Validating MAX-SMT Solvers

June 19, 2023

Problem description

MAX-SMT solvers (such as νZ [1] from Z3 [4]) are used in various optimization problems, e.g., scheduling, choosing the best strategy for a player within a game, minimizing the manufacturing or the transport cost of a product, etc.

Given an input formula consisting of *hard constraints* (which have to hold) and of *weighted soft constraints* (which may hold), the scope of a MAX-SMT solver is:

1. to determine if the hard constraints hold (i.e., are satisfiable)
2. if they hold, then to also identify a subset of the soft constraints which are satisfiable together with the hard constraints, such that the sum of the corresponding weights is maximized. To identify them, MAX-SMT solvers rely on complex algorithms, thus implementing them correctly is very challenging.

The goal of this project is to automatically validate the results produced by MAX-SMT solvers, by generating input formulas that are satisfiable or unsatisfiable by construction; for satisfiable formulas, we will also consider as part of the test oracle (that is, of the expected output) the maximum sum of the weights of the satisfiable soft constraints. We focus on formulas with known ground truth, as they allow us to identify *soundness* bugs. These are the most severe types of bugs and occur when the solver provides a wrong answer (i.e., returns *sat* for an unsatisfiable formula or *unsat* for a satisfiable one) or when it constructs an *incorrect model* (i.e., when it assigns incorrect values to the free variables).

Example

Let us consider the formula from Listing 1, where a is a Boolean variable. It consists of three soft constraints (there are no hard constraints), with the weights 4, 2, and 3, respectively. The first constraint can never hold, while the last two are unsatisfiable together. The maximum sum of the weights of the satisfiable constraints is 3 and can be achieved if $\neg a = true$, i.e., $a = false$. However, Z3's MAX-SMT solver from January 2016 unsoundly[1] generated the model $a = true$.

[1] https://github.com/Z3Prover/z3/issues/425

Listing 1: MAX-SMT formula that exposed an unsoundness in Z3

```
assert-soft(false, weight=4)
assert-soft(a, weight=2)
assert-soft(¬a, weight=3)
```

Approach

We will use as a starting point the approach described in [3], which presents a solution for generating formulas with known ground truth from the string theory. It first constructs simple satisfiable and unsatisfiable formulas and then it creates more complex ones by applying satisfiability-preserving transformations.

However, all these formulas include only hard constraints, so in a first step, we will extend the approach to encode the simple formulas as hard constraints and the transformations as soft constraints, such that the truth value remains unchanged (following, for example, the ideas from [2] Section 3.5.3). Then, we will explore ways of generating formulas consisting only of soft constraints, which are either unsatisfiable or have, by construction, a unique optimal solution. Moreover, we will further extend our technique to also generate formulas with multiple *objectives*.

In our example from Listing 1, all the constraints were implicitly part of the same objective. However, if we rewrite it as shown in Listing 2, where both a and b are Boolean variables, the solver has to maximize two objectives: obj_1, consisting of the first two soft constraints and obj_2 consisting only of the third constraint. In this case, the maximum possible sum per objective is 2 for obj_1, when $b = true$, and 3 for obj_2, when $a = false$.

Listing 2: Multi-objective MAX-SMT formula

```
assert-soft(false, weight=4, id=obj1)
assert-soft(b, weight=2, id=obj1)
assert-soft(¬a, weight=3, id=obj2)
```

For this second example from Listing 2, the sets of variables used in the two objectives are disjoint ($\{b\}$ in obj_1 and $\{a\}$ in obj_2), but realistic optimization problems include objectives with overlapping sets of variables. Therefore, we will also address the more challenging problem of constructing formulas with multiple objectives over shared variables that still have a known ground truth.

We will evaluate our approach on known bugs from Z3's MAX-SMT solver. As possible extensions, we will explore how our technique can be adapted to support different solving strategies for multiple objectives (e.g., Pareto fronts instead of the default, lexicographic combinations [1]).

Prerequisites

The student is expected to have good programming skills. Prior experience with SMT solvers is a plus.

Opportunities

The student will have the chance to gain a deep understanding of MAX-SMT solvers and to learn about state-of-the-art testing techniques.

Contact

Alexandra Bugariu: bugariua@mpi-sws.org

References

[1] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. νZ - an optimizing SMT solver. In *TACAS*, 2015.

[2] Alexandra Bugariu. *Automatically Identifying Soundness and Completeness Errors in Program Analysis Tools.* PhD thesis, ETH Zürich, 2022.

[3] Alexandra Bugariu and Peter Müller. Automatically testing string solvers. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1459–1470, 2020.

[4] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.