Efficient Ground Satisfiability Solving in an Instantiation-based Method for First-order Theorem Proving

Christoph Sticksel[⋆]
[⋆]University of Manchester
csticksel@cs.man.ac.uk

1 Introduction

In the domain of first-order theorem proving, recently so-called instantiation-based methods have gained some attention as they promise advantages over longer established methods. Although modern instantiation-based methods for first-order logic were proposed only in this decade, they are already competitive with and in some areas outperform classical methods that have a history of development since the 1970s. Because of the novelty of instantiation-based methods, not many results exist for equational reasoning and reasoning modulo theories. An integration of these features would make a calculus more mature and open up new areas of application. The main subject of this research is the Inst-Gen calculus as initially presented in Ganzinger and Korovin (2003).

The central rationale behind the Inst-Gen calculus and its implementation in the iProver system is the combination of industrial-strength ground satisfiability solving with true first-order reasoning. In particular, the iProver approach aims to harness the strengths of an off-the-shelf ground satisfiability solver in a first-order procedure for satisfiability modulo theories (SMT). The lack of complete handling of quantification is a shortcoming of many SMT provers and their limited support of quantified clauses makes them essentially reason on ground satisfiability. However, they are efficient in ground solving and iProver therefore delegates detecting unsatisfiability to a ground SMT solver and focuses on generating instances from first-order clauses such that ground unsatisfiability can be witnessed.

The work presented here is concerned with efficient integration of ground solving modulo equality into the first-order instantiation process of Inst-Gen. So to speak, the first-order reasoning needs to stand firmly on the ground, but robust integration of ground reasoning also guides the instantiation process through provision of satisfiable models and is used to justify simplifications.

2 The Inst-Gen Calculus

Formally, the set of first-order clauses S is abstracted to a set of ground clauses $S\bot$ by substituting all variables with a distinguished constant $\bot$ with a substitution that is also denoted by $\bot$. If this ground abstraction $S\bot$ is unsatisfiable, then, by Herbrand's Theorem, S is unsatisfiable as well and the procedure can terminate. In the case of satisfiability of $S\bot$, the first-order clause set S is satisfiable if it is saturated under instantiation inferences of the calculus. Otherwise, one needs to continue generating instances either as witnesses for the unsatisfiability of $S\bot$ or to reach saturation.

If the ground abstraction $S\bot$ has been proved to be satisfiable by the ground solver, the first-order reasoning can make use of a ground model $I_\bot$ for $S\bot$ by means of a selection function sel that constrains possible inferences between clauses to their respective selected literals. Although Inst-Gen inferences are sound regardless of the selection function, the selection function adds an amount of goal-direction that is necessary to be efficient in practice. In each clause C, the selection $\mathrm{sel}(C) = L$ returns a literal L in C such that the ground literal $L\bot$ is true in $I_\bot$. See Korovin (2009) for a detailed description of the calculus, especially for arguments of its completeness.

It is important to note that literal selection in the Inst-Gen calculus is different from literal selection in resolution, as described e.g. in Bachmair and Ganzinger (2001). The selection function does select exactly one literal from every clause, but is not fixed throughout the derivation. It may need to be adapted to reflect changes of the ground model $I_\bot$ when instances are added to S.

3 The Saturation Process, Ground Solving and Selection

The implementation of the iProver system as described in Korovin (2008) uses a variant of the given-clause algorithm. It keeps two disjoint sets of *active clauses* $\mathcal{A}$ and *passive clauses* $\mathcal{P}$ where all inferences between clauses in $\mathcal{A}$ have been generated. In every step, a clause C_g (the *given clause*) is taken from $\mathcal{P}$, all possible inferences with clauses in $\mathcal{A}$ are added to $\mathcal{P}$ and C_g is moved from $\mathcal{P}$ to $\mathcal{A}$. If $\mathcal{P}$ is empty, then all clauses are in $\mathcal{A}$ and it is saturated, which establishes satisfiability of S. Separately, all clauses in $\mathcal{A}$ and $\mathcal{P}$ are grounded with the $\bot$ constant and checked for unsatisfiability in the ground solver. The procedure then either terminates with unsatisfiable as a result or obtains a model $I_\bot$ to be used for the selection function.

The model $I_\bot$ is given by a ground solver that may choose to discard as much of the model from the previous step as it sees fit, therefore giving rise to an uncontrollable number of changes to the selection function. When $\mathrm{sel}(C)$ for a clause C in the active set $\mathcal{A}$ changes, the clause C has to be moved to the passive set $\mathcal{P}$ as with a different selection new inferences may become possible that have not been generated, violating the invariant of saturation of $\mathcal{A}$ under inferences.

Therefore, it seems beneficial to attempt to keep the selection in the active clause set $\mathcal{A}$ and only to change it when necessary, i.e. when there is no model of $S\bot$ containing the selected literals $\bigcup_{C\in\mathcal{A}} \mathrm{sel}(C)\bot$. This lazy strategy allows for a deviation from the particular model the ground solver provides by maintaining a separate model that follows the literal selection and the heuristics involved while still being sound and complete in the case of clauses without equality.

With equality or other background theories, induced equalities have to be considered on potentially many selected literals, making it seem inefficient to check if a set of selected literals can be extended to a model. Possible alternative strategies for selection would either eagerly follow the model as it is obtained from the ground solver, thus weakening the selection heuristics and causing more frequent moves of clauses from the active set. Alternatively, the incompleteness of checking if the selected literals are a model could be accepted and deferred until the active set becomes saturated. Then, a full check on all selected literals would either bring up inconsistent literals and subsequently move clauses or find the selected literals to be consistent and thus prove satisfiability.

4 Results and Future Work

Which strategy is most successful will certainly depend on the solver, its strategies and the amount of cooperation, and, of course, on the clause set itself. Ongoing work is evaluating CVC3 and Z3, two leading SMT solvers, as ground satisfiability solvers modulo equality and modulo theories as well as the most successful strategies for literal selection as described above. Early experiments with the problems of the TPTP library show that the lazy strategy leaves only relatively few problems with an inconsistent set of selected literals on saturation. Moreover, in most cases the problem is proved satisfiable with a literal selection that is a model for $S\bot$ after only one or two further checks, indicating that the lazy strategy might be viable for ground solving modulo equality and theories.

Comparing a version of iProver using MiniSAT for ground solving without equality with an early implementation integrating CVC3 as a ground solver modulo equality on the TPTP library (detailed figures in table 1) show that on the one hand, there is a significant overhead for the SMT solving such that SAT solving is still faster in many cases. On the other hand, there are cases where ground equational reasoning is profitable and the overhead pays off such that additional problems can be solved. As expected, ground solving with equality is most useful for pure equational problems.

Equational	Num.	only in iProver +		faster in iProver +	
atoms	Probl.	CVC3	MiniSAT	CVC3	MiniSAT
0%	3130	6	346	522	1736
0% - 10%	1915	24	118	54	449
10% - 20%	2622	49	76	205	408
20% - 30%	1593	15	44	47	336
		⋮			
100%	1515	205	26	85	100

Table 1: Solved problems in TPTP v3.5.0 grouped by percentage of equational atoms. No significant number of problems available with between 30% and 100% of equational atoms. Run on AMD Athlon XP 2200+ with 500 MB and a timeout of 3 mins.

Future work will use the good grip the iProver system is getting on ground solving modulo equations and theories to progress with instance generation of first-order clauses modulo equality and theories where lifting the ground model to obtain a literal selection is even more important and thus help iProver climb up to the important application areas of satisfiability modulo theories.

Acknowledgements

Joint work with Konstantin Korovin and Renate Schmidt, based on the iProver system, as in described in Korovin (2008).

References

Leo Bachmair and Harald Ganzinger. Resolution Theorem Proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume 1. Elsevier Science and MIT Press, 2001.

Harald Ganzinger and Konstantin Korovin. New Directions in Instantiation-Based Theorem Proving. In *Symposium on Logic in Computer Science, LICS 2003. Proceedings*, pages 55–64, 2003.

Konstantin Korovin. An Invitation to Instantiation-Based Reasoning: From Theory to Practice. In A. Podelski, A. Voronkov, and R. Wilhelm, editors, *Volume in memoriam of Harald Ganzinger*, Lecture Notes in Computer Science. Springer, 2009. Invited paper. To appear.

Konstantin Korovin. iProver - An Instantiation-Based Theorem Prover for First-Order Logic (System Description). In *International Joint Conference on Automated Reasoning, IJCAR 2008. Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 292–298. Springer Berlin / Heidelberg, 2008.