Dedukti : un vérificateur de preuves universel

Ronan Saillard

Dedukti : un vérificateur de preuves universel

Ronan Saillard

Centre de recherche en informatique - MINES ParisTech

ronan.saillard@mines-paristech.fr

Contexte et objectif

- **Dedukti** est un vérificateur de type pour $\lambda\Pi$-calcul modulo.
- Le **$\lambda\Pi$-calcul modulo** est une extension du λ-calcul avec des **types dépendants** et une relation de conversion étendue par des **règles de réécriture**.
- Dedukti est capable de vérifier les preuves provenant de **prouveurs de théorèmes** (Zenon, iProver) et d'**assistants de preuves** (Coq, HOL, Focalize).
- On montre à travers plusieurs exemples que l'utilisation de la réécriture caractéristique du $\lambda\Pi$-calcul modulo permet d'obtenir des **preuves plus petites** et **plus rapides** à vérifier.

Traducteurs

Pour vérifier une preuve avec **Dedukti**, il faut d'abord l'exprimer dans le **$\lambda\Pi$-calcul modulo**. Pour cela, on utilise des **traducteurs** spécialisés :

- **Holide** traduit les preuves de l'assistant de preuve **HOL Light**.
- **Coqine** traduit les preuves de l'assistant de preuve **Coq**.
- **Zenonide** traduit les preuves du prouveur de premier ordre **Zenon**.
- **Focalide** traduit les programmes certifiés de **FoCaLiZe**.
- Une extension du prouveur **iProver** permet de traduire ses preuves.

Ces logiciels sont développés au sein de l'équipe **Deducteam** de **INRIA** par Ali Assaf, Guillaume Burel, Raphaël Cauderlier, Frédéric Gilbert et Pierre Halmagrand.

Règles de typage

$$\textbf{(Empty)}\ \frac{}{\emptyset\ wf} \qquad \textbf{(Dec)}\ \frac{\Gamma \vdash A : s \quad x \notin \Gamma}{\Gamma(x : A)\ wf}$$

$$\textbf{(Rw)}\ \frac{\Gamma\Delta \vdash l : T \quad \Gamma\Delta \vdash r : T \quad FV(r) \cap \Delta \subset FV(l)}{\Gamma([\Delta]l \hookrightarrow r)\ wf}$$

$$\textbf{(Type)}\ \frac{\Gamma\ wf}{\Gamma \vdash \textbf{Type} : \textbf{Kind}} \qquad \textbf{(Var/Cst)}\ \frac{\Gamma\ wf \quad (x : A) \in \Gamma}{\Gamma \vdash x : A}$$

$$\textbf{(Abs)}\ \frac{\Gamma \vdash A : \textbf{Type} \quad \Gamma(x : A) \vdash t : B \quad B \neq \textbf{Kind}}{\Gamma \vdash \lambda x^A.t : \Pi x^A.B}$$

$$\textbf{(Prod)}\ \frac{\Gamma \vdash A : \textbf{Type} \quad \Gamma(x : A) \vdash B : s}{\Gamma \vdash \Pi x^A.B : s}$$

$$\textbf{(App)}\ \frac{\Gamma \vdash t : \Pi x^A.B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B[x \backslash u]} \qquad \textbf{(Conv)}\ \frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s \quad A \equiv_{\beta\Gamma} B}{\Gamma \vdash t : B}$$

Règles de typage du $\lambda\Pi$-calcul modulo.

Mise en œuvre

- Un millier de lignes de code **OCaml** pour le vérificateur de types du $\lambda\Pi$-calcul modulo.
- Une machine de réduction **call-by-need** pour la **β-réduction** et les **règles de réécriture** ajoutées.
- La compilation des **règles de réécriture** (possiblement non linéaires) en **arbres de décision** pour une réécriture efficace.
- Une **licence libre** : CeCILL-B.

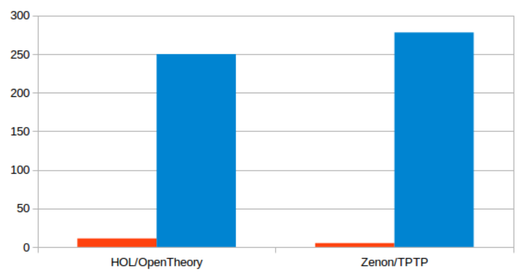
Expérience

On compare, sur deux jeux de tests différents, des encodages avec règles de réécriture et sans règle de réécriture.

- Le premier test concerne 86 fichiers de la bibliothèque de preuves et théorèmes **OpenTheory** tels qu'encodés par **Holide**.
- Le deuxième test concerne 520 fichiers correspondant à des preuves de théorèmes de la bibliothèque **TPTP**, obtenus grâce au prouveur **Zenon**. La bibliotèque TPTP est le jeu de test standard des prouveurs automatiques.

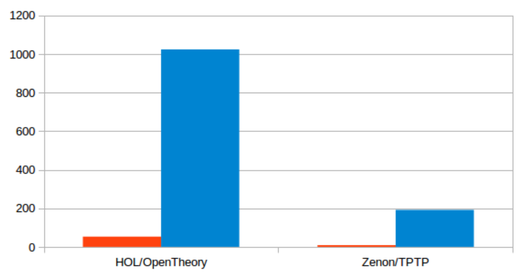
Les tests ont été réalisés sur un ordinateur Linux muni d'un processeur Intel Core i7-3520M CPU @ 2.90GHz x 4 et de 16GB de Ram.

Résultats



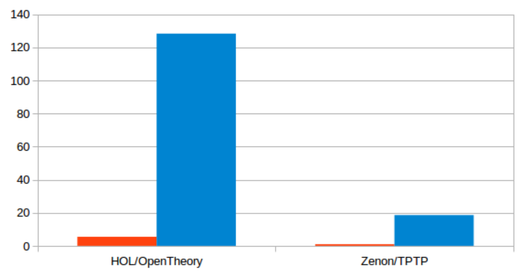
Temps de vérification (en secondes)

La vérification des preuves encodées en utilisant les règles de réécriture est **23 fois plus rapide** pour OpenTheory et est **55 fois plus rapide** pour TPTP.



Taille des fichiers (en mégaoctets)

La taille des fichier obtenus après encodage est environ **20 fois inférieure** lorsqu'on utilise des règles de réécriture.



Nombre de tests de conversion (en millions)

Le nombre de tests de conversion effectués lors de la vérification des preuves de **OpenTheory** est **23 fois inférieur** pour l'encodage avec règles de réécriture et **18 fois inférieur** pour TPTP.

Légende : avec réécriture - sans réécriture

Interprétation : Un encodage avec réécriture est nettement plus rapide a vérifier. Cela s'explique, d'abord par la taille réduite des termes obtenus par ces encodage, et ensuite par le nombre moins important de tests de conversion nécessaires à leur vérification.

Vers l'interopérabilité

- Les systèmes de preuves actuels souffrent d'un manque d'**interopérabilité**. Il est difficile de réutiliser une théorie d'un système dans un autre sans refaire toutes les preuves.
- La traduction de différents systèmes dans un formalisme commun et facilement vérifiable permettra de **combiner** leurs preuves pour construire des théories plus larges.

Référence

R. Saillard, Towards explicit rewrite rules in the lambda-Pi calculus modulo, 10th International Workshop on the Implementation of Logics (IWIL), Stellenbosch, 2013.

Dedukti et les différents **traducteurs** sont disponibles à l'adresse suivante : `https://www.rocq.inria.fr/deducteam/software.html`