Département d'informatique
et de recherche opérationnelle
Hiver 2012

Marc Feeley
bureau 3341
feeley@iro.umontreal.ca

Projet

IFT3065 et IFT6232

1 Rappel du plan de cours

50% de la note finale est attribuée à la réalisation d'un projet, seul ou par équipe de deux. Le projet consiste à développer un nouveau compilateur complet ou d'étendre un compilateur existant (par exemple pour y intégrer des phases d'analyse et d'optimisation). D'ici le **16 janvier** vous devez élaborer une spécification de votre projet et me le soumettre pour approbation par courriel. Cette spécification doit indiquer vos objectifs et donner une bonne idée de la méthodologie (quel langage sera traité? quel compilateur sera étendu? quelles parties? dans quel but?). Vous devrez me remettre deux rapports d'étape (le **13 février** et le **12 mars**) et un rapport final (le **9 avril**) qui explique ce qui a été accompli, les performances de votre compilateur, une batterie de tests pour exercer le compilateur, etc. Vous devez également faire une présentation sur votre compilateur à la fin du cours (pondération: 1/5 de la note du projet).

2 Détails supplémentaires

Vous pouvez développer un compilateur à partir de zéro ou étendre un compilateur existant. Dans les deux cas vous devez choisir un ensemble de tâches de compilation à réaliser parmis les choix ci-dessous qui totalisent un niveau de difficulté approprié pour le cours (i.e. un niveau plus difficile pour IFT6232). Le but pédagogique du projet est de vous donner une expérience pratique de réalisation de compilateur. Le compilateur final devra pouvoir compiler des programmes simples (par exemple: trier un ensemble de nombres, compter le nombre de mots dans un texte, générer un labyrinthe aléatoire, ...).

Chaque phase du projet doit être documentée dans un rapport contenant une description claire des objectifs visés par cette phase (la spécification du problème), la méthodologie adoptée pour atteindre les objectifs, une discussion des problèmes informatiques rencontrés et l'approche utilisée pour les résoudre (algorithmes), une section présentant les résultats des suites de tests utilisées, et une évaluation honnête de l'atteinte des objectifs (qu'est-ce qui a été accompli et les problèmes restants). S'il reste des problèmes à la fin d'une phase, il faut tenter de les régler pour la phase suivante de sorte que le compilateur final soit fiable. N'hésitez pas à utiliser des diagrammes et choisir quelques bons exemples pour illustrer vos propos dans le rapport. La présentation du rapport doit être sobre (pas de fontes énormes, ni des grosses tables et des diagrammes touffus!). Utilisez un logiciel de traitement de texte (Latex est fortement recommandé!). Je m'attend à des rapports #1 et #2 de 8 à 10 pages pour IFT3065 et 12 à 15 pages pour IFT6232. Le rapport final (#3), qui porte sur les 3 phases, doit avoir de 12 à 15 pages pour IFT3065 et 15 à 20 pages pour IFT6232. Le rapport final doit récapituler ce qui a été fait dans les phases précédentes.

3 Choix d'un langage

La réalisation d'un compilateur pour les langages "industriels" (tel Java, C, C++, Python, JavaScript) est une tâche laborieuse vu la complexité sémantique de ces langages. Pour que l'objectif pédagogique du projet soit atteint dans le temps disponible il est important de se limiter à un petit langage source, ou à un sous-ensemble d'un langage industriel (par exemple, pour C++, retirer les templates, l'héritage multiple, la surcharge des opérateurs, ...).

Pour cette raison il est fortement recommandé d'utiliser **Scheme** comme langage source car il est basé sur un petit nombre de concepts puissants. Les langages Pascal et C sont aussi intéressants vu leur simplicité relative, mais il faudra quand même se limiter à un sous-ensemble. Le langage hôte du compilateur doit être Scheme, sauf si vous choisissez de faire un compilateur autogène (qui peut se compiler lui-même).

Voici des compilateurs existants parmis lesquels vous pouvez choisir :

- Scheme
  - Gambit-C
    (`http://gambit.iro.umontreal.ca`)
  - Javascript Scheme
    (`http://dynamo.iro.umontreal.ca/~gambit/wiki/index.php/Dumping_Grounds`)
- JavaScript
  - Tachyon et Photon (des compilateurs JavaScript autogènes conçus au DIRO, voir Erick Lavoie et Maxime Chevalier-Boisvert)
- C
  - Tiny C (`http://bellard.org/tcc/`)
  - Small C (`http://www.cpm.z80.de/small_c.html`)
- Pascal
  - Pascal-S (`http://www.246.dk/pascals.html`)

Sachez que l'extension d'un compilateur existant n'est pas nécessairement plus simple que la conception d'un compilateur à partir de zéro car il faut mettre un certain temps pour comprendre comment s'y intégrer. Cependant si vous comprenez bien sa structure vous pouvez profiter de l'infrastructure existante et ainsi éviter l'implantation et le déboguage de certaines parties moins intéressantes. Il est sage d'investir un certain temps à étudier le code source des compilateurs ci-dessus avant de faire un choix pour le projet.

Votre compilateur peut cibler une machine réelle (Intel x86 ou ARM), une machine virtuelle (JVM) ou un langage de bas niveau (C).

4 Tâches de compilation

La liste de tâches ci-dessous est annotée avec des "niveaux" qui indiquent le niveau de difficulté des tâches, et aussi la contribution relative pour le calcul de la note. Vous devez choisir un ensemble de tâches qui totalisent au moins un niveau de 10. La note sera calculée sur 100. Un étudiant recevra 100% pour le projet si toutes les tâches qu'il a choisi sont réalisées correctement (même si les tâches totalisent plus que 10).

1. IFT3065:niv-2, IFT6232:niv-1 : Grammaire, parseur et messages d'erreur de syntaxe (pour les compilateurs conçus à partir de zéro)
2. IFT3065:niv-2, IFT6232:niv-1 : Compilation autogène (pour les compilateurs conçus à partir de zéro)
3. Langage source qui supporte
   - IFT3065:niv-2, IFT6232:niv-1 : les fermetures
     - IFT3065:niv-1, IFT6232:niv-1 : les continuations et call/cc
   - IFT6232:niv-3 : la programmation orientée objet avec héritage simple
   - IFT6232:niv-4 : la programmation orientée objet avec héritage multiple
4. Récupérateur mémoire automatique ("garbage collector")
   - IFT3065:niv-2, IFT6232:niv-1 : simple (mark-and-sweep, mark-and-compact, ou stop-and-copy)
   - IFT3065:niv-4, IFT6232:niv-3 : générationel
   - IFT6232:niv-4 : incrémental (concurrent ou temps-réel)
5. Analyses pour déterminer ou vérifier les propriétés suivantes
   - IFT3065:niv-1, IFT6232:niv-1 : types
   - IFT3065:niv-1, IFT6232:niv-1 : vivacité des variables
   - IFT3065:niv-1, IFT6232:niv-1 : expressions communes
   - IFT6232:niv-3 : ensemble de valeurs possibles pour les variables (0-CFA)
6. Transformations
   - IFT3065:niv-2, IFT6232:niv-1 : propagation de constante, propagation de copies, calculs constants
   - IFT3065:niv-2, IFT6232:niv-1 : élimination des expressions communes
   - IFT3065:niv-3, IFT6232:niv-2 : inlining de fonctions (visant la vitesse d'exécution)
   - IFT6232:niv-4 : évaluation partielle
7. Génération de code
   - IFT3065:niv-2, IFT6232:niv-1 : simple avec optimisation "peephole"
     - IFT3065:niv-1, IFT6232:niv-1 : allocation locale de registres
   - IFT6232:niv-3 : sélection d'instruction par programmation dynamique

5 Propositions

Vous devez me remettre **le 16 janvier** votre proposition de projet, incluant la composition des équipes, le langage choisi et si applicable, le compilateur que vous étendrez, les tâches que vous accomplirez et la répartition du travail sur les 3 phases du projet.