# haRVey: combining reasoners

David Déharbe, Pascal Fontaine

# haRVey: combining reasoners

David Déharbe[1]

*UFRN - CCET - DIMAp*
*Campus Universitário - Lagoa Nova*
*59072-970 Natal, RN, BRAZIL*

Pascal Fontaine[2]

*LORIA and University of Nancy*
*BP 239, Campus scientifique*
*54506 Vandœuvre les Nancy, FRANCE*

**Abstract**

We present the architecture of the oncoming version of the SMT (Satisfiability Modulo Theories) solver haRVey [5]. haRVey checks the satisfiability of a formula written in a first-order language with interpreted symbols from various theories. Its new architecture is original, first in the sense that it is a combination of reasoners, rather than the traditional combination of decision procedures. Second, one of these reasoners is a full-featured first-order saturation-based prover. Finally, some of those reasoners in the combination may only be sporadically activated not using computer time when inactive. We believe those new features will contribute to the efficiency and expressivity of the new version of the tool.

*Keywords:* first-order logic, combination of theories, decision procedures, automatic theorem proving

## 1 Introduction

The Satisfiability Modulo Theories (SMT) problem is the satisfiability problem for first-order formulas (most often, quantifier-free) containing interpreted symbols from one theory or a combination of theories (for instance, linear arithmetics, the theories of arrays, lists, or of other programming data structures, ...). This problem has attracted much interest recently,[3] because the language the solvers tackle is particularly appropriate for verification problems, in bounded model-checking but also in formal verification of algorithms [3].

SMT solvers usually use techniques from SAT-solvers to deal with the propositional structure of the formulas, thus lowering the problem to checking the satisfiability of *sets of literals* modulo a theory or a combination of theories. The

[1] Email: david@dimap.ufrn.br

[2] Email: Pascal.Fontaine@loria.fr

[3] A competition between SMT solvers has been organised for the first time in 2005 and is reconducted this year [1].

SAT-solvers enumerate models of the propositional abstraction of the formulas, and the theory reasoner refutes the set of literals corresponding to those models. [4] For instance, while checking a formula that contains interpreted symbols from linear arithmetics, and uninterpreted symbols, the theory reasoner may have to refute the following set of literals:

$$L = \{x \leq y,\ y \leq x + f(x),\ P(h(x) - h(y)),\ \neg P(0),\ f(x) = 0\}.$$

The Nelson-Oppen [6,8] framework allows to build a decision procedure for a union of theories from the decision procedures for elementary theories: here, a decision procedure for linear arithmetics, and a decision procedure for uninterpreted symbols. The first step of the technique is purification: in our example, $L$ is split in $L_1$ and $L_2$ using new variables, such that $L_1 \cup L_2$ is equisatisfiable to $L$, and $L_1$ only contains linear arithmetic symbols (and variables), whereas $L_2$ contains only uninterpreted symbols (and variables):

$$L_1 = \{x \leq y,\ y \leq x + v_1,\ v_1 = 0,\ v_2 = v_3 - v_4,\ v_5 = 0\}$$
$$L_2 = \{P(v_2),\ \neg P(v_5),\ v_1 = f(x),\ v_3 = h(x),\ v_4 = h(y)\}.$$

Each set is then handled by its corresponding decision procedure. However this is not sufficient to conclude to the unsatisfiability of $L$: $L_1$ and $L_2$ are satisfiable, even if their union is not. It is also required for the decision procedures to exchange (disjunctions of) equalities between shared variables. For instance, $x = y$ is entailed by $L_1$. Adding this new fact to $L_2$ allows to deduce more equalities. All those entailed facts will eventually lead to the unsatisfiability of one set or the other, *using only decision procedures for the elementary languages*. Combining decision procedures is the key to obtain a rich language for SMT solvers. For the framework to be complete, the decision procedures in the combination should be stably-infinite [8], and mutually disjoint (i.e., no interpreted symbol can be shared).

The new implementation of the haRVey solver allows to combine not only disjoint decision procedures, but more generally, reasoners that may handle non-disjoint languages (Section 2). In particular, one of those reasoners is a full-featured first-order theorem prover (that is, the E prover [7]). Although haRVey focus on expressivity rather than efficiency, it is crucial for efficiency to have some intelligent scheduling capabilities, if some reasoners are very expensive. We describe these capabilities in Section 3.

## 2 Combining reasoners

In a combination of decision procedures, each elementary decision procedure has to propagate equalities between shared variables to the other decision procedures. Non-convex decision procedures [5] may also have to propagate disjunctions of equalities. Those disjunctions would involve case splitting at the theory reasoning level. However case splitting is a work that suits SAT-solvers better; it is thus natural to move this splitting from the theory reasoning module to the propositional reasoning module. As a consequence, we consider that the decision procedures interact

[4] Some techniques allow to avoid all models to be enumerated. This is crucial for efficiency.

[5] A theory is said to be non-convex when a disjunction of equalities can be derived from a conjunction of equalities.

directly only through equality exchanges, relying on the SAT-solver to exchange disjunction of equalities (like for instance in [2]).

More generally, we assume that each reasoner in the combination may

- produce entailed equalities. Those equalities are passed directly to the other reasoners in the combination;
- produce arbitrary formulas that are valid according to its own logic, but nonetheless informative for the rest of the combination. Those lemmas are *anded* to the original formula, at the SAT-solver level.

As an example, consider the non-convex theory for linear arithmetic on integers. While deciding the satisfiability of a formula $\varphi$, the reasoner may have to deal with the set $\{x < 5, x > 2, y = 3, z = 4\}$. In the classic approach, it would have been necessary to do a case split, and consider separately $x = y$ and $x = z$. To discharge the case split to the SAT-solver, the reasoner produces the valid lemma $\psi = (x < 5 \wedge x > 2 \wedge y = 3 \wedge z = 4) \Rightarrow (x = y \vee x = z)$. The formula considered by the SAT-solver will then be $\psi \wedge \varphi$, which is logically equivalent to $\varphi$. Subsequent sets of literals that are generated by the SAT-solver and passed to the theory reasoners will all assign the value of one literal in the lemma to make the lemma true. Implicitly, the different cases are split by the SAT-solver, at the propositional level.

Introducing lemmas at the propositional level is convenient to deal with non-convex theories, but foremost, it allows to extend the language in an easy and sound way. For instance, assume one want to decide the satisfiability of formulas containing some set constructions, like in:

$$\{a = b, (\{f(a)\} \cup E) \subseteq A, f(b) \notin C, A \cup B = C \cap D\}.$$

This can be simply rewritten by replacing the set operations by operations on the characteristic functions of the sets. It becomes:

$$\varphi = \{a = b, \forall x[(x = f(a) \vee E(x)) \Rightarrow A(x)], \\ \neg C(f(b)), \forall x.\ [A(x) \vee B(x)] \equiv [C(x) \wedge D(x)]\}$$

This last set of formulas contains literals, and quantified formulas. Assume now we have an *instantiation reasoner* aware of the quantified formulas and symbols used in the set. The instantiation reasoner may generate the (valid) lemmas:

$$\psi_1 = \forall x[(x = f(a) \vee E(x)) \Rightarrow A(x)] \Rightarrow [(f(a) = f(a) \vee E(f(a))) \Rightarrow A(f(a))]$$

and

$$\psi_2 = \forall x[[A(x) \vee B(x)] \equiv [C(x) \wedge D(x)]] \\ \Rightarrow [[A(f(b)) \vee B(f(b))] \equiv [C(f(b)) \wedge D(f(b))]].$$

Like $\varphi$, the formula $\psi_1 \wedge \psi_2 \wedge \varphi$ is unsatisfiable. But, with those lemmas from the instantiation reasoner, the resulting formula can be showed to be unsatisfiable by using only simple reasoning on equalities and uninterpreted symbols, all quantified formulas being abstracted by propositional atoms.

Assuming a generic mechanism of lemma generation may also be used to improve efficiency of the prover. Indeed, techniques such as theory propagation—shown to yield significative performance improvement of SMT provers based on DPLL SAT-solvers [2]—can be viewed as instances of lemma generation.

As long as all generated lemmas are valid, soundness is guaranteed. Completeness can be guaranteed in some cases, for instance (but not only) for the combination of stably-infinite disjoint (convex or non-convex) theories.

## 3 Scheduling reasoners

The Nelson-Oppen module in haRVey has been designed under the requirement that it should be tightly integrated with a propositional SAT-solver. A SAT-solver maintains a database of clauses representing a propositional abstraction of the formula to be checked for satisfiability. The SAT-solver builds propositional models incrementally, by repeatedly propagating unit clauses, making decisions, assigning propositional variables, and backtracking when a conflict occurs at the propositional level. The interaction with the Nelson-Oppen module occurs to check the satisfiablity of the current assignment modulo the background theory. This interaction can be realized on the fly while building the propositional model. In that case, we expect that the cost of this interaction remains low, since the number of decisions and propagations made by the SAT-solver is potentially large. However, when the SAT-solver has found a complete propositional model (i.e. that cannot be further extended), the Nelson-Oppen module needs to guarantee completeness.

The Nelson-Oppen module should meet the following design requirements:

- it should accept literals incrementally, and these literals may be retracted, following a LIFO order. The cost of such operations should be very low;
- solving may be realized in full force (when propositional models are complete), or not (when models are partial). For instance, we may want to apply a low-cost solve operation before a variable assignment at the propositional level.

For the design of the module that provide these facilities, we took into account the fact that the individual reasoners that make up the Nelson-Oppen combination may also have different capabilities:

- some reasoners are incremental and backtrackable, others are incremental only, and some may be neither;
- the computational cost of the reasoners may differ widely. Also, there may be several procedures for a theory (say, arithmetics), with varying complexities, and degrees of completeness. The reasoners are thus parameterized by the effort that they are expected to spend.

In order to address all these constraints, we devised a scheduler for the different reasoners. This scheduler only knows the basic characteristics of each reasoner (incrementality, backtrackability, effort levels they can address). The scheduler is also (indirectly) aware of the current phase of the SAT-solver, since it operates differently when a new literal is pushed, when the model is partial, or when the SAT-solver has found a complete model for the propositional abstraction. The scheduler is responsible also for disactivating reasoners that are not required: for instance, a very expensive reasoner (like the saturation prover) is only activated to check complete models. The total number of reasoners has no impact on the cost of the combination schema. Moreover, the congruence closure module [4] centralises all deduced

equalities; when an expensive reasoner is activated, it gets summarized information about the proof state. This information, which is computed by the congruence closure, already takes into account all the entailed equalities. The expensive reasoner only has to deal with the relevant terms.

## 4 Conclusion

We presented some aspects of the design of the oncoming version of the haRVey SMT solver. This version unifies two branches of haRVey: the haRVey-SAT branch is mainly a cooperation of a SAT-solver with congruence closure (plus an instantiation reasoner, and limited linear arithmetic support), whereas the main branch is basically a cooperation of a saturation-based first-order theorem prover, an incomplete arithmetic module, and a propositional reasoner (originally based on BDDs). At the present time, we are reaching the point of the first runs of the prototype. We hope to have a working version in a few weeks, and a full-featured tool by this year.

Future work includes tuning the tool, by studying the effect on the performances of different scheduling strategies. The main efforts will then be directed to the design of supplementary reasoners (e.g. based on rewriting), and to a better integration of the different modules in the tool. For instance, we plan to investigate how to use several instances of the saturation-based first-order theorem prover, that would work independently on the disjoint theories.

On the theoretical side, we will investigate the completeness of the combination of reasoners. Soundness is simply guaranteed by the fact that all produced lemmas are valid. Completeness however heavily depends on the nature of the reasoners, and their interactions (notably, the scheduling strategy, and the lemma production).

## References

[1] Barrett, C., L. de Moura and A. Stump, *SMT-COMP: Satisfiability Modulo Theories Competition*, in: K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification* (2005), pp. 20–23.

[2] Barrett, C., R. Nieuwenhuis, A. Oliveras and C. Tinelli, *Splitting on demand in satisfiability modulo theories*, in: *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, Lecture Notes in Computer Science (2006), to be published.

[3] Barsotti, D., L. P. Nieto and A. Tiu, *Verification of clock synchronization algorithms: Experiments on a combination of deductive tools*, in: R. Lazic and R. Nagarajan, editors, *Proceedings of the 5th International Workshop on Automated Verification of Critical Systems (AVoCS 2005)*, Electronic Notes in Theoretical Computer Science **145** (2005), pp. 63–78.

[4] Fontaine, P., "Techniques for verification of concurrent systems with invariants," Ph.D. thesis, Institut Montefiore, Université de Liège, Belgium (2004).

[5] *The haRVey prover*, http://harvey.loria.fr/.

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

[7] Schulz, S., *System Description: E 0.81*, in: D. Basin and M. Rusinowitch, editors, *Proc. of the 2nd IJCAR, Cork, Ireland*, LNAI **3097** (2004), pp. 223–228.

[8] Tinelli, C. 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 (1996), pp. 103–120.
URL citeseer.nj.nec.com/tinelli96new.html