

Introduction

Sémantique des langages informatiques

Didier Buchs

Université de Genève

18 février 2019

Info Pratiques

- Cours : Mardi 10h15 à 12h
- Exercices : Lundi 16h15 à 18h
- Site du cours : https://gitlab.unige.ch/semantique/semantique_2019
- Note finale = $(\text{Notes TPS} + 2 * \text{Note examen écrit}) / 3$
- Condition de passage : 3.0 aux TPS
- pas de rattrapage des TPS
- Assistant : Damien Morard bureau : 221

Introduction

- Langages informatiques et de programmation
- Pourquoi étudier les langages de programmation ?
- A quoi servent-ils ?
- Historique
- Classification par paradigmes
- Petite bibliographie

Quelques types de langages informatiques

- Modélisation : UML
- Spécification formelle : Z, Réseaux de Petri, Types abstraits algébriques
- Programmation : assembleurs, Java, Prolog, Lisp
- Script : DOS, macros dans Excel
- Interrogation/mise à jour de bases de données : SQL
- Description de circuits électroniques : VHDL
- Mise en page : HTML, TEX, PostScript
- Méta-langage : XML, souvent associé à XSL pour le web, JSON

Ce cours se concentre sur les langages de programmation de haut niveau, de modélisation et comment en donner une sémantique

Pourquoi étudier les langages ?

Pour comprendre :

- quel modèle conceptuel est le plus approprié pour traduire un problème du monde réel en un programme
- comment décomposer un programme informatique en éléments plus faciles à gérer
- comment décrire un langage de programmation
- comment un langage devrait être implémenté
- quelles facilités un langage devrait fournir pour simplifier la vie du programmeur (et lui éviter des erreurs)

... et pour avoir le recul nécessaire pour être un vrai professionnel et éviter les guerres de religion à leur sujet !

Pourquoi les langages informatiques ?

- Un langage sert à communiquer
- Un langage informatique doit être précis, non-ambigu
- Le langage parlé, les textes de lois, les textes sacrés sont truffés d'ambiguïtés
- Langage formel = absolument sans ambiguïté
- La résolution de problèmes informatiques consiste à :
 - factoriser ce qui est commun à certaines classes de problèmes
 - gérer les variantes, donc les différences entre ces problèmes
- Pour simplifier la tâche du programmeur, les langages de haut niveau cherchent à intégrer ces aspects, par leurs structures de données et de contrôle.

Pourquoi des langages de haut niveau ?

- Chaque famille de processeur possède un langage binaire qui lui est propre : le langage machine
 - 00000010101111001010
 - 00000010111111001000
 - 00000011001110101000
- Pour chaque langage machine, on peut avoir une variante symbolique, dite langage d'assembleur
 - LOAD I
 - ADD J
 - STORE K

alors qu'il est tellement plus facile et lisible d'utiliser :
 $k = i + j$ (mais il a fallu plus de 10 ans pour y arriver !)

Abstraction = élimination d'erreurs, augmentation de la productivité (lignes/an constantes) , programmes plus efficaces, portabilité

Qu'est-ce qu'un langage de programmation ?

- Un programme est la spécification d'un calcul (algorithme).
- Un langage de programmation est une notation pour écrire des programmes ; il permet d'écrire des phrases, formées d'une suite de mots appelés unités syntaxiques.
- La syntaxe du langage régit la forme des phrases.
 - Pascal correct : If $3+5=9$ then write ('bravo')
 - non correct : If $3+5=9$ then write 'bravo'
- La sémantique statique, ou typage vérifie que les phrases aie un sens.
 - Non correct : if $3=false$ then write ('bravo')
- La sémantique du langage (dynamique) est la signification véhiculée par les phrases valides du langage.

Compilateurs et Interprètes

- Un interprète (ou interpréteur) lit, puis exécute immédiatement le texte source
- Un compilateur prend un texte source et le traduit dans un langage cible ; le résultat est un fichier objet, dont l'exécution est remise à plus tard

Petit historique des langages de programmation

https://fr.wikipedia.org/wiki/Chronologie_des_langages_de_programmation

195x Fortran, Algol, Lisp

196x Cobol, Simula, Logo

197x Pascal, SQL, Prolog, C, Smaltalk

198x Ada, C++, Objective-C, Eiffel, Perl, Caml

199x Haskell, Python, Ruby, Lua, Java, PHP, JavaScript, Erlang

200x C#, Scala, Clojure, Go

201x Rust, Kotlin, Swift

A consulter

Panorama des paradigmes courants :

- https://en.wikipedia.org/wiki/Programming_paradigm

Une perspective du futur des langages :

- <http://channel9.msdn.com/Blogs/pdc2008/TL57>
- <https://medium.freecodecamp.org/best-programming-languages-to-learn-in-2018-ultimate-g>

Un peu de culture divertissante :

- 99 bottles of beer (comparaison humoristique de langages de programmation) <http://www.99-bottles-of-beer.net>

Programme du cours

- Introduction
- Programmation logique et fonctionnelle
- Sémantique des langages
 - Induction
 - Types de sémantique (évaluation, computationnelle, transformationnelle)
 - Sémantique d'expression arithmétique
 - Variables
 - Structures de contrôles
 - Fonctions
 - Evaluation paresseuses
- Analyse de programmes
- Sémantique des langages logiques
- Systèmes de types
- Visibilité et gestion de la mémoire

Bibliographie générale

- Concepts in Programming Languages, John C. Mitchell, Cambridge Univ Press, 2002, 450 pages, ISBN 0521780985
- Programming Language Pragmatics, Michael L. Scott, Morgan Kaufmann Publishers, October 1999, 880 pages, ISBN 1-55860-442-1
- Comparative Programming Languages, Robert Clark and Leslie B. Wilson, Addison-Wesley, 3ème édition, 2000, 384 pages, ISBN 0-201-71012-9
- Formal Semantics of Programming Languages, Peter D. Mosses, Electronic Notes in Theoretical Computer Science (ENTCS Volume 148 Issue 1, February, 2006, Pages 41-73)
- The Formal Semantics of Programming Languages : An Introduction, G. Winskel, MIT Press (1993)