Dedukti : un vérificateur de preuves universel

Ronan SAILLARD – MINES ParisTech – Centre de recherche en informatique

Les systèmes informatiques sont de plus en plus présents dans nos vies. On les retrouve dans des secteurs aussi divers que la communication (téléphonie, Internet), la sécurité (cryptographie), le transport (aéronautique, véhicules autonomes) ou encore le divertissement (jeux vidéos). Accompagnant cet essor, le domaine des méthodes formelles est lui aussi en pleine croissance. Il s'agit d'appliquer un ensemble de techniques visant à utiliser un raisonnement mathématique pour établir de façon rigoureuse certaines propriétés d'un programme informatique. Autrement dit, le but est de prouver l'absence de bug dans un programme. Cette croissance se traduit par une grande diversité de logiciels et langages dédiés aux preuves, rendant l'interopérabilité de ces systèmes souvent difficile.

Les méta-logiques (*Logical Frameworks*) constituent une solution naturelle au problème de l'interopérabilité. Leur intérêt est en effet de pouvoir exprimer et décrire dans un même langage différents formalismes logiques. Dans notre travail, nous proposons le λΠ-calcul modulo comme langage de preuve universel. Cette méta-logique est une extension du λΠ-calcul (λ-calcul avec types dépendants) avec des règles de réécriture arbitraires sur les termes et sur les types.

Dedukti est un vérificateur de preuves pour le λΠ-calcul modulo. Il implémente l'algorithme classique de vérification de type pour les systèmes de types purs semi-complets ainsi qu'une machine abstraite de réduction inspirée de celle de l'assistant de preuve Matita [1]. Cette dernière a été modifiée pour permettre la réduction par des règles de réécriture arbitraires, qu'elles soient linéaires ou non. Dedukti est écrit en Ocaml et est long d'un millier de ligne de code environ. Le programme est disponible à l'adresse suivante :

https://www.rocq.inria.fr/deducteam/Dedukti/

De manière à pouvoir évaluer le gain de performance à imputer à l'utilisation des règles de réécriture, nous avons effectué plusieurs tests à partir de deux bibliothèques de théorèmes différentes.

Le premier test concerne la bibliothèque de OpenTheory [2], un projet permettant l'échange de théorèmes entre les différents assistants de preuve de la famille HOL. L'outil Holide [3] permet de traduire ces théorèmes et leurs preuves dans le format de Dedukti.

Le deuxième test concerne la bibliothèque TPTP [4] qui constitue le jeu de test standard des prouveurs automatiques et les preuves obtenues grâce au prouveur Zenon [5].

Dans les deux cas, nous comparons deux traductions différentes : l'une dans le λΠ-calcul (c'est-à-dire sans règle de réécriture) et l'autre dans le λΠ-calcul modulo.

	λΠ-calcul	λΠ-calcul modulo
Opentheory	395 sec.	17 sec.
TPTP/Zenon	1764 sec.	18 sec.

Fig. 1 Temps de vérification par Dedukti de différentes bibliothèques de théorèmes.

Les résultats sont présentés Figure 1. On peut voir que la traduction utilisant les règles de réécriture est 23 fois plus rapide à vérifier que la traduction sans réécriture pour OpenTheory et 100 fois plus rapide pour TPTP/Zenon. Deux raisons principales permettent d'expliquer ces résultats. D'abord l'utilisation des règles de réécriture permet une traduction plus concise, conduisant à des fichiers plus petits. Ensuite leur vérification fait proportionnellement moins appel au test de conversion et ces tests sont plus simples. Or les tests de conversion constituent généralement la partie la plus coûteuse de la vérification de preuve.

Pour devenir un langage de preuve universel, il est important que la vérification des preuves dans le λΠ-calcul modulo soit efficace. Nous avons montré, avec Dedukti, que cette vérification est accélérée grâce à l'utilisation des règles de réécriture.

Références

[1] A. Asperti, W. Ricciotti, C. Sacerdoti Coen, E. Tassi, A Compact Kernel for the Calculus of Inductive Constructions, Sadhana, 2009.
[2] J. Hurd, The OpenTheory Standard Theory Library, NASA Formal Methods, 2011.
[3] A. Assaf, G. Burel, Holide, https://www.rocq.inria.fr/deducteam/Holide/index.html
[4] G. Sutcliffe, The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0, Journal of Automated Reasoning, 2009.
[5] R. Bonichon, D. Delahaye and D. Doligez, Zenon: An Extensible Automated Theorem Prover Producing Checkable Proofs, LPAR 2007.