Playing with Quantified Satisfaction

Nikolaj Bjørner[1] and Mikoláš Janota[2]

[1] Microsoft Research, Redmond, USA
[2] Microsoft Research, Cambridge, UK

Abstract

We develop an algorithm for satisfiability of quantified formulas. The algorithm is based on recent progress in solving Quantified Boolean Formulas, but it generalizes beyond propositional logic to theories, such as linear arithmetic over integers (Presburger arithmetic), linear arithmetic over reals, algebraic data-types and arrays. Compared with previous algorithms for satisfiability of quantified arithmetical formulas our new implementation outperforms previous implementations in Z3 by a significant margin.

1 Introduction

Evaluating a quantified formula of the form $\exists x \forall y \exists z \forall u \,.\, F[x, y, z, u]$ can be naturally seen as a game between two parties: the existential player that seeks to find values for x and z that satisfy F and a universal player (spoiler) that attempts to counter the moves for x with values for y and the move for z with values for u such that falsify F. The connection between games and quantifiers is used in many connections, in model-theoretical tools such as Ehrenfeucht-Fraïssé games [1, 12], and also in algorithms for QBF formulas [15].

This paper takes as starting point the recent algorithm Qesto from [15] and develops generalizations for non-propositional formulas. Let us illustrate one central ingredient of Qesto on a simple example.

Example 1. *Consider the formula G, where:*

$$\begin{aligned} G &= \forall u_1, u_2 \exists e_1, e_2 \,.\, F \\ F &= (u_1 \wedge u_2 \rightarrow e_1) \wedge (u_1 \wedge \neg u_2 \rightarrow e_2) \wedge (e_1 \wedge e_2 \rightarrow \neg u_1) \end{aligned}$$

We wish to evaluate the formula G and the algorithm proceeds as a game between a universal and existential player. The universal player seeks to falsify F and the existential player seeks to satisfy F.

- *In the first round, the $\forall$ player can falsify F using a model M, where $M = [u_1 \mapsto \top, u_2 \mapsto \top, e_1 \mapsto \bot, e_2 \mapsto \bot]$. Thus, $\neg F[\top, \top, e_1, e_2]$ is satisfiable.*
- *In its quest to falsify F, the $\forall$ player took control over some existential variables. The $\exists$ player can now strike back and change values to satisfy F. For example, $F[\top, \top, \top, \bot]$ is true.*
- *The $\forall$ player has no move to counter this assignment. It has to backtrack over the assignment to e_1, e_2. In particular $\neg F[u_1, u_2, \top, \bot] \wedge u_2$ is already unsatisfiable without regard to the assignment to u_1. Therefore, the $\forall$ player learns $\neg u_2$ and reconsiders its original move and instead settles on the assignment $M = [u_1 \mapsto \top, u_2 \mapsto \bot, e_1 \mapsto \bot, e_2 \mapsto \bot]$ which satisfies $\neg F$.*
- *But the $\exists$ player can easily counter this move by setting $e_1 \mapsto \bot, e_2 \mapsto \top$.*
- *Therefore, the $\forall$ player has yet again to consider what went wrong by determining the core $\neg u_2 \wedge \neg F[u_1, u_2, \bot, \top]$, which is empty. The universal player has to give up and the existential player established that the formula is true.*

Several approaches to solving QBF formulas already fit into this level of abstraction. For example, Lintao Zhang's scheme for combining DNF and CNF [22] can be seen as realization of this approach. The approach was extended to finite domains in [4], but not infinite domains, such as linear arithmetic, as the approach did not incorporate finitary lemma learning. Qesto builds on top of this scheme by also tracking dependencies between universally and existentially quantified variables and carefully encoding auxiliary variables to capture these dependencies. The aim in this paper is to develop methods that work for theories that admit quantifier elimination. There are several key ingredients in our approach: at the highest level, quantifier satisfiability is solved as a game between two quantifiers, and we present a general algorithm based on this approach. We rely on theories admitting model-based quantifier elimination to compute effective cores and in the process we show how theories, such as linear arithmetic and algebraic datatypes can (separately) be solved using this approach. There is also a cute analogy with how Property Directed Reachability (PDR or IC3) [5] works. Similar to PDR, the approach we develop explores, then refines, a set of frames by adding properties in an incremental way. But in contrast to PDR, the frames alternate with respect to logical implication. The analogy only goes so far, of course, as we don't enjoy inductive generalization for quantified formulas (but a possible generalization of our methods to infinite games may readily do so). Yet, there are several other connections with various methods developed in QBF and SMT. For instance, dual propagation [11] exploits that a satisfying assignment to F is a core of $\neg F$, from which a reduced size implicant of F can be derived. In a similar way, we use partial satisfying assignments to F to search for cores of $\neg F$. The process of finding a small core and forming model-based projection on top of the core serves the purpose of finding implicants for F (with some variables projected away). We finally extend the algorithm for checking satisfiability of quantified formulas to also quantifier elimination. Similar to the work that uses dependency sequents [10] we produce formulas in CNF, but our use of dual propagation replaces a special purpose SAT solver with dependency sequents. Our approach furthermore extends to theories, while dependency sequents only apply to propositional logic. There are several approaches that have been developed for limited quantifier prefixes, such as validity of $\forall\exists$ fragment, e.g., satisfiability of the $\exists\forall$ fragments. Our algorithm, when restricted to the $\exists\forall$ fragment, is highly related to these methods. In [7], model-based projection is used to find instantiations of a (universally quantified) formula. The set of instances is finite as long as model-based projection produces a finite number of projections. Similar, in [21] an algorithm is presented for satisfying $\exists\forall$ formulas. Their procedure is a special case of ours for the two level case. Zachary Kincaid presented, at Dagstuhl on Information from Deduction: Models and Proofs, an algorithm for satisfying formulas over LRA that also uses a game-based approach. These other approaches do not consider strategies. Furthermore, a common differentiator is that our approach limits instantiations to clauses (and not the original formulas), this reduces the set of new atoms produced by model-based projection (it only has to consider the atoms in a core). RAReQs [14] also uses instantiations of full formulas, but is able to leverage the Boolean domain to produce strong projections.

2 Preliminaries

2.1 Cores

Suppose A is a formula and L is a conjunction of literals, an (minimal) unsatisfiable core, $L' \subseteq L$, is any (minimal) subset of L such that $A \wedge L'$ is unsatisfiable. We will overload the notion of core and also consider a formula L', which is inconsistent with A, uses the same vocabulary as L and which is a consequence of L, as a core.

2.2 Model Based Projection

Several quantifier elimination algorithms can be formulated as transformations that take a formula $\exists\vec{x} \,.\, F$ and produce an equivalent formula in the form of a disjunction of formulas. Each of the disjuncts is linear size in F, but the number of disjunctions may also be exponential in the number of bound variables $\vec{x}$. Dually, for a formula $\forall\vec{x} \,.\, F$, these procedures can be used to create an equivalent conjunction of formulas. However, to establish unsatisfiability of a conjunction $G \land \forall\vec{x} \,.\, F$ it is not necessarily useful to first expand $\forall\vec{x} \,.\, F$ into a large conjunction and then check ground unsatisfiability of the rest. Instead, it is only required to enumerate just the conjuncts for $\forall\vec{x} \,.\, F$ that are sufficient to show unsatisfiability. In the terminology of grounding, it is only interesting to enumerate the groundings of $\forall\vec{x} \,.\, F$ that suffice to establish unsatisfiability.

Model-based projection [17] was introduced to tailor a quantifier-elimination algorithm into an engine for producing partial instantiations. It follows the theme of model-based quantifier instantiation [3,6,8] by selecting instances based on models that should be blocked by a quantifier.

The model-based projection function $\mathit{Mbp}(M, \vec{x}, F)$ is required to compute a formula that implies $\exists\vec{x} \,.\, F$ and evaluates to true in M. In summary, the requirements for model-based projection are as follows:

Requirement 1 (Correctness, Complexity, Coverage, Convergence)**.** *Assume* $M \models F$ *and* $\varphi = \mathit{Mbp}(M, \vec{x}, F)$. *Then:*

Correctness $FV(\varphi) \cap \vec{x} = \emptyset$, $M \models \varphi$, *and* $\varphi \implies \exists\vec{x}F$.

Complexity *The size of* $\mathit{Mbp}(M, x, F)$ *is no larger than the size of* F.

Coverage *There is a sequence* $M_1, \ldots,$ *of models of* F, *such that* $(\exists x \,.\, F) \quad\equiv\quad \bigvee_i \mathit{Mbp}(M_i, x, F)$.

Convergence *Let* $M_1, M_2, \ldots,$ *be an infinite sequence of models of literals* F, *then there is a finite set of equivalent projections* $\mathit{Mbp}(M_i, x, F)$.

It suffices to define model based projection for a single variable and a conjunction of literals that contain the variable by using the following definitions:

$$\begin{array}{rcl}
\mathit{Mbp}(M, \emptyset, \varphi) & = & \varphi \\
\mathit{Mbp}(M, \vec{x}, \varphi) & = & \mathit{Mbp}\left(M, \vec{x}, \bigwedge\{\mathit{sign}(M, a) \;\mid\; a \text{ is an atom in } \varphi\}\right) \\
\mathit{Mbp}(M, x\vec{x}, \varphi) & = & \mathit{Mbp}(M, x, \mathit{Mbp}(M, \vec{x}, \varphi)) \\
\mathit{Mbp}(M, x, \varphi \land \psi) & = & \mathit{Mbp}(M, x, \varphi) \land \psi \quad \text{if} \quad x \notin FV(\psi)
\end{array}$$

where

$$\mathit{sign}(M, a) \quad := \quad \textbf{if } M(a) \textbf{ then } a \textbf{ else } \neg a$$

2.3 Model Based Projection for Linear Real Arithmetic

Suppose we have a variable x and literals L of the form $x \simeq t, x \not\simeq t, t \leq x, t < x, x \leq s$ or $x < s$, where x does not occur in t, s, and a model M, such that $M \models L$. Let us first simplify the case analysis by replacing strict inequalities by non-strict inequalities by using infinitesimals. In other words $t < x$ is replaced by the inequality $t + \epsilon \leq x$. If the set of literals L contains an equality $x \simeq t$, then the projection of L with respect to x is the substitution of t for x. If L contains a disequality

$x \not\simeq t$ then if $M(x) < M(t)$, replace the disequality by the inequality $x + \epsilon \leq t$, otherwise since $M \models x \not\simeq t$ it is the case that $M(x) > M(t)$ and we can replace the disequality by $x \geq \epsilon + t$. We can now assume that $M \models L$, where L only contains inequalities. Among the lower bounds select a greatest lower bound t_0 such that $M(t_0) \geq M(t_i)$ for every term t_i; or among the upper bounds a least upper bound s_0 such that $M(s_0) \leq M(s_j)$ for every term s_j. The projection with respect to x is the set $\{t \leq t_0 \mid (t \leq x) \in L\} \cup \{t_0 \leq s \mid (x \leq s) \in L\}$; or alternatively $\{t \leq s_0 \mid (t \leq x) \in L\} \cup \{s_0 \leq s \mid (x \leq s) \in L\}$.

Using equations, the projection can be defined as (we give the case where t_0 is the greatest lower bound for x under M):

$$\begin{array}{rcll}
\mathit{Mbp}(M, x, x \simeq t \wedge L) & = & L[t/x] & \\
\mathit{Mbp}(M, x, x \not\simeq t \wedge L) & = & \mathit{Mbp}(M, x, x \geq t + \epsilon \wedge L) \ \text{ if } & M(x) > M(t) \\
\mathit{Mbp}(M, x, x \not\simeq t \wedge L) & = & \mathit{Mbp}(M, x, x + \epsilon \leq t \wedge L) \ \text{ if } & M(x) < M(t) \\
\mathit{Mbp}(M, x, \bigwedge_i t_i \leq x \wedge \bigwedge_j x \leq s_j) & = & \bigwedge_i t_i \leq t_0 \wedge \bigwedge_j t_0 \leq s_j \ \text{ where } & M(t_0) \geq M(t_i), \forall i
\end{array}$$

Projection produces one disjunction from the Loos-Weispfenning quantifier elimination method [18].

Lemma 1. *Requirement 1 holds for Mbp of Linear Real Arithmetic.*

2.4 Model Based Projection for Linear Integer Arithmetic

Projection for integer linear arithmetic is somewhat similar. It splits bounds into greatest lower and least upper bounds. This time the dividing bound is of the form $t \leq ax$ and $bx \leq s$, where a, b are positive integer constants. To ensure that there exists an x between these two bounds we can use a model-based resolution rule originally used in the Omega test [20], where formulated in the style of [2]. It takes the form:

$$\begin{array}{lll}
\multicolumn{3}{l}{\mathit{resolve}(M, ax \leq t, bx \geq s) =} \\
\quad as + (a-1)(b-1) \leq bt & \textbf{if} & (a-1)(b-1) \leq M(bt - as) \\
\quad as \leq bt \wedge b|(s+d) \wedge a(s+d) \leq bt) & \textbf{elif} & a \geq b, d := M(-s) \mod b \\
\quad as \leq bt \wedge a|(t-d) \wedge as \leq b(t-d) & \textbf{else} & b > a, d := M(t) \mod a
\end{array}$$

Lemma 2. *Suppose $M \models ax \leq t, bx \geq s$, then $M \models \mathit{resolve}(M, ax \leq t, bx \geq s)$ and $\mathit{resolve}(M, ax \leq t, bx \geq s) \implies \exists x \,.\, ax \leq t \wedge bx \geq s$.*

Resolution accumulates divisibility constraints of the form $c \mid (ax + t)$, where c and a are positive integer constants and x is not free in t. Divisibility constraints can be seen as short-hands for equalities of the form $cy = ax + t$, where y is a fresh existentially bound variable. But this does not simplify the problem, instead it introduces new variables. We can eliminate divisibility constraints by using the model M to focus case analysis. Given $\bigwedge_{i=1}^n d_i \mid (a_i x + t_i)$ true in model M we define:

$$\begin{array}{rcl}
d & := & \mathit{lcm}(d_1, \ldots, d_n) \\
u & := & M(x) \mod d
\end{array}$$

Then we can replace the divisibility constraints by

$$\exists x'.x = u + d \cdot x' \wedge \bigwedge_{i=1}^{n} d_i \mid (a_i u + t_i)$$

We can use these two transformations to define model based projection for a conjunction of integer linear constraints as follows:

$$\begin{array}{rcl} \mathit{Mbp}(M, x, \bigwedge_{i=1}^{n} d_i \mid (a_i x + t_i) \wedge L) & = & \mathit{Mbp}\,(M, x', L[u + d \cdot x'/x]) \wedge \bigwedge_{i=1}^{n} d_i \mid (a_i u + t_i) \\ \mathit{Mbp}\left(M, x, \bigwedge_i t_i \leq a_i x \wedge \bigwedge_j b_j x \leq s_j\right) & = & \bigwedge_i t_i a_0 \leq t_0 a_i \wedge \bigwedge_j \mathit{resolve}(M, a_0 x \leq t_0, b_j x \geq s_j) \\ & & \text{if} \quad M(t_0/a_0) \geq M(t_i/a_i), \forall i \end{array}$$

Lemma 3. *Requirement 1 holds for Mbp of Linear Integer Arithmetic.*

2.5 Model Based Projection for Algebraic Datatypes

Let us for simplicity consider the case of LISP S-expressions. Generalization to other algebraic data-types is straight-forward. We assume we are given a conjunction of equalities, disequalities and constructor tests over data-type terms. The constructors are $\mathit{cons}(u, v)$, nil, there is a constructor test $\mathit{cons}?(u)$ which holds when u is a cons, and the functions $\mathit{car}(u)$, $\mathit{cdr}(u)$ are defined to take the first, respectively second argument, when u is a cons. Furthermore, we can assume that equalities are normalized, such that the right-hand side is a variable or expression using selectors and that there are no equalities whose one side occurs properly in the other side. We also maintain the following property: whenever L contains a subterm $\mathit{car}(u)$ or $\mathit{cdr}(u)$, then L also contains a conjunct $\mathit{cons}?(u)$. So whenever $M \models L$, then u evaluates to a cons under M. If x occurs in the scope of a car or cdr, then we apply the following transformation:

$$\mathit{Mbp}(M, x, L) \quad = \quad \mathit{Mbp}(M[y \mapsto M(\mathit{car}(x)), z \mapsto M(\mathit{cdr}(x))], yz, L[\mathit{cons}(y, z)/x])$$

and simplify L under the rewrites $\mathit{car}(\mathit{cons}(s, t)) = s$, $\mathit{cdr}(\mathit{cons}(s, t)) = t$. So in the following we assume that x is not in the scope of an accessor. We can solve for x (assume x occurs in t but not in u) by using the transformations corresponding to unification:

$$\begin{array}{rcl} \mathit{Mbp}(M, x, \mathit{cons}(t, s) \simeq u \wedge L) & = & \mathit{cons}?(u) \wedge \mathit{Mbp}(M, x, t \simeq \mathit{car}(u) \wedge s \simeq \mathit{cdr}(u) \wedge L) \\ \mathit{Mbp}(M, x, x \simeq x \wedge L) & = & \mathit{Mbp}(M, x, L) \\ \mathit{Mbp}(M, x, u \simeq x \wedge L) & = & L[u/x] \end{array}$$

This leaves us with disequalities where the variable x we would like to project occurs. Disequalities can be decomposed using the model M:

$$
\begin{array}{lcl}
\mathit{Mbp}(M, x, \mathit{cons}(t,s) \not\simeq u \wedge L) & = & \mathit{cons?}(u) \wedge \mathit{Mbp}(M, x, s \not\simeq \mathit{cdr}(u) \wedge L) \\
 & & \quad \text{if} \quad M(s) \neq M(\mathit{cdr}(u)), M(\mathit{cons?}(u)) \\
\mathit{Mbp}(M, x, \mathit{cons}(t,s) \not\simeq u \wedge L) & = & \mathit{cons?}(u) \wedge \mathit{Mbp}(M, x, t \not\simeq \mathit{car}(u) \wedge L) \\
 & & \quad \text{if} \quad M(s) = M(\mathit{cdr}(u)), M(t) \neq M(\mathit{car}(u)), M(\mathit{cons?}(u)) \\
\mathit{Mbp}(M, x, \mathit{cons}(t,s) \not\simeq u \wedge L) & = & \neg\mathit{cons?}(u) \wedge \mathit{Mbp}(M, x, L) \quad \text{if} \quad M(\neg\mathit{cons?}(u))
\end{array}
$$

Thus, every occurrence of x is isolated as $t_1 \not\simeq x \wedge t_2 \not\simeq x \wedge \ldots$. We can assume that x does not occur in t_i, because otherwise the disequality is a tautology by the occurs check. At this point we can always solve for x by constructing a suitably large term that is different from each of $t_1, t_2, \ldots$. Thus,

$$\mathit{Mbp}(M, x, t_1 \not\simeq x \wedge \ldots \wedge t_n \not\simeq x) \quad = \quad \top$$

Lemma 4. *Requirement 1 holds for* Mbp *of Algebraic Datatypes.*

2.6 Model Based Projection for Arrays

The paper [16] develops model-based projection techniques that apply to constraints with arrays. It generally does not satisfy the Convergence criteria as the number of projections can in this case be infinite.

3 A Quantified Satisfiability Game

We are now ready to describe our algorithm for quantified satisfiability.

In the following assume that we are given a formula $G := \exists \vec{x}_1 \forall \vec{x}_2 \exists \vec{x}_3 \ldots F$, where F is quantifier free. Assume G is closed (top level existential quantification can be used to create closed formulas). Initialize $F_1, F_3, \ldots F_j$, where j is odd, to F and $F_2, F_4, \ldots F_j$, where j is even, to $\neg F$. Let $a_1, \ldots, a_n$ be the atoms, denoted $\mathcal{A}$, occurring in $F_1, F_2, \ldots$. Each atom may contain any subset of the bound variables. An interpretation M assigns truth values to the atoms $\mathcal{A}$. The set $\mathcal{A}$ is updated every time new constraints are added to one of $F_1, F_2, \ldots$.

For each atom we associate two levels: $\mathit{level}_\forall$ and $\mathit{level}_\exists$. For the atom a, $\mathit{level}_\forall(a)$ is the highest index of a universally quantified variable in a, similarly for $\mathit{level}_\exists(a)$. We set $\mathit{level}(a) = \max(\mathit{level}_\exists(a), \mathit{level}_\forall(a))$. We also extend levels to depend on indices: If j is odd, then $\mathit{level}_j = \mathit{level}_\exists$. If j is even, then $\mathit{level}_j = \mathit{level}_\forall$.

Furthermore, define the auxiliary functions:

$$
\begin{array}{rcl}
\mathit{strategy}(M, j) & := & \bigwedge\{\mathit{sign}(M, a) \mid M \neq \mathit{null}, a \in \mathcal{A},\ \mathit{level}_j(a) \leq j - 2\} \\
\mathit{tail}(j) & := & \vec{x}_{j-1}, \vec{x}_j, \vec{x}_{j+1}, \ldots
\end{array}
$$

In other words, the literals in $\mathit{strategy}(M, j)$ use the free variables $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{j-2}, \vec{x}_{j-1}, \vec{x}_{j+1}, \vec{x}_{j+3}, \vec{x}_{j+5}, \ldots$ and they do not contain the variables $\vec{x}_j, \vec{x}_{j+2}, \vec{x}_{j+4}$ etc. For the special case where $M = \mathit{null}$, we note that $\mathit{strategy}(M, j) = \top$.

Algorithm 1 shows the QSAT algorithm. It initializes the level j to 1, model M to null and loops until it determines whether G is true or false. The algorithm applies also to the case where G has free variables.

In this case the algorithm determines whether G is satisfiable or unsatisfiable. If $F_j \wedge \mathit{strategy}(M, j)$ is satisfiable, the level is increased and M is updated to the model corresponding to a satisfying assignment to the formula. Notice that the level cannot increase forever because F_{j+1} is complementary to F_j, so when a strategy assigns a truth value to all atoms in F_j, then F_{j+1} is false under that strategy.

Algorithm 1: QSAT

```
$j \leftarrow 1$;
$M \leftarrow null$;
while True do
    if $F_j \wedge strategy(M, j)$ is unsat then
        if j = 1 then
            return $G$ is false
        if j = 2 then
            return $G$ is true
        $C \leftarrow Core(F_j, strategy(M, j))$;
        $J \leftarrow Mbp(M, tail(j), C)$;
        $j \leftarrow$ index of max variable in $J \cup \{1, 2\}$ of same parity as $j$;
        $F_j \leftarrow F_j \wedge \neg J$;
        $M \leftarrow null$;
    else
        $M \leftarrow$ the current model;
        $j \leftarrow j + 1$;
```

Example 2. *Assume we have the formula*

$$
\begin{aligned}
F &= z \geq 0 \wedge ((x \geq 0 \wedge y \geq 0) \vee y + z \leq 1) \\
G &= \exists x \forall y \exists z \,.\, F
\end{aligned}
$$

Then $F_1 = F, F_2 = \neg F, F_3 = F, F_4 = \neg F$.

1. *First we determine if* F_1 *is satisfiable, it is with model* $M = [x \mapsto 0, z \mapsto 3, y \mapsto -2]$. *The atomic formulas* $x \geq 0$ *and* $z \geq 0$ *and* $y + z \leq 1$ *are true.*

2. $\mathit{strategy}(M, 2) = z \geq 0 \wedge x \geq 0$, *because* $\mathit{level}_2(z \geq 0) = \mathit{level}_2(x \geq 0) = 0$, *but* $\mathit{level}_2(y \geq 0) = \mathit{level}_2(y + z \leq 1) = 2$. *Recall, that* $\mathit{level}_2 = \mathit{level}_\forall$, *which is the highest index of a universally quantified variable, while* $\mathit{level}_3 = \mathit{level}_\exists$, *which is the highest index of an existentially quantified variable. Since there are no universal quantifiers in the formula* $z \geq 0$, *we have* $\mathit{level}_\forall(z \geq 0) = 0$.

3. $F_2 \wedge x \geq 0 \wedge z \geq 0$ *is satisfiable with model* $M = [x \mapsto 0, z \mapsto 3, y \mapsto -1]$.

4. $\mathit{strategy}(M, 3) = z \geq 0 \wedge x \geq 0 \wedge \neg(y \geq 0)$ *because* $\mathit{level}_3(y + z \leq 1) = 3$ *and* $\mathit{level}_3(y \geq 0) = 0$.

5. $F_3 \wedge z \geq 0 \wedge x \geq 0 \wedge \neg(y \geq 0)$ *is satisfiable with model* $M = [x \mapsto 0, z \mapsto 0, y \mapsto -1]$.

6. $F_4 \wedge z \geq 0 \wedge x \geq 0 \wedge \neg(y \geq 0) \wedge y + z \leq 1$ *is unsatisfiable, in particular,*

$$\neg F \implies \neg(z \geq 0 \wedge y + z \leq 1)$$

So we set $I := \neg(z \geq 0 \wedge y + z \leq 1)$ *and compute* $J := Mbp(M, z, z \geq 0 \wedge y + z \leq 1) = y \leq 1$. *So* $F_2 \leftarrow F_2 \wedge \neg(y \leq 1)$ *and search back-jumps to level 2. Note that* $y \leq 1$ *is added to the set of atoms* $\mathcal{A}$.

7. F_2 *is satisfiable with model* $M = [x \mapsto -1, y \mapsto 2, z \mapsto -3]$.

8. $F_3 \wedge \neg(x \geq 0) \wedge \neg(y \leq 1)$ *is unsatisfiable with core* $\neg(x \geq 0)$*, and we back-jump to* $j = 1$.

9. F_1 *is satisfiable with* $M = [x \mapsto 0, z \mapsto 3, y \mapsto 2]$.

10. $F_2 \wedge z \geq 0 \wedge z \geq 0$ *is unsatisfiable, hence* F *is true.*

Let us now establish correctness for algorithm QSAT.

Invariant 1. $M = null$ *or* $j > 1, M \models F_{j-1}$.

Invariant 2. *For odd* j*:* $(\forall \vec{x}_{j+1} \exists \vec{x}_{j+2} \ldots F) \implies (\forall \vec{x}_{j+1} \exists \vec{x}_{j+2} \ldots F_j)$

Invariant 3. *For even* j*:* $(\forall \vec{x}_{j+1} \exists \vec{x}_{j+2} \ldots \neg F) \implies (\forall \vec{x}_{j+1} \exists \vec{x}_{j+2} \ldots F_j)$

Theorem 1. *Algorithm QSAT is partially correct: when it terminates it correctly determines whether G is true or false.*

Proof. Invariant 1 holds at initialization, is easily seen to be maintained during the loop. Similarly, invariants 2 and 3 are established at initialization time.

If $j = 1$ and F_1 is unsatisfiable, then from invariant 2 we know that the original formula is unsatisfiable.

If $j = 2$ and $F_2 \wedge \mathit{strategy}(M, 2)$ is unsatisfiable, then $M = null$ and already F_2 is unsatisfiable, hence $\neg\forall \vec{x}_2 \exists \vec{x}_3 \ldots .\ F$ is also unsatisfiable and therefore $\forall \vec{x}_2 \exists \vec{x}_3 \ldots .\ F$ is true. The case where $M \neq null$ is more general and in the following we will assume that $M \neq null$. In this case $\forall \vec{x}_2 \exists \vec{x}_3 \ldots, F$ holds for any assignment to $\vec{x}_1$ that satisfies the premise $strategy(M, 2)$. In particular it holds for any assignment to $\vec{x}_1$ that can be extended to an assignment satisfying the premise. We know that this premise is satisfiable by the way j is incremented.

Finally assume $j > 2$ and $F_j \wedge \mathit{strategy}(M, j)$ is unsatisfiable for $M \neq null$. In this case $F_j \implies \neg\mathit{strategy}(M, j)$, so there is a core of $strategy(M, j)$. In other words, the conjunction $F_j \wedge \mathit{strategy}(M, j)$ has a minimal, or reduced size, unsatisfiable core C that selects sufficient literals from $strategy(M, j)$ to obtain unsatisfiability. The core implies $\neg F_j$, but contains variables from $\vec{x}_{j-1}, \vec{x}_j, \ldots$ that we would like to eliminate. By taking $Mbp(M, \mathit{tail}(j), C)$ we obtain a formula $J := Mbp(M, \mathit{tail}(j), C)$, such that $J \implies (\exists \mathit{tail}(j)\ .\ C)$. Thus, $(\forall \mathit{tail}(j)\ .\ \neg C) \implies \neg J$.

We claim that $\forall tail(j)\ .\ \neg C$, which is the same as $\forall \vec{x}_{j-1}, \vec{x}_j \vec{x}_{j+1}, \ldots\ .\ \neg C$, is equivalent to

$$\forall \vec{x}_{j-1} \exists \vec{x}_j \forall \vec{x}_{j+1} \exists \vec{x}_{j+2}, \ldots,\ .\ \neg C.$$

To establish this, we examine the atoms in C. Since the free variables in C is a subset of the free variables in $strategy(M, j)$, they do not contain the variables $\vec{x}_j, \vec{x}_{j+2}, \ldots$. Therefore, it makes no difference whether we bind those variables universally or existentially.

By monotonicity $(\forall \vec{x}_{j-1} \exists \vec{x}_j \forall \vec{x}_{j+1} \ldots\ .\ F_j) \implies \neg J$, and thus we can add $\neg J$ to $F_{j-2}, F_{j-4}, \ldots$ while maintaining invariants 2 and 3.

□

Theorem 2. *Algorithm QSAT terminates for theories satisfying requirement 1.*

Proof. We work by induction on the nesting depth of quantifiers. At the inner-most levels, $F_j = F$ and $F_{j+1} = \neg F$. These formulas contain a finite number of atomic sub-formulas. The assumption is that model-based projection (Requirement 1) can only create a finite set of projections to eliminate the inner-most variables corresponding to F_{j+1}. At this point, levels F_{j-1} and F_j are complementary for variables $\vec{x}_1, \ldots, \vec{x}_{j-1}$, and therefore the maximal iteration depth of QSAT has decreased. □

4 On Cores

Taking any unsatisfiable core from $\mathit{strategy}(M, j)$ as C preserves correctness and termination for our definition of strategy. However, it is not the only approach. In [13] we use Farkas lemma to find weaker cores. The weaker cores use atoms that are linear combinations of inequalities occurring in F_j. Let us briefly recall the approach for linear real arithmetic.

Suppose $F_j \wedge \mathit{strategy}(M, j)$ is unsatisfiable, then then there is a resolution proof using a subset of literals from F_j resolved against literals from $\mathit{strategy}(M, j)$ and theory axioms, or T-axioms. Literals from T-axioms can be partitioned into two groups corresponding to the two conjuncts. Thus, axioms are of the form: $\neg \left(\left[\begin{array}{c} C \\ D \end{array} \right] \left[\begin{array}{c} \vec{x}_0 \\ \vec{x} \end{array} \right] \leq \left[\begin{array}{c} \vec{c} \\ \vec{d} \end{array} \right] \right)$, where the inequalities with coefficients $C, \vec{c}$ resolve against $\mathit{strategy}(M, j)$. The coefficients from Farkas' lemma are $\vec{\lambda}_C$ and $\vec{\lambda}_D$ respectively, such that:

$$\vec{\lambda}_C C \left[\begin{array}{c} \vec{x}_0 \\ \vec{x} \end{array} \right] + \vec{\lambda}_D D \left[\begin{array}{c} \vec{x}_0 \\ \vec{x} \end{array} \right] > \vec{\lambda}_C \vec{c} + \vec{\lambda}_D \vec{d},$$

and therefore:

$$D \left[\begin{array}{c} \vec{x}_0 \\ \vec{x} \end{array} \right] \leq \vec{d} \rightarrow \vec{\lambda}_D D \left[\begin{array}{c} \vec{x}_0 \\ \vec{x} \end{array} \right] \leq \vec{\lambda}_D \vec{d} \rightarrow \neg (\vec{\lambda}_C C \left[\begin{array}{c} \vec{x}_0 \\ \vec{x} \end{array} \right] \leq \vec{\lambda}_C \vec{c}) \ .$$

The disjunction of all such consequents $\vec{\lambda}_D D \left[\begin{array}{c} \vec{x}_0 \\ \vec{x} \end{array} \right] \leq \vec{\lambda}_D \vec{d}$ from all T-axioms is a sufficient consequence of F_j that does not contain any variables from $\vec{x}_0$ and is inconsistent with $\mathit{strategy}(M, j)$. So the negation of this disjunction forms a stronger core.

5 On Strategies

So far $\mathit{strategy}(M, j)$ selected atoms using M based on the deepest existentially or universally quantified variable occurring in the atom. We established that this approach was sound, but it is far from the most powerful way to constrain the moves of opposing players. For example, we can use a refined definition of strategy that fixes values to variables of the same quantifier.

Example 3. *Consider the formula $\exists x \forall y \exists z \,.\, 2z + y \leq x \vee 6 + y > x$. By playing $[x \mapsto 0, y \mapsto 1, z \mapsto 3]$ we can satisfy the formula. Furthermore, if we set $\mathit{strategy}(M, j) = x = 0 \wedge z = 3$, there are no possible moves for the $\forall y$ player and we immediately learn that the formula is true.*

Fixing variables to constants does not produce particularly strong strategies.

Example 4. *Consider the formula $\exists x \forall y \exists z \,.\, x + y \leq z$. The formula is true, but our approach so far has to first play the inner most $\exists z$ quantifier to counter $\forall y$'s move for any fixed x, z. A better strategy is to set $z := x + y$. In this way, no matter what y chooses, it is unable to satisfy $\neg(x + y \leq z)$, and the $\forall y$ player will have immediately lost.*

A generalization of the previous example suggests a strategy of the form: Given a model M, such that $M \models F_{j-1}$, set

$$\begin{array}{rcl} \mathit{strategy}^*(M, j) & = & \mathit{strategy}(M, j) \\ & \wedge & \bigwedge \left\{ \begin{array}{l} x = \max\{t_i \mid \mathit{level}(x) > \mathit{level}(t_i), (t_i \leq x) \in \mathcal{A}, M(t_i \leq x)\} \mid \\ \mathit{level}(x) \text{ is of same parity as } j \end{array} \right\} \end{array}$$

There are several possible improvements to this strategy as well. For example, the most straight-forward improvement is to consider the upper bounds for x instead of the lower bounds if these are more instrumental for falsifying F_j. Furthermore, there may be atoms where x occurs together with other quantified variables of the same level.

In the propositional case, strategies that are represented as arbitrary clauses can be shown to be more powerful. We leave exploration of stronger strategies as future work.

6 Quantifier Elimination

Suppose we have a formula $G[\vec{x}_0]$ where variables $\vec{x}_0$ are free. and we wish to compute a quantifier free formula that is equivalent to $G[\vec{x}_0]$. We can modify the quantifier satisfiability algorithm to compute this formula in stages. In each round we compute a quantifier-free formula J that implies $\neg G$. Thus, $G \implies \neg J$. We take the conjunction of these $\neg J$. In more detail, assume that $G[\vec{x}_0]$ is of the form $\exists \vec{x}_2 \forall \vec{x}_3, \ldots F$, then $\neg G[\vec{x}_0] \equiv \forall \vec{x}_2 \exists \vec{x}_3, \ldots \neg F$. Note that we omit the variables $\vec{x}_1$ in the expansion of G. When referring to $\vec{x}_1$ we will assume that it is an empty sequence of variables. Set $F_0 \leftarrow F, F_2 \leftarrow \neg F, F_3 \leftarrow F, \ldots$. Assume as induction hypothesis that $G[\vec{x}_0] \implies \mathit{Ans}$, where Ans is a quantifier-free formula containing at most variables $\vec{x}_0$. If $F_2 \wedge \mathit{Ans} \wedge \mathit{strategy}(M, 2)$ is unsatisfiable, then F_2 implies $\neg(\mathit{Ans} \wedge \mathit{strategy}(M, 2))$, and therefore $F_2 \implies \neg C$ for the core C. Then by model-based projection $J \implies \exists \vec{x}_2, \vec{x}_3 \ldots . C$, and since the free variables in C only include $\vec{x}_0, \vec{x}_2, \vec{x}_4, \vec{x}_6, \ldots$ we have $J \implies \exists \vec{x}_2 \forall \vec{x}_3 \exists \vec{x}_4 \ldots . C$. So by monotonicity ($F_2 \implies \neg C$), we get $J \implies \exists \vec{x}_2 \forall \vec{x}_3 \exists \vec{x}_4 \ldots . \neg F_2$, thus, $J \implies \neg G[\vec{x}_0]$.

Algorithm 2 shows a variant of QSAT adapted to quantifier elimination.

7 Evaluation

Figures 1 and 2 summarize evaluation of QSAT versus algorithm QT [19]. We see that overall the new algorithm, QSAT, substantially improves over QT.

8 Conclusions

We presented algorithms for satisfiability and quantifier elimination of quantified formulas. The algorithms take inspiration from recent developments in QBF solving and we show how to handle some theories that admit quantifier elimination, such as linear integer and real arithmetic and a theory of algebraic data-types. In the future we would like to extend the algorithms to use more powerful strategies, which can also be very useful in the Boolean case. We would also like to investigate other theories for model-based projection and when one can combine theories with projection. For example, it is known that algebraic data-types and linear arithmetic can be combined, but our current approach with separate projection operations for arithmetic and algebraic datatypes does not realize such a combination. Theories do not necessarily have to admit quantifier elimination for inclusion in this scheme and we would like

Algorithm 2: QE-SAT

```
$j \leftarrow 1$;
$M \leftarrow null$;
$Ans \leftarrow \top$;
while $True$ do
    if $F_j \wedge strategy(M, j) \wedge Ans$ is unsat then
        if $j = 1$ then
            return $Ans$
        $C \leftarrow Core(F_j, strategy(M, j) \wedge Ans)$;
        $J \leftarrow Mbp(M, tail(j), C)$;
        if $j = 2$ then
            $j \leftarrow 1$;
            $Ans \leftarrow Ans \wedge \neg J$;
        else
            $j \leftarrow$ index of max variable in $J \cup \{1, 2\}$ of same parity as $j$;
            $F_j \leftarrow F_j \wedge \neg J$;
        $M \leftarrow null$;
    else
        $M \leftarrow$ current model;
        $j \leftarrow j + 1$;
```

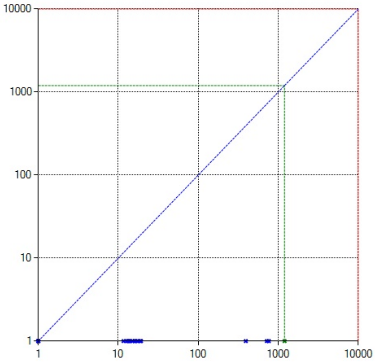

Figure 1: Time taken to solve 64 quantified integer linear arithmetic benchmarks from [19]. The method described in [19] is on the x-axis, our new method is on the y-axis. All benchmarks are solved almost instantaneously by QSAT with the longest run time being 0.08 seconds. This contrasts with 10 timeouts for QT.

.

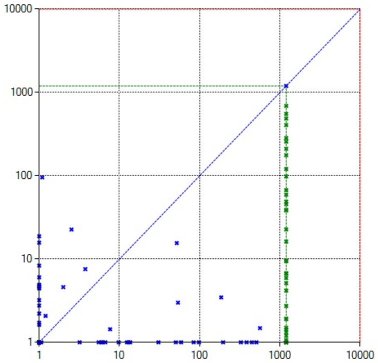

Figure 2: Time taken to solve benchmarks from SMT-LIB2 benchmark suite from the LRA benchmark suite. The x-axis shows running time for Algorithm QT and the y-axis shows running times for the new algorithm QSAT. Note that many of these benchmarks have been randomly generated and this reflects in overall fluctuations in the number of Simplex pivoting steps taken to check satisfiability. Still our new method performs much better overall. Algorithm QT times out for 42 benchmarks (within 1200 seconds), while QSAT times out for 2 benchmarks.

to investigate when taking the alternating view helps solving general formulas. This would contrast incremental grounding provided by quantifier instantiation methods. Finally, extending this method to solve reachability games should be within reach, and it opens up formulating and solving reachability games that rely on data and not just finite domain reachability games.

References

[1] K. Apt and E. Grädel, editors. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.

[2] N. Bjørner. Linear quantifier elimination as an abstract decision procedure. In Giesl and Hähnle [9], pages 316–330.

[3] M. P. Bonacina, C. Lynch, and L. M. de Moura. On Deciding Satisfiability by DPLL(G+T) and Unsound Theorem Proving. In R. A. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2009.

[4] L. Bordeaux and L. Zhang. A solver for quantified boolean and linear constraints. In Y. Cho, R. L. Wainwright, H. Haddad, S. Y. Shin, and Y. W. Koo, editors, *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC), Seoul, Korea, March 11-15, 2007*, pages 321–325. ACM, 2007.

[5] A. R. Bradley. SAT-Based Model Checking without Unrolling. In *VMCAI*, pages 70–87, 2011.

[6] L. M. de Moura and N. Bjørner. Bugs, moles and skeletons: Symbolic reasoning for software development. In Giesl and Hähnle [9], pages 400–411.

[7] G. Fedyukovich, A. Gurfinkel, and N. Sharygina. Automated Discovery of Simulation Between Programs. In *LPAR*, 2015.

[8] Y. Ge and L. M. de Moura. Complete Instantiation for Quantified Formulas in Satisfiabiliby Modulo Theories. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.

[9] J. Giesl and R. Hähnle, editors. *Automated Reasoning, 5th International Joint Conference, IJCAR 2010, Edinburgh, UK, July 16-19, 2010. Proceedings*, volume 6173 of *Lecture Notes in Computer Science*. Springer, 2010.

[10] E. Goldberg and P. Manolios. Quantifier elimination by dependency sequents. *Formal Methods in System Design*, 45(2):111–143, 2014.

[11] A. Goultiaeva, M. Seidl, and A. Biere. Bridging the gap between dual propagation and cnf-based QBF solving. In E. Macii, editor, *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*, pages 811–814. EDA Consortium San Jose, CA, USA / ACM DL, 2013.

[12] E. Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, and S.Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. Springer, 2007.

[13] K. Hoder and N. Bjørner. Generalized Property Directed Reachability. In *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, pages 157–171, 2012.

[14] M. Janota, W. Klieber, J. Marques-Silva, and E. M. Clarke. Solving QBF with counterexample guided refinement. In A. Cimatti and R. Sebastiani, editors, *SAT*, volume 7317, pages 114–128. Springer, 2012.

[15] M. Janota and J. Marques-Silva. Solving QBF by clause selection. In Q. Yang and M. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 325–331. AAAI Press, 2015.

[16] A. Komuravelli, N. Bjørner, A. Gurfinkel, and K. L. McMillan. Compositional verification of procedural programs using horn clauses over integers and arrays. In *FMCAD*, 2015.

[17] A. Komuravelli, A. Gurfinkel, and S. Chaki. SMT-Based Model Checking for Recursive Programs. In A. Biere and R. Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 17–34. Springer, 2014.

[18] R. Loos and V. Weispfenning. Applying linear quantifier elimination. *Comput. J.*, 36(5):450–462, 1993.

[19] A. Phan, N. Bjørner, and D. Monniaux. Anatomy of alternating quantifier satisfiability (work in progress). In P. Fontaine and A. Goel, editors, *10th International Workshop on Satisfiability Modulo Theories, SMT 2012, Manchester, UK, June 30 - July 1, 2012*, volume 20 of *EPiC Series*, pages 120–130. EasyChair, 2012.

[20] W. Pugh. A practical algorithm for exact array dependence analysis. *Commun. ACM*, 35(8):102–114, August 1992.

[21] A. Tiwari, A. Gascón, and B. Dutertre. Program synthesis using dual interpretation. In A. P. Felty and A. Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 482–497. Springer, 2015.

[22] L. Zhang. Solving QBF by combining conjunctive and disjunctive normal forms. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 143–150. AAAI Press, 2006.