QSMA: A New Algorithm for Quantified Satisfiability Modulo Theory and Assignment

Maria Paola Bonacina[1], Stéphane Graham-Lengrand[2], and Christophe Vauthier[3]

[1] Università degli Studi di Verona, Verona, Italy
mariapaola.bonacina@univr.it
[2] SRI International, Menlo Park, USA
stephane.graham-lengrand@csl.sri.com
[3] École Normale Supérieure, Paris, France
cvauthier@clipper.ens.psl.eu

Abstract. This paper presents and proves totally correct a new algorithm, called QSMA, for the satisfiability of a quantified formula modulo a complete theory and an initial assignment. The optimized variant of QSMA implemented in YicesQS is described and shown to preserve total correctness. A report on the performance of YicesQS at the 2022 SMT competition is included. YicesQS ran in the LIA, NIA, LRA, NRA, and BV categories and ranked second for the "largest contribution" award (single queries). It was the only solver to solve all LRA instances, where it was about two orders of magnitude faster than the second best solver (Z3).

1 Introduction

Applications of automated reasoning generate formulas involving both quantifiers and symbols defined in background theories. For example, software verification needs reasoners that decide the satisfiability of quantified formulas modulo theories such as data structures and arithmetic (e.g., [20]). Therefore, endowing SMT solvers with quantifier reasoning (e.g., [3,9,11–14,22]), enriching first-order theorem provers with built-in theories (e.g., [1, 2, 19]), and integrating provers and solvers [7], are major research objectives.

If there is a single background theory $\mathcal{T}$, the $\mathcal{T}$-satisfiability of quantified formulas can be reduced to that of quantifier-free formulas if $\mathcal{T}$ admits quantifier elimination (QE): for every formula φ there exists a quantifier-free formula F that is $\mathcal{T}$-equivalent to φ. Since computing F can be prohibitively expensive (e.g., exponential in linear rational arithmetic (LRA) and doubly exponential in linear integer arithmetic (LIA) [8]), QE is not a practical solution.

In this paper we propose a practical solution in the form of a new algorithm called QSMA. In QSMA the computation of quantifier-free *model-based under-approximations* (MBU) and *model-based over-approximations* (MBO) of quantified formulas embodies a lazy approach to QE, which is tailored for $\mathcal{T}$-satisfiability. MBU generates a quantifier-free implicant of the given formula that is true in the given model. *Model(-guided) generalization* for linear [12] and nonlinear real arithmetic (NRA) [17] is an instance of MBU. MBO generates a quantifier-free implied formula that is false in the given model. *Model interpolation* for NRA [17] is an instance of MBO.

The QSMA algorithm assumes that the theory $\mathcal{T}$ is *complete*. By its recursive nature, QSMA solves a generalized form of the satisfiability problem, called *quantified SMA* (*satisfiability modulo theory and assignment*): given a formula φ with *arbitrary quantification*, and an *initial assignment* to Boolean or first-order subterms of φ, find a theory model of φ that extends the initial assignment, or report that none exists. In addition to QSMA and its total correctness, we present an optimized variant named OptiQSMA, which preserves total correctness and is implemented in the YicesQS solver built on top of Yices 2. A report on experimental results from the 2022 SMT competition and a discussion complete the paper. We begin with a high-level view of QSMA.

1.1 High-Level View of the QSMA Algorithm

The QSMA algorithm works by progressively instantiating quantified variables. Consider a formula φ of the form $\exists\bar{x}_1.\forall\bar{x}_2.\exists\bar{x}_3\ldots F[\bar{x}_1,\bar{x}_2,\bar{x}_3,\ldots]$ where F is quantifier-free. For example, suppose the theory is LRA, $\varphi = \exists x.\forall y.\exists z.F$ and $F = z \geq 0 \wedge x \geq 0 \wedge y + z \geq 0$. Say that QSMA assigns $x{\leftarrow}0$. Whatever value is chosen for y, the algorithm can show that φ is true in LRA by assigning $z{\leftarrow}max(0,-y)$. If $F = z \geq 0 \wedge x \geq 0 \wedge y + z \leq 0$, no matter which (non-negative) value QSMA chooses for x, it can show that φ is false in LRA by picking $y{\leftarrow}1$, because there is no value for z that satisfies $z \geq 0 \wedge z \leq -1$.

For an example that is not in prenex normal form, consider a formula φ of the form $\exists x.((\forall y_1.F_1[x,y_1]) \Rightarrow (\forall y_2.F_2[x,y_2]))$, where F_1 and F_2 are quantifier-free. QSMA sees the formula as $\exists x.((\exists y_1.\neg F_1[x,y_1]) \vee (\neg\exists y_2.\neg F_2[x,y_2]))$, and then as $\exists x.(p_1 \vee \neg p_2)$, where p_1 and p_2 are proxy Boolean variables for the quantified subformulas. QSMA assigns values to x, p_1, and p_2. If p_1 is assigned true, the algorithm tries to extend the assignment with a value for y_1 that satisfies $\neg F_1[x,y_1]$. If p_2 is assigned false, the algorithm tries to show that there is no value for y_2 that satisfies $\neg F_2[x,y_2]$.

Without loss of generality ($\neg\neg$ converts $\forall$ into $\neg\exists\neg$), we consider formulas

$$\varphi = \exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]\{p_i{\leftarrow}\exists\bar{y}_i.G_i[\bar{z},\bar{x},\bar{y}_i]\}_{i=1}^k.$$

$F[\bar{z},\bar{x},\bar{p}]$ denotes a quantifier-free formula where the variables $\bar{z}$, $\bar{x}$, and $\bar{p}$ occur. Tuples $\bar{z}$ and $\bar{x}$ contain the first-order variables occurring free in F. Formula F is quantifier-free because the quantified subformulas $\varphi_i = \exists\bar{y}_i.G_i[\bar{z},\bar{x},\bar{y}_i]$ are replaced by proxy Boolean variables $\bar{p} = p_1,\ldots p_k$. Given an initial assignment to the free variables $\bar{z}$, we construct a QSMA-*tree* for φ. QSMA starts trying to satisfy $F[\bar{z},\bar{x},\bar{p}]$. If it fails, it means that φ is false under the initial assignment. If it succeeds, there are two cases. If $k = 0$, formula φ is true under the initial assignment. If $k > 0$, the algorithm descends recursively to consider the QSMA-subtrees for the φ_i subformulas ($1 \leq i \leq k$). If QSMA assigned true to p_i, it tries to show that φ_i is true. If QSMA assigned false to p_i, it tries to show that

φ_i is false. If it succeeds for all QSMA-subtrees, formula φ is true under the initial assignment. For this, the model built by QSMA should satisfy $F[\bar{z}, \bar{x}, \bar{p}] \wedge \bigwedge_{i=1}^{n}(p_i \Leftrightarrow \varphi_i)$. Otherwise, formula φ is false under the initial assignment.

2 Preliminaries

A signature Σ is given by a set S of sorts and a set of sorted symbols. Given a class $\mathcal{V} = (\mathcal{V}^s)_{s\in S}$ of disjoint sets of sorted variables, $\Sigma[\mathcal{V}]$-formulas, Σ-sentences, and $\Sigma[\mathcal{V}]$-interpretations are defined as usual. A Σ-structure is a $\Sigma[\emptyset]$-interpretation. We use x, y, z for first-order variables, p for Boolean ones, and $\bar{x}$, $\bar{y}$, $\bar{z}$, and $\bar{p}$ for tuples of such variables. We also use φ and ψ for formulas, F and G for quantifier-free formulas, $\mathcal{M}$ for interpretations, $\models$ for satisfaction and entailment, $=$ for identity, $\uplus$ for disjoint union, and $\setminus$ for set difference. $FV(\varphi)$ is the set of the variables occurring free in φ. Slightly abusing the notation, $FV(\varphi)$ is also treated as a tuple. Implication is written $\Rightarrow$ and logical equivalence is written $\Leftrightarrow$. If $\mathcal{V}_1 \subseteq \mathcal{V}_2$ (i.e., $\mathcal{V}_1^s \subseteq \mathcal{V}_2^s$ for all $s \in S$), a $\Sigma[\mathcal{V}_2]$-interpretation $\mathcal{M}_2$ is an *extension* of a $\Sigma[\mathcal{V}_1]$-interpretation $\mathcal{M}_1$ to $\mathcal{V}_2$, if $\mathcal{M}_2$ interprets the variables in $\mathcal{V}_2^s \setminus \mathcal{V}_1^s$ for all $s \in S$ and is otherwise identical to $\mathcal{M}_1$.

A theory $\mathcal{T}$ is defined by a signature Σ and a set of Σ-sentences called $\mathcal{T}$-axioms. A model of $\mathcal{T}$, or $\mathcal{T}$-model, is a Σ-structure that satisfies the $\mathcal{T}$-axioms. A $\mathcal{T}[\mathcal{V}]$-model is a $\Sigma[\mathcal{V}]$-interpretation that is a $\mathcal{T}$-model when the interpretation of variables is ignored. A theory $\mathcal{T}$ is *complete*, if it is consistent, and for all Σ-sentences F, either F or $\neg F$ is provable from the $\mathcal{T}$-axioms. In this paper we deal with a single theory $\mathcal{T}$ that has a unique $\mathcal{T}$-model $\mathcal{M}_0$, so that the interpretation of everything except variables is fixed. Therefore $\mathcal{T}$ is complete, for Σ-sentences $\mathcal{T}$-validity, $\mathcal{T}$-satisfiability, and truth in $\mathcal{M}_0$ coincide, all $\mathcal{T}[\mathcal{V}]$-models are extensions of $\mathcal{M}_0$, and a $\mathcal{T}$-satisfiability procedure is concerned only with assignments to variables. Since there are one theory and one signature, we write formula for $\Sigma[\mathcal{V}]$-formula and model for $\mathcal{T}$-model or $\mathcal{T}[\mathcal{V}]$-model. A *conservative theory extension* $\mathcal{T}^+$ of $\mathcal{T}$ adds to Σ special constants, called *values*, to name elements in the domain of $\mathcal{M}_0$ as needed. Conservative means that a $\mathcal{T}$-satisfiable formula is also $\mathcal{T}^+$-satisfiable.

The *quantified SMA problem* for theory $\mathcal{T}$ asks whether $\mathcal{M}_0 \models \varphi$ for an arbitrary formula φ and an initial assignment of values to the variables in $FV(\varphi)$. Formulas have the form $\varphi = \exists\bar{x}.F[\bar{z}, \bar{x}, \bar{p}]\{p_i \leftarrow \exists\bar{y}_i.G_i[\bar{z}, \bar{x}, \bar{y}_i]\}_{i=1}^{k}$ described in the introduction, where $FV(\varphi) = \bar{z}$ and quantified variables are standardized apart. If $FV(\varphi) = \emptyset$, we still have SMA problems when considering subformulas under an assignment to existentially quantified variables.

3 The QSMA Framework

The QSMA algorithm works with a tree representation of a formula φ. A node n in the tree is labeled with a pair $(\bar{x}, F)$, where $\bar{x}$ is a tuple of first-order variables, called the *local variables* of n, and F is a quantifier-free formula. The local variables are implicitly existentially quantified: they are existentially quantified variables whose quantifers have been stripped, so that they are locally free, so to speak, and can be assigned by the algorithm. An arc from a node n to a child node b is labeled with a Boolean variable p. This Boolean variable stands as a *proxy* for the quantified subformula represented by the subtree rooted at node b. Therefore, the Boolean variable p is also considered a proxy of b itself.

A formula φ may have free variables $FV(\varphi) = \bar{z}$, whose assignment is given initially as part of the SMA problem instance. These variables are called *rigid*, because their assignments do not change during the tree traversal. As the algorithm traverses the tree, the local variables of a node n are *rigid* from the point of view of a child node b: their assignments do not change during the traversal of the subtree rooted at b. Therefore, we represent a formula φ as a pair formed by a tuple of rigid variables and a labeled tree. Slightly abusing the terminology, we call this pair a QSMA-*tree*. The root of a tree T is denoted $root(T)$.

Definition 1 (QSMA-tree). *Given* $\varphi = \exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]\{p_i{\leftarrow}\exists\bar{y}_i.G_i[\bar{z},\bar{x},\bar{y}_i]\}_{i=1}^k$, *where* $FV(\varphi) = \bar{z}$ *and* $\varphi_i = \exists\bar{y}_i.G_i[\bar{z},\bar{x},\bar{y}_i]$, $1 \leq i \leq k$, *the* QSMA-tree *for* φ *is the pair* $\mathcal{G} = (\bar{z}, T)$, *where* $\bar{z}$ *is called the tuple of the rigid variables of* $\mathcal{G}$, *and* T *is a labeled tree defined inductively as follows:*

- *If* $k = 0$, T *consists of a single node* r *labeled* $(\bar{x}, F[\bar{z},\bar{x}])$*;*
- *If* $k > 0$, *for all* i, $1 \leq i \leq k$, *let* $\mathcal{G}_i = ((\bar{z},\bar{x}), T_i)$ *be the* QSMA-*tree for* φ_i, *where* $root(T_i)$ *is a node* b_i *labeled* $(\bar{y}_i, G_i[\bar{z},\bar{x},\bar{y}_i])$. *Then* T *is the tree with a new node* r *labeled* $(\bar{x}, F[\bar{z},\bar{x},\bar{p}])$ *as root,* k *outgoing arcs labeled* $p_1,\ldots,p_k$, *and* $b_1,\ldots,b_k$ *as children.*

If subformula φ_i occurs more than once in φ, the same proxy variable p_i is used for all occurrences. The *ancestors* of a node n in T are the nodes on the unique path from $root(T)$ to n excluding n itself. If node n in T is labeled $(\bar{x}, F)$, its k outgoing arcs are labeled $p_1,\ldots,p_k$, and $\bar{x}_1,\ldots,\bar{x}_m$ are the local variables of the ancestors of n, then $FV(F) \subseteq \{\bar{z},\bar{x}_1,\ldots,\bar{x}_m,\bar{x},p_1,\ldots,p_k\}$. The set of the *assignable variables at node* n is $Var(n) = \bar{x} \uplus \{p_1,\ldots,p_k\}$. The set of the *rigid variables at node* n is $Rigid(n) = \bar{z} \uplus \bar{x}_1 \uplus \ldots \uplus \bar{x}_m$. Thus, $FV(F) \subseteq Rigid(n) \cup Var(n)$, $Rigid(root(T)) = \bar{z}$, and the QSMA-subtree rooted at node n is $\mathcal{G}_n = (Rigid(n), T_n)$. For a node n with label $(\bar{x}, F)$, the components of the label are denoted $n.\bar{x}$ and $n.F$. The label of the arc from n to a child b is denoted $b.p$.

Example 1. Given $\exists x.((\forall y_1.F_1[x,y_1]) \Rightarrow (\forall y_2.F_2[x,y_2]))$ from Sect. 1.1, let $\varphi = \exists x.((\exists y_1.\neg F_1[x,y_1]) \vee (\neg\exists y_2.\neg F_2[x,y_2])) = \exists x.(p_1 \vee \neg p_2)\{p_i \leftarrow \exists y_i.\neg F_i[x,y_i]\}_{i=1}^2$. The QSMA-tree for φ has root r labeled $(x, p_1 \vee \neg p_2)$ with left child b_1 labeled $(y_1, \neg F_1[x,y_1])$, right child b_2 labeled $(y_2, \neg F_2[x,y_2])$, and arcs from r to b_1 and from r to b_2 labeled p_1 and p_2, respectively. Note how $FV(r.F) \subseteq \{x,p_1,p_2\}$, $Var(r) = \{x,p_1,p_2\}$, and $Rigid(r) = \emptyset$. Also, $FV(b_1.F) \subseteq \{x,y_1\}$, $FV(b_2.F) \subseteq \{x,y_2\}$, $Var(b_1) = \{y_1\}$, $Var(b_2) = \{y_2\}$, and $Rigid(b_1) = Rigid(b_2) = \{x\}$.

Example 2. Consider $\forall x.((\exists y_1.(x \simeq 2{\cdot}y_1)) \Rightarrow (\exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2)))$. A double negation eliminates the $\forall$, yielding $\neg(\exists x.((\exists y_1.(x \simeq 2{\cdot}y_1)) \wedge (\forall y_2.(3{\cdot}x \not\simeq 2{\cdot}y_2))))$.

Again, a double negation eliminates the $\forall$, producing $\neg(\exists x.((\exists y_1.(x \simeq 2{\cdot}y_1)) \wedge (\neg(\exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2)))))$. Let $\varphi = \exists x.((\exists y_1.(x \simeq 2{\cdot}y_1)) \wedge (\neg(\exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2)))) = \exists x.(p_1 \wedge \neg p_2)\{p_1 \leftarrow \exists y_1.(x \simeq 2{\cdot}y_1),\ p_2 \leftarrow \exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2)\}$. The original formula is true in LRA iff φ is false in LRA. The QSMA-tree for φ has root r labeled $(x, p_1 \wedge \neg p_2)$ with left child b_1 labeled $(y_1, x \simeq 2{\cdot}y_1)$, right child b_2 labeled $(y_2, 3{\cdot}x \simeq 2{\cdot}y_2)$, and arcs from r to b_1 and from r to b_2 labeled p_1 and p_2, respectively. The variable sets of this tree are as in Example 1.

Conversely, given a QSMA-tree $\mathcal{G} = (\bar{z}, T)$, we can associate a formula $n.\psi$ to any node n in T and hence to the QSMA-subtree $\mathcal{G}_n = (\mathit{Rigid}(n), T_n)$.

Definition 2 (Formula at a node). *Given a* QSMA*-tree* $\mathcal{G} = (\bar{z}, T)$*, for all nodes* n *of* T*, the* formula $n.\psi$ at node n *is defined inductively as follows:*

- *If* n *is a leaf labeled* $(\bar{x}, F[\bar{z}, \bar{x}])$*, then* $n.\psi = \exists \bar{x}.F[\bar{z}, \bar{x}]$*;*
- *If* n *has label* $(\bar{x}, F[\bar{z}, \bar{x}, \bar{p}])$ *and outgoing arcs labeled* $p_1, \ldots, p_k$*,* $k > 0$*, connecting* n *to children* $b_1, \ldots, b_k$*, let* $b_1.\psi, \ldots, b_k.\psi$ *be the formulas at* $b_1, \ldots, b_k$*. Then* $n.\psi = \exists \bar{x}.F[\bar{z}, \bar{x}, \bar{p}]\{p_i \leftarrow b_i.\psi\}_{i=1}^{k}$*.*

If $\mathcal{G} = (\bar{z}, T)$ is the QSMA-tree for φ and $r = \mathit{root}(T)$, then $r.\psi = \varphi$.

Example 3. For the QSMA-tree in Example 2, $b_1.\psi = \exists y_1.(x \simeq 2{\cdot}y_1)$, $b_2.\psi = \exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2)$, and $r.\psi = \exists x.(p_1 \wedge \neg p_2)\{p_1 \leftarrow \exists y_1.(x \simeq 2{\cdot}y_1),\ p_2 \leftarrow \exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2)\} = \exists x.((\exists y_1.(x \simeq 2{\cdot}y_1)) \wedge \neg(\exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2))) = \varphi$.

Since the input formula φ is represented as a QSMA-tree $\mathcal{G} = (\bar{z}, T)$, the problem of satisfying φ becomes the problem of satisfying $\mathcal{G}$. Therefore, we define *satisfaction of a* QSMA*-tree* next. Slightly abusing the notation, we use $\models$ also for satisfaction of QSMA-trees.

Definition 3 (Satisfaction of a QSMA-tree). *Given a* QSMA*-tree* $\mathcal{G} = (\bar{z}, T)$ *with* $r = \mathit{root}(T)$*, and an extension* $\mathcal{M}$ *of* $\mathcal{M}_0$ *to* $\mathit{Rigid}(r) = \bar{z}$*,* $\mathcal{M} \models \mathcal{G}$ *if there exists an extension* $\mathcal{M}'$ *of* $\mathcal{M}$ *to* $\mathit{Var}(r)$ *such that (i)* $\mathcal{M}' \models r.F$*, and (ii) for all children* b *of* r*,* $\mathcal{M}'(b.p) = \mathsf{true}$ *iff* $\mathcal{M}' \models \mathcal{G}_b$*.*

The QSMA algorithm works by traversing the QSMA-tree $\mathcal{G} = (\bar{z}, T)$, and at each node n in T it assigns the assignable variables in $\mathit{Var}(n) = \bar{x} \uplus \{p_1, \ldots, p_k\}$. This assignment corresponds to the extension $\mathcal{M}'$ in Definition 3. Let b be a child of n: the Boolean variable $b.p$ labeling the arc from n to b is a proxy for the quantified subformula $b.\psi$ of the formula $n.\psi$. If $\mathcal{M}'(b.p) = \mathsf{true}$, the aim of the algorithm is to show that $b.\psi$ is true, and if $\mathcal{M}'(b.p) = \mathsf{false}$, the aim is to show that $b.\psi$ is false. Therefore Condition (ii) in Definition 3 says $\mathcal{M}' \models \mathcal{G}_b$ if $\mathcal{M}'(b.p) = \mathsf{true}$ and $\mathcal{M}' \not\models \mathcal{G}_b$ if $\mathcal{M}'(b.p) = \mathsf{false}$. The next theorem shows that satisfying a formula φ and satisfying the QSMA-tree for φ correspond.

Theorem 1. *For all formulas* φ *with* $FV(\varphi) = \bar{z}$*, for all models* $\mathcal{M}$ *extending* $\mathcal{M}_0$ *to* $\bar{z}$*, if* $\mathcal{G}$ *is the* QSMA*-tree for* φ *then* $\mathcal{M} \models \mathcal{G}$ *iff* $\mathcal{M} \models \varphi$*.*

Proof. The formula φ and the QSMA-tree $\mathcal{G} = (\bar{z}, T)$ are as in Def. 1. The proof is by induction on the number k of the quantified subformulas φ_i of φ.
Base case: $k = 0$ and T consists of the single node r with label $(\bar{x}, F[\bar{z}, \bar{x}])$. By Def. 3, $\mathcal{M} \models \mathcal{G}$ iff there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\mathit{Var}(r) = \bar{x}$ such that $\mathcal{M}' \models F[\bar{z}, \bar{x}]$, that is, iff $\mathcal{M} \models \varphi$.
Induction hypothesis: $k \geq 0$, and for all i, $1 \leq i \leq k$, for all models $\mathcal{M}$ extending $\mathcal{M}_0$ to $\mathit{Rigid}(b_i) = \bar{z} \uplus \bar{x}$, $\mathcal{M} \models \mathcal{G}_i$ iff $\mathcal{M} \models \varphi_i$.
Induction step: we distinguish the two directions.
$\Rightarrow$) Let $\mathcal{M}$ be an extension of $\mathcal{M}_0$ to $\mathit{Rigid}(r) = \bar{z}$, such that $\mathcal{M} \models \mathcal{G}$. By Def. 3, there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\mathit{Var}(r)$ such that $\mathcal{M}' \models F[\bar{z}, \bar{x}, \bar{p}]$ and for all i, $1 \leq i \leq k$, $\mathcal{M}'(p_i) = \mathsf{true}$ iff $\mathcal{M}' \models \mathcal{G}_i$. By induction hypothesis, $\mathcal{M}'(p_i) = \mathsf{true}$ iff $\mathcal{M}' \models \varphi_i$. Therefore, $\mathcal{M}' \models F[\bar{z}, \bar{x}, \bar{p}] \wedge \bigwedge_{i=1}^{k} p_i \Leftrightarrow \varphi_i$, and hence $\mathcal{M} \models \varphi$.
$\Leftarrow$) Let $\mathcal{M}$ be an extension of $\mathcal{M}_0$ to $\mathit{Rigid}(r) = \bar{z}$, such that $\mathcal{M} \models \varphi$. Under $\mathcal{M}$'s interpretation of $\bar{z} \uplus \bar{x}$, φ is equisatisfiable to $\psi = F[\bar{z}, \bar{x}, \bar{p}] \wedge \bigwedge_{i=1}^{n} p_i \Leftrightarrow \varphi_i$. Let $\mathcal{M}'$ be a model of ψ: $\mathcal{M}'$ is a model of $F[\bar{z}, \bar{x}, \bar{p}]$ such that $\mathcal{M}'(p_i) = \mathsf{true}$ iff $\mathcal{M}' \models \varphi_i$. By induction hypothesis, $\mathcal{M}'(p_i) = \mathsf{true}$ iff $\mathcal{M}' \models \mathcal{G}_i$. By Def. 3, $\mathcal{M} \models \mathcal{G}$.

Checking whether $\mathcal{M} \models \mathcal{G}$ by testing all possible extensions $\mathcal{M}'$ would not do, because for most theories (e.g., LRA) there is an infinite number of extensions. We need a way to weed out large parts of the space of candidate models. Let $[\![\varphi]\!]$ denote the set of φ's models. We introduce *under-approximations* and *over-approximations* of φ in order to under-approximate and over-approximate $[\![\varphi]\!]$.

Definition 4 (Under- and over-approximation). *Let φ be a formula with $FV(\varphi) = \bar{z}$. Quantifier-free formulas U and O with $FV(U) = FV(O) = \bar{z}$ are, respectively, an* under-approximation *and an* over-approximation *of φ, if for all extensions $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$, $\mathcal{M} \models U$ implies $\mathcal{M} \models \varphi$ and $\mathcal{M} \models \varphi$ implies $\mathcal{M} \models O$.*

It follows that $[\![U]\!] \subseteq [\![\varphi]\!] \subseteq [\![O]\!]$. Let $\mathcal{G} = (\bar{z}, T)$ be the QSMA-tree for φ, and U and O under- and over-approximations of φ, respectively. Then, $\mathcal{M} \models U$ implies $\mathcal{M} \models \varphi$ which implies $\mathcal{M} \models \mathcal{G}$ by Theorem 1. Thus, satisfying an under-approximation is a sufficient condition to have a solution. On the other hand, $\mathcal{M} \models \neg O$ implies $\mathcal{M} \not\models \varphi$ which implies $\mathcal{M} \not\models \mathcal{G}$ by Theorem 1. By the contrapositive, if $\mathcal{M} \models \mathcal{G}$ then $\mathcal{M} \not\models \neg O$, that is, $\mathcal{M} \models O$. Thus, satisfying an over-approximation is a necessary condition to have a solution. In order to construct such approximations, we assume to have a solver for theory $\mathcal{T}$ (and model $\mathcal{M}_0$) offering:

- *Model extension*: A function SMA such that for all formulas $\exists \bar{x}.F[\bar{z}, \bar{x}]$, where $F[\bar{z}, \bar{x}]$ is quantifier-free, and all extensions $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$, $\mathsf{SMA}(F[\bar{z}, \bar{x}], \mathcal{M})$ returns either an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\bar{x}$ such that $\mathcal{M}' \models F[\bar{z}, \bar{x}]$, or *nil* if there is no such extension.
- *Model-based under-approximation*: A function MBU such that for all formulas $\exists \bar{x}.F[\bar{z}, \bar{x}]$, where $F[\bar{z}, \bar{x}]$ is quantifier-free, and all extensions $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$ such that $\mathcal{M} \models \exists \bar{x}.F[\bar{z}, \bar{x}]$, $\mathsf{MBU}(F[\bar{z}, \bar{x}], \bar{x}, \mathcal{M})$ returns a quantifier-free formula $U[\bar{z}]$ such that $\mathcal{M} \models U[\bar{z}]$ and $\mathcal{T} \models U[\bar{z}] \Rightarrow (\exists \bar{x}.F[\bar{z}, \bar{x}])$.

– *Model-based over-approximation*: A function MBO such that for all formulas $\exists\bar{x}.F[\bar{z},\bar{x}]$, where $F[\bar{z},\bar{x}]$ is quantifier-free, and all extensions $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$ such that $\mathcal{M} \not\models \exists\bar{x}.F[\bar{z},\bar{x}]$, $\mathsf{MBO}(F[\bar{z},\bar{x}],\bar{x},\mathcal{M})$ returns a quantifier-free formula $O[\bar{z}]$ such that $\mathcal{M} \not\models O[\bar{z}]$ and $\mathcal{T} \models (\exists\bar{x}.F[\bar{z},\bar{x}]) \Rightarrow O[\bar{z}]$.

MBU and MBO produce, respectively, an under-approximation and an over-approximation. Formula $U[\bar{z}]$ is true in model $\mathcal{M}$ and implies $\exists\bar{x}.F[\bar{z},\bar{x}]$, and hence can be seen as an *interpolant between model and formula*. It was called *model generalization* [12,17], because $U[\bar{z}]$ may have other models in addition to $\mathcal{M}$. Formula $O[\bar{z}]$ follows from $\exists\bar{x}.F[\bar{z},\bar{x}]$ and is false in $\mathcal{M}$, and hence can be seen as a *reverse interpolant between formula and model*, called *model interpolant* [17].

4 The QSMA Algorithm and Its Total Correctness

Let $\mathcal{G} = (\bar{z}, T)$ be the QSMA-tree for input formula φ with $FV(\varphi) = \bar{z}$. Given a model $\mathcal{M}$ extending $\mathcal{M}_0$ to $\bar{z}$, the QSMA algorithm determines whether $\mathcal{M} \models \mathcal{G}$. Suppose that U and O are under- and over-approximations of φ, respectively. Picture $[\![U]\!]$, $[\![\varphi]\!]$, and $[\![O]\!]$ as bubbles. The $[\![U]\!]$ bubble is inside the $[\![\varphi]\!]$ bubble, which is inside the $[\![O]\!]$ bubble. The idea of the algorithm is to zoom in on a model of φ, by progressively weakening U, so that the $[\![U]\!]$ bubble inflates, and progressively strengthening O, so that the $[\![O]\!]$ bubble deflates. The algorithm operates in this manner for all subformulas of φ: for all nodes n of T it maintains under and over-approximations $n.U$ and $n.O$ of $n.\psi$, progressively weakening $n.U$ and strengthening $n.O$. The weakening of $n.U$ is done by introducing a disjunction with an MBU. The strengthening of $n.O$ is done by introducing a conjunction with an MBO. The goal is that $\mathcal{M}$ satisfies $n.U \vee \neg n.O$. As soon as $\mathcal{M}$ satisfies $n.U$, we know that $\mathcal{M} \models \mathcal{G}_n$. As soon as $\mathcal{M}$ satisfies $\neg n.O$, we know that $\mathcal{M} \not\models \mathcal{G}_n$.

@pre: $\mathcal{G} = (\bar{z}, T)$: QSMA-tree for φ with $FV(\varphi) = \bar{z}$; $\mathcal{M}$: extension of $\mathcal{M}_0$ to $\bar{z}$
@post: rv iff $\mathcal{M} \models \mathcal{G}$ (rv is "returned value")

function QSMA($\mathcal{M}$, T)
  for all nodes n in T **do**
    $n.U \leftarrow \bot$
    $n.O \leftarrow \top$
  return SUBTREEISSOLVED($root(T)$, $\mathcal{M}$)

Fig. 1. Pseudocode of the main function of the QSMA algorithm

The main function QSMA (Fig. 1) initializes $n.U$ to $\bot$ (under-approximation of all formulas and identity for disjunction) and $n.O$ to $\top$ (over-approximation of all formulas and identity for conjunction) for all nodes n of T. Then QSMA calls the function `subtreeIsSolved` (Fig. 2) with arguments $root(T)$ and $\mathcal{M}$.

@pre: $\mathcal{M}$: extension of $\mathcal{M}_0$ to $\mathit{Rigid}(n)$, and $I = \forall b \in T.\ [\![b.U]\!] \subseteq [\![b.\psi]\!] \subseteq [\![b.O]\!]$
@post: I and $\mathcal{M} \models (n.U \vee \neg n.O)$ and (rv iff $\mathcal{M} \models \mathcal{G}_n$) and ($rv$ iff $\mathcal{M} \models n.U$) and ($\neg rv$ iff $\mathcal{M} \models \neg n.O$)

1: **function** SUBTREEISSOLVED(n, $\mathcal{M}$)
2:     **if** $\mathcal{M} \models n.U$ **then**
3:         **return** *true*
4:     **else if** $\mathcal{M} \models \neg n.O$ **then**
5:         **return** *false*
6:     **while** true **do**
7:         $L \leftarrow n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U))$
8:         $\mathcal{M}' \leftarrow \mathsf{SMA}(L, \mathcal{M})$
9:         **if** $\mathcal{M}' = \mathit{nil}$ **then**
10:             $n.O \leftarrow n.O \wedge \mathsf{MBO}(L, FV(L) \setminus \mathit{Rigid}(n), \mathcal{M})$
11:             **return** *false*
12:         **else**
13:             **if** SOLUTIONFORALLCHILDREN(n, $\mathcal{M}'$) **then**
14:                 $L' \leftarrow n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$
15:                 $n.U \leftarrow n.U \vee \mathsf{MBU}(L', FV(L') \setminus \mathit{Rigid}(n), \mathcal{M})$
16:                 **return** *true*
17:
18: **function** SOLUTIONFORALLCHILDREN(n, $\mathcal{M}$)
19:     **for** all children b of n **do**
20:         **if** $\mathcal{M}(b.p) \neq$ SUBTREEISSOLVED(b, $\mathcal{M}$) **then**
21:             **return** *false*
22:     **return** *true*

Fig. 2. Pseudocode of the auxiliary functions of the QSMA algorithm

Function `subtreeIsSolved` takes a node n and a model $\mathcal{M}$ extending $\mathcal{M}_0$ to $\mathit{Rigid}(n)$ and determines whether $\mathcal{M} \models \mathcal{G}_n$. If $\mathcal{M} \models n.U$ it returns *true*; if $\mathcal{M} \models \neg n.O$ it returns *false* (lines 3-5 in Fig. 2). Otherwise (i.e., $\mathcal{M} \models \neg n.U \wedge n.O$), it enters a loop whose body contains the following steps:

1. Build a formula L as the conjunction of $n.F$ and a formula for every child b of n, denoted $n \to b$ (line 7 in Fig. 2). The shape of the formula for b is better explained by considering a model of L and hence in the next step.
2. Invoke the SMA function to search for an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\mathit{Var}(n)$ such that $\mathcal{M}' \models L$ (line 8). For all children b of n, $b.p \in \mathit{Var}(n)$ and $\mathcal{M}'$ assigns a Boolean value to $b.p$. If $\mathcal{M}'(b.p) = \mathsf{true}$, the subformula for b in L reduces to $b.O$, so that $\mathcal{M}' \models L$ implies $\mathcal{M}' \models b.O$. Since QSMA seeks to satisfy $b.\psi$ and $[\![b.\psi]\!] \subseteq [\![b.O]\!]$, it starts at least from a model of $b.O$. If $\mathcal{M}'(b.p) = \mathsf{false}$, the subformula for b in L reduces to $\neg b.U$, so that $\mathcal{M}' \models L$ implies $\mathcal{M}' \models \neg b.U$. Since QSMA seeks to falsify $b.\psi$ and $[\![b.U]\!] \subseteq [\![b.\psi]\!]$, it starts at least from a model of $\neg b.U$. The proof of partial correctness of `subtreeIsSolved` shows that the existence of an $\mathcal{M}'$ such that $\mathcal{M}' \models L$ is necessary for $\mathcal{M} \models \mathcal{G}_n$.

3. If SMA returns *nil*, then $\mathcal{M} \not\models \mathcal{G}_n$; `subtreeIsSolved` updates $n.O$ to its conjunction with $\mathsf{MBO}(L, FV(L) \setminus \mathit{Rigid}(n), \mathcal{M})$ (line 10). Since $\mathcal{M} \not\models L$, by MBO's specification we know that $\mathcal{M} \not\models \mathsf{MBO}(L, FV(L) \setminus \mathit{Rigid}(n), \mathcal{M})$. This update ensures that $\mathcal{M} \not\models n.O$, so that $\mathcal{M} \models \neg n.O$. Then `subtreeIsSolved` returns *false* (line 11).
4. Otherwise, we have an extension $\mathcal{M}'$ that satisfies L and hence $n.F$, so that there is potential for $\mathcal{M} \models \mathcal{G}_n$. Function `solutionForallChildren` is invoked to determine whether this is the case.
5. The function `solutionForallChildren` calls `subtreeIsSolved` for every child b of n. As soon as it finds a child b such that $\mathcal{M}(b.p) = \mathsf{true}$ and the call `subtreeIsSolved`(b,$\mathcal{M}$) returns *false*, or $\mathcal{M}(b.p) = \mathsf{false}$ and the call `subtreeIsSolved`(b,$\mathcal{M}$) returns *true*, it returns *false*, because it found a QSMA-subtree where candidate model $\mathcal{M}$ fails. If this does not happen, `solutionForallChildren` returns *true*.
6. If `solutionForallChildren` returns *true*, `subtreeIsSolved` builds a formula L' as the conjunction of $n.F$ and a formula for every child b of n (line 14). If $\mathcal{M}'(b.p) = \mathsf{true}$, the subformula for b in L' reduces to $b.U$. If $\mathcal{M}'(b.p) = \mathsf{false}$, the subformula for b in L' reduces to $\neg b.O$. The proof of partial correctness of `subtreeIsSolved` shows that $\mathcal{M}' \models L'$ and that $\mathcal{M}' \models L'$ is a sufficient condition for $\mathcal{M} \models \mathcal{G}_n$. Then `subtreeIsSolved` updates $n.U$ to its disjunction with $\mathsf{MBU}(L', FV(L') \setminus \mathit{Rigid}(n), \mathcal{M})$ (line 15). Since $\mathcal{M}' \models L'$, by MBU's specification we know that $\mathcal{M}' \models \mathsf{MBU}(L', FV(L') \setminus \mathit{Rigid}(n), \mathcal{M})$. This update ensures that $\mathcal{M}' \models n.U$. Then `subtreeIsSolved` returns *true* (line 16).
7. If `solutionForallChildren` returns *false*, the control returns to line 7. Suppose that `solutionForallChildren` returned *false*, because it found a child b of n such that $\mathcal{M}(b.p) = \mathsf{true}$ and `subtreeIsSolved`(b,$\mathcal{M}$) returned *false*. Then the call `subtreeIsSolved`(b,$\mathcal{M}$) updated the formula $b.O$ (line 10). Suppose that `solutionForallChildren` returned *false*, because it found a child b of n such that $\mathcal{M}(b.p) = \mathsf{false}$ and `subtreeIsSolved`(b,$\mathcal{M}$) returned *true*. Then the call `subtreeIsSolved`(b,$\mathcal{M}$) updated the formula $b.U$ (line 15). Either way the state has changed, variable L gets a new formula on line 7, and the subsequent call to SMA will not produce the same model.

Example 4. Apply `subtreeIsSolved` to the root of the QSMA-tree in Example 1. Formula L gets $p_1 \vee \neg p_2$. SMA produces an $\mathcal{M}'$ that assigns values to x, p_1, and p_2. Suppose that $\mathcal{M}'$ satisfies $p_1 \vee \neg p_2$ by assigning true to p_1. In the recursive call on b_1, formula L gets $\neg F_1[x, y_1]$. If SMA produces an $\mathcal{M}''$ that extends $\mathcal{M}'$ with an assignment to y_1 such that $\mathcal{M}'' \models \neg F_1[x, y_1]$, we have a model. Suppose that $\mathcal{M}'$ satisfies $p_1 \vee \neg p_2$ by assigning false to p_2. In the recursive call on b_2, formula L gets $\neg F_2[x, y_2]$. If SMA fails to produce an $\mathcal{M}''$ that extends $\mathcal{M}'$ with an assignment to y_2 such that $\mathcal{M}'' \models \neg F_2[x, y_2]$, we have a model.

Theorem 2. *The function* `subtreeIsSolved` *is partially correct: if the preconditions hold and the function halts, then the postconditions hold.*

Proof. Consider a call `subtreeIsSolved`$(n, \mathcal{M})$. We assume that the preconditions hold and the call terminates, and we show that the postconditions hold.

The proof is by structural induction on the tree T_n in $\mathcal{G}_n$.
Base case: n is a leaf. If $\mathcal{M} \models n.U$ and the function returns *true* on line 3 in Fig. 2, we have $\mathcal{M} \models (n.U \vee \neg n.O)$, $rv = \mathit{true}$, and $\mathcal{M} \models \mathcal{G}_n$, since $\mathcal{M} \models n.U$ implies $\mathcal{M} \models n.\psi$. If $\mathcal{M} \models \neg n.O$ and the function returns *false* on line 5, we have $\mathcal{M} \models (n.U \vee \neg n.O)$, $rv = \mathit{false}$, and $\mathcal{M} \not\models \mathcal{G}_n$, since $\mathcal{M} \models \neg n.O$ implies $\mathcal{M} \not\models n.\psi$. Otherwise, L is assigned $n.F$ since n has no children, and SMA is invoked to find an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(n.F)$ such that $\mathcal{M}' \models n.F$.
If no such extension is found, $\mathsf{MBO}(n.F, FV(n.F) \setminus \mathit{Rigid}(n), \mathcal{M})$ returns a quantifier-free formula that is false in $\mathcal{M}$, formula $n.O$ is conjoined with this formula, and the function returns *false* on line 11. Thus, $\mathcal{M} \not\models n.O$, $\mathcal{M} \models \neg n.O$, $\mathcal{M} \models (n.U \vee \neg n.O)$, $rv = \mathit{false}$, and $\mathcal{M} \not\models \mathcal{G}_n$, since $n.F$, and hence $n.\psi$, cannot be satisfied. If SMA returns an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(n.F)$ such that $\mathcal{M}' \models n.F$, `solutionForallChildren`$(n, \mathcal{M}')$ returns *true* because n has no children, L' is assigned $n.F$ for the same reason, $\mathsf{MBU}(n.F, FV(n.F) \setminus \mathit{Rigid}(n), \mathcal{M})$ returns a quantifier-free formula that is true in $\mathcal{M}$, formula $n.U$ is disjoined with this formula, and the function returns *true* on line 16. Thus, $\mathcal{M} \models n.U$, $\mathcal{M} \models (n.U \vee \neg n.O)$, $rv = \mathit{true}$, and $\mathcal{M} \models \mathcal{G}_n$, since $\mathcal{M} \models n.U$ implies $\mathcal{M} \models n.\psi$.
Induction hypothesis: for all children b of node n, if the preconditions are satisfied and `subtreeIsSolved`$(b, \mathcal{M})$ halts, the postconditions are satisfied.
Induction step: if `subtreeIsSolved`$(n, \mathcal{M})$ returns on line 3 or on line 5, the reasoning is the same as in the base case. Otherwise, L is assigned the formula $n.F \wedge \bigwedge_{n\to b}((b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U))$. This formula is constructed in such a way that if $\mathcal{M} \models \mathcal{G}_n$ then L is satisfied. Indeed, suppose that $\mathcal{M} \models \mathcal{G}_n$. This means that there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ such that: (i) $\mathcal{M}' \models n.F$; (ii) for all children b of n with $\mathcal{M}'(b.p) = \mathsf{true}$, $\mathcal{M}' \models \mathcal{G}_b$ (by Def. 3), so that $\mathcal{M}' \models b.\psi$ (by Thm. 1), and by induction hypothesis ($[\![b.\psi]\!] \subseteq [\![b.O]\!]$) $\mathcal{M}' \models b.O$; and (iii) for all children b of n with $\mathcal{M}'(b.p) = \mathsf{false}$, $\mathcal{M}' \not\models \mathcal{G}_b$ (by Def. 3), so that $\mathcal{M}' \not\models b.\psi$ (by Thm. 1), and hence $\mathcal{M}' \models \neg b.\psi$, and by induction hypothesis ($[\![b.U]\!] \subseteq [\![b.\psi]\!]$) $\mathcal{M}' \not\models b.U$, and hence $\mathcal{M}' \models \neg b.U$. By (i), (ii), and (iii), $\mathcal{M}' \models L$.
Function SMA is invoked to find precisely an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(L)$ such that $\mathcal{M}' \models L$. If no such extension exists, $\mathsf{MBO}(L, FV(L) \setminus \mathit{Rigid}(n), \mathcal{M})$ returns a quantifier-free formula that is false in $\mathcal{M}$, formula $n.O$ is conjoined with this formula, and the function returns *false* on line 11. Therefore, $\mathcal{M} \not\models n.O$, $\mathcal{M} \models \neg n.O$, $\mathcal{M} \models (n.U \vee \neg n.O)$, $rv = \mathit{false}$, and $\mathcal{M} \not\models \mathcal{G}_n$, so that the postconditions of `subtreeIsSolved`$(n, \mathcal{M})$ are satisfied. If there exists an extension $\mathcal{M}'$ such that $\mathcal{M}' \models L$, `solutionForallChildren`$(n, \mathcal{M}')$ is invoked. If it returns *true*, L' is assigned the formula $n.F \wedge \bigwedge_{n\to b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$. This formula is constructed in such a way that it has two properties.
The first one is that $\mathcal{M}' \models L'$. Indeed, $\mathcal{M}' \models n.F$, because $\mathcal{M}' \models L$, and from the knowledge that $\mathcal{M} \models \mathcal{G}_n$ (`solutionForallChildren` returned *true*) we know that for all children b of n, if $\mathcal{M}'(b.p) = \mathsf{true}$, `subtreeIsSolved`$(b, \mathcal{M}')$ returned *true*, so that $\mathcal{M}' \models b.U$ by induction hypothesis, and if $\mathcal{M}'(b.p) = \mathsf{false}$, `subtreeIsSolved`$(b, \mathcal{M}')$ returned *false*, so that $\mathcal{M}' \models \neg b.O$ by induction hypothesis.

The second property is that $\mathcal{M}' \models L'$ is a sufficient condition for $\mathcal{M} \models \mathcal{G}_n$. Indeed, $\mathcal{M}' \models L'$ implies (i) $\mathcal{M}' \models n.F$, and (ii) for all children b of n, if $\mathcal{M}'(b.p) = \mathsf{true}$, then $\mathcal{M}' \models L'$ implies $\mathcal{M}' \models b.U$, and by induction hypothesis $\mathcal{M}' \models \mathcal{G}_b$, if $\mathcal{M}'(b.p) = \mathsf{false}$, then $\mathcal{M}' \models L'$ implies $\mathcal{M}' \models \neg b.O$, and by induction hypothesis $\mathcal{M}' \not\models \mathcal{G}_b$. Then, $\mathsf{MBU}(L', FV(L') \setminus \mathit{Rigid}(n), \mathcal{M})$ returns a quantifier-free formula that is true in $\mathcal{M}$, formula $n.U$ is disjoined with this formula, and the function returns *true* on line 16. Thus, $\mathcal{M} \models n.U$, $\mathcal{M} \models (n.U \vee \neg n.O)$, $rv = \mathit{true}$, and $\mathcal{M} \models \mathcal{G}_n$, so that the postconditions are satisfied.

For termination, we begin with the MBU and MBO functions. Let $\mathcal{T}$ be LRA with a theory extension LRA^+ that adds constant symbols $\tilde{q}$ for all rational numbers q. Consider an MBU function such that $\mathsf{MBU}(F[\bar{z}, x], x, \mathcal{M}) = F[\bar{z}, x]\{x \leftarrow \tilde{q}\}$ and $\mathcal{M} \models F[\bar{z}, \tilde{q}]$. This kind of MBU is called *generalization-by-substitution* [12]. While $F[\bar{z}, \tilde{q}]$ is an under-approximation of $\exists x.F[\bar{z}, x]$, this MBU is not a good choice for termination. By applying MBU repeatedly with an infinite enumeration of rational constants, the QSMA algorithm could build an infinite sequence of under-approximations $(\bigvee_{i=1}^{n} F[\bar{z}, x]\{x \leftarrow \tilde{q}_i\})_{n \in \mathbb{N}}$ none of which is LRA-equivalent to $\exists x.F[\bar{z}, x]$. The next definition excludes such MBU functions, by requiring that for a given formula and variable tuple (that depends on the formula), MBU can generate only finitely many formulas.

Definition 5 (Finite basis). *An* MBU *function has* finite basis *if the set* $\{\mathsf{MBU}(F[\bar{z}, \bar{x}], \bar{x}, \mathcal{M}) \mid \mathcal{M}$: *extension of* $\mathcal{M}_0$ *to* $\bar{z}$ *such that* $\mathcal{M} \models \exists \bar{x}.F[\bar{z}, \bar{x}]\}$ *is finite for all quantifier-free formulas* $F[\bar{z}, \bar{x}]$ *and tuples* $\bar{x}$*.*

The notion of an MBO function having a finite basis is defined in the same way with $\not\models$ in place of $\models$.

Lemma 1. *If* MBU *and* MBO *have finite basis, for all (possibly infinite) series of calls* $\{\mathtt{subtreeIsSolved}(n, \mathcal{M}_i)\}_i$*, all satisfying the preconditions and all terminating, formulas* $n.U$ *and* $n.O$ *are updated only a finite number of times.*

Proof. The proof is by structural induction on the tree T_n rooted at node n. The base case (n is a leaf) is trivial. The induction hypothesis is that the claim holds for all children b of n. For the induction step, given a series of calls as in the claim, let $(n.U)_i$ and $(n.O)_i$ denote the values of $n.U$ and $n.O$ upon entering call $\mathtt{subtreeIsSolved}(n, \mathcal{M}_i)$. The same notation applies to all children b of n. By induction hypothesis, for all children b of n, $b.U$ and $b.O$ are updated only a finite number of times. Therefore, there exists an i_0 such that for all $i \geq i_0$, for all children b of n, $(b.U)_{i+1} = (b.U)_i$ and $(b.O)_{i+1} = (b.O)_i$. Then for all i, $i \geq i_0$, either (I) $(n.O)_{i+1} = (n.O)_i$ or (II) $(n.O)_{i+1} = (n.O)_i \wedge \mathsf{MBO}(L_i, FV(L_i) \setminus \mathit{Rigid}(n), \mathcal{M}_i)$ where $L_i = n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow (b.O)_i) \wedge (\neg b.p \Rightarrow \neg (b.U)_i))$.
Case (II) applies only if $\mathcal{M}_i \models (n.O)_i$ (if we enter the main loop $\mathcal{M}_i \models \neg(n.U)_i \wedge (n.O)_i$), $\mathcal{M}_i \models \neg(n.O)_{i+1}$, and $\mathtt{subtreeIsSolved}(n, \mathcal{M}_i)$ returns *false* (see lines 10-11 in Fig. 2 and Step (3) in the description of $\mathtt{subtreeIsSolved}$). Since for all i, $i \geq i_0$, $(b.U)_{i+1} = (b.U)_i$ and $(b.O)_{i+1} = (b.O)_i$, it follows that for all i,

$i \geq i_0$, $L_{i+1} = L_i$. Therefore, for all i, $i \geq i_0$, whenever we hit Case (II), MBO is applied to the same formula and variable tuple, while the third argument (the model) may vary. By hypothesis, MBO has finite basis (see Def. 5) and hence it can generate only finitely many formulas for a given formula. Thus, $n.O$ is updated only a finite number of times.
Similarly, for all i, $i \geq i_0$, either (I) $(n.U)_{i+1} = (n.U)_i$ or (II) $(n.U)_{i+1} = (n.U)_i \vee \mathsf{MBU}(L'_i, FV(L'_i) \setminus \mathit{Rigid}(n), \mathcal{M}_i)$ where $L'_i = n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow (b.U)_i) \wedge (\neg b.p \Rightarrow \neg(b.O)_i))$. Case (II) applies only if $\mathcal{M}_i \models \neg(n.U)_i$, $\mathcal{M}_i \models (n.U)_{i+1}$, and $\texttt{subtreeIsSolved}(n, \mathcal{M}_i)$ returns *true* (see lines 15-16 in Fig. 2 and Step (6) in the description of `subtreeIsSolved`). Since for all i, $i \geq i_0$, $(b.U)_{i+1} = (b.U)_i$ and $(b.O)_{i+1} = (b.O)_i$, it follows that for all i, $i \geq i_0$, $L'_{i+1} = L'_i$. Therefore, for all i, $i \geq i_0$, whenever we hit Case (II), MBU is applied to the same formula and variable tuple, while the third argument (the model) may vary. By hypothesis, MBU has finite basis (see Def. 5) and hence it can generate only finitely many formulas for a given formula. Thus, $n.U$ is updated only a finite number of times.

Once nontermination due to MBU or MBO is excluded even for an infinite series of halting calls, termination is proved by induction on the QSMA-tree.

Theorem 3. *If the* MBU *and* MBO *functions have finite basis, whenever the preconditions are satisfied the function* `subtreeIsSolved` *halts.*

Proof. Consider a call $\texttt{subtreeIsSolved}(n, \mathcal{M})$ for a node n in T. The base case (n is a leaf) is trivial. The induction hypothesis is that the claim holds for all children $b_1, \ldots, b_k$ of n. For the induction step, if $\texttt{subtreeIsSolved}(n, \mathcal{M})$ does not enter the main loop, it halts. Suppose that it enters the main loop. For this case we reason by way of contradiction, assuming that $\texttt{subtreeIsSolved}(n, \mathcal{M})$ does not halt. This means that the SMA function produces an infinite series of candidate models $\{\mathcal{M}_i\}_{i \geq 1}$ such that for all i, $i \geq 1$, there exists a child $b_{j(i)}$, $1 \leq j(i) \leq k$, for which $\mathcal{M}_i(b_{j(i)}.p) \neq \texttt{subtreeIsSolved}(b_{j(i)}, \mathcal{M}_i)$ so that `solutionForallChildren` returns *false* (lines 20-21 in Fig. 2). It follows that $\texttt{subtreeIsSolved}(n, \mathcal{M})$ generates an infinite series $\mathcal{S}$ of recursive calls.
Let W be a matrix with a row for each M_i, $i \geq 1$, a column for each b_h, $1 \leq h \leq k$, and such that $W_{i,h} = 1$ if $\mathcal{M}_i(b_h.p) = \texttt{subtreeIsSolved}(b_h, \mathcal{M}_i)$, $W_{i,h} = 0$ if $\mathcal{M}_i(b_h.p) \neq \texttt{subtreeIsSolved}(b_h, \mathcal{M}_i)$, and $W_{i,h} = \bot$ if `subtreeIsSolved` is not invoked on $(b_h, \mathcal{M}_i)$. For all h, $1 \leq h \leq k$, let $D_h = \{i \mid W_{i,h} = 0\}$. By projecting on the node argument, we extract from $\mathcal{S}$ up to k (possibly infinite) series of calls $\{\texttt{subtreeIsSolved}(b_h, \mathcal{M}_i)\}_{i \in D_h}$. Consider anyone of these series and let us temporarily rename b_h as b for simplicity. For all the calls $\texttt{subtreeIsSolved}(b, \mathcal{M}_i)$ in the series, since $\mathcal{M}_i$ was produced by SMA (line 8 in Fig. 2), we know that $\mathcal{M}_i \models L$, so that before the call

$$\mathcal{M}_i \models (b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U).$$

If $\mathcal{M}_i(b.p) = \texttt{true}$, then before the call $\mathcal{M}_i \models b.O$, and since the call returns *false*, it means that the call has updated $b.O$ to ensure that $\mathcal{M}_i \models \neg b.O$ (line 10 in Fig. 2 and Step (3) in the description of `subtreeIsSolved`). Similarly, if $\mathcal{M}_i(b.p) = \mathsf{false}$, then before the call $\mathcal{M}_i \models \neg b.U$, and since the call returns *true* it means that the call has updated $b.U$ to ensure that $\mathcal{M}_i \models b.U$ (line 15 in Fig. 2 and Step (6) in the description of `subtreeIsSolved`). In summary, at least one of $b.U$ or $b.O$ gets updated for each call in the series. However, by induction hypothesis all the calls in all the possibly infinite series $\{\texttt{subtreeIsSolved}(b_h, \mathcal{M}_i)\}_{i \in D_h}$ are terminating. Therefore, Lemma 1 applies to each of these series, establishing that $b_h.U$ and $b_h.O$ get updated only a finite number of times. Therefore, all the series $\{\texttt{subtreeIsSolved}(b_h, \mathcal{M}_i)\}_{i \in D_h}$ are finite, which contradicts the existence of the infinite series $\mathcal{S}$.

Example 5. Apply `subtreeIsSolved` to the root of the QSMA-tree in Example 2. Formula L gets $p_1 \wedge \neg p_2$. SMA produces an $\mathcal{M}'$ that assigns values to x, p_1, and p_2. Suppose that $\mathcal{M}'$ assigns 1 to x, while it must assign true to p_1 and false to p_2. In the recursive call on b_1, formula L gets $x \simeq 2{\cdot}y_1$. If SMA produces an $\mathcal{M}''$ that extends $\mathcal{M}'$ with $y_1 {\leftarrow} \frac{1}{2}$, we have a model of $\mathcal{G}_{b_1}$. In the recursive call on b_2, formula L gets $3{\cdot}x \simeq 2{\cdot}y_2$. If SMA produces an $\mathcal{M}''$ that extends $\mathcal{M}'$ with $y_2 {\leftarrow} \frac{3}{2}$, we have a model of $\mathcal{G}_{b_2}$, but because $\mathcal{M}'(p_2) = \mathsf{false}$, there is no model of $\mathcal{G}$. Indeed, formula φ of Example 2 is false as the original formula is true.

5 The OptiQSMA Algorithm and Its Total Correctness

YicesQS implements an optimized variant of QSMA, called OptiQSMA, that reduces the number of recursive calls to `subtreeIsSolved` by entrusting more work to each call to SMA. Reconsider the behavior of QSMA in Example 4. We can avoid a recursive call to `subtreeIsSolved` by asking SMA to satisfy $(p_1 \vee \neg p_2) \wedge (p_1 \Rightarrow \neg F_1[x, y_1])$ in lieu of $p_1 \vee \neg p_2$. This way, if the candidate model returned by SMA assigns true to p_1, it also assigns to x and y_1 values that satisfy $\neg F_1[x, y_1]$. This means that $\exists y_1 . \neg F_1[x, y_1]$ is found true without recursion. On the other hand, if p_2 is assigned false, the algorithm still has to make the recursive call to see if it can satisfy $\exists y_2 . \neg F_2[x, y_2]$.

The idea of OptiQSMA is to do a look-ahead on a path in the QSMA-tree, doing the work in one shot rather then through recursive calls on all the nodes in the path. The look-ahead applies to a path such that the Boolean labels of all the arcs in the path are assigned true by the candidate model. The following definition builds a formula to allow the look-ahead.

Definition 6 (Look-ahead formula). *Given a* QSMA*-tree $\mathcal{G} = (\bar{z}, T)$, for all nodes n of T the* look-ahead formula *of n is $LF(n) = n.F \wedge \bigwedge_{n \to b} (b.p \Rightarrow LF(b))$.*

The next definition distinguishes the nodes that are handled together in one shot without recursion and those where recursion is still needed. Nodes of the first kind are called *no alternation nodes*, because such nodes are on a path as described above, where all Boolean labels are assigned true and hence there is no alternation between true and false. Nodes of the second kind are called *first alternation nodes*, because they are the nodes reached by the first arc whose Boolean label is assigned false.

Definition 7 (No alternation nodes and first alternation nodes). *Given a* QSMA*-tree* $\mathcal{G} = (\bar{z}, T)$ *for all nodes* n *of* T *and extensions* $\mathcal{M}$ *of* $\mathcal{M}_0$ *to* $FV(LF(n))$*, the set* $\mathsf{NAN}(n, \mathcal{M})$ *of the* no-alternation nodes from n according to $\mathcal{M}$ *(resp. the set* $\mathsf{FAN}(n, \mathcal{M})$ *of the* first-alternation nodes from n according to $\mathcal{M}$*) contains all and only the nodes* b *such that: (i)* b *is a descendant of* n *through a path* $n \to n_1 \to \ldots \to n_q \to b$ $(q \geq 0)$*, (ii)* $\forall i$*,* $1 \leq i \leq q$*,* $\mathcal{M}(n_i.p) = \mathsf{true}$*, and (iii)* $\mathcal{M}(b.p) = \mathsf{true}$ *(resp.* $\mathcal{M}(b.p) = \mathsf{false}$*).*

A node $b \in \mathsf{FAN}(n, \mathcal{M})$ such that $q = 0$ in Condition (i) of Definition 7 is a child of n: for a child there is no optimization. The $\mathsf{OptiQSMA}$ algorithm seeks a candidate model $\mathcal{M}$ that satisfies $LF(n)$ and recurses only on the nodes in $\mathsf{FAN}(n, \mathcal{M})$. Therefore, the definition of *satisfaction with look-ahead*, denoted $\models_{la}$, follows the pattern of Definition 3, replacing $r.F$ with $LF(r)$ and Condition (ii) of Definition 3 with a condition for the nodes in the FAN set.

Definition 8 (Satisfaction with look-ahead). *Given a* QSMA*-tree* $\mathcal{G} = (\bar{z}, T)$ *with* $r = root(T)$ *and an extension* $\mathcal{M}$ *of* $\mathcal{M}_0$ *to* $Rigid(r) = \bar{z}$*,* $\mathcal{M} \models_{la} \mathcal{G}$ *if there exists an extension* $\mathcal{M}'$ *of* $\mathcal{M}$ *to* $FV(LF(r))$ *such that (i)* $\mathcal{M}' \models LF(r)$ *and (ii) for all nodes* $b \in \mathsf{FAN}(r, \mathcal{M}')$*,* $\mathcal{M}' \not\models_{la} \mathcal{G}_b$*.*

Since for the nodes $b \in \mathsf{FAN}(r, \mathcal{M}')$ it is $\mathcal{M}'(b.p) = \mathsf{false}$, the $\models_{la}$ relation is negated in Condition (ii). The next theorem shows that the optimization does not change the problem.

Theorem 4. *Given a* QSMA*-tree* $\mathcal{G} = (\bar{z}, T)$ *and an extension* $\mathcal{M}$ *of* $\mathcal{M}_0$ *to* $\bar{z}$*,* $\mathcal{M} \models \mathcal{G}$ *if and only if* $\mathcal{M} \models_{la} \mathcal{G}$*.*

Proof. The proof is by structural induction on the tree T. Let $r = root(T)$.
Base case: if r is the only node in T, the claim is trivially true, because $LF(r) = r.F$ and Condition (ii) in both Defs. 3 and 8 is vacuously true.
Induction hypothesis: the claim holds for all children b of r.
Induction step: we distinguish the two directions.
$\Rightarrow$) By hypothesis, $\mathcal{M} \models \mathcal{G}$, that is, there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $Var(r)$ that fulfills Def. 3. We build an extension $\mathcal{M}''$ of $\mathcal{M}'$ to $FV(LF(r))$ that fits Def. 8. First, $FV(LF(r)) = FV(r.F) \cup \{b.p \mid r \to b\} \cup \bigcup_{r \to b} FV(LF(b))$. Note that $FV(r.F) \subseteq Rigid(r) \cup Var(r)$ and $\{b.p \mid r \to b\} \subseteq Var(r)$. Since $\mathcal{M}$ interprets the variables in $Rigid(r)$ and $\mathcal{M}'$ extends $\mathcal{M}$ to interpret the variables in $Var(r)$, we need to consider only the variables in $\bigcup_{r \to b} FV(LF(b))$. Since $FV(LF(b))$ may contain variables that are in $Rigid(b) = Rigid(r) \cup \{\bar{x}\}$ for $\bar{x}$ the local variables of r and $\bar{x} \subseteq Var(r)$, $\mathcal{M}''$ only needs to add interpretations of the variables in $FV(LF(b)) \setminus Rigid(b)$ for all children b of r. Let b be a child of r such that $\mathcal{M}'(b.p) = \mathsf{false}$. Then, for all $y \in FV(LF(b)) \setminus Rigid(b)$, let $\mathcal{M}''$ assign an arbitrary value to y. Let b be a child of r such that $\mathcal{M}'(b.p) = \mathsf{true}$. Since $\mathcal{M} \models \mathcal{G}$, by Def. 3, $\mathcal{M}' \models \mathcal{G}_b$, and by induction hypothesis $\mathcal{M}' \models_{la} \mathcal{G}_b$, that is, there exists an extension $\mathcal{M}'_b$ of $\mathcal{M}'$ fulfilling Def. 8 for $\mathcal{G}_b$. Then, for all $y \in FV(LF(b)) \setminus Rigid(b)$, let $\mathcal{M}''(y) = \mathcal{M}'_b(y)$ (†).

@pre: $\mathcal{G} = (\bar{z}, T)$: QSMA-tree for φ with $FV(\varphi) = \bar{z}$; $\mathcal{M}$: extension of $\mathcal{M}_0$ to $\bar{z}$
@post: rv iff $\mathcal{M} \models_{la} \mathcal{G}$

function OptiQSMA($\mathcal{M}$, T)
    for all nodes n in T **do**
        $n.U \leftarrow \perp$
    ans $\leftarrow$ OPTISUBTREEISSOLVED($root(T)$, $\mathcal{M}$)
    if ans = SAT(_) **then**
        return *true*
    else if ans = UNSAT(_) **then**
        return *false*

Fig. 3. Pseudocode of the main function of the OptiQSMA algorithm

This construction of $\mathcal{M}''$ does not assign two different values to the same variable, because if b and b' are distinct children of r, we have

$$FV(LF(b)) \cap FV(LF(b')) \subseteq Rigid(b) = Rigid(b').$$

We show that $\mathcal{M}''$ fulfills Condition (i) in Def. 8. First, $\mathcal{M}' \models r.F$ implies $\mathcal{M}'' \models r.F$, since $FV(r.F) \subseteq Rigid(r) \cup Var(r)$. Second, for all children b of r, $b.p \in Var(r)$ and hence $\mathcal{M}''(b.p) = \mathcal{M}'(b.p)$. For all children b of r such that $\mathcal{M}'(b.p) = \mathcal{M}''(b.p) = \mathsf{true}$, we know that $\mathcal{M}'_b \models LF(b)$ and hence $\mathcal{M}'' \models LF(b)$ by (†). Therefore, $\mathcal{M}'' \models LF(r)$.
We show that $\mathcal{M}''$ fulfills Condition (ii) in Def. 8. Let $b \in \mathsf{FAN}(r, \mathcal{M}'')$ be a descendant of r via a path $r \to n_1 \to \ldots \to n_q \to b$. If $q = 0$, b is a child of r, and $\mathcal{M}''(b.p) = \mathcal{M}'(b.p) = \mathsf{false}$. Since $\mathcal{M} \models \mathcal{G}$ with extension $\mathcal{M}'$, we have that $\mathcal{M}' \not\models \mathcal{G}_b$. Since $\mathcal{M}''$ is an extension of $\mathcal{M}'$, also $\mathcal{M}'' \not\models \mathcal{G}_b$ holds. If $q > 0$, n_1 is a child of r, and $\mathcal{M}''(n_1.p) = \mathcal{M}'(n_1.p) = \mathsf{true}$. Since $\mathcal{M} \models \mathcal{G}$ with extension $\mathcal{M}'$, we have that $\mathcal{M}' \models \mathcal{G}_{n_1}$. By induction hypothesis, $\mathcal{M}' \models_{la} \mathcal{G}_{n_1}$ with some extension $\mathcal{M}'_{n_1}$. By Def. 8 applied to n_1, we have that $\mathcal{M}'_{n_1} \not\models_{la} \mathcal{G}_b$. By (†), $\mathcal{M}''$ is an extension of $\mathcal{M}'$ that interprets all the variables in $FV(LF(n_1)) \setminus Rigid(n_1)$ like $\mathcal{M}'_{n_1}$ does. Thus, also $\mathcal{M}'' \not\models_{la} \mathcal{G}_b$ holds as desired.
$\Leftarrow$) By hypothesis, $\mathcal{M} \models_{la} \mathcal{G}$, that is, there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(LF(r))$ that fulfills Def. 8. We show that $\mathcal{M}'$ fulfills Condition (i) in Def. 3: indeed, $\mathcal{M}' \models LF(r)$ implies $\mathcal{M}' \models r.F$. We show that $\mathcal{M}'$ fulfills Condition (ii) in Def. 3. For all children b of r such that $\mathcal{M}'(b.p) = \mathsf{false}$, $b \in \mathsf{FAN}(r, \mathcal{M}')$, and hence by Def. 8, $\mathcal{M}' \not\models_{la} \mathcal{G}_b$. By induction hypothesis, $\mathcal{M}' \not\models \mathcal{G}_b$. For all children b of r such that $\mathcal{M}'(b.p) = \mathsf{true}$, $\mathcal{M}' \models LF(r)$ implies $\mathcal{M}' \models LF(b)$, so that Condition (i) in Def. 8 is satisfied. Since $\mathsf{FAN}(b, \mathcal{M}') \subseteq \mathsf{FAN}(r, \mathcal{M}')$, Condition (ii) in Def. 8 is satisfied by hypothesis. Thus, $\mathcal{M}' \models_{la} \mathcal{G}_b$. By induction hypothesis, $\mathcal{M}' \models \mathcal{G}_b$. Therefore, $\mathcal{M}'$ fulfills Def. 3 and $\mathcal{M} \models \mathcal{G}$.

The OptiQSMA algorithm maintains under-approximations $n.U$ of $n.\psi$ for all nodes n, but not over-approximations. Accordingly, the main function OptiQSMA (Fig. 3) initializes only $n.U$ for all nodes n, and then calls `optiSubtreeIsSolved` (Fig. 4). This function returns SAT(U) if $\mathcal{M} \models_{la} \mathcal{G}$ and UNSAT(O) if $\mathcal{M} \not\models_{la}$

@pre: $\mathcal{M}$ is an extension of $\mathcal{M}_0$ to $\mathit{Rigid}(n)$, and $I = \forall b \in T.\ [\![b.U]\!] \subseteq [\![b.\psi]\!]$
@post: I and
$\{rv = \mathsf{UNSAT}(O)$ implies $[(\forall b \in T.\ [\![b.\psi]\!] \subseteq [\![O]\!])$ and $\mathcal{M} \not\models O]\}$ and
$\{rv = \mathsf{SAT}(U)$ implies $[(\forall b \in T.\ [\![b.U]\!] \subseteq [\![b.\psi]\!])$ and $\mathcal{M} \models U]\}$

1: **function** OPTISUBTREEISSOLVED(n, $\mathcal{M}$)
2:     **while** true **do**
3:         $L \leftarrow LF(n) \wedge \bigwedge_{n \to^+ b}(\neg b.p \Rightarrow \neg b.U)$
4:         $\mathcal{M}' \leftarrow \mathsf{SMA}(L,\ \mathcal{M})$
5:         **if** $\mathcal{M}' = \mathit{nil}$ **then**
6:             **return** $\mathsf{UNSAT}(\mathsf{MBO}(L,\ FV(L) \setminus \mathit{Rigid}(n),\ \mathcal{M}))$
7:         **else**
8:             reasons $\leftarrow \top$
9:             **if** SOLUTIONFORALLDESCENDANTS(n, $\mathcal{M}'$, reasons) **then**
10:                 $L' \leftarrow LF(n) \wedge$ reasons
11:                 **return** $\mathsf{SAT}(\mathsf{MBU}(L',\ FV(L') \setminus \mathit{Rigid}(n),\ \mathcal{M}))$
12:
13: **function** SOLUTIONFORALLDESCENDANTS(n, $\mathcal{M}$, reasons)
14:     **for all** $b \in \mathsf{FAN}(n, \mathcal{M})$ **do**
15:         ans $\leftarrow$ OPTISUBTREEISSOLVED(b, $\mathcal{M}$)
16:         **if** ans $= \mathsf{SAT}(U)$ **then**
17:             $b.U \leftarrow b.U \vee U$
18:             **return** *false*
19:         **else if** ans $= \mathsf{UNSAT}(O)$ **then**
20:             reasons $\leftarrow$ reasons $\wedge\ (\neg b.p \Rightarrow \neg O)$
21:     **for all** $b \in \mathsf{NAN}(n, \mathcal{M})$ **do**
22:         reasons $\leftarrow$ reasons $\wedge\ b.p$
23:     **return** *true*

Fig. 4. Pseudocode of the auxiliary functions of the optiQSMA algorithm

$\mathcal{G}$. The formula U is an under-approximation of $r.\psi$ ($r = \mathit{root}(T)$) such that $\mathcal{M} \models U$. The formula O is an over-approximation of $r.\psi$ such that $\mathcal{M} \not\models O$. The main function $\mathsf{OptiQSMA}$ has no usage for U and O and merely returns *true* or *false* accordingly. Function `optisubtreeIsSolved` builds and returns under-approximations and over-approximations recursively. The reason for saving only under-approximations is practical, and will become clear after the illustration of `optisubtreeIsSolved`. This function takes a node n and a model $\mathcal{M}$ extending $\mathcal{M}_0$ to $\mathit{Rigid}(n)$ and determines whether $\mathcal{M} \models_{la} \mathcal{G}_n$, by executing a loop whose body contains the following steps:

1. Build a formula L (line 3 in Fig. 4) as the conjunction of the look-ahead formula $LF(n)$ (in lieu of $n.F$ in line 7 of Fig. 2) and a formula for every descendant b of n, denoted $n \to^+ b$ (in lieu of child as in Fig. 2).
2. Invoke the SMA function to search for an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\mathit{Var}(n)$ such that $\mathcal{M}' \models L$. For those descendants b for which $\mathcal{M}'(b.p) = \mathsf{false}$, the subformula for b in L reduces to $\neg b.U$ as in Step 2 of the description of

`subtreeIsSolved`. For those descendants b for which $\mathcal{M}'(b.p) = \mathtt{true}$, the subformula for b in L reduces to true, in agreement with the fact that over-approximations are not kept.

3. If SMA returns *nil*, `optiSubtreeIsSolved` returns $\mathsf{UNSAT}(O)$, where O is simply the outcome of applying MBO to L and $\mathcal{M}$, as over-approximations are not kept. Otherwise, there is potential for satisfaction with look-ahead. Function `optiSubtreeIsSolved` initializes the formula `reasons` to $\top$ and invokes `solutionForallDescendants` passing `reasons` by reference.
4. Function `solutionForallDescendants` considers first all descendants b in $\mathsf{FAN}(n, \mathcal{M})$, and calls `optiSubtreeIsSolved`$(b, \mathcal{M})$ for each of them. If this call returns $\mathsf{SAT}(U)$, it means that $\mathcal{M} \models_{la} \mathcal{G}_b$; `solutionForallDescendants` weakens $b.U$ by disjunction with U and returns *false*.
   If `optiSubtreeIsSolved`$(b, \mathcal{M})$ returns $\mathsf{UNSAT}(O)$, it means that $\mathcal{M} \not\models_{la} \mathcal{G}_b$, and we move on to the next descendant in $\mathsf{FAN}(n, \mathcal{M})$. Prior to that, `reasons` is strengthened by conjunction with $\neg b.p \Rightarrow \neg O$. For all descendants b in $\mathsf{NAN}(n, \mathcal{M})$, `solutionForallDescendants` strengthens `reasons` by conjunction with $b.p$.
5. If `solutionForallDescendants` returns *true*, `optiSubtreeIsSolved` builds formula L' as $LF(n) \wedge$ `reasons`, and returns $\mathsf{SAT}(U)$, where U is the outcome of the application of MBU to L' and $\mathcal{M}$. Otherwise, the control returns to line 3. Since `solutionForallDescendants` returned *false*, it means that it found a node b in $\mathsf{FAN}(n, \mathcal{M})$ for which `optiSubtreeIsSolved`$(b,\mathcal{M})$ returned $\mathsf{SAT}(U)$ and the formula $b.U$ was updated (line 17). Therefore the state has changed, variable L gets a new formula on line 3, and the subsequent call to SMA will not produce the same model.

In the experiments it turned out that storing over-approximations for all nodes is less efficient than using them to compute L' and then forget them. Thus, the over-approximation O encapsulated in the $\mathsf{UNSAT}(O)$ value returned by a recursive call to `optiSubtreeIsSolved` is used to build the temporary formula `reasons`, but it is not saved, and `reasons` is used to compute L'.

Theorem 5. *The function* `optiSubtreeIsSolved` *is partially correct: if the preconditions hold and the function halts, then the postconditions hold.*

Proof. Consider a call `optiSubtreeIsSolved`$(n, \mathcal{M})$. We assume that the preconditions hold and the call terminates, and we show that the postconditions hold. The proof is by structural induction on the tree T_n in $\mathcal{G}_n$.
Base case: n is a leaf. Formula L is assigned $LF(n) = n.F$ since n has no children, and SMA is invoked to find an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(n.F)$ such that $\mathcal{M}' \models n.F$. If no such extension is found, $\mathsf{MBO}(n.F, FV(n.F) \setminus Rigid(n), \mathcal{M})$ returns a quantifier-free formula O and the function returns $\mathsf{UNSAT}(O)$ on line 6 in Fig. 4. By Def. 2, $n.\psi = \exists \bar{x}.(n.F)$. By the specification of MBO, we have $\mathcal{M} \not\models O$ and $[\![n.\psi]\!] \subseteq [\![O]\!]$, so that the postconditions hold.
If SMA returns an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(n.F)$ such that $\mathcal{M}' \models n.F$, `reasons` is assigned $\top$. Since n has no descendants, `solutionForallDescendants` returns *true* leaving `reasons` unchanged. Thus, L' is assigned $LF(n) = n.F$,

$\mathsf{MBU}(n.F, FV(n.F) \setminus \mathit{Rigid}(n), \mathcal{M})$ returns a quantifier-free formula U and the function returns $\mathsf{SAT}(U)$ on line 11. By the specification of MBU, we have $\mathcal{M} \models U$ and $[\![U]\!] \subseteq [\![n.\psi]\!]$, so that the postconditions hold.
Induction hypothesis: for all descendants b of node n, if the preconditions are satisfied and `optiSubtreeIsSolved`$(b, \mathcal{M})$ halts, the postconditions are satisfied.
Induction step: By induction hypothesis, for all descendants b of n, formula $b.U$ is an under-approximation of $b.\psi$. Indeed, when $b.U$ is updated on line 17 of Fig. 4, it gets $b.U \vee U$, where U is an under-approximation of $b.\psi$ returned by a recursive call `optiSubtreeIsSolved`$(b, \mathcal{M}')$. We distinguish two cases for the two exit points of `optiSubtreeIsSolved` (see Fig. 4).

– Suppose `optiSubtreeIsSolved`$(n, \mathcal{M})$ returns $\mathsf{UNSAT}(O)$ on line 6, because SMA could not extend $\mathcal{M}$ to a model of

$$L = LF(n) \wedge \bigwedge_{n \to^+ b} (\neg b.p \Rightarrow \neg b.U).$$

We must show that $[\![n.\psi]\!] \subseteq [\![O]\!]$ and $\mathcal{M} \not\models O$. The latter is directly a consequence of O being generated by MBO from L and $\mathcal{M}$. For the former, let $\mathcal{M}_O$ be a model such that $\mathcal{M}_O \models n.\psi$. It follows that $\mathcal{M}_O \models \mathcal{G}_n$ by Thm. 1 and $\mathcal{M}_O \models_{la} \mathcal{G}_n$ by Thm. 4. By Def. 8, $\mathcal{M}_O$ can be extended into a model $\mathcal{M}'_O$ of $LF(n)$ such that for all $b \in \mathsf{FAN}(n, \mathcal{M}'_O)$, $\mathcal{M}'_O \not\models_{la} \mathcal{G}_b$. It follows that $\mathcal{M}'_O \not\models b.\psi$ by Thm. 1, and hence $\mathcal{M}'_O \not\models b.U$ by pre-condition I, so that $\mathcal{M}'_O \models \neg b.U$ (*).
Now we have that $\mathcal{M}'_O \models LF(n)$ and we want to show that $\mathcal{M}'_O \models L$. To this end, we assume that $\mathcal{M}'_O(c.p) = \mathsf{true}$ for all descendants c of n beyond the first alternation nodes, that is, for all nodes c such that $n \to^+ c$ and $c \notin \mathsf{NAN}(n, \mathcal{M}'_O) \cup \mathsf{FAN}(n, \mathcal{M}'_O)$ (†).
We show that this assumption causes no loss of generality. Indeed, forcing $\mathcal{M}'_O(c.p) = \mathsf{true}$ for such nodes affects neither $\mathsf{NAN}(n, \mathcal{M}'_O)$, nor $\mathsf{FAN}(n, \mathcal{M}'_O)$, nor $\mathcal{M}'_O(b.p) = \mathsf{false}$ for all $b \in \mathsf{FAN}(n, \mathcal{M}'_O)$. Also, this assumption does not affect the fact that $\mathcal{M}'_O \models LF(n)$. Indeed, $LF(n)$ has the form:

$$n.F \wedge \bigwedge_{n \to^+ b} \{n_1.p \Rightarrow \cdots \Rightarrow n_q.p \Rightarrow b.F \mid n \to n_1 \to \cdots \to n_q \to b\}.$$

Therefore, forcing $\mathcal{M}'_O(c) = \mathsf{true}$ for every node c that is below some node $b \in \mathsf{FAN}(n, \mathcal{M}'_O)$ does not affect the truth value of $LF(n)$, because $\mathcal{M}'_O(b.p) = \mathsf{false}$ and hence any implication in $LF(n)$ involving c necessarily evaluates to true.
Next, $\mathcal{M}'_O$ satisfies L, because it satisfies $LF(n)$ and also $\neg b.p \Rightarrow \neg b.U$ for all descendants b of n: if $b \in \mathsf{FAN}(n, \mathcal{M}'_O)$ then $\mathcal{M}'_O(b.p) = \mathsf{false}$ and we know that $\mathcal{M}'_O \models \neg b.U$ by (*); if $b \in \mathsf{NAN}(n, \mathcal{M}'_O)$ then $\mathcal{M}'_O(b.p) = \mathsf{true}$, so that $\mathcal{M}'_O(\neg b.p \Rightarrow \neg b.U) = \mathsf{true}$; and if $b \notin \mathsf{NAN}(n, \mathcal{M}'_O) \cup \mathsf{FAN}(n, \mathcal{M}'_O)$, then $\mathcal{M}'_O(b.p) = \mathsf{true}$ by the assumption (†), so that $\mathcal{M}'_O(\neg b.p \Rightarrow \neg b.U) = \mathsf{true}$.
Since O is generated by MBO from L and $\mathcal{M}$, by the specification of MBO we know that L implies O in the theory. Since $\mathcal{M}'_O$ and hence $\mathcal{M}_O$ satisfies

L, it follows that $\mathcal{M}_O$ also satisfies O. Therefore, also the postcondition $[\![n.\psi]\!] \subseteq [\![O]\!]$ holds.

– Suppose `optiSubtreeIsSolved`$(n, \mathcal{M})$ returns $\mathsf{SAT}(U)$ on line 11. Function SMA found an extension $\mathcal{M}'$ satisfying L, and hence $LF(n)$. Furthermore, `solutionForallDescendants` returned *true* after constructing a formula

$$\texttt{reasons} \quad = \quad (\textstyle\bigwedge_{b \in \mathsf{NAN}(n,\mathcal{M}')} b.p) \wedge (\bigwedge_{b \in \mathsf{FAN}(n,\mathcal{M}')} (\neg b.p \Rightarrow \neg O_b))$$

where, for all $b \in \mathsf{FAN}(n, \mathcal{M}')$, O_b is an over-approximation of $b.\psi$ that was returned as $\mathsf{UNSAT}(O_b)$ by a recursive call `optiSubtreeIsSolved`$(b, \mathcal{M}')$. By the post-condition of that recursive call, $\mathcal{M}' \not\models O_b$. By Thm. 1, $\mathcal{M}' \not\models \mathcal{G}_b$. Since this holds for all $b \in \mathsf{FAN}(n, \mathcal{M}')$, we have that $\mathcal{M}'$ fulfills Def. 8.

We show that this property holds in general: every model that satisfies $L' = (LF(n) \wedge \texttt{reasons})$ fulfills Def. 8. To this end, we show that every model that satisfies `reasons` fulfills Condition (ii) in Def. 8. Let $\mathcal{M}''$ be a model that satisfies `reasons`. It follows that $\mathsf{NAN}(n, \mathcal{M}'') = \mathsf{NAN}(n, \mathcal{M}')$, $\mathsf{FAN}(n, \mathcal{M}'') = \mathsf{FAN}(n, \mathcal{M}')$, and for all $b \in \mathsf{FAN}(n, \mathcal{M}')$, $\mathcal{M}'' \models \neg O_b$. It follows that $\mathcal{M}'' \not\models O_b$ and hence $\mathcal{M}'' \not\models b.psi$, so that $\mathcal{M}'' \not\models \mathcal{G}_b$ by Thm. 1. Thus, $\mathcal{M}'$ fulfills Condition (ii) of Def. 8.

By the specification of MBU, the application of MBU to L' and $\mathcal{M}$ yields a quantifier-free formula U such that $\mathcal{M} \models U$, and for all models $\mathcal{M}_U \in [\![U]\!]$, $\mathcal{M}_U$ can be extended into a model that satisfies L', and hence fulfills Def. 8 by the argument above. This means that for all models $\mathcal{M}_U \in [\![U]\!]$, $\mathcal{M}_U \models_{la} \mathcal{G}_n$, and hence by Thm. 4 $\mathcal{M}_U \models \mathcal{G}_n$, and hence $\mathcal{M}_U \models n.\psi$ by Thm. 1. This shows that $[\![n.U]\!] \subseteq [\![n.\psi]\!]$, so that the postconditions hold.

The proof of partial correctness of `optiSubtreeIsSolved` shows that every model that satisfies $L' = (LF(n) \wedge \texttt{reasons})$ fulfills Definition 8. In this sense, `reasons` is an explanation of why a model is found with look-ahead.

Theorem 6. *If the* MBU *and* MBO *functions have finite basis, whenever the preconditions are satisfied the function* `optiSubtreeIsSolved` *halts.*

Proof. For all nodes n in T, where $\bar{z}_n = Rigid(n)$, we build

– A finite set $\mathcal{U}_n = \{U_1^n[\bar{z}_n], \ldots, U_{l_n}^n[\bar{z}_n]\}$ of under-approximations of $n.\psi$, and
– A finite set $\mathcal{O}_n = \{O_1^n[\bar{z}_n], \ldots, O_{m_n}^n[\bar{z}_n]\}$ of over-approximations of $n.\psi$,

such that

(i) The property "*For all descendants b of n, $b.U$ is the disjunction of a subset of $\mathcal{U}_b$*" (*) is an invariant of `optiSubtreeIsSolved`$(n, \mathcal{M})$, where $\mathcal{M}$ extends $\mathcal{M}_0$ to $\bar{z}_n$ (see Fig. 4), and

(ii) If Property (*) holds as a pre-condition of `optiSubtreeIsSolved`$(n, \mathcal{M})$, then the call halts returning either $\mathsf{SAT}(U)$ for some $U \in \mathcal{U}_n$ or $\mathsf{UNSAT}(O)$ for some $O \in \mathcal{O}_n$.

The construction of the sets $\mathcal{U}_n$ and $\mathcal{O}_n$ and the proof are by induction on T_n. Consider a call `optiSubtreeIsSolved`$(n, \mathcal{M})$ with Property (*) holding as a pre-condition. We show simultaneously that

1. Property (*) holds throughout the execution of both `optiSubtreeIsSolved` and `solutionForallDescendants`, and

2. Both functions terminate.

The first claim involves showing that Property (*) is an invariant for both the `while` loop in `optiSubtreeIsSolved` and the `for` loop ranging over $b \in \mathsf{FAN}(n, \mathcal{M})$ in `solutionForallDescendants`.
Consider a call `optiSubtreeIsSolved`$(b, \mathcal{M})$ in the `for` loop.
Assume that Property (*) holds at that point as a loop invariant and the preconditions of `optiSubtreeIsSolved`$(b, \mathcal{M})$ are satisfied. Then by induction hypothesis `optiSubtreeIsSolved`$(b, \mathcal{M})$ terminates, returning either $\mathsf{SAT}(U)$ for some $U \in \mathcal{U}_b$ or $\mathsf{UNSAT}(O)$ for some $O \in \mathcal{O}_b$. Whenever $b.U$ is updated by instruction $b.U \leftarrow b.U \vee U$ (line 17 of Fig. 4), formula $b.U$ remains a disjunction of a subset of $\mathcal{U}_b$, so that the loop invariant (*) is preserved. Therefore, (*) is an invariant of `solutionForallDescendants` and `solutionForallDescendants` terminates. Since no instruction in the body of the `while` loop modifies $b.U$, Property (*) is an invariant also for the `while` loop.
We prove next the termination of the `while` loop. The invariant (*) implies that for all descendants b of n, formula $b.U$ can be updated at most $|\mathcal{U}_b|$ times, where $|\mathcal{U}_b|$ is the cardinality of the set $\mathcal{U}_b$. By way of contradiction, suppose that the `while` loop does not halt. This means that it generates an infinite series of calls to `solutionForallDescendants` each returning *false*. Since each call to `solutionForallDescendants` that returns *false* updates some $b.U$ at least once, such an infinite series contradicts the fact that for each node b only finitely many updates to $b.U$ are available.
To complete the proof, we observe that the space of possible values for $b.U$ is finite, because $b.U$ is a disjunction of a subset of $\mathcal{U}_b$. To be precise, if $|\mathcal{U}_b| = v_b$, the number of possible values for $b.U$ is $\sum_{i=0}^{v_b} \binom{v_b}{i} = \sum_{i=0}^{v_b} \frac{v_b!}{i! \cdot (v_b - i)!}$. This implies that also the space of possible values for L is finite. Therefore, MBO gets applied to finitely many formulas and for each of them it can produce only finitely many formulas by the finite basis hypothesis. This guarantees the existence of the finite set $\mathcal{O}_n$.
By the finiteness of $\mathcal{O}_n$, also the space of possible values for the variable `reasons` is finite. This implies that also the space of possible values for L' is finite. Therefore, MBU gets applied to finitely many formulas and for each of them it can produce only finitely many formulas by the finite basis hypothesis. This guarantees the existence of the finite set $\mathcal{U}_n$ and concludes the proof.

6 The YicesQS Solver and Experimental Results

The $\mathsf{OptiQSMA}$ algorithm is implemented in YicesQS to equip Yices 2 with support for quantifiers for complete theories (unrelated to Yices 2 support for quantifiers in UF).[1] MBO is available as model interpolation from Yices's MC-SAT [10] solver for quantifier-free formulas, including theory-specific techniques for bitvectors (BV) [15] and arithmetic. The latter are based on NLSAT [16] and ultimately on Cylindrical Algebraic Decomposition (CAD). Basic MBU is done

[1] See https://github.com/disteph/yicesQS and https://yices.csl.sri.com/.

BV.		
CVC5	854/970	25,584s
Q3B	835/970	13,510s
Z3	775/970	7,712s
Bitwuzla	759/970	15,572s
Q3B-pBDD	754/970	15,553s
YicesQS	708/970	3,862s
Ultim.Elim.	304/970	4,204s

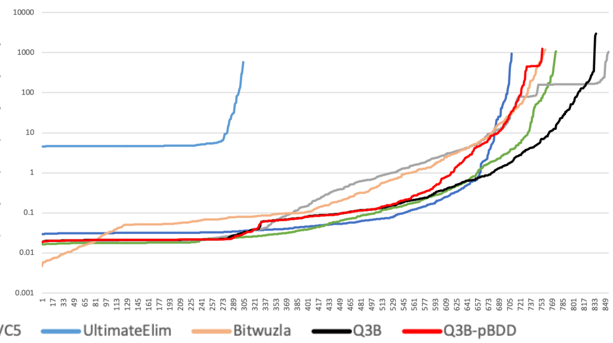


Fig. 5. Plot for BV.

as generalization-by-substitution [12] and improved with *model-based projection* (e.g., [18]) for arithmetic, and *invertibility conditions* [21], including ϵ-terms, for BV. In YicesQS model-based projection also is based on CAD.

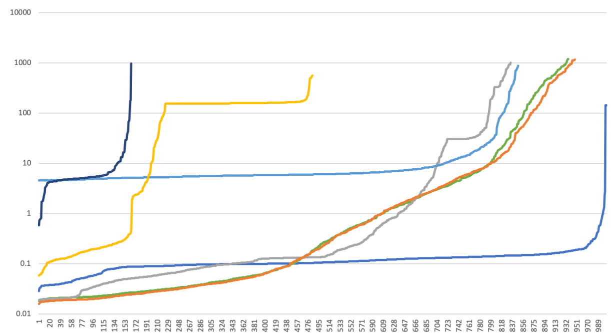
In the 2022 SMT competition, YicesQS entered the single-query, non-incremental tracks of BV, LRA, LIA, NRA, and NIA (nonlinear integer arithmetic). The experiments were run on the Starexec cluster with a 20 min timeout per benchmark and 60GB of memory. The benchmarks were a subset of the SMT-LIB collection. The results presented below were computed by running the competition script `join.sh` on the raw data from StarExec,[2] sorting the data, and producing the plots that are available online.[3] A description of the participating solvers can be found on the competition website.[4]

Figure 5 shows the results for BV, where YicesQS solved quickly a high number of benchmarks (compared for example with CVC5), but was not outstanding, possibly because YicesQS 2022 makes a limited use of invertibility conditions for model interpolation. Figure 6 shows the results for the four arithmetics. The columns on the left list number of solved instances and time to solve them for each logic and solver. In the plot on the right, each color corresponds to a solver and point (x, y) of that color means that the x^{th} fastest-solved benchmark was solved by that solver in time y (log scale). 2021 Z3 is included because in some of these logics it performed slightly better than 2022 Z3. The logic where YicesQS performed best is LRA: it was the only solver to solve all 1,003 benchmarks. Z3 2021 was second best, solving 948 benchmarks with a total runtime about 100 times higher. YicesQS has neither a special treatment (e.g., simplex-based) of linear problems, nor integer-specific techniques: it relies on CAD-based techniques for MBU and MBO also for integer problems. Thus, it is somewhat average on LIA and NIA. These two theories are undecidable (NRA due to division by 0) and hence they lie outside of the theoretical framework of QSMA. YicesQS answers should still be correct, but termination can be lost. With Z3 being a

[2] https://github.com/SMT-COMP/smt-comp/tree/master/2022/results
[3] http://www.csl.sri.com/users/sgl/Work/Cade2023-data/index.html
[4] https://smt-comp.github.io/2022/participants.html

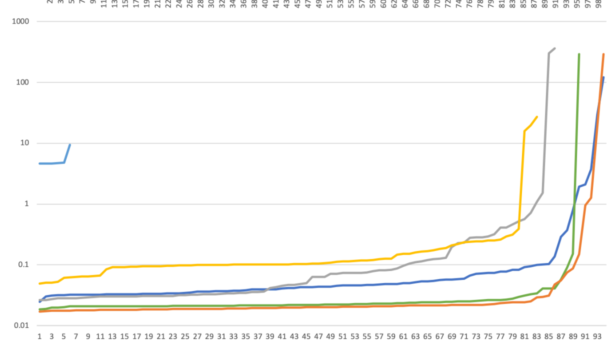
LRA.

YicesQS	1003/1003	414s
Z3 2021	948/1003	41,068s
Z3	936/1003	41,240s
Ultim.Elim.	847/1003	16,136s
CVC5	834/1003	21,197s
Vampire	484/1003	45,326s
SMTInterpol	164/1003	2,584s



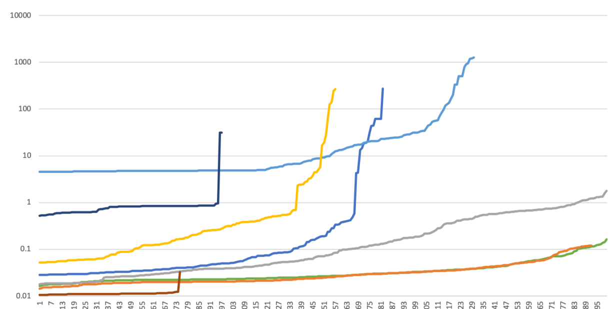
NRA.

YicesQS	94/99	165s
Z3 2021	94/99	315s
Z3	90/99	294s
CVC5	86/99	672s
Vampire	83/99	73s
Ultim.Elim.	6/99	33s



LIA.

Z3	300/300	11s
CVC5	300/300	78s
Z3 2021	292/300	10s
Ultim.Elim.	230/300	11,789s
YicesQS	182/300	750s
Vampire	157/300	985s
SMTInterpol	97/300	134s
VeriT	75/300	1s



NIA.

CVC5	190/208	3,642s
Ultim.Elim.	129/208	701s
Z3	88/208	317s
Z3 2021	87/208	53s
YicesQS	80/208	290s
Vampire	66/208	13,744s

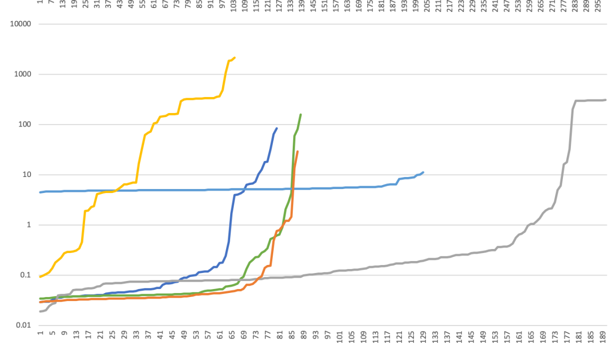


Fig. 6. Plots for the four arithmetics.

non-competing participant in the SMT 2022 competition, YicesQS came second for *Largest Contribution* (single queries), because of its overall performance in the four arithmetics, where it also came first for satisfiable instances and in the 24 sec timeout setup (instead of 20 min).

7 Discussion: Related Work and Future Work

Quantified SMT was approached by a procedure with an $\exists$-solver and a $\forall$-solver for prenex normal form formulas with $\exists\forall$ prefix [12]. A formulation as a game between an $\exists$-player and a $\forall$-player appeared with the *QSAT algorithm* [3] for prenex normal form formulas with $(\exists\forall)^+$ prefix. QSMA accepts arbitrary formulas with quantifiers in arbitrary positions.

Both QSAT and QSMA work for a generic theory $\mathcal{T}$ over basic $\mathcal{T}$-specific components. QSAT uses *model-based projection* [3,18] and a solver for quantifier-free satisfiability that supports UNSAT cores. Model-based projection is an instance of MBU. An UNSAT core (as a conjunction) is an MBO in the special case where the input assignment is Boolean. While MBO can produce UNSAT cores, MBO generalizes the concept of UNSAT core with theory-specific reasoning when there are *non-Boolean input assignments*, as it is the case in QSMA. It is unclear whether the combination of UNSAT cores and theory-specific MBU can emulate MBO or provide the same benefits. QSAT is implemented in Z3 and it is the default solver for LIA, LRA, and NRA.

YicesQS is a recent implementation that only participated in the SMT competition in 2021 and 2022. Directions for further development include augmenting integer reasoning, and improving model interpolation in BV by a better usage of invertibility conditions. Another lead for future work is to compose QSMA within the *CDSAT framework for conflict-driven reasoning in unions of theories* [4–6]. For this, one may need to drop the assumption that there is a unique model $\mathcal{M}_0$ and only its extensions need to be considered, which will be a generalization also in the single theory case. As most known MBU and MBO functions are for single theories, one may have to study how to get MBU and MBO functions for a union of theories from such functions for the component theories. Another issue is the interplay between QSMA's recursive descent over the QSMA-tree for the formula and CDSAT's conflict-driven search.

Acknowledgements Part of this work was done while the first and third authors were visiting SRI International, whose support is much appreciated. This material is based upon work supported by NSF with awards CCF-1816936 and CCF-1817204. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Government or NSF.

References

1. Althaus, E., Kruglov, E., Weidenbach, C.: Superposition modulo linear arithmetic SUP(LA). In: Ghilardi, S., Sebastiani, R. (eds.) Proc. FroCoS-7. LNAI, vol. 5749, pp. 84–99. Springer (2009). https://doi.org/10.1007/978-3-642-04222-5_5
2. Baumgartner, P., Waldmann, U.: Hierarchic superposition with weak abstraction. In: Bonacina, M.P. (ed.) Proc. CADE-24. LNAI, vol. 7898, pp. 39–57. Springer (2013). https://doi.org/10.1007/978-3-642-38574-2_3
3. Bjørner, N., Janota, M.: Playing with quantified satisfaction (Short paper). In: Fehnker, A., McIver, A., Sutcliffe, G., Voronkov, A. (eds.) Short Presentations at LPAR-20. EPiC Series in Computing, vol. 35, pp. 15–27. EasyChair (2015)
4. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Conflict-driven satisfiability for theory combination: transition system and completeness. J. Autom. Reason. **64**(3), 579–609 (2020). https://doi.org/10.1007/s10817-018-09510-y
5. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: CDSAT for nondisjoint theories with shared predicates: arrays with abstract length. In: Hyvärinen, A., Déharbe, D. (eds.) Proc. SMT-20. CEUR Proceedings, vol. 3185, pp. 18–37. CEUR WS-org (2022)
6. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Conflict-driven satisfiability for theory combination: lemmas, modules, and proofs. J. Autom. Reason. **66**(1), 43–91 (2022). https://doi.org/10.1007/s10817-021-09606-y
7. Bonacina, M.P., Lynch, C.A., de Moura, L.: On deciding satisfiability by theorem proving with speculative inferences. J. Autom. Reason. **47**(2), 161–189 (2011). https://doi.org/10.1007/s10817-010-9213-y
8. Bradley, A.R., Manna, Z.: The Calculus of Computation - Decision Procedures with Applications to Verification. Springer (2007). https://doi.org/10.1007/978-3-540-74113-8
9. de Moura, L., Bjørner, N.: Efficient E-matching for SMT-solvers. In: Pfenning, F. (ed.) Proc. CADE-21. LNAI, vol. 4603, pp. 183–198. Springer (2007). https://doi.org/10.1007/978-3-540-73595-3_13
10. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) Proc. VMCAI-14. LNCS, vol. 7737, pp. 1–12. Springer (2013). https://doi.org/10.1007/978-3-642-35873-9_1
11. Detlefs, D.L., Nelson, G., Saxe, J.B.: Simplify: a theorem prover for program checking. J. ACM **52**(3), 365–473 (2005). https://doi.org/10.1145/1066100.1066102
12. Dutertre, B.: Solving exists/forall problems with Yices. In: Proc. SMT-13 (2015)
13. Ge, Y., Barrett, C., Tinelli, C.: Solving quantified verification conditions using satisfiability modulo theories. In: Pfenning, F. (ed.) Proc. CADE-21. LNAI, vol. 4603, pp. 167–182. Springer (2007). https://doi.org/10.1007/978-3-540-73595-3_12
14. Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiability modulo theories. In: Bouajjani, A., Maler, O. (eds.) Proc. CAV-21. LNCS, vol. 5643, pp. 306–320. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_25
15. Graham-Lengrand, S., Jovanović, D., Dutertre, B.: Solving bitvectors with MC-SAT: explanations from bits and pieces. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Proc. IJCAR-10. LNAI, vol. 12166, pp. 103–121. Springer (2020). https://doi.org/10.1007/978-3-030-51074-9_7
16. Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) Proc. IJCAR-6. LNAI, vol. 7364, pp. 339–354. Springer (2012). https://doi.org/10.1007/978-3-642-31365-3_27

17. Jovanović, D., Dutertre, B.: Interpolation and model checking for nonlinear arithmetic. In: Silva, A., Leino, K.R.M. (eds.) Proc. CAV-35. LNCS, vol. 12760, pp. 266–288. Springer (2021). https://doi.org/10.1007/978-3-030-81688-9_13
18. Komuravelli, A., Gurfinkel, A., Chaki, S.: SMT-based model checking for recursive programs. Form. Methods Syst. Des. **48**(3), 175–205 (2016). https://doi.org/10.1007/s10703-016-0249-4
19. Korovin, K., Voronkov, A.: Integrating linear arithmetic into superposition calculus. In: Duparc, J., Henzinger, T.A. (eds.) Proc. CSL-16. LNCS, vol. 4646, pp. 223–237. Springer (2007). https://doi.org/10.1007/978-3-540-74915-8_19
20. Moskal, M.: Fx7 or in software, it is all about quantifiers. System Descriptions at SMT-COMP (2007), http://smtcomp.cs.uiowa.edu/2007/descriptions/fx7.pdf
21. Niemetz, A., Preiner, M., Reynolds, A., Barrett, C., Tinelli, C.: Solving quantified bit-vectors using invertibility conditions. In: Chockler, H., Weissenbacher, G. (eds.) Proc. CAV-30. LNCS, vol. 10982, pp. 236–255. Springer (2018). https://doi.org/10.1007/978-3-319-96142-2_16
22. Reynolds, A., Tinelli, C., de Moura, L.: Finding conflicting instances of quantified formulas in SMT. In: Claessen, K., Kuncak, V. (eds.) Proc. FMCAD 2014. pp. 195–202. ACM and IEEE (2014), https://dl.acm.org/doi/10.5555/2682923.2682957