« Paradigmes et Interprétation »
L3 – UEF S6

ECTS : 4

Nombre d'heures : 20h CM / 0h TD / 20h TP

Équipe pédagogique : JP. Roy, S. Julia

Objectif : Approfondir les principaux paradigmes de programmation en implémentant une galerie d'interprètes de petits langages pour chaque paradigme.

Prérequis : Connaissance de base du langage Scheme et d'un second langage impératif à objets.

Programme. (I) On commence par une révision du langage Scheme, étudié en 1ère année, qui sera le langage support à l'écriture des interprètes. Une exposition des analyseurs lexicaux et syntaxiques permettra d'analyser un texte de programme pour produire un arbre syntaxique. Les premiers interprètes forment le *noyau* de tout langage de programmation, avec l'arithmétique, les variables locale et leur portée, la notion de fonction (fermeture) et les environnements, le tout basé sur une sémantique « stricte ». Les conditionnelles sont introduites, mais la récursivité exige une construction spéciale de point fixe, qui est rajoutée à l'interprète et met l'accent sur les environnements cycliques. On étudie en intermède diverses structures de données possibles. (II) La notion d'*état* d'un calcul est introduite, avec les données mutables. On présente un modèle de mémoire, sur le modèle de la sémantique dénotationnelle. L'envoi de messages entre objets à états locaux s'insère naturellement dans ce cadre conceptuel. (III) On se penche sur le *contrôle*, en introduisant les continuations à partir d'une analogie avec le Web et l'on montre comment dérécursiver toute fonction avec des passages explicites de continuations, en style CPS. Les échappements et les exceptions sont modélisées grâce aux continuations, ainsi que les mécanismes producteur/consommateur. Le cas particulier de l'instruction GOTO est étudié. On montre comment réifier une continuation et la rendre explicite au programmeur. (IV) Une introduction - avec le langage Haskell - est donnée ensuite à la sémantique paresseuse (« lazy »), et on présente l'analogie avec le langage d'un shell Unix, puis l'on s'interroge sur l'implémentation d'une stratégie paresseuse dans un interprète. (V) Si le temps imparti le permet, on jettera un œil sur l'unification, avec des applications à l'inférence de type et aux requêtes logiques.

Supports : Cours sur le Web, augmenté des références suivantes :
- « Premiers cours de programmation avec Scheme », JP. Roy, 2010.
- « Programming languages. Application and Interpretation », S. Krishnamurti, 2007.

Compétences : (I = initiation, U = utilisation, M = maîtrise)

- scientifiques
  - *Faire preuve de capacité d'abstraction (U),*
  - *Mettre en œuvre une démarche expérimentale (M).*

-transversales
  d) *Effectuer une recherche d'information (U),*
  e) *Travailler en équipe (U) : s'intégrer, se positionner, collaborer.*

Modalités de contrôle des connaissances :

2 projets en binôme + examen final