νZ - Maximal Satisfaction with Z3

Nikolaj Bjørner\(^1\) and Anh-Dung Phan\(^2\)∗

\(^1\) Microsoft Research nbjorner@microsoft.com
\(^2\) Technical University of Denmark padu@dtu.dk

Abstract

Satisfiability Modulo Theories, SMT, solvers are used in many applications. These applications benefit from the power of tuned and scalable theorem proving technologies for supported logics and specialized theory solvers. SMT solvers are primarily used to determine whether formulas are satisfiable. Furthermore, when formulas are satisfiable, many applications need models that assign values to free variables. Yet, in many cases arbitrary assignments are insufficient, and what is really needed is an optimal assignment with respect to objective functions. So far, users of Z3, an SMT solver from Microsoft Research, build custom loops to achieve objective values. This is no longer necessary with νZ (new-Z, or max-Z), an extension within Z3 that lets users formulate objective functions directly with Z3. Under the hood there is a portfolio of approaches for solving linear optimization problems over SMT formulas, MaxSMT, and their combinations. Objective functions are combined as either Pareto fronts, lexicographically, or each objective is optimized independently.

1 SMT and Optimization

SMT solvers have enjoyed a decade of significant impact for many applications in program analysis, verification, testing and to some extent synthesis. A common premise for the use of SMT solvers in these applications have been that logic serves as a suitable calculus of computation. In other words, at the core of most symbolic program analysis/testing/verification engines, there is a natural reduction to a logical form, and it makes better sense to use a common well-tuned engine than hand-roll a custom solver. Key enabling factors have been technological advances with core search algorithms coupled with built-in support for theories that are typical to programs, such as reasoning with bit-vectors, arithmetic, applicative stores (arrays), algebraic data-types to mention a few. The ability to also handle quantified formulas, and even, in many cases, extend to decision procedures for quantified formulas has furthered the scope of areas where SMT solvers can be used.

Yet, knowing whether a logical formula is satisfiable or not is not always sufficient. In particular, being able to state and solve optimization objectives in the context of logical constraints has been well recognized in the SMT community \([1,3,15,5,4,9]\) and it is a recurring feature request for Z3 \([7]\) as well. We therefore created νZ as a new toy for Z3 users and as a service to users who do not wish to maintain their own optimization algorithms on top of Z3.

We here describe optimization features in Z3 in their current state. We briefly touch on the tool usage, but our emphasis is on the algorithms we have implemented. A tool demonstration presentation is the subject of another paper.

∗νZ was initiated with Anh-Dung Phan during an internship at Microsoft Research and Microsoft Dynamics. Thanks to Lars Fleckenstein and his group for collaborating on applications of νZ.
1.1 Stating objectives in SMT-LIB

The easiest way to get acquainted with optimization in Z3 is by trying it out online at the tutorial [http://rise4fun.com/z3opt/tutorial/](http://rise4fun.com/z3opt/tutorial/). The examples used here use the SMT-LIB2 syntax and extend it with a few primitives:

- \((\text{maximize } t)\) - instruct the solver to maximize \(t\).
- \((\text{minimize } t)\) - instruct the solver to minimize \(t\).
- \((\text{assert-soft } F :\text{weight } n)\) - assert soft constraint \(F\), optionally with weight \(n\). If no weight is given then the default weight is 1.

In \(\nu Z\), the type of the term \(t\) can be either Integer, Real or Bit-vector. If \(t\) has type bit-vector with, say width 4, then \((\text{maximize } t)\) is equivalent to the four soft assertions:

\[(\text{assert-soft } (= t[3:3] 1) :\text{weight } 8)\]  
\[(\text{assert-soft } (= t[2:2] 1) :\text{weight } 4)\]  
\[(\text{assert-soft } (= t[1:1] 1) :\text{weight } 2)\]  
\[(\text{assert-soft } (= t[0:0] 1) :\text{weight } 1)\]

For example, we can ask to optimize the term \(x+y\) under the constraints \(x < 2\) and \(y-x < 1\).

\[(\text{declare-const x Int})\]  
\[(\text{declare-const y Int})\]  
\[(\text{assert } (< x 2))\]  
\[(\text{assert } (< (- y x) 1))\]  
\[(\text{maximize } (+ x y))\]  
\[(\text{check-sat})\]

The optimal answer is given as 2 and \(\nu Z\) returns a model where \(x = y = 1\).

1.2 \(\nu Z\) Capabilities

Of course, the more exciting things happen under the hood. \(\nu Z\) comprises of two main components: (1) a MaxSMT (in reality it is a collection of MaxSAT solvers) module that solves soft constraints and (2) an OptSMT module that optimizes linear arithmetic objective functions. The two modules are controlled by an upper layer that translates optimization objectives into either of these modules. The upper layer also invokes the MaxSMT and OptSMT components in a suitable combination if there are multiple objectives.

To summarize, \(\nu Z\) allows to maximize or minimize terms over integers, reals and bit-vectors. Weighted constraints over Booleans can be entered as weighted soft constraints, or as a summation. Multiple objectives can be combined using lexicographic, Pareto fronts or as independent objectives. We call the last combination box priorities. The next sections describe the MaxSMT and OptSMT components in more detail.

2 Weighted MaxSMT

Weighted MaxSMT is the following problem. Given a set of numeric weights \(w_1, \ldots, w_n\) and formulas \(F_0, F_1, \ldots, F_n\), find the subset \(I \subseteq \{1, \ldots, n\}\) such that

1. \(F_0 \land \bigwedge_{i \in I} F_i\) is satisfiable.
2. The award \(\Sigma_{i \in I} w_i\) is maximized. Dually, the cost \(\Sigma_{i \notin I} w_i\) is minimized.

In other words, the weight \(w_i\) encodes the award for including \(F_i\) in the satisfying assignment, or dually, the penalty for a formula \(F_i\) to be excluded from a satisfying assignment.
2.1 WMax

We here recall from [1] an approach first proposed in [13]. In their solver, an important point is that the theory evolves as search progresses: once a satisfiable state is reached with a given cost $c$, then assignments that meet or exceed $c$ are useless. Their encoding is as follows: Initially we assert $F_0$ and $F_i ∨ p_i$ for each $i$, where $p_i$ is a fresh propositional variable. We also maintain a cost $c$ that is initialized to 0, and a min_cost that is set to nil. Then, repeat the following steps until the asserted formulas are unsatisfiable:

1. When some $p_i$ is assigned to true, then update $c ← c + w_i$.

2. If nil $\neq$ min_cost $\leq c$, then block the current state by adding the clause

$$\bigvee \{-p_i \mid p_i \text{ is assigned to true}\}.$$

3. When all $p_i$ are assigned to true or false without exceeding the minimal cost, it must be that $c <$ min_cost or min_cost is nil. So it is safe to set min_cost $← c$. To block this current cost, add the assertion

$$\bigvee \{-p_i \mid p_i \text{ is assigned to true}\}.$$

The approach is implemented as a satellite theory solver in Z3’s SMT core. It inter-operates with the other theories in this core: linear arithmetic, bit-vectors, arrays, and algebraic data-types. The stand-alone solver for non-linear polynomial arithmetic over the reals is not part of this core. This basic approach to MaxSMT, using a satellite solver works remarkably well in many cases. It even has the advantage of always producing better approximations of the optimal values. However, it falls flat on its face in most large scale benchmark applications circulating in the MaxSAT community. $\nu Z$ therefore implements also an alternative based on Maximal Resolution.

2.2 MaxRes

A very efficient weighted MaxSAT solving method based on Maximal Resolution (MaxRes) [2] was recently developed by Nina Narodytska and Fahiem Bacchus [12]. Their solver won the MaxSAT competition for 2014 by a noticeable margin. MaxRes is a core-based method. In contrast, WMax narrows satisfying assignments. We will here rephrase the main result from [12], but without appealing to maximal resolution. We show soundness of the main step of the algorithm using a direct argument. To keep the treatment simple, we consider just unit weights. The technique used in other core-based MaxSAT solvers applies to handle weighted constraints: Given a core $F_1, \ldots, F_k$ with weights $w_1, \ldots, w_k$, let $w_0 = \min(w_1, \ldots, w_k)$. Create the new soft constraints $F_1 : w_0, \ldots, F_k : w_0, F_1 : w_1 - w_0, \ldots, F_k : w_k - w_0$, and remove constraints with weight 0 (there is at least one).

We are given a formula $F$ and a set of soft constraints $F_1, \ldots, F_n$. Let $F_1, \ldots, F_k$ be a subset of soft constraints (with unit weights) that are inconsistent with $F$. For the next iteration we
produce the new set \( F', F'_1, F'_2, \ldots, F'_{k-1}, F_{k+1}, \ldots, F_n \), where\(^1\):

\[
\begin{align*}
F' &\leftarrow F \land (\neg F_1 \lor \neg F_2 \lor \ldots \lor \neg F_k) \\
F'_1 &\leftarrow F_2 \lor F_1 \\
F'_2 &\leftarrow F_3 \lor (F_1 \land F_2) \\
\vdots \\
F'_{k-1} &\leftarrow F_k \lor ((F_1 \land F_2) \land \ldots \land F_{k-1})
\end{align*}
\]

The process eventually terminates. It either ends up with a set where all the soft constraints can be satisfied, or it ends up with the empty set of soft constraints, e.g., none of the soft constraints could be satisfied. We can reproduce the optimal assignment from either of these cases thanks to the following property:

**Claim 1.** \( F, F_1, \ldots, F_n \) has a maximal satisfying assignment of weight \( w \) if and only if \( F', F'_1, \ldots, F'_{k-1}, F_{k+1}, \ldots, F_n \) has a maximal satisfying assignment of weight \( w - 1 \).

**Proof.** Let \( M \) be an arbitrary truth assignment to \( F_1, \ldots, F_k \), assuming \( F_1, \ldots, F_k \) are mutually inconsistent (is a core). Consider two cases:

1. \( M(F_i) = \text{false} \). Then we claim \( M(F'_i) = M(F_{i+1}) \) for \( i = 1, \ldots, k - 1 \).

2. \( M(F_i) = \text{true} \), and let \( j \) be the first index where \( M(F_{j+1}) = \text{false} \). Then we claim that \( M(F'_j) = M(F_{j+1}) \) for \( i = 1, \ldots, j - 1, j + 1, \ldots, k - 1 \), \( M(F'_j) = \text{true} \).

In both cases the weight of the new set of clauses is one less than the previous weight. If \( F_1, \ldots, F_k \) is a core, then at least one truth assignment has to be false. \( \square \)

An important observation in \[12\] is to control the number of new formulas that are created in each round. In our setting it amounts to the following observation: There are a linear number of different sub-formulas in the definition of \( F'_1, \ldots, F'_{k-1} \). When performing clasification, it helps to exploit this and introduce auxiliary names for linear number of sub-formulas. For example \( F_1 \land F_2 \) is represented using a name \( d_1 \):

\[
\begin{align*}
F' &\leftarrow F \land (\neg F_1 \lor \neg F_2 \lor \ldots \lor \neg F_k) \land \\
&\quad (d_1 \rightarrow F_1) \land (d_1 \rightarrow F_2) \land \\
&\quad (d_2 \rightarrow d_1) \land (d_2 \rightarrow F_3) \land \ldots \\
F'_1 &\leftarrow F_2 \lor F_1 \\
F'_2 &\leftarrow F_3 \lor d_1 \\
\vdots \\
F'_{k-1} &\leftarrow F_k \lor d_{k-1}
\end{align*}
\]

\( \nu Z \) shows significantly better performance than \( WMax \) on many larger instances. In \( \nu Z \), we furthermore have the option to use a dedicated SAT solver for SMT formulas that use only propositional logic, bit-vectors and 0-1 linear variables. There is a very cute duality to \( \text{MaxRes} \) based on correction sets (instead of cores). Given a correction set \( F_1, \ldots, F_k \), we can obtain a smaller problem by dualizing the definitions for \( F'_1, \ldots, F'_{k-1} \) (interchange \( \lor \) with \( \land \)). We are currently investigating ways to leverage this duality with Nina Narodytska.

\( \nu Z \) also implements a number of other algorithms for \( \text{MaxSAT} \), including \( \text{BCD2} \) [11] which is based on a binary search over upper and lower bounds, and a basic version of \( \text{MaxHS} \) [6] which is based on finding hitting sets.

\(^1\)Since the conjunction \( F_1 \land F_2 \land \ldots \land F_k \) is inconsistent with \( F \), it follows that \( F \) implies the clause \( (\neg F_1 \lor \neg F_2 \lor \ldots \lor \neg F_k) \). Nevertheless, we add this (redundant) clause to \( F \) as it can participate in propagation.
Input: Objective $t$ to maximize
Input: Formula $F$
Output: Maximal value $v$, such that $v = t \land F$ is satisfiable

$v \leftarrow -\infty$

while $F$ is satisfiable do
    Let $L$ be a set of literals (from $F$) that imply $F$.
    if $t$ is unbounded in $L$ then
        return $\infty$
    end
    Let $M$ be an interpretation that satisfies $L$ and maximizes $t$
    $v \leftarrow \max(v, M(t))$
    $F \leftarrow F \land t > v \land \neg \bigwedge L$
end

return $v$

Algorithm 1: Sequential Bound Increase. During each iteration we have the choice to add either $t > v$ or $\neg \bigwedge L$, or both to force convergence. For the case of linear difference logic, $\nu Z$ allows general linear objective functions. It uses primal Simplex (and not yet Network Simplex) to maximize $t$ under $L$. The inequality $t > v$ may however not be expressible in difference logic. Instead $\nu Z$ uses the assertion $\neg \bigwedge L$ to ensure convergence.

3 $\nu Z$ for Linear Arithmetic

We have augmented the theory solvers for (integer) linear arithmetic and (integer) difference logic with a primal Simplex phase. Z3 contained for some time an internal implementation for primal Simplex. It is used for guiding search while solving integer linear and non-linear problems. This hidden feature was “discovered” and exploited in [10]. We here mainly discuss the variant we use in $\nu Z$.

3.1 Basic Arithmetic Optimization

The main idea used in [15, 10] is, for a quantifier free formulas $F$, extract a set of literals $L$ that implies $F$ (an implicant, preferably prime), and for this set of literals compute a local optimum. One can now search for the next implicant of $F$ that improves the current local optimum. If the set of literals $L$ is taken from linear arithmetic inequalities, then the local optimum can be found using primal Simplex. Algorithm [1] illustrates this approach. $\nu Z$ implements Algorithm [1] for both linear real and linear integer, and mixed integer/real arithmetic. It furthermore, as the caption to Algorithm [1] advertises, implements this loop for the specialized difference logic solvers. We note, however, that the performance of the difference logic solvers is not always better than the Simplex solver even on problems that are in the difference logic domain.

3.2 Unbounded Objectives

One of the main techniques introduced in [10] is to discover (multiple) unbounded objectives in one call without iterating over potentially many solutions $L$. It uses a clever geometric argument to discover unbounded objectives by considering how hyper-planes can bound objectives. In $\nu Z$ we provide an alternative option for finding unbounded objectives using a single SMT call. The technique extends the existing use of non-standard rational numbers in Z3 [8] (and most other SMT solvers). The usual technique is to allow variables to take values of the form $b + \epsilon a$,
\begin{algorithm}
\textbf{Input}: Objective $t$ to maximize  \\
\textbf{Input}: Formula $F$  \\
\textbf{Output}: Maximal value $v$, such that $v = t \land F$ is satisfiable  \\
if $F \land t \geq \infty$ is satisfiable then  \\
\hspace{1em} return $\infty$  \\
end  \\
v $\leftarrow -\infty$  \\
while $F$ is satisfiable do  \\
\hspace{1em} Let $L$ be a set of literals (from $F$) that imply $F$.  \\
\hspace{1em} Let $M$ be an interpretation that satisfies $L$ and maximizes $t$  \\
\hspace{1em} $v \leftarrow \max(v, M(t))$  \\
\hspace{1em} $F \leftarrow F \land t > v \land \neg \bigwedge L$  \\
end  \\
return $v$  \\
\end{algorithm}

\textbf{Algorithm 2: Finding unbounded objectives using non-standard arithmetic}

where $\epsilon$ is treated as an infinitesimal and $a, b$ are rational numbers. This allows treating strict inequalities as non-strict inequalities. The extension used to discover unbounded objectives is to allow variables to take values of the form $c\infty + b + \epsilon a$, where $\infty$ is treated as a constant that is always larger than any finite constant. All standard Simplex operations can be performed directly for this number representation. Algorithm 2 illustrates our variant that uses a separate pass to detect unbounded objectives.

### 3.3 An Experiment with Core-based Optimization

In the case of MaxSMT, we were able to exploit core-based methods to find maximal assignments. One may wonder, is there any reasonable core-based method for linear arithmetic? At least we wondered and experimented with an approach sketched in Algorithm 3. The idea is to assert a lower bound $\text{mid} < t$ and use the outcome of the solver to either find a new improved value for $t$ (a new lower bound for the maximal value for $t$), or refine the upper bound for $t$. When $F \land \text{mid} < t$ is satisfiable, we can obtain a new improved lower bound as we did in the previous algorithms. If $F \land \text{mid} < t$ is unsatisfiable, we can improve the upper bound from inspecting the justifications for $\text{mid} < t$ being inconsistent with respect to $F$. The idea is that a proof of unsatisfiability produces a justification that separates $F$ from $\text{mid} < t$. When the justification is a set of clauses it produces a separating plane as a certificate. We can find the separating plane by appealing to Farkas lemma. Farkas lemma says that if a clause $(Ax \leq b \rightarrow t \leq \text{mid})$ is a tautology, then dually $Ax \leq b \land t < -\text{mid}$ is inconsistent, so there is a set of non-negative coefficients, $r_1, r_2$, such that

$$r_1 Ax - r_2 t \leq r_1 b - r_2 (\text{mid} + \epsilon) \quad \equiv \quad 1 \leq 0 .$$

Said in a different way: $r_1 Ax - r_1 b > r_2 t - r_2 \text{mid}$. Thus, since $\frac{r_1 Ax}{r_2} = t$, we get the tighter bound $\frac{r_1 b}{r_2} < \text{mid}$. Without loss of generality we can normalize $r_2$ to 1, so we just have to worry about $r_1$. There is in general more than one lemma used for separating $t$ from the bound $\text{mid}$. One has to take the maximal separator and Algorithm 3 illustrates the resulting approach for finding tighter bounds than $\text{mid}$.

Regrettably, our experience with an implementation of this idea has so far not been as good as the more straightforward approach from Algorithm 1. Z3 spends a lot of time refuting infeasible bounds on the instances we tried this algorithm on.
Input: Objective $t$ to maximize  
Input: Formula $F$  
Output: Maximal value $v$, such that $v = t \land F$ is satisfiable  

\[
\begin{align*}
\text{lo} & \leftarrow -\infty, \text{hi} \leftarrow \infty \\
\textbf{while } \text{lo} < \text{hi} & \textbf{ do} \\
& \quad \text{Pick mid such that lo} < \text{mid} < \text{hi} \\
& \quad \textbf{if } \text{mid} < t \land F \text{ is satisfiable } \textbf{then} \\
& \quad \quad \text{Let } M \text{ be an evaluation that satisfies } F \text{ and maximizes } t \\
& \quad \quad \text{lo} \leftarrow M(t) \\
& \quad \textbf{end} \\
& \quad \textbf{else} \\
& \quad \quad \text{Let } (A_i x \leq b_i \rightarrow t \leq \text{mid}) \text{ be T-lemmas for } i \in I \\
& \quad \quad \text{That is } F \rightarrow \bigvee_i A_i x \leq b_i \\
& \quad \quad \text{Let } r_i \text{ be Farkas coefficients for the T-lemmas, such that } r_i A_i > r_i b_i, r_i A_i = t \\
& \quad \quad \text{hi} \leftarrow \max\{r_i b_i \mid i \in I\} \\
& \quad \textbf{end} \\
\textbf{return hi}
\end{align*}
\]

Algorithm 3: Bisection Search with Farkas Lemmas

4 Combining Objectives

Multiple objectives can be combined using lexicographic, Pareto fronts or as independent box objectives. We briefly summarize these combination methods here.

Lexicographic combinations Suppose we are given two objectives $t_1, t_2$ to maximize subject to the constraint $F$. The lexicographic combination is to find model $M$, such that $M$ satisfies $F$ and the pair $(M(t_1), M(t_2))$ is lexicographically maximal. In other words, there is no model $M'$ of $F$, such that either $M'(t_1) > M(t_1)$ or $M'(t_1) = M(t_1), M'(t_2) > M(t_2)$.

Pareto fronts Again, given two objectives $t_1, t_2$, the set of Pareto fronts under $F$ are the set of models $M_1, \ldots, M_i, \ldots, M_k, \ldots$, such that either $M_i(t_1) > M_j(t_1)$ or $M_i(t_2) > M_j(t_2)$, and at the same time either $M_i(t_1) < M_j(t_1)$ or $M_i(t_2) < M_j(t_2)$; and for each $M_i$, there is no $M'$ that dominates $M_i$. $\nu Z$ uses the Guided Improvement Algorithm [14] to produce multiple objectives. We recall it in Algorithm [4].

Box objectives Finally, given two objectives $t_1, t_2$ finding maximal independent assignments to $t_1$ and $t_2$ can be accomplished simultaneously by slightly modifying algorithms from Section[3]. The resulting algorithm is shown in Algorithm [4].

5 Conclusion

We have provided some of the main algorithms used in $\nu Z$. The $\nu Z$ capabilities in Z3 are at the time of writing still in active development and improvements. Many new ideas have appeared in MaxSAT in recent years and the integration and extension of these ideas in the context of SMT is an exciting opportunity. On the other hand, $\nu Z$ allows users with somewhat richer theories
\[ \text{Input: Objectives } t_1, t_2 \text{ to maximize} \]
\[ \text{Input: Formula } F \]
\[ \text{Output: Pareto maximal front} \]
\[ \text{while } F \text{ is satisfiable do} \]
\[ G \leftarrow F \]
\[ \text{while } G \text{ is satisfiable do} \]
\[ \text{Let } L \text{ be a set of literals (from } G \text{) that imply } G. \]
\[ \text{if either } t_1 \text{ or } t_2 \text{ is unbounded in } L \text{ then} \]
\[ \text{return} \]
\[ \text{end} \]
\[ \text{Let } M \text{ be an interpretation that satisfies } L \text{ and maximizes } t_1 \text{ or } t_2 \]
\[ v_1 \leftarrow M(t_1), v_2 \leftarrow M(t_2) \]
\[ G \leftarrow G \land t_1 \geq v_1 \land t_2 \geq v_2 \land (t_1 > v_1 \lor t_2 > v_2) \]
\[ \text{end} \]
\[ F \leftarrow F \land (t_1 > v_1 \lor t_2 > v_2) \]
\[ \text{yield } \langle v_1, v_2 \rangle \]
\[ \text{end} \]
Algorithm 4: Guided Improvement Algorithm for finding Pareto fronts.

\[ \text{Input: Objectives } t_1, t_2 \text{ to maximize} \]
\[ \text{Input: Formula } F \]
\[ \text{Output: Box-maximal front} \]
\[ v_1 \leftarrow -\infty, v_2 \leftarrow -\infty \]
\[ \text{while } F \text{ is satisfiable do} \]
\[ \text{Let } L \text{ be a set of literals (from } F \text{) that imply } F. \]
\[ \text{Let } M \text{ be an interpretation that satisfies } L \text{ and maximizes } t_1, t_2 \]
\[ v_1 \leftarrow \max(v_1, M(t_1)) \]
\[ v_2 \leftarrow \max(v_2, M(t_2)) \]
\[ F \leftarrow F \land (t_1 > v_1 \lor t_2 > v_2) \land \neg \bigwedge L \]
\[ \text{end} \]
\[ \text{return } \langle v_1, v_2 \rangle \]
Algorithm 5: Finding a independent upper bounds for \( t_1, t_2 \). We here work models that can return non-standard numbers, such that a possible evaluation \( M(t_1) \) is \( \infty \). In this case the inequality \( t_1 > \infty \) is equivalent to \( \text{false} \).

than propositional logic to take advantage of some of the advances in MaxSAT and similarly OptSMT. We hope to enable new applications of \( \nu Z \) using the optimization capabilities.

Acknowledgment We would like to express our gratitude to Lars Fleckenstein and his group for co-hosting Phan during his internship and working on driving scenarios and applications of \( \nu Z \) in the context of warehouse logistics. We also thank Jessica Davies for fruitful discussions around MaxHS and Nina Narodytska for inspiring input around MaxRes and extensions.

References


