Université Laval
Faculté des sciences et de génie
Département d'informatique et de génie logiciel
GLO-7004

Danny Dubé
Automne 2010
Version : 29 septembre

Projet de session
Parties 1, 2 et 3

Partie 1—Définition du projet—10%

Tâche à effectuer

D'ici le **7 octobre**, vous devez choisir un projet, en préparer une description et faire approuver le choix du projet par le professeur. La description du projet doit faire au moins deux pages. Elle doit présenter le(s) document(s) décrivant les travaux originaux et elle doit donner une ébauche du travail d'adaptation, d'extension ou d'innovation que vous comptez réaliser. La description doit être suffisamment précise pour qu'il soit possible de juger de l'ampleur des travaux proposés. De plus, la description doit inclure la référence aux articles, livres, rapports ou autres que vous avez consultés pour définir votre projet. Bien entendu, deux étudiants ne pourront choisir le même projet : c'est premier arrivé, premier servi.

Histoire de situer le genre et l'ampleur des projets attendus, voici quelques suggestions de projets.

- Traduction de formules ou d'automates entre un formalisme et un autre. Par exemple, l'article de Svensson présente une traduction des formules de la *linear temporal logic* vers des automates de Büchi [7].
- Implantation de la technique de défonctionnalisation présentée par Mitchell et Runciman [5].
- Implantation d'un analyseur syntaxique incrémental fonctionnel, tel que présenté par Bernardy [1].
- Compilation de patrons de filtrage (*pattern matching*) sous la forme d'arbres de décision, selon Maranget [3].
- Implantation du vérificateur de suffisance pour des patrons non-exhaustifs présenté par Mitchell et Runciman [4].
- Implantation de l'algorithme de minimisation des patrons présenté par Krauss [2].
- Sélection d'un algorithme préexistant normalement implanté impérativement et développement d'une implantation purement fonctionnelle. Discussion sur les stratégies et les structures de données utilisées ainsi que sur la complexité en temps et en espace de l'implantation résultante. Voir le livre d'Okasaki [6].
- Sélection de l'implantation d'un langage fonctionnel disponible publiquement et ajout d'une ou de plusieurs facilités désirées par la communauté.
- Remplacement du ramasse-miettes (*garbage collector*) fourni dans l'implantation d'un langage fonctionnel pour un ramasse-miettes incrémentiel ou temps réel.

Pour tous les projets d'implantation, il convient de joindre une batterie de tests passés par votre implantation et démontrant que l'implantation effectue bien le travail désiré.

Partie 2—Rapport d'étape—10%

Tâche à effectuer

Au cours de la semaine du **8 novembre**, vous devez me rencontrer afin de me présenter l'état d'avancement de votre projet. Le rapport doit comporter au moins trois pages et expliquer les démarches (et possiblement l'atteinte de résultats) qui ont été faites à date. Le projet doit avoir avancé significativement. Le professeur pourra émettre des commentaires sur l'état d'avancement et sur l'orientation du projet, commentaires que l'étudiant devrait prendre en compte pour la suite de la réalisation du projet. La rencontre a lieu au cours de la période de disponibilité hebdomadaire ou sur rendez-vous.

Partie 3—Remise et présentation finale—80%

Tâche à effectuer

Au plus tard le **10 décembre**, vous devez remettre votre rapport final pour le projet. De plus, au cours de la semaine du **10 décembre** ou la semaine suivante, vous devez me rencontrer pour faire une démonstration ou une présentation du projet réalisé. Le rapport doit compter au moins une vingtaine de pages et présenter tous les éléments pertinents ; par exemple : détails d'implantation, la valeur pour la communauté d'utilisateurs ou de recherche, les caractéristiques de l'algorithme développé, etc.

Références

[1] Jean-Philippe Bernardy. Lazy functional incremental parsing. In *Proceedings of the ACM SIGPLAN Haskell Symposium*, pages 49–60, 2009.

[2] Alexander Krauss. Pattern minimization problems over recursive data types. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, pages 267–274, 2008.

[3] Luc Maranget. Compiling pattern matching to good decision trees. In *Proceedings of the ACM SIGPLAN Workshop on ML*, pages 35–46, 2008.

[4] Neil Mitchell and Colin Runciman. Not all patterns, but enough : An automatic verifier for partial but sufficient pattern matching. In *Proceedings of the ACM SIGPLAN Haskell Symposium*, pages 49–60, 2008.

[5] Neil Mitchell and Colin Runciman. Losing functions without gaining data – another look at defunctionalisation. In *Proceedings of the ACM SIGPLAN Haskell Symposium*, pages 13–24, 2009.

[6] Chris Okasaki. *Purely Functional Data Structures*. Cambridge University Press, 1998.

[7] Hans Svensson. Implementing an LTL-to-Büchi translator in Erlang. In *Proceedings of the ACM SIGPLAN Erlang Workshop*, pages 63–70, 2009.