UCLouvain

Résumé de LINFO1104

compilation du 12 février 2023

Thomas Debelle

Juin 2023

Table des matières

Préface

Bonjour à toi !

Cette synthèse recueille toutes les informations importantes données au cours, pendant les séances de tp et est amélioré grâce au note du Syllabus. Elle ne remplace pas le cours donc écoutez bien les conseils et potentielles astuces que les professeurs peuvent vous donner. Notre synthèse est plus une aide qui on l'espère vous sera à toutes et tous utiles.

Elle a été réalisée par toutes les personnes que tu vois mentionné. Si jamais cette synthèse a une faute, manque de précision, typo ou n'est pas à jour par rapport à la matière actuelle ou bien que tu veux simplement contribuer en y apportant ta connaissance ? Rien de plus simple ! Améliore la en te rendant ici où tu trouveras toutes les infos pour mettre ce document à jour. (*en plus tu auras ton nom en gros ici et sur la page du github*)

Nous espérons que cette synthèse te sera utile d'une quelconque manière ! Bonne lecture et bonne étude.

Chapitre 1

Introduction

1.1 Les Paradigmes

Une paradigme, est une façon d'approcher et apporter une solution à un problème. De ce fait, chaque langage de programmation utilise 1 voir 2 paradigmes. Ce cours couvrat 5 paradigmes cruciaux qui sont :

1. "Functionnal Programming"
2. "Object Oriented Programming"
3. "Functional DataFlow Programming"
4. "Actor DataFlow Programming or Multi-Agent"
5. "Active Objects"

Et pour découvrir ces paradigmes, nous utiliserons les langages de programmations "Oz" qui est un langage de recherche multi paradigme ainsi que "Erlang".

Chapitre 2

Les différents Paradigmes

Comme mentionner plus haut, on rencontrera 5 paradigmes dont voici le premier.

2.1 Functional Programming

Avec ce paradigme, on impose qu'une variable peut être nommé qu'une seule fois ! Donc : X = 10 mais on ne peut pas plus loin dire X = 9. X est déjà attribué. On peut penser que cela risque d'être handicapant alors qu'en réalité, cela rend notre code plus simple à débugger. De plus, nombreux sont les langages et microservices utilisés qui implémentent la programmation fonctionnelle. Formellement, quand on déclare une variable et qu'on l'assigne à une valeur ceci se passe. Une chose importante à noter est que cette façon de programmer peut être réalisé dans n'importe quel langage de programmation. On peut également redéclarer un identificateur. C'est-à-dire écrire "X = 42" et plus loin en ayant redéclarer une variable "X = 11" car ces deux déclarations pointent à deux éléments totalement différents dans la mémoire.

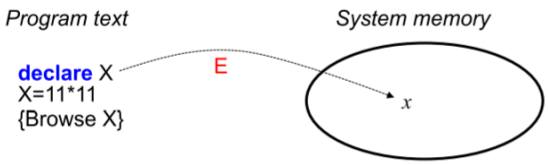

FIGURE 2.1 – Déclaration d'une variable

Un "Scope" ou portée est une propriété centrale en programmation. En effet, c'est le scope qui nous permet d'avoir différente valeur pour des variables qui ont le même nom. Naturellement, elle ne représente pas la même chose car elle diffère de leur scope. On peut déterminer le scope d'une variable sans même exécuter le code. Il nous suffit d'analyser le code qui comprend un "lexical scoping" ou un "static scoping".

```
local
   X
in
   X = 42  {Browse X}
   local
      X
   in
      X = 11  {Browse X}
   end
   {Browse X}
end
```

FIGURE 2.2 – Exemple de code avec des scope différents

2.2 Conseils pour la syntaxe d'Oz