Local reasoning in verification: Implementation and experiments

Carsten Ihlemann, Swen Jacobs, Viorica Sofronie-Stokkermans

Max-Planck-Institut für Informatik, Campus E 1.4, Saarbrücken, Germany

We have implemented the approach for hierarchical reasoning described in [1] as well as the extensions to reasoning in chains of theory extensions [2, 3]. Our tool (LoRe) allows us to reduce satisfiability problems in an extended theory to a base theory for which we can then use existing solvers. It takes as input the axioms of the theory extension, the ground goal and the list of extension function symbols. Chains of extensions are handled by having a list of axiom sets, and correspondingly a list of lists of extension function symbols.

- The input is analyzed for ground terms with extension functions at the root.
- After instantiating the axioms w.r.t. these terms, the extension symbols are hierarchically removed.
- The resulting formula is either given to a prover for a base theory, or taken as goal for another reduction (if we have a chain of extensions).

The program has two modes of execution. The first mode automatically produces inputs for a class of theorem provers for the base theory. Currently, we can produce base theory output for Yices, Mathsat and Redlog, but other solvers can be integrated easily. In the second mode, the interaction with the theorem prover for the base theory is integrated in the prover itself (the choice of the prover is settled by command line switch).

We run tests on various examples, including different versions of the train controller example [2, 3], an array version of the insertion algorithm, and reasoning in theories of arrays and lists. In Fig. 1 we list running times for 8 different versions of the train example, each of the following four versions in a satisfiable and an unsatisfiable variant:

- simple (fixed number of trains, no length measure),
- length (fixed number of trains with length),
- variable (variable number of trains, no length measure),
- full (variable number of trains with length).

We set the timeout to 300 seconds. In Fig. 1, *not supported* means that we would have to hand a non-linear arithmetic problem to Yices. We can instead use Redlog. We have successfully used it for these problems.

	$\lvert\mathcal{K}_\infty\rvert$	$\lvert\mathcal{K}_\infty[\mathcal{G}]\rvert$	$\lvert\mathcal{K}_\in\rvert$	$\lvert\mathcal{K}_\in[\mathcal{G}]\rvert$	Runtimes (sec)		
					LoRe	LoRe+Yices	Yices
simple $_{\mathsf{SAT}}$	5	20	4	8	0.028	0.049	timeout
simple $_{\mathsf{UNSAT}}$	6	21	4	8	0.027	0.047	0.032
length $_{\mathsf{SAT}}$	6	21	4	8	0.031	0.052*	not supported
length $_{\mathsf{UNSAT}}$	7	22	4	8	0.031	0.053*	not supported
variable $_{\mathsf{SAT}}$	8	71	9	21	0.128	0.183	timeout
variable $_{\mathsf{UNSAT}}$	9	72	9	21	0.135	0.181	0.052
full $_{\mathsf{SAT}}$	9	72	9	21	0.156	not supported	not supported
full $_{\mathsf{UNSAT}}$	10	73	9	21	0.167	not supported	not supported

The times for benchmark 3 and 4 are marked with '*' in order to point out that the replacement of the differences $(u - v) * \mathsf{LengthTrain}$, where $u, v \in \{k, k', k'', k'''\}$ has been done manually. This was necessary because the input for Yices cannot contain non-linear arithmetical expressions. We used the following abbreviations:

- $|\mathcal{K}_\rangle|$: the number of clauses which define extension i in the chain of local extensions.
- $|\mathcal{K}_\rangle[\mathcal{G}]|$: the number of instances which we generate.
- **LoRe time:** the time needed by LoRe for the hierarchical reduction.
- **LoRe+Yices time:** the total time for reduction and use of Yices for SAT checking.
- **Yices time:** the time needed by Yices for solving the original problem.

Fig. 1. Tests with the RBC case study (various versions)

While Yices can be used successfully also directly for unsatisfiable formulae, this does not hold if we change the input problem to a formula which is satisfiable w.r.t. the extended theory. In this case, Yices returns "unknown" after a 300 second timeout. After the reduction with our tool, Yices (applied to the problem for the base theory) returns "satisfiable" in a fraction of a second, and even supplies a model for this problem that can be lifted to a model in the extended theory for the initial set of clauses[1]. Even more information can be obtained using the quantifier elimination facilities offered e.g. by Redlog for determining *constraints between the parameters* of the problems which guarantee safety.

In Fig. 2 we list running times for two examples involving reasoning about arrays: an array insertion algorithm and an example considered in [4].

We are working towards extending the tool support to stable locality, and for extensions with clauses containing proper first-order formulae.

[1] The lifting is straightforward, given the output of our tool, but is not automated at the moment.

	$\lvert\mathcal{K}\rvert$	$\lvert G\rvert$	$\lvert\ \mathcal{K}[\mathcal{G}]$	Runtimes (sec) (Mode 2; line switch -Yices)
array-insert	9	4	64	0.080
array-example [4]	10	11	640	0.400

We used the following abbreviations:

– $\lvert\mathcal{K}\rvert$: the number of clauses which define the extension.
– $\lvert G\rvert$: the size of the set of ground clauses to be proved/disproved.
– $\lvert\mathcal{K}[\mathcal{G}]\rvert$: the number of instances which we generate.
– **Runtimes:** times in mode 2, using a direct call of Yices.

Fig. 2. Reasoning about arrays

References

1. Sofronie-Stokkermans, V.: Hierarchic reasoning in local theory extensions. In: 20th Int. Conf. on Automated Deduction (CADE-20), LNAI 3632, Springer (2005) 219–234
2. Jacobs, S., Sofronie-Stokkermans, V.: Applications of hierarchical reasoning in the verification of complex systems. Electronic Notes in Theoretical Computer Science **174** (2007) 39–54
3. Faber, J., Jacobs, S., Sofronie-Stokkermans, V.: Verifying CSP-OZ-DC specifications with complex data types and timing parameters. In: Integrated Formal Methods (IFM 2007), LNCS 4591, Springer (2007) 233–252
4. Bradley, A., Manna, Z., Sipma, H.: What's decidable about arrays? In: Verification, Model-Checking, and Abstract-Interpretation, 7th Int. Conf. (VMCAI 2006). LNCS 3855, Springer (2006) 427–442