Le solveur SMT veriT*

David Déharbe†(Univ. Féd. Rio Grande do Norte, Brésil)
Pablo Federico Dobal (Inria Nancy - Grand-Est, Loria)
Pascal Fontaine‡(Inria Nancy - Grand-Est, Loria, Université de Lorraine)

Les méthodes formelles dépendent souvent, *in fine*, de la validation de conditions de vérification, parfois nombreuses. Valider ces formules a très tôt été identifié comme un des verrous majeurs pour l'application des méthodes formelles à des problèmes industriels. S'il est possible de vérifier manuellement ces formules, des techniques puissantes de déduction automatique ont vu le jour parallèlement aux développement des plate-formes de vérification. Ces techniques maintenant arrivées à un certain niveau de maturité sont progressivement intégrées, comme dans Rodin [5] (pour B [1]), et dans TLAPS [6] (pour TLA$^+$ [11, 12]), en diminuant grandement le besoin d'interaction humaine pour la validation des formules. Par exemple, l'intégration de solveurs SMT [8] dans Rodin permet, sur un large dossier d'obligations de preuves académiques et industrielles, de diminuer d'un facteur 4 le nombre de formules nécessitant une intervention humaine.

Nous présentons ici le prouveur de formules veriT [4]. Un ancêtre de ce prouveur, haRVey, a déjà fait l'objet d'une démonstration à AFADL en 2007 [7], mais la version actuelle présente de nombreuses et sensibles améliorations par rapport à son prédécesseur.

Le logiciel veriT, comme la plupart des solveurs SMT (*Satisfiability Modulo Theories* [2]), s'appuie, pour la gestion de la structure booléenne des formules, sur un solveur SAT (voir par exemple [3, 9]). Ces logiciels décident efficacement le problème de la satisfaisabilité booléenne. Un exemple simple de formule pouvant être réfuté au moyen d'un solveur SAT est la suivante :

$$\neg\big[\,(p \Rightarrow q) \Rightarrow \big[\,(\neg p \Rightarrow q) \Rightarrow q\big]\big].$$

Les solveurs SMT enrichissent le langage accepté par les solveurs SAT en un langage toujours décidable, mais plus expressif que la logique booléenne. Le logiciel veriT, comme nombre de ses concurrents, accepte l'égalité, et les symboles non interprétés, en utilisant l'algorithme de clôture de congruence [10, 14]. Il est donc capable par exemple de reconnaître que la formule

$$a = b \wedge \big[f(a) \neq f(b) \vee (p(a) \wedge \neg p(b))\big]$$

est inconsistante. Les techniques de combinaison de théories à la Nelson-Oppen [13, 15] permettent d'intégrer le raisonnement de divers théories décidables. Par exemple, il est possible, à partir de la fermeture de congruence, et d'une procédure de décision pour l'arithmétique linéaire, de construire une procédure de décision qui comprend à la fois les symboles non-interprétés, et les symboles de l'arithmétique linéaire, afin d'analyser une formule du type :

$$a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big[f(a) \neq f(b) \vee (p(a) \wedge \neg p(b + x))\big].$$

Le but de cette démonstration est de donner à la communauté des méthodes formelles une idée précise du langage accepté par l'outil, et de lui permettre d'apprécier les performances du logiciel. Par

*This work has been partially supported by CNPq/INRIA project SMT-SAVeS, and CNPq grant 573964/2008-4 (National Institute of Science and Technology for Software Engineering—INES).

†david@dimap.ufrn.br
‡Pascal.Fontaine@inria.fr

cette démonstration et l'échange qui en suivra, nous espérons aussi avoir de la communauté des retours qui pourront guider le développement des versions futures du logiciel. Le logiciel est disponible sous licence *open-source* BSD sur le site `http://www.veriT-solver.org`.

Références

[1] J.-R. Abrial. *The B-Book : Assigning Programs to Meanings*. Cambridge University Press, 1996.

[2] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, Feb. 2009.

[3] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[4] T. Bouton, D. C. B. de Oliveira, D. Déharbe, and P. Fontaine. veriT : an open, trustable and efficient SMT-solver. In R. Schmidt, editor, *Proc. Conference on Automated Deduction (CADE)*, volume 5663 of *Lecture Notes in Computer Science*, pages 151–156, Montreal, Canada, 2009. Springer.

[5] J. Coleman, C. Jones, I. Oliver, A. Romanovsky, and E. Troubitsyna. RODIN (Rigorous open Development Environment for Complex Systems). In *Fifth European Dependable Computing Conference : EDCC-5 supplementary volume*, pages 23–26, 2005.

[6] D. Cousineau, D. Doligez, L. Lamport, S. Merz, D. Ricketts, and H. Vanzetto. TLA+ proofs. In *FM*, volume 7436 of *Lecture Notes in Computer Science*, pages 147–154. Springer, 2012.

[7] D. C. B. de Oliveira, D. Déharbe, and P. Fontaine. haRVey : satisfaisabilité et théories, June 2007. Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL).

[8] D. Déharbe, P. Fontaine, Y. Guyot, and L. Voisin. SMT solvers for Rodin. In J. Derrick, J. A. Fitzgerald, S. Gnesi, S. Khurshid, M. Leuschel, S. Reeves, and E. Riccobene, editors, *ABZ*, volume 7316 of *Lecture Notes in Computer Science*, pages 194–207. Springer, 2012.

[9] N. Eén and N. Sörensson. An extensible SAT-solver. In E. Giunchiglia and A. Tacchella, editors, *SAT*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[10] P. Fontaine. *Techniques for verification of concurrent systems with invariants*. PhD thesis, Institut Montefiore, Université de Liège, Belgium, Sept. 2004.

[11] L. Lamport. *Specifying Systems*. Addison-Wesley, Boston, Mass., 2002.

[12] S. Merz. On the logic of TLA$^+$. *Computers and Informatics*, 22 :351–379, 2003.

[13] G. Nelson and D. C. Oppen. Simplifications by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2) :245–257, Oct. 1979.

[14] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2) :356–364, Apr. 1980.

[15] C. Tinelli and M. T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems (FroCoS)*, Applied Logic, pages 103–120. Kluwer Academic Publishers, Mar. 1996.