Projet Hi-Lite
Livrable 5.3

Rapport sur les améliorations apportées aux outils logiciels Why et Alt-Ergo

---

Claude Marché[1,2] Sylvain Conchon[2,1] Evelyne Contejean[2,1] Denis Cousineau[1,2]
Jean-Christophe Filliâtre[2,1] Mohamed Iguernelala[2,1] Alain Mebsout[2,1]
Andrei Paskevich[2,1]

23 mai 2013

Résumé

Ce rapport décrit les améliorations apportées aux outils Why et Alt-Ergo au cours du projet Hi-Lite.

[1] INRIA Saclay - Île-de-France, ProVal, Orsay, F-91893
[2] LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

Table des matières

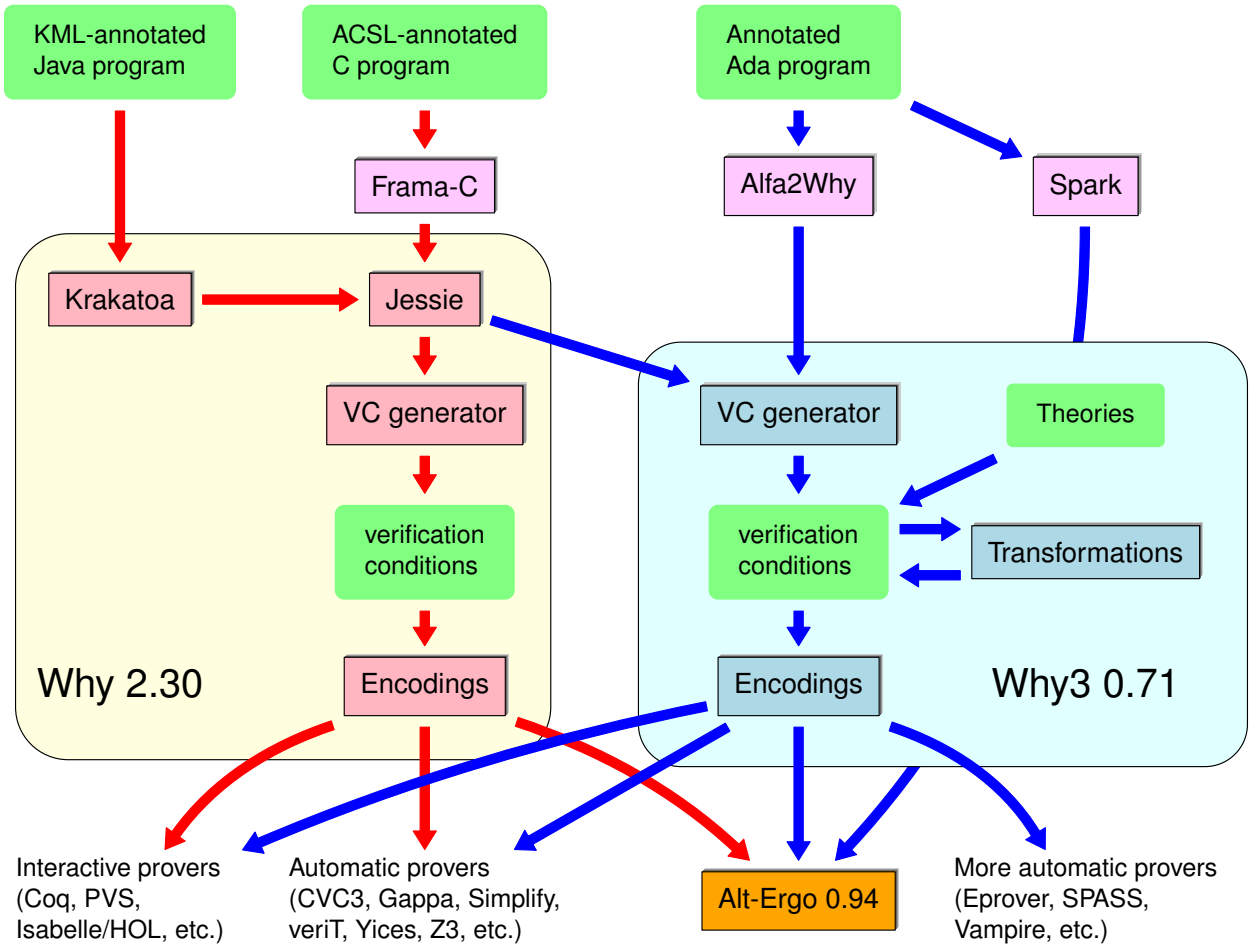

FIGURE 1 – Chaînes de Vérification avant et dans Hi-Lite

1 Introduction

Ce document résume les évolutions apportées aux outils Why et Alt-Ergo au cours du projet FUI Hi-Lite.

Ces outils étaient utilisés auparavant au sein de chaînes de vérification pour les langages C et Java. Au cours de Hi-Lite il s'est agit de construire une chaîne similaire pour les programmes Ada. Au début du projet, nous avons décidé de construire une nouvelle version majeure de Why, nommée Why3, dont les nouveautés seront détaillées dans la section 2.

L'outil GNATProve développé par Adacore traduit des programmes Ada, plus précisément des programmes dans le fragment Alfa défini dans Hi-Lite, vers des programmes Why3, qui génère des obligations de preuves qui peuvent être déchargées par des prouveurs comme Alt-Ergo. Par ailleurs, au cours de Hi-Lite, une interface entre le vérificateur SparkAda de Altran-Praxis vers Alt-Ergo a été réalisée, pour permettre d'utiliser Alt-Ergo comme prouveur alternatif au prouveur Victor interne à SparkAda.

Ces différentes chaînes de vérification sont schématisées sur la figure 1. La partie droite de la figure, avec les flèches en bleu, représente les nouveautés du projet Hi-Lite.

2 Évolutions de l'environnement Why

2.1 Création et vie de Why3

La toute première release de Why3 (0.64) est parue en février 2011, et est donc une réimplantation complète de l'ancien outil Why. L'accent a été mis en premier lieu sur les extensions du langage de spécification logique : ont été en particulier réalisés les extensions promises dans la tâche 5.3 pour le support des types énumérés et inductifs. En second lieu, un effort particulier à été fait pour permettre de soumettre les buts de preuve vers un maximum de prouveurs automatiques différents. Le schéma de traduction des types énumérés et inductifs, utilisé quand ceux-ci ne sont pas connus des prouveurs externes, sont décrits dans un rapport interne Inria [43].

Une nouvelle réimplantation du langage des programmes et du générateur d'obligations de preuve a été faite, avec des fonctionnalités en particulier adaptées à l'utilisation comme cible du traducteur depuis ALFA, en collaboration avec le partenaire Adacore.

Au cours du projet, plusieurs releases de Why3 ont été produites : release 0.71 en novembre 2011 [9], 0.72 [4] en mai 2012, 0.73 [5] en juillet 2012, 0.80 [6] en octobre 2012 et 0.81 [8] en mars 2013. Why3 est distribué depuis la page Web `http://why3.lri.fr/`.

2.2 Améliorations au niveau de la logique

Une première version du nouveau langage de la logique de Why3 a été présenté au workshop sur les langages intermédiaires en 2011 [10]. L'accent a été mis sur le mécanisme de structuration des spécifications à l'aide de *théories*.

Les mécanismes de traduction du polymorphisme de type ont été rendus plus efficaces [11, 2]. La librarie standard de Why3 s'est enrichie par de nouvelles théories pour les collections de données : après les conteneurs proches de Ada [24], des théories pour les ensembles et les multi-ensembles ont été construites. Celles-ci ont été validées en particulier par un travail de traduction des obligations de preuves issues de l'Atelier B [41].

2.3 Améliorations au niveau des programmes

De multiples améliorations ont été réalisées au cours de Hi-Lite concernant le langage de programmation, la génération d'obligations de preuves, et les front-ends du langage (C, Java, Ada, etc.). Le langage des programmes WhyML a subi des évolutions en fonction des besoins du traducteur de Alfa vers Why3 développé par Adacore. En parallèle, le plug-in Jessie de Frama-C, qui traduit des programmes C annotés, a été adapté pour produire maintenant des programmes Why3 (release de Jessie en décembre 2011 [42]) au lieu de l'ancienne version de Why. Dans ce contexte, deux thèses ont été soutenues concernant les preuves de programmes avec pointeurs [1, 2].

En 2012, le langage intermédiaire WhyML a été modifié en premier lieu dans sa syntaxe et sa sémantique, mais surtout ces programmes peuvent être maintenant construits programmatiquement en OCaml grâce à l'existence d'une API de programmes et de modules WhyML (release 0.80 [6]). Le clonage de module (mécanisme d'instanciation de modules WhyML, similaire à des functors d'OCaml) a été rendu possible, à la demande d'Adacore. Ce nouveau WhyML a fait l'objet d'un article de présentation d'outil à la conférence internationale ESOP en mars 2013 [34]. La release 0.81 apporte des améliorations supplémentaires diverses, en particulier de performance et de support des prouveurs.

D'autre part, une variante du calcul de plus faible précondition a été ajoutée à Why3 (en grande partie grâce à des contributions d'Adacore), ce calcul appelé "fastwp" étant conçu pour limiter les possibles explosions combinatoires de la génération des obligations de preuve. Cette variante reste encore à finaliser.

2.4 Sessions de Preuves

En liaison avec la tâche 6 du projet Hi-Lite, un système de *sessions de preuve* a été réalisé. Depuis la release 0.72 de Why3, ce système de sessions est accessible via une API, pour faciliter l'intégration de Why3 avec les front-ends. La gestion de ces sessions et en particulier la détection fine des modifications du programme source (aussi bien au niveau du code que des spécifications) est présentée dans un article à la conférence VSTTE en mai 2013 [7].

3 Evolutions du prouveur Alt-Ergo

3.1 Vie d'Alt-Ergo

Plusieurs releases d'Alt-Ergo ont été faite durant le projet Hi-Lite : 0.93 en avril 2011, 0.94 en décembre 2011, et enfin 0.95 en janvier 2013 puis 0.95.1 en mars 2013. Alt-Ergo est distribué depuis la page Web http://alt-ergo.lri.fr/.

3.2 Mécanismes d'explications et de traces

Dès la release 0.93, nous avons ajouté à Alt-Ergo un mécanisme d'explications qui donne les hypothèses et les lemmes utilisés pour prouver la validité d'une formule. Avec cette nouvelle version vient également un front-end graphique, AltGr-Ergo, qui permet de visualiser facilement ces explications. AltGr-Ergo propose également des fonctionnalités interactives permettant à l'utilisateur d'aider le démonstrateur à prouver une formule (ajout d'instances de lemmes, nettoyage du contexte, sélection d'hypothèses etc.).

Dans la version 0.94, l'interface graphique AltGr-Ergo a été étendue avec de nouvelles fonctionnalités : sauvegarde et rejeu des interactions effectuées dans l'interface, affichage en temps réel d'informations sur l'activité du démonstrateur (temps passé dans les différentes procédures de décision, nombre d'instances de chaque lemme, etc.), affichage d'avertissements divers, et aide à la preuve (instances manuelles de lemmes, limitation des instances, modifications interactives du contexte). Les versions 0.95.x contiennent également un certain nombre d'améliorations de l'interface graphique AltGr-Ergo. En particulier, on peut maintenant connaître en temps réel le nombre d'instances pour chaque lemme et fixer une limite à ce nombre. L'interface graphique permet également d'afficher les modèles pour les termes annotés.

Dans les versions 0.95.x, nous avons implanté un nouveau système de production de contre-exemples, pour les cas où la preuve d'un but échoue. Ce mécanisme permet d'extraire des valeurs modulo théories pour des termes annotés dans le fichier initial. Ces annotations du but initial doivent être placées de façon à indiquer les variables et les propositions pertinentes que l'on souhaite voir dans le contre-exemple produit.

3.3 Nouvelles procédures de décision

Dans la version 0.93, nous avons ajouté une procédure de décision pour la théorie des types énumérés. Par ailleurs, la théorie des tableaux a été étendue pour traiter des tableaux polymorphes dans les types des éléments et des indices. Un nouveau prédicat distinct(a,b,c,..) permet d'indiquer qu'un ensemble de

constantes sont distinctes deux à deux. Enfin, la théorie de l'arithmétique dispose maintenant d'un support partiel pour la division euclidienne.

Les nouveautés de la version 0.94 concernant les théories supportées sont : théorie de l'associativité-commutativité (publiés dans [16, 17, 18], la publications TACAS étant récompensée par le prix EATCS 2011 du meilleur papier d'informatique théorique), théorie des types enregistrements, et amélioration de l'efficacité de la théorie de l'arithmétique (formalisation et preuve de correction publiées à la conférence IJCAR [3]). Ces trois ajouts sont utilisés par le back-end de Why3 pour Alt-Ergo. Un travail en cours consiste à définir des procédures de décision génériques via une théorie des triggers [22, 23]. Nous avons travaillé également sur l'intégration d'une théorie des flottants dans Alt-Ergo [20].

La version 0.95.1 propose une nouvelle méthode pour la combinaison des solveurs d'égalités des différentes théories. Par ailleurs, une extension du langage d'entrée qui permet à l'utilisateur de forcer les types des termes et ainsi de gérer plus finement les variables dans les types polymorphes.

3.4 Qualification d'Alt-Ergo

Des documents en vue de la qualification DO-178 d'Alt-Ergo par Airbus Industrie ont été produits (spécification formelle, et description de 500 jeux de tests pertinents permettant de montrer qu'Alt-Ergo respecte bien ces spécifications), puis validés par Airbus après une visite sur site.

3.5 Évaluation

Nous pouvons mesurer les progrès réalisés sur Alt-Ergo tout au long du projet Hi-Lite à travers les chiffres donnés dans le tableau suivant. Il s'agit des résultats des différentes versions du démonstrateur, depuis la version 0.9 (juillet 2009) jusqu'à la dernière version 0.95.1 (mars 2013), sur des obligations de preuves extraites de la galerie de programmes de la plate-forme Why3 disponible à l'adresse `http://toccata.lri.fr/gallery/why3.en.html`.

Il s'agit de 1920 obligations de preuve valides. Les résultats sont obtenus sur une machine quadri-coeur Intel Xeon à 3.2GHz. La limite de temps donnée au démonstrateur est de 30 secondes et la mémoire est limitée à 2 GB. La signification des lignes est la suivante :

- "valid" : obligations prouvées (v)
- "time" : temps total (en secondes) pour démontrer les (v) obligations
- "unknown" : obligations non prouvées, sans faire timeout
- "timeout" : timeout
- "errors" : erreur (débordement mémoire, etc.)

version	0.95.1	0.94	0.93	0.92.2	0.91	0.9
valid	1841	1811	1773	1737	1685	1306
time	362	465	411	527	555	290
unknown	20	25	24	22	21	233
timeout	59	83	99	128	175	328
errors	0	1	24	33	39	53

Le tableau suivant compare ces résultats avec les démonstrateurs automatiques concurrents d'Alt-Ergo.

	ALT-ERGO 0.95.1	CVC3 2.4.1	YICES 1.0.38	Z3 3.2	Z3 4.2
valid	1841	1767	515	1592	1524
time	362	357	103	446	256
unknown	20	30	1029	3	1
timeout	59	107	164	295	305
errors	0	16	212	30	90

4 Vers la certification des résultats

Avec l'utilisation croissante de nos outils Why3 et Alt-Ergo dans des contextes industriels sur la vérification de logiciels critiques, nous sommes amenés à nous poser la question de la confiance que l'on peut accorder aux outils eux-mêmes. En effet, un bug dans Why3 ou dans Alt-Ergo pourrait faire croire qu'un programme analysé respecte bien ses spécifications alors que ce n'est pas le cas. Ce n'est pas seulement une hypothèse de travail : nous avons eu par le passé des soucis de ce genre, par exemple des problèmes d'incohérence entre outils concernant le comportement d'une opération aussi simple que la division entière.

Nous avons ainsi mené plusieurs actions visant à vérifier les résultats des outils, voire prouver les outils eux-mêmes.

L'outil Why3 s'est doté de fonctionnalités dans ce but. La technique de "réalisation" permet de construire un modèle en Coq de chaque théorie Why3. Ceci a été mis en œuvre pour la théorie des ensembles [41]. Par ailleurs, nous avons montré comment on pouvait développer des générateurs d'obligations de preuves de façon sûre : calcul de WP pour le langage C certifié en Coq [37, 36], calcul de WP certifié en Why3 [39, 40].

Nous avons poursuivi des travaux de certification formelle autour d'Alt-Ergo, dans le but d'augmenter la confiance dans les preuves faites, dans la lignée de la thèse de Lescuyer [38]. Les 12 mois de post-doc financés par Hi-Lite ont été l'occasion d'investiguer la certification a posteriori des résultats d'Alt-Ergo. Denis Cousineau s'est d'abord intéressé à la certification des résultats Alt-Ergo dans Coq, en implémentant un système de traces légères de preuves pour Alt-ergo, permettant d'utiliser la tactique interne à Coq (ergo, de S. Lescuyer en 2011) pour vérifier les résultats d'Alt-Ergo à l'aide de ces traces. La tactique ergo ne gère pas certaines des fonctionnalités d'Alt-Ergo, comme les formules avec quantificateurs, le polymorphisme (et l'inférence de types), et les inégalités arithmétiques. L'implémentation actuelle du système de preuves d'Alt-Ergo permet d'expliciter quelles sont les formules universellement quantifiées qui ont été instanciées par Alt-ergo lors de sa recherche de preuves, ce qui permet de ne fournir à la tactique ergo que les formules instanciées (conjointement à leurs preuves Coq générées automatiquement). De même, le système de preuves explicite les types implicites inférés par Alt-ergo afin de ne fournir à la tactique ergo que des formules monomorphes (et leurs preuves Coq). Dans un second temps, Denis a réalisé une deuxième version de ce système de production de trace par Alt-Ergo, qui s'appuie en particulier sur la production de modèles décrite dans la section précdédente. Cette technique dissocie les différentes théories utilisées lors de la recherche d'une preuve par Alt-Ergo, afin d'utiliser les différentes procédures de décision Coq existantes pour chacune de ces théories (en particulier la tactique omega, en ce qui concerne les inégalités arithmétiques). Cette approche a ainsi pu montrer qu'il était faisable de revérifier a posteriori les preuves faites par Alt-Ergo en utilisant le système Coq, ce qui augmente fortement la confiance que l'on peut avoir dans les résultats. La perspective de ce travail est ainsi de proposer, lorsqu'un programme a été analysé et prouvé, de faire une deuxième vérification a posteriori apportant plus de garantie. Une telle vérification prendrait un temps non négligeable, mais ce n'est pas génant car c'est un processus entièrement automatique.

5 Dissémination et impact

De nombreuses actions de dissémination ont été faites autour de Why3 et Alt-Ergo.

Exposés invités exposés invités à la conférence majeure sur la déduction automatique CADE [33], aux workshops PLMW [31] et AIPA [27], ainsi qu'au prochain workshop SMT [15].

Études de cas Des études de cas ont été publiées [30, 44] dont des exemples avec calculs en virgule flottante [12]. De nombreux exemples de complexités variées sont disponibles sur la galerie de programmes prouvés de ProVal (http ://toccata.lri.fr/gallery/index.en.html).

Compétitions de vérification Nous nous sommes également impliqués dans les nouvelles "compétitions" de preuve de programmes, aussi bien en participant : VerifyThis 2011 [13] et VerifyThis@FM2012 (Why3 récompensé comme "tool used by most teams" ex-aequo avec Dafny) ; que en tant qu'organisateurs (Competition VSTTE 2012, https://sites.google.com/site/vstte2012/compet,[35]).

Écoles de jeunes chercheurs Nous avons aussi fait des séances de formation à Why3 : aux écoles de jeunes chercheurs EJCP [29] et DigiCosme [32], et également lors du workshop Boogie [28].

Thèses Jean-Christophe Filliâtre a soutenu son habilitation à diriger des recherches en 2011 [25], son mémoire résumant la démarche scientifique sous-tendant le développement de Why3. Sylvain Conchon a soutenu son habilitation en 2012 [14], son mémoire résumant la démarche scientifique sous-tendant le développement de Alt-Ergo. Nos deux doctorants membres de Hi-Lite soutiendront leur thèse de doctorat dans un avenir proche : la soutenance de Mohamed Iguernelala est prévue pour le 10 juin 2013 (Strengthening the Heart of an SMT-Solver : Design and Implementation of Efficient Decision Procedures), et Alain Mebsout terminera sa thèse en 2013-2014.

6 Conclusions et perspectives

6.1 Auto-évaluation

Les outils Why3 et Alt-Ergo ont beaucoup évolué au cours du projet Hi-Lite, et ont permis d'améliorer significativement les performances des chaînes de vérification qui les utilisent.

Parmi les travaux promis dans la proposition de projet, le seul point qui n'a pas réalisé est la conception d'une procédure de décision pour les types algébriques dans Alt-Ergo. Nous avons par contre implanté une procédure de décision pour les types énumérés, et une procédure de décision pour les records, tous deux étant des cas particuliers des types algébriques. Ce n'est pas un manque important car ce sont les seuls sous-cas utiles pour un front-end comme GNATProve.

6.2 Perspectives après le projet Hi-Lite

Les projets logiciel Why3 et Alt-Ergo vont naturellement se poursuivre après Hi-Lite.

Trois versions expérimentales d'Alt-Ergo sont actuellement en cours de développement. La première intègre un SAT solveur efficace couplé à une nouvelle procédure de décision pour l'arithmétique linéaire sur les entiers [3]. La deuxième implémente une procédure de décision pour la théorie des flottants. Elle est basée sur un mécanisme qui combine un traitement axiomatique de cette théorie avec une procédure de décision efficace pour l'arithmétique linéaire [20]. La troisième version est une extension d'Alt-Ergo qui permet à l'utilisateur d'ajouter des procédures de décision par l'intermédiaire d'un nouveau mécanisme de traitement des formules avec triggers [22, 23] (dans le cadre d'une thèse Cifre avec Adacore).

Nous avons également développé une nouvelle bibliothèque OCaml pour Alt-Ergo, appelée Alt-Ergo-Zero. Le code est disponible à l'adresse suivante : http ://cubicle.lri.fr/alt-ergo-zero/. Cette bibliothèque ne permet de traiter que des formules sans quantificateurs. Elle est basée sur le même mécanisme de combinaison de procédures de décision et elle intègre un nouveau SAT solveur efficace inspiré de minisat. Alt-Ergo-Zero est utilisée dans le model-checker Cubicle [19].

Le projet ANR BWare (http ://bware.lri.fr) vise à améliorer l'automatisation des preuves des obligations issues de l'Atelier B. Dans ce contexte nous continuons à améliorer les performances de Why3 et Alt-Ergo, et ces améliorations seront bénéfiques pour toutes les applications de Why3, donc pour GNATProve et pour les plug-in de vérification déductive de Frama-C (Jessie et WP).

La question du niveau de confiance que l'on peut avoir dans les résultats des preuves continuera à être examinée. Le projet BWare contient aussi un volet spécifique à la certification des résultats obtenus. Nous participons à un autre projet ANR, Verasco (http ://verasco.imag.fr), qui se destine spécifiquement à la production d'outils certifiés par preuve formelle. Enfin, la question de la confiance dans les outils eux-mêmes est un axe qui a été mis en avant pour la création de l'équipe-projet Inria "Toccata" qui fait suite à Proval.

Références

[1] R. Bardou. *Verification of Pointer Programs Using Regions and Permissions*. Thèse de doctorat, Université Paris-Sud, Oct. 2011. http://proval.lri.fr/publications/bardou11phd.pdf.

[2] F. Bobot. *Logique de séparation et vérification déductive*. Thèse de doctorat, Université Paris-Sud, Dec. 2011.

[3] F. Bobot, S. Conchon, E. Contejean, M. Iguernelala, A. Mahboubi, A. Mebsout, and G. Melquiond. A Simplex-based extension of Fourier-Motzkin for solving linear integer arithmetic. In B. Gramlich, D. Miller, and U. Sattler, editors, *IJCAR 2012 : Proceedings of the 6th International Joint Conference on Automated Reasoning*, volume 7364 of *Lecture Notes in Computer Science*, pages 67–81, Manchester, UK, June 2012. Springer.

[4] F. Bobot, J.-C. Filliâtre, C. Marché, G. Melquiond, and A. Paskevich. *The Why3 platform, version 0.72*. LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, version 0.72 edition, May 2012.

[5] F. Bobot, J.-C. Filliâtre, C. Marché, G. Melquiond, and A. Paskevich. *The Why3 platform, version 0.73*. LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, version 0.73 edition, July 2012.

[6] F. Bobot, J.-C. Filliâtre, C. Marché, G. Melquiond, and A. Paskevich. *The Why3 platform, version 0.80*. LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, version 0.80 edition, Oct. 2012. https://gforge.inria.fr/docman/view.php/2990/8186/manual-0.80.pdf.

[7] F. Bobot, J.-C. Filliâtre, C. Marché, G. Melquiond, and A. Paskevich. Preserving user proofs across specification changes. In E. Cohen and A. Rybalchenko, editors, *Verified Software : Theories, Tools, Experiments (5th International Conference VSTTE)*, Lecture Notes in Computer Science, Atherton, USA, May 2013. Springer.

[8] F. Bobot, J.-C. Filliâtre, C. Marché, G. Melquiond, and A. Paskevich. *The Why3 platform, version 0.81*. LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, version 0.81 edition, Mar. 2013. http://why3.lri.fr/download/manual-0.81.pdf.

[9] F. Bobot, J.-C. Filliâtre, C. Marché, and A. Paskevich. *The Why3 platform, version 0.71*. LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, version 0.71 edition, Oct. 2011. https://gforge.inria.fr/docman/view.php/2990/7635/manual.pdf.

[10] F. Bobot, J.-C. Filliâtre, C. Marché, and A. Paskevich. Why3 : Shepherd your herd of provers. In *Boogie 2011 : First International Workshop on Intermediate Verification Languages*, pages 53–64, Wrocław, Poland, August 2011.

[11] F. Bobot and A. Paskevich. Expressing Polymorphic Types in a Many-Sorted Language. In C. Tinelli and V. Sofronie-Stokkermans, editors, *Frontiers of Combining Systems, 8th International Symposium, Proceedings*, volume 6989 of *Lecture Notes in Computer Science*, pages 87–102, Saarbrücken, Germany, Oct. 2011.

[12] S. Boldo and C. Marché. Formal verification of numerical programs : from C annotated programs to mechanical proofs. *Mathematics in Computer Science*, 5 :377–393, 2011.

[13] T. Bormer, M. Brockschmidt, D. Distefano, G. Ernst, J.-C. Filliâtre, R. Grigore, M. Huisman, V. Klebanov, C. Marché, R. Monahan, W. Mostowski, N. Polikarpova, C. Scheben, G. Schellhorn, B. Tofan, J. Tschannen, and M. Ulbrich. The COST IC0701 verification competition 2011. In B. Beckert, F. Damiani, and D. Gurov, editors, *Formal Verification of Object-Oriented Software, Revised Selected Papers Presented at the International Conference, FoVeOOS 2011*, volume 7421 of *Lecture Notes in Computer Science*. Springer, 2012.

[14] S. Conchon. *SMT Techniques and their Applications : from Alt-Ergo to Cubicle*. Thèse d'habilitation, Université Paris-Sud, Dec. 2012. In English, http://www.lri.fr/~conchon/publis/conchonHDR.pdf.

[15] S. Conchon. From Alt-Ergo to Cubicle (invited talk). In *Satisfiability Modulo Theory Workshop*, Helsinki, Finland, July 2013.

[16] S. Conchon, E. Contejean, and M. Iguernelala. Ground Associative and Commutative Completion Modulo Shostak Theories. In A. Voronkov, editor, *LPAR, 17th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, EasyChair Proceedings, Yogyakarta, Indonesia, Oct. 2010. (short paper).

[17] S. Conchon, E. Contejean, and M. Iguernelala. Canonized Rewriting and Ground AC Completion Modulo Shostak Theories. In P. A. Abdulla and K. R. M. Leino, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 6605 of *Lecture Notes in Computer Science*, pages 45–59, Saarbrücken, Germany, Apr. 2011. Springer.

[18] S. Conchon, E. Contejean, and M. Iguernelala. Canonized rewriting and ground AC completion modulo Shostak theories : Design and implementation. *Logical Methods in Computer Science*, 8(3) :1–29, Sept. 2012. Selected Papers of the Conference *Tools and Algorithms for the Construction and Analysis of Systems* (TACAS 2011), Saarbrücken, Germany, 2011.

[19] S. Conchon, A. Mebsout, and F. Zaïdi. Vérification de systèmes paramétrés avec Cubicle. In *Vingt-quatrièmes Journées Francophones des Langages Applicatifs*, Aussois, France, Feb. 2013.

[20] S. Conchon, G. Melquiond, C. Roux, and M. Iguernelala. Built-in treatment of an axiomatic floating-point theory for SMT solvers. In P. Fontaine and A. Goel, editors, *SMT workshop*, pages 12–21, Manchester, UK, 2012. LORIA.

[21] D. Cousineau and O. Hermant. A semantic proof that reducibility candidates entail cut elimination. In A. Tiwari, editor, *23nd International Conference on Rewriting Techniques and Applications*, volume 15

of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 133–148, Nagoya, Japan, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[22] C. Dross, S. Conchon, J. Kanig, and A. Paskevich. Reasoning with triggers. In P. Fontaine and A. Goel, editors, *SMT workshop*, Manchester, UK, 2012. LORIA.

[23] C. Dross, S. Conchon, J. Kanig, and A. Paskevich. Reasoning with triggers. Research Report RR-7986, INRIA, June 2012.

[24] C. Dross, J.-C. Filliâtre, and Y. Moy. Correct Code Containing Containers. In *5th International Conference on Tests and Proofs (TAP'11)*, volume 6706 of *Lecture Notes in Computer Science*, pages 102–118, Zurich, June 2011. Springer.

[25] J.-C. Filliâtre. *Deductive Program Verification*. Thèse d'habilitation, Université Paris-Sud, Dec. 2011. In English, http://www.lri.fr/~filliatr/hdr/memoire.pdf.

[26] J.-C. Filliâtre. Deductive software verification. *International Journal on Software Tools for Technology Transfer (STTT)*, 13(5) :397–403, Aug. 2011.

[27] J.-C. Filliâtre. Combining Interactive and Automated Theorem Proving in Why3 (invited talk). In K. Heljanko and H. Herbelin, editors, *Automation in Proof Assistants 2012*, Tallinn, Estonia, April 2012.

[28] J.-C. Filliâtre. Combining Interactive and Automated Theorem Proving using Why3 (invited tutorial). In Z. Rakamarić, editor, *Second International Workshop on Intermediate Verification Languages (BOOGIE 2012)*, Berkeley, California, USA, July 2012.

[29] J.-C. Filliâtre. *Course notes EJCP 2012*, chapter Vérification déductive de programmes avec Why3. June 2012.

[30] J.-C. Filliâtre. Verifying two lines of C with Why3 : an exercise in program verification. In R. Joshi, P. Müller, and A. Podelski, editors, *Verified Software : Theories, Tools, Experiments (4th International Conference VSTTE)*, volume 7152 of *Lecture Notes in Computer Science*, pages 83–97, Philadelphia, USA, Jan. 2012. Springer.

[31] J.-C. Filliâtre. Deductive Program Verification. In N. Foster, P. Gardner, A. Schmitt, G. Smith, P. Thieman, and T. Wrigstad, editors, *Programming Languages Mentoring Workshop (PLMW 2013)*, Rome, Italy, January 2013.

[32] J.-C. Filliâtre. Deductive program verification with Why3. Lecture notes for the First DigiCosme Spring School, https://www.lri.fr/~marche/DigiCosmeSchool/filliatre.html, 2013.

[33] J.-C. Filliâtre. One logic to use them all. In *24th International Conference on Automated Deduction (CADE-24)*, volume 7898 of *Lecture Notes in Artificial Intelligence*, pages 1–20, Lake Placid, USA, June 2013. Springer.

[34] J.-C. Filliâtre and A. Paskevich. Why3 — where programs meet provers. In M. Felleisen and P. Gardner, editors, *Proceedings of the 22nd European Symposium on Programming*, volume 7792 of *Lecture Notes in Computer Science*, pages 125–128. Springer, Mar. 2013.

[35] J.-C. Filliâtre, A. Paskevich, and A. Stump. The 2nd verified software competition : Experience report. In V. Klebanov and S. Grebing, editors, *COMPARE2012 : 1st International Workshop on Comparative Empirical Evaluation of Reasoning Systems*, Manchester, UK, June 2012. EasyChair.

[36] P. Herms. *Certification of a Tool Chain for Deductive Program Verification*. Thèse de doctorat, Université Paris-Sud, Jan. 2013.

[37] P. Herms, C. Marché, and B. Monate. A certified multi-prover verification condition generator. In R. Joshi, P. Müller, and A. Podelski, editors, *Verified Software : Theories, Tools, Experiments (4th International Conference VSTTE)*, volume 7152 of *Lecture Notes in Computer Science*, pages 2–17, Philadelphia, USA, Jan. 2012. Springer.

[38] S. Lescuyer. *Formalisation et développement d'une tactique réflexive pour la démonstration automatique en Coq*. Thèse de doctorat, Université Paris-Sud, Jan. 2011.

[39] C. Marché and A. Tafat. Weakest precondition calculus, revisited using Why3. Research Report RR-8185, INRIA, Dec. 2012.

[40] C. Marché and A. Tafat. Calcul de plus faible précondition, revisité en Why3. In *Vingt-quatrièmes Journées Francophones des Langages Applicatifs*, Aussois, France, Feb. 2013.

[41] D. Mentré, C. Marché, J.-C. Filliâtre, and M. Asuka. Discharging proof obligations from Atelier B using multiple automated provers. In S. Reeves and E. Riccobene, editors, *ABZ'2012 - 3rd International Conference on Abstract State Machines, Alloy, B and Z*, volume 7316 of *Lecture Notes in Computer Science*, pages 238–251, Pisa, Italy, June 2012. Springer. http://hal.inria.fr/hal-00681781/en/.

[42] Y. Moy and C. Marché. *The Jessie plugin for Deduction Verification in Frama-C — Tutorial and Reference Manual*. INRIA & LRI, 2011. http://krakatoa.lri.fr/.

[43] A. Paskevich. Algebraic types and pattern matching in the logical language of the Why verification platform (version 2). Technical Report 7128, INRIA, 2010. http://hal.inria.fr/inria-00439232/en/.

[44] A. Tafat and C. Marché. Binary heaps formally verified in Why3. Research Report 7780, INRIA, Oct. 2011. http://hal.inria.fr/inria-00636083/en/.