Institut
Mines-Télécom

Paradigmes et langages non classiques

INF355

Samuel TARDIEU <sam@rfc1149.net>
12 mai 2014

Exposé du problème

- Tout le monde connaît les langages classiques :
  - Java
  - C
  - PHP
  - Python
- Ces langages sont très similaires.
- La plupart des programmeurs et des entreprises en utilisent le plus petit dénominateur commun.

(Q) Pourquoi les concepts et langages plus avancés ne percolent-ils pas dans les formations informatiques classiques ?

(R1) Parce que les entreprises hésitent à les adopter de peur de ne pas trouver d'ingénieurs formés.

(R2) Parce que les enseignants n'y forment pas les ingénieurs, de peur que cela soit du temps perdu.

Pourquoi ce cours ?

- Il est utile d'être polyglotte. Les abstractions sont modelées par le langage.
- La plupart des paradigmes offerts par un langage peuvent être facilement utilisés dans d'autres langages.
- La simplicité est souvent la clé du succès : la majorité des abstractions étudiées s'expriment très simplement.
- L'informatique est belle, il faut la faire connaître.

Les langages

Les langages de l'occurrence 2014, seront :

Haskell langage fonctionnel, fortement et statiquement typé, non objet, mettant l'accent sur l'absence d'effets de bord (pureté) et l'évaluation paresseuse.

Scala langage fonctionnel, fortement et statiquement typé, objet avec héritage multiple, interopérable avec Java, promouvant la programmation parallèle et réactive.

Factor langage concaténatif, à pile, fortement et dynamiquement typé, supportant de nombreux modèles de programmation objet, privilégiant la compacité et l'expressivité.

Tous ces langages disposent d'une interface interactive.

Qu'allons-nous voir (1/2) ?

- Les types de données algébriques (Haskell, Scala)
- Les catamorphismes, anamorphismes, hylomorphismes (Haskell, Scala)
- Les typeclasses (Haskell)
- Les tests unitaires auto-générés (Haskell, Scala)
- Les traits et la linéarisation de types (Scala)
- Le typage par prédicats et mixins (Factor)
- Les types dynamiques
- La covariance et la contravariance (Scala)
- Les langages spécifiques aux domaines (Scala, Factor)
- La programmation *point-free* (Haskell)

Qu'allons-nous voir (2/2) ?

- Les fonctions partielles et leur combinaison (Haskell)
- Les fonctions en tant qu'objet de premier ordre et les fermetures transitives (Scala, Factor)
- La factorisation de code (Factor)
- La programmation concaténative à pile et les combinateurs (Factor)
- La programmation par agents asynchrones (Scala)
- La programmation réactive (Scala)
- La programmation monadique (Haskell)
- Les environnements par image (Factor)
- L'évaluation lors de la compilation (Scala, Factor)
- L'évaluation paresseuse (Haskell)
- Les continuations (Factor)

Que ne verrons-nous pas ?

- Les langages “jouets” (Intercal, Brainfuck, Befunge, Whitespace).
- Les langages de tableau (APL ou J) : ceux-ci peuvent néanmoins être utilisés en projet.
- Les langages moins évolués ou proches (Forth est remplacé utilement par Factor, Lisp et Clojure par Scheme).

Organisation et charge de travail

- Les cours sont interactifs et vous êtes invités à reproduire et étendre ce qui est fait.
- La présence aux cours est fortement conseillée.
- La présence aux cours suffit pour peu que le cours soit relu.
- Les exercices (TD assistés) alternent avec les cours.
- Une partie des heures sera consacrée à un projet personnel (ou à deux), qui devra faire intervenir des concepts ou langages non classiques (vus en cours ou non).

Exemple de projets passés (1/2)

- Un multi-user dunjeon (MUD) réparti extensible écrit en Erlang avec des agents autonomes.
- Un traducteur Scheme vers Javascript, écrit en Scheme et exécuté en Javascript.
- Un serveur IRC réparti tolérant aux pannes en Erlang.
- Résolution de problèmes ACM en Haskell.
- Pilotage d'un bras mécanique à travers un langage dédié en Factor.
- Implémentation d'une bibliothèque de gestions du parallélisme basée sur les continuations en Factor.
- Un analyseur et réducteur de complexité au sens de Kolmogorov en Factor.

Exemple de projets passés (2/2)

- Une bibliothèque d'algorithmie génétique en Scala.
- Un visualisateur de carte Minecraft en Scala.
- Un vote de Condorcet implémenté en OPA.
- Videolift pour gérer les vidéos de Rezel en Scala avec le framework Lift.
- Un typetracker pour la théorie des types dépendants en Haskell.
- Un typeur pour les expressions Haskell complexes.

Et maintenant ?

- Installez tous GHC, une implémentation de Haskell libre et gratuite, disponible pour GNU/Linux, Microsoft Windows et OS/X.
- Crééz chacun un dépôt Git auquel je dois avoir accès (`bitbucket.org` vous offre des dépôts privés si vous vous inscrivez avec votre adresse `telecom-paristech.fr`, mon compte chez eux est `samueltardieu`), puis envoyez moi un mail avec l'adresse de votre dépôt (en mettant `INF355` dans le sujet).