

# ***Introduction***

*version du September 2, 2003 – 11 : 11*

# Qu'est-ce que la sémantique?

---

**Sémantique:** *Étude du sens, de la signification dans le langage.*

Dictionnaire Robert.

## Pourquoi la sémantique? 1/2

---

Définir la sémantique d'un langage formel consiste à lui donner une signification mathématique, dans deux buts.

**La description**, qui doit permettre de cerner très précisément ce que réalise un programme (ou un circuit)

- afin de permettre au programmeur de comprendre ce qu'il fait,
- afin de réaliser des traducteurs (compilateurs, interprètes) et des optimiseurs,
- afin de pouvoir appliquer des méthodes formelles de vérification,

## Pourquoi la sémantique? 2/2

---

- La prescription**, (dans les sens du verbe **prescrire**) qui doit aider les concepteurs de langages à définir des langages cohérents, puissants et corrects, (qui se souvient du monstre PL/1?). Ceci est très important avec la définition des **langages de domaines spécifiques**, par exemple
- un langage pour aider les ingénieurs à calculer les trajectoires des satellites,
  - un langage pour commander un micro-onde lors de la cuisson d'un plat surgelé.

## Un concept de sémantique: la transparence référentielle 1/3

---

La sémantique a mis en évidence un concept important: la **transparence référentielle**.

Un langage formel est **référentiellement transparent** si, dans la même portée, la signification d'une portion de code est indépendante de sa position dans le code.

## Un concept de sémantique: la transparence référentielle 2/3

---

Ainsi CAML n'est pas référentiellement transparent.

On s'attend à ce que pour chaque invocation

```
de (f 1) * (f 1) - (f 1) * (f 1) dans
(f 1) * (f 1) - (f 1) * (f 1);;
(f 1) * (f 1) - (f 1) * (f 1);;
(f 1) * (f 1) - (f 1) * (f 1);;
(f 1) * (f 1) - (f 1) * (f 1);;
(f 1) * (f 1) - (f 1) * (f 1);;
(f 1) * (f 1) - (f 1) * (f 1);;
```

on obtienne la valeur 0.

## Un concept de sémantique: la transparence référentielle 3/3

---

Mais en fait on obtient successivement 14 , 30 , 46 , 62 , 78  
si on définit f par

```
let r = ref 0;;  
let f x = (r := !r+1; x + !r);;
```

# Les types de sémantique

---

**opérationnelle**, on définit la sémantique par sa mise en œuvre sur une machine abstraite (ou quelque chose qui lui ressemble comme un système de réduction), on distingue

- la **sémantique en grandes étapes**, qui associe au programme une transition données-résultats,
- la **sémantique en petites étapes**, qui décrit le comportement du programme étape par étape,

**dénotationnelle**, on associe à chaque construction syntaxique un objet mathématique qui est défini par récurrence sur la structure

**axiomatique**, chaque construction est décrite par la manière dont elle transforme des propriétés ou des prédicats.



## Et la syntaxe dans tout ça ?

---

«*La sémantique n'a rien à faire du sucre syntaxique.*»  
d'après Christoffer Strachey

La sémantique suppose que tout le travail d'analyse syntaxique est fait.

Elle travaille au niveau de la syntaxe abstraite : objet structuré et/ou arborescent construit à partir de l'arbre d'analyse syntaxique, lui-même construit à partir du texte linéaire du programme.

## Le plan (provisoire) du cours 1/3

---

- La sémantique d'un langage impératif,
  - la sémantique opérationnelle,
  - la sémantique dénotationnelle,
  - la sémantique axiomatique

} Le but est d'étudier un cas  
particulier

## Le plan (provisoire) du cours 2/3

---

- La sémantique du lambda calcul,
  - la sémantique opérationnelle,
    - \* les substitutions explicites,
    - \* la sémantique à grandes étapes,
    - \* les machines abstraites,
    - \* le style de passages de continuations,
  - la sémantique dénotationnelle,
    - \* les modèles par retracts,
    - \* le modèle de Scott

} Le but est d'introduire des concepts

## Le plan (provisoire) du cours 3/3

---

- Un modèle de concurrence: le  $\pi$ -calcul.      Un modèle important