Titre du stage : Récriture compilée

Lieu du stage : LIX, École polytechnique

Encadrants : Frédéric Blanqui, Jean-Pierre Jouannaud

Sujet du stage : Actuellement, dans Coq, fonctions et prédicats sont définis comme dans OCaml, par cas sur les constructeurs d'un type inductif. Xavier Leroy et Benjamin Grégoire ont récemment adapté à Coq les techniques de compilation efficaces utilisées dans OCaml. Ce travail a été la base de plusieurs avancées importantes comme la géneération d'un certificat de primalité prouvée pour la mise en oeuvre de cryptosystèmes à clé publique. D'un autre côté, les thèses successives de Maribel Fernández, Frédéric Blanqui et Daria Walukiewicz ont montré la possibilité théorique d'intégrer des définitions par récriture dans Coq.

Il s'agit dans ce stage d'étudier les techniques de compilation utilisées en récriture, et de les intégrer à la compilation de la beta-réduction implémentée par Benjamin Grégoire dans Coq. Ce travail doit servir de base au développement d'une nouvelle version de Coq, et constitue la première étape d'une thèse sur le sujet, dont le but sera de considérer des notions de récriture plus complexes importantes en pratique ou d'un point de vue conceptuel (récriture associative-commutative et récriture d'ordre supérieur).

Bibliographie :

[Kirchner and Moreau] H. Kirchner and P.-E. Moreau. Promoting rewriting to a programming language : A compiler for non-deterministic rewrite programs in associative-commutative theories. *Journal of Functional Programming*, 11(2) :207–251, 2001. Voir `http://www.loria.fr/~moreau/`.

[Grégoire and Leroy] B. Grégoire and X. Leroy. A compiled implementation of strong reduction. In *Proceedings of the 7th ACM SIGPLAN International Conference on Functional Programming*, 2002. Voir `http://pauillac.inria.fr/~xleroy/`.

[Blanqui] F. Blanqui. *Théorie des Types et Récriture*. PhD thesis, Université Paris XI, Orsay, France, 2001. Voir `http://www.lix.polytechnique.fr/~blanqui/`.

Titre du stage : Contraintes en Coq

Lieu du stage : LIX, École polytechnique

Encadrants : Frédéric Blanqui, Jean-Pierre Jouannaud

Sujet du stage :

Le typage du calcul des constructions Coquand et Huet utilise une règle appelée conversion qui permet d'inclure des calculs dans le typage sans les faire figurer dans les termes preuves. Dans ce calcul, la relation de conversion est engendrée par la $beta$-reduction seule, mais elle peut être étendue par des règles de récriture adaptées. Par exemple, dans le calcul des constructions inductives de Coquand et Paulin-Mohring, la relation de conversion est engendrée par la $beta$-reduction ainsi que par les relations d'élimination (faibles ou fortes) associées aux types inductifs définis par l'utilisateur. Le calcul des constructions algébriques généralise la part du calcul définie par l'utilisateur en lui permettant d'ajouter des règles de récriture d'ordre supérieur.

Dans le logiciel PVS concurrent de Coq, la relation de conversion peut être engendrée par un mécanisme à la fois plus complexe, plus souple et plus naturel, hérité de la "congruence closure" de Shostak pour le premier ordre. Ce travail dû à Jean-Christophe Filliatre, Sam Owre et Natarajan Shankar, peut s'interpréter comme le tout premier apport des techniques de contraintes en théorie des types.

La règle de conversion du calcul des constructions a été étendue au cours d'un stage l'an dernier par une congruence engendrée par un nombre fini (et croissant) d'équations closes introduites par l'utilisateur dans l'environnement de preuve, en obéissant à un principe ancien mais complexe du a Shostak. Une implementation du calcul a été faite avec l'outil "Open Calculus of Constructions" réalisé dans le langage Maude par l'équipe de José Meseguer.

Dans ce stage, il s'agit de poursuivre ce travail à la fois sur le plan pratique, et sur le plan théorique. Sur le plan pratique, en implémentant de nouvelles théories Shostak. Sur le plan théorique, en montrant la décidabilité du typage dans ce cadre. Ce travail est la première étape d'une thèse dont le but est d'aboutir à une théorie et une implèmentation à ce jour manquantes d'un calcul des constructions avec contraintes.

Bibliographie :

[Shankar and Rueß] N. Shankar and H. Rueß. Deconstructing Shostak. In *Proceedings of the 16th IEEE Symposium on Logic in Computer Science*, 2001. Voir `http://www.csl.sri.com/users/shankar/shankar.html`.

[Filliâtre *et al*] J.-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. Deciding propositional combinations of equalities and inequalities. Unpublished. Voir `http://www.lri.fr/~filliatr/`.

[Blanqui] F. Blanqui. *Théorie des Types et Récriture*. PhD thesis, Université Paris XI, Orsay, France, 2001. Voir `http://www.lix.polytechnique.fr/~blanqui/`.

[strub] Pierre-Yves Strub. Intégration de procédures de décision en théorie de types. *Stage de DEA*.

Titre du stage : Théorème de Toyama

Lieu du stage : LIX, École polytechnique

Encadrant : Jean-Pierre Jouannaud

Sujet du stage :

Le théorème de Toyama énonce un principe de modularité de la confluence : l'union de deux systèmes de règles confluents disjoints (ne partageant pas de symboles de fonction) est confluente. Ce théorème a été généralisé dans différentes directions, qui consistent à autoriser un partage limité de symboles. La preuve du théorème de base, due à Toyama, a été améliorée par Barendregt, De Vrijer et Toyama, mais elle reste complexe.

Une preuve simple existe, non publiée, qu'il s'agit d'étendre de manière à couvrir les généralisations existantes du théorème. Cela peut demander la généralisation d'un principe de modularité de la complétion sans échec, sur lequel repose la preuve simple. Ce travail se prolonge dans une autre direction : le cas de la récriture d'ordre supérieur "à la Nipkow". Ce stage devrait donner lieu à une publication rapide.

Bibliographie :

[Jouannaud *et al*] N. Dershowitz, M. Fernández, and J.-P. Jouannaud. Modular confluence revisited, 1999. Unpublished. Voir `http://www.lri.fr/~jouannau/`.

[Klop *et al*] J. W. Klop, A. Middeldorp, Y. Toyama, and R. de Vrijer. Modularity of confluence : A simplified proof. *Information Processing Letters*, 49 :101–109, 1994.

[Toyama] Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1) :128–143, 1987.

Titre du stage : Preuves et complexité

Lieu du stage : LIX, École polytechnique

Encadrants : Jean-Pierre Jouannaud

Sujet du stage :

L'avènement des cartes à puces et des applications bancaires qui y résident a ouvert un grand champs d'application aux méthodes formelles : la preuve de logiciels embarqués. Outre les contraintes habituelles de correction du code, il s'avère important de pouvoir quantifier les besoins en ressource des logiciels embarqués, en particulier en temps et en espace.

Le but de ce stage consiste à développer des méthodes de calcul de la complexité (en temps et en espace) des programmes extraits par l'assistant Coq. Une expérience similaire a été menée en NuPRL, qui pourra donc constituer un bon point de départ. Un autre point de départ est un travail en cours qui vise à relier les deux problèmes d'extraction et de calcul de complexité du programme extrait via la réalisabilité de Kleene.

Bibliographie

[Benzinger] R. Benzinger. *Automated Computational Complexity Analysis*. PhD thesis, Cornell University, United States, 2001. Voir `http://techreports.library.cornell.edu:8081/Dienst/UI/1.0/Display/cul.cs/TR2002-1880`.

[Letouzey and Paulin] Voir `http://www.lri.fr/~letouzey/`.

[Jouannaud] Draft.

Titre :

Systèmes de transition dynamiques
Lieu du stage : LIX, École polytechnique
Encadrants : Jean-Pierre Jouannaud, Frédéric Blanqui

Sujet de stage : L'objectif est de vérifier des propriétés de sureté de systèmes de transitions dynamiques, spécifiées sous la forme d'états dits dangereux. Les méthodes classiques permettent d'appréhender des systèmes statiques par exploration systématique du graphe des états, une technique appelée "Model checking". Dans le système Fatalis développé en collaboration avec une équipe japonaise, la spécification d'un système de transition par l'utilisateur comprend :

1. la liste des agents (ou familles d'agents) qui le composent ;
2. pour chaque agent (ou famille d'agents), la liste des états qu'il peut prendre ;
3. la liste des règles de transition décrivant le comportement des agents ;
4. l'état initial du système et la liste des états dangereux.

Cette spécification est ensuite traduite automatiquement en logique linéaire, de telle sorte que tout comportement dangereux du système s'identifie avec une preuve linéaire que la (traduction de la) situation intiale implique (la traduction d') une situation dangereuse. Fatalis permet de rechercher une telle preuve (il s'agit d'un algorithme de recherche de chemin dans le graphe des états construit à la volée, car il peut être gigantesque) puis de s'assurer de sa correction (malgré de possibles erreurs de programmation ...) en la type-checkant dans le système Coq.

Il s'agit maintenant d'aborder le cas de systèmes dynamiques dans le but de modéliser des situations où les agents du système sont mobiles. Fatalis permet de les spécifier, et il faut donc étendre l'algorithme de recherche de chemin dans le graphe des états construit à la volée. Cela présente des difficultés, puisque l'écoulement du temps provoque l'apparition ou la disparition d'agents, et donc des modifications structurelles du graphe des états. La génération de la preuve linéaire à partir du chemin trouvé fait partie du stage.

Bibliographie

[Hasebe] Vicent Cremet, Koji Hasebe, Jean-Pierre Jouannaud, Antoine Kremer, Mitsuhiro Okada and Roland Zumkeller. Fatalis : Real Time Processes as Linear Logic Specifications. *International Workshop on Automated Verification of Infinite State Systems*, Varsaw, 2003.