Restructuration Interactive des Programmes
Projet de thèse

Cédric Bastoul & Stéphane Huot

École Doctorale : École Doctorale Informatique Paris-Sud

Laboratoire : LRI UMR8623 – Laboratoire de Recherche en Informatique, Univ. Paris-Sud & CNRS

Équipe(s) de recherche : Archi (`http://www.lri.fr/equipe.php?eq=2`) et InSitu (`http://insitu.lri.fr`)

Encadrement scientifique et direction de thèse : Cédric Bastoul – Archi (`cedric.bastoul@lri.fr`) et Stéphane Huot – InSitu (`huot@lri.fr`)

Titre de la thèse : *Restructuration Interactive des Programmes*

Mots clefs : Optimisation de programmes, Visualisation de code, Aide à la programmation, Techniques d'interaction

1 Résumé

Cette thèse s'inscrit dans la continuité des travaux de l'équipe Archi sur le développement de techniques semi-automatiques d'optimisation des programmes, l'objectif étant de concevoir et d'évaluer de nouvelles techniques de visualisation et d'interaction pour la restructuration des programmes. Il s'agit de définir et d'exploiter des représentations graphiques du modèle mathématique utilisé dans les théories et les outils développés par l'équipe afin d'assister les programmeurs dans l'optimisation de leurs programmes de manière sûre et efficace, sans qu'ils aient pour autant une expertise forte en restructuration de codes. Ce projet est une étape clef pour une utilisation efficace et une acceptation plus large des techniques semi-automatiques d'optimisation qui demandent encore beaucoup trop d'efforts de la part des utilisateurs. Plus généralement, ces travaux permettront aussi l'étude de nouvelles techniques de présentation visuelle et de manipulation interactive de code et de programmes, dont les applications peuvent être multiples : collaboration, suivi et vérification de modifications, recherche visuelle dans de grandes quantités de code, pédagogie, etc.

2 Sujet de la thèse

L'équipe Archi développe un ensemble d'outils pour l'optimisation et la parallélisation de codes. Afin d'exploiter au maximum les possibilités des architectures modernes, nous utilisons un modèle mathématique des programmes qui permet des restructurations importantes d'un code tout en préservant sa sémantique originale. L'utilisation directe de ce modèle est toutefois, par les connaissances qu'elle requiert et sa complexité, hors de portée du programmeur dont l'objectif final est d'optimiser ses applications et non de maîtriser un nouveau formalisme mathématique. Ainsi, nous avons pour le moment développé deux stratégies complémentaires pour l'utilisation de ces outils. La première est basée sur une approche entièrement automatique destiné à être directement intégrée dans les compilateurs. Cette approche est un succès en terme de gains de performance obtenus [10, 9, 3] et par son intégration en

cours dans différents compilateurs tels que GCC [8], LLVM [1] ou IBM XL [2]. Elle ne permet toutefois pas un contrôle fin de la restructuration du programme, ce qui peut s'avérer nécessaire quand les décisions prises par le compilateur ne sont pas les bonnes. La seconde stratégie est d'offrir au programmeur des extensions de haut niveau du langage de programmation, lui permettant de restructurer ses programmes en utilisant le modèle mathématique de manière transparente. Nous avons développé pour cela le framework URUK [5] (des alternatives existent telles UTF [7] ou CHiLL [4]) . Cette approche semi-automatique offre plus de contrôle et de possibilités de retour sur les optimisations réalisées, mais elle est laborieuse et nécessite une véritable expertise de la part du programmeur, ce qui en limite son intérêt malgré son efficacité.

Le modèle mathématique sous-jacent se prête pourtant à des représentations graphiques (dite *polyédriques*) qui n'ont jusqu'à maintenant jamais été mises à profit. Ces représentations permettraient de représenter les programmes graphiquement et d'effectuer des restructurations complexes *via* des opérations graphiques et des techniques d'interaction appropriées. De fait, une telle utilisation de ce modèle permettrait d'affranchir le programmeur d'une large part de l'expertise nécessaire à la restructuration de code, et d'utiliser toute la puissance de ces outils d'une manière inédite. Réciproquement, ces représentations graphiques pourraient aussi être utilisées pour visualiser de manière interactive les résultats d'optimisations automatiques, avec plusieurs objectifs qui sont entre autres : (i) comprendre les restructurations apportées aux programmes et donc le fonctionnement des technologies sous-jacentes ; (ii) modifier, contrôler et ajuster finement ces restructurations par rapport aux besoins ; (iii) explorer et comparer différentes restructurations, et donc les impacts différents qu'elles ont sur l'optimisation du programme.

Plus généralement, ces techniques pourraient être étendues à l'optimisation et la parallélisation interactive de simulations complexes (dans le cadre du projet Equipex Digiscope) et à la visualisation interactive d'algorithmes et de programmes, dans un but collaboratif ou pédagogique. Dans ce but, un soin particulier devra être apporté aux maintien de la cohérence entre les différentes représentations d'un même concept, ce qui nécessitera l'étude de techniques appropriées de mise en correspondance et de transition entre ces différentes représentations (i.e., le code et sa représentation graphique), voire même entre représentations d'un même niveau (e.g., deux restructuration différentes d'un même code). Cette partie des travaux pourra s'appuyer sur les travaux récents de l'équipe InSitu (en collaboration avec l'équipe Inria AVIZ) dans ce domaine [6].

L'objectif de cette thèse est donc d'étudier, de concevoir et d'évaluer de telles techniques de représentation et d'interaction pour la restructuration de programmes. Plus généralement, ces travaux seront les prémices d'une approche nouvelle et complémentaire aux recherches actuelles en programmation visuelle et visualisation d'algorithmes/programmes, en combinant ces deux approches.
Dans un premier temps, le doctorant devra prendre en main les différentes technologies et outils de compilation polyédrique développé par l'équipe Archi. Il devra ensuite procéder de l'approche générative de conception [1] développée dans l'équipe InSitu en (i) dressant une taxonomie des techniques de visualisation pertinentes pour le problème posé (aspect descriptif) à partir d'une revue de l'état de l'art ; (ii) définissant un modèle de visualisation et d'interaction permettant de comparer entre elles ces techniques et de faire des choix informés (aspect évaluatif) ; et (iii) proposant des principes de conception pour stimuler et faciliter la création de nouvelles techniques (aspect génératif).

Ces trois points seront illustrés par les nouvelles techniques conçues au cours de la thèse, qui seront validées par une approche empirique, avec la réalisation d'expérimentations contrôlées et d'études lon-

1. `http://polly.llvm.org/`

gitudinales, ainsi que d'éventuelles études préliminaires avec des utilisateurs potentiels de ce type de systèmes (conception participative).

3 Connaissances et compétences requises

Connaissances en Interaction Homme-Machine, programmation d'interfaces graphiques et maîtrise de bon niveau de Java et C/C++ indispensables.
Le candidat devra aussi être intéressé par le calcul haute performance et le parallélisme. Une expérience en compilation, architecture ou algèbre linéaire serait un plus.

Références

[1] M. Beaudouin-Lafon and W. E. Mackay. Reification, polymorphism and reuse : three principles for designing visual interfaces. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '00, pages 102–109, New York, NY, USA, 2000. ACM.

[2] U. Bondhugula, O. Günlük, S. Dash, and L. Renganarayanan. A model for fusion and code motion in an automatic parallelizing compiler. In *PACT*, pages 343–352, 2010.

[3] U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral parallelizer and locality optimizer. In *PLDI*, pages 101–113, 2008.

[4] C. Chen, J. Chame, and M. W. Hall. CHiLL : A framework for composing high-level loop transformations. Technical Report 08-897, University of Southern California, Jun 2008.

[5] A. Cohen, S. Girbal, and O. Temam. A polyhedral approach to ease the composition of program transformations. In *Euro-Par'04*, number 3149 in LNCS, pages 292–303, Pisa, Italy, Aug. 2004. Springer-Verlag.

[6] P. Dragicevic, S. Huot, and F. Chevalier. Gliimpse : Animating from markup code to rendered documents and vice versa. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 257–262, New York, NY, USA, 2011. ACM.

[7] W. Kelly. *Optimization within a Unified Transformation Framework*. PhD thesis, Univ. of Maryland, 1996.

[8] S. Pop, A. Cohen, C. Bastoul, S. Girbal, P. Jouvelot, G.-A. Silber, and N. Vasilache. GRAPHITE : Loop optimizations based on the polyhedral model for GCC. In *Proc. of the 4th GCC Developper's Summit*, pages 179–198, Ottawa, Canada, June 2006.

[9] L.-N. Pouchet, C. Bastoul, A. Cohen, and S. Cavazos. Iterative optimization in the polyhedral model : Part II, multidimensional time. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'08)*, Tucson, Arizona, June 2008. To appear.

[10] L.-N. Pouchet, U. Bondhugula, C. Bastoul, A. Cohen, J. Ramanujam, and P. Sadayappan. Combined iterative and model-driven optimization in and automatic parallelization framework. In *Conference on Supercomputing (SC'10)*, New Orleans, LA, Nov. 2010.