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
[2] SRI International, USA
[3] École Normale Supérieure, Paris, France

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 2 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., [19]). Therefore, endowing SMT solvers with quantifier reasoning (e.g., [10, 8, 12, 13, 21, 11, 3]), enriching first-order theorem provers with built-in theories (e.g., [18, 1, 2]), and integrating provers and solvers [6], 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) [7]), QE is not a practical solution.

In this paper we propose a practical solution where the computation of quantifier-free under- and over-approximations of quantified formulas embodies a lazy approach to QE that is tailored to determining $\mathcal{T}$-satisfiability. It takes the form of a new algorithm, called QSMA, that assumes that $\mathcal{T}$ is *complete*. By its recursive nature, the QSMA algorithm takes as input a generalized form of the satisfiability problem, namely *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.

After the preliminaries (Sect. 2), the problem is formalized as a *satisfiability game* (Sect. 3). The core of the paper describes the QSMA algorithm (Sect. 4) and an optimized variant (Sect. 5) both proved to be totally correct. The variant is implemented in the YicesQS solver built on top of Yices 2. Data about YicesQS' performance at the 2022 SMT competition are presented (Sect. 6).

1.1 High-Level View of the QSMA Algorithm

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. Our approach sees φ as a *game* between an $\exists$-player and an $\forall$-player, who take turns choosing values for the $\bar{x}_i$ trying, respectively, to satisfy and to refute the formula F. For example, suppose the theory is LRA, $\varphi = \exists x.\forall y.\exists z.F$ and $F = z \geq 0 \wedge x \geq 0 \wedge y + z \geq 0$. The $\exists$-player chooses a value for x, then the $\forall$-player chooses a value for y, and last the $\exists$-player chooses a value for z. If at the end F is true, the $\exists$-player wins, otherwise the $\forall$-player wins. Say that the $\exists$-player chooses $x{\leftarrow}0$. Whatever value the $\forall$-player chooses for y, the $\exists$-player can win by picking $z{\leftarrow}max(0, -y)$. Indeed, φ is true in LRA. If $F = z \geq 0 \wedge x \geq 0 \wedge y + z \leq 0$ no matter what the $\exists$-player chooses for x, the $\forall$-player can win with $y{\leftarrow}1$. Indeed, φ is false in LRA.

A formula $\varphi = \exists x.((\forall y_1.F_1[x, y_1]) \Rightarrow (\forall y_2.F_2[x, y_2]))$, where F_1 and F_2 are quantifier-free, can still be seen as a game between two players A and B. Player A tries to find a value for x such that $\neg(\forall y_1.F_1) \vee (\forall y_2.F_2)$, that is, a value for which either A can find a value of y_1 satisfying $\neg F_1$ (see $\neg(\forall y_1.F_1)$ as $\exists y_1.\neg F_1$), or B cannot find a value of y_2 satisfying $\neg F_2$ (see $\forall y_2.F_2$ as $\neg\exists y_2.\neg F_2$). Without loss of generality ($\neg\neg$ converts $\exists$ into $\neg\forall\neg$ and $\forall$ into $\neg\exists\neg$), we consider formulas

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

where $F[\bar{z}, \bar{x}, \bar{p}]$ denotes a quantifier-free formula where $\bar{z}$, $\bar{x}$, and $\bar{p}$ occur, $\bar{z}$ and $\bar{x}$ are tuples of first-order variables occurring free in F, and $\bar{p}$ is a tuple of Boolean variables $p_1, \ldots p_n$ proxies for the universally quantified subformulas $\varphi_i = \forall\bar{y}_i.G_i[\bar{z}, \bar{x}, \bar{y}_i]$. Given an initial assignment to the free variables $\bar{z}$, we construct a two-player game $\mathcal{G}$ where player A claims that φ is true under the assignment and player B claims the opposite. A player wins iff their claim is correct. The variables $\bar{z}$ are called *rigid*, because their assignments do not change during the game. The game starts with A trying to satisfy $F[\bar{z}, \bar{x}, \bar{p}]$. If A fails, B wins. If A succeeds and $n = 0$, A wins. If A succeeds and $n > 0$, the opponent B challenges the model found by A by choosing one of the p_i, and playing a game $\mathcal{G}_i$ corresponding to $\neg\varphi_i = \exists\bar{y}_i.\neg G_i[\bar{z}, \bar{x}, \bar{y}_i]$. If A assigned true to p_i, B plays first in $\mathcal{G}_i$, otherwise A plays first. Player A wins if she is able to win all of B's possible challenges. For this, A's model should satisfy $F[\bar{z}, \bar{x}, \bar{p}] \wedge \bigwedge_{i=1}^n (p_i \Leftrightarrow \varphi_i)$. Therefore, A wins if and only if φ is true under the initial assignment to $\bar{z}$.

1.2 Related Work

The idea of formulating quantified SMT as a two-player game appeared in [11] for the $\exists\forall$ instance in prenex normal form, and in the QSAT algorithm [3] for the more general $(\exists\forall)^+$ instance in prenex normal form. QSMA works on a formula with quantifiers in arbitrary positions, and it decides dynamically which player must try to instantiate the outermost quantifier of a sub-formula, depending on whether the sub-formula should be true or false according to the last player. This issue does not arise with prenex normal form.

Both QSAT and QSMA work for a generic theory $\mathcal{T}$ over basic $\mathcal{T}$-specific components. QSAT uses *model-based projection* (MBP) [17], which is a weaker and asymmetric version of QE. QSMA uses *model generalization* (MG) [11], which produces model-based under-approximations. MBP is an instance of MG. QSAT then uses a solver for quantifier-free satisfiability that supports UNSAT cores. In contrast, QSMA uses a solver for quantifier-free satisfiability modulo input assignments (SMA), supporting also *model interpolation* (MI) [16], which is dual to MG as it produces over-approximations (*model interpolants*). UNSAT cores (as conjunctions) are a special case of model interpolants when the input assignment is Boolean. While model interpolation can be instrumented to produce UNSAT cores, it is more general: it generalizes UNSAT cores 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 MG can emulate the generation of model interpolants or provide the same benefits. QSAT is implemented in Z3 and we understand it is the default solver for the SMT logics LIA, LRA, NRA (nonlinear real arithmetic). Sect. 6 includes the evaluation of both solvers YicesQS and Z3 on those logics.

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, φ, ψ for formulas, F, G for quantifier-free formulas, $\mathcal{M}$ for interpretations, $=$ for identity, $\uplus$ for disjoint union, and $\setminus$ for set difference. $FV(\varphi)$ is the set of the variables occurring free in φ. Implication is $\Rightarrow$ and logical equivalence is $\Leftrightarrow$. 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 $\mathcal{M}_0$. Therefore $\mathcal{T}$ is complete, for Σ-sentences $\mathcal{T}$-validity, $\mathcal{T}$-satisfiability, and truth in $\mathcal{M}_0$ coincide, and all $\mathcal{T}[\mathcal{V}]$-models are extensions of $\mathcal{M}_0$. 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 input problem is a formula $\varphi = \exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]\{p_i \leftarrow \forall\bar{y}_i.G_i(\bar{z},\bar{x},\bar{y}_i)\}_{i=1}^n$ as in the introduction. Such a formula φ is $\mathcal{T}$-satisfiable if there exists an assignment of values to the variables in $\bar{z}$ such that φ is true in $\mathcal{M}_0$. The quantified SMA problem is the question of whether φ is true in $\mathcal{M}_0$ when given an assignment of values to the variables in $\bar{z}$.

3 Satisfiability Games

A satisfiability game is a two-player game represented as a finite labeled tree T. On each level of T the nodes represent the available moves. Each node a in T is the root of a subtree T_a that represents a subgame. The label of a contains the information about subgame T_a. The *ancestors* of a are the nodes on the unique path from the root of T, denoted $\mathit{root}(T)$, to node a.

Definition 1 (Satisfiability Game). *A* satisfiability game *is a pair* $\mathcal{G} = (\bar{z}, T)$*, where* $\bar{z}$ *is a tuple of variables, called the* rigid variables *of the game, and* T *is a finite labeled tree, called* game tree*, such that for all its nodes* a*:*

- *The label of* a *is a pair* $(\bar{x}, F)$*, where* $\bar{x}$ *is a tuple of new first-order variables, called the* local (free) variables *of* a*, and* F *is a quantifier-free formula;*
- *Every outgoing arc from node* a *to a child* b *is labeled with a new Boolean variable* $b.p$ *that is the* proxy *of* b*; and*
- *If* $p_1, \ldots, p_n$ *are the proxies of the children of* a*, and* $\bar{x}_1, \ldots, \bar{x}_m$ *are the local variables of the ancestors of* a*, then* $FV(F) \subseteq \{\bar{z}, \bar{x}_1, \ldots, \bar{x}_m, \bar{x}, p_1, \ldots, p_n\}$*, and for all* i*,* $1 \le i \le n$*,* p_i *occurs once and only once in* F*.*

For a node a with label $(\bar{x}, F)$, let $a.\bar{x}$, and $a.F$ denote the components of the label. If $p_1, \ldots, p_n$ are the proxies of a's children and $a_1 = (\bar{x}_1, \ldots), \ldots, a_m = (\bar{x}_m, \ldots)$ are a's ancestors, the set of the *free variables at* a is $\mathit{Var}(a) = \bar{x} \uplus \{p_1, \ldots, p_n\}$ and the set of the *rigid variables at* a is $\mathit{Rigid}(a) = \bar{z} \uplus \bar{x}_1 \uplus \ldots \uplus \bar{x}_m$. Thus, $FV(a.F) \subseteq \mathit{Rigid}(a) \cup \mathit{Var}(a)$, $\mathit{Rigid}(\mathit{root}(T)) = \bar{z}$, and the subgame rooted at node a is $\mathcal{G}_a = (\mathit{Rigid}(a), T_a)$. The *first player in a game* is the one who moves first in that game. A move for the first player at node a consists of assigning values to the variables in $\mathit{Var}(a)$. A move for the second player at node a consists of choosing a child b of a to challenge the assignment. The assignment to $b.p \in \mathit{Var}(a)$ determines whether the first player in $\mathcal{G}_b$ is the same as (if $b.p \leftarrow \mathsf{false}$) or different from (if $b.p \leftarrow \mathsf{true}$) the first player in $\mathcal{G}_a$.

Definition 2 (Winning). *For all games* $\mathcal{G} = (\bar{z}, T)$ *with* $r = \mathit{root}(T)$ *and extensions* $\mathcal{M}$ *of* $\mathcal{M}_0$ *to* $\mathit{Rigid}(r) = \bar{z}$*, the first player in* $\mathcal{G}$ wins from $\mathcal{M}$*, 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* a *of* r*,* $\mathcal{M}'(a.p) = \mathsf{false}$ *iff the first player in* $\mathcal{G}_a$ *wins from* $\mathcal{M}'$*.*

Game $\mathcal{G}$ is *winning* for model $\mathcal{M}$ iff the first player in $\mathcal{G}$ wins from $\mathcal{M}$. Otherwise, $\mathcal{G}$ is *losing* for $\mathcal{M}$.

Definition 3 (Game for a formula). *If $\varphi = \exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]\{p_i \leftarrow \forall\bar{y}_i.G_i[\bar{z},\bar{x},\bar{y}_i]\}$, where $i = 1,\dots,n$, $FV(\varphi) = \bar{z}$, and $\varphi_i = \forall\bar{y}_i.G_i[\bar{z},\bar{x},\bar{y}_i]$, the* game for φ *is the satisfiability game $\mathcal{G} = (\bar{z},T)$, where T is defined inductively as follows:*

- *If $n = 0$, T consists of a single node a labeled $(\bar{x},F[\bar{z},\bar{x}])$;*
- *If $n > 0$, for all i, $1 \le i \le n$, let $\mathcal{G}_i = ((\bar{z},\bar{x}),T_i)$ be the game for $\neg\varphi_i$, where $root(T_i)$ is a node b_i labeled $(\bar{y}_i,\neg G_i[\bar{z},\bar{x},\bar{y}_i])$. Then T is the tree with a new node a labeled $(\bar{x},F[\bar{z},\bar{x},\bar{p}])$ as root, n outgoing arcs labeled $p_1,\dots,p_n$, and $b_1,\dots,b_n$ as children.*

Since a game is defined for a formula with outermost existential quantifier, every subformula φ_i with outermost universal quantifier is negated so that its subgame $\mathcal{G}_i$ can be defined inductively. Given φ and an extension $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$, two players A and B play the game $\mathcal{G}$ for φ with the goal of showing φ true or false, respectively. Player A begins and assigns values to $Var(a)$. If A assigns false to p_i, A's aim is to show that $\varphi_i = \forall\bar{y}_i.G_i[\bar{z},\bar{x},\bar{y}_i]$ is false, as p_i stands for φ_i. Therefore, A plays first in $\mathcal{G}_i$, to show that $\neg\varphi_i = \exists\bar{y}_i.\neg G_i[\bar{z},\bar{x},\bar{y}_i]$ is true. If A assigns true to p_i, the first player in $\mathcal{G}_i$ is B, whose aim is to challenge the truth of φ_i, by showing that $\neg\varphi_i$ is true.

Example 1. The game tree for $\varphi = \exists x.((\forall y_1.F_1[x,y_1]) \Rightarrow (\forall y_2.F_2[x,y_2])) = \exists x.(p_1 \Rightarrow p_2)\{p_i \leftarrow \forall y_i.F_i[x,y_i]\}_{i=1,2}$ has root a labeled $(x, p_1 \Rightarrow p_2)$ with left child b_1 labeled $(y_1,\neg F_1)$, right child b_2 labeled $(y_2,\neg F_2)$, and arcs from a to b_1 and from a to b_2 labeled p_1 and p_2, respectively. Note how $FV(a.F) \subseteq \{x,p_1,p_2\}$, $Var(a) = \{x,p_1,p_2\}$, and $Rigid(a) = \emptyset$. 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 $\forall x.((\exists y_1.(x \simeq 2{\cdot}y_1)) \Rightarrow (\exists y_2.(3{\cdot}x \simeq 2{\cdot}y_2)))$. Since there is an outermost $\forall$, a double negation gets $\neg(\exists x.((\exists y_1.(x \simeq 2{\cdot}y_1)) \wedge (\forall y_2.(3{\cdot}x \not\simeq 2{\cdot}y_2))))$. Since there is an innermost $\exists$, a double negation gets $\neg(\exists x.(\neg(\forall y_1.(x \not\simeq 2{\cdot}y_1)) \wedge (\forall y_2.(3{\cdot}x \not\simeq 2{\cdot}y_2))))$. Let $\varphi = \exists x.(\neg(\forall y_1.(x \not\simeq 2{\cdot}y_1)) \wedge (\forall y_2.(3{\cdot}x \not\simeq 2{\cdot}y_2))) = \exists x.(\neg p_1 \wedge p_2)\{p_1 \leftarrow \forall y_1.(x \not\simeq 2{\cdot}y_1),\ p_2 \leftarrow \forall y_2.(3{\cdot}x \not\simeq 2{\cdot}y_2)\}$. Then the original formula is true in LRA iff φ is false in LRA. The game tree for φ has root a labeled $(x,\neg p_1 \wedge p_2)$ 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 a to b_1 and from a to b_2 labeled p_1 and p_2, respectively. The variable sets of this tree are as in Ex. 1.

Conversely, given a game $\mathcal{G} = (\bar{z},T)$, we can associate a formula $a.\psi$ to any node a in T and hence to game $\mathcal{G}_a$.

Definition 4 (Formula at a node). *Given a satisfiability game $\mathcal{G} = (\bar{z},T)$, for all nodes a of T, the* formula $a.\psi$ at node a *is defined inductively as follows:*

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

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

Example 3. For the game as in Ex. 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 $a.\psi = \exists x.(\neg p_1 \wedge p_2)\{p_1 \leftarrow \forall y_1.(x \not\simeq 2{\cdot}y_1),\ p_2 \leftarrow \forall y_2.(3{\cdot}x \not\simeq 2{\cdot}y_2)\} = \exists x.(\neg(\forall y_1.(x \not\simeq 2{\cdot}y_1)) \wedge (\forall y_2.(3{\cdot}x \not\simeq 2{\cdot}y_2))) = \varphi$.

Theorem 1. *For all formulas φ with $FV(\varphi) = \bar{z}$, for all models $\mathcal{M}$ extending $\mathcal{M}_0$ to $\bar{z}$, the game $\mathcal{G}$ for φ is winning for $\mathcal{M}$ iff $\mathcal{M} \models \varphi$.*
Proof. See Appendix A.1.

Given node a and extension $\mathcal{M}$ of $\mathcal{M}_0$ to $\mathit{Rigid}(a)$, one could check whether $\mathcal{G}$ is winning for $\mathcal{M}$ by testing all possible extensions $\mathcal{M}'$. Since 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 the notions of *under-approximation* and *over-approximation* of φ in order to under-approximate and over-approximate $[\![\varphi]\!]$.

Definition 5 (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 game for φ, and U and O under- and over-approximations of φ, respectively. Then, $\mathcal{M} \models U$ implies that $\mathcal{G}$ is winning for $\mathcal{M}$, so that satisfying an under-approximation is a sufficient condition to win. On the other hand, $\mathcal{M} \models \neg O$ implies that $\mathcal{G}$ is losing for $\mathcal{M}$. By the contrapositive, if $\mathcal{G}$ is winning for $\mathcal{M}$ then $\mathcal{M} \not\models \neg O$, that is, $\mathcal{M} \models O$, so that satisfying an over-approximation is a necessary condition to win. 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 generalization*: A function MG 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 $\mathcal{M} \models \exists\bar{x}.F[\bar{z},\bar{x}]$, $\mathsf{MG}(F[\bar{z},\bar{x}],\bar{x},\mathcal{M})$ 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 interpolation*: A function MI 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 $\mathcal{M} \not\models \exists\bar{x}.F[\bar{z},\bar{x}]$, $\mathsf{MI}(F[\bar{z},\bar{x}],\bar{x},\mathcal{M})$ returns a quantifier-free formula $O[\bar{z}]$ such that $\mathcal{M} \not\models O[\bar{z}]$ and $\mathcal{T} \models (\exists\bar{x}.F[\bar{z},\bar{x}]) \Rightarrow O[\bar{z}]$.

MG and MI 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.* Following [11] we call it *model generalization*, 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.* Following [16] we call it *model interpolation.*

4 The QSMA Algorithm and Its Total Correctness

Let $\mathcal{G} = (\bar{z}, T)$ be the game 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 $\mathcal{G}$ is winning for $\mathcal{M}$. 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 strenghtening O, so that the $[\![O]\!]$ bubble deflates. The algorithm operates in this manner for all subformulas of φ: for all nodes a of T it maintains under and over-approximations $a.U$ and $a.O$ of $a.\psi$, progressively weakening $a.U$ and strenghtening $a.O$. The weakening of $a.U$ is done by introducing a disjunction with a model generalization, and the strenghtening of $a.O$ is done by introducing a conjunction with a model interpolation, in such a way that $\mathcal{M}$ satisfies $a.U \vee \neg a.O$. As soon as $\mathcal{M}$ satisfies $a.U$, game $\mathcal{G}_a$ is winning for $\mathcal{M}$. As soon as $\mathcal{M}$ satisfies $\neg a.O$, game $\mathcal{G}_a$ is losing for $\mathcal{M}$.

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

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

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

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

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

1. Build a formula L as the conjunction of $a.F$ and a formula for every child b of a, denoted $a \to b$ (line 7 in Fig. 2).
2. Invoke the SMA function to search for an extension $\mathcal{M}'$ of $\mathcal{M}$ to $Var(a)$ such that $\mathcal{M}' \models L$ (line 8). For all children b of a, $b.p \in Var(a)$ 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 $\neg b.U$ and B plays first in game $\mathcal{G}_b$. If $\mathcal{M}'(b.p) = \mathsf{false}$, the subformula for b in L reduces to $b.O$ and A plays first in game $\mathcal{G}_b$. The proof of partial correctness of `subgameIsWinning` shows that the existence of an $\mathcal{M}'$ such that $\mathcal{M}' \models L$ is necessary for game $\mathcal{G}_a$ to be winning for $\mathcal{M}$.

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

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

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

3. If no such $\mathcal{M}'$ exists, then game $\mathcal{G}_a$ is losing for $\mathcal{M}$; `subgameIsWinning` updates $a.O$ to its conjunction with $\mathsf{MI}(L, FV(L) \setminus Rigid(a), \mathcal{M})$ (line 10). Since $\mathcal{M} \not\models L$, by the specification of MI we know that $\mathcal{M} \not\models \mathsf{MI}(L, FV(L) \setminus Rigid(a), \mathcal{M})$. This update ensures that $\mathcal{M} \not\models a.O$, so that $\mathcal{M} \models \neg a.O$. Then `subgameIsWinning` returns *false* (line 11).
4. Otherwise, we have an extension $\mathcal{M}'$ that satisfies L and hence $a.F$, so that there is the potential for game $\mathcal{G}_a$ to be winning for $\mathcal{M}$, and we invoke `winningForallChildren` to check that this is indeed the case.
5. If this test succeeds, `subgameIsWinning` builds a formula L' as the conjunction of $a.F$ and a formula for every child b of a (line 14). If $\mathcal{M}'(b.p) = \mathsf{true}$, the subformula for b in L' reduces to $\neg b.O$ and B plays first in $\mathcal{G}_b$. If $\mathcal{M}'(b.p) = \mathsf{false}$, the subformula for b in L' reduces to $b.U$ and A plays first in $\mathcal{G}_b$. The proof of partial correctness of `subgameIsWinning` shows that $\mathcal{M}' \models L'$ and that $\mathcal{M}' \models L'$ is a sufficient condition for $\mathcal{G}_a$ to be winning for $\mathcal{M}$. Then `subgameIsWinning` updates $a.U$ to its disjunction with $\mathsf{MG}(L', FV(L') \setminus Rigid(a), \mathcal{M})$ (line 15). Since $\mathcal{M}' \models L'$, by the specifica-

tion of MG we know that $\mathcal{M}' \models \mathsf{MG}(L', FV(L') \setminus Rigid(a), \mathcal{M})$. This update ensures that $\mathcal{M}' \models a.U$. Then `subgameIsWinning` returns *true* (line 16).

6. If `winningForallChildren` returned *false*, the control returns to line 7.

The function `winningForallChildren` calls `subgameIsWinning` for every child b of a. As soon as it finds a child b such that $\mathcal{M}(b.p) = \mathsf{false}$ (A plays first) and `subgameIsWinning` returns *false* (i.e., A loses), or $\mathcal{M}(b.p) = \mathsf{true}$ (B plays first) and `subgameIsWinning` returns *true* (i.e., B wins), `winningForallChildren` returns *false*, because it found a subgame where candidate model $\mathcal{M}$ fails. If this does not happen, `winningForallChildren` returns *true*.

Example 4. Apply `subgameIsWinning` to the root of the game tree in Ex. 1. Formula L gets $p_1 \Rightarrow p_2$. SMA produces an $\mathcal{M}'$ that assigns values to x, p_1, and p_2. If $\mathcal{M}'$ satisfies $p_1 \Rightarrow p_2$ by assigning false to p_1, A plays first in game $\mathcal{G}_{b_1}$. In the recursive call on b_1, formula L gets $\neg F_1$. If SMA produces an $\mathcal{M}''$ that extends $\mathcal{M}'$ with an assignment to y_1 such that $\mathcal{M}'' \models \neg F_1$, player A wins. If $\mathcal{M}'$ satisfies $p_1 \Rightarrow p_2$ by assigning true to p_2, B plays first in game $\mathcal{G}_{b_2}$. In the recursive call on b_2, formula L gets $\neg F_2$. If SMA fails to produces an $\mathcal{M}''$ that extends $\mathcal{M}'$ with an assignment to y_2 such that $\mathcal{M}'' \models \neg F_2$, player A wins.

Theorem 2. *The function* `subgameIsWinning` *is partially correct: if the preconditions hold and the function halts, then the postconditions hold.*
Proof. See Appendix A.2.

For termination, we begin with the MG and MI functions. Let $\mathcal{T}$ be LRA with a theory extension LRA^+ that adds constant symbols $\tilde{q}$ for all rational numbers q. Consider an MG function such that $\mathsf{MG}(F[\bar{z}, x], x, \mathcal{M}) = F[x]\{x \leftarrow \tilde{q}\}$ and $\mathcal{M} \models F[\bar{z}, \tilde{q}]$. This kind of model generalization is called *generalization-by-substitution* [11]. While $F[\bar{z}, \tilde{q}]$ is an under-approximation of $\exists x.F[\bar{z}, x]$, this MG is not a good choice for termination. By applying MG repeatedly with an infinite enumeration of rational constants, the QSMA algorithm could build an infinite sequence of under-approximations $(\bigvee_{i=1}^{n} F[x]\{x \leftarrow \tilde{q}_i\})_{n \in \mathbb{N}}$ none of which is LRA-equivalent to $\exists x.F[\bar{z}, x]$. The next definition excludes such MG functions.

Definition 6 (Convergence). *A model generalization function* MG *is* convergent *if for all series of calls* $\{\mathsf{MG}(F[\bar{z}, \bar{x}], \bar{x}, \mathcal{M}_i)\}_{i \geq 1}$*, producing a series of formulas* $\{U_i[\bar{z}]\}_{i \geq 1}$*, there exist finitely many indices* $i_1, \ldots, i_n$ *such that for all* i*,* $i \geq 1$*, there exists an* i_j*,* $1 \leq j \leq n$*, for which* $\mathcal{T} \models U_i[\bar{z}] \Leftrightarrow U_{i_j}[\bar{z}]$*.*

Def. 6 applies to MI with $O[\bar{z}]$ in place of $U[\bar{z}]$.[4]

Lemma 1. *If* MG *and* MI *are convergent, for all (possibly infinite) series of calls* $\{$`subgameIsWinning`$(a, \mathcal{M}_i)\}_i$*, all satisfying the preconditions and all terminating,* $a.U$ *and* $a.O$ *are updated only a finite number of times.*
Proof. See Appendix A.3.

Once nontermination due to MG or MI has been excluded even for an infinite series of halting calls, termination is proved by induction on the game tree.

[4] The existence of convergent MG and MI functions implies quantifier elimination.

Theorem 3. *If the* MG *and* MI *functions are convergent, whenever the preconditions are satisfied the function* `subgameIsWinning` *halts.*
Proof. See Appendix A.4.

Example 5. Apply `subgameIsWinning` to the root of the game tree in Ex. 2. Formula L gets $\neg p_1 \wedge p_2$. 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 false to p_1 and true to p_2. A plays first in game $\mathcal{G}_{b_1}$. 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 $y_1{\leftarrow}\frac{1}{2}$, player A wins $\mathcal{G}_{b_1}$. B plays first in game $\mathcal{G}_{b_2}$. 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 $y_2{\leftarrow}\frac{3}{2}$, player B wins $\mathcal{G}_{b_2}$, so that A loses $\mathcal{G}$. Indeed, formula φ of Ex. 2 is false as the original formula is true.

5 The YicesQS Variant of the QSMA Algorithm

YicesQS implements an optimized variant of QSMA, called $\mathsf{optiQSMA}$, that reduces the number of recursive calls to `subgameIsWinning` by entrusting more work to each call to SMA. Reconsider the behavior of QSMA in Ex. 4. We can avoid a recursive call to `subgameIsWinning` by asking SMA to satisfy $(p_1 \Rightarrow p_2) \wedge (\neg p_1 \Rightarrow \neg F_1)$ in lieu of $p_1 \Rightarrow p_2$. This way, if the candidate model returned by SMA assigns false to p_1, it also assigns to x and y_1 values that satisfy $\neg F_1$. This means that $\forall y_1.F_1$ is found false without the recursive call where A plays first. On the other hand, if p_2 is assigned true, we still have to make the recursive call to see if B can satisfy $\neg F_2$. The idea of $\mathsf{optiQSMA}$ is to do a look-ahead on what would be a series of recursive calls where A plays first, doing the work in one shot rather then through all such calls. The following definition builds a formula to allow the look-ahead.

Definition 7 (Look-ahead formula). *Given a game* $\mathcal{G} = (\bar{z}, T)$*, for all nodes* a *of* T *the* look-ahead formula *of* a *is* $LF(a) = a.F \wedge \bigwedge_{a \to b}(\neg b.p \Rightarrow LF(b))$.

The next definition distinguishes the nodes that are handled together in one shot without recursion and those where recursion is still needed.

Definition 8 (No alternation nodes and first alternation nodes). *Given a game* $\mathcal{G} = (\bar{z}, T)$ *for all nodes* a *of* T *and extensions* $\mathcal{M}$ *of* $\mathcal{M}_0$ *to* $FV(LF(a))$*, the set* $\mathsf{NAN}(a, \mathcal{M})$ *of the* no-alternation nodes from a according to $\mathcal{M}$ *(resp. the set* $\mathsf{FAN}(a, \mathcal{M})$ *of the* first-alternation nodes from a according to $\mathcal{M}$*) contains all and only the nodes* b *such that: (i)* b *is a descendant of* a *through a path* $a \to a_1 \to \ldots \to a_n \to b$ $(n \geq 0)$*, (ii)* $\forall i,\ 1 \leq i \leq n,\ \mathcal{M}(a_i.p) = \mathsf{false}$*, and (iii)* $\mathcal{M}(b.p) = \mathsf{false}$ *(resp.* $\mathcal{M}(b.p) = \mathsf{true}$*).*

The $\mathsf{optiQSMA}$ algorithm seeks a candidate model $\mathcal{M}$ that satisfies $LF(a)$ and recurses only on the nodes in $\mathsf{FAN}(a, \mathcal{M})$. A $b \in \mathsf{FAN}(a, \mathcal{M})$ for which $n = 0$ in Condition (i) of Def. 8 is a child of a: for such a child there is no optimization.

```
@pre: G = (z̄, T): game for φ with FV(φ) = z̄; M: extension of M_0 to z̄
@post: rv iff M ⊨ φ
function optiQSMA(M, T)
    for all nodes a in T do
        a.U ←⊥
    ans ← optiSubgameIsWinning(root(T), 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

Definition 9 (Winning with look-ahead). *For all games $\mathcal{G} = (\bar{z}, T)$ with $r = root(T)$ and extensions $\mathcal{M}$ of $\mathcal{M}_0$ to $Rigid(r) = \bar{z}$, the first player in $\mathcal{G}$* wins with look-ahead from $\mathcal{M}$, *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 $b \in \mathsf{FAN}(r, \mathcal{M}')$, $\mathcal{G}_b$ is losing for $\mathcal{M}'$.*

Game $\mathcal{G}$ is *winning with look-ahead* for model $\mathcal{M}$ iff the first player in $\mathcal{G}$ wins with look-ahead from $\mathcal{M}$. The optimization does not change the game:

Theorem 4. *Given game $\mathcal{G} = (\bar{z}, T)$ and extension $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$, $\mathcal{G}$ is winning for $\mathcal{M}$ if and only if $\mathcal{G}$ is winning with look-ahead for $\mathcal{M}$.*

Proof. See Appendix A.5.

The optiQSMA algorithm maintains under-approximations $a.U$ of $a.\psi$ for all nodes a, but not over-approximations. Accordingly, the main function optiQSMA (Fig. 3) initializes only $a.U$ for all nodes a, and then calls `optiSubgameIsWinning` (Fig. 4). This function returns $\mathsf{SAT}(U)$ if $\mathcal{G}$ is winning with look-ahead for $\mathcal{M}$ and $\mathsf{UNSAT}(O)$ otherwise, where U is an under-approximation of $r.\psi$ ($r = root(T)$) such that $\mathcal{M} \models U$, and O is an over-approximation of $r.\psi$ such that $\mathcal{M} \not\models O$. The main function optiQSMA has no usage for U and O, and merely returns *true* or *false* accordingly, but in `optisubgameIsWinning` under-approximations and over-approximations are returned through the recursion. The reason for saving only under-approximations is practical, and will become clear after the illustration of `optisubgameIsWinning`.

According to the optimization, `optiSubgameIsWinning` builds formula L (line 3 in Fig. 4) as the conjunction of the look-ahead formula $LF(a)$ (in lieu of $a.F$ in line 7 of Fig. 2) and a formula for every descendant b of a, denoted $a \to^{+} b$ (in lieu of child as in Fig. 2). Then, `optiSubgameIsWinning` invokes SMA to search for an extension $\mathcal{M}'$ of $\mathcal{M}$ to $Var(a)$ such that $\mathcal{M}' \models L$. If there is no such extension, `optiSubgameIsWinning` returns $\mathsf{UNSAT}(O)$, where O is simply the outcome of the application of MI to L and $\mathcal{M}$, because over-approximations are not kept. Suppose that SMA returns an extension $\mathcal{M}'$ that satisfies L. For those descendants b for which $\mathcal{M}'(b.p) = \mathsf{true}$, the subformula for b in L reduces to $\neg b.U$ as in Step 2 of the description of `SubgameIsWinning`. For those descen-

@pre: $\mathcal{M}$ is an extension of $\mathcal{M}_0$ to $\mathit{Rigid}(a)$, 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 $\mathcal{M} \models U]\}$

function OPTISUBGAMEISWINNING(a, $\mathcal{M}$)
    while true **do**
        $L \leftarrow LF(a) \wedge \bigwedge_{a \to^+ b}(b.p \Rightarrow \neg b.U)$
        $\mathcal{M}' \leftarrow \mathsf{SMA}(L, \mathcal{M})$
        if $\mathcal{M}' = nil$ **then**
            return $\mathsf{UNSAT}(\mathsf{MI}(L,\ FV(L) \setminus \mathit{Rigid}(a),\ \mathcal{M}))$
        else
            reasons $\leftarrow \top$
            if WINNINGFORALLDESCENDANTS(a, $\mathcal{M}'$, reasons) **then**
                $L' \leftarrow LF(a) \wedge$ reasons
                return $\mathsf{SAT}(\mathsf{MG}(L',\ FV(L') \setminus \mathit{Rigid}(a),\ \mathcal{M}))$

function WINNINGFORALLDESCENDANTS(a, $\mathcal{M}$, reasons)
    for all $b \in \mathsf{FAN}(a, \mathcal{M})$ **do**
        ans $\leftarrow$ OPTISUBGAMEISWINNING(b, $\mathcal{M}$)
        if ans $= \mathsf{SAT}(U)$ **then**
            $b.U \leftarrow b.U \vee U$
            return *false*
        else if ans $= \mathsf{UNSAT}(O)$ **then**
            reasons $\leftarrow$ reasons $\wedge\ (b.p \wedge \neg O)$
    for all $b \in \mathsf{NAN}(a, \mathcal{M})$ **do**
        reasons $\leftarrow$ reasons $\wedge\ \neg b.p$
    return *true*

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

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

Having found an $\mathcal{M}'$, there is the potential to be winning with look-ahead, and `optiSubgameIsWinning` invokes `winningForallDescendants` to check this. Prior to the call, `optiSubgameIsWinning` initializes the formula `reasons` to $\top$ and passes it by reference to `winningForallDescendants`. If the latter returns *true*, `optiSubgameIsWinning` builds formula L' as the conjunction of $LF(a)$ and `reasons`, and returns $\mathsf{SAT}(U)$, where U is the outcome of the application of MG to L' and $\mathcal{M}$. Function `winningForallDescendants` considers first all descendants b in $\mathsf{FAN}(a, \mathcal{M})$, and calls `optiSubgameIsWinning` for each of them. If `optiSubgameIsWinning` returns $\mathsf{SAT}(U)$, it means that player B wins in $\mathcal{G}_b$, and hence `winningForallDescendants` returns *false*. Prior to that, it weakens $b.U$ by disjunction with U. If `optiSubgameIsWinning` returns $\mathsf{UNSAT}(O)$, it means that player B wins in $\mathcal{G}_b$, and we move on to the next descendant in $\mathsf{FAN}(a, \mathcal{M})$. Prior to that, `reasons` is strenghtened by conjunction with $b.p \wedge \neg O$. For all descendants b in $\mathsf{NAN}(a, \mathcal{M})$, `winningForallDescendants` only strenghtens `reasons` by conjunction with $\neg b.p$. The proof of partial correctness of `optiSubgameIsWinning` shows that `reasons` is an explanation of why the game is winning with look-ahead.

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 `optiSubgameIsWinning` is used to build the temporary formula `reasons`, but it is not saved, and `reasons` is used to compute L'.

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

Proof. See Appendix A.6.

Since optiQSMA does not keep over-approximations, for termination we assume that the MG and MI functions satisfy a stronger property than convergence.

Definition 10 (Finite basis). *A model generalization function* MG *has* finite basis *if the set* $\{\mathsf{MG}(F[\bar{z}, \bar{x}], \bar{x}, \mathcal{M}) \mid \mathcal{M} : \text{extension of } \mathcal{M}_0 \text{ to } \bar{z} \text{ such that } \mathcal{M} \models \exists\bar{x}.F[\bar{z}, \bar{x}]\}$ *is finite for all quantifier-free formulas* $F[\bar{z}, \bar{x}]$ *and tuples* $\bar{x}$*.*

Def. 10 applies to MI with $\not\models$ in place of $\models$.

Theorem 6. *If the* MG *and* MI *functions have finite basis, whenever the preconditions are satisfied the function* `optiSubgameIsWinning` *halts.*

Proof. See Appendix A.7.

6 YicesQS, Experimental Results, and Discussion

YicesQS extends the Yices 2 solver[5] with support for quantifiers for complete theories (unrelated to Yices 2 support for quantifiers in UF). *Model interpolation* is available in Yices's MCSAT [9] solver for quantifier-free formulas, including theory-specific techniques for arithmetic based on NLSAT [15] (and ultimately, Cylindrical Algebraic Decomposition–CAD), and bitvectors (BV) [14]. Basic *model generalization* is done generically by substitution [11] and improved with theory-specific techniques: model-based projection (also based on CAD) for arithmetic, and *invertibility conditions* [20] for BV, including ϵ-terms (cegqi [20] is likely to be the closest solver to YicesQS on BV).

YicesQS is a recent implementation that only participated to the SMT competition in 2021 and 2022. In 2022, 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 have been computed by running the competition script `join.sh` on the raw data from StarExec,[6] then sorting the

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

[6] https://github.com/SMT-COMP/smt-comp/tree/master/2022/results

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

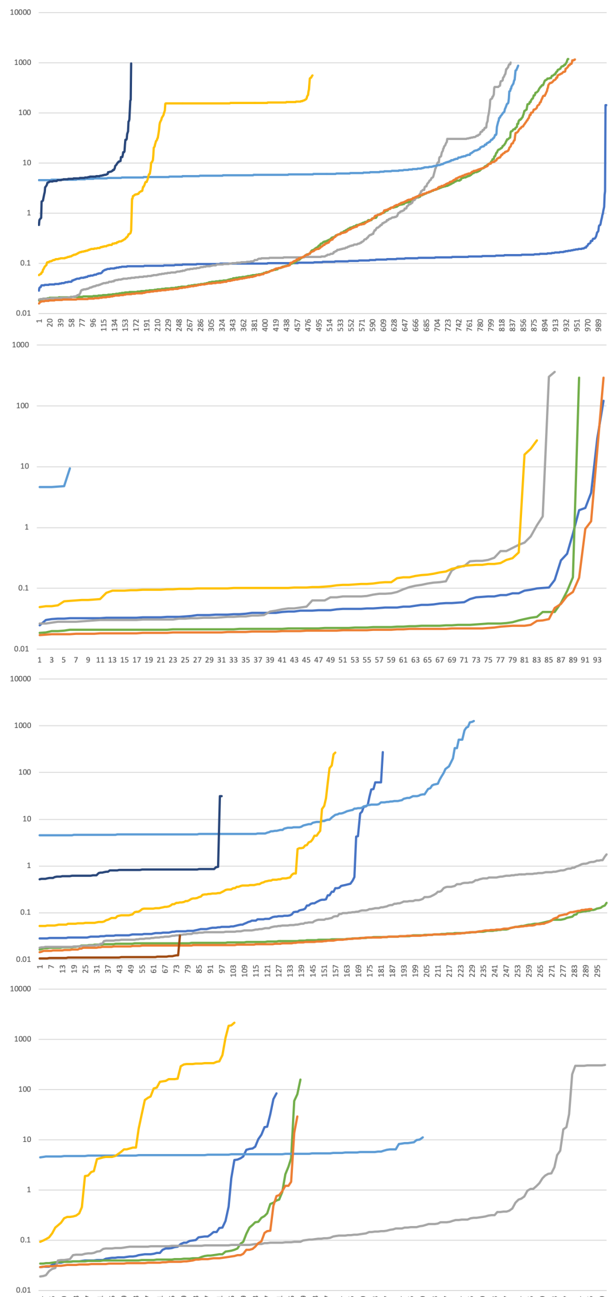

Fig. 5. Plots for the four arithmetics.

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

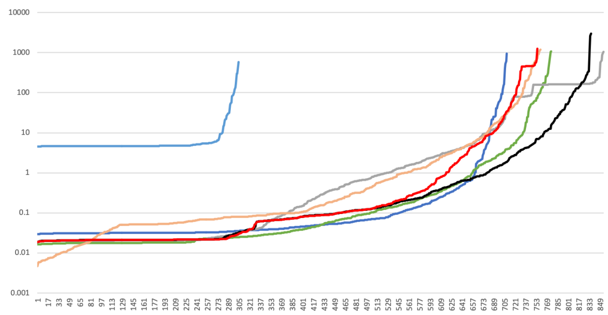

Fig. 6. Plot for BV.

data and producing the plots in spreadsheets that we make available online.[7] A description of the participating solvers can be found on the competition website.[8]

Fig. 5 shows the results for the four arithmetics. The left-hand side shows, for each logic and each solver, the number of instances solved and the time it took to solve these. In the right-hand side plot, 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. The second best solver, Z3 2021, solved 948 benchmarks with a total runtime about 100 times higher. Interestingly, YicesQS does not have a special treatment (e.g., simplex-based) of linear problems, but relies on CAD-based techniques for model generalization and interpolation. YicesQS does not use sophisticated techniques to reason about integers rather than reals. Consequently, it is somewhat average on integers. Note that, technically, both NIA and (because of division by 0) NRA are undecidable, and hence stand outside of the theoretical framework of QSMA. YicesQS answers should still be correct, but termination can be lost. With Z3 being a non-competing participant to 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 first for the 24s timeout setup (instead of 20 minutes).

Fig. 6 shows the results for BV, where YicesQS did not perform as well, despite a high number of quickly solved benchmarks compared to, e.g., CVC5. A possible explanation is that model interpolation makes no or poor use of invertibility conditions. Improving model interpolation via invertibility conditions should provide much better results.

In addition to such improvements to the solver, plans for future work include investigating how to compose QSMA with the CDSAT framework for conflict-driven reasoning in unions of theories [4, 5].

[7] http://www.csl.sri.com/users/sgl/Work/Cade2023-data/index.html
[8] https://smt-comp.github.io/2022/participants.html

Acknowledgements Part of this work was done while the first and third authors were visiting CSL at SRI International, whose support is greatly appreciated. This material is based upon work supported by NSF under Awards No. 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 United States 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)
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)
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)
5. 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)
6. 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)
7. Bradley, A.R., Manna, Z.: The Calculus of Computation - Decision Procedures with Applications to Verification. Springer (2007)
8. 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)
9. 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)
10. Detlefs, D.L., Nelson, G., Saxe, J.B.: Simplify: a theorem prover for program checking. J. ACM **52**(3), 365–473 (2005)
11. Dutertre, B.: Solving exists/forall problems with Yices. In: Proc. SMT-13 (2015)
12. 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)
13. 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)
14. 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)
15. 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)
16. Jovanović, D., Dutertre, B.: Interpolation and model checking for nonlinear arithmetic. In: Silva, A., Leino, R. (eds.) Proc. CAV-35. LNCS, vol. 12760, pp. 266–288. Springer (2021)

17. Komuravelli, A., Gurfinkel, A., Chaki, S.: SMT-based model checking for recursive programs. In: Biere, A., Bloem, R. (eds.) Proc. CAV-26. LNCS, vol. 8559, pp. 17–34. Springer (2014)
18. 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)
19. Moskal, M.: Fx7 or in software, it is all about quantifiers. System Descriptions at SMT-COMP (2007), available as http://smtcomp.cs.uiowa.edu/2007/descriptions/fx7.pdf
20. Niemetz, A., Preiner, M., Reynolds, A., Barrett, C.W., 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)
21. 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)

A Proofs

A.1 Proof of Lemma 1

Game $\mathcal{G}$ and formula φ are as in Def. 3. The proof is by induction on the number n of the subformulas φ_i of φ with outermost universal quantifier.
Base case: $n = 0$ and T consists of the single node a with label $(\bar{x}, F[\bar{z}, \bar{x}])$. By Def. 2, $\mathcal{G}$ is winning for $\mathcal{M}$ iff there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\mathit{Var}(a) = \bar{x}$ such that $\mathcal{M}' \models F[\bar{z}, \bar{x}]$, that is, iff $\mathcal{M} \models \varphi$.
Induction hypothesis: $n \geq 0$, and for all i, $1 \leq i \leq n$, for all models $\mathcal{M}$ extending $\mathcal{M}_0$ to $\mathit{Rigid}(b_i) = \bar{z} \uplus \bar{x}$, game $\mathcal{G}_i$ is winning for $\mathcal{M}$ iff $\mathcal{M} \models \neg\varphi_i$.
Induction step: we distinguish the two directions.
$\Rightarrow$) Let $\mathcal{M}$ be an extension of $\mathcal{M}_0$ to $\mathit{Rigid}(a) = \bar{z}$, such that $\mathcal{G}$ is winning for $\mathcal{M}$. By Def. 2, there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\mathit{Var}(a)$ such that $\mathcal{M}' \models F[\bar{z}, \bar{x}, \bar{p}]$ and for all i, $1 \leq i \leq n$, $\mathcal{M}'(p_i) = \mathsf{false}$ iff $\mathcal{G}_i$ is winning for $\mathcal{M}'$. By induction hypothesis, $\mathcal{M}'(p_i) = \mathsf{false}$ iff $\mathcal{M}' \models \neg\varphi_i$, or, equivalently, $\mathcal{M}'(p_i) = \mathsf{true}$ iff $\mathcal{M}' \models \varphi_i$. Therefore, $\mathcal{M}' \models F[\bar{z}, \bar{x}, \bar{p}] \wedge \bigwedge_{i=1}^{n} p_i \Leftrightarrow \varphi_i$, and hence $\mathcal{M} \models \varphi$.
$\Leftarrow$) Let $\mathcal{M}$ be an extension of $\mathcal{M}_0$ to $\mathit{Rigid}(a) = \bar{z}$, 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 $\mathcal{M}' \models \varphi_i$, or, equivalently, $\mathcal{M}'(p_i) = \mathsf{false}$ iff $\mathcal{M}' \models \neg\varphi_i$. By induction hypothesis, $\mathcal{M}'(p_i) = \mathsf{false}$ iff game $\mathcal{G}_i$ is winning for $\mathcal{M}'$. By Def. 2, $\mathcal{G}$ is winning for $\mathcal{M}$.

A.2 Proof of Theorem 2

Consider a call `subgameIsWinning`$(a, \mathcal{M})$. 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 game tree T_a of $\mathcal{G}_a$.
Base case: a is a leaf. If $\mathcal{M} \models a.U$ and the function returns *true* on line 3 in Fig. 2, we have $\mathcal{M} \models (a.U \vee \neg a.O)$, $rv = \mathit{true}$, and $\mathcal{G}_a$ is winning for $\mathcal{M}$, since $\mathcal{M} \models a.U$ implies $\mathcal{M} \models a.\psi$. If $\mathcal{M} \models \neg a.O$ and the function returns *false* on

line 5, we have $\mathcal{M} \models (a.U \vee \neg a.O)$, $rv = \mathit{false}$, and $\mathcal{G}_a$ is losing for $\mathcal{M}$, since $\mathcal{M} \models \neg a.O$ implies $\mathcal{M} \not\models a.\psi$. Otherwise, L is assigned $a.F$ since a has no children, and SMA is invoked to find an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(a.F)$ such that $\mathcal{M}' \models a.F$.

If no such extension is found, $\mathsf{MI}(a.F, FV(a.F) \setminus \mathit{Rigid}(a), \mathcal{M})$ returns a quantifier-free formula that is false in $\mathcal{M}$, $a.O$ is conjoined with this formula, and the function returns *false* on line 11. Thus, $\mathcal{M} \not\models a.O$, $\mathcal{M} \models \neg a.O$, $\mathcal{M} \models (a.U \vee \neg a.O)$, $rv = \mathit{false}$, and $\mathcal{G}_a$ is losing for $\mathcal{M}$, since $a.F$, and hence $a.\psi$, cannot be satisfied. If SMA returns an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(a.F)$ such that $\mathcal{M}' \models a.F$, $\texttt{winningForallChildren}(a, \mathcal{M}')$ returns *true* because a has no children, L' is assigned $a.F$ for the same reason, $\mathsf{MG}(a.F, FV(a.F) \setminus \mathit{Rigid}(a), \mathcal{M})$ returns a quantifier-free formula that is true in $\mathcal{M}$, $a.U$ is disjoined with this formula, and the function returns *true* on line 16. Thus, $\mathcal{M} \models a.U$, $\mathcal{M} \models (a.U \vee \neg a.O)$, $rv = \mathit{true}$, and $\mathcal{G}_a$ is winning for $\mathcal{M}$, since $\mathcal{M} \models a.U$ implies $\mathcal{M} \models a.\psi$.

Induction hypothesis: for all children b of node a, if the preconditions are satisfied and $\texttt{subgameIsWinning}(b, \mathcal{M})$ halts, the postconditions are satisfied.

Induction step: if $\texttt{subgameIsWinning}(a, \mathcal{M})$ returns on line 3 or on line 5, the reasoning is the same as in the base case. Otherwise, L is assigned the formula $a.F \wedge \bigwedge_{a\to b}((b.p \wedge \neg b.U) \vee (\neg b.p \wedge b.O))$. This formula is constructed in such a way that if game $\mathcal{G}_a$ is winning for $\mathcal{M}$ then L is satisfied. Indeed, suppose that $\mathcal{G}_a$ is winning for $\mathcal{M}$. This means that there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ such that: (i) $\mathcal{M}' \models a.F$; (ii) for all children b of a with $\mathcal{M}'(b.p) = \mathsf{false}$ (A plays first in $\mathcal{G}_b$), $\mathcal{M}' \models b.\psi$ (since A wins), so that by induction hypothesis ($[\![b.\psi]\!] \subseteq [\![b.O]\!]$) $\mathcal{M}' \models b.O$; and (iii) for all children b of a with $\mathcal{M}'(b.p) = \mathsf{true}$ (B plays first in $\mathcal{G}_b$), $\mathcal{M}' \not\models b.\psi$ (since B loses), and hence $\mathcal{M}' \models \neg b.\psi$, so that 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, $\mathsf{MI}(L, FV(L) \setminus \mathit{Rigid}(a), \mathcal{M})$ returns a quantifier-free formula that is false in $\mathcal{M}$, $a.O$ is conjoined with this formula, and the function returns *false* on line 11. Therefore, $\mathcal{M} \not\models a.O$, $\mathcal{M} \models \neg a.O$, $\mathcal{M} \models (a.U \vee \neg a.O)$, $rv = \mathit{false}$, and $\mathcal{G}_a$ is losing for $\mathcal{M}$, so that the postconditions of $\texttt{subgameIsWinning}(a, \mathcal{M})$ are satisfied. If there exists an extension $\mathcal{M}'$ such that $\mathcal{M}' \models L$, $\texttt{winningForallChildren}(a, \mathcal{M}')$ is invoked. If it returns *true*, L' is assigned the formula $a.F \wedge \bigwedge_{a\to b}((b.p \wedge \neg b.O) \vee (\neg b.p \wedge b.U))$. This formula is constructed in such a way that it has two properties.

The first one is that $\mathcal{M}' \models L'$. Indeed, $\mathcal{M}' \models a.F$, because $\mathcal{M}' \models L$, and from the knowledge that game $\mathcal{G}_a$ is winning for $\mathcal{M}$ ($\texttt{winningForallChildren}$ returned *true*) we know that for all children b of a, if $\mathcal{M}'(b.p) = \mathsf{true}$ (B plays first in $\mathcal{G}_b$), $\mathcal{G}_b$ is losing for $\mathcal{M}'$ (i.e., $\texttt{subgameIsWinning}(b, \mathcal{M}')$ returned *false*), so that $\mathcal{M}' \models \neg b.O$ by induction hypothesis, and if $\mathcal{M}'(b.p) = \mathsf{false}$ (A plays first in $\mathcal{G}_b$), $\mathcal{G}_b$ is winning for $\mathcal{M}'$ (i.e., $\texttt{subgameIsWinning}(b, \mathcal{M}')$ returned *true*), so that $\mathcal{M}' \models b.U$ by induction hypothesis.

The second property is that $\mathcal{M}' \models L'$ is a sufficient condition for $\mathcal{G}_a$ to be winning for $\mathcal{M}$. Indeed, $\mathcal{M}' \models L'$ implies (i) $\mathcal{M}' \models a.F$, and (ii) for all children b

of a, by induction hypothesis, if $\mathcal{M}'(b.p) = \mathsf{false}$ (A plays first in $\mathcal{G}_b$), $\mathcal{M}' \models b.U$ implies that $\mathcal{G}_b$ is winning for $\mathcal{M}'$ (i.e., A wins), if $\mathcal{M}'(b.p) = \mathsf{true}$ (B plays first in $\mathcal{G}_b$), $\mathcal{M}' \models \neg b.O$ implies that $\mathcal{G}_b$ is losing for $\mathcal{M}'$ (i.e., B loses). Then, $\mathsf{MG}(L', FV(L') \setminus \mathit{Rigid}(a), \mathcal{M})$ returns a quantifier-free formula that is true in $\mathcal{M}$, $a.U$ is disjoined with this formula, and the function returns *true* on line 16. Thus, $\mathcal{M} \models a.U$, $\mathcal{M} \models (a.U \vee \neg a.O)$, $rv = \mathit{true}$, and $\mathcal{G}_a$ is winning for $(A, \mathcal{M})$, so that the postconditions are satisfied.

A.3 Proof of Lemma 1

The proof is by structural induction on the game tree T_a. The base case (a is a leaf) is trivial. The induction hypothesis is that the claim holds for all children b of a. For the induction step, given a series of calls as in the claim, let $(a.U)_i$ and $(a.O)_i$ denote the values of $a.U$ and $a.O$ upon entering call `subgameIsWinning`$(a, \mathcal{M}_i)$. The same notation applies to all children b of a. By induction hypothesis, for all children b of a, $b.U$ 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 a, $(b.U)_{i+1} = (b.U)_i$ and $(b.O)_{i+1} = (b.O)_i$. Then for all i, $i \geq i_0$, either (I) $(a.O)_{i+1} = (a.O)_i$ or (II) $(a.O)_{i+1} = (a.O)_i \wedge \mathsf{MI}(L_i, FV(L_i) \setminus \mathit{Rigid}(a), \mathcal{M}_i)$ where $L_i = a.F \wedge \bigwedge_{a \to b}((b.p \wedge \neg(b.U)_i) \vee (\neg b.p \wedge (b.O)_i))$.
Case (II) applies only if $\mathcal{M}_i \models (a.O)_i$ (if we enter the main loop $\mathcal{M}_i \models \neg(a.U)_i \wedge (a.O)_i$), $\mathcal{M}_i \models \neg(a.O)_{i+1}$, and `subgame_is_winning`$(a, \mathcal{M}_i)$ returns *false* (see lines 10-11 in Fig. 2 and Step (3) in the description of `subgameIsWinning`). 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), MI is applied to the same formula, yielding a series of calls to MI as in Def. 6. By convergence of MI, for all i, $i \geq i_0$, $(a.O)_i$ is updated only a finite number of times.
Similarly, for all i, $i \geq i_0$, either (I) $(a.U)_{i+1} = (a.U)_i$ or (II) $(a.U)_{i+1} = (a.U)_i \vee \mathsf{MG}(L'_i, FV(L'_i) \setminus \mathit{Rigid}(a), \mathcal{M}_i)$ where $L'_i = a.F \wedge \bigwedge_{a \to b}((b.p \wedge \neg(b.O)_i) \vee (\neg b.p \wedge (b.U)_i))$. Case (II) applies only if $\mathcal{M}_i \models \neg(a.U)_i$, $\mathcal{M}_i \models (a.U)_{i+1}$, and `subgame_is_winning`$(a, \mathcal{M}_i)$ returns *true* (see lines 15-16 in Fig. 2 and Step (5) in the description of `subgameIsWinning`). 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), MG is applied to the same formula, yielding a series of calls to MG as in Def. 6. By convergence of MG, for all i, $i \geq i_0$, $(a.U)_i$ is updated only a finite number of times.

A.4 Proof of Theorem 3

Consider a call `subgameIsWinning`$(a, \mathcal{M})$ for a node a in T. The base case (a is a leaf) is trivial. The induction hypothesis is that the claim holds for all children $b_1, \ldots, b_n$ of a. For the induction step, if `subgameIsWinning`$(a, \mathcal{M})$ 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 `subgameIsWinning`$(a, \mathcal{M})$ 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 $b_{j(i)}$, $1 \leq j(i) \leq n$, for which $\mathcal{M}_i(b_{j(i)}.p) = \texttt{subgameIsWinning}(b_{j(i)}, \mathcal{M}_i)$ (line 19 in Fig. 2) so that `winningForallChildren` returns *false*. It follows that $\texttt{subgameIsWinning}(a, \mathcal{M})$ 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_k, $1 \leq k \leq n$, and such that $W_{i,k} = 1$ if $\mathcal{M}_i(b_k.p) \neq \texttt{subgameIsWinning}(b_k, \mathcal{M}_i)$, $W_{i,k} = 0$ if $\mathcal{M}_i(b_k.p) = \texttt{subgameIsWinning}(b_k, \mathcal{M}_i)$, and $W_{i,k} = \perp$ if `subgameIsWinning` is not invoked on $(b_k, \mathcal{M}_i)$. For all k, $1 \leq k \leq n$, let $D_k = \{i \mid W_{i,k} = 0\}$. By projecting on the node argument, we extract from $\mathcal{S}$ up to n (possibly infinite) series of calls $\{\texttt{subgameIsWinning}(b_k, \mathcal{M}_i)\}_{i\in D_k}$. Consider anyone of these series and let us temporarily rename b_k as b for simplicity. For all the calls $\texttt{subgameIsWinning}(b, \mathcal{M}_i)$ 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

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

If $\mathcal{M}_i(b.p) = \mathit{true}$, then before the call $\mathcal{M}_i \models \neg b.U$, and since the call also 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 (5) in the description of `subgameIsWinning`). Similarly, if $\mathcal{M}_i(b.p) = \mathit{false}$, then before the call $\mathcal{M}_i \models b.O$, and since the call also 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 `subgameIsWinning`). 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 $\{\texttt{subgameIsWinning}(b_k, \mathcal{M}_i)\}_{i\in D_k}$ are terminating. Therefore, Lemma 1 applies to each of these series, establishing that $b_k.U$ and $b_k.O$ get updated only a finite number of times. Therefore, all the series $\{\texttt{subgameIsWinning}(b_k, \mathcal{M}_i)\}_{i\in D_k}$ are finite, which contradicts the existence of the infinite series $\mathcal{S}$.

A.5 Proof of Theorem 4

The proof is by structural induction on the game tree T. Let $r = \mathit{root}(T)$.
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. 9 and 2 is vacuously true.
Induction hypothesis: the claim holds for all children b of r.
Induction step: we distinguish the two directions.
$\Rightarrow$) By hypothesis, $\mathcal{G}$ is winning for $\mathcal{M}$, that is, there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $\mathit{Var}(r)$ that fulfills Def. 2. We build an extension $\mathcal{M}''$ of $\mathcal{M}'$ to $FV(LF(r))$ that fits Def. 9. First, $FV(LF(r)) = FV(r.F) \cup \{b.p \mid r \rightarrow b\} \cup \bigcup_{r\rightarrow b} FV(LF(b))$. By Def. 1, $FV(r.F) \subseteq \mathit{Rigid}(r) \cup \mathit{Var}(r)$ and $\{b.p \mid r \rightarrow b\} \subseteq \mathit{Var}(r)$. Since $\mathcal{M}$ interprets the variables in $\mathit{Rigid}(r)$ and $\mathcal{M}'$ extends $\mathcal{M}$ to interpret the variables in $\mathit{Var}(r)$, we need to consider only the variables in $\bigcup_{r\rightarrow b} FV(LF(b))$. Since $FV(LF(b))$ may contain variables that are in $\mathit{Rigid}(b) = \mathit{Rigid}(r) \cup \{\bar{x}\}$ for $\bar{x}$ the local variables of r and $\bar{x} \subseteq \mathit{Var}(r)$, $\mathcal{M}''$ only needs to add interpretations of the variables in $FV(LF(b)) \setminus \mathit{Rigid}(b)$ for all children b of r. Let b be a child of r such that $\mathcal{M}'(b.p) = \mathsf{true}$. Then, for all $y \in FV(LF(b)) \setminus \mathit{Rigid}(b)$, let $\mathcal{M}''$ assign

an arbitrary value to y. Let b be a child of r such that $\mathcal{M}'(b.p) = \mathsf{false}$. Since $\mathcal{G}$ is winning for $\mathcal{M}$, by Def. 2, $\mathcal{G}_b$ is winning for $\mathcal{M}'$, and by induction hypothesis $\mathcal{G}_b$ is winning with look-ahead for $\mathcal{M}'$, that is, there exists an extension $\mathcal{M}'_b$ of $\mathcal{M}'$ fulfilling Def. 9 for $\mathcal{G}_b$. Then, for all $y \in FV(LF(b)) \setminus Rigid(b)$, let $\mathcal{M}''(y) = \mathcal{M}'_b(y)$ (†).
This construction of $\mathcal{M}''$ does not assign two different values to the same variable, because if b and b' are two 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. 9. 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 $\mathcal{M}'(b.p) = \mathcal{M}''(b.p) = \mathsf{false}$, 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. 9. Let $b \in \mathsf{FAN}(r, \mathcal{M}'')$ be a descendant of r via a path $r \to a_1 \to \ldots \to a_n \to b$. If $n = 0$, b is a child of r, and $\mathcal{M}''(b.p) = \mathcal{M}'(b.p) = \mathsf{true}$. Since $\mathcal{G}$ is winning for $\mathcal{M}$ with extension $\mathcal{M}'$, $\mathcal{G}_b$ is losing for $\mathcal{M}'$. Since $\mathcal{M}''$ is an extension of $\mathcal{M}'$, $\mathcal{G}_b$ is losing also for $\mathcal{M}''$. If $n > 0$, a_1 is a child of r, and $\mathcal{M}''(a_1.p) = \mathcal{M}'(a_1.p) = \mathsf{false}$. Since $\mathcal{G}$ is winning for $\mathcal{M}$ with extension $\mathcal{M}'$, $\mathcal{G}_{a_1}$ is winning for $\mathcal{M}'$. By induction hypothesis, $\mathcal{G}_{a_1}$ is winning with look-ahead for $\mathcal{M}'$ with some extension $\mathcal{M}'_{a_1}$. By Def. 9 applied to a_1, $\mathcal{G}_b$ is losing for $\mathcal{M}'_{a_1}$. By (†), $\mathcal{M}''$ is an extension of $\mathcal{M}'$ that interprets all variables in $FV(LF(a_1)) \setminus Rigid(a_1)$ like $\mathcal{M}'_{a_1}$ does. Thus, $\mathcal{G}_b$ is losing for $\mathcal{M}''$.
$\Leftarrow$) By hypothesis, $\mathcal{G}$ is winning with look-ahead for $\mathcal{M}$, that is, there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(LF(r))$ that fulfills Def. 9. We show that $\mathcal{M}'$ fulfills Condition (i) in Def. 2: indeed, $\mathcal{M}' \models LF(r)$ implies $\mathcal{M}' \models r.F$. We show that $\mathcal{M}'$ fulfills Condition (ii) in Def. 2. For all children b of r such that $\mathcal{M}'(b.p) = \mathsf{true}$, by Def. 9, $\mathcal{G}_b$ is losing for $\mathcal{M}'$. For all children b of r such that $\mathcal{M}'(b.p) = \mathsf{false}$, $\mathcal{M}' \models LF(r)$ implies $\mathcal{M}' \models LF(b)$, and $\mathcal{M}'$ fulfills Def. 9 also for $\mathcal{G}_b$, so that $\mathcal{G}_b$ is winning with look-ahead for $\mathcal{M}'$. By induction hypothesis, $\mathcal{G}_b$ is winning for $\mathcal{M}'$. Therefore, $\mathcal{M}'$ fulfills Def. 2 and $\mathcal{G}$ is winning for $\mathcal{M}$.

A.6 Proof of Theorem 5

Consider a call `optiSubgameIsWinning`$(a, \mathcal{M})$. 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 game tree T_a of $\mathcal{G}_a$.
Base case: a is a leaf. Formula L is assigned $LF(a) = a.F$ since a has no children, and SMA is invoked to find an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(a.F)$ such that $\mathcal{M}' \models a.F$. If no such extension is found, $\mathsf{MI}(a.F, FV(a.F) \setminus Rigid(a), \mathcal{M})$ returns a quantifier-free formula O and the function returns $\mathsf{UNSAT}(O)$ on line 6 in Fig.4. By Def. 4, $a.\psi = \exists \bar{x}.a.F$. By the specification of MI, we have $\mathcal{M} \not\models O$ and $[\![a.\psi]\!] \subseteq [\![O]\!]$, so that the postconditions hold.
If SMA returns an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(a.F)$ such that $\mathcal{M}' \models a.F$, `reasons` is assigned $\top$. Since a has no descendants, `winningForallDescendants`

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

– Suppose $\texttt{optiSubgameIsWinning}(a, \mathcal{M})$ returns $\mathsf{UNSAT}(O)$ on line 6, because SMA could not extend $\mathcal{M}$ to a model of
$$L = LF(a) \wedge \bigwedge_{a \to^+ b} (b.p \Rightarrow \neg b.U).$$
We must show that $[\![a.\psi]\!] \subseteq [\![O]\!]$ and $\mathcal{M} \not\models O$. The latter is directly a consequence of O being generated by MI from L and $\mathcal{M}$. For the former, let $\mathcal{M}_O$ be a model such that $\mathcal{M}_O \models a.\psi$. By Thm. 1, $\mathcal{G}_a$ is winning for $\mathcal{M}_O$. By Thm. 4, $\mathcal{G}_a$ is winning with look-ahead for $\mathcal{M}_O$. By Def. 9, $\mathcal{M}_O$ can be extended into a model $\mathcal{M}'_O$ of $LF(a)$ such that for all $b \in \mathsf{FAN}(a, \mathcal{M}'_O)$, $\mathcal{G}_b$ is losing for $\mathcal{M}'_O$, which means $\mathcal{M}'_O \not\models b.\psi$ (Thm. 1), which implies $\mathcal{M}'_O \not\models b.U$ by pre-condition I, so that $\mathcal{M}'_O \models \neg b.U$ (*).
Now we have that $\mathcal{M}'_O \models LF(a)$ and we want to show that $\mathcal{M}'_O \models L$. To this end, we assume that $\mathcal{M}'_O(c.p) = \mathsf{false}$ for all descendants c of a beyond the first alternation nodes, that is, for all nodes c such that $a \to^+ c$ and $c \notin \mathsf{NAN}(a, \mathcal{M}'_O) \cup \mathsf{FAN}(a, \mathcal{M}'_O)$ (†).
We show that this assumption causes no loss of generality. Indeed, forcing $\mathcal{M}'_O(c.p) = \mathsf{false}$ for such nodes affects neither $\mathsf{NAN}(a, \mathcal{M}'_O)$, nor $\mathsf{FAN}(a, \mathcal{M}'_O)$, nor $\mathcal{M}'_O(b.p) = \mathsf{true}$ for all $b \in \mathsf{FAN}(a, \mathcal{M}'_O)$. Also, this assumption does not affect the fact that $\mathcal{M}'_O \models LF(a)$. Indeed, $LF(a)$ has the form:
$$a.F \wedge \bigwedge_{a \to^+ b} \{\neg a_1.p \Rightarrow \cdots \Rightarrow \neg a_n.p \Rightarrow b.F \mid a \to a_1 \to \cdots \to a_n \to b\}.$$
Therefore, forcing $\mathcal{M}'_O(c) = \mathsf{false}$ for every node c that is below some node $b \in \mathsf{FAN}(a, \mathcal{M}'_O)$ does not affect the truth value of $LF(a)$, because $\mathcal{M}'_O(b.p) = \mathsf{true}$ so that $\mathcal{M}'_O(\neg b.p) = \mathsf{false}$ and hence any implication in $LF(a)$ involving c necessarily evaluates to true.
Next, $\mathcal{M}'_O$ satisfies L, because it satisfies $LF(a)$ and also $b.p \Rightarrow \neg b.U$ for all descendants b of a: if $b \in \mathsf{FAN}(a, \mathcal{M}'_O)$ then $\mathcal{M}'_O(b.p) = \mathsf{true}$ and we know that $\mathcal{M}'_O \models \neg b.U$ by (*); if $b \in \mathsf{NAN}(a, \mathcal{M}'_O)$ then $\mathcal{M}'_O(b.p) = \mathsf{false}$, so that $\mathcal{M}'_O(b.p \Rightarrow \neg b.U) = \mathsf{true}$; and if $b \notin \mathsf{NAN}(a, \mathcal{M}'_O) \cup \mathsf{FAN}(a, \mathcal{M}'_O)$, then $\mathcal{M}'_O(b.p) = \mathsf{false}$ by the assumption (†), so that $\mathcal{M}'_O(b.p \Rightarrow \neg b.U) = \mathsf{true}$.
Since $\mathcal{M}'_O$ satisfies L, and O is generated by MI from L and $\mathcal{M}$, by the specification of MI it follows that $\mathcal{M}_O$ satisfies O. Therefore, also the postcondition $[\![a.\psi]\!] \subseteq [\![O]\!]$ holds.

– Suppose $\texttt{optiSubgameIsWinning}(a, \mathcal{M})$ returns $\mathsf{SAT}(U)$ on line 11, because SMA found an extension $\mathcal{M}'$ satisfying L, and hence $LF(a)$. Furthermore, `winningForallDescendants` constructed a formula

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

where, for all $b \in \mathsf{FAN}(a, \mathcal{M}')$, O_b is an over-approximation of $b.\psi$ that was returned as $\mathsf{UNSAT}(O_b)$ by a recursive call $\texttt{optiSubgameIsWinning}(b, \mathcal{M}')$. By the post-condition of that recursive call, $\mathcal{M}' \not\models O_b$. By Thm. 1, $\mathcal{G}_b$ is losing for $\mathcal{M}'$. Since this holds for all $b \in \mathsf{FAN}(a, \mathcal{M}')$, we have that $\mathcal{M}'$ fulfills Def. 9.
We show that this property holds in general: every model that satisfies $L' = (LF(a) \wedge \texttt{reasons})$ fulfills Def. 9. To this end, we show that every model that satisfies `reasons` fulfills Condition (ii) in Def. 9. Let $\mathcal{M}''$ be a model that satisfies `reasons`. It follows that $\mathsf{FAN}(a, \mathcal{M}'') = \mathsf{FAN}(a, \mathcal{M}')$ and $\mathsf{NAN}(a, \mathcal{M}'') = \mathsf{NAN}(a, \mathcal{M}')$. Also, for all $b \in \mathsf{FAN}(a, \mathcal{M}')$, $\mathcal{M}'' \models \neg O_b$, and hence by Thm. 1, $\mathcal{G}_b$ is losing for $\mathcal{M}''$. Thus, $\mathcal{M}'$ fulfills Condition (ii) of Def. 9.
By the specification of MG, the application of MG 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. 9. This means that for all models $\mathcal{M}_U \in [\![U]\!]$, game $\mathcal{G}_a$ is winning with look-ahead for $\mathcal{M}_U$, and hence by Thm. 4 game $\mathcal{G}$ is winning for $\mathcal{M}_U$, and hence $\mathcal{M}_U \models a.\psi$ by Thm. 1. This shows that $[\![a.U]\!] \subseteq [\![a.\psi]\!]$, so that the postconditions hold.

A.7 Proof of Theorem 6

For every node a, where $\bar{z}_a = \mathit{Rigid}(a)$, we build
– a finite set of under-approximations $U_{a,1}[\bar{z}_a], \ldots, U_{a,n_a}[\bar{z}_a]$ of $a.\psi$ and
– a finite set of over-approximations $O_{a,1}[\bar{z}_a], \ldots, O_{a,m_a}[\bar{z}_a]$ of $a.\psi$
such that
(i) the following property is an invariant of $\texttt{optiSubgameIsWinning}(a, \mathcal{M})$ where $\mathcal{M}$ extends $\mathcal{M}_0$ to $\bar{z}_a$ (see Fig. 4):
(*) *For all descendants b of a, $b.U$ is a disjunction of a subset of* $\{U_{b,i}[\bar{z}_b]\}_{i=1}^{n_b}$;
(ii) and if Property (*) holds as a pre-condition to $\texttt{optiSubgameIsWinning}(a, \mathcal{M})$, then the call halts and returns either $\mathsf{SAT}(U_{a,i}[\bar{z}_a])$ for some i, $1 \leq i \leq n_a$, or $\mathsf{UNSAT}(O_{a,j}[\bar{z}_a])$ for some j, $1 \leq j \leq m_a$.
The construction is by induction on T_a.
Consider a call $\texttt{optiSubgameIsWinning}(a, \mathcal{M})$ with Property (*) holding as a pre-condition. We simultaneously show that each recursive call halts and that Property (*) holds throughout the execution of $\texttt{optiSubgameIsWinning}(a, \mathcal{M})$, and in particular Property (*) is a loop invariant for all loops in `optiSubgameIsWinning`. For the purpose of this proof, we pretend that function `winningForallDescendants` is inlined, so in particular we show that Property (*) is a loop invariant for the loop ranging over $b \in \mathsf{FAN}(a, \mathcal{M})$.

When reaching a recursive call in that loop, assuming that Property (*) holds at that point as a loop invariant, the recursive call's own pre-condition is satisfied and the induction hypothesis on T_b concludes that the recursive call terminates, returning either $\mathsf{SAT}(U_{b,i}[\bar{z}_b])$ for some i, $1 \leq i \leq n_b$, or $\mathsf{UNSAT}(O_{b,j}[\bar{z}_b])$ for some j, $1 \leq j \leq m_b$. If and when $b.U$ is updated, with instruction $b.U \leftarrow b.U \vee U$ (line 17 of Fig. 4), $b.U$ remains a disjunction of a subset of the $\{U_{b,i}[\bar{z}_b]\}_{i=1}^{n_b}$, so that the loop invariant (*) is preserved. Therefore, the entire block represented by the call to `winningForallDescendants`$(a, \mathcal{M}, \text{reasons})$ terminates and Property (*) is an invariant of that block. Hence, it is also a loop invariant for the outer (`while` true) loop in `optiSubgameIsWinning`$(a, \mathcal{M})$.
Moreover, for every descendant b of a, $b.U$ can be updated at most n_b times. This implies the termination of the outer (`while` true) loop, since every iteration that does not exit `optiSubgameIsWinning`$(a, \mathcal{M})$ must update some $b.U$ at least once.
Furthermore, since $b.U$ is a disjunction of a subset of the $U_{b,i}[\bar{z}_b]$, it means that the space of possible values for $b.U$ is finite (of size $\sum_{n=0}^{n_b} \binom{n_b}{n} = \sum_{n=0}^{n_b} \frac{n_b!}{n! \cdot (n_b - n)!}$), and so is the space of possible values for L. Therefore, we can use the hypothesis that MI has finite basis, to infer the existence of the $O_{a,1}[\bar{z}_a], \ldots, O_{a,m_a}[\bar{z}_a]$.
By the same reasoning, at the end of the main loop the space of possible values for the variable `reasons` is finite, and we use the hypothesis that MG has finite basis, to infer the existence of the $U_{a,1}[\bar{z}_a], \ldots, U_{a,n_a}[\bar{z}_a]$. This terminates the proof.