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

Danny Dubé
Automne 2010
Version : 30 septembre

Présentation orale
Parties 1 et 2

Partie 1—Recherche d'articles

Tâche à effectuer

Vous devez choisir trois articles reliés aux sujets abordés dans le cours et me faire connaître votre liste afin de les réserver. Ceci constitue la première étape de la préparation de l'exposé oral. Par la suite, vous devrez lire ces trois articles et choisir celui sur lequel vous souhaitez faire votre présentation. Afin de bien vous préparer pour votre présentation, vous devrez très bien comprendre l'article choisi. Ceci peut demander d'aller lire certains des articles sur lesquels le vôtre se base.

Vous devez choisir les trois articles et les faire connaître au plus tard le **14 octobre** midi. Vous devez choisir vos trois articles parmi une liste d'articles récents. La liste d'articles se retrouve dans la bibliographie à la fin de cet énoncé. Notez que vous avez accès à ces articles en accédant à la bibliothèque numérique de l'ACM via `http://portal.acm.org`. L'Université Laval a un abonnement à la bibliothèque de l'ACM et vous pouvez obtenir les articles à partir des ordinateurs branchés sur le campus.

Cette étape ne devrait pas demander beaucoup de temps. Toutefois, j'insiste sur l'importance de cette "formalité" et j'attribue 5% de votre note sur la présentation. Notez que la liste est plutôt longue. Toutefois, il se peut que vous essayiez de choisir un article déjà choisi par un autre étudiant.[1] La règle du jeu est : premier arrivé, premier servi.

La présentation orale se fait individuellement.

La suite des directives est fournie dans la partie 2 de l'énoncé.

Partie 2—Préparation et présentation

Tâches à effectuer

1. Lecture des trois articles.
2. Sélection de l'article de présentation (un seul). Veuillez m'avertir de votre choix au plus tard le **1er novembre**.
3. Préparation d'un plan du déroulement de la présentation. Par exemple, une liste item–durée. Vous devez viser une présentation d'une vingtaine de minutes. Vous pouvez venir discuter de votre plan à mon bureau.
4. Montage des acétates. Montage du discours à faire.
5. Présentation devant la classe le **2 novembre**.

[1] Note : Seul le masculin est utilisé afin d'éviter d'allourdir le texte.

Ordre des présentations

Les présentations se dérouleront le **2 novembre**. On vise d'obtenir une durée de 25 minutes chacune : 20 pour la présentation magistrale et 5 pour la période de questions. Je tirerai au hasard l'ordre des présentateurs un peu plus tard dans la session. Le résultat du tirage sera indiqué sur la page web du cours.

Barème

5% Sélection de trois articles. (Partie 1)

10% Respect des règles. Ceci inclut le fait d'indiquer à la date prévue l'article que vous choisissez, de respecter la durée de 20 minutes pour l'exposé et d'assister à la présentation des autres étudiants de la classe.

15% Qualité des acétates : lisibilité, surcharge, pertinence.

15% Structure de la présentation : exposition du problème (nature, motivation), ébauche de solution, détails de la solution, résultats "expérimentaux" (ne s'applique pas à tous les articles), comparaison avec les travaux connexes (en autant que l'article en inclut une), conclusion.

20% Compréhensibilité, clarté de la présentation.

10% Performance d'orateur : force de la voix, qualité de la présence face à l'audience, débit du discours.

25% Compréhension du sujet par le présentateur : quant au contenu de l'article et des bases sur lesquels il repose, quant aux conséquences, à l'impact, au contexte, aux qualités et aux défauts de la recherche dans l'article.

Conseils

- Durant la présentation, veuillez ne pas lire les acétates. *Vous* devez être le centre d'intérêt. Pour diminuer la tentation de lire les acétates, un truc consiste à ne pas y mettre de phrases complètes ou très peu.
- Procédez de telle sorte que vous fassiez la présentation et que les acétates vous apportent de l'aide et non l'inverse.
- N'essayez pas de régler votre débit pour faire passer une quantité prédéterminée d'information. Réglez plutôt la quantité d'information en fonction de ce que vous êtes capable de passer en 20 minutes avec un débit correct et un niveau de compréhensibilité adéquat.
- Étirez ou raccourcissez chaque élément de la structure de la présentation selon l'importance ou la difficulté qui s'y rattache. Par exemple, un article présentant une solution archi-complexe à un problème obscur mérite qu'on s'attarde peut-être plus à la nature du problème plutôt qu'aux détails de la solution. Au contraire, un article présentant une solution à un problème déjà identifié en classe devrait voir sa solution mieux exposée au détriment de la motivation.
- Lorsque vous parlez à l'audience, vous devriez la regarder. Quand vous voulez attirer l'attention sur l'acétate courante, vous devriez regarder l'écran. Normalement, l'au-

dience a tendance à vous suivre et vous devez donc lui donner des “directives” par votre comportement.
– Vous pouvez vous pratiquer de vive voix chez vous. On constate souvent que la durée d'une explication n'est pas toujours celle qu'on a prévue.

Références

[1] Martin Abadi and Gordon Plotkin. A model of cooperative threads. In *POPL '09 : Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 29–40, New York, NY, USA, 2009. ACM.

[2] Lars Bergstrom, Mike Rainey, John Reppy, Adam Shaw, and Matthew Fluet. Lazy tree splitting. In *ICFP '10 : Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 93–104, New York, NY, USA, 2010. ACM.

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

[4] Rastislav Bodik, Satish Chandra, Joel Galenson, Doug Kimelman, Nicholas Tung, Shaon Barman, and Casey Rodarmor. Programming with angelic nondeterminism. In *POPL '10 : Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 339–352, New York, NY, USA, 2010. ACM.

[5] James Brotherston, Richard Bornat, and Cristiano Calcagno. Cyclic proofs of program termination in separation logic. *SIGPLAN Not.*, 43(1) :101–112, 2008.

[6] Benjamin Canou and Alexis Darrasse. Fast and sound random generation for automated testing and benchmarking in Objective Caml. In *ML '09 : Proceedings of the 2009 ACM SIGPLAN workshop on ML*, pages 61–70, New York, NY, USA, 2009. ACM.

[7] Koen Claessen, Michal Palka, Nicholas Smallbone, John Hughes, Hans Svensson, Thomas Arts, and Ulf Wiger. Finding race conditions in Erlang with QuickCheck and PULSE. In *ICFP '09 : Proceedings of the 14th ACM SIGPLAN international conference on Functional programming*, pages 149–160, New York, NY, USA, 2009. ACM.

[8] Nils Anders Danielsson. Total parser combinators. In *ICFP '10 : Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 285–296, New York, NY, USA, 2010. ACM.

[9] Olivier Danvy. Defunctionalized interpreters for programming languages. In *ICFP '08 : Proceeding of the 13th ACM SIGPLAN international conference on Functional programming*, pages 131–142, New York, NY, USA, 2008. ACM.

[10] Joshua Dunfield. Greedy bidirectional polymorphism. In *ML '09 : Proceedings of the 2009 ACM SIGPLAN workshop on ML*, pages 15–26, New York, NY, USA, 2009. ACM.

[11] Laura Effinger-Dean, Matthew Kehrt, and Dan Grossman. Transactional events for ML. *SIGPLAN Not.*, 43(9) :103–114, 2008.

[12] Sebastian Fischer, Frank Huch, and Thomas Wilke. A play on regular expressions : functional pearl. In *ICFP '10 : Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 357–368, New York, NY, USA, 2010. ACM.

[13] J. Nathan Foster, Alexandre Pilkiewicz, and Benjamin C. Pierce. Quotient lenses. *SIGPLAN Not.*, 43(9) :383–396, 2008.

[14] Ashutosh Gupta, Thomas A. Henzinger, Rupak Majumdar, Andrey Rybalchenko, and Ru-Gang Xu. Proving non-termination. In *POPL '08 : Proceedings of the 35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 147–158, New York, NY, USA, 2008. ACM.

[15] Chris Hawblitzel and Erez Petrank. Automated verification of practical garbage collectors. In *POPL '09 : Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 441–453, New York, NY, USA, 2009. ACM.

[16] Ralf Hinze. Functional pearl : la tour d'Hanoï. In *ICFP '09 : Proceedings of the 14th ACM SIGPLAN international conference on Functional programming*, pages 3–10, New York, NY, USA, 2009. ACM.

[17] Alexander Krauss. Pattern minimization problems over recursive data types. *SIGPLAN Not.*, 43(9) :267–274, 2008.

[18] Ruy Ley-Wild, Matthew Fluet, and Umut A. Acar. Compiling self-adjusting programs with continuations. *SIGPLAN Not.*, 43(9) :321–334, 2008.

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

[20] Moe Masuko and Kenichi Asai. Direct implementation of shift and reset in the MinCaml compiler. In *ML '09 : Proceedings of the 2009 ACM SIGPLAN workshop on ML*, pages 49–60, New York, NY, USA, 2009. ACM.

[21] Conor McBride. Clowns to the left of me, jokers to the right (pearl) : dissecting data structures. In *POPL '08 : Proceedings of the 35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 287–295, New York, NY, USA, 2008. ACM.

[22] Jan Midtgaard and Thomas P. Jensen. Control-flow analysis of function calls and returns by abstract interpretation. In *ICFP '09 : Proceedings of the 14th ACM SIGPLAN international conference on Functional programming*, pages 287–298, New York, NY, USA, 2009. ACM.

[23] Neil Mitchell. Rethinking supercompilation. In *ICFP '10 : Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 309–320, New York, NY, USA, 2010. ACM.

[24] Neil Mitchell and Colin Runciman. Not all patterns, but enough : an automatic verifier for partial but sufficient pattern matching. In *Haskell '08 : Proceedings of the first ACM SIGPLAN symposium on Haskell*, pages 49–60, New York, NY, USA, 2008. ACM.

[25] 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.

[26] Akimasa Morihata, Kiminori Matsuzaki, Zhenjiang Hu, and Masato Takeichi. The third homomorphism theorem on trees : downward & upward lead to divide-and-conquer. In *POPL '09 : Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 177–185, New York, NY, USA, 2009. ACM.

[27] Andreas Podelski and Thomas Wies. Counterexample-guided focus. In *POPL '10 : Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 249–260, New York, NY, USA, 2010. ACM.

[28] Colin Runciman, Matthew Naylor, and Fredrik Lindblad. Smallcheck and lazy smallcheck : automatic exhaustive testing for small values. In *Haskell '08 : Proceedings of the first ACM SIGPLAN symposium on Haskell*, pages 37–48, New York, NY, USA, 2008. ACM.

[29] Daniel Spoonhower, Guy E. Blelloch, Robert Harper, and Phillip B. Gibbons. Space profiling for parallel functional programs. *SIGPLAN Not.*, 43(9) :253–264, 2008.

[30] Jean-Baptiste Tristan and Xavier Leroy. Formal verification of translation validators : a case study on instruction scheduling optimizations. *SIGPLAN Not.*, 43(1) :17–27, 2008.

[31] David Van Horn and Matthew Might. Abstracting abstract machines. In *ICFP '10 : Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 51–62, New York, NY, USA, 2010. ACM.

[32] Marcos Viera, S. Doaitse Swierstra, and Eelco Lempsink. Haskell, do you read me ? : constructing and composing efficient top-down parsers at runtime. In *Haskell '08 : Proceedings of the first ACM SIGPLAN symposium on Haskell*, pages 63–74, New York, NY, USA, 2008. ACM.

[33] Marcos Viera, S. Doaitse Swierstra, and Wouter Swierstra. Attribute grammars fly first-class : how to do aspect oriented programming in Haskell. In *ICFP '09 : Proceedings of the 14th ACM SIGPLAN international conference on Functional programming*, pages 245–256, New York, NY, USA, 2009. ACM.

[34] Dimitrios Vytiniotis and Andrew J. Kennedy. Functional pearl : every bit counts. In *ICFP '10 : Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 15–26, New York, NY, USA, 2010. ACM.

[35] Dana N. Xu, Simon Peyton Jones, and Koen Claessen. Static contract checking for Haskell. In *POPL '09 : Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 41–52, New York, NY, USA, 2009. ACM.