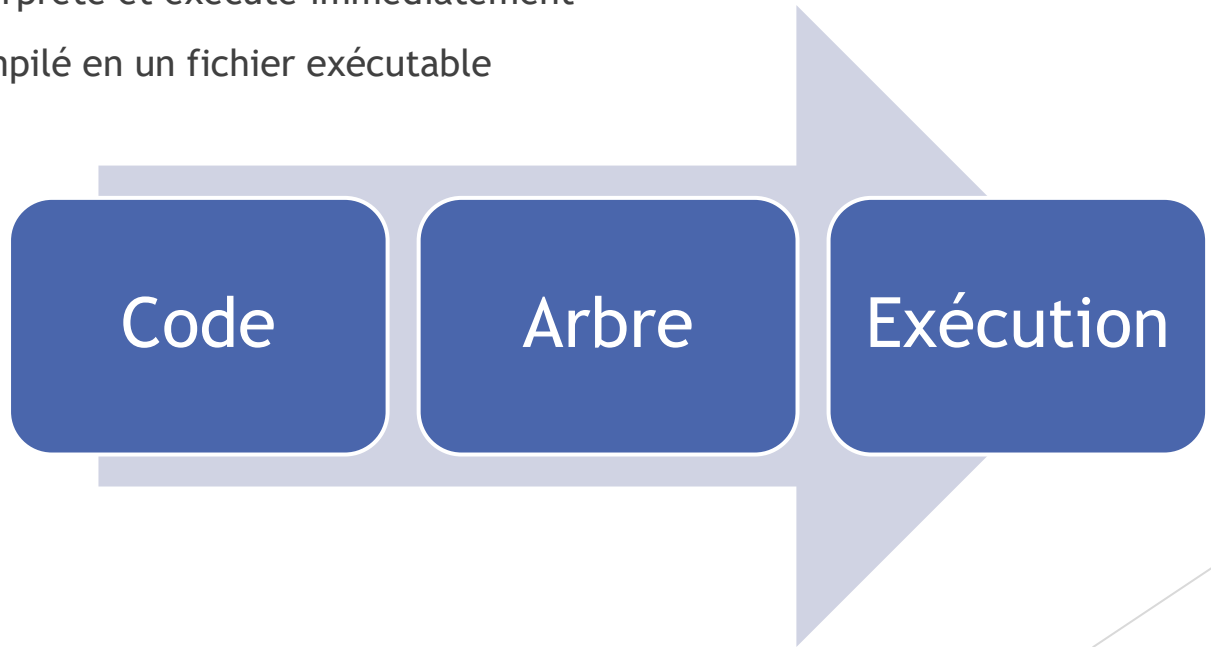
- Dans les mauvaises mains, complexifie le code énormément
  - Chaque programme a le potentiel de définir son propre langage
  - Chaque fichier dans un programme peut utiliser une syntaxe différente
- Peut de support au niveau des outils
  - Champ de recherche peu exploré
  - Difficulter de déboguer des macros
- Complexité générale de générer du code
  - Un algorithme qui génère un autre algorithme est intrinsèquement complexe

Introduction
Langages avec meta-programming

- Common Lisp
- Scheme
- Racket
- Clojure
- Template Haskell
- Scala
- Rust

Structure d'un langage de programmation

- La plupart des langages populaires utilisent la même structure générale pour la compilation et l'exécution de leur code
  - Un programme est lu, analysé et compilé en un arbre de syntaxe abstrait
  - Dépendamment du type de langage, cet arbre est soit
    - Interprété et exécuté immédiatement
    - Compilé en un fichier exécutable

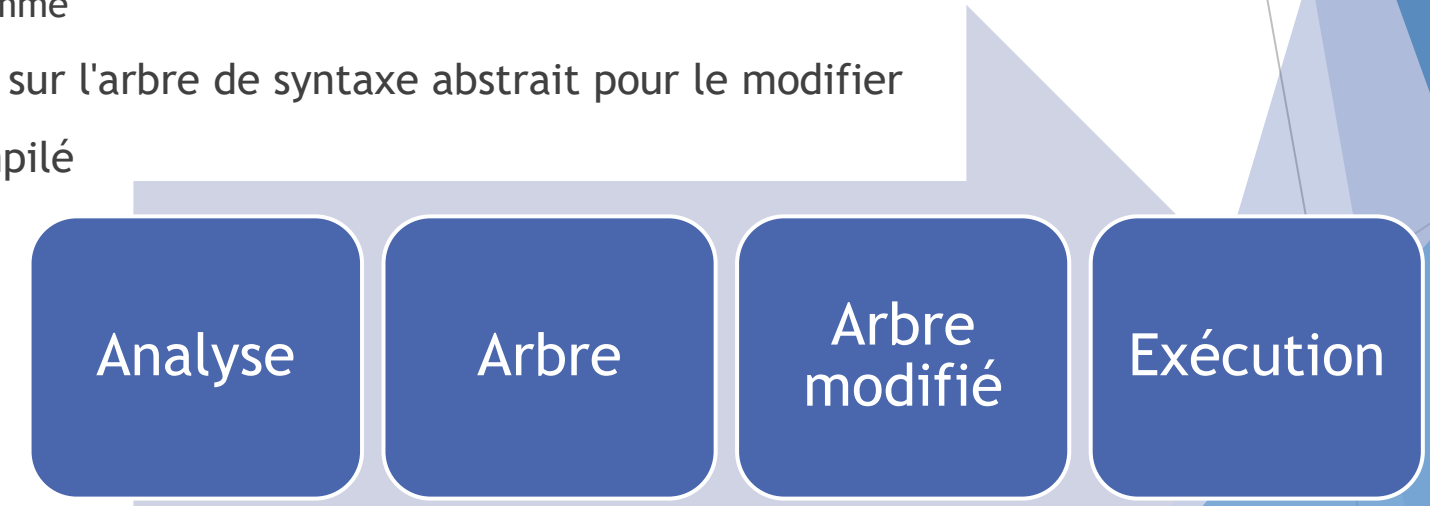


Structure d'un langage de programmation
Arbre de syntaxe

- L'arbre de syntaxe généré est une structure de données
  - Elle peut être parcourue, modifiée et interprétée de différentes façon
- Typiquement, les compilateurs exécuteront des optimisations sur cet arbre
  - Élimination des variables non-utilisées
  - Optimisations des boucles
  - Tail Call Optimization
  - Etc.
- Pourquoi pas donner la possibilité au programmeur d'accéder à cet arbre?

Structure d'un langage de programmation Compilateur avec macros

- Un langage supportant le meta-programming, spécifiquement avec le support macro, ajouter une étape au processus de compilation/interprétation
  - Un programme est lu, analysé et compilé en un arbre de syntaxe abstrait
    - Certaines sections du programme sont reconnues comme étant des macros, dont l'objectif est de modifier le programme
  - Les macros sont exécutées sur l'arbre de syntaxe abstrait pour le modifier
  - L'arbre est exécuté ou compilé



Macros
Qu'est-ce qu'une macro?

- Une macro est un fonction ou un programme qui prend en entrée du code et le transforme en un code différent
- Il existe différents types de macros dans les langages de programmation
  - C : Macro préprocesseur
  - C++ : Templates
  - Racket/Scheme : Macro hygiénique
  - Common Lisp : Macro syntactique

Macros
Différents types de macros

- Macro préprocesseur
  - Utilisé par C/C++
  - Permet de transformer le texte du code
  - Utilise son propre langage différent de C/C++
  - Aucun outil fourni pour l'analyse de code
- Template
  - Utilisé par C++
  - Permet de générer du code dynamiquement selon un patron prédéfini
  - Utilise une syntaxe unique, mais s'intègre dans le code C++
  - Très rigide et structuré
  - Pas d'analyse de code possible

Macros
Différents types de macros

- Macro hygiénique
  - Utilisé par les langages comme Racket, Scheme, Template Haskell, Scale, etc.
  - Reçoit une structure de données représentant le code
  - Utilise la même syntaxe que le langage pour modifier cette structure
- Macro syntactique
  - Utilisé par Common Lisp
  - Reçoit une structure de données représentant le code
  - La structure de données est homoiconique
  - Utilise la même syntaxe que pour une fonction pour modifier la structure
  - Le code résultant s'inscrit de manière syntactique dans le programme
    - Il a accès au contexte du programme pour son exécution

Langage de domaine spécifique

- Avec le metaprogramming, il est possible de modifier un langage de programmation en lui ajoutant de nouvelle fonctionnalités
  - Nouveaux mots clés
  - Redéfinition des mots clés existant
- Il est donc possible de créer un nouveau langage à partir d'un langage existant
  - Adapter un langage à une situation particulière
  - Langage de domaine spécifique

Langage de domaine spécifique

- Quelques situations où un langage de domaine spécifique peut être de mise
  - Définir une interface graphique (HTML)
  - Gérer les communications entre appareils (TCP/IP)
  - Effectuer des requêtes dans une base de données (SQL)
  - Définir des interactions logiques (Prolog)
  - Bâtir une intelligence artificielle (?)

Références

- Doug Hoyte, *Let Over Lambda*, Lulu.com, 2008.
- Peter Seibel, *Practical Common Lisp*, 1st Corrected ed., Apress, 2005. https://gigamonkeys.com/book/