# Tuning the Alt-Ergo SMT Solver for B Proof Obligations

Sylvain Conchon[1,2] and Mohamed Iguernelala[3,1]

[1] LRI, Université Paris-Sud, 91405 Orsay, France
[2] INRIA Saclay – Ile-de-France, Toccata, 91893 Orsay, France
[3] OCamlPro SAS, 91190 Gif-sur-Yvette, France

**Abstract.** In this paper, we present recent developments in the Alt-Ergo SMT-solver to efficiently discharge proof obligations (POs) generated by Atelier B. This includes a new plugin architecture to facilitate experiments with different SAT engines, new heuristics to handle quantified formulas, and important modifications in its internal data structures to boost performances of core decision procedures. Experiments realized on more than 10,000 POs generated from industrial B projects show significant improvements.

**Keywords:** SMT solvers, B Proof Obligations, B Method.

## 1 The Alt-Ergo SMT Solver

Alt-Ergo is an open-source SMT solver capable of reasoning in a combination of several built-in theories such as uninterpreted equality, integer and rational arithmetic, arrays, records, enumerated data types and AC symbols. It is the unique SMT solver that natively handles polymorphic first-order quantified formulas, which makes it particularly suitable for program verification. For instance, Alt-Ergo is used as a back-end of SPARK and Frama-C to discharge proof obligations generated from Ada and C programs, respectively.

Recently, we started using Alt-Ergo in the context of the ANR project BWare [8] which aims at integrating SMT solvers as back-ends of Atelier B. The proof obligations sent to Alt-Ergo are extracted from Atelier B as logical formulas that are combined with a (polymorphic) model of B's set theory [7]. This process relies on the Why3 platform [6] which can target a wide range of SMT solvers. However, we show (Section 2) on a large benchmark of industrial B projects that it is not immediate to obtain a substantial gain of performances by using SMT solvers. Without a specific tunning for B, Alt-Ergo together with other SMT solvers compete just equally with Atelier B's prover on those industrial benchmarks.

In this paper, we report on recent developments in Alt-Ergo that significantly improve its capacities to handle POs coming from Atelier B. Our improvements are: (1) better heuristics for instantiating polymorphic quantified formulas from B model; (2) new efficient internal data structures; (3) a plugin architecture to facilitate experiments with different SAT engines; and (4) the implementation of a new CDCL-based SAT solver.

## 2 Benchmarks

Currently, the test-suite of BWare contains 10572 formulas[1] obtained from three industrial B projects provided by ClearSy. The first one, called DAB, is an automated teller machine (ATM). The last two, called P4 and P9, are obfuscated and unsourced programs.

The formulas generated from these projects are composed of two parts. The first one is the *context*: a large set of axioms (universally and polymorphic quantified formulas). This part contains the B's set theory model, as well as huge (in size) predicates describing the B state machines. The second part of each PO is the *goal*: a ground formula involving these predicates (see [7] for detailed explanations about the structure of these POs).

A quick look to the shape of these formulas shows that they mainly contain equalities over uninterpreted function symbols and atoms involving enumerated data types. Only a small portion of atoms contains linear integer arithmetic and polymorphic records. In comparison with our other benchmarks coming from deductive program verification platforms, the average number of axioms, as well as the size of the POs are much larger in this test suite, as shown below:

| | number of POs | avg. number of axioms | avg. size (Ko) |
|---|---|---|---|
| VSTTE-Comp | 125 | 32 | 8 |
| Why3's gallery | 1920 | 41 | 9 |
| Hi-Lite | 3431 | 125 | 23 |
| DAB | 860 | *257* | 236 |
| P4 | 9341 | *259* | 248 |
| P9 | 371 | *284* | 402 |

At the beginning of the project and without specific improvements, we ran some SMT solvers (z3, cvc3, and Alt-Ergo) on this test-suite. All measures were obtained on a 64-bit machine with a quad-core Intel Xeon processor at 3.2 GHz and 24 GB of memory. Solvers were given a time limit of 60 seconds and a memory limit of 2 GB for each PO. The results of our experiments are reported in the following table.

| Provers<br>Versions | Alt-Ergo<br>*0.95.1* | Alt-Ergo<br>*0.95.2* | z3<br>*4.3.1* | cvc3<br>*2.4.1* |
|---|---|---|---|---|
| DAB | 707<br>82.2 % | **822**<br>**95.6 %** | 716<br>83.3% | 684<br>79.5 % |
| P4 | 4709<br>50.4 % | **8402**<br>**89.9 %** | 7974<br>85.4 % | 7981<br>85.4 % |
| P9 | 181<br>48.8 % | **213**<br>**57.4 %** | 162<br>43.7 % | 108<br>29.1 % |
| Total | 5597<br>52.9 % | **9437**<br>**89.3 %** | 8852<br>83.7 % | 8773<br>83.0 % |

[1] These benchmarks will be available for the community at the end of the project.

For every solver, we report the number (and the percentage) of solved POs for each project. Except for Alt-Ergo *0.95.1*, the other versions of SMT solvers compete equally. From what we know from our BWare partners, these results are similar to those obtained by the prover of Atelier B 4.0, which proves **84%** of this test-suite. Concerning Alt-Ergo, the low rate obtained with version *0.95.1* is due to a quantifier instantiation heuristic that disabled the use of large axioms and predicates during proof search. This was unfortunate for the BWare benchmark (especially in P4 and P9) since goals mainly involve large predicates, as explained above. The minor release *0.95.2* mainly relaxes this (misguided) heuristic.

## 3 Improvements

Profiling Alt-Ergo on this test-suite shows: (1) a large number of axiom instanciations; (2) a high activity of the congruence closure decision procedure; and (3) an important workload for the SAT engine. A more thorough investigation shows that (1) is due to some *administrative* axioms of the B model that represent properties of basic set theoretic operations (AC properties, transitive closures, etc). The structure of theses instances mixes Boolean operators and equalities over uninterpreted terms, which explains the behavior (2) and (3).

One of our main objective was to efficiently handle such axioms by limiting the number of their instances. For that, we added the possibility of disabling the generation of new instances modulo known ground equalities. We also modified the default behavior of the matching algorithm to only consider terms that are in the active branch of the SAT engine. Finally, we have reimplemented literal representation and some parts of the `Formula` module of Alt-Ergo to enable trivial contextual simplifications, and thus to identify equivalent formulas. In addition, we have modified the core architecture of Alt-Ergo to facilitate the use of different SAT-solvers, implemented as plugins.

The results in the following table show the impact of our optimizations. The *master branch* version of Alt-Ergo contains all the modifications described above. As we can see, this version outperforms *0.95.2*. The last column contains the result of a wrapper, called Ctrl-Alt-Ergo, that uses the time given to the solver to try different strategies and heuristics[2].

| Versions | Alt-Ergo *0.95.2* | Alt-Ergo *master branch* | Ctrl-Alt-Ergo *master branch* |
|---|---|---|---|
| DAB | 822<br>95.6 % | 858<br>99.8 % | **860**<br>**100 %** |
| P4 | 8402<br>89.9 % | 8980<br>96.1 % | 9236<br>**98.9 %** |
| P9 | 213<br>57.4 % | 234<br>63.1 % | 277<br>**74.7 %** |
| Total | 9437<br>89.3 % | 10072<br>95.3 % | **10373**<br>**98.1 %** |

[2] All these improvements will be available in future public releases of Alt-Ergo at `http://alt-ergo.ocamlpro.com`

## 4 Conclusion and Future Works

As demonstrated by our experiments, the results of our first investigations and optimizations are promising. It turns out that B proof obligations have some specificities that should be taken into account to obtain a good success rate.

In the near future, we want to study how to extend the core of Alt-Ergo to handle administrative axioms (definition of union and intersection, AC properties, etc.) as conditional rewriting rules. For that, we are studying the design of a new combination algorithm which will extend our parametrized algorithms AC(X) [2] and CC(X) [3] to handle, in a uniform way, user-defined rewriting systems. This would allow us to handle a fragment of the set theory of B in a built-in way.

We also plan to investigate whether *deduction modulo* techniques, like those in Zenon Modulo [4,5] and iProver Modulo [1], would be helpful to efficiently handle quantified formulas in the SMT world. Another line of research would be the generation of verifiable traces for external proof checkers in order to augment our confidence in the SMT solver.

## References

1. Burel, G.: Experimenting with Deduction Modulo. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS (LNAI), vol. 6803, pp. 162–176. Springer, Heidelberg (2011)
2. Conchon, S., Contejean, E., Iguernelala, M.: Canonized Rewriting and Ground AC Completion Modulo Shostak Theories: Design and Implementation. Logical Methods in Computer Science 8(3) (2012)
3. Conchon, S., Contejean, E., Kanig, J., Lescuyer, S.: CC(X): Semantic Combination of Congruence Closure with Solvable Theories. Electronic Notes in Theoretical Computer Science 198(2), 51–69 (2008)
4. Delahaye, D., Doligez, D., Gilbert, F., Halmagrand, P., Hermant, O.: Proof Certification in Zenon Modulo: When Achilles Uses Deduction Modulo to Outrun the Tortoise with Shorter Steps. In: International Workshop on the Implementation of Logics (IWIL), Stellenbosch (South Africa). EasyChair (December 2013) (to appear)
5. Delahaye, D., Doligez, D., Gilbert, F., Halmagrand, P., Hermant, O.: Zenon Modulo: When Achilles Outruns the Tortoise Using Deduction Modulo. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR-19. LNCS, vol. 8312, pp. 274–290. Springer, Heidelberg (2013)
6. Filliâtre, J.-C., Paskevich, A.: Why3 — Where Programs Meet Provers. In: Felleisen, M., Gardner, P. (eds.) ESOP 2013. LNCS, vol. 7792, pp. 125–128. Springer, Heidelberg (2013)
7. Mentré, D., Marché, C., Filliâtre, J.-C., Asuka, M.: Discharging Proof Obligations from Atelier B Using Multiple Automated Provers. In: Derrick, J., Fitzgerald, J., Gnesi, S., Khurshid, S., Leuschel, M., Reeves, S., Riccobene, E. (eds.) ABZ 2012. LNCS, vol. 7316, pp. 238–251. Springer, Heidelberg (2012)
8. The Bware Project (2012), http://bware.lri.fr/