haRVey: satisfaisabilité et théories

Diego Caminha B. de Oliveira (Univ. Rio Grande do Norte, Brésil)
David Déharbe* (Univ. Rio Grande do Norte, Brésil),
Pascal Fontaine** (LORIA – Université de Nancy)

Univ. Rio Grande do Norte, Brésil / LORIA – Université de Nancy

Le problème de la satisfaisabilité de formules logiques est au centre des méthodes formelles: les modèles formels (par exemple la méthode B [1], et TLA$^+$ [6, 7]) et les raffinements sont validés au moyen d'un générateur de conditions de vérification, la preuve de ces formules étant déléguée à un module particulier de l'atelier de modélisation. Nous présentons ici le prouveur de formules haRVey.

Le logiciel haRVey, comme la plupart des solveurs SMT (Satisfiability Modulo Theories [10]), s'appuie, pour la gestion de la structure booléenne des formules, sur un solveur SAT (voir par exemple [12, 4]). 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 élèvent le langage accepté par les solveurs SAT à un langage toujours décidable, mais plus expressif que la logique booléenne. Le logiciel haRVey, comme nombre de ses concurrents, accepte l'égalité, et les symboles non interprétés, en utilisant l'algorithme de clôture de congruence [5, 9]. 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 [8, 11] 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 comprends à la fois les symboles non-interprétés, et les symboles de l'arithmétique linéaire, afin d'analyser une formule du type:

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

Nous montrons aussi deux capacités originales de l'outil: son aptitude à accepter dans une combinaison, des théories du premier ordre fixées par l'utilisateur (par un ensemble d'axiomes, voir par exemple [2]), et sa capacité à traiter certains opérateurs ensemblistes, comme dans:

$$a = b \wedge f(a) \in (A \cap B) \wedge \big[f(a) \in A \setminus B \vee f(b) \notin B\big].$$

* david@dimap.ufrn.br
** Pascal.Fontaine@loria.fr

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. Nous voulons identifier les méthodes de modélisation et de développement prouvé pouvant bénéficier, pour la vérification des formules générées pendant le processus, de l'assistance de ce logiciel de preuve automatique. Enfin, les besoins des utilisateurs pourront guider le développement de la version future du logiciel [3].

On peut télécharger le logiciel, et trouver les publications le décrivant, sur le site `http://harvey.loria.fr`.

References

1. J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.
2. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures. *Information and Computation*, 183(2):140–164, 2003.
3. D. Déharbe and P. Fontaine. haRVey: combining reasoners. In *Sixth International Workshop on Automated Verification of Critical Systems*, 2006.
4. 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.
5. P. Fontaine. *Techniques for verification of concurrent systems with invariants*. PhD thesis, Institut Montefiore, Université de Liège, Belgium, Sept. 2004.
6. L. Lamport. *Specifying Systems*. Addison-Wesley, Boston, Mass., 2002.
7. S. Merz. On the logic of TLA$^+$. *Computers and Informatics*, 22:351–379, 2003.
8. G. Nelson and D. C. Oppen. Simplifications by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, Oct. 1979.
9. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, Apr. 1980.
10. S. Ranise and C. Tinelli. The SMT-LIB standard : Version 1.2, Aug. 2006.
11. 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.
12. L. Zhang and S. Malik. The quest for efficient Boolean satisfiability solvers. In A. Voronkov, editor, *Proc. Conference on Automated Deduction (CADE)*, volume 2392 of *Lecture Notes in Computer Science*, pages 295–313. Springer-Verlag, 2002.