# Incremental Linearization for Satisfiability and Verification Modulo Nonlinear Arithmetic and Transcendental Functions

ALESSANDRO CIMATTI, Fondazione Bruno Kessler, Italy
ALBERTO GRIGGIO, Fondazione Bruno Kessler, Italy
AHMED IRFAN, Fondazione Bruno Kessler and DISI, University of Trento, Italy
MARCO ROVERI, Fondazione Bruno Kessler, Italy
ROBERTO SEBASTIANI, DISI, University of Trento, Italy

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a first-order formula with respect to some theory or combination of theories; Verification Modulo Theories (VMT) is the problem of analyzing the reachability for transition systems represented in terms of SMT formulae. In this paper we tackle the problems of SMT and VMT over the theories of nonlinear arithmetic over the reals (NRA) and of NRA augmented with transcendental (exponential and trigonometric) functions (NTA).

We propose a new abstraction-refinement approach for SMT and VMT on NRA or NTA, called *Incremental Linearization*. The idea is to abstract nonlinear multiplication and transcendental functions as uninterpreted functions in an abstract space limited to linear arithmetic on the rationals with uninterpreted functions. The uninterpreted functions are incrementally axiomatized by means of upper- and lower-bounding piecewise-linear constraints. In the case of transcendental functions, particular care is required to ensure the soundness of the abstraction.

The method has been implemented in the MathSAT SMT solver and in the nuXmv model checker. An extensive experimental evaluation on a wide set of benchmarks from verification and mathematics demonstrates the generality and the effectiveness of our approach.

CCS Concepts: • **Theory of computation → Automated reasoning**; **Constraint and logic programming**; **Verification by model checking**; *Logic and verification*; • **Mathematics of computing →** *Solvers*;

**ACM Reference Format:**
Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. 2018. Incremental Linearization for Satisfiability and Verification Modulo Nonlinear Arithmetic and Transcendental Functions . *ACM Trans. Comput. Logic* 1, 1, Article 1 (January 2018), 52 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

*General context of SMT and VMT. Satisfiability Modulo Theories (SMT)* is the problem of deciding the satisfiability of a first-order formula with respect to some theories of interest (e.g. theory of linear arithmetic, of arrays, of bit-vectors) and their combination thereof [7]. Since its inception, SMT has been a thriving research area, also leveraging developments in propositional satisfiability (SAT), with applications in formal verification, planning, security, and synthesis. SMT formulae allow for the symbolic representation of many expressive forms of infinite-state transition systems. We refer to the problem of analysing such transition systems (e.g. checking reachability) as *Verification*

Authors' addresses: Alessandro Cimatti, Fondazione Bruno Kessler, Trento, Italy; Alberto Griggio, Fondazione Bruno Kessler, Trento, Italy; Ahmed Irfan, Fondazione Bruno Kessler and DISI, University of Trento, Trento, Italy; Marco Roveri, Fondazione Bruno Kessler, Trento, Italy; Roberto Sebastiani, DISI, University of Trento, Trento, Italy.


1529-3785/2018/1-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Modulo Theories (VMT).* Various algorithms for VMT exist [3, 22, 32]. They extend effective SAT-based verification algorithms from the finite case [10, 13, 54, 63] by means of SMT solving and abstraction [66].

*The crux of nonlinearity.* Powerful and effective SMT and VMT techniques and tools are available for the quantifier-free theories[1] of Uninterpreted Functions (UF) and Linear Arithmetic (LA), either over the reals (LRA) or the integers (LIA), as well as their combinations (UFLRA, UFLIA). A fundamental challenge is to go beyond the linear case, by introducing nonlinear polynomials – over the reals (NRA) or over the integers (NIA) – plus transcendental functions such as exponentiation and trigonometric functions. (We denote the theory NRA enriched with transcendental functions as NTA.) In fact, the expressive power of nonlinear arithmetic and transcendental functions is required by many application domains (e.g. railways, aerospace, control software, and cyber-physical systems).

Unfortunately, dealing with nonlinearity is a very hard challenge. Going from SMT(LRA) to SMT(NRA) yields a complexity gap that results in a computational barrier in practice – most available complete solvers rely on Cylindrical algebraic decomposition (CAD) techniques [27], which require double exponential time in worst case. Adding transcendental functions to NRA exacerbates the problem even further, because reasoning on NTA has been shown to be undecidable [61]. Similarly, reasoning in NIA is undecidable [53] (the result of Hilbert's 10th problem).

A similar situation can be observed for VMT. In general, given a theory, VMT is harder than SMT, due to the underlying notion of reachability: VMT is undecidable even for relatively simple theories such as LRA [42]. Yet, VMT(LRA) tools are reasonably effective in practice, based on the power of SMT(LRA) solvers and the ability to automatically construct abstractions. This is hardly the case for reachability over NTA transition systems.

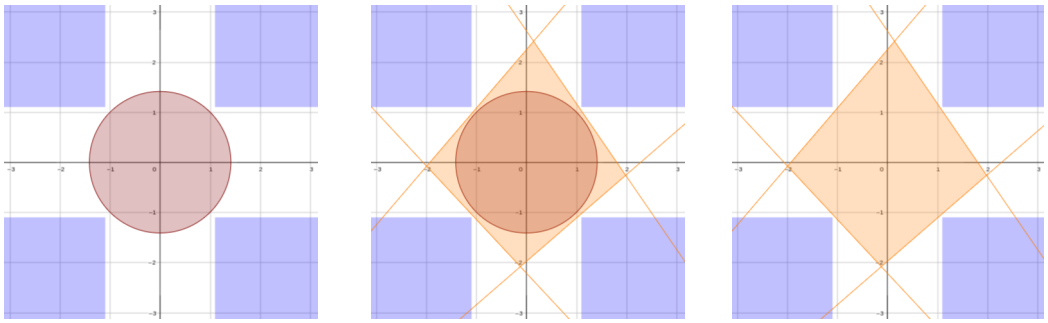
*Incremental Linearization.* In this paper, we take on the challenge of dealing with the quantifier-free theory of nonlinear arithmetic, also with transcendental functions, over the reals in SMT as well as in VMT. We propose a practical and unifying approach, referred to as *Incremental Linearization*, that trades the use of expensive, exact solvers for nonlinear arithmetic for an abstraction-refinement loop on top of much less expensive solvers for linear arithmetic and uninterpreted functions. The approach is based on an abstraction-refinement loop, using SMT(UFLRA) as abstract space. The Uninterpreted Functions are used to model nonlinear and transcendental functions. Then, we iteratively and incrementally axiomatize nonlinear multiplication and transcendental functions with a lemma-on-demand approach. Specifically, we eliminate spurious interpretations in SMT(UFLRA) (or spurious counterexample traces, in the case of VMT), by tightening the piecewise-linear envelope around the (uninterpreted counterpart of the) transcendental functions.

The underlying rationale is that, for many practical problems, reasoning with full precision over nonlinear and transcendental functions may not be necessary. For example, constructing a piecewise-linear invariant may be sufficient to prove that the (nonlinear) transition system at hand satisfies a given property. The linearization is performed incrementally and only when and where needed, driven by the spurious counterexamples.

We remark that the proposed approach also works in the case of NIA. However, its evaluation over NIA benchmarks is left as future work.

*Example 1.1.* Consider the following constraints: $x * x + y * y \leq 2 \wedge (x \geq 1.1 \vee x \leq -1.1) \wedge (y \geq 1.1 \vee y \leq -1.1)$, which are graphically shown in Fig. 1a. We want to check whether their intersection is nonempty or not. One way is to use some nonlinear solving method to answer that question.

[1] In the following, we only consider quantifier-free theories, and we abuse the accepted notation and omit the "QF_" prefix in the names of the theories.



(a) Is the intersection of the circle and squares nonempty?

(b) Linear approximation of circle.

(c) Linear constraints for solving the problem.

Fig. 1. Illustration of Example 1.1

However, notice that the intersection can be shown empty by approximating the circle: this can be done by replacing nonlinear multiplications, i.e., $x * x$ and $y * y$ with uninterpreted functions $f_*(x, x)$ and $f_*(y, y)$, respectively, and adding the following linear constraints over the uninterpreted functions $(f_*(x, x) \geq -2.8 * x - 1.96) \wedge (f_*(x, x) \geq -3 * x - 2.25) \wedge (f_*(x, x) \geq 3.2 * x - 2.56) \wedge (f_*(x, x) \geq 2.6 * x - 1.69) \wedge (f_*(y, y) \geq 2.4 * y - 1.44) \wedge (f_*(y, y) \geq -2.8 * y - 1.96) \wedge (f_*(y, y) \geq 2.2 * y - 1.21) \wedge (f_*(y, y) \geq -3 * y - 2.25)$. Clearly, as depicted in Fig. 1b the additional constraints approximate the circle. Therefore, we can answer the question by solving the linear problem (as shown in Fig. 1c) using some linear method.

*Incremental linearization for SMT.* Consider first the case of Satisfiability Modulo Theories. In the abstract space, nonlinear multiplication between variables is modeled as a binary uninterpreted function $f_*$. When spurious models are found, the abstraction is tightened by the incremental introduction of linear constraints, including tangent planes resulting from differential calculus, and monotonicity constraints.

In order to deal with NTA, we rely on several insights. First, irrational numbers are basically inevitable with the most common transcendental functions, such as exponential and sine, as shown by the Hermite and Niven theorems (see [57]). The challenge is to obtain provably correct (rational-coefficient) approximations in LRA. We use Taylor series to exactly compute suitable accurate rational coefficients for the piecewise-linear envelopes. Notice that, nonlinear polynomials are only used to numerically compute the coefficients, i.e., no SMT solving in the theory of nonlinear arithmetic is needed.

The refinement is based on the addition, in the abstract space, of piecewise-linear axiom instantiations, which upper- and lower-bound the candidate solutions, ruling out spurious interpretations. To compute such piecewise-linear bounding functions, the concavity of the curve is taken into account to identify the actual approximation interval. In order to deal with *trigonometric functions*, we leverage the property of periodicity, so that the axiomatization is only done in the interval between $-\pi$ and $\pi$, and deal with the external intervals by reduction.

Finally, we adopt a logical method to conclude the existence of a solution without explicitly constructing it. We use a sufficient criterion that consists in checking whether the formula is satisfiable under all possible interpretations of the uninterpreted functions (representing the transcendental functions) that are consistent with some rational interval bounds within which the correct values for the transcendental functions are guaranteed to exist. We encode the problem as

an SMT(LRA) satisfiability check, such that an unsatisfiable result implies the satisfiability of the original SMT(NTA) formula.

*Incremental linearization for VMT.* By means of Incremental linearization, we also tackle invariant checking for transition systems over NRA and transcendental functions. The key insight is to implement the Counterexample Guided Abstraction Refinement (CEGAR) loop on top of an abstract version of the transition system expressed over UFLRA. We leverage the characteristics of a recently introduced approach [22] based on the combination of IC3 and predicate abstraction.

We remark that this approach based on incremental linearization has strong advantages over other VMT(NTA) approaches that could be obtained from traditional SMT-based algorithms by delegating the management of nonlinearity to an SMT(NTA) solver. Bounded Model Checking and k-induction [34, 63] would be relatively simple to implement (given an SMT solver for the required theory) but have very limited effectiveness when it comes to proving properties over infinite-state transition systems. Extending other approaches (e.g. interpolation, IC3 [22, 54]) to handle nonlinearities at the level of the solver would require the SMT(NRA) solver (or, worse, the SMT(NTA) solver), to carry out interpolation or quantifier elimination, and to proceed incrementally. These extra functions are usually not available, or they have a very high computational cost.

*Implementation and Experimental Evaluation.* We have implemented our SMT and VMT approaches based on incremental linearization, and a thorough experimental evaluation has been carried out. The MathSAT SMT solver [23] has been extended with a tightly integrated abstraction-refinement loop for NRA and, in the transcendental case, with exponentiation and trigonometric functions. We experimentally evaluated MathSAT on all the SMT-LIB benchmarks from the NRA categories, as well as with benchmarks from SMT-based verification queries over nonlinear transition systems, including Bounded Model Checking of hybrid automata, several mathematical properties from the MetiTarski [1] suite and from other competitor solver distributions. We contrasted MathSAT against various SMT solvers: z3 [31], Yices [33], SMT-RAT [28] based on expensive techniques like CAD, iSAT3 and dReal, based on interval constraint propagation, and with the deductive approach of MetiTarski [1].

As for VMT, the approach has been implemented in the nuXmv model checker [17], relying on the IC3 with Implicit Abstraction [22] engine for VMT(UFLRA). We selected benchmarks over VMT(NRA) and VMT(NTA), and compared nuXmv against multiple approaches working at NRA level, including BMC and k-induction using SMT(NRA), the recent interpolation-based iSAT3 engine [50], and the static abstraction approach proposed in [18].

*Discussion.* The experimental results show the merits of incremental linearization. The technique is surprisingly effective in SMT, even compared to other complete and more mature approaches. In VMT, incremental linearization significantly outperforms its competitors which are based on interval propagation.

The effectiveness of our approach is possibly explained with the following insights. On the one hand, in contrast to LRA, NRA is a hard-to-solve theory: in practice, most available complete solvers rely on expensive solvers; we try to avoid NRA reasoning, trading it for LRA and UF reasoning. On the other hand, proving properties of practical NRA transition systems may not require the full power of nonlinear solving. In fact, some systems are "mostly-linear" (i.e. nonlinear constraints are associated to a very small part of the system), an example being the Transport Class Model (TCM) for aircraft simulation from the Simulink model library [44]. Furthermore, even NRA transition systems with significant nonlinear dynamics may admit a piecewise-linear invariant of the transition system that is strong enough to prove the property.

Overall, incremental linearization is a general and relatively simple idea that supports approaches to SMT and VMT, and is able to successfully tackle many practical problems.

*Structure of the paper.* The rest of this paper is organized as follows. We present the background in §2. We discuss our technique at the SMT level in §3–§5, first outlining the general ideas in §3, and then providing details on the refinement mechanisms in §4 and on the detection of satisfiable results in §5. We extend our approach to VMT in §6. In §7 we prove the correctness of the overall approach. In §8 we describe some heuristics and provide some implementation details. In §9 we compare our approach with the related work. In §10 we present the results of the experimental evaluation. Finally, in §11 we draw some conclusions and outline directions for future work.

*Note.* This paper brings together and extends the ideas presented in the conference papers [19] (where the case of VMT(NRA) is discussed), and [20] (where the SMT(NTA) case is discussed). The paper differs from and extends [19] and [20] in various directions. First, the paper has been completely rewritten to cover in a systematic manner all the combinations of both SMT and VMT with both NRA and NTA, and present it in the light of incremental linearization. Second, the case VMT(NTA) was not covered in [19, 20]. Third, the SMT approach is now based on a much tighter integration into SMT(UFLRA). Fourth, we provide a theoretical underpinning of the approach. Finally, a much stronger implementation of incremental linearization is now integrated within the official versions of MathSAT and nuXmv, and a more comprehensive set of experiments is provided.

## 2 BACKGROUND

We assume the standard first-order quantifier-free logical setting and standard notions of theory, satisfiability, and logical consequence. We denote formulae with $\varphi, \psi, I, T, P$, terms with $t, s$, variables with $x, y$, constants with $a, b, c$, functions with $f$, TF, each possibly with subscripts. [2] If $X$ is a set of variables, we write $\varphi(X)$ to denote the fact that the all the variables of $\varphi$ are in $X$. We denote with $\varphi\{x \mapsto t\}$ the formula obtained by replacing all the occurrences of $x$ in $\varphi$ with $t$. We use the same notation for terms and models, and we extend it to ordered sequences of variables in the natural way. (E.g., if $\mathbf{x} \doteq x_1, .., x_k$ and $\mathbf{t} \doteq t_1, .., t_k$, then $\varphi\{\mathbf{x} \mapsto \mathbf{t}\}$ s.t. denotes $\varphi\{x_1 \mapsto t_1\}\{... \mapsto ...\}\{x_k \mapsto t_k\}$.) If $\mu$ is a model and $x$ is a variable, we write $\mu[x]$ to denote the value of $x$ in $\mu$, and we extend this notation to terms and formulae in the usual way. If $\Gamma$ is a set of formulae, we write $\bigwedge \Gamma$ (or simply $\Gamma$) to denote the conjunction of all the formulae in $\Gamma$. We write $t_1 < t_2 < t_3$ for $t_1 < t_2 \wedge t_2 < t_3$. (A similar notation is used with "$\leq$".) We abuse the notation and write $t \in \varphi$ to denote that term $t$ occurs in $\varphi$. $abs(t)$ stands for $\text{ITE}(t < 0, -t, t)$, ITE being the standard if-then-else term operator.

### 2.1 Satisfiability Modulo Theories

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a first-order formula with respect to some first-order theory ($\mathcal{T}$) or combination of first-order theories ($\mathcal{T}_1 \cup \mathcal{T}_2$). A formula is satisfiable in $\mathcal{T}$ (or $\mathcal{T}$-satisfiable) if it is satisfiable in a model of $\mathcal{T}$ (also written as $\mathcal{T}$-model). Similarly, a formula is valid in $\mathcal{T}$ (or $\mathcal{T}$-valid) if it is satisfiable in every model of $\mathcal{T}$.

An SMT solver is a decision procedure which solves the SMT problem. The most efficient implementations of SMT solvers use the so-called "lazy approach", where a SAT solver is tightly integrated with a $\mathcal{T}$-solver, that is demanded to decide conjunctions of constraints in the theory $\mathcal{T}$.

There exist several theories that the modern SMT solvers support. In this work we are interested in the following theories: *Equality and Uninterpreted Functions*, *Linear Arithmetic* and *Nonlinear*

[2] As is standard practice in SAT, SMT, CSP, and OR communities, and with a little abuse of terminology, in quantifier-free formulae like "$(A_1 \vee (2 * x_1 + 3 * x_2 \leq 4)) \wedge A_2$" we call "Boolean variables" the propositional atoms $A_i$ and we call "variables" the Skolem constants $x_i$.

*Arithmetic* over the Reals, and in their combinations thereof. The theory of linear real arithmetic (LRA) is the first-order theory with equality whose atoms are linear polynomial constraints interpreted over $\mathbb{R}$, whereas the theory of nonlinear real arithmetic (NRA) is the first-order theory with equality whose atoms are nonlinear polynomial constraints interpreted over $\mathbb{R}$.

We denote with UFLRA the combined theory of UF and LRA.

### 2.2 Verification Modulo Theories

A symbolic transition system $\mathcal{S} \doteq \langle X, I, T\rangle$ is a tuple where $X$ is a finite set of (state) variables, $I(X)$ is a formula denoting the initial states of the system, and $T(X, X')$ is a formula expressing its transition relation, where $X'$ is the set obtained by replacing each element $x \in X$ with $x'$. A state $s_i$ of $\mathcal{S}$ is an assignment to the variables $X$. A path (execution trace) $\sigma_k \doteq s_0, s_1, s_2, \ldots, s_{k-1}$ of length $k$ (possibly infinite) for $\mathcal{S}$ is a sequence of states such that $s_0 \models I$ and $s_i \wedge s_{i+1}\{X \mapsto X'\} \models T$ for all $0 \leq i < k-2$. We denote with $X^{\langle i\rangle}$ the set obtained by replacing $x$ with $x^{\langle i\rangle}$. We call an unrolling of $\mathcal{S}$ of length $k$ the formula $I\{X \mapsto X^{\langle 0\rangle}\} \wedge \bigwedge_{i=0}^{k-1} T\{X \mapsto X^{\langle i\rangle}\}\{X' \mapsto X^{\langle i+1\rangle}\}$.

Verification Modulo Theories (VMT) is the problem of verifying the properties of a symbolic transition system where $I$ and $T$ are expressed as SMT($\mathcal{T}$) formulae for some background theory $\mathcal{T}$. Let $P(X)$ be a formula whose assignments represent a property over the state variables $X$. ($P$ can be seen as representing the "good" states, while $\neg P$ represents the "bad" states.) The invariant verification problem, denoted with $S \models P$, is the problem of checking if for all the finite paths $s_0, s_1, \ldots, s_k$ of $\mathcal{S}$, for all $i$, $0 \leq i \leq k$, $s_i \models P$. Its dual formulation in terms of reachability of $\neg P$ is the problem of finding a path $s_0, s_1, \ldots, s_k$ of $\mathcal{S}$ such that $s_k \models \neg P$.

### 2.3 Nonlinear Arithmetic and Transcendental Functions

We denote with $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ the set of integer, rational and real numbers, respectively. The absolute value of $a \in \mathbb{R}$, denoted by $|a|$, is defined as $|a| = a$ if $a \geq 0$, and $-a$ otherwise. A *monomial* in variables $x_1, x_2, \ldots, x_n$ is a product $x_1^{\alpha_1} * x_2^{\alpha_2} * \ldots * x_n^{\alpha_n}$, where each $\alpha_i$ is a non-negative integer called exponent of the variable $x_i$. When clear from context, we may omit the multiplication symbol $*$ and simply write $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$. A *polynomial* $p$ is a linear combination of monomials with coefficients in $\mathbb{Q}$. We write $\mathbb{Q}[x_1, \ldots, x_n]$ as the sets of the polynomials containing the variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{Q}$. A *univariate polynomial* is a polynomial containing one variable, whereas a *multivariate polynomial* contains more than one variable. A *polynomial constraint* $P$ is of the form $p \bowtie 0$ where $p$ is a polynomial and $\bowtie \in \{<, \leq, =, \neq, >, \geq\}$.

The *total degree* of a monomial is the sum of the exponents of its variables. The *total degree* of a polynomial is the highest degree among its monomials. A monomial is *linear* if it has total degree less than or equal to one, otherwise it is *nonlinear*, and similarly for polynomials. A polynomial constraint $p \bowtie 0$ is a *linear constraint* if $p$ is linear and is a *nonlinear constraint* if $p$ is nonlinear.

A real number $c \in \mathbb{R}$ is a real root of $p \in \mathbb{Q}[x]$ iff $p(c) = 0$. A real number $a \in \mathbb{R}$ is an *algebraic* number iff it is a root of some $p \in \mathbb{Q}[x]$, otherwise it is *transcendental number*. An example of algebraic number is $\sqrt{2}$, while $\pi$ and $e$ are transcendental numbers.

A function over the reals $f : \mathbb{R}^n \to \mathbb{R}$ maps every element in $\mathbb{R}^n$ into a corresponding element in $\mathbb{R}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *univariate* when $n = 1$, and *multivariate* when $n > 1$. A function $y = f(x_1, \ldots, x_n)$ is *algebraic* iff it satisfies a polynomial equation, i.e. there exists a polynomial $p \in \mathbb{Q}[y, x_1, \ldots, x_n]$ such that $\forall x_1, \ldots, x_n.(p(y, x_1, \ldots, x_n) = 0)$. A function is *transcendental* if it is not algebraic [41, 67].

We assume that we have continuous and differentiable functions. If $f$ is a univariate function, we write $\frac{d}{dx} f$ for the first-order derivative of $f$. We also write $f^{(i)}$ for the $i$-th derivative of $f$, and $f'$ and $f''$ for $f^{(1)}$ for $f^{(2)}$, respectively.

Let $l$ and $u$ be two real numbers. We denote open and closed intervals between them as $]l, u[$ and $[l, u]$ respectively. Given a univariate function $f$ over the reals, the *graph* of $f$ is the set of pairs $\{(x, f(x)) \mid x \in \mathbb{R}\}$. We refer to an element $(x, f(x))$ of the graph as a point of $f$.

*Definition 2.1.* The *tangent line* at $a$ to the function $f$, denoted with $\textsc{TanLine}_{f,a}(x)$, is the straight line defined as follows:

$$\textsc{TanLine}_{f,a}(x) \doteq f(a) + \frac{d}{dx}f(a) * (x - a)$$

*Definition 2.2.* Given $a \neq b$, the *secant line* at $[a, b]$ to a function $f$, denoted with $\textsc{SecLine}_{f,a,b}(x)$, is the straight line defined as follows:

$$\textsc{SecLine}_{f,a,b}(x) \doteq \frac{f(a) - f(b)}{a - b} * (x - a) + f(a).$$

*Definition 2.3.* Let $f$ be a univariate function twice differentiable at point $c$. The *concavity* of $f$ at $c$ is the sign of $f''(c)$.

*Definition 2.4.* A univariate function $f$ has an inflection point at $c$ iff it is twice differentiable at $c$, $f''(c) = 0$, and there exists $\epsilon > 0$ such that for all $x \in [c - \epsilon, c[$ has sign $s \neq 0$ and for all $x \in ]c, c + \epsilon]$ has sign $s' \neq 0$ opposite to $s$.

Proposition 2.5. *Let $f$ be a univariate function. If $f''(x) \geq 0$ for all $x \in [l, u]$, then for all $a, x \in [l, u]$ $\textsc{TanLine}_{f,a}(x) \leq f(x)$, and for all $a, b, x \in [l, u]$ $((a \neq b \wedge a \leq x \leq b) \rightarrow \textsc{SecLine}_{f,a,b}(x) \geq f(x))$.*

If $f''(x) \leq 0$, then the dual property holds.

Let $f(x, y)$ be a bivariate function. We write $\frac{d}{dx}f(x, y)$ and $\frac{d}{dy}f(x, y)$ for the first-order partial derivatives of $f(x, y)$ w.r.t. $x$ and $y$, respectively.

*Definition 2.6.* The *tangent plane* at $(a, b)$ to a bivariate function $f(x, y)$, denoted with $\textsc{TanPlane}_{f,a,b}(x, y)$, is defined as follows:

$$\textsc{TanPlane}_{f,a,b}(x, y) \doteq f(a, b) + \frac{d}{dx}f(a, b) * (x - a) + \frac{d}{dy}f(a, b) * (y - b)$$

*Taylor Series and Taylor's Theorem.*

*Definition 2.7.* Let $f(x)$ be $n$-differentiable at $a$. The *Taylor series* of $f$ of degree $n$ centered around $a$ is the polynomial:

$$P_{n,f(a)}(x) \doteq \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!} * (x - a)^i$$

The Taylor series centered around zero is also called *Maclaurin series*.

According to *Taylor's theorem*, any continuous function $f(x)$ that is $(n + 1)$-differentiable can be written as the sum of the Taylor series and the remainder term:

$$f(x) = P_{n,f(a)}(x) + R_{n+1,f(a)}(x)$$

where $R_{n+1,f(a)}(x)$ is the Lagrange form of the remainder, expressible as

$$R_{n+1,f(a)}(x) \doteq \frac{f^{(n+1)}(b)}{(n + 1)!} * (x - a)^{n+1}.$$

for some $b$ between $x$ and $a$.

Although the value of $b$ is not known, an upper bound on the size of the remainder $R^{\mathrm{U}}_{n+1,f(a)}(x)$ at a point $x$ can be defined as:

$$R^{\mathrm{U}}_{n+1,f(a)}(x) \doteq \max_{c\in[\min(a,x),\max(a,x)]}(|f^{(n+1)}(c)|) * \frac{|(x-a)^{n+1}|}{(n+1)!}.$$

From this, we obtain a lower- and an upper-bound for $f(x)$, given by $P_{n,f(a)}(x) - R^{\mathrm{U}}_{n+1,f(a)}(x)$ and $P_{n,f(a)}(x) + R^{\mathrm{U}}_{n+1,f(a)}(x)$ respectively. Clearly, the closer is $a$ to $x$, the tighter the approximation of $f(x)$ will be.

Within this paper we consider univariate exponential and trigonometric transcendental functions. We recall that the graphs of exponential and trigonometric functions have only a finite number of points in $\mathbb{Q}\times\mathbb{Q}$ (e.g. $(0,1)$ for exp, $(0,0)$ for sin). A trigonometric number is an irrational number expressible as $\sin(a\pi)$ with $a\in\mathbb{Q}$. Remarkable points in the graph of sin are $\sin(1/2\pi) = \sqrt{2}/2$ and $\sin(5/6\pi) = 1/2$. Finally, we recall that trigonometric functions like sine are periodic. For example, for all $i\in\mathbb{Z}$, $\sin(a) = \sin(a+2i\pi)$.

## 3 SMT VIA INCREMENTAL LINEARIZATION

We now provide a high-level description of the algorithm for SMT solving on NTA (and hence NRA) based on incremental linearization. Further details will be provided in the next sections.

To simplify the presentation, and when not explicitly stated otherwise, we often implicitly assume w.l.o.g. that all multiplications in the input formula $\varphi$ are either between variables (e.g. $x*y$) or between one constant and one variable (e.g. $3x$), and that all transcendental functions in $\varphi$ are applied to variables (e.g., exp(x)). This can be obtained by recursively substituting non-variable terms $t$ inside multiplications and transcendental function applications with fresh variables $x_t$, and by conjoining $(x_t = t)$ to $\varphi$.

### 3.1 The Main Procedure

The main algorithm is shown in Fig. 2. The main function SMT-NTA-CHECK takes as input a formula $\varphi$ containing non-linear constraints with polynomials and transcendental functions, and returns a Boolean value asserting if $\varphi$ is satisfiable or not. When the formula is found to be unsatisfiable it returns also the set of UFLRA constraints $\Gamma$ such that the formula $\varphi\wedge\bigwedge\Gamma$ can be shown unsatisfiable using an UFLRA SMT solver. Notice that, SMT-NTA-CHECK is not guaranteed to terminate, so that we implicitly assume that it is stopped as soon as some given resource budget (e.g. time, memory, number of iterations) is exhausted.

In order to deal with transcendental models, we adopt a mechanism based on rational approximations of irrational values; a variable $\epsilon$ keeps track of the current precision of approximation, and it is incremented on demand.

First, in line 1 the formula undergoes some NTA-satisfiability-preserving preprocessing step, which produces the formula $\varphi' \doteq \varphi\wedge\varphi_{shift}$ by introducing some fresh real variables $\omega_x$ and by conjoining to $\varphi$ a formula $\varphi_{shift}$ which defines univocally the values of the $\omega_x$'s in terms of some variables $x$'s in $\varphi$ (see §4).

Then the formula $\varphi'$ is abstracted into an over-approximating formula $\widehat{\varphi}$ over the combined theory of linear arithmetic and uninterpreted functions (UFLRA) by invoking SMT-INITIAL-ABSTRACTION (line 2). $\widehat{\varphi}$ is the result of replacing each nonlinear term $x*y$ with $f_*(x,y)$, and each transcendental term $\mathrm{TF}(x)$ with $f_{\mathrm{TF}}(x)$, s.t. $f_*(.)$ and $f_{\mathrm{TF}}(.,.)$ are uninterpreted functions, and the symbol $\pi$ with the new symbol $\widehat{\pi}$ (see §4). (We remark that, linear multiplications, like e.g. $c*x$ where $c$ is a constant, are not replaced.) The set of constraints $\Gamma$ is initialized to the empty set, and the precision

⟨**bool**, constraint-set⟩ SMT-NTA-CHECK ($\varphi$):
1. $\varphi' :=$ SMT-PREPROCESS($\varphi$)
2. $\widehat{\varphi} :=$ SMT-INITIAL-ABSTRACTION($\varphi'$)
3. $\Gamma := \emptyset$
4. $\epsilon :=$ INITIAL-PRECISION ()
5. **while true**:
6. $\quad \langle sat, \widehat{\mu} \rangle :=$ SMT-UFLRA-CHECK ($\widehat{\varphi} \wedge \bigwedge \Gamma$)
7. $\quad$ **if not** $sat$:
8. $\quad\quad$ **return** ⟨**false**, $\Gamma$⟩
9. $\quad \langle sat, \Gamma' \rangle :=$ CHECK-REFINE ($\widehat{\varphi}, \widehat{\mu}, \epsilon$)
10. $\quad$ **if** $sat$:
11. $\quad\quad$ **return** ⟨**true**, $\emptyset$⟩
12. $\quad \Gamma := \Gamma \cup \Gamma'$

Fig. 2. The main procedure for solving SMT(NTA) via abstraction to SMT(UFLRA) and refinement.

variable $\epsilon$ is initialized to some (small) real value by calling the function INITIAL-PRECISION (lines 3-4).

Then the algorithm enters a loop (lines 5-12). At each iteration, the approximation $\widehat{\varphi}$ of $\varphi'$ is refined by adding new UFLRA constraints to $\Gamma$ that rule out spurious solutions. The loop maintains the invariant that $\widehat{\varphi} \wedge \bigwedge \Gamma$ is an over-approximation of $\varphi$, (see Lemma 7.3). The process iterates until either the formula $\widehat{\varphi} \wedge \bigwedge \Gamma$ is proved unsatisfiable in SMT(UFLRA) by invoking the standard SMT-solving function SMT-UFLRA-CHECK (lines 6-8), or $\widehat{\varphi} \wedge \bigwedge \Gamma$ is proved UFLRA-satisfiable and the satisfiability result can be lifted to a satisfiability result for the original formula $\varphi$ by means of a refinement process (lines 9-11). The function CHECK-REFINE (see §3.2) takes as input the abstracted formula $\widehat{\varphi}$, the current abstract model $\widehat{\mu}$ and the precision $\epsilon$, and it returns ⟨**true**, $\emptyset$⟩ if it achieves proving the NTA-satisfiability of $\varphi'$, it returns ⟨**false**, $\Gamma'$⟩ if it fails, $\Gamma'$ being a non-empty set of UFLRA constraints that rule out $\widehat{\mu}$ (and other spurious solutions). When none of the above loop-exit condition occurs, the novel constraints in $\Gamma'$ which were found by CHECK-REFINE are added to $\Gamma$ before entering into next loop.

*Unsat-core extraction.* As a byproduct of the algorithm in Fig. 2, since state-of-the-art SMT solvers for UFLRA can return unsatisfiable cores when the input formula is found unsatisfiable, we can easily modify SMT-NTA-CHECK to produce also an unsatisfiable core for NTA when $\varphi$ in NTA-unsatisfiable. This is done by a variation of the lemma-lifting technique in [24]: when SMT-UFLRA-CHECK returns **false**, then it can be asked to produce an unsatisfiable core $\widehat{\psi}$ of $\widehat{\varphi} \wedge \Gamma$. Then we drop from $\widehat{\psi}$ all the conjuncts which belong either to $\Gamma$ or to the abstraction of $\varphi_{shift}$, and produce the final NTA unsatisfiable core by un-abstracting the result, rewriting back each $f_*$, $f_{\text{TF}}$ and $\widehat{\pi}$ into $*$, TF and $\pi$ respectively. The conjuncts in $\bigwedge \Gamma$ and in the abstraction of $\varphi_{shift}$ are safely ignored because they would respectively produce NTA-valid subformulae and simple definitions of variables which do not occur in $\varphi$. (See §4.)

### 3.2 Spuriousness Check and Abstraction Refinement

The process of checking for spuriousness and refining the abstraction is carried out by the procedure CHECK-REFINE (reported in Fig. 3). CHECK-REFINE first calls the function CHECK-MODEL on the abstracted formula $\widehat{\varphi}$, the abstract model $\widehat{\mu}$ and the value $\epsilon$, which tries to determine whether $\widehat{\mu}$ does indeed imply the existence of a model for $\varphi$ (lines 1-2). (CHECK-MODEL is described in §5). The check

```
⟨bool, constraint-set⟩ CHECK-REFINE (φ̂, μ̂, ϵ):
1.  if CHECK-MODEL (φ̂, μ̂, ϵ):
2.      return ⟨true, ∅⟩
3.  Γ := BLOCK-SPURIOUS-PRODUCT-TERMS(φ̂, μ̂)          # refinement of products
4.  while true:  # refinement of transcendental functions, for progressively improving precision
5.      Γ := Γ ∪ BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS(φ̂, μ̂, ϵ)
6.      if Γ ≠ ∅:
7.          return ⟨false, Γ⟩
8.      ϵ := IMPROVE-PRECISION (ϵ)
9.      if CHECK-MODEL (φ̂, μ̂, ϵ):
10.         return ⟨true, ∅⟩
```

Fig. 3. The main procedure for spuriousness check and refinement.

is formulated as a SMT(UFLRA) search problem over a constrained version of $\varphi$, guided by the current abstract model and the current precision, either yielding a sufficient criterion for concluding the existence of a model for $\varphi$, returning **true**, or stating that $\widehat{\mu}$ is spurious, returning **false**. If CHECK-MODEL succeeds, then CHECK-REFINE returns $\langle$**true**, $\emptyset\rangle$, and the whole process terminates. Otherwise $\widehat{\mu}$ is spurious –because it violates some multiplications, or some transcendental functions, or both– and CHECK-MODEL could not prove the existence of another model within the current precision. (Nevertheless, one such model could exist, and might be found using a better precision.)

The rest of the procedure tries to refine the spurious model $\widehat{\mu}$ by adding UFLRA refinement constraints that rule out $\widehat{\mu}$ (and other spurious solutions), which are collected into the set $\Gamma$, interleaving this process with calls to CHECK-MODEL with increasingly improved precision. This is performed in two steps.

The first step is the *refinement of products* (line 3). BLOCK-SPURIOUS-PRODUCT-TERMS is invoked on $\widehat{\varphi}$ and $\widehat{\mu}$ and looks for UFLRA constraints on multiplication terms in the form $f_*(x, y)$ occurring in $\widehat{\varphi}$ which are violated by $\widehat{\mu}$. These constraints are stored in $\Gamma$. Importantly, this process does not depend on the precision $\epsilon$. (BLOCK-SPURIOUS-PRODUCT-TERMS is described in §4.)

The second step is the *refinement of transcendental functions*, which is performed for progressively-improving precision (lines 4-10). At each iteration, first BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS is invoked on $\widehat{\varphi}$, $\widehat{\mu}$ and $\epsilon$, and looks for UFLRA constraints on transcendental terms in the form $f_{\mathrm{TF}}(x)$ occurring in $\widehat{\varphi}$ which are violated by $\widehat{\mu}$. These constraints (if any) are added to $\Gamma$. (BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS is described in §4.) Then, if $\Gamma$ contains at least one refinement constraint ruling out $\widehat{\mu}$, then $\langle$**false**, $\Gamma\rangle$ is returned. If not so, then no result in either direction was obtained with the current precision. Then, the current precision is increased (in the current implementation, we simply reduce $\epsilon$ by one order of magnitude), and CHECK-MODEL is invoked again with the improved precision, returning $\langle$**true**, $\emptyset\rangle$ if it succeeds, like in lines 1-2. The whole process in lines 5-10 is iterated until either $\varphi$ is found satisfiable, or some refinement constraint is produced (or the process is terminated due to resource-budget exhaustion).

Remark 1. It is important to notice that Fig. 3 describes the strategy which is currently implemented, which is only one of the many alternative strategies by which refinement can be performed. E.g., the calls to CHECK-MODEL, BLOCK-SPURIOUS-PRODUCT-TERMS and BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS are not bound to be executed necessarily in this sequence, and can be interleaved in different ways. For instance, one could adopt the strategy to call BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS only if $\Gamma = \emptyset$, so that CHECK-REFINE will be repeatedly called to refine

$$
\begin{aligned}
&\textit{Zero:} && \forall x, y.((x = 0 \vee y = 0) \leftrightarrow f_*(x, y) = 0)\\
&&& \forall x, y.(((x > 0 \wedge y > 0) \vee (x < 0 \wedge y < 0)) \leftrightarrow f_*(x, y) > 0)\\
&&& \forall x, y.(((x < 0 \wedge y > 0) \vee (x > 0 \wedge y < 0)) \leftrightarrow f_*(x, y) < 0)\\
&\textit{Sign:} && \forall x, y. f_*(x, y) = f_*(-x, -y)\\
&&& \forall x, y. f_*(x, y) = -f_*(-x, y)\\
&&& \forall x, y. f_*(x, y) = -f_*(x, -y)\\
&\textit{Commutativity:} && \forall x, y. f_*(x, y) = f_*(y, x)\\
&\textit{Monotonicity:} && \forall x_1, y_1, x_2, y_2.((abs(x_1) \leq abs(x_2) \wedge abs(y_1) \leq abs(y_2)) \rightarrow abs(f_*(x_1, y_1)) \leq abs(f_*(x_2, y_2)))\\
&&& \forall x_1, y_1, x_2, y_2.((abs(x_1) < abs(x_2) \wedge abs(y_1) \leq abs(y_2) \wedge y_2 \neq 0) \rightarrow abs(f_*(x_1, y_1)) < abs(f_*(x_2, y_2)))\\
&&& \forall x_1, y_1, x_2, y_2.((abs(x_1) \leq abs(x_2) \wedge abs(y_1) < abs(y_2) \wedge x_2 \neq 0) \rightarrow abs(f_*(x_1, y_1)) < abs(f_*(x_2, y_2)))\\
&\textit{Tangent plane:} && \forall x, y.(f_*(a, y) = a * y \wedge f_*(x, b) = b * x \wedge\\
&&& (((x > a \wedge y < b) \vee (x < a \wedge y > b)) \rightarrow f_*(x, y) < \textsc{TanPlane}_{*,a,b}(x, y)) \wedge\\
&&& (((x < a \wedge y < b) \vee (x > a \wedge y > b)) \rightarrow f_*(x, y) > \textsc{TanPlane}_{*,a,b}(x, y)))
\end{aligned}
$$

Fig. 4. The refinement UFLRA constraint schemata for multiplication. (We recall that "$abs(x)$" is a shortcut for "ite$(x < 0, -x, x)$" and that "$\textsc{TanPlane}_{*,a,b}(x, y)$" is a shortcut for "$b * x + a * y - a * b$".)

only multiplications (line 3) until a fixpoint is reached, and to refine the transcendental functions only after then. Further information on these issues will be provided in §4 and §8.

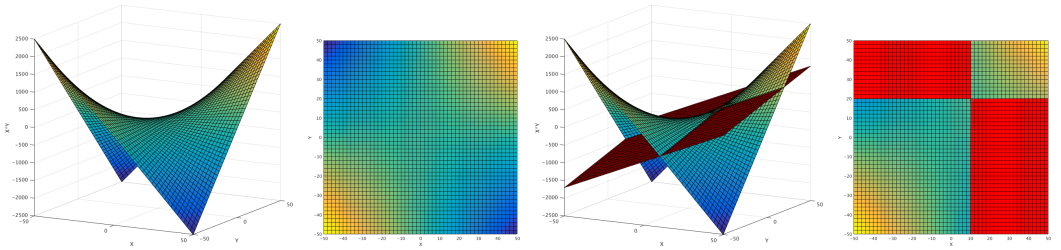
## 4 REFINEMENT

In this section we focus on the refinement part of our procedure. We first describe how to perform the refinement of multiplication terms $f_*(x, y)$ in the function BLOCK-SPURIOUS-PRODUCT-TERMS (§4.1) and then how to perform refinement of transcendental functions in the function BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS (§4.2).

### 4.1 Refinement for NRA

We describe the refinement of multiplication terms in the function BLOCK-SPURIOUS-PRODUCT-TERMS. The refinement is based on selecting suitable instantiations of given constraint schemata which prevent spurious assignments to multiplication terms. We consider the refinement constraint schemata in Fig. 4, where $x, x_i, y, y_i$ are variables and $a, b$ are generic rational values. It is straightforward to verify that all the constraints are valid formulae in any theory interpreting $f_*()$ as $*$. Notice that, the Zero constraints refer to a single multiplication term, that the Sign, Commutativity and Monotonicity constraints refer to pairs of multiplication terms, whereas the Tangent plane constraints refer to a single multiplication term and a single point $(a, b)$. The Zero, Sign, Commutativity and Monotonicity constraints are self-explanatory; the Tangent-plane constraints, instead, deserve some explanations.

The equalities in the Tangent-plane constraints are providing multiplication lines that enforce the correct value of $f_*(x, y)$ when $x = a$ or $y = b$; the inequalities are providing bounds for $f_*(x, y)$ when $x$ and $y$ are not on the multiplication lines. The constraints specialize the notion of tangent plane to the case of nonlinear multiplication. Geometrically, the surface generated by the multiplication function $mul(x, y) \doteq x * y$ is shown in Fig. 5a and 5b. This kind of surface is known in geometry as hyperbolic paraboloid. A hyperbolic paraboloid is a doubly-ruled surface, i.e. for every point on the surface, there are two distinct lines projected from the surface such that they pass through the point. A tangent plane to a hyperbolic paraboloid has the property that the two projected lines from the surface are also in the tangent plane, and they define how the



(a) $x * y$ (b) $x * y$ (top view) (c) $x * y$ and tangent plane (d) $x * y$ and tangent plane (top view)

Fig. 5. Multiplication function and tangent plane.

plane cuts the surface. In case of the multiplication surface, the projected lines basically lie on the surface (see Fig. 5c and 5d). Another way to show the validity of the inequalities is by using the fact that $x * y - \textsc{TanPlane}_{*,a,b}(x, y) = (x - a) * (y - b)$. Consider the following case of the Tangent-plane inequalities (the other cases can be proven in the same way): $(x > a) \wedge (y < b) \rightarrow (x * y < \textsc{TanPlane}_{*,a,b}(x, y))$. Using the previous fact, we can rewrite the right hand-side of the implication as $(x - a) * (y - b) < 0$, which follows immediately from the premises of the implication.

Let $\widehat{\mu}$ be the spurious interpretation which is given as input to BLOCK-SPURIOUS-PRODUCT-TERMS. Let $f_*(x, y)$ be one generic multiplication term occurring in $\widehat{\varphi}$, let $a \doteq \widehat{\mu}[x]$ and $b \doteq \widehat{\mu}[y]$. If $\widehat{\mu}[f_*(x, y)] \neq a*b$, then $f_*(x, y)$ is a *spurious term* in $\widehat{\mu}$. The idea is thus to add refinement constraints to block the spurious terms $f_*(x, y)$ in $\widehat{\varphi}$ so that a new interpretation will be constructed. In what follows, $\mathcal{ST}_*^{\widehat{\mu}}$ denotes the set of multiplication terms in $\widehat{\varphi}$ which are made spurious by $\widehat{\mu}$.

*Single-term Refinements.* If $f_*(x, y)$ is the only multiplication term occurring in a constraint schema $\forall x, y.\psi$ in Fig. 4 (i.e., a Zero constraint schema), and $f_*(t_1, s_1) \in \mathcal{ST}_*^{\widehat{\mu}}$, then we say that $\widehat{\mu}$ *violates* $\forall x, y.\psi$ *on* $f_*(t_1, s_1)$ *wrt.* $f_*(x, y)$ if and only if $\psi\{x, y \mapsto \widehat{\mu}[t_1], \widehat{\mu}[s_1]\}$ is false in any theory interpreting $f_*()$ as $*$. Then, for every term $f_*(t_1, s_1) \in \mathcal{ST}_*^{\widehat{\mu}}$, and for every constraint schema $\forall x, y.\psi$ as above, if $\widehat{\mu}$ violates $\forall x, y.\psi$ on $f_*(t_1, s_1)$ wrt. $f_*(x, y)$, then we can produce the refinement constraint $\psi\{x, y \mapsto t_1, s_1\}$. By construction, $\widehat{\mu} \not\models \psi'$, that is, $\psi'$ rules out $\widehat{\mu}$.

*Double-term Refinements.* Similarly, if $f_*(t, s)$ and $f_*(u, w)$ for some terms $s, t, u, w$ are the only multiplication terms occurring in a constraint schema $\forall \mathbf{x}.\psi$ (i.e. a Sign, Commutativity, or Monotonicity constraint schema), and $f_*(t_1, s_1), f_*(u_1, w_1) \in \mathcal{ST}_*^{\widehat{\mu}}$ s.t. $\langle f_*(t, s), f_*(u, w)\rangle$ can be mapped into $\langle f_*(t_1, s_1), f_*(u_1, w_1)\rangle$ by some variable instantiation $\sigma : \mathbf{x} \mapsto \mathbf{t}$, we say that $\widehat{\mu}$ *violates* $\psi$ *on* $\langle f_*(t_1, s_1), f_*(u_1, w_1)\rangle$ *wrt* $\langle f_*(t, s), f_*(u, w)\rangle$ if and only if $\psi\{\mathbf{x} \mapsto \widehat{\mu}[\mathbf{t}]\}$ is false in any theory interpreting $f_*()$ as $*$. Then, for every pair of terms $\langle f_*(t_1, s_1), f_*(u_1, w_1)\rangle$ and for every constraint schema $\psi$ as above, if $\widehat{\mu}$ violates $\psi$ on $f_*(t_1, s_1)$ wrt $\langle f_*(t, s), f_*(u, w)\rangle$ as above, then we can produce the refinement constraint $\psi' \doteq \psi\{\mathbf{x} \mapsto \mathbf{t}\}$. By construction, $\widehat{\mu} \not\models \psi'$, that is, $\psi'$ rules out $\widehat{\mu}$.

*Tangent-Plane Refinements.* For every term $f_*(t_1, s_1) \in \mathcal{ST}_*^{\widehat{\mu}}$, and for each Tangent-plane constraint schema $\forall x, y.\psi$, we can produce the refinement constraint $\psi' \doteq \psi\{x, y, a, b \mapsto t_1, s_1, \widehat{\mu}[t_1], \widehat{\mu}[s_1]\}$. By construction, $\widehat{\mu} \not\models \psi'$, that is, $\psi'$ rules out $\widehat{\mu}$.

*Example 4.1.* Consider the case where $\varphi$ contains the multiplications $u_1 * w_1$ and $u_2 * w_2$, so that $\widehat{\varphi}$ contains the multiplication terms $f_*(u_1, w_1)$ and $f_*(u_2, w_2)$. Let $\widehat{\mu}$ be a spurious assignment s.t.

$$\widehat{\mu}[u_1] = 2,\ \widehat{\mu}[w_1] = 3,\ \widehat{\mu}[f_*(u_1, w_1)] = 7,\ \widehat{\mu}[u_2] = 3,\ \widehat{\mu}[w_2] = -4,\ \widehat{\mu}[f_*(u_2, w_2)] = 5.$$

| | |
|---|---|
| *Zero:* | $((u_2 < 0 \wedge w_2 > 0) \vee (u_2 > 0 \wedge w_2 < 0)) \leftrightarrow f_*(u_2, w_2) < 0$ |
| *Monotonicity:* | $(abs(u_1) \leq abs(u_2) \wedge abs(w_1) \leq abs(w_2)) \rightarrow abs(f_*(u_1, w_1)) \leq abs(f_*(u_2, w_2))$ |
| | $(abs(u_1) < abs(u_2) \wedge abs(w_1) \leq abs(w_2) \wedge w_2 \neq 0) \rightarrow abs(f_*(u_1, w_1)) < abs(f_*(u_2, w_2))$ |
| | $(abs(u_1) \leq abs(u_2) \wedge abs(w_1) < abs(w_2) \wedge u_2 \neq 0) \rightarrow abs(f_*(u_1, w_1)) < abs(f_*(u_2, w_2))$ |
| *Tangent plane:* | $f_*(2, w_1) = 2 * w_1$ |
| | $f_*(u_1, 3) = 3 * u_1$ |
| | $((u_1 > 2 \wedge w_1 < 3) \vee (u_1 < 2 \wedge w_1 > 3)) \rightarrow f_*(u_1, w_1) < 3 * u_1 + 2 * w_1 - 6$ |
| | $((u_1 < 2 \wedge w_1 < 3) \vee (u_1 > 2 \wedge w_1 > 3)) \rightarrow f_*(u_1, w_1) > 3 * u_1 + 2 * w_1 - 6$ |
| | $f_*(3, w_2) = 3 * w_2$ |
| | $f_*(u_2, -4) = -4 * u_2$ |
| | $((u_2 > 3 \wedge w_2 < -4) \vee (u_2 < 3 \wedge w_2 > -4)) \rightarrow f_*(u_2, w_2) < -4 * u_2 + 3 * w_2 + 12$ |
| | $((u_2 < 3 \wedge w_2 < -4) \vee (u_2 > 3 \wedge w_2 > -4)) \rightarrow f_*(u_2, w_2) > -4 * u_2 + 3 * w_2 + 12$ |
| *Sign:* | $f_*(u_1, w_1) = -f_*(u_1, -w_1)$ |
| *Commutativity:* | $f_*(u_2, w_2) = f_*(w_2, u_2)$ |

Fig. 6. Top: example of instantiation of constraint schemata for the multiplication terms $f_*(u_1, w_1)$ and $f_*(u_2, w_2)$, where $\widehat{\mu}[u_1] = 2$, $\widehat{\mu}[w_1] = 3$, $\widehat{\mu}[f_*(u_1, w_1)] = 7$, $\widehat{\mu}[u_2] = 3$, $\widehat{\mu}[w_2] = -4$, $\widehat{\mu}[f_*(u_2, w_2)] = 5$. Bottom: example of instantiation of Sign and Commutativity if we consider also $f_*(u_1, -w_1)$ and $f_*(w_2, u_2)$.

$\widehat{\mu}$ violates the third Zero constraint on $f_*(u_2, w_2)$, it does not violate any Sign or Commutativity constraint, and it violates the three Monotonicity constraints on $\langle f_*(u_1, w_1), f_*(u_2, w_2)\rangle$. Overall, this leads to the addition of the Zero and Monotonicity constraints, plus the Tangent-plane ones in the points $(2, 3)$ and $(3, -4)$, which are reported at the top of Fig. 6.

If $\varphi$ contains also $f_*(u_1, -w_1)$ and $f_*(w_2, u_2)$, and if $\widehat{\mu}[f_*(u_1, -w_1)] = -3$ and $\widehat{\mu}[f_*(w_2, u_2)] = 9$, then we add also the Sign and Commutativity constraints in the bottom part of Fig. 6 (plus the other Zero, Monotonicity and Tangent-plane constraints).

Remark 2. The above narration describes a very "eager" refinement strategy, in which all possible refinement constraints for all possible spurious multiplication terms are generated. Notice, however, that in order to rule out $\widehat{\mu}$ it is sufficient to produce one single refinement constraint for one single spurious multiplication term. Thus, a great variety of more "lazy" strategies can be adopted, in which only some of the schemata are instantiated, and only for some of the multiplication terms. For example, rather than refining all spurious terms, it might be a good idea to refine only terms occurring in atomic subformulae whose truth-value assignment in $\widehat{\mu}$ actually contributed to satisfy $\widehat{\varphi}$. (E.g., atoms occurring only positively in $\widehat{\varphi}$ which are true in $\widehat{\mu}$, see e.g. [7].)

Also, the instantiation of each constraint schema can be implemented at different levels of granularity, because some constraint schema may correspond to the conjunction of more than one clause (e.g., three clauses in the case of each Zero constraint schema [3]) so that one may decide to instantiate all, some or only one of the clauses that are actually violated by $\widehat{\mu}$.

Notice that, even more eager refinement strategies are possible, e.g., by instantiating the Sign and Commutativity constraint schemata $\forall x, y.\psi$ also when only one multiplication term $f_*(t_1, s_1)$ in $\mathcal{ST}_*^{\widehat{\mu}}$ matches one of the two multiplication terms in $\forall x, y.\psi$, adding a new multiplication term.

We will discuss the strategies which we have actually chosen and implemented in §8.

[3] A formula in the form $(A \vee B) \leftrightarrow C$ can be rewritten as $(A \vee B \vee \neg C) \wedge (\neg A \vee C) \wedge (\neg B \vee C)$.

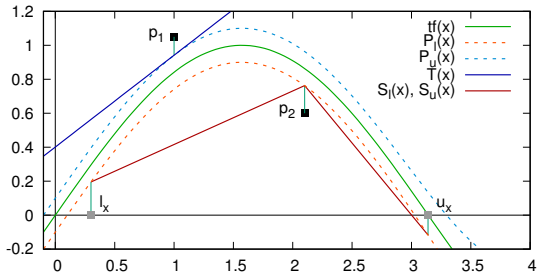



Fig. 7. Piecewise-linear refinement illustration.

constraint-set BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS $(\widehat{\varphi}, \widehat{\mu}, \epsilon)$:

1. $\Gamma := \emptyset$
2. **for all** $f_{\mathrm{TF}}(x) \in \widehat{\varphi}$:
3. $\quad \langle P_l(x), P_u(x) \rangle :=$ GET-POLYNOMIAL-BOUNDS $(f_{\mathrm{TF}}(x), \widehat{\mu}, \epsilon)$
4. $\quad$ **if** $\widehat{\mu}[f_{\mathrm{TF}}(x)] \notin [P_l(\widehat{\mu}[x]), P_u(\widehat{\mu}[x])]$:
5. $\quad\quad \Gamma := \Gamma \cup$ BLOCK-SPURIOUS-NTA-TERM$(f\mathrm{TF}(x), \widehat{\mu}, P_l(x), P_u(x))$
6. **return** $\Gamma$

Fig. 8. Refinement of transcendental functions.

$\langle$**polynomial, polynomial**$\rangle$ GET-POLYNOMIAL-BOUNDS $(f_{\mathrm{TF}}(x), \widehat{\mu}, \epsilon)$:

1. $c := \widehat{\mu}[x]$
2. $\mathrm{conc}_{\mathrm{TF}} :=$ GET-CONCAVITY $(\mathrm{TF}, c)$
3. $i := 1$
4. **while true**:
5. $\quad \langle P_l(x), P_u(x) \rangle :=$ MACLAURIN-APPROX $(\mathrm{TF}, i, x, c)$
6. $\quad \delta := P_u(c) - P_l(c)$
7. $\quad \mathrm{conc}_l :=$ GET-CONCAVITY $(P_l, c)$
8. $\quad \mathrm{conc}_u :=$ GET-CONCAVITY $(P_u, c)$
9. $\quad$ **if** $\mathrm{conc}_l = \mathrm{conc}_u = \mathrm{conc}_{\mathrm{TF}}$ **and** $\delta \leq \epsilon$:
10. $\quad\quad$ **return** $\langle P_l(x), P_u(x) \rangle$
11. $\quad$ **else**:
12. $\quad\quad i := i + 1$

Fig. 9. Polynomial bounds computation for transcendental functions.

### 4.2 Refinement for NTA

We consider now the problem of eliminating spurious assignments to transcendental functions. The pseudo-code for BLOCK-SPURIOUS-TRANSCENDENTAL-TERMS is shown in Fig. 8. We iterate on all the (abstract) transcendental function applications $f_{\mathrm{TF}}(x)$ in $\widehat{\varphi}$, in order to check whether the SMT(UFLRA)-model $\widehat{\mu}$ is consistent with the NTA semantics. In principle, this amounts to checking if $\widehat{\mu}[f_{\mathrm{TF}}(x)] = \mathrm{TF}(\widehat{\mu}[x])$. In practice, the check cannot be implemented, since transcendental functions at rational points most often have irrational values (see e.g. [57]), which cannot be represented in SMT(UFLRA).

Therefore, for each term $\text{TF}(x)$ in $\varphi$, we instead compute two rational values, namely $\text{QL}$ and $\text{QU}$, with the property that $\text{QL} \leq \text{TF}(\widehat{\mu}[x]) \leq \text{QU}$. The computation of $\text{QL}$ and $\text{QU}$ is based on polynomials computed using Taylor series, according to the given current precision, by the function GET-POLYNOMIAL-BOUNDS of Fig. 9. This is done by expanding the Maclaurin series of TF, generating polynomials for the upper and lower bounds according to Taylor's theorem (see §2.3), until the requested precision $\epsilon$ is met. The two values $\text{QL}$ and $\text{QU}$ are simply the results of evaluating the two computed polynomials $P_l(x)$ and $P_u(x)$ at $\widehat{\mu}[x]$. As an additional requirement that will be explained below, the function also ensures (lines 7–9) that the concavity of the Taylor polynomials is the same as that of TF at $\widehat{\mu}[x]$.

If the value of $\text{TF}(x)$ in $\widehat{\mu}$ is not included in the interval $[\text{QL}, \text{QU}]$, the function BLOCK-SPURIOUS-NTA-TERM of Fig. 10 is used to generate (piecewise) linear constraints that remove the point $(\widehat{\mu}[x], \widehat{\mu}[\text{TF}(x)])$ (and possibly many others) from the graph of $f_{\text{TF}}$, thus refining the abstraction.

The refinement is performed by BLOCK-SPURIOUS-NTA-TERM of Fig. 10 in two steps. First, it attempts to exclude the bad point by invoking BLOCK-SPURIOUS-NTA-BASIC, which instantiates some *basic constraint schemata* describing very general properties of the transcendental function TF under consideration (lines 1-3). These constraints encode some simple properties of transcendental functions (such as sign and monotonicity conditions, or bounds at noteworthy values) via linear relations, and are described in §4.2.1 and §4.2.2. If the current abstract model $\widehat{\mu}$ violates any of the basic constraints, BLOCK-SPURIOUS-NTA-TERM simply returns their corresponding instantiations.

If none of the basic constraints is violated, then two situations are possible, as illustrated in Fig. 7. Let the green line be the graph of some transcendental function TF. The points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, say (1.0,1.0) and (2.2, 0.5), represent transcendental terms that are spurious in the current assignment $\widehat{\mu}$ —that is, $p_1 = (\widehat{\mu}[x_1], \widehat{\mu}[f_{\text{TF}}(x_1)])$ and $p_2 = (\widehat{\mu}[x_2], \widehat{\mu}[f_{\text{TF}}(x_2)])$, for some $x_1, x_2$. In order to eliminate them, we need to discover linear constraints that are guaranteed to safely approximate TF. Clearly, a major role is played by the position of the spurious value $\widehat{\mu}[f_{\text{TF}}(x)]$ relative to the correct value $\text{TF}(\widehat{\mu}[x])$, and by the concavity of TF around the point $\widehat{\mu}[x]$. If the concavity is negative or equal to zero, and the point lies above the function, then the tangent to the function would be adequate to block the spurious assignment –and tighten the approximation of TF. (This is the case for $p_1$.) However, if the concavity is negative but the point lies below the function, then a tangent would be not adequate. (This is the case for $p_2$.) For this reason, secants are required, which are unfortunately not unique.

The main problem, however, is that the coefficients of the tangent or of a secant to a transcendental function for a rational value are likely to be irrational, which means that the constraints of the tangent/secant line cannot be readily dealt with by an SMT(UFLRA) solver. As shown in Fig. 7, the idea is to rely on polynomials to approximate the transcendental function, making sure that they also agree on the concavity with the transcendental function. In this way, the polynomial $P_u$ approximating TF from above, depicted as a dashed blue line, has tangent that is guaranteed to approximate TF from above (see Proposition 2.5 in §2.3). Similarly, for the (red dashed) polynomial $P_l$ approximating TF from below, any piecewise combination of secants is guaranteed to approximate TF from below. The key property of polynomials is that the coefficients for tangents and secants are guaranteed to be rationals, and thus amenable to LRA reasoning. These polynomials are computed using the Maclaurin series of the corresponding transcendental function and Taylor's theorem. Notice that, we use the Maclaurin series (i.e. the Taylor series centered around 0) because we can always compute the exact derivative of any order at 0 for the transcendental functions we support, namely the exponential (exp) and the sine (sin) function. In fact, $\exp(0) = 1$, $\sin(0) = 0$, $\exp^{(i)}(x) = \exp(x)$ for all $i$, and $|\sin^{(i)}(x)|$ is $|\cos(x)|$ if $i$ is odd and $|\sin(x)|$ otherwise. Thus, the computation of the Maclaurin series and of the remainder polynomial is exact.

constraint-set BLOCK-SPURIOUS-NTA-TERM ($\mathrm{TF}(x)$, $\widehat{\mu}$, $P_l(x)$, $P_u(x)$):
*# basic refinement*
1. $\Gamma$ := BLOCK-SPURIOUS-NTA-BASIC ($\mathrm{TF}(x)$, $\widehat{\mu}$)
2. **if** $\Gamma \neq \emptyset$:
3. **return** $\Gamma$
*# general refinement*
4. $c := \widehat{\mu}[x]$
5. $v := \widehat{\mu}[f_{\mathrm{TF}}(x)]$
6. conc := GET-CONCAVITY ($\mathrm{TF}(x)$, $c$)
7. **if** $(v \leq P_l(c)$ **and** conc $\geq 0)$ **or** $(v \geq P_u(c)$ **and** conc $\leq 0)$:
*# tangent refinement*
8. $P := (v \leq P_l(c))\ ?\ (P_l)\ :\ (P_u)$
9. $T(x)$ := $\mathrm{TANLINE}_{P,c}(x)$ *# tangent of $P$ at $c$*
10. $\langle l, u\rangle$ := GET-TANGENT-BOUNDS ($\mathrm{TF}(x)$, $c$, $\frac{d}{dx}P(c)$)
11. $\psi := (\text{conc} < 0)\ ?\ (f_{\mathrm{TF}}(x) \leq T(x))\ :\ (f_{\mathrm{TF}}(x) \geq T(x))$
12. $\Gamma := \{((x \geq l) \wedge (x \leq u)) \rightarrow \psi\}$
13. **else**: *# $(v \leq P_l(c) \wedge conc < 0) \vee (v \geq P_u(c) \wedge conc > 0)$*
*# secant refinement*
14. prev := GET-PREVIOUS-SECANT-POINTS ($\mathrm{TF}(x)$)
15. $l := \max\{p \in \text{prev} \mid p < c\}$
16. $u := \min\{p \in \text{prev} \mid p > c\}$
17. $P := (v \leq P_l(c))\ ?\ (P_l)\ :\ (P_u)$
18. $S_l(x)$ := $\mathrm{SECLINE}_{P,l,c}(x)$ *# secant of $P$ between $l$ and $c$*
19. $S_u(x)$ := $\mathrm{SECLINE}_{P,c,u}(x)$
20. $\psi_l := (\text{conc} < 0)\ ?\ (f_{\mathrm{TF}}(x) \geq S_l(x))\ :\ (f_{\mathrm{TF}}(x) \leq S_l(x))$
21. $\psi_u := (\text{conc} < 0)\ ?\ (f_{\mathrm{TF}}(x) \geq S_u(x))\ :\ (f_{\mathrm{TF}}(x) \leq S_u(x))$
22. $\phi_l := (x \geq l) \wedge (x \leq c)$
23. $\phi_u := (x \geq c) \wedge (x \leq u)$
24. STORE-SECANT-POINT ($\mathrm{TF}(x)$, $c$)
25. $\Gamma := \{(\phi_l \rightarrow \psi_l), (\phi_u \rightarrow \psi_u)\}$
26. **return** $\Gamma$

Fig. 10. Piecewise-linear refinement for the transcendental function $\mathrm{TF}(x)$ at point $c$.

The rest of the primitive BLOCK-SPURIOUS-NTA-TERM (lines 4-26), which blocks spurious transcendental terms, is based on the above considerations. If the concavity is positive (resp. negative) or equal to zero, and the point lies below (resp. above) the function, then the linear approximation is given by a tangent to the lower (resp. upper) bound polynomial $P_l$ (resp. $P_u$) at $\widehat{\mu}[x]$ (lines 7–12 of Fig. 10); otherwise, i.e. the concavity is negative (resp. positive) and the point is below (resp. above) the function, the linear approximation is given by a pair of secants to the lower (resp. upper) bound polynomial $P_l$ (resp. $P_u$) around $\widehat{\mu}[x]$ (lines 13–25 of Fig. 10) – an example is shown in Fig. 7.

In the case of tangent refinement, the function GET-TANGENT-BOUNDS (line 10) returns an interval $[l, u]$ inside which the tangent line is guaranteed not to cross the transcendental function TF. In practice, this interval can be (under)approximated quickly by exploiting known properties of the specific function TF under consideration. For example, for the exponential function GET-TANGENT-BOUNDS always returns $[-\infty, +\infty]$; for other functions, the computation can be based e.g. on an

$$
\begin{aligned}
\textit{Lower Bound:}\quad & \forall y.(f_{\exp}(y) > 0)\\
\textit{Zero:}\quad & \forall y.(y = 0 \leftrightarrow f_{\exp}(y) = 1)\\
& \forall y.(y < 0 \leftrightarrow f_{\exp}(y) < 1)\\
& \forall y.(y > 0 \leftrightarrow f_{\exp}(y) > 1)\\
\textit{Zero Tangent Line:}\quad & \forall y.(y \neq 0 \leftrightarrow f_{\exp}(y) > y + 1)\\
\textit{Monotonicity:}\quad & \forall y_1, y_2.(y_1 < y_2 \leftrightarrow f_{\exp}(y_1) < f_{\exp}(y_2))
\end{aligned}
$$

Fig. 11. Basic constraint schemata for the exponential function.

analysis of the (known, precomputed) inflection points of TF around the point of interest $\widehat{\mu}[x]$ and the slope $\frac{d}{dx}P(c)$ of the tangent line.

In the case of secant refinement, a second value, different from $\widehat{\mu}[x]$, is required to draw a secant line. The function GET-PREVIOUS-SECANT-POINTS returns the set of all the points at which a secant refinement was performed in the past for $\text{TF}(x)$. From this set, we take the two points $l$ and $u$ closest to $\widehat{\mu}[x]$, such that $l < \widehat{\mu}[x] < u$ and that there is no inflection point in $[l, u]$ [4], and use those points to generate two secant lines and their validity intervals. Before returning the set of the two corresponding constraints, we also store the new secant refinement point $\widehat{\mu}[x]$ by calling STORE-SECANT-POINT.

Remark 3. Note that we have some freedom in choosing the extra points for secant refinement, as long as they lie within the interval in which the concavity doesn't change; our reuse of previous secant points is therefore simply a heuristic, based on the intuition that the points are relevant (because they have been generated in previous refinements); furthermore, by reusing previous points we avoid introducing too many new literals, which might pollute the search space of the solver.

Remark 4. Similarly to the case of NRA (see Remark 2), we remark that also the above description is only one of the possible strategies for refinement, and that in particular it is possible to adopt lazier variants.

*Example 4.2.* In order to rule out a spurious interpretation $\widehat{\mu}[x] = 2.0, \widehat{\mu}[f_{\exp}(x)] = 3.0$ (where $f_{\exp}(x)$ is the abstraction of the exponential function) we may exploit the positive concavity of $\exp(x)$ and obtain a linear lower-bound constraint, e.g. $f_{\exp}(x) > \frac{155}{21} + \frac{331}{45} * (x - 2)$. Notice that, $\exp(2.0) \cong 7.389$, $\frac{155}{21} \cong 7.381 \lessapprox \exp(2.0)$, and $\frac{331}{45} \cong 7,356 \lessapprox \frac{d}{dx}exp(2.0)$. These values are such that the above linear constraint "approximates" the tangent of $\exp(x)$ in $x = 2$, since it always lower-bounds $exp(x)$ and its value and derivative are very near to those of $exp(x)$ for $x = 2.0$.

In the following, we discuss our approach for generating refinement constraints for the transcendental functions exp and sin. Other transcendental functions such as log, cos, tan, arcsin, arccos, arctan can be handled by means of rewriting. For example, $\cos(x)$ is rewritten to $\sin(x + \frac{\pi}{2})$, whereas if $\varphi$ contains $\log(x)$, we rewrite it as $\varphi\{\log(x) \mapsto l_x\} \wedge \exp(l_x) = x$, where $l_x$ is a fresh variable.

#### 4.2.1 *The Exponential Function.*

*Basic linear constraints.* Our implementation of BLOCK-SPURIOUS-NTA-BASIC for exp uses the linear constraint schemata in Fig. 11. For each exponential function $\exp(x)$ violating its rational

[4]For simplicity, we assume that this is always possible. If needed, this can be implemented e.g. by generating the two points at random while ensuring that $l < \widehat{\mu}[x] < u$ and that there is no inflection point in $[l, u]$.

bounds, we instantiate the basic constraint schemata with $\widehat{\mu}[x]$; if the result evaluates to false, we generate the corresponding instantiation by replacing the quantified variable $y$ with $x$ and removing the quantifier. In the case of the monotonicity constraint schema, we check all possible combinations of exponential function applications $\exp(x_1)$ and $\exp(x_2)$ that are violating their rational bounds.

*Polynomial approximation.* Since $\frac{d}{dx}\exp(x) = \exp(x)$, all the derivatives of exp are positive. The polynomial $P_{n,\exp(0)}(x)$ is given by the Maclaurin series

$$P_{n,\exp(0)}(x) = \sum_{i=0}^{n} \frac{x^i}{i!}$$

and behaves differently depending on the sign of $x$. Thus, GET-POLYNOMIAL-BOUNDS distinguishes three cases for finding the polynomials $P_l(x)$ and $P_u(x)$:

**Case $x = 0$:** since $\exp(0) = 1$, we have $P_l(0) = P_u(0) = 1$;

**Case $x < 0$:** we have that $P_{n,\exp(0)}(x) < \exp(x)$ if $n$ is odd, and $P_{n,\exp(0)}(x) > \exp(x)$ if $n$ is even; we therefore set $P_l(x) = P_{n,\exp(0)}(x)$ and $P_u(x) = P_{n+1,\exp(0)}(x)$ for a suitable $n$ so that the required precision $\epsilon$ is met;

**Case $x > 0$:** we have that $P_{n,\exp(0)}(x) < \exp(x)$ and $P_{n,\exp(0)}(x) * (1 - \frac{x^{n+1}}{(n+1)!})^{-1} > \exp(x)$ when $(1 - \frac{x^{n+1}}{(n+1)!}) > 0$, therefore we set $P_l(x) = P_{n,\exp(0)}(x)$ and $P_u(x) = P_{n,\exp(0)}(x) * (1 - \frac{x^{n+1}}{(n+1)!})^{-1}$ for a suitable $n$. [5] This upper bound is derived using Taylor's theorem: We can write exp as a sum of the Taylor series centered at zero and Lagrange remainder.

$$\exp(x) = P_{n,\exp(0)}(x) + \exp(c) * \frac{x^{n+1}}{(n+1)!}$$

When $x > 0$ then $\exp(c) \leq \exp(x)$, therefore the above equality can be written as:

$$\exp(x) \leq P_{n,\exp(0)}(x) + \exp(x) * \frac{x^{n+1}}{(n+1)!}$$

$$\exp(x) - \exp(x) * \frac{x^{n+1}}{(n+1)!} \leq P_{n,\exp(0)}(x)$$

$$\exp(x) * (1 - \frac{x^{n+1}}{(n+1)!}) \leq P_{n,\exp(0)}(x)$$

We can obtain an upper bound for $\exp(x)$ for the case when $(1 - \frac{x^{n+1}}{(n+1)!}) > 0$, that is

$$\exp(x) \leq P_{n,\exp(0)}(x) * (1 - \frac{x^{n+1}}{(n+1)!})^{-1}$$

Since the concavity of exp is always positive, the tangent refinement will always give lower bounds for $\exp(x)$, and the secant refinement will give upper bounds. Moreover, as exp has no inflection points, GET-TANGENT-BOUNDS always returns $[-\infty, +\infty]$.

#### 4.2.2 *The Sine Function.*

[5] We slightly abuse the notation: $P_u(x)$ is not a polynomial but a rational function.

*Dealing with periodicity: base period shifting.* The correctness of our refinement procedure relies crucially on being able to compute the concavity of the transcendental function TF at a given point $c$. This is needed in order to know whether a computed tangent or secant line constitutes a valid upper or lower bound for TF around $c$ (see Fig. 7 and 10). In the case of the sin function, computing the concavity at an arbitrary point $c$ is problematic, since this essentially amounts to computing the value $c' \in [-\pi, \pi[$ s.t. $c = 2\pi n + c'$ for some integer $n$, because in $[-\pi, \pi[$ the concavity of $\sin(c')$ is the opposite of the sign of $c'$. This is not easy to compute because $\pi$ is a transcendental number.

In order to solve this problem, we exploit another property of sin, namely its periodicity (with period $2\pi$). More precisely, we split the reasoning about sin depending on two kinds of periods: base period, with argument from $-\pi$ to $\pi$, and extended period. For each $\sin(x)$ term we introduce an "artificial" sin term $\sin(\omega_x)$, where $\omega_x$ is a fresh variable called *base variable*. Base variables are constrained to be interpreted over the base period, where the sin value for the corresponding variable in the extended period is computed. This is done by adding the following constraint during formula preprocessing:

$$\varphi_{shift} \doteq \bigwedge_{\sin(x)\in\varphi} (-\pi \leq \omega_x < \pi \ \wedge \ ((-\pi \leq x < \pi) \rightarrow x = \omega_x) \ \wedge \ \sin(x) = \sin(\omega_x)) \qquad (1)$$

The first conjunct constrains $\omega_x$ to the base period. The second conjunct [6] states that if $x$ is interpreted in the base period then it has the same value as its base variable. The third conjunct constrains $\sin(x)$ to have the same value as $\sin(\omega_x)$. In order to reason about the irrational $\pi$, we introduce a variable $\widehat{\pi}$, and add the constraint $l_\pi < \widehat{\pi} < u_\pi$ to $\varphi$. $l_\pi$ and $u_\pi$ are valid rational lower and upper bounds for the actual value of $\pi$ that can be computed with various methods. Using this transformation, we can easily compute the concavity of sin at $\widehat{\mu}[\omega_x]$ by just looking at the sign of $\widehat{\mu}[\omega_x]$, provided that $-l_\pi \leq \widehat{\mu}[\omega_x] \leq l_\pi$, where $l_\pi$ is the current lower bound for $\widehat{\pi}$. (We recall that in the interval $[-\pi, \pi[$, the concavity of $\sin(c)$ is the opposite of the sign of $c$.)

Let $\mathcal{F}^{\mathrm{B}}_{\sin}$ be the set of $f_{\sin}(\omega_x)$ terms in $\widehat{\varphi}$ that have base variables as arguments, $\mathcal{F}_{\sin}$ be the set of all $f_{\sin}(x)$ terms, and $\mathcal{F}^{\mathrm{E}}_{\sin} \doteq \mathcal{F}_{\sin} \setminus \mathcal{F}^{\mathrm{B}}_{\sin}$, where $f_{\sin}$ is the uninterpreted function that represents sin in $\widehat{\varphi}$. Both the basic linear refinement and the tangent/secant refinement is performed for the terms in $\mathcal{F}^{\mathrm{B}}_{\sin}$ only; we then use *linear shift* constraints (described below) for refining terms in $\mathcal{F}^{\mathrm{E}}_{\sin}$, as follows.

For each $f_{\sin}(x) \in \mathcal{F}^{\mathrm{E}}_{\sin}$ with the corresponding base variable $\omega_x$, we check whether the value $\widehat{\mu}[x]$ after shifting to the base period is equal to the value of $\widehat{\mu}[\omega_x]$. We calculate the integer shift value $s$ of $x$ as the rounding towards zero of $(\widehat{\mu}[x] + \widehat{\mu}[\widehat{\pi}])/(2 * \widehat{\mu}[\widehat{\pi}])$, and we then compare $\widehat{\mu}[\omega_x]$ with $\widehat{\mu}[x] - 2 * s * \widehat{\mu}[\widehat{\pi}]$. If the values are different, we add the following *linear shift* constraint for relating $x$ with $\omega_x$ in the extended period $s$:

$$(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \wedge f_{\sin}(x) = f_{\sin}(\omega_x) \wedge \widehat{\pi} * (2 * s - 1) \leq x < \widehat{\pi} * (2 * s + 1)) \rightarrow \omega_x = x - 2 * s * \widehat{\pi}.$$

In this way, we do not need the tangent and secant refinement for the extended period and we can reuse the refinements done in the base period. Notice that, even if the calculated shift value is wrong (due to the imprecision of $\widehat{\mu}[\widehat{\pi}]$ with respect to the real value $\pi$), the constraint we generate may be useless, but it is never wrong.

*Basic linear constraints.* We use the constraint schemata of Fig. 12 for implementing BLOCK-SPURIOUS-NTA-BASIC for sin. As written above, these constraints are only checked for terms in $\mathcal{F}^{\mathrm{B}}_{\sin}$.

[6] Notice that we can skip the addition of this constraint, which could be added during the refinement as a linear shift constraint. However, we chose to add it eagerly because it is a very common case.

$$
\begin{aligned}
\textit{Symmetry:}\ & \forall \omega_x.(f_{\sin}(\omega_x) = -f_{\sin}(-\omega_x)) \\
\textit{Phase:}\ & \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (0 < \omega_x \leftrightarrow f_{\sin}(\omega_x) > 0)) \\
& \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (-\widehat{\pi} < \omega_x < 0 \leftrightarrow f_{\sin}(\omega_x) < 0)) \\
\textit{Zero Tangent:}\ & \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (\omega_x > 0 \leftrightarrow f_{\sin}(\omega_x) < \omega_x)) \\
& \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (\omega_x < 0 \leftrightarrow f_{\sin}(\omega_x) > \omega_x)) \\
\pi\ \textit{Tangent:}\ & \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (f_{\sin}(\omega_x) < -\omega_x + \widehat{\pi})) \\
& \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (\omega_x > -\widehat{\pi} \leftrightarrow f_{\sin}(\omega_x) > -\omega_x - \widehat{\pi})) \\
\textit{Significant Values:}\ & \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (f_{\sin}(\omega_x) = 0 \leftrightarrow (\omega_x = 0 \vee \omega_x = -\widehat{\pi}))) \\
& \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (f_{\sin}(\omega_x) = 1 \leftrightarrow \omega_x = \frac{\widehat{\pi}}{2})) \\
& \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (f_{\sin}(\omega_x) = -1 \leftrightarrow \omega_x = -\frac{\widehat{\pi}}{2})) \\
& \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (f_{\sin}(\omega_x) = \frac{1}{2} \leftrightarrow (\omega_x = \frac{\widehat{\pi}}{6} \vee \omega_x = \frac{5 * \widehat{\pi}}{6}))) \\
& \forall \omega_x.(-\widehat{\pi} \leq \omega_x < \widehat{\pi} \rightarrow (f_{\sin}(\omega_x) = -\frac{1}{2} \leftrightarrow (\omega_x = -\frac{\widehat{\pi}}{6} \vee \omega_x = -\frac{5 * \widehat{\pi}}{6}))) \\
\textit{Monotonicity:}\ & \forall \omega_{x_1}, \omega_{x_2}.(-\widehat{\pi} \leq \omega_{x_1} < \omega_{x_2} \leq -\frac{\widehat{\pi}}{2} \rightarrow f_{\sin}(\omega_{x_1}) > f_{\sin}(\omega_{x_2})) \\
& \forall \omega_{x_1}, \omega_{x_2}.(-\frac{\widehat{\pi}}{2} \leq \omega_{x_1} < \omega_{x_2} \leq \frac{\widehat{\pi}}{2} \rightarrow f_{\sin}(\omega_{x_1}) < f_{\sin}(\omega_{x_2})) \\
& \forall \omega_{x_1}, \omega_{x_2}.(\frac{\widehat{\pi}}{2} \leq \omega_{x_1} < \omega_{x_2} < \widehat{\pi} \rightarrow f_{\sin}(\omega_{x_1}) > f_{\sin}(\omega_{x_2}))
\end{aligned}
$$

Fig. 12. Basic constraint schemata for sin function.

*Polynomial approximation.* For each term $f_{\sin}(\omega_x)$ that needs to be refined, we first check whether $\widehat{\mu}[\omega_x] \in [-l_\pi, l_\pi]$, where $l_\pi$ is the current lower bound for $\widehat{\pi}$. If this is the case, then we derive the concavity of sin at $\widehat{\mu}[\omega_x]$ by just looking at the sign of $\widehat{\mu}[\omega_x]$. We can therefore perform tangent or secant refinement as shown in Fig. 10. More precisely, GET-POLYNOMIAL-BOUNDS finds the lower and upper polynomials using Taylor's theorem, which ensures that:

$$P_{n,\sin(0)}(\omega_x) - R^{\mathrm{U}}_{n+1,\sin(0)}(\omega_x) \leq \sin(\omega_x) \leq P_{n,\sin(0)}(\omega_x) + R^{\mathrm{U}}_{n+1,\sin(0)}(\omega_x)$$

where $P_{n,\sin(0)}(\omega_x) = \sum_{k=0}^{n} \frac{(-1)^k * \omega_x^{2k+1}}{(2k+1)!}$ and $R^{\mathrm{U}}_{n+1,\sin(0)}(\omega_x) = \frac{\omega_x^{2(n+1)}}{(2(n+1))!}$. We set $P_l(x) = P_{n,\sin(0)}(x) - R^{\mathrm{U}}_{n+1,\sin(0)}(x)$ and $P_u(x) = P_{n,\sin(0)}(x) + R^{\mathrm{U}}_{n+1,\sin(0)}(x)$. Under the above hypothesis that $\widehat{\mu}[\omega_x] \in [-l_\pi, l_\pi]$, also the function GET-TANGENT-BOUNDS can easily be implemented by looking at the sign of $\widehat{\mu}[\omega_x]$: if $\widehat{\mu}[\omega_x] \geq 0$, then the validity interval is $[0, \widehat{\pi}[$, otherwise, it is $[-\widehat{\pi}, 0]$.

The remaining case to discuss is when the value of $\omega_x$ in $\widehat{\mu}$ is not within the interval $[-l_\pi, l_\pi]$ (which means that $|\widehat{\mu}[\omega_x]| \in (l_\pi, u_\pi]$). In this case, we cannot reliably compute the concavity of sin at $\widehat{\mu}[\omega_x]$. Therefore, instead of performing a tangent/secant refinement, we refine the precision of $\widehat{\pi}$ by computing a tighter interval $(l'_\pi, u'_\pi)$ for it, using Machin's formula [9]. [7]

[7] This is not explicitly shown in the pseudocode of Fig. 10, but it is part of BLOCK-SPURIOUS-NTA-BASIC.

⟨**bool**, model⟩ CHECK-NRA-MODEL ($\widehat{\varphi}$, $\widehat{\mu}$):
1. $\widehat{\psi}$ := GET-ASSIGNMENT($\widehat{\mu}$) *# truth assignment induced by $\widehat{\mu}$ on the atoms of $\widehat{\varphi}$ (2)*
2. $\widehat{\psi}^*$ := $\widehat{\psi}$ ∧ LINEARIZATION-AXIOMS($\widehat{\psi}$) *# add multiplication-line constraints (3) to $\widehat{\psi}$*
3. **return** SMT-UFLRA-CHECK ($\widehat{\psi}^*$)

Fig. 13. An incomplete procedure using an SMT(UFLRA) solver.

## 5 SPURIOUSNESS CHECK AND DETECTING SATISFIABILITY

We concentrate now on the problems of checking the spuriousness of the abstract model $\widehat{\mu}$ and of detecting the existence of solutions for satisfiable formulae. We deal with NRA first in §5.1, and then consider NTA in §5.2.

### 5.1 Finding Rational Models for NRA.

We first describe the behaviour of the function CHECK-MODEL restricted to NRA, which we represent by the function CHECK-NRA-MODEL. Notice that, the parameter $\epsilon$ has no role for NRA, so that CHECK-NRA-MODEL receives as input only $\widehat{\varphi}$ and $\widehat{\mu}$.

In its simplest form, the function CHECK-NRA-MODEL could simply be implemented by checking if $\widehat{\mu}[x] * \widehat{\mu}[y] = \widehat{\mu}[f_*(x,y)]$ for every multiplication term $f_*(x,y) \in \widehat{\varphi}$, returning **true** if and only if this is the case. It is easy to see, however, that this very simple algorithm can return **true** only if the UFLRA solver "guesses" a model that is consistent with all the nonlinear multiplications. In an infinite and dense domain like the rationals or the reals, the chances that this will happen are close to zero in general.

Thus, in practice it is not enough for CHECK-NRA-MODEL to check the spuriousness of $\widehat{\mu}$. In order to detect satisfiable cases more effectively in the very-likely case in which $\widehat{\mu}$ is spurious, we also want that CHECK-NRA-MODEL searches for the existence of an actual model for $\varphi$ "in the surroundings" of $\widehat{\mu}$. Our idea is to extract the truth assignment $\widehat{\psi}$ induced by $\widehat{\mu}$ on the atoms of $\widehat{\varphi}$:

$$\widehat{\psi} \doteq \bigwedge_{[\widehat{A} \in \text{atoms}(\widehat{\varphi})\ s.t.\ \widehat{\mu} \models \widehat{A}]} \widehat{A} \ \wedge \bigwedge_{[\widehat{A} \in \text{atoms}(\widehat{\varphi})\ s.t.\ \widehat{\mu} \not\models \widehat{A}]} \neg \widehat{A}, \tag{2}$$

and then to look for another model $\widehat{\eta}$ for $\widehat{\psi}$. Notice that, any such model $\widehat{\eta}$ shares with $\widehat{\mu}$ the truth assignment on the atoms $\widehat{\psi}$, but with different values of the real variables.

The algorithm we propose is outlined in Fig. 13, where we extract the truth assignment $\widehat{\psi}$ induced by the UFLRA model $\widehat{\mu}$ on the atoms of $\widehat{\varphi}$, and we conjoin to it the *multiplication-line constraints*:

$$\widehat{\psi}^* \quad = \quad \widehat{\psi} \ \wedge \bigwedge_{f_*(x,y) \in \widehat{\psi}} \left( \begin{array}{l} (\ x = \widehat{\mu}[x] \wedge f_*(x,y) = \widehat{\mu}[x] * y\ ) \vee \\ (\ y = \widehat{\mu}[y] \wedge f_*(x,y) = \widehat{\mu}[y] * x\ ) \end{array} \right). \tag{3}$$

The main idea is to build an UFLRA underapproximation $\widehat{\psi}^*$ of the NRA formula $\psi$, in which all multiplications are forced to be linear. Notice that, this corresponds to searching for a solution along the multiplication lines described in §4.1, Fig. 5c and Fig. 5d. This idea is demonstrated by the following example.

*Example 5.1.* Consider the following formula $\varphi \doteq x * y = 10 \wedge (2 \le x \le 4) \wedge (2 \le y \le 4)$. Its abstraction is given by $\widehat{\varphi} \doteq f_*(x,y) = 10 \wedge (2 \le x \le 4) \wedge (2 \le y \le 4)$. Suppose the following model $\widehat{\mu}$ is returned by SMT-UFLRA-CHECK (line 6 in Fig. 3.2): $\widehat{\mu}[x] = 2$, $\widehat{\mu}[y] = 4$, $\widehat{\mu}[f_*(x,y)] = 10$. $\widehat{\mu}$ is a spurious interpretation because $2 * 4 \neq 10$ in NRA. However, using CHECK-NRA-MODEL we can still

**bool** CHECK-MODEL ($\widehat{\varphi}$, $\widehat{\mu}$, $\epsilon$):
1. $\langle sat, \widehat{\mu}^* \rangle$ := CHECK-NRA-MODEL ($\widehat{\varphi}$, $\widehat{\mu}$)
2. **if** $sat$:
3. **let** $\varphi^{\text{sat}}_{\widehat{\mu}^*}$ be the formula as defined in (5)
4. **return not** SMT-LRA-CHECK ($\neg\varphi^{\text{sat}}_{\widehat{\mu}^*}$)
5. **else**:
6. **return false**

Fig. 14. Detecting Satisfiability using an SMT(LRA) solver.

find an NRA-compliant model from the "guesses", which solves the following UFLRA-satisfiable formula (see (3)):

$$\widehat{\psi}^* \doteq f_*(x,y) = 10 \wedge (2 \leq x \leq 4) \wedge (2 \leq y \leq 4) \wedge \\ ((x = 2 \wedge f_*(x,y) = 2 * y) \vee (y = 4 \wedge f_*(x,y) = 4 * x)).$$

A possible UFLRA-model $\widehat{\mu}^*$ for $\widehat{\varphi}$ is:

$$\widehat{\mu}^*[x] = \frac{5}{2},\ \widehat{\mu}^*[y] = 4,\ \widehat{\mu}^*[f_*(x,y)] = 10,$$

that is also compliant with NRA.

Given the simplicity of the Boolean structure of the underapproximated formula, the check should in general be very cheap. In substance, we trade the complexity of NRA-solving with some extra Boolean reasoning. The drawback is that this is (clearly) still an incomplete procedure (see Example 5.2). However, in our experiments (for which we refer to §10) we have found it to be surprisingly effective for many problems.

*Example 5.2.* Consider the following NRA-satisfiable formula: $\varphi \doteq x * x = 2$. CHECK-NRA-MODEL would never find a model that is compliant with NRA, because it is driven by the "guesses" returned by SMT-UFLRA-CHECK which only produces rational number guesses (line 6 in Fig. 3.2) and $\varphi$ has models with irrational numbers only. Moreover, for the same reason the procedure SMT-NTA-CHECK would never terminate and will keep performing the refinement.

### 5.2 Detecting Satisfiability with NTA

We describe now how to extend the CHECK-MODEL procedure to deal with transcendental functions. The pseudo-code for CHECK-MODEL is shown in Fig. 14. As already written earlier, since our UFLRA solver is not able to deal with irrational numbers, in general we are not able to precisely represent a model $\mu$ for a formula with transcendental functions, since in most cases the model value for a term $\text{TF}(x)$ is irrational if the value for $x$ is rational. [8]

In general, therefore, we are not able to construct a model for a formula with transcendental functions. However, we may exploit this simple observation: we can still conclude that $\varphi$ is satisfiable if we are able to show that $\widehat{\varphi}$ is satisfiable *under all possible interpretations of* $f_{TF}$ that are guaranteed to include also TF. In order to do this, we proceed as follows.

Starting from the abstract model $\widehat{\mu}$ for $\widehat{\varphi}$, we first try to obtain a model $\widehat{\mu}^*$ that is consistent with multiplication terms, using the procedure CHECK-NRA-MODEL described in §5.1, while still treating all the transcendental functions as uninterpreted. Then, we compute safe lower and upper bounds

[8] With the notable exception of 0, at which both exp and sin have rational values.

$\underline{\mathrm{TF}(\widehat{\mu}^*[x])}_{\,l}$ and $\overline{\mathrm{TF}(\widehat{\mu}^*[x])}^{\,u}$ for the function TF at point $\widehat{\mu}^*[x]$ with the GET-POLYNOMIAL-BOUNDS function (see §4.2), with the current value of $\epsilon$.

Let $\psi$ be the formula obtained by substituting every $f_*(x,y) \in \widehat{\varphi}$ by $x * y$ and every variable $x \in \widehat{\varphi}$ by $\widehat{\mu}^*[x]$ with the exception of $\widehat{\pi}$. [9] We notice that, the satisfiability of the original formula $\varphi$ follows from the validity of the following formula $\rho$:

$$\rho \doteq \left( l_\pi < \widehat{\pi} < u_\pi \wedge \bigwedge_{f_{\mathrm{TF}}(x) \in \widehat{\varphi}} \underline{\mathrm{TF}(\widehat{\mu}^*[x])}_{\,l} \leq f_{\mathrm{TF}}(\widehat{\mu}^*[x]) \leq \overline{\mathrm{TF}(\widehat{\mu}^*[x])}^{\,u} \right) \rightarrow \psi, \tag{4}$$

that is, from the fact that $\psi$ holds for all possible interpretations of $\widehat{\pi}$ and the uninterpreted functions $f_{\mathrm{TF}}$ which fit in the bounds in the given points. In fact, since by construction $l_\pi < \pi < u_\pi \wedge \underline{\mathrm{TF}(\widehat{\mu}^*[x])}_{\,l} \leq \mathrm{TF}(\widehat{\mu}^*[x]) \leq \overline{\mathrm{TF}(\widehat{\mu}^*[x])}^{\,u}$, then the validity of the above formula implies that the formula is satisfied also by all interpretations which assign to $\widehat{\pi}$ the value of $\pi$ and all $f_{\mathrm{TF}}(\widehat{\mu}^*[x])$ the values of $\mathrm{TF}(\widehat{\mu}^*[x])$, which by construction satisfy the original formula $\varphi$.

In order to be able to use a quantifier-free SMT(LRA)-solver, we reduce the problem to the validity check of a pure LRA formula. Let $CT$ be the set of all terms $f_{\mathrm{TF}}(\widehat{\mu}^*[x])$ occurring in $\psi$. We replace each occurrence of $f_{\mathrm{TF}}(\widehat{\mu}^*[x])$ in $CT$ with a corresponding fresh variable $y_{f_{\mathrm{TF}}(\widehat{\mu}^*[x])}$ from a set $Y$. We then check the validity of the formula:

$$\varphi^{\mathrm{sat}}_{\widehat{\mu}^*} \doteq \forall \widehat{\pi} Y . \rho\{CT \mapsto Y\}. \tag{5}$$

If $\neg\varphi^{\mathrm{sat}}_{\widehat{\mu}^*}$ is unsatisfiable, we conclude that $\varphi$ is NTA-satisfiable. Clearly, this can be checked with a quantifier-free SMT(LRA)-solver, since $\neg\forall x.\phi$ is equivalent to $\exists x.\neg\phi$, and $x$ can then be removed by Skolemization.

*Example 5.3.* Consider the following NTA-satisfiable formula: $\varphi \doteq \exp(x) > 0$, and its initial abstraction: $\widehat{\varphi} \doteq f_{\exp}(x) > 0$. We demonstrate an execution of CHECK-MODEL (Fig. 14). Let $\widehat{\mu}^*$ be a model returned by CHECK-NRA-MODEL (line 1 in Fig. 14), where: $\widehat{\mu}^*[x] = 1$, $\widehat{\mu}^*[f_{\exp}(x)] = 1$. Using the bounds computed by GET-POLYNOMIAL-BOUNDS (Fig. 9), (4) and (5) become:

$$\rho \doteq \left( (\frac{333}{106} < \widehat{\pi} < \frac{355}{113}) \wedge (\frac{65}{24} \leq f_{\exp}(1) \leq \frac{325}{119}) \right) \rightarrow f_{\exp}(1) > 0$$

$$\varphi^{\mathrm{sat}}_{\widehat{\mu}^*} \doteq \forall \widehat{\pi}, y. \left( \left( (\frac{333}{106} < \widehat{\pi} < \frac{355}{113}) \wedge (\frac{65}{24} \leq y \leq \frac{325}{119}) \right) \rightarrow y > 0 \right)$$

$$\neg\varphi^{\mathrm{sat}}_{\widehat{\mu}^*} \doteq (\frac{333}{106} < \widehat{\pi} < \frac{355}{113}) \wedge (\frac{65}{24} \leq y \leq \frac{325}{119}) \wedge y \leq 0$$

Note that $\neg\varphi^{\mathrm{sat}}_{\widehat{\mu}^*}$ clearly LRA-unsatisfiable because of the constraints on $y$ and that can be shown by using any complete SMT(LRA) solver. Therefore, CHECK-MODEL returns true.

## 6 SOLVING VMT(NTA)

We now consider the problem of VMT(NTA), exploring several approaches. A first direction is the direct use of an SMT(NTA) solver, like the one described in previous sections, to extend known SMT-based verification algorithms. This extension can be relatively easy in the case of Bounded Model Checking (BMC) and k-induction, given that both techniques interact with the SMT(NTA)

[9] Notice that, we are treating $\pi$ as a zero-argument transcendental function.

solver largely as if it were a "black box". BMC presents to the SMT solver a sequence of proof obligations of the form

$$BMC(\mathcal{S}, P, k) \doteq I(X^{\langle 0\rangle}) \wedge T(X^{\langle 0\rangle}, X^{\langle 1\rangle}) \wedge \ldots \wedge T(X^{\langle k-1\rangle}, X^{\langle k\rangle}) \wedge \neg P(X^{\langle k\rangle})$$

for increasing values of $k$, until a counterexample trace is found, or resource limit is exhausted. BMC is an incomplete technique, oriented to finding violations, and likely to work well on unsafe instances. k-induction interacts with the SMT solver in a similar way, but also attempting to prove the validity of an inductive safety argument of the form

$$P(X^{\langle 0\rangle}) \wedge T(X^{\langle 0\rangle}, X^{\langle 1\rangle}) \wedge \ldots \wedge P(X^{\langle k-1\rangle}) \wedge T(X^{\langle k-1\rangle}, X^{\langle k\rangle}) \rightarrow P(X^{\langle k\rangle})$$

Although effective in the finite-state case for safe instances in some practical cases, k-induction turns out to be quite weak even for NRA. Consider for example the transition system $\mathcal{S} \doteq \langle I(X) \doteq x \geq 2 \wedge y \geq 2 \wedge z = x * y,\ T(X, X') \doteq x' = x + 1 \wedge y' = y + 1 \wedge z' = x' * y' \rangle$. The property $P(X) \doteq z \geq x + y$ is not k-inductive, not even for a very large value of $k$. Thus, the typical proving techniques based on k-induction using an SMT(NRA) solver will not be able to prove it. Thus, one could explore the extension to NTA of other SMT-based approaches, such as interpolation-based verification or IC3 (e.g. [11, 22, 43, 46]), that are often more effective than k-induction. However, there are substantial difficulties in lifting them to the case of NTA. First, they require an interpolating SMT(NRA) solver, which is not available. Second, the effectiveness of IC3 depends on an efficient, incremental interaction with the underlying SMT engine, which is asked to solve a large number of relatively cheap and often satisfiable queries. The procedure of §3, however, can be very expensive, especially for satisfiable queries. Finally, some IC3 extensions require the ability of performing (approximated) quantifier eliminations, a functionality not provided by the algorithm of Fig. 2.

Therefore, we propose a new direction, that radically differs from plugging a solver for SMT(NTA) into a known SMT-based algorithm. Instead, we adopt an abstraction-refinement approach, lifting the concepts of *incremental linearization* of NTA at the transition system level. In this approach, the VMT(NTA) problem is abstracted to a VMT(UFLRA) verification problem. A very efficient verification engine for VMT(UFLRA) is used as a black box, and the abstraction is refined in order to rule out spurious counterexamples.

### 6.1 The Main Procedure

The pseudo-code of the main algorithm is reported in Fig. 15. The main function IC3-NTA-PROVE takes as input a transition system $\mathcal{S}$ and a formula $\varphi$ and checks if $\varphi$ is an invariant property of $\mathcal{S}$. Notice that, as with SMT-NTA-CHECK in §3, IC3-NTA-PROVE is not guaranteed to terminate, so that we implicitly assume that it is stopped as soon as some given resource budget (e.g. time, memory, number of iterations) is exhausted.

We first apply a preprocessing step to $\langle \mathcal{S}, \varphi \rangle$ (function VMT-PREPROCESS) which is analogous to that of SMT-PREPROCESS in SMT-NTA-CHECK, producing $\langle \mathcal{S}', \varphi' \rangle$. Then we generate $\langle \widehat{\mathcal{S}}, \widehat{\varphi} \rangle$, an abstract UFLRA version of the input NTA transition system $\mathcal{S}$ and invariant $\varphi$, by invoking VMT-INITIAL-ABSTRACTION. As with the SMT case, the VMT-INITIAL-ABSTRACTION function replaces every non-linear multiplication and transcendental function in the transition system and property with the corresponding uninterpreted function symbol.

Then the procedure enters a loop (lines 3-11). At each iteration, the pair $\langle \widehat{\mathcal{S}}, \widehat{\varphi} \rangle$ is first checked by IC3-UFLRA-PROVE which implements IC3ia [10] [22], a procedure for VMT(UFLRA) that extends IC3

[10]Notice that, other approaches, such as interpolation-based model checking, could in principle be used. Here we consider IC3ia because it is currently the most effective procedure for VMT(UFLRA), and also because it allows us to leverage incrementality.

**bool** IC3-NTA-PROVE ($\mathcal{S}$ : transition system $\langle X, I, T\rangle$, $\varphi$ : invariant property):

1. $\langle \mathcal{S}', \varphi' \rangle$ := VMT-PREPROCESS($\mathcal{S}, \varphi$)
2. $\langle \widehat{\mathcal{S}}, \widehat{\varphi} \rangle$ := VMT-INITIAL-ABSTRACTION($\mathcal{S}', \varphi'$)
3. **while true**:
4. $\quad \langle ok, \widehat{cex} \rangle$ := IC3-UFLRA-PROVE($\widehat{\mathcal{S}}, \widehat{\varphi}$)
5. $\quad$ **if** ok: *# property proved*
6. $\quad\quad$ **return true**
7. $\quad \langle$is_cex, $\Gamma\rangle$ := SMT-NTA-CONCRETIZE-ABSTRACT-CEX($\mathcal{S}', \widehat{\mathcal{S}}, \widehat{\varphi}, \widehat{cex}$)
8. $\quad$ **if** is_cex: *# counterexample found*
9. $\quad\quad$ **return false**
10. $\quad \langle \Gamma_I, \Gamma_T \rangle$ := REFINE-TRANSITION-SYSTEM($\widehat{\mathcal{S}}, \Gamma$)
11. $\quad \widehat{\mathcal{S}} := \langle X, \widehat{I} \wedge \bigwedge \Gamma_I, \widehat{T} \wedge \bigwedge \Gamma_T \rangle$

Fig. 15. Verification of NTA transition systems via abstraction to UFLRA.

$\langle$constraint set, constraint set$\rangle$ REFINE-TRANSITION-SYSTEM ($\widehat{\mathcal{S}}, \Gamma$):

1. **let** $\langle X, \widehat{I}, \widehat{T} \rangle = \widehat{\mathcal{S}}$
2. $\langle \Gamma_I, \Gamma_T \rangle := \langle \emptyset, \emptyset \rangle$
3. **for each** $\gamma$ in $\Gamma$:
4. $\quad$ **if** vars($\gamma$) $\subseteq X^{\langle 0 \rangle}$:
5. $\quad\quad \Gamma_I := \Gamma_I \cup \{\gamma\{X^{\langle 0 \rangle} \mapsto X\}\}$
6. $\quad$ **else if** there exists $i > 0$ s.t. vars($\gamma$) $\subseteq X^{\langle i \rangle}$:
7. $\quad\quad \Gamma_T := \Gamma_T \cup \{\gamma\{X^{\langle i \rangle} \mapsto X\}, \gamma\{X^{\langle i \rangle} \mapsto X'\}\}$
8. $\quad$ **else** there exists $i > 0$ s.t. vars($\gamma$) $\subseteq X^{\langle i \rangle} \cup X^{\langle i+1 \rangle}$:
9. $\quad\quad \Gamma_T := \Gamma_T \cup \{\gamma\{X^{\langle i \rangle} \mapsto X\}\{X^{\langle i+1 \rangle} \mapsto X'\}\}$
10. **return** $\langle \Gamma_I, \Gamma_T \rangle$

Fig. 16. Refinement of the UFLRA transition system.

with implicit predicate abstraction [66]. If the invariant property is verified in the abstract domain UFLRA, then it is also verified in the original NTA domain, so that the whole procedure returns **true**. Otherwise, a counterexample is produced, and the SMT-NTA-CONCRETIZE-ABSTRACT-CEX primitive is used to check whether it is spurious. If not so, then the whole procedure returns **false**. If so, the linear constraints generated by SMT-NTA-CONCRETIZE-ABSTRACT-CEX are used to refine the abstraction of the transition system, and the procedure enters a new iteration.

### 6.2 Spuriousness Check and Abstraction Refinement

When IC3-UFLRA-PROVE returns a counterexample trace $\widehat{cex}$ for the abstract system $\widehat{\mathcal{S}}$, we use the dedicated routine SMT-NTA-CONCRETIZE-ABSTRACT-CEX to check for its spuriousness. The first step is to build a formula $\psi$ whose unsatisfiability implies that $\widehat{cex}$ is spurious. The formula $\psi$ is built by unrolling the transition relation of $\widehat{\mathcal{S}}$, and optionally adding constraints that restrict the allowed transitions to be compatible with the states in $\widehat{cex}$. If SMT-NTA-CONCRETIZE-ABSTRACT-CEX returns true, the property is violated. If SMT-NTA-CONCRETIZE-ABSTRACT-CEX returns false, we use the constraints $\Gamma$ produced during search to refine the transition system $\widehat{\mathcal{S}}$, using the procedure shown in Fig. 16. Essentially, REFINE-TRANSITION-SYSTEM translates back the linearization constraints

from their unrolled version (on variables $X^{\langle 0\rangle}, X^{\langle 1\rangle}, \ldots, X^{\langle k\rangle}$) to their "untimed" version (on variables $X$ and $X'$). Each $\gamma$ constraint is added either to the initial-states formula or to the transition relation formula, depending on the distance (in terms of steps) of the variables occurring in it. Care must be taken in order to deal with the case where $\gamma$ spawns across multiple time points: The routine SMT-NTA-CONCRETIZE-ABSTRACT-CEX is similar in spirit to the SMT-NTA-CHECK (discussed in §3), but it is designed in such a way that the constraints in $\Gamma$ never span more than a single transition step – for example, a monotonicity constraint over terms $\text{TF}(x^{\langle i\rangle})$ and $\text{TF}(y^{\langle j\rangle})$ is instantiated only if $j = i$ or $j = i + 1$. Note that, despite the restriction in the constraints instantiation, SMT-NTA-CONCRETIZE-ABSTRACT-CEX is as powerful as SMT-NTA-CHECK in detecting the spuriousness of a counterexample. This is because SMT-NTA-CONCRETIZE-ABSTRACT-CEX can basically instantiate a finite number of constraints restricted to a single transition step such that their conjunction is equivalent to any constraint instantiated by SMT-NTA-CHECK.

*Example 6.1.* We now demonstrate the execution of the IC3-NTA-PROVE procedure shown in Fig. 15 by an example. Consider the earlier mentioned transition system $\mathcal{S} \doteq \langle I(X), T(X, X')\rangle$ with the property $P(X)$, where $I(X) \doteq x \geq 2 \wedge y \geq 2 \wedge z = x * y, T(X, X') \doteq x' = x+1 \wedge y' = y+1 \wedge z' = x' * y'$, and $P(X) \doteq z \geq x + y$. After the initial abstraction, line 1-2, we have $\langle \widehat{\mathcal{S}} \doteq \langle \widehat{I}(X), \widehat{T}(X, X')\rangle, \widehat{P}(X)\rangle$ where: $\widehat{I}(X) \doteq x \geq 2 \wedge y \geq 2 \wedge z = f_*(x, y)$, $\widehat{T}(X, X') \doteq x' = x + 1 \wedge y' = y + 1 \wedge z' = f_*(x', y')$, $\widehat{P}(X) \doteq z \geq x + y$. In the loop (line 3-11), after execution of line 4, IC3-UFLRA-PROVE returns an abstract counterexample of length 2. Then SMT-NTA-CONCRETIZE-ABSTRACT-CEX tries to concretize the abstract counterexample: in its simplest form it builds a BMC unrolling of length 2 (more details are given in §8.2). SMT-NTA-CONCRETIZE-ABSTRACT-CEX returns false (meaning that the abstract counterexample is spurious) and the following set of linear constraints:

$$\Gamma \quad \doteq \quad \{ \, (x^{\langle 1\rangle} > 2 \wedge y^{\langle 1\rangle} > 2) \rightarrow f_*(x^{\langle 1\rangle}, y^{\langle 1\rangle}) > 2 * x^{\langle 1\rangle} + 2 * y^{\langle 1\rangle} - 4 \, \}$$

Next, REFINE-TRANSITION-SYSTEM refines $\widehat{\mathcal{S}}$ using $\Gamma$, such that:

$$\begin{aligned} \widehat{I}(X) \quad &\doteq \quad x \geq 2 \wedge y \geq 2 \wedge z = f_*(x, y), \\ \widehat{T}(X, X') \quad &\doteq \quad x' = x + 1 \wedge y' = y + 1 \wedge z' = f_*(x', y') \wedge \\ &\quad\; (x > 2 \wedge y > 2) \rightarrow f_*(x, y) > 2 * x + 2 * y - 4 \;\wedge \\ &\quad\; (x' > 2 \wedge y' > 2) \rightarrow f_*(x', y') > 2 * x' + 2 * y' - 4 \end{aligned}$$

After the refinement, another iteration of the loop is performed. Now IC3-UFLRA-PROVE returns true (meaning that the property is safe in $\widehat{\mathcal{S}}$), which means $P(X)$ is safe in $\mathcal{S}$. Therefore, true is returned.

## 7 PROOFS OF CORRECTNESS

In this section, we present the correctness proofs of the SMT procedure (discussed in §3–§5) and the VMT procedure (§6).

### 7.1 Correctness of the SMT Procedure

We prove the correctness of the SMT-NTA-CHECK procedure (Fig. 2), by first proving that the procedure maintains an overapproximation of the input problem $\varphi$. This is done by showing that the SMT-PREPROCESS function (Line 1) is NTA-satisfiability preserving – Lemma 7.1; and that the overapproximation is maintained in the loop (Line 5-12) – Lemma 7.3.

In what follows $\varphi'$ is the result of SMT-PREPROCESS($\varphi$), so that $\varphi' \doteq \varphi \wedge \varphi_{shift}$ (see (1) in §4.2.2).

LEMMA 7.1. *$\varphi$ is NTA-satisfiable if and only if $\varphi'$ is NTA-satisfiable.*

Proof. The "if" case is obvious. For the "only if" case, let $\mu$ be an NTA-interpretation satisfying $\varphi$. Then we can build another NTA-interpretation $\mu'$ by extending $\mu$ to the new symbols $\omega_x \in \varphi_{shift}$ introduced during preprocessing as follows. For each $\omega_x \in \varphi'$, if $\mu[x] \in [-\pi, \pi[$ then $\mu'[\omega_x] \doteq \mu[x]$; otherwise, we can choose $\mu'[\omega_x] \doteq c$ such that $\mu[sin(c)] = \mu[sin(x)]$, where $c \in \mathbb{R}$ and $c \in [-\pi, \pi[$. We can always choose such $c$ because sin is periodic, and $[-\pi, \pi]$ defines a period for it. Then by construction $\mu' \models \varphi_{shift}$, so that the statement holds. □

Let $\widehat{\varphi}$ be the result of SMT-initial-abstraction($\varphi'$). Consider a generic loop in SMT-NTA-check($\varphi$) and the value $\Gamma$ at the end of such loop. By construction all constraints in $\Gamma$ are either instances of those in Fig. 4, Fig. 11, Fig. 12, or tangent/secant constraints from Fig. 10, or linear shift constraints as discussed in §4.2.2. Let $\Gamma_{\text{TF}}$ be the constraints added by block-spurious-transcendental-terms($\widehat{\varphi}, \widehat{\mu}, \epsilon$) – line 5 of Fig. 3.

Lemma 7.2. *The constraints in $\Gamma_{TF}$ are NTA-valid when $f_{TF}()$ is interpreted as TF().*

Proof. The constraints in $\Gamma_{\text{TF}}$ are generated either by the basic refinement (block-spurious-nta-basic– line 1 in Fig. 10) or the tangent/secant refinement (line 4-26 in Fig. 10). In the former case, the constraints are instances of those in Fig. 11 or Fig. 12, and these constraints are valid in any theory which interprets $f_{\text{TF}}()$ and $\widehat{\pi}$ as TF() and $\pi$ respectively. For the latter case, the generated constraints are also NTA-valid when $f_{\text{TF}}()$ is interpreted as TF() because:

- The lower polynomial $P_l(x)$ and the upper polynomial $P_u(x)$ (in Fig. 10) for the TF() function are constructed using Taylor's theorem, such that the concavity (within the interval of interest) of the polynomials and the function is the same at the point of interest (spurious point). Therefore, $P_l(x)$ and $P_u(x)$ are guaranteed to be below and above TF(), respectively.
- When the concavity of TF() is positive (negative), then by construction the concavity of $P_l(x)$ is also positive (negative) and the tangent to $P_l(x)$ at the spurious point is below (above) $P_l(x)$ ($P_u(x)$) because of Proposition 2.5; therefore the tangent constraint is also below (above) the TF(). A similar argument applies to the constraints generated by the secant refinement.

□

Lemma 7.3. *If $\varphi$ is NTA-satisfiable, then $\widehat{\varphi} \wedge \bigwedge \Gamma$ is UFLRA-satisfiable.*

Proof. Let $\mu$ be an NTA-interpretation satisfying $\varphi$. By Lemma 7.1, we have an NTA-interpretation $\mu'$ that satisfies $\varphi'$. We build an UFLRA-interpretation $\widehat{\mu}$ satisfying $\widehat{\varphi} \wedge \bigwedge \Gamma$ as follows:

- for each $x \in \varphi'$, $\widehat{\mu}[x] \doteq \mu'[x]$
- for each $\omega_x \in \varphi'$, $\widehat{\mu}[\omega_x] \doteq \mu'[\omega_x]$
- for $\widehat{\pi}$, $\widehat{\mu}[\widehat{\pi}] \doteq \mu'[\pi]$
- for each $f_*(x, y) \in \widehat{\varphi}$, $\widehat{\mu}[f_*(x, y)] \doteq \mu'[x * y]$
- for each $f_{\text{TF}}(x) \in \widehat{\varphi}$, $\widehat{\mu}[f_{\text{TF}}(x)] \doteq \mu'[\text{TF}(x)]$

$\widehat{\mu}$ clearly satisfies $\widehat{\varphi}$ and all the constraints added by block-spurious-product-terms (constraints in Fig. 4 and the Tangent-plane constraints) are NTA-valid in any theory which interprets $f_*()$ as $*$, and all the constraints added by block-spurious-transcendental-terms are also NTA-valid by Lemma 7.2. Hence the statement holds. □

We now prove the correctness of the method to detect satisfiability. Let $\varphi$, $\varphi'$ be as above and let $\widehat{\varphi}$, $\widehat{\mu}$, *sat*, $\widehat{\mu}^*$, and $\varphi^{\text{sat}}_{\widehat{\mu}^*}$ be as in check-model (Fig. 14), so that $\widehat{\mu}$ is an UFLRA-model for $\widehat{\varphi}$. $\varphi^{\text{sat}}_{\widehat{\mu}^*}$ (5) is the formula for detecting the NTA-satisfiability of $\varphi$ as discussed in §5.2. (We recall that $\varphi^{\text{sat}}_{\widehat{\mu}^*}$ is a closed LRA formula.)

Lemma 7.4. *If* CHECK-MODEL$(\widehat{\varphi}, \widehat{\mu}, \epsilon)$ *returns true, then* $\varphi$ *is NTA-satisfiable.*

Proof. From Fig. 14, CHECK-MODEL$(\widehat{\varphi}, \widehat{\mu}, \epsilon)$ returns *true* if and only if CHECK-NRA-MODEL returns $\langle True, \widehat{\mu}^* \rangle$ and $\neg\varphi^{\text{sat}}_{\widehat{\mu}^*}$ is LRA-unsatisfiable –that is, $\varphi^{\text{sat}}_{\widehat{\mu}^*}$ is LRA-valid. From Fig. 13, if CHECK-NRA-MODEL returns $\langle True, \widehat{\mu}^* \rangle$, then $\widehat{\mu}^*$ is an UFLRA-model of a conjunction of literals $\widehat{\psi}^*$ (3) which tautologically entails $\widehat{\varphi}$ and it is s.t. each $f_*(x, y)$ equals $x * y$ in $\widehat{\mu}^*$. Thus $\widehat{\mu} \models \widehat{\varphi}$ and $\widehat{\mu}[f_*(x, y)] = \widehat{\mu}[x] * \widehat{\mu}[y]$ for each term $f_*(x, y) \in \widehat{\varphi}$.

Then we can construct an NTA-interpretation $\mu$ for $\varphi'$ as follows:

- for every $x, \omega_x \in \varphi'$, $\mu[x] \doteq \widehat{\mu}^*[x]$ and $\mu[\omega_x] \doteq \widehat{\mu}^*[\omega_x]$
- for $\widehat{\pi}$, $\mu[\widehat{\pi}] \doteq \pi$
- for every $\text{TF}(x) \in \varphi : \mu[\text{TF}(x)] \doteq \text{TF}(\widehat{\mu}^*[x])$

We show that $\mu$ is a model for $\varphi'$. The validity of $\varphi^{\text{sat}}_{\widehat{\mu}^*}$, and hence of (4), implies that $\varphi'$ is satisfied by every interpretation extending $\widehat{\mu}^*$ which assigns to each uninterpreted term $f_{\text{TF}}(\widehat{\mu}^*[x])$ a value within the bounds $[\underline{\text{TF}(\widehat{\mu}^*[x])}_{\,l}, \overline{\text{TF}(\widehat{\mu}^*[x])}^{\,u}]$ and assigns $\widehat{\pi}$ a value in $]l_\pi, u_\pi[$. $\mu$ is one of such interpretations because $\mu[\widehat{\pi}] \doteq \pi$ and $\mu[f_{\text{TF}}(\widehat{\mu}^*[x])] \doteq \text{TF}(\widehat{\mu}^*[x])$ which is in $[\underline{\text{TF}(\widehat{\mu}^*[x])}_{\,l}, \overline{\text{TF}(\widehat{\mu}^*[x])}^{\,u}]$ by Taylor's Theorem (§2.3). Thus $\mu$ is an NTA-model for $\varphi'$, and hence for $\varphi$. Therefore $\varphi$ is NTA-satisfiable. □

Theorem 7.5. *SMT-NTA-*CHECK *is sound, i.e. when it returns* $\langle True, \emptyset \rangle$ *then* $\varphi$ *is NTA-satisfiable and when it returns* $\langle False, \Gamma \rangle$ *then* $\varphi$ *is not NTA-satisfiable.*

Proof. SMT-NTA-CHECK returns $\langle False, \Gamma \rangle$ only when SMT-UFLRA-CHECK returns false. In this case, by Lemma 7.1 and Lemma 7.3, $\varphi$ is not NTA-satisfiable.

SMT-NTA-CHECK returns true only when CHECK-MODEL inside CHECK-REFINE returns true. In this case $\varphi$ is NTA-satisfiable by Lemma 7.4. □

### 7.2 Correctness of the VMT Procedure

We prove that the IC3-NTA-PROVE procedure (Fig. 15) is sound. First we show that the procedure maintains an overapproximation of the input transition system. Consider a generic loop in IC3-NTA-PROVE (Line 10-11) and let $\widehat{\mathcal{S}} := \langle X, \widehat{I} \wedge \bigwedge \Gamma_I, \widehat{T} \wedge \bigwedge \Gamma_T \rangle$ at the end of the loop.

Lemma 7.6. $\widehat{\mathcal{S}}$ *is an overapproximation of* $\mathcal{S}$.

Proof. We need to show that for every path $\sigma_k \doteq s_0, s_1, \ldots, s_{k-1}$ in $\mathcal{S}$, there is a path $\widehat{\sigma}_k$ in $\widehat{\mathcal{S}}$. This is true because:

- by definition of $\sigma_k$, we have $s_0 \models I$ and $s_i \wedge s_{i+1}\{X \mapsto X'\} \models T$ for $0 \leq i \leq k-2$, and
- similar to the the proof of Lemma 7.3, we can construct a path $\widehat{\sigma}_k$ in a such way that $\widehat{s}_0 \models \widehat{I} \wedge \bigwedge \Gamma_I$ and $\widehat{s}_i \wedge \widehat{s}_{i+1}\{X \mapsto X'\} \models \widehat{T} \wedge \bigwedge \Gamma_T$ for $0 \leq i \leq k-2$.

Hence the statement holds. □

Lemma 7.7. $\mathcal{S} \not\models \varphi$ *implies* $\widehat{\mathcal{S}} \not\models \widehat{\varphi}$.

Proof. $\mathcal{S} \not\models \varphi$ means there is a path $\sigma_k = s_0, s_1, \ldots, s_{k-1}$ in $\mathcal{S}$ such that $s_{k-1} \models \neg\varphi$. Then by Lemma 7.6, we can also construct a path $\widehat{\sigma}_k = \widehat{s}_0, \widehat{s}_1, \ldots, \widehat{s}_{k-1}$ in $\widehat{\mathcal{S}}$ in a way that $\widehat{s}_{k-1} \models \neg\widehat{\varphi}$ (similar to the proof of Lemma 7.3). Hence the statement holds. □

Lemma 7.8. *IC3-UFLRA-*PROVE$(\widehat{\mathcal{S}}, \widehat{\varphi})$ *is sound, i.e. when it returns* $\langle True, \ldots \rangle$ *then* $\widehat{\mathcal{S}} \models \widehat{\varphi}$*, and when it returns* $\langle False, \widehat{cex} \rangle$ *then* $\widehat{\mathcal{S}} \not\models \widehat{\varphi}$ *where* $\widehat{cex}$ *is a counterexample.*

Proof. Proof omitted – see [22]. □

Theorem 7.9. *IC3-NTA-prove$(\mathcal{S}, \varphi)$ is sound, i.e. when it returns True then $\mathcal{S} \models \varphi$ and when it returns False then $\mathcal{S} \not\models \varphi$.*

Proof. IC3-NTA-prove$(\mathcal{S}, \varphi)$ return true only when IC3-UFLRA-prove$(\widehat{\mathcal{S}}, \widehat{\varphi})$ returns $\langle True, \ldots \rangle$. By Lemma 7.6 and Lemma 7.7, $\mathcal{S} \models \varphi$. IC3-NTA-prove$(\mathcal{S}, \varphi)$ return false only when the simulation of the abstract counterexample $\widehat{cex}$ (Lemma 7.8) by the SMT-NTA-concretize-abstract-cex succeeds, and by Theorem 7.5 $\mathcal{S} \not\models \varphi$. □

## 8 IMPLEMENTATION

The SMT and VMT algorithms described in the previous sections have been implemented within the MathSAT SMT solver [23] and within the nuXmv symbolic model checker [17] respectively. The description of the procedures leaves some flexibility for different heuristics regarding refinement strategies and implementation choices. In this section, we describe the most important ones. We remark that these choices do not affect the soundness of the approach, but they can have an important impact on performance.

*MathSAT.* We have extended MathSAT with the procedures for NRA and NTA. The core solver only supports the exp and sin transcendental functions and nonlinear multiplications. Other functions (such as nonlinear division, square root, log, cos, tan, arcsin, arccos, arctan) are handled by encoding them in terms of the supported ones. For example, if the input formula $\varphi$ contains $\sqrt{x}$, it is rewritten as $\varphi\{\sqrt{x} \mapsto y\} \wedge (y \geq 0 \rightarrow y * y = x)$ where $y$ is a fresh variable. For formulae not involving transcendental functions, MathSAT can produce a model (in which all variables are assigned a rational value) when the formula is found to be satisfiable. Model generation is however not supported for transcendental functions. In this case, in principle it would be possible to produce rational bounds for the transcendental functions within which a model is guaranteed to exist (see §5.2), but this has not been implemented yet.

*nuXmv.* The nuXmv model checker has been extended to deal with non-linear multiplications and with the transcendental functions supported by its underlying SMT solver MathSAT. The verification engines of nuXmv have been extended to deal with invariant checking based on Bounded Model Checking, k-induction, and incremental linearization over IC3. The implementation of BMC and k-induction extends the algorithms already implemented in nuXmv for the LRA case. This extension was relatively simple, given the augmented capabilities of the underlying MathSAT solver. We also extended nuXmv with the capability to output BMC and k-induction proof obligations in SMT-LIB format. [11] The implementation of the VMT(NTA) algorithm based on incremental linearization was more involved. It builds on top of the best nuXmv engine for VMT(UFLRA), i.e. IC3 with Implicit Abstraction [22], leveraging its stateful, incremental nature. We chose not to implement the incremental linearization loop on top of an induction-based engine. We have no reason to believe that it would bring significant added value. Given the complementarity between IC3 and interpolation-based methods demonstrated in the finite-state case, an incremental linearization loop over an interpolation-based solver for VMT(UFLRA) might yield additional solving capability, and it is currently under consideration.

### 8.1 Implementation Details for SMT(NTA)

*8.1.1 Normalization.* We normalize (during preprocessing) the polynomials by applying the distributivity property of multiplication over addition, and by sorting both the monomials and the variables in each monomial using a total order (lexicographic). The goal of this is to reduce the

[11] More precisely, with the extension of the SMT-LIB format to transcendental function symbols.

number of unique multiplication terms, and therefore increase the number of shared $f_*()$ terms in the abstract SMT problem.[12]

*8.1.2 Arbitrary-precision Rationals.* We use the GMP arbitrary-precision library to represent rational numbers. In our (model-driven) approach, we may have to deal with numbers with very large numerators and/or denominators. It may happen that we get such rational numbers from the bad model $\widehat{\mu}$ for the variables appearing as arguments of transcendental functions. As a result of the piecewise-linear refinement (e.g. for the instantiation of the tangent plane constraints), we may end-up feeding to the SMT(UFLRA) solver numbers that may have (even exponentially) larger numerators and/or denominators (due to the fact that GET-POLYNOMIAL-BOUNDS uses power series). This might significantly slow-down the performance of the solver.

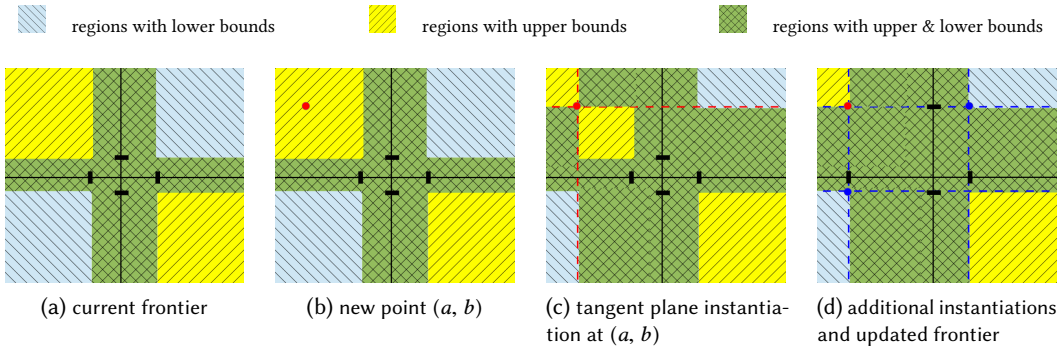
We address this issue along the following dimensions:

(1) *Continued fractions.* We approximate values in $\widehat{\mu}[x]$ having too large numerators and/or denominators by using continued fractions [56]. The precision of the rational approximation is increased periodically over the number of iterations. Thus, we delay the use of numbers with larger numerator and/or denominator, and eventually find those numbers if they are really needed.
(2) *Selective instantiation of tangent plane constraints.* While instantiating tangent plane constraints at the point $(a, b)$ for $x * y$ we observe that, in order to block a model $\widehat{\mu}$ such that $\widehat{\mu}[f_*(x, y)] \neq \widehat{\mu}[x] * \widehat{\mu}[y]$, it is sufficient to add one of the two equalities of the tangent plane constraints in Fig. 4; instead of instantiating a constraint at $(a, b)$, we can instantiate it at either $(a + \delta, b)$ or at $(a, b + \delta)$, for any value of $\delta$. In practice, if $a$ (resp. $b$) is a rational constant with a very large numerator or denominator, instead of instantiating one constraint at $(a, b)$, we instantiate two tangent constraints at $(\lfloor a \rfloor, b)$ and $(\lceil a \rceil, b)$.
(3) *Initial approximation for $\pi$.* We initialize the algorithm by choosing the values $l_\pi = \frac{333}{106}$ and $u_\pi = \frac{355}{113}$ since they give a very small difference ($\frac{1}{11978}$) with a limited number of digits.

*8.1.3 Heuristics for Refinement.* The descriptions of BLOCK-SPURIOUS-NRA-TERM and GET-POLYNOMIAL-BOUNDS leave some flexibility in deciding which constraints to add (and how many of them) at each iteration. It is possible to conceive strategies with an increasing degree of eagerness, from very lazy (e.g. adding only a single constraint per iteration) to more aggressive ones. The simple strategy we currently adopted consists in eagerly adding all the refinement constraints that are violated by the current abstract solution $\widehat{\mu}$, leaving the investigation of alternative strategies as future work.

*8.1.4 Tangent Plane Frontiers for Multiplications.* $x * y$ is a hyperbolic paraboloid surface, and a tangent plane to such surface cuts the surface into four regions: in two of them, the tangent plane is above the surface, thus providing an upper bound, whereas in the other two regions the tangent plane is below the surface (see Fig. 5). Each instantiation of the tangent plane constraints, therefore, only provides either a lower or an upper bound for a given region. Hence, there is a risk that the refinement procedure may go into an infinite loop, in which at each iteration a refined lower bound (respectively, upper bound) for a given point is added, when instead an upper bound (a lower bound, resp.) would be appropriate. In order to address the problem, we adopt the following strategy. For each $f_*(x, y)$ in the input formula, we maintain a *frontier* $\langle [l_x, u_x], [l_y, u_y] \rangle$ with the invariant that whenever $x \in [l_x, u_x]$ or $y \in [l_y, u_y]$, then $f_*(x, y)$ has both an upper and a lower bound in the abstract formula $\widehat{\varphi}$. Fig. 17 shows a graphical illustration of the strategy. Initially, the frontiers are set to $\langle [0, 0], [0, 0] \rangle$, corresponding to the "Zero" constraint of Fig. 4. Whenever a tangent plane

[12] We leave the investigation of more sophisticated methods for reducing the number of multiplications to future work.




(a) current frontier (b) new point $(a, b)$ (c) tangent plane instantiation at $(a, b)$ (d) additional instantiations and updated frontier

Fig. 17. Illustration of the tangent lemma frontier strategy.

constraint for $f_*(x, y)$ is instantiated at a point $(a, b)$, we also add further instantiations of the constraint and update the frontier as follows:

**case** $a < l_x$ **and** $b < l_y$**:** add tangent planes at $(a, u_y)$ and at $(u_x, b)$, and set the frontier to $\langle [a, u_x], [b, u_y] \rangle$;
**case** $a < l_x$ **and** $b > u_y$**:** add tangent planes at $(a, l_y)$ and at $(u_x, b)$, and set the frontier to $\langle [a, u_x], [l_y, b] \rangle$;
**case** $a > u_x$ **and** $b > u_y$**:** add tangent planes at $(a, l_y)$ and at $(l_x, b)$, and set the frontier to $\langle [l_x, a], [l_y, b] \rangle$;
**case** $a > u_x$ **and** $b < l_y$**:** add tangent planes at $(a, u_y)$ and at $(l_x, b)$, and set the frontier to $\langle [l_x, a], [b, u_y] \rangle$.

Remark 5. We remark that in each of the above four cases only one of the tangent lemma instantiation is required to preserve the tangent frontier invariant. However, rather than heuristically choosing one instantiation over the other, we instantiate both tangent lemmas.

### 8.2 Implementation Details for VMT(NTA)

*8.2.1 Counterexample Checking and Refinement.* Different heuristics can be considered for implementing the abstract counterexample check routine of Fig. 15 (function SMT-NTA-concretize-abstract-cex), trading generality for complexity. In particular, the unrolling of the transition system to check the feasibility of the abstract counterexample could be fully constrained by the states in $\widehat{cex}$ (thus checking only one abstract counterexample path per iteration); it could be only partially constrained (e.g. by considering only the Boolean variables and/or the state variables occurring only in linear constraints); or it could be left unconstrained, considering only the length of the abstract counterexample. In our current implementation, we only consider the length of $\widehat{cex}$ to build a BMC formula that checks for any counterexample of the given length, leaving the investigation of alternative strategies to future work.

*8.2.2 Reduction in the Number of Constraints.* In general, not all the constraints generated during a call to SMT-NTA-concretize-abstract-cex are needed to successfully block a counterexample, especially when using eager constraint instantiation strategies at the SMT level and when considering (like described above) all possible counterexample traces of a given length at each call to SMT-NTA-concretize-abstract-cex. In the long run, having a large number of redundant constraints can be quite harmful for performance.

In order to mitigate this problem, we apply a filtering strategy to the set of constraints, before adding them to the transition system. The strategy is based on the use of unsatisfiable cores, and it

$\langle$constraint set, constraint set$\rangle$ REDUCE-CONSTRAINTS $(\langle X, \widehat{I}, \widehat{T}\rangle, \widehat{cex}, \langle \Gamma_I, \Gamma_T\rangle)$:
1. $\psi := I^{\langle 0\rangle} \wedge \Gamma_I^{\langle 0\rangle} \wedge \widehat{cex}[0]^{\langle 0\rangle} \wedge \bigwedge_{i=0}^{|\widehat{cex}|-1} \left(T^{\langle i\rangle} \wedge \Gamma_T^{\langle i\rangle} \wedge \Gamma_T^{\langle i+1\rangle} \wedge \widehat{cex}[i+1]^{\langle i+1\rangle}\right)$
2. sat := SMT-UFLRA-CHECK$(\psi)$
3. **assert not** sat
4. **let** $C$ be an unsatisfiable core of $\psi$
5. $\Gamma_I = \{\gamma \in \Gamma_I \mid \gamma\{X \mapsto X^{\langle 0\rangle}\} \in C\}$
6. $\Gamma_T = \{\gamma \in \Gamma_T \mid \exists j > 0 \text{ s.t. } \gamma\{X \mapsto X^{\langle j\rangle}\}\{X' \mapsto X^{\langle j+1\rangle}\} \in C\}$
7. **return** $\langle \Gamma_I, \Gamma_T\rangle$

Fig. 18. Reducing the constraints needed for refinement.

is shown as pseudo-code in Fig. 18. The function REDUCE-CONSTRAINTS takes as input the current abstract transition system, the current abstract counterexample $\widehat{cex}$, and the sets of refinement constraints $\Gamma_I$ and $\Gamma_T$ returned by the function REFINE-TRANSITION-SYSTEM of Fig. 16. Then it builds an abstract BMC formula constrained by $\widehat{cex}$ (line 1 of Fig. 18). This formula is unsatisfiable, and we can extract a reduced set of constraints that is still sufficient for blocking the abstract counterexample by removing all the constraints that are not in the unsatisfiable core produced by the SMT solver (lines 5–6).

## 9 RELATED WORK

In the following, we overview the state of the art and compare it to incremental linearization. We first analyze the literature in SMT and then discuss VMT.

### 9.1 SMT

Various techniques have been explored for SMT(NRA) solving, that include complete methods based on quantifier elimination and incomplete methods based on interval constraint propagation and linearization. For SMT(NTA) solving, the approaches include incomplete methods based on interval constraint propagation, deductive methods, and linearization.

*9.1.1 Quantifier Elimination.* The two well-known and well-studied quantifier elimination procedures for the theory of nonlinear polynomials are *cylindrical algebraic decomposition* (CAD) [27] and *virtual substitution* (VS) [71]. They have doubly-exponential worst-case complexity [16, 29, 70]. Although these procedures have been studied for decades, their use in the SMT(NRA) solving is relatively recent. SMT-RAT [28] represents the first attempt to integrate CAD and VS in an SMT(NRA) solver. Then, z3 [31] and YICES [33] (winners of the SMT competition 2017 in the QF-NRA division) also implemented a variant [45] of CAD. In contrast to these approaches, incremental linearization is an incomplete technique and is not based on quantifier elimination.

*9.1.2 Interval Constraint Propagation.* Interval constraint propagation (ICP) [8] is an incomplete technique for solving constraints over NRA and NTA. Initially it has been investigated in [40, 59]. Now it has been integrated in several SMT solvers, most noticeably RASAT [68] for NRA, and ISAT3 [37] and DREAL [39] for NTA. Interestingly, DREAL relies on the notion of delta-satisfiability [38], which basically guarantees that there exists a variant (within a user-specified $\delta$ "radius") of the original problem such that it is satisfiable. The approach cannot guarantee that the original problem is satisfiable, since it relies on numerical approximation techniques that only compute safe overapproximations of the solution space.

There are a few key insights that differentiate our approach. First, our approach is based on linearization, it relies on solvers for SMT(UFLRA), and it proceeds by incrementally axiomatizing

transcendental functions. Compared to interval propagation, we avoid numerical approximations (even if within the bounds from DeltaSat). In a sense, the precision of the approximation is selectively detected at run time, while in iSAT3 and dReal this is a user defined threshold that is uniformly adopted in the computations. Second, our method relies on piecewise linear approximations, which can provide substantial advantages when approximating a slope – intuitively, interval propagation ends up computing a piecewise-constant approximation. Third, a distinguishing feature of our approach is the ability to (sometimes) prove the existence of a solution even if the actual values are irrationals, by reduction to an SMT-based validity check.

*9.1.3 Deductive Methods.* The MetiTarski [1] theorem prover relies on resolution and on a decision procedure for NRA to prove quantified inequalities involving transcendental functions. It works by replacing transcendental functions with upper- or lower-bound functions specified by means of axioms (corresponding to either truncated Taylor series or rational functions derived from continued fraction approximations), and then using an external decision procedure for NRA for solving the resulting formulae. Differently from our approach, MetiTarski cannot prove the existence nor compute a satisfying assignment of a solution, while we are able to (sometimes) prove the existence of a solution even if the actual values are irrationals. Finally, we note that MetiTarski may require the user to manually write axioms if the ones automatically selected from a predefined library are not enough. Our approach is much simpler, and it is completely automatic.

The approach presented in [35], where the NTA theory is referred to as NLA, is similar in spirit to MetiTarski in that it combines the SPASS theorem prover [69] with the iSAT3 SMT solver. The approach relies on the SUP(NLA) calculus that combines superposition-based first-order logic reasoning with SMT(NTA). Similarly to our work, the authors also use a UFLRA approximation of the original problem. This is however done only as a first check before calling iSAT3. In contrast, we rely on solvers for SMT(UFLRA), and we proceed by incrementally axiomatizing transcendental functions instead of calling directly an NTA solver. Another similarity with our work is the possibility of finding solutions in some cases. This is done by post-processing an inconclusive iSAT3 answer, trying to compute a certificate for a (point) solution for the narrow intervals returned by the solver, using an iterative analysis of the formula and of the computed intervals. Although similar in spirit, our technique for detecting satisfiable instances is completely different, being based on a logical encoding of the existence of a solution as an SMT(UFLRA) problem.

*Combination of interval propagation and theorem proving.* Gappa [30, 52] is a standalone tool and a tactic for the Coq proof assistant, that can be used to prove properties about numeric programs (C-like) dealing with floating-point or fixed-point arithmetic. Another related Coq tactic is Coq.Interval [55]. Both Gappa and Coq.Interval combine interval propagation and Taylor approximations for handling transcendental functions. A similar approach is followed also in [64], where a tool written in Hol-Light to handle conjunctions of non-linear equalities with transcendental functions is presented. The work uses Taylor polynomials up to degree two. NLCertify [49] is another related tool which uses interval propagation for handling transcendental functions. It approximates polynomials with sums of squares and transcendental functions with lower and upper bounds using some quadratic polynomials [2]. Internally, all these tools/tactics rely on multi-precision floating point libraries for computing the interval bounds.

A similarity between these approaches and our approach is the use of the Taylor polynomials. However, one distinguishing feature is that we use them to find lower and upper linear constraints by computing tangent and secant lines. Moreover, we do not rely on any floating point arithmetic library, and unlike the mentioned approaches, we can also prove the existence of a solution. On the other hand, some of the above tools employ more sophisticated/specialised approximations

for transcendental functions, which might allow them to succeed in proving unsatisfiability of formulae for which our technique is not sufficiently precise.

Finally, since we are in the context of SMT, our approach also has the benefits of being:

(1) fully automatic, unlike some of the above which are meant to be used within interactive theorem provers;
(2) able to deal with formulae with an arbitrary Boolean structure, and not just conjunctions of inequalities; and
(3) capable of handling combinations of theories (including uninterpreted functions, bit-vectors, arrays), which are beyond what the above, more specialised tools, can handle.

*9.1.4 Linearization.* Recent versions of the CVC4 [5] SMT solver also implement a variant of the incremental linearization procedure, as discussed in [60]. In comparison to our work, it does not support SMT(NTA). Moreover, it does not uses the NRA model-finding heuristic, as described in §5.1. In another work [58], the idea of using tangent planes has been explored in the context of SMT(NRA). A key difference is that the tangent planes are used to under-approximate predicates, while in our approach they are used to refine the over-approximation of the multiplication function.

A recent development in solving SMT(NRA) is the method of subtropical satisfiability [36], which is an incomplete method to detect satisfiability of conjunctions of strict inequality constraints. The method is efficient in returning satisfiable or unknown. It has been implemented in veriT [12]. Interestingly, the method encodes a sufficient condition for satisfiability into an LRA problem, which is a similarity to our approach for checking satisfiability of NRA constraints. Another linearization technique [51], though proposed for program analysis (see §9.2.2), has been implemented as an external theory solver for the CVC4 SMT solver. This approach can detect only unsatisfiability of NRA problems. In contrast to the subtropical satisfiability method and the approach based on [51], our approach (though incomplete) can be used to detect both satisfiable and unsatisfiable cases.

In the context of SMT(NTA), the work in [65] approximates the natural logarithm $ln$ with tangent lines, whereas we approximate not only a monotonic exp function ($ln$ can be rewritten in terms exp) but we can also handle periodic trigonometric functions. Moreover, we also exploit other properties (e.g. monotonicity) to derive additional axioms.

### 9.2 VMT

There are not many tools that deal with NRA and NTA transition systems. Here we discuss two classes of approaches: based on non-linear solving and based on linearization.

*9.2.1 Non-Linear Solving.* The most relevant is the recently proposed iSAT3 [50], that uses an interpolation-based [48, 50] approach to prove invariants. In addition to being an SMT solver, iSAT3 is also a bounded model checker for transition systems. It supports both NRA and NTA, and it additionally has support for some kinds of differential equations. iSAT3 is built on an SMT solver based on numeric techniques (interval arithmetic), and is able to provide results that are accurate up to the specified precision. In fact, in addition to "safe" and "unsafe" answers, iSAT3 may return "maybe unsafe" when it finds an envelope of given precision that may (but is not guaranteed to) contain a counterexample. Another relevant tool is dReach [47], a bounded model checker implemented on top of the dReal [39] SMT solver, that adopts numerical techniques similar to iSAT3. dReach has an expressiveness similar to iSAT3, but being a bounded model checker it is unable to prove properties.

*9.2.2 Linearization.* The work in [18] follows a reduction-based approach to check invariants of NRA transition systems. It over-approximates the non-linear terms with a coarse abstraction,

encoding into LRA some weak properties of multiplication like identity and sign. Another reduction-based approach is presented in [51] in the context of program analysis. The idea is to find a (tight) convex approximation of polynomials in form of polyhedron, thus obtaining a conservative linear transition system. The key differences of our approach with respect to [18, 51] are that we iteratively refine the abstraction, and we adopt a reduction to UFLRA. Furthermore, to the best of our knowledge, there is no available implementation of the approach [51] in a program analysis tool.

## 10 EXPERIMENTS

We now experimentally evaluate the proposed approaches for SMT (§10.1) and for VMT (§10.2). The experiments were run on a cluster of identical machines equipped with 2.6GHz Intel Xeon X5650 processors. The memory limit was set to 6 GB. We used 1000 seconds for the SMT experiments and 3600 seconds for the VMT experiments, respectively. [13] The results of the various solvers were automatically cross checked, and no discrepancies were reported.

We present the (most significant) data using tables and survival plots. (In Appendix A, we also present the scatter plots.) An archive containing the complete experimental evaluation, including benchmarks and results, is available at https://es.fbk.eu/people/irfan/papers/tocl17-data.tar.gz. The survival plots compare the performance of multiple approaches. The x-axis shows the solving time in log-scale, and the y-axis shows the number of instances solved within the corresponding time. (Notice that we may use different scales on different plots.) In the comparison, by VirtualBest we mean the results of a virtual portfolio solver that performs on each benchmark as the best of the solvers in the portfolio.

### 10.1 Experiments for SMT

*10.1.1 Benchmarks.* For the experimental evaluation on SMT we selected the following benchmarks. For NRA, we used all the SMT-LIB [6] benchmarks from the QF-NRA category. This is a class of 11354 benchmarks, among which 4963 are satisfiable, 5296 are unsatisfiable, and 1095 have unknown status. Since SMT(NTA) is not standardized in SMT-LIB, for NTA we adopted an extended version of SMT-LIB including special function symbols for sin, exp and $\pi$. We collected and encoded SMT(NTA) benchmarks from the following sources, for a total of 2512 benchmarks:

- verification queries over transcendental transition systems [50] deriving from SMT-based verification engines, including discretization of Bounded Model Checking of hybrid automata [4, 62];
- all the benchmarks from the MetiTarski distribution [1];
- all the SMT(NTA) benchmarks [14] from the dReal distribution [39];
- all the benchmarks from the iSAT3 distribution [50].

iSAT3 requires that the variables are constrained to small intervals when there are transcendental functions. Thus, it is unable to deal with the benchmarks in their original formulation. In order to include iSAT3 in the comparison we generated scaled-down versions of the NTA benchmarks, by adding bound constraints that force all the real variables in the problem to assume values in the $[-300, 300]$ interval. Since the SMT-LIB format is not accepted by iSAT3 and MetiTarski, we wrote scripts to cross-convert the benchmarks in the respective formats.

[13] The SMT problems are in general easier. The adopted time out for SMT is very close to the time out adopted in the SMT competition. The time out for VMT was dictated by restrictions in the computing power available, but appears to be reasonable given the number of benchmarks.

[14] dReal is also able to deal with Ordinary Differential Equations.

Table 1. Summary of SMT(NRA) experimental results.

| | Total | Total w/o MetiTarski | MetiTarski | Heizmann | Hong | HyComp | Kissing | LassoRanker | Sturm-MBO | Sturm-MGC | UltimateAutomizer | Zankl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | (11354) | (4348) | (7006) | (69) | (20) | (2752) | (45) | (821) | (405) | (9) | (61) | (166) |
| MathSAT | 3514/**5338** | 400/3045 | 3114/2293 | **3**/1 | 0/**20** | 17/2267 | 18/0 | **299**/432 | 0/**285** | 0/0 | 32/6 | 31/**34** |
| CVC4 | 3050/5322 | 315/**3068** | 2735/2254 | 0/2 | 0/**20** | 9/2253 | 4/0 | 298/**465** | 0/**285** | 0/2 | 4/8 | 0/33 |
| z3 | **4883**/5039 | **492**/2428 | **4391**/**2611** | 0/0 | 0/9 | **243**/2210 | **33**/0 | 105/115 | 0/47 | **2**/**7** | **47**/**13** | **62**/27 |
| Yices | 4817/5057 | 449/2480 | 4368/2577 | 0/**11** | 0/8 | 219/2182 | 10/0 | 120/233 | 0/2 | 0/0 | 40/12 | 60/32 |
| SMT-RAT | 4317/4475 | 133/1957 | 4184/2518 | 0/0 | 0/**20** | 98/1632 | 9/0 | 0/0 | 0/**285** | 0/1 | 2/0 | 24/19 |
| dReal | 0(5396)/4239 | 0(516)/2154 | 0(4880)/2085 | 0(0)/0 | 0(0)/**20** | 0(351)/2129 | 0(17)/**1** | 0(0)/0 | 0(0)/0 | 0(0)/0 | 0(45)/0 | 0(103)/4 |
| iSAT3 | 2404(2517)/4402 | 10(667)/2486 | 2394(1850)/1916 | 0(15)/0 | 0(0)/14 | 0(393)/**2276** | 1(29)/**1** | 0(121)/27 | 0(7)/148 | 0(3)/0 | 0(27)/2 | 9(72)/18 |
| VirtualBest | 5158/5756 | 767/3141 | 4391/2615 | 3/12 | 0/20 | 299/2303 | 33/1 | 310/466 | 0/285 | 2/7 | 48/13 | 72/34 |

Each column represents a benchmark family. Each entry shows the number of sat/unsat results reported. For tools reporting MaybeSat, the number is shown in parentheses. The best performer for each family is highlighted in boldface. The overall best is underlined.

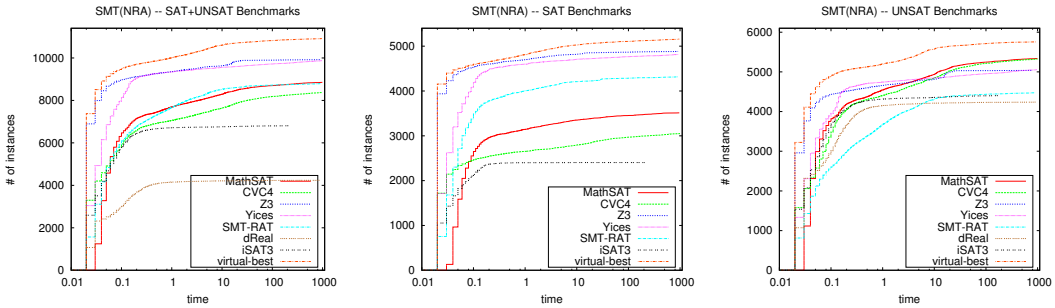
*10.1.2 Other Approaches.* We compared MathSAT with various SMT solvers. For SMT(NRA), we considered Yices [33], Z3 [31, 45] (Yices and Z3 were the winner and the runner-up in the QF-NRA division of the SMT competition 2017), and SMT-RAT [28], that implement expensive and complete techniques based on variants of CAD, and iSAT3 and dReal, based on interval constraint propagation. We also considered the recent version of CVC4 [5] that, as discussed in [60], is an independent implementation of our incremental linearization approach as described in [19]. For SMT(NTA), we considered iSAT3, dReal and MetiTarski [1], that implements a deductive approach. MetiTarski has been developed to deal with hard NTA problems with a few dozens of predicates, and it is not particularly suited for problems with a complicated Boolean structure.

*10.1.3 Results for SMT(NRA).* The results for SMT(NRA) are reported in Table 1. Each column shows a benchmark family, and each entry gives the number of satisfiable/unsatisfiable instances found. In the case of dReal and iSAT3, which can provide a MaybeSat answer, the value shown for the satisfiable instances contains only the definitive answers produced by the tools. In this case, we also report in parentheses the number of MaybeSat answers.

The MetiTarski benchmark class in SMT-LIB contains proof obligations deriving from MetiTarski executions. The MetiTarski benchmarks are about two thirds of the SMT(NRA) benchmarks in SMT-LIB, and have a very specific structure – they are conjunctions with no Boolean part, so that the SMT solvers are in fact being activated as $\mathcal{T}$-solvers for NRA. Given that MathSAT is also able to deal with the original problems in SMT(NTA), we also report the results of the comparison limited to the other benchmarks.

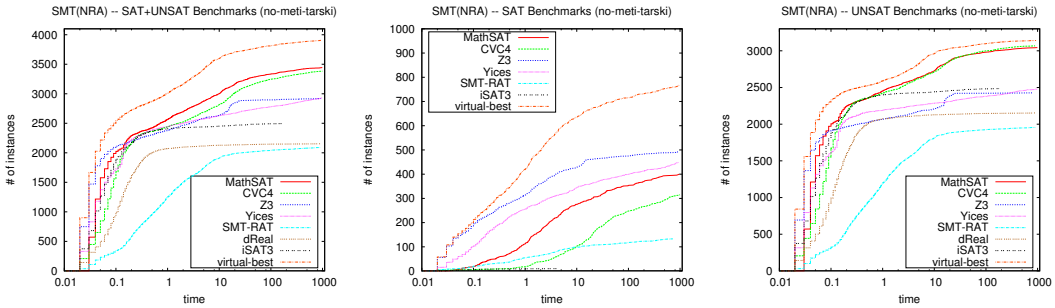
Table 1 demonstrates several interesting trends. First, on satisfiable instances, the complete techniques of Yices and z3 have superior performances than MathSAT in the overall case. However, if we exclude the MetiTarski benchmarks, we obtain comparable performance to complete solvers, and substantial advantage over incomplete solvers. We also notice that dReal is unable to conclude Sat in any of the benchmarks, and – regardless of the precision adopted – it claims MaybeSat on 583 unsatisfiable benchmarks.

Incremental linearization shines on unsatisfiable benchmarks. MathSAT (and also CVC4, that basically implements the same technique) demonstrates very good performance. Overall, MathSAT solves 8852 benchmarks (behind z3 with 9922 and Yices 9874), and is the strongest solver with




For each tool, the plots show the number of instances that could be solved (y axis) within the given time (x axis).

Fig. 19. Survival plots for all SMT(NRA) Benchmarks.




For each tool, the plots show the number of instances that could be solved (y axis) within the given time (x axis).

Fig. 20. Survival plots for SMT(NRA) Benchmarks excluding MetiTarski.

3445 solved (followed by CVC4 with 3383) on the benchmarks without MetiTarski problems. Interestingly, incremental linearization is highly complementary with respect to the more expensive techniques implemented in z3, Yices and SMT-RAT. MathSAT is able to solve 317 benchmarks that cannot be solved by other solvers (with the exception of CVC4), and it has been able to prove 579 benchmarks that have unknown status in the SMT-LIB. [15]

The scatter plots for all SMT(NRA) benchmarks are reported in Fig. 26; in Fig. 27 are reported the results excluding the MetiTarski benchmarks. The diagrams comparing MathSAT with z3, Yices and SMT-RAT show that the techniques to detect satisfiable instances are very complementary. The diagrams comparing MathSAT with iSAT3 and dReal show that the interval-based techniques are not good at detecting satisfiable instances. We also notice that incremental linearization and interval-based solvers are complementary on unsatisfiable instances. This suggests that integrating interval propagation within incremental linearization may be sometimes beneficial for efficiency.

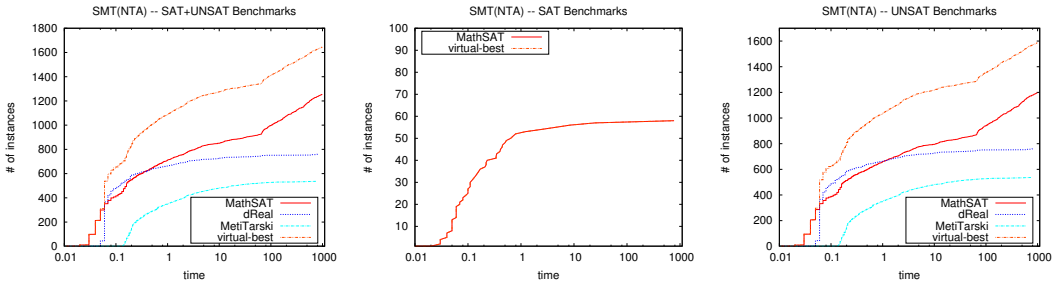
*10.1.4 Results for SMT(NTA).* The results on the SMT(NTA) benchmarks are reported in Table 2. We can see that MathSAT is able to solve more benchmarks than dReal and iSAT3. MetiTarski is unable to deal with benchmarks involving Boolean combinations, and thus could not be run on the BMC and dReal benchmarks. Similarly to the case of SMT(NRA), dReal is able to solve only unsatisfiable benchmarks, and returns MaybeSat in many situations that are in fact unsatisfiable. If we consider the scaled-down case, iSAT3 demonstrates better performance than dReal on BMC benchmarks, being able to prove more satisfiable benchmarks. Overall, however, it is still behind

[15] This value refers to the tagging of the SMT-LIB on 2017-06-17.

Table 2. Summary of SMT(NTA) experimental results (original problems above, bounded version below).

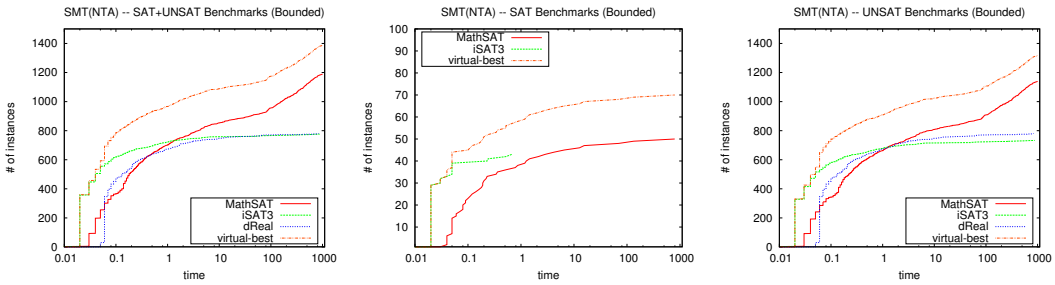
| | **Total** | **BMC** | **MetiTarski** | **dReal** |
|---|---|---|---|---|
| | **(2512)** | **(887)** | **(681)** | **(944)** |
| MathSAT | **58/1198** | **44/541** | 0/299 | **14/358** |
| dReal | 0(1177)/761 | 0(294)/392 | 0(368)/267 | 0(528)/102 |
| MetiTarski | 0/536 | - | **0/536** | - |
| VirtualBest | 58/1588 | 44/546 | 0/621 | 14/421 |
| MathSAT | **50/1139** | **39/547** | **0/313** | **11/279** |
| iSAT3 | 43(1316)/733 | 36(270)/475 | 0(455)/195 | 7(591)/63 |
| dReal | 0(1056)/782 | 0(267)/403 | 0(293)/269 | 0(496)/110 |
| VirtualBest | 70/1317 | 53/565 | 0/406 | 17/346 |

Each column represents a benchmark family. Each entry shows the number of sat/unsat results reported. For tools reporting MaybeSat, the number is shown in parentheses. The best performer for each family is highlighted in boldface. The overall best is underlined.




For each tool, the plots show the number of instances that could be solved (y axis) within the given time (x axis).

Fig. 21. Survival plots for SMT(NTA) – Unbounded Benchmarks.




For each tool, the plots show the number of instances that could be solved (y axis) within the given time (x axis).

Fig. 22. Survival plots for SMT(NTA) – Bounded Benchmarks.

MathSAT. The scatter plots are reported in Fig. 28. The comparison between MathSAT and MetiTarski only considers the MetiTarski benchmarks. Despite MetiTarski being superior to MathSAT in terms of solved instances, we notice that MathSAT can be at times much faster

than MetiTarski, and that it is not strictly dominated – MetiTarski runs out of memory in many benchmarks that MathSAT can solve. In fact, there is a substantial difference between the number of benchmarks solved by MetiTarski (536) and the virtual best solver (621). The scatters in Fig. 28 also confirm the complementarity between incremental linearization and interval constraint propagation solvers. It is possible to notice a stripe of points that are solved in nearly constant time by dReal and iSAT3 that are increasingly harder for MathSAT. On the bounded benchmarks, dReal and iSAT3 contribute with almost 200 instances solved to the virtual best (see Table 2). This confirms the potential benefits of integrating interval propagation within incremental linearization.

### 10.2 Experiments for VMT

*10.2.1 Benchmarks.* We collected 114 benchmarks over VMT(NRA) and 126 benchmarks over VMT(NTA). The VMT(NRA) benchmarks set consists of the following families:

- *Handcrafted:* 14 (13 safe and 1 unsafe) benchmarks.
- *HyComp:* 7 (3 safe, 4 unsafe) benchmarks from [26] and converted to VMT(NRA) using HyComp [21].
- *HYST:* 65 benchmarks generated from the hybrid systems examples in the HYST [4] distribution, by approximating the continuous time with a fixed rate sampling. This process is done automatically using an extended version of HYST. Since the generated benchmarks are approximations, their status is unknown.
- *iSAT3 and iSAT3-CFG:* 11 (7 safe, 4 unsafe) benchmarks from [50] and the iSAT3 examples available online.
- *nuXmv:* 2 safe benchmarks, with complex Boolean structure, from the nuXmv users' mailing list.
- *SAS13:* 13 benchmarks are generated from the C programs used in [14], but interpreted over NRA instead of the theory of IEEE floating-point numbers. This makes some of the instances unsafe.
- *TCM:* 2 safe benchmarks from the Simulink models (taken from the case study [15]) by first generating the C code using Embedded Coder [16] and then encoding the programs into VMT(NRA) models.

The VMT(NTA) benchmarks set consists of the following families:

- *Handcrafted:* 3 (safe) benchmarks.
- *HARE, HYST, and WBS:* 50, 57, and 12 benchmarks obtained from the discretization (using HyComp [21]) of the hybrid systems benchmarks from [4, 25, 62] (unknown status).
- *iSAT3:* 4 benchmarks from the iSAT3 webpage.

Similar to the case of SMT(NTA), we included a scaled-down version of the VMT(NTA) benchmarks, with real-valued variables constrained to the $[-300, 300]$ interval, in order to include iSAT3 in the comparison.

The benchmarks of the comparison are expressed in the VMT-LIB language [17], that extends the SMT-LIB language so that the SMT formulae are interpreted as the initial condition and the transition relation of the transition system. Cross-translation scripts were developed for the iSAT3 and dReal benchmarks.

*10.2.2 Other Approaches.* We compared the incremental linearization algorithm implemented in nuXmv (in the following referred to as IncreLin-nuXmv) against the static abstraction approach

[16] https://www.mathworks.com/products/embedded-coder/

[17] Information available at http://www.vmt-lib.org/. A complete specification of the VMT-LIB language can be found in the nuXmv User Manual available at https://nuxmv.fbk.eu/downloads/nuxmv-user-manual.pdf.

Table 3. Summary of VMT(NRA) experimental results.

| | Total (114) | Handcrafted (14) | HyComp (7) | HYST (65) | ISAT3 (1) | ISAT3-CFG (10) | nuXmv (2) | SAS13 (13) | TCM (2) |
|---|---|---|---|---|---|---|---|---|---|
| IncreLin-nuXmv | 16/**65** | **1**/**13** | 1/**3** | 7/**34** | 0/0 | 2/**6** | 0/**2** | **5**/**5** | 0/**2** |
| StaticLin-nuXmv | 0/37 | 0/4 | 0/1 | 0/19 | 0/0 | 0/4 | 0/**2** | 0/**5** | 0/**2** |
| Interpolation-iSAT3$_{[1e-1]}$ | 2(47)/48 | 0(8)/2 | 0(3)/0 | 2(23)/**34** | 0/0 | 0(4)/**6** | 0/0 | 0(9)/4 | 0/**2** |
| Interpolation-iSAT3$_{[1e-9]}$ | 2(19)/47 | 0(3)/2 | 0(2)/0 | 2(3)/32 | 0/0 | 0(3)/**6** | 0/0 | 0(8)/**5** | 0/**2** |
| K-induction-NRA-MathSAT | 13/20 | **1**/2 | 0/0 | 6/12 | 0/0 | 1/4 | 0/0 | **5**/0 | 0/**2** |
| K-induction-NRA-Z3 | 25/22 | **1**/2 | **2**/0 | **15**/12 | 0/0 | 2/**6** | 0/0 | **5**/0 | 0/**2** |
| K-induction-NRA-dReal | 0(32)/16 | 0(4)/2 | 0(2)/0 | 0(19)/9 | 0/0 | 0(2)/5 | 0/0 | 0(5)/0 | 0/0 |
| BMC-NRA-MathSAT | 15/0 | **1**/0 | 0/0 | 7/0 | 0/0 | 2/0 | 0/0 | **5**/0 | 0/0 |
| BMC-NRA-Z3 | **26**/0 | **1**/0 | **2**/0 | **15**/0 | 0/0 | **3**/0 | 0/0 | **5**/0 | 0/0 |
| BMC-NRA-dReal | 0(39)/0 | 0(8)/0 | 0(2)/0 | 0(19)/0 | 0/0 | 0(3)/0 | 0/0 | 0(7)/0 | 0/0 |
| VirtualBest | 26/71 | 1/13 | 2/3 | 15/39 | 0/0 | 3/7 | 0/2 | 5/5 | 0/2 |

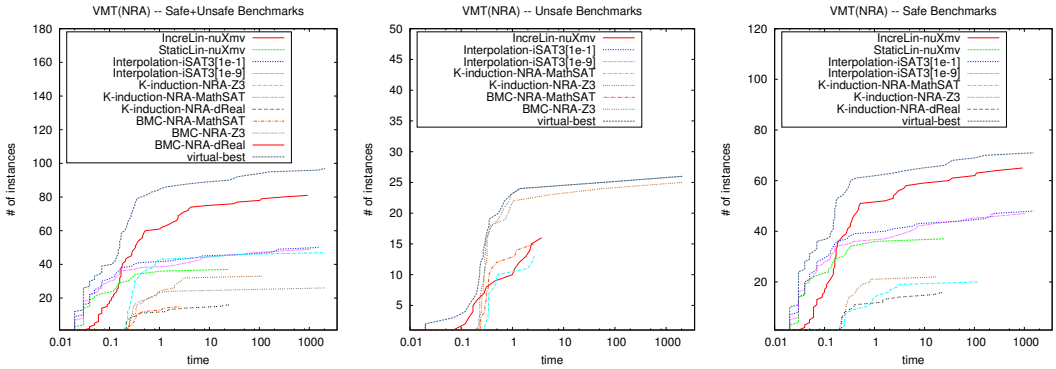
Each column represents a benchmark family. Each entry shows the number of unsafe/safe results reported. For tools reporting MaybeUnsafe, the number is shown in parentheses. The best performer for each family is highlighted in boldface. The overall best is underlined.

proposed (for the case of NRA) in [18] (referred to as StaticLin-nuXmv). In the case of NTA, StaticLin-nuXmv also uses some basic constraints to limit the interpretation of transcendental functions. We also compared IncreLin-nuXmv against various approaches based on the direct use of an SMT(NRA) or SMT(NTA) solver.

- BMC: BMC-NRA-Z3, based on z3 (limited to NRA); BMC-NRA-dReal and BMC-NTA-dReal, based on dReal; BMC-NRA-MathSAT and BMC-NTA-MathSAT, based on MathSAT.
- k-induction: [18] K-induction-NRA-Z3, based on z3 (limited to NRA); K-induction-NRA-dReal and K-induction-NTA-dReal, based on dReal; K-induction-NRA-MathSAT and K-induction-NTA-MathSAT, based on MathSAT.
- The interpolation-based iSAT3 engine [50], with two different levels of precision – Interpolation-iSAT3$_{[1e-1]}$ and Interpolation-iSAT3$_{[1e-9]}$.

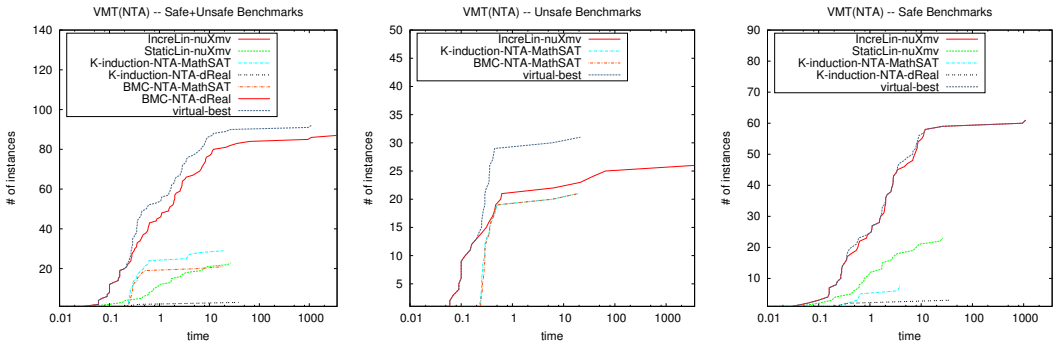
*10.2.3 Results for VMT(NRA).* The results for VMT(NRA) are summarized in Table 3. Each column shows a benchmark family, and each entry gives the number of unsafe/safe instances found. In parenthesis we report the number of undefinite MaybeUnsafe answers. The experimental results for VMT(NRA) clearly demonstrate the merits of incremental linearization compared to BMC and k-induction approaches based on the direct usage of SMT(NRA). IncreLin-nuXmv is by far the best solver among all the available ones, with 81 benchmarks solved against the 50 solved by the runner-up Interpolation-iSAT3$_{[1e-1]}$. The purely-static approach of StaticLin-nuXmv only solves 37, thus confirming the importance of incremental refinement of the abstraction. The scatter

[18] The BMC and k-induction solvers using dReal are based on a script developed with the specific objective of this evaluation. When dReal returns a MaybeSat result on a BMC query (or a base case query) the script returns MaybeUnsafe. When dReal returns MaybeSat on an inductive step query the script considers it a failed induction and increases $k$. This does not hamper the correctness of Safe results. However, we notice that the loop is slightly different from the one that is implemented in nuXmv on top of MathSAT and of z3, in that they may run out of resources trying to prove that an inductive query is satisfiable. The ability to "bail out" from hard satisfiable inductive checks gives dReal a slight advantage.




For each tool, the plots show the number of instances that could be solved (y axis) within the given time (x axis).

Fig. 23. Survival plots for VMT(NRA).




For each tool, the plots show the number of instances that could be solved (y axis) within the given time (x axis).

Fig. 24. Survival plots for VMT(NTA) – Unbounded Benchmarks.

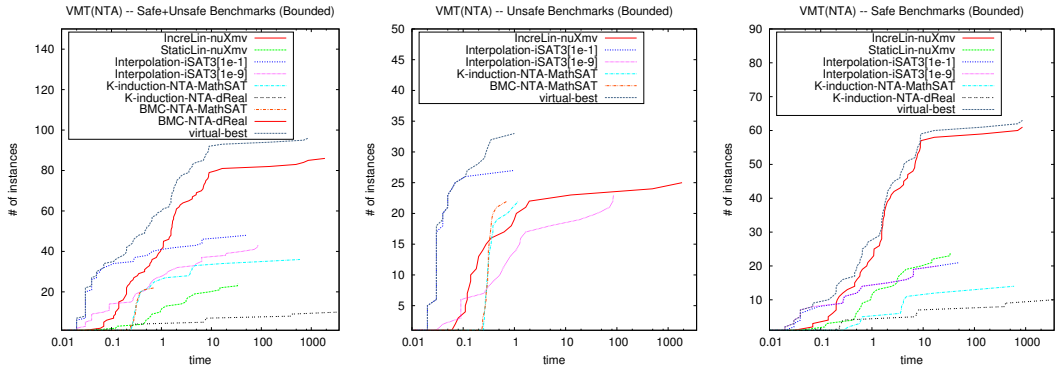
plot in Fig. 30 reports the comparison on VMT(NRA) problems. We notice that IncreLin-nuXmv dominates the approaches based on k-induction: every safe instance solved by k-induction is also solved by incremental linearization (with the exception of one instance). Interpolation has some complementarity w.r.t. incremental linearization: it solves 5 benchmarks where IncreLin-nuXmv (as well as all the other engines) times out. We conjecture that incremental linearization for SMT(NRA) could be extended to produce interpolants applicable to verification. We also see that the BMC approach based on complete SMT(NRA) techniques is the best in terms of unsafe instances. This can be due to the fact that we are adopting a concretization approach that is based on incremental linearization and not a complete solver.

*10.2.4 Results for VMT(NTA).* The results for VMT(NTA) are reported in Table 4. Overall, the experimental results clearly demonstrate the merits of incremental linearization at the level of VMT. IncreLin-nuXmv is by far the best solver among all the available ones, confirming the trends of the VMT(NRA) case. In fact, incremental linearization dominates even more in VMT(NTA), given the lack of complete techniques: the performance of IncreLin-nuXmv is very close to the virtual best (see Table 4). The survival plots reported in Fig. 24 and 25, and the scatter plots reported in Fig. 31 and 32 corroborate these observations. Compared to VMT(NRA), we notice many more memory-outs, suggesting that the solvers may proceed by brute force and generate useless lemmas.

Table 4. Summary of VMT(NTA) experimental results (original problems above, bounded version below).

| | **Total** (126) | **Handcrafted** (3) | **HARE** (50) | **HYST** (57) | **ISAT3** (4) | **WBS** (12) |
|---|---|---|---|---|---|---|
| IncreLin-nuXmv | **26/61** | 0/**3** | 0/**43** | **24/4** | 0/**2** | **2/9** |
| StaticLin-nuXmv | 0/23 | 0/0 | 0/15 | 0/1 | 0/0 | 0/7 |
| K-induction-NTA-MathSAT | 21/8 | 0/1 | 0/0 | 20/1 | 0/0 | 1/6 |
| K-induction-NTA-dReal | 0(66)/3 | 0(0)/0 | 0(3)/0 | 0(54)/2 | 0(0)/0 | 0(9)/1 |
| BMC-NTA-MathSAT | 21/0 | 0/0 | 0/0 | 19/0 | 0/0 | 1/0 |
| BMC-NTA-dReal | 0(67)/0 | 0(0)/0 | 0(4)/0 | 0(54)/0 | 0(0)/0 | 0(9)/0 |
| VirtualBest | 31/61 | 0/3 | 0/43 | 29/4 | 0/2 | 2/9 |
| IncreLin-nuXmv | 25/**61** | 0/**3** | 0/**43** | 23/4 | 0/2 | **2/9** |
| StaticLin-nuXmv | 0/24 | 0/0 | 0/15 | 0/1 | 0/1 | 0/7 |
| Interpolation-iSAT3$_{[1e-1]}$ | **27**(36)/21 | 0(0)/**3** | 0(4)/8 | **27**(21)/**5** | 0(1)/**3** | 0(10)/2 |
| Interpolation-iSAT3$_{[1e-9]}$ | 23(27)/20 | 0(0)/**3** | 0(2)/7 | 23(19)/**5** | 0(0)/**3** | 0(6)/2 |
| K-induction-NTA-MathSAT | 22/14 | 0/**3** | 0/4 | 22/1 | 0/0 | 0/6 |
| K-induction-NTA-dReal | 0(55)/10 | 0(0)/**3** | 0(4)/4 | 0(42)/1 | 0(0)/1 | 0(9)/1 |
| BMC-NTA-MathSAT | 22/0 | 0/0 | 0/0 | 22/0 | 0/0 | 0/0 |
| BMC-NTA-dReal | 0(55)/0 | 0(0)/0 | 0(4)/0 | 0(42)/0 | 0(0)/0 | 0(9)/0 |
| VirtualBest | 33/63 | 0/3 | 0/43 | 31/5 | 0/3 | 2/9 |

Each column represents a benchmark family. Each entry shows the number of unsafe/safe results reported. For tools reporting MaybeUnsafe, the number is shown in parentheses. The best performer for each family is highlighted in boldface. The overall best is underlined.




For each tool, the plots show the number of instances that could be solved (y axis) within the given time (x axis).

Fig. 25. Survival plots for VMT(NTA) – Bounded Benchmarks.

### 10.3 Discussion

The experimental evaluation suggests several interesting general remarks:

- Incremental linearization is highly competitive for NRA. Given the maturity of the other solvers, we found it quite surprising to discover how well it compares on SMT as well as VMT benchmarks.

- Incremental linearization appears to be a very effective technique to deal with transcendental functions, a theory for which no complete methods are available. A key factor in the case of trigonometric functions appears to be the idea of reduction to the base interval. Incremental linearization could be improved further by integration with interval-based techniques.
- Despite their inherent theoretical limitations, in practice the satisfiability-oriented methods presented in this paper are often able to conclude SAT in cases where complete techniques bail out or do not exist.[19]

## 11 CONCLUSIONS AND FUTURE WORK

In this paper we have proposed incremental linearization as a general framework for automated reasoning about nonlinear polynomials and transcendental functions such as exponentiation and trigonometric functions.

The key idea is to abstract nonlinear and transcendental functions as uninterpreted functions in the combined theories of Linear Real Arithmetic (LRA) and Equalities and Uninterpreted Functions (UF), for which efficient solvers exist. The uninterpreted functions in the abstract space, corresponding to nonlinear and transcendental functions in the original theory, are incrementally axiomatized by means of upper- and lower-bounding piecewise-linear constraints. The refinement is driven by the existence of spurious models (or traces, in the case of VMT). In the case of transcendental functions, the management of irrational values is particularly tricky, and care is required to ensure the soundness of the abstraction.

The approach is proved to be correct, and it has been implemented in the MATHSAT SMT solver and in the NUXMV model checker. We carried out an extensive experimental evaluation on a wide set of SMT and VMT benchmarks, and the results clearly demonstrate the effectiveness of incremental linearization.

In the future, we plan to work along two main directions. At the level of basic SMT solving, we will investigate the integration of incremental linearization with complementary techniques, such as interval constraint propagation - in order to further improve efficiency - or CAD - that is a more powerful technique but also more expensive. Other important topics include sufficient criteria for satisfiability without necessarily producing a model [36], and the use of factorization techniques for the polynomial pre-processing and polynomial-level axiomatization. We are also interested in exploring the impact of a native approach for handling transcendental functions that are currently supported via a reduction to sin and exp by way of rewriting. Finally, we will evaluate the incremental linearization to solve NIA benchmarks.

At the level of VMT, a very important step is the extension beyond invariant checking, to deal with full LTL specifications over nonlinear transition systems. We conjecture that the computation of limit values for series may help to deal with transition systems deriving from discrete-step controllers. Finally, we will extend incremental linearization to deal with hybrid automata featuring polynomial and transcendental dynamics.

## ACKNOWLEDGEMENTS

This work was partly funded by the SC-SQUARE project.

## REFERENCES

[1] Behzad Akbarpour and Lawrence C. Paulson. 2010. MetiTarski: An Automatic Theorem Prover for Real-Valued Special Functions. *J. Autom. Reasoning* 44, 3 (2010), 175–205.

[19] However, we remark the converse is also true. For example, our techniques will never be able to prove the satisfiability of formulae which admit only models with irrational values, such as $x * x = 2$ (see Example 5.2).

[2] Xavier Allamigeon, Stéphane Gaubert, Victor Magron, and Benjamin Werner. 2013. Certification of inequalities involving transcendental functions: combining SDP and max-plus approximation. In *Control Conference (ECC), 2013 European*. IEEE, 2244–2250.

[3] Alessandro Armando, Jacopo Mantovani, and Lorenzo Platania. 2009. Bounded model checking of software using SMT solvers instead of SAT solvers. *STTT* 11, 1 (2009), 69–83.

[4] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. 2015. HYST: a source transformation and translation tool for hybrid automaton models. In *HSCC*. ACM, 128–133.

[5] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli. 2011. CVC4. In *CAV (LNCS)*, Vol. 6806. Springer, 171–177.

[6] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. 2016. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`. (2016).

[7] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. 2009. Satisfiability Modulo Theories. In *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press, 825–885.

[8] Frédéric Benhamou and Laurent Granvilliers. 2006. Continuous and Interval Constraints. In *Handbook of Constraint Programming*. Foundations of Artificial Intelligence, Vol. 2. Elsevier, 571–603.

[9] J.L. Berggren, J. Borwein, and P. Borwein. 2013. *Pi: A Source Book*. Springer New York.

[10] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. 1999. Symbolic Model Checking Without BDDs. In *TACAS (TACAS '99)*. Springer-Verlag, London, UK, 193–207.

[11] Johannes Birgmeier, Aaron R. Bradley, and Georg Weissenbacher. 2014. Counterexample to Induction-Guided Abstraction-Refinement (CTIGAR). In *CAV (LNCS)*, Vol. 8559. Springer, 831–848.

[12] Thomas Bouton, Diego Caminha Barbosa De Oliveira, David Déharbe, and Pascal Fontaine. 2009. veriT: An Open, Trustable and Efficient SMT-Solver. In *CADE-22 (LNCS)*, Vol. 5663. Springer, 151–156.

[13] Aaron R. Bradley. 2011. SAT-Based Model Checking without Unrolling. In *VMCAI (LNCS)*, Vol. 6538. Springer, 70–87.

[14] Martin Brain, Vijay D'Silva, Alberto Griggio, Leopold Haller, and Daniel Kroening. 2013. Interpolation-Based Verification of Floating-Point Programs with Abstract CDCL. In *SAS (LNCS)*, Vol. 7935. Springer, 412–432.

[15] Guillaume Brat, David H. Bushnell, Misty Davies, Dimitra Giannakopoulou, Falk Howar, and Temesghen Kahsai. 2015. Verifying the Safety of a Flight-Critical System. In *FM (LNCS)*, Vol. 9109. Springer, 308–324.

[16] Christopher W. Brown and James H. Davenport. 2007. The complexity of quantifier elimination and cylindrical algebraic decomposition. In *ISSAC*. ACM, 54–60.

[17] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. 2014. The nuXmv Symbolic Model Checker. In *CAV (LNCS)*, Vol. 8559. Springer, 334–342.

[18] Adrien Champion, Arie Gurfinkel, Temesghen Kahsai, and Cesare Tinelli. 2016. CoCoSpec: A Mode-Aware Contract Language for Reactive Systems. In *SEFM (LNCS)*, Vol. 9763. Springer, 347–366.

[19] Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. 2017. Invariant Checking of NRA Transition Systems via Incremental Reduction to LRA with EUF. In *TACAS (LNCS)*, Vol. 10205. 58–75.

[20] Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. 2017. Satisfiability Modulo Transcendental Functions via Incremental Linearization. In *CADE 26 (LNCS)*, Leonardo de Moura (Ed.), Vol. 10395. Springer, 95–113.

[21] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. 2015. HyComp: An SMT-Based Model Checker for Hybrid Systems. In *TACAS (LNCS)*, Vol. 9035. Springer, 52–67.

[22] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. 2016. Infinite-state invariant checking with IC3 and predicate abstraction. *Formal Methods in System Design* 49, 3 (2016), 190–218.

[23] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. 2013. The MathSAT5 SMT Solver. In *TACAS (LNCS)*, Vol. 7795. Springer, 93–107.

[24] Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani. 2011. Computing Small Unsatisfiable Cores in SAT Modulo Theories. *Journal of Artificial Intelligence Research, JAIR* 40 (April 2011), 701–728.

[25] Alessandro Cimatti, Sergio Mover, and Mirko Sessa. 2016. From Electrical Switched Networks to Hybrid Automata. In *FM (LNCS)*, Vol. 9995. Springer, 164–181.

[26] Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. 2012. A quantifier-free SMT encoding of non-linear hybrid automata. In *FMCAD*. IEEE, 187–195.

[27] George E. Collins. 1974. Quantifier elimination for real closed fields by cylindrical algebraic decomposition-preliminary report. *ACM SIGSAM Bulletin* 8, 3 (1974), 80–90.

[28] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp, and Erika Ábrahám. 2015. SMT-RAT: An Open Source C++ Toolbox for Strategic and Parallel SMT Solving. In *SAT (LNCS)*, Vol. 9340. Springer, 360–368.

[29] James H. Davenport and Joos Heintz. 1988. Real Quantifier Elimination is Doubly Exponential. *J. Symb. Comput.* 5, 1/2 (1988), 29–35.

[30] Florent de Dinechin, Christoph Quirin Lauter, and Guillaume Melquiond. 2011. Certifying the Floating-Point Implementation of an Elementary Function Using Gappa. *IEEE Trans. Computers* 60, 2 (2011), 242–253.

[31] Leonardo Mendonça de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *TACAS (LNCS)*, Vol. 4963. Springer, 337–340.

[32] Leonardo Mendonça de Moura, Harald Rueß, and Maria Sorea. 2003. Bounded Model Checking and Induction: From Refutation to Verification (Extended Abstract, Category A). In *CAV (LNCS)*, Vol. 2725. Springer, 14–26.

[33] Bruno Dutertre. 2014. Yices 2.2. In *CAV (LNCS)*, Vol. 8559. Springer, 737–744.

[34] Niklas Eén and Niklas Sörensson. 2003. Temporal induction by incremental SAT solving. *Electr. Notes Theor. Comput. Sci.* 89, 4 (2003), 543–560.

[35] Andreas Eggers, Evgeny Kruglov, Stefan Kupferschmid, Karsten Scheibler, Tino Teige, and Christoph Weidenbach. 2011. Superposition Modulo Non-linear Arithmetic. In *FroCoS (LNCS)*, Cesare Tinelli and Viorica Sofronie-Stokkermans (Eds.), Vol. 6989. Springer, 119–134.

[36] Pascal Fontaine, Mizuhito Ogawa, Thomas Sturm, and Xuan-Tung Vu. 2017. Subtropical Satisfiability. In *FroCoS (LNCS)*, Clare Dixon and Marcelo Finger (Eds.), Vol. 10483. Springer, 189–206.

[37] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. 2007. Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure. *JSAT* 1, 3-4 (2007), 209–236.

[38] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. 2012. $\delta$-Complete Decision Procedures for Satisfiability over the Reals. In *IJCAR (LNCS)*, Vol. 7364. Springer, 286–300.

[39] Sicun Gao, Soonho Kong, and Edmund M. Clarke. 2013. dReal: An SMT Solver for Nonlinear Theories over the Reals. In *CADE-24 (LNCS)*, Vol. 7898. Springer, 208–214.

[40] Laurent Granvilliers and Frédéric Benhamou. 2006. Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques. *ACM Trans. Math. Softw.* 32, 1 (2006), 138–156.

[41] M. Hazewinkel. 1993. *Encyclopaedia of Mathematics: Stochastic Approximation – Zygmund Class of Functions.* Springer Netherlands.

[42] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. 1998. What's Decidable about Hybrid Automata? *J. Comput. System Sci.* 57, 1 (1998), 94 – 124. http://www.sciencedirect.com/science/article/pii/S0022000098915811

[43] Krystof Hoder and Nikolaj Bjørner. 2012. Generalized Property Directed Reachability. In *SAT (LNCS)*, Vol. 7317. Springer, 157–171.

[44] Richard M Hueschen. 2011. *Development of the Transport Class Model (TCM) aircraft simulation from a sub-scale Generic Transport Model (GTM) simulation.* Technical Report. NASA Langley Research Center.

[45] Dejan Jovanovic and Leonardo de Moura. 2012. Solving non-linear arithmetic. *ACM Comm. Computer Algebra* 46, 3/4 (2012), 104–105.

[46] Anvesh Komuravelli, Arie Gurfinkel, and Sagar Chaki. 2016. SMT-based model checking for recursive programs. *Formal Methods in System Design* 48, 3 (2016), 175–205.

[47] Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. 2015. dReach: $\delta$-Reachability Analysis for Hybrid Systems. In *TACAS (LNCS)*, Vol. 9035. Springer, 200–205.

[48] Stefan Kupferschmid and Bernd Becker. 2011. Craig Interpolation in the Presence of Non-linear Constraints. In *FORMATS (LNCS)*, Vol. 6919. Springer, 240–255.

[49] Victor Magron. 2014. NLCertify: A Tool for Formal Nonlinear Optimization. In *ICMS (LNCS)*, Vol. 8592. Springer, 315–320.

[50] Ahmed Mahdi, Karsten Scheibler, Felix Neubauer, Martin Fränzle, and Bernd Becker. 2016. Advancing Software Model Checking Beyond Linear Arithmetic Theories. In *Haifa Verification Conference (Lecture Notes in Computer Science)*, Vol. 10028. 186–201.

[51] Alexandre Maréchal, Alexis Fouilhé, Tim King, David Monniaux, and Michaël Périn. 2016. Polyhedral Approximation of Multivariate Polynomials Using Handelman's Theorem. In *VMCAI (LNCS)*, Vol. 9583. Springer, 166–184.

[52] Érik Martin-Dorel and Guillaume Melquiond. 2016. Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq. *J. Autom. Reasoning* 57, 3 (2016), 187–217.

[53] Yuri Vladimirovich Matiyasevich. 1993. *Hilbert's Tenth Problem.* MIT Press.

[54] Kenneth L. McMillan. 2003. Interpolation and SAT-Based Model Checking. In *CAV (LNCS)*, Vol. 2725. Springer, 1–13.

[55] Guillaume Melquiond. 2011. Coq-interval. (2011).

[56] George L. Nemhauser and Laurence A. Wolsey. 1988. *Integer and Combinatorial Optimization.* Wiley-Interscience, New York, NY, USA.

[57] Ivan Niven. 1961. *Numbers: Rational and Irrational.* Mathematical Association of America.

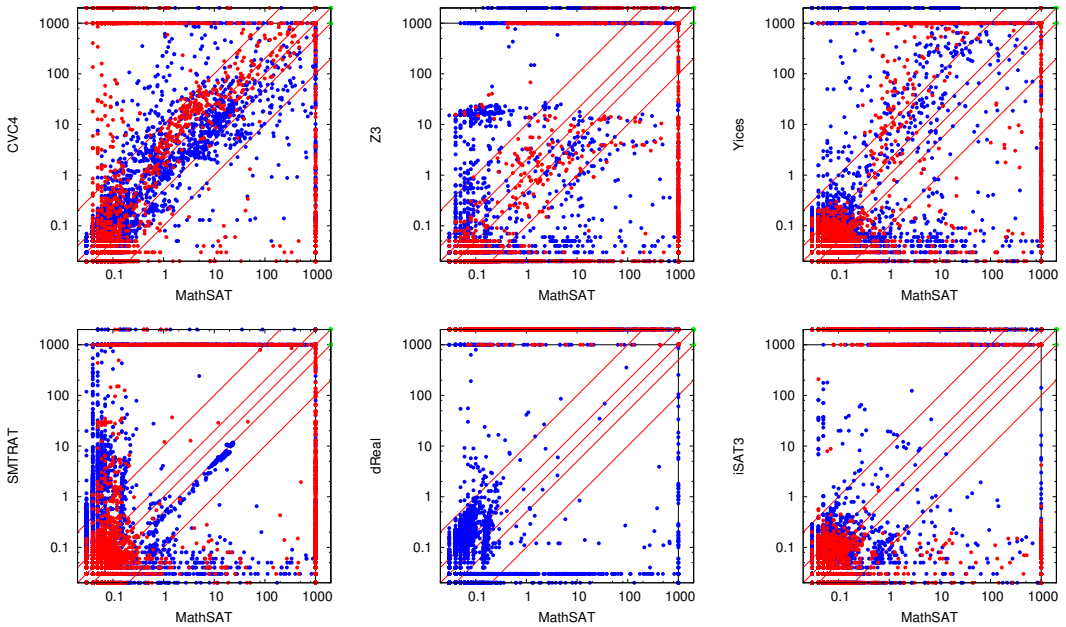
[58] Pierluigi Nuzzo, Alberto Puggelli, Sanjit A. Seshia, and Alberto L. Sangiovanni-Vincentelli. 2010. CalCS: SMT solving for non-linear convex constraints. In *FMCAD*. IEEE, 71–79.

[59] Stefan Ratschan. 2006. Efficient solving of quantified inequality constraints over the real numbers. *ACM Trans. Comput. Log.* 7, 4 (2006), 723–748.

[60] Andrew Reynolds, Cesare Tinelli, Dejan Jovanovic, and Clark Barrett. 2017. Designing Theory Solvers with Extensions. In *FroCoS (LNCS)*, Vol. 10483. Springer.
[61] Daniel Richardson. 1968. Some Undecidable Problems Involving Elementary Functions of a Real Variable. *J. Symb. Log.* 33, 4 (1968), 514–520.
[62] Nima Roohi, Pavithra Prabhakar, and Mahesh Viswanathan. 2017. HARE: A Hybrid Abstraction Refinement Engine for Verifying Non-linear Hybrid Automata. In *TACAS (LNCS)*, Vol. 10205. 573–588.
[63] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. 2000. Checking Safety Properties Using Induction and a SAT-Solver. In *FMCAD (LNCS)*, Vol. 1954. Springer, 108–125.
[64] Alexey Solovyev and Thomas C. Hales. 2013. Formal Verification of Nonlinear Inequalities with Taylor Interval Approximations. In *NFM (LNCS)*, Vol. 7871. Springer, 383–397.
[65] Ashish Tiwari. 2015. Time-Aware Abstractions in HybridSal. In *CAV (LNCS)*, Vol. 9206. Springer, 504–510.
[66] Stefano Tonetta. 2009. Abstract Model Checking without Computing the Abstraction. In *FM (LNCS)*, Vol. 5850. Springer, 89–105.
[67] E.J. Townsend. 2007. *Functions of a Complex Variable.* Read Books.
[68] Vu Xuan Tung, To Van Khanh, and Mizuhito Ogawa. 2016. raSAT: An SMT Solver for Polynomial Constraints. In *IJCAR (LNCS)*, Vol. 9706. Springer, 228–237.
[69] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. 2009. SPASS Version 3.5. In *CADE-22 (LNCS)*, Vol. 5663. Springer, 140–145.
[70] Volker Weispfenning. 1988. The Complexity of Linear Problems in Fields. *J. Symb. Comput.* 5, 1/2 (1988), 3–27.
[71] Volker Weispfenning. 1997. Quantifier Elimination for Real Algebra - the Quadratic Case and Beyond. *Appl. Algebra Eng. Commun. Comput.* 8, 2 (1997), 85–101.

## A SCATTER PLOTS

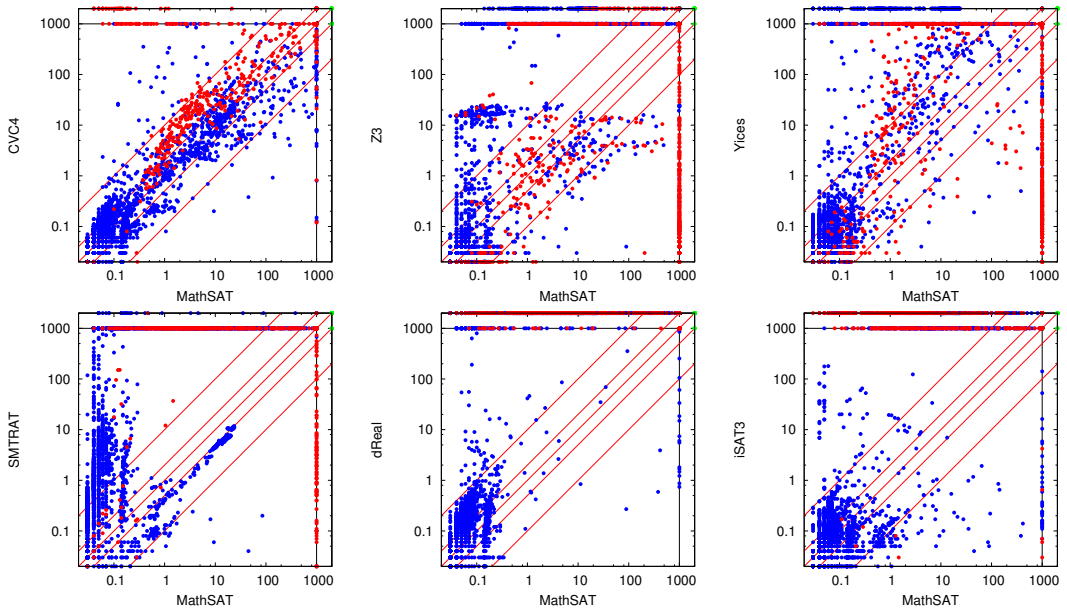
The scatter plots compare pairwise solvers $S_1$ and $S_2$ on individual benchmarks: each point $(t_1, t_2)$ in a scatter represents a benchmark problem that was solved in $t_i$ time by solver $S_i$. We adopt a logarithmic scale, and report time out and memory out as separate lines.




Satisfiable instances are shown in red, unsatisfiable ones in blue. Green dots indicate unknown instances. Diagonal lines mark 2x and 10x performance differences. Points on the inner edges indicate timeouts, those on the outer edges indicate other errors (memory outs or aborts).

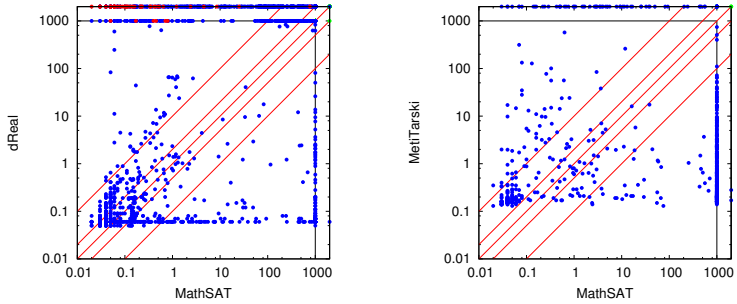
Fig. 26. Scatter plots for all SMT(NRA) Benchmarks.

Received December 2017; revised March 2018; revised May 2018; accepted May 2018



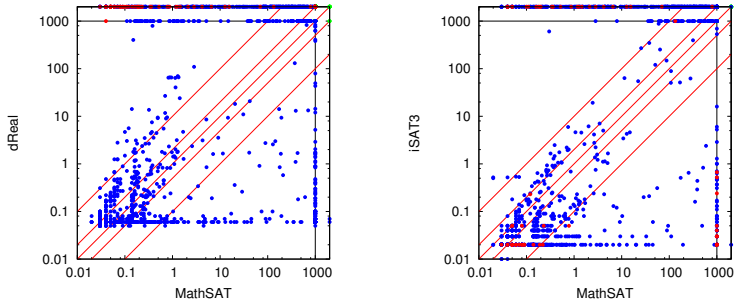

Satisfiable instances are shown in red, unsatisfiable ones in blue. Green dots indicate unknown instances. Diagonal lines mark 2x and 10x performance differences. Points on the inner edges indicate timeouts, those on the outer edges indicate other errors (memory outs or aborts).

Fig. 27. Scatter plots for SMT(NRA) Benchmarks excluding MetiTarski.



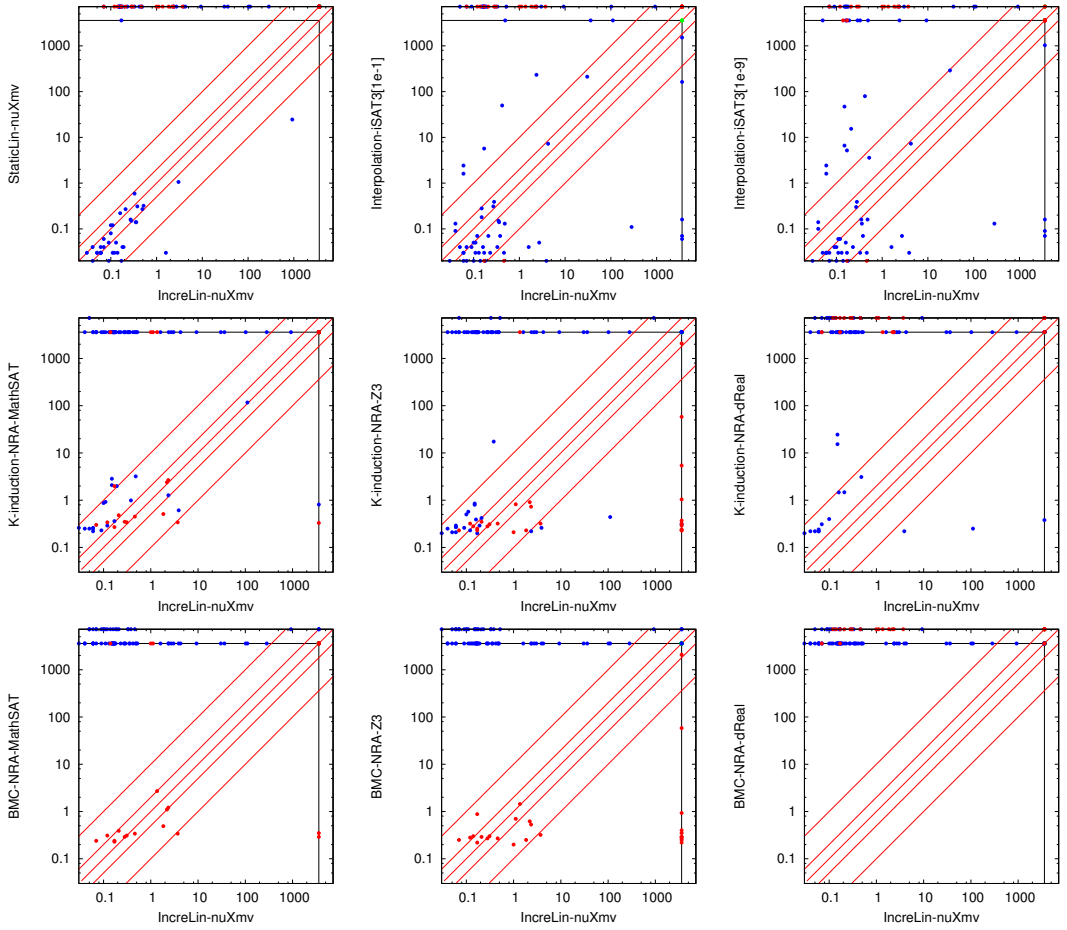

Satisfiable instances are shown in red, unsatisfiable ones in blue. Green dots indicate unknown instances. Diagonal lines mark 2x and 10x performance differences. Points on the inner edges indicate timeouts, those on the outer edges indicate other errors (memory outs or aborts).

Fig. 28. Scatter plots for SMT(NTA) – Unbounded Benchmarks.



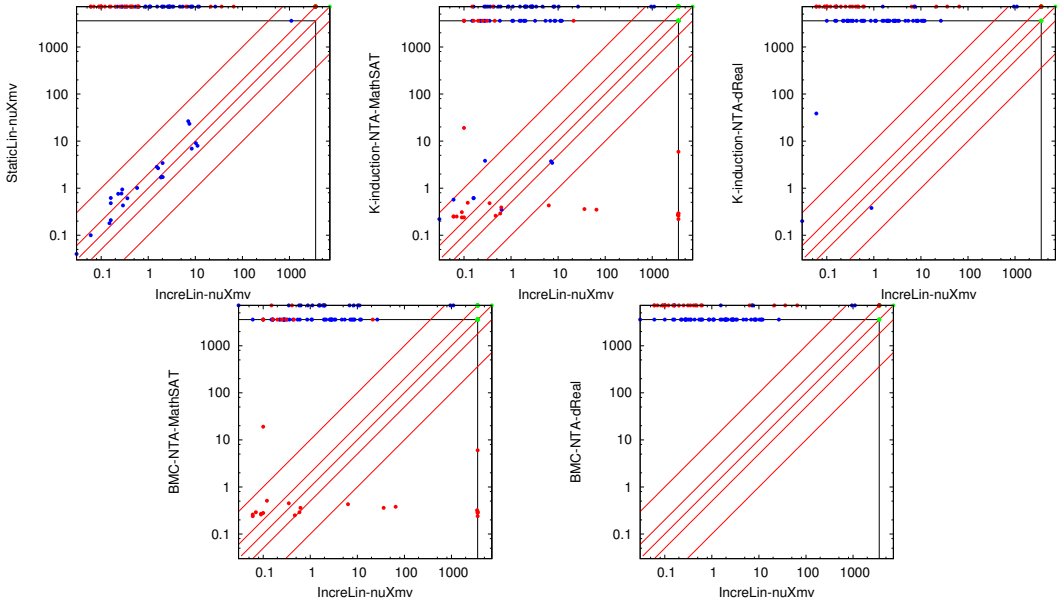

Satisfiable instances are shown in red, unsatisfiable ones in blue. Green dots indicate unknown instances. Diagonal lines mark 2x and 10x performance differences. Points on the inner edges indicate timeouts, those on the outer edges indicate other errors (memory outs or aborts).

Fig. 29. Scatter plots for SMT(NTA) – Bounded Benchmarks.



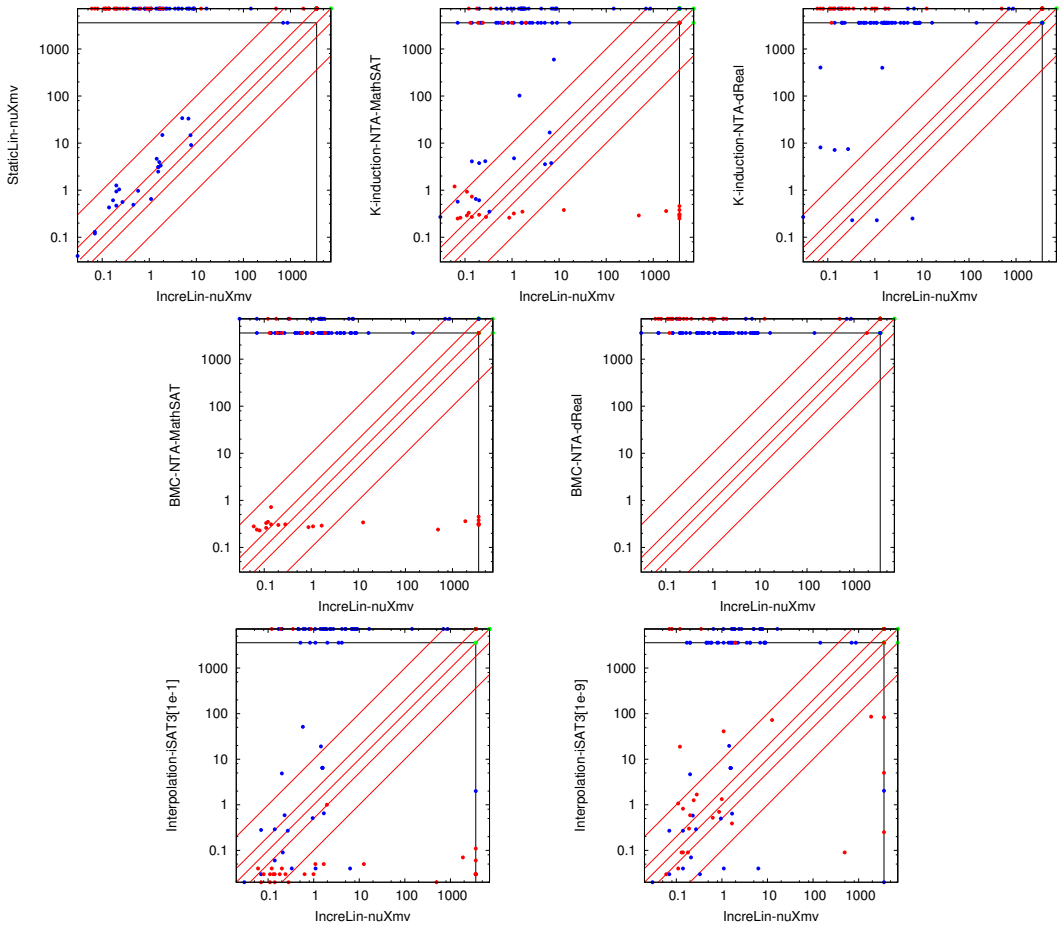

Safe instances are shown in blue, unsafe ones in red. Green dots indicate unknown instances. Diagonal lines mark 2x and 10x performance differences. Points on the inner edges indicate timeouts, those on the outer edges indicate other errors (memory outs or aborts).

Fig. 30. Scatter plots of VMT(NRA) results.




Safe instances are shown in blue, unsafe ones in red. Green dots indicate unknown instances. Diagonal lines mark 2x and 10x performance differences. Points on the inner edges indicate timeouts, those on the outer edges indicate other errors (memory outs or aborts).

Fig. 31. Scatter plots of VMT(NTA) – Unbounded Benchmarks.




Safe instances are shown in blue, unsafe ones in red. Green dots indicate unknown instances. Diagonal lines mark 2x and 10x performance differences. Points on the inner edges indicate timeouts, those on the outer edges indicate other errors (memory outs or aborts).

Fig. 32. Scatter plots of VMT(NTA) – Bounded Benchmarks.