The ksmt calculus is a δ-complete decision procedure for non-linear constraints

DOI:
10.1016/j.tcs.2023.114125

Document Version
Final published version

Theoretical Computer Science 975 (2023) 114125

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

The ksmt calculus is a δ-complete decision procedure for non-linear constraints ☆,☆☆

Franz Brauße [a,*], Konstantin Korovin [a], Margarita V. Korovina [b], Norbert Th. Müller [c]

[a] The University of Manchester, Manchester, UK
[b] A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russia
[c] Abteilung Informatikwissenschaften, Universität Trier, Trier, Germany

ARTICLE INFO

Article history:
Received 31 August 2022
Received in revised form 6 August 2023
Accepted 9 August 2023
Available online 16 August 2023

Keywords:
Computable analysis
Non-linear constraints
Automated reasoning
Satisfiability modulo theories (SMT)

ABSTRACT

ksmt is a CDCL-style calculus for solving non-linear constraints over the real numbers involving polynomials and transcendental functions. In this article we investigate properties of the ksmt calculus and show that it is a δ-complete decision procedure for bounded problems. For that purpose we provide concrete algorithms computing linearisations based on either uniform or local moduli of continuity of non-linear functions. The latter method is called local linearisation and is shown to have desirable properties sufficient for termination and which also allow for more efficient treatment of non-linear constraints. Our methods for constructing linearisations are based on computable analysis, in particular we introduce the Cauchy-compatible compact representation of reals and prove its names to be locally compact, allowing for more efficient computation of local linearisations while maintaining δ-completeness.

1. Introduction

Solving non-linear constraints is important in many applications, including verification of cyber-physical systems, software verification, proof assistants for mathematics [31,25,2,1,16,7]. Hence there has been a number of approaches for solving non-linear constraints, involving symbolic methods [17,28,34,22] as well as numerically inspired ones, in particular for dealing with transcendental functions [14,35], and combinations of symbolic and numeric methods [8,12,13].

In [8] we introduced the ksmt calculus for solving non-linear constraints over a large class of functions including polynomial, exponential and trigonometric functions. The ksmt calculus[1] combines conflict driven constraint learning (CDCL) [26,27,4] over the reals based on conflict resolution [23] with linearisations of non-linear functions using methods from computable analysis [37,30]. An axiom-based form of linearisations for solving non-linear constraints was introduced in [12], tailored specifically to multiplication $x \cdot y$ and Taylor approximations of some analytic functions. Our approach to linearisations is based on computable analysis which is applicable to the larger class of computable functions and allows for more

☆ This article belongs to Section B: Logic, semantics and theory of programming, Edited by Don Sannella.

☆☆ This research was supported by the DFG grant WERA MU 1801/5-1, the EU's Horizon 2020 programme under the Marie Skłodowska-Curie grant agreement No 731143, by the EPSRC grant EP/V000497/1 and by the Intel research grant CG# 56000469.

* Corresponding author.
E-mail addresses: franz.brausse@manchester.ac.uk (F. Brauße), konstantin.korovin@manchester.ac.uk (K. Korovin).

[1] Implementation is available at http://informatik.uni-trier.de/~brausse/ksmt/.

https://doi.org/10.1016/j.tcs.2023.114125

flexibility in constructing concrete linearisations of non-linear functions. It naturally supports exact real arithmetic which avoids limitations of double precision computations caused by rounding errors and instabilities in numerical methods. In particular, satisfiable and unsatisfiable results returned by `ksmt` are exact as required in many applications. This approach also supports implicit representations of functions such as solutions of ordinary and partial differential equations (ODEs and PDEs) [32,9].

It is well known that in the presence of transcendental functions the constraint satisfiability problem is undecidable [33]. However if we only require solutions up to some specified precision δ, then the problem can be solved algorithmically on bounded instances and that is the motivation behind δ-completeness, which was introduced in [14]. In essence a δ-complete procedure decides if a formula is unsatisfiable or a δ-weakening of the formula is satisfiable.

In this paper we investigate theoretical properties of the `ksmt` calculus, and its δ-complete extension δ-`ksmt`. Our main results are as follows:

1. We introduce a notion of ϵ-*full linearisations* and prove that all ϵ-full runs of `ksmt` are terminating on bounded instances.
2. We extend the `ksmt` calculus to the δ-satisfiability setting and prove that δ-`ksmt` is a δ-*complete decision procedure* for bounded instances.
3. We provide an algorithm for computing ϵ-full *local linearisations* and integrate it into δ-`ksmt`. Local linearisations can be used to considerably narrow the search space by taking into account local behaviour of non-linear functions avoiding computationally expensive global analysis.
4. We prove δ-`ksmt` with local linearisations to be δ-complete. For this purpose we introduce the Cauchy-compatible compact (CCC) representation of the reals and prove local compactness of its domain via Tychonoff's theorem as one of the key properties.

In Section 3, we give an overview about the `ksmt` calculus and introduce the notion of ϵ-full linearisation used throughout the rest of the paper. We also show completeness in case of ϵ-full linearisations. Section 4 introduces δ-completeness and the slightly more refined notion of $[\delta]$-completeness together with related concepts. In Section 5 we introduce the δ-`ksmt` adaptation, prove it is correct and δ-complete, and give concrete effective linearisations based on a uniform modulus of continuity. Finally in Section 6, we introduce local linearisations and show that termination is independent of computing uniform moduli of continuity. We prove this using the Cauchy-compatible compact representation of real numbers that we introduce in this section. We conclude in Section 7.

2. Preliminaries

The following conventions are used throughout this paper. By $\|\cdot\|$ we denote the maximum-norm $\|(x_1, x_2, \ldots, x_n)\| = \max\{|x_i| : 1 \le i \le n\}$. When it helps clarity, we write finite and infinite sequences $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_i)_i$ with an arrow above. We use open balls $B(\vec{c}, \epsilon) = \{\vec{x} : \|\vec{x} - \vec{c}\| < \epsilon\} \subseteq \mathbb{R}^n$ for $\vec{c} \in \mathbb{R}^n$ and $\epsilon > 0$ and $\overline{A}$ to denote the closure of the set $A \subseteq \mathbb{R}^n$ in the standard topology induced by the norm. By $\mathbb{Q}_{>0}$ we denote the set $\{q \in \mathbb{Q} : q > 0\}$. For sets X, Y, a function f from X to Y, total or partial, is written as $f : X \rightarrowtail Y$ and in case f is total we write $f : X \to Y$. We use the notion of compactness: a set A is compact iff every open cover of A has a finite subcover. In Euclidean spaces this is equivalent to A being bounded and closed [38]. As is common when considering formulas F in conjunctive normal form (CNF), we will consider F as a set of conjunctively connected clauses, each of which is represented by a set of disjunctively connected atomic constraints.

2.1. Basic notions of computable analysis

Let us recall the notion of computability of functions over real numbers used throughout this paper. A rational number q is an *approximation accurate up to* 2^{-n} of a real number x if $\|q - x\| \le 2^{-n}$. Informally, a function f is *computed* by a function-oracle Turing machine $M_f^?$, where $^?$ is a placeholder for the oracle representing the argument of the function, in the following way. The real argument x is represented by an oracle function $\varphi : \mathbb{N} \to \mathbb{Q}$, for each n returning an n-approximation φ_n of x. For simplicity, we refer to φ by the sequence $(\varphi_n)_n$. When run with an argument $p \in \mathbb{N}$, $M_f^{\varphi}(p)$ computes a rational approximation accurate up to 2^{-p} of $f(x)$ by querying its oracle φ for approximations of x. Let us note that the definition of the oracle machine does not depend on the concrete oracle, i.e., the oracle can be seen as a parameter. In the case only the machine without a concrete oracle is of interest, we write $M_f^?$. We refer to [19] for a precise definition of the model of computation by function-oracle Turing machines which is standard in computable analysis.

Definition 2.1 *([19]).* Consider $\vec{x} \in \mathbb{R}^n$. A *(Cauchy) name* for $\vec{x}$ is a rational sequence $\vec{\varphi} = (\vec{\varphi}_k)_k$ such that $\forall k : \|\vec{\varphi}_k - \vec{x}\| \le 2^{-k}$. A function $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ is *computable* iff there is a function-oracle Turing machine $M_f^?$ such that for all $\vec{x} \in \operatorname{dom} f$ and names $\vec{\varphi}$ for $\vec{x}$, $|M_f^{\vec{\varphi}}(p) - f(\vec{x})| \le 2^{-p}$ holds for all $p \in \mathbb{N}$. Such a Turing machine is called a *realiser* of f.

These concepts can be generalised using so-called representations of topological spaces [37], here we only mention notions relevant for this article. A *representation* ϱ of such a space A is a surjective partial map to A. Given $x \in A$, the

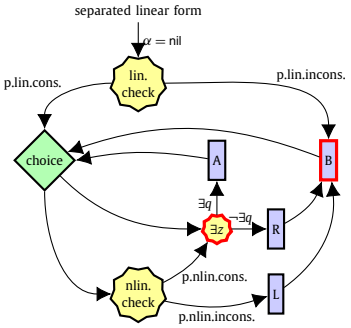


Fig. 1. Core of `ksmt` calculus. Derivations terminate either in node B or $\exists z$.

preimage $\varrho^{-1}(x)$ denotes the set of ϱ-*names* of x. In this article, a name of $x \in \mathbb{R}$ is an infinite sequence of rational approximations to x. When 'name' is used without prefix specifying the concrete representation, throughout this article it is assumed to refer to a Cauchy-name, as defined above. Note that there can be multiple representations of A that differ in the properties they provide. In Section 6 we introduce the Cauchy-compatible compact representation of $\mathbb{R}$ which provides desirable compactness properties.

The above Definition 2.1 is closely related to interval arithmetic with unrestricted precision, but enhanced with the guarantee of convergence and it is equivalent to the notion of computability used in [37]. The class of computable functions contains polynomials and transcendental functions like sin, cos, exp, among others. It is well known [19,37] that this class is closed under composition and that computable functions are continuous. By continuity, a computable function $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ total on a compact $D \subset \mathbb{R}^n$ has a computable *uniform modulus of continuity* $\mu_{f,D} : \mathbb{N} \to \mathbb{N}$ on D [37, Theorem 6.2.7], that is,

$$\forall k \in \mathbb{N}\, \forall \vec{y}, \vec{z} \in D : \|\vec{y} - \vec{z}\| \leq 2^{-\mu_{f,D}(k)} \implies |f(\vec{y}) - f(\vec{z})| \leq 2^{-k}. \tag{2.1}$$

When the compact set D is clear from the context, we will just write μ_f instead. A uniform modulus of continuity of f expresses how changes in the value of f depend on changes of the arguments in a uniform way.

3. The `ksmt` calculus

We first describe the `ksmt` calculus for solving linear and non-linear constraints [8] informally, and subsequently recall the main definitions which we use in this paper. The `ksmt` calculus consists of transition rules, which, for any formula in linear separated form, allow deriving lemmas implied by the formula and, in case of termination, produce a satisfying assignment for the formula or show that it is unsatisfiable by deriving a trivial contradiction. A quantifier-free formula is in separated linear form $\mathcal{L} \cup \mathcal{N}$ if $\mathcal{L}$ is a set of clauses over linear constraints and $\mathcal{N}$ is a set of non-linear atomic constraints; this notion is rigorously defined below.

In the `ksmt` calculus there are four transition rules applied to its states: Assignment refinement (A), Conflict resolution (R), Backjumping (B) and Linearisation (L). The final `ksmt` states are sat and unsat. A non-final `ksmt` state is a triple $(\alpha, \mathcal{L}, \mathcal{N})$ where α is a (partial) assignment of variables to rationals. A `ksmt` derivation starts with an initial state where α is empty and tries to extend this assignment to a solution of $\mathcal{L} \cup \mathcal{N}$ by repeatedly applying the Assignment refinement rule. When such assignment extension is not possible we either obtain a linear conflict which is resolved using the Conflict resolution rule, or a non-linear conflict which is resolved using the Linearisation rule. For an overview of this process of obtaining `ksmt` derivations see Fig. 1.

The main idea behind the Linearisation rule is to approximate the non-linear constraints around the conflict using linear constraints in such a way that the conflict will be shifted into the linear part where it will be resolved using conflict resolution. Application of either of these two rules results in a state containing a clause evaluating to false under the current assignment. This is followed by either application of the Backjumping rule, which undoes assignments or by termination, in which case the original formula is unsat. In this procedure, only the assignment and linear part of the state change and the non-linear part stays fixed.

3.1. Notations

Let $\mathcal{F}_{\text{lin}}$ consist of rational constants, addition, and multiplication by rational constants; $\mathcal{F}_{\text{nl}}$ denotes an arbitrary collection of non-linear computable functions including transcendental functions and polynomials over the reals. We consider the structure $(\mathbb{R}, \langle \mathcal{F}_{\text{lin}} \cup \mathcal{F}_{\text{nl}}, \mathcal{P} \rangle)$ where $\mathcal{P} = \{<, \leq, >, \geq, =, \neq\}$ and a set of variables $V = \{x_1, x_2, \ldots, x_n, \ldots\}$. We will use, possibly with indices, x for variables and q, c, e for rational constants. Define terms, predicates and formulas over V in the standard way. An *atomic linear constraint* is a formula of the form: $c_0 + c_1 x_1 + \ldots + c_n x_n \diamond 0$ where $c_0, \ldots, c_n \in \mathbb{Q}$ and $\diamond \in \mathcal{P}$. A *linear constraint* is a disjunction of atomic linear constraints, also called *(linear) clause*. An *atomic non-linear* constraint is a

formula of the form $f(\vec{x}) \diamond 0$, where $\diamond \in \mathcal{P}$ and f is a composition of linear functions from F_{lin} and at least one computable non-linear function from F_{nl} over variables $\vec{x}$. Since the computable functions are closed under composition, in the following we will treat such a composition as a single computable function. Throughout this paper for every computable real function f we use $M_f^?$ to denote a function-oracle Turing machine computing f. We assume quantifier-free formulas in *separated linear form* [8, Definition 1], that is, $\mathcal{L} \cup \mathcal{N}$ where $\mathcal{L}$ is a set of linear constraints and $\mathcal{N}$ is a set of non-linear atomic constraints. Arbitrary quantifier-free formulas can be transformed equi-satisfiable into separated linear form in polynomial time [8, Lemma 1]. Note that negations of atomic constraints can be eliminated by rewriting the predicate symbol $\diamond$ in the standard way, hence we assume that all constraints occur positively in the formula in separated linear form. Since in separated linear form all non-linear constraints are atomic, we will call them just *non-linear constraints*.

Let $\alpha = (x_{i_1} \mapsto \alpha(x_{i_1}) :: \cdots :: x_{i_n} \mapsto \alpha(x_{i_n}))$ be a *partial variable assignment* to rationals such that all assigned variables x_{i_j} are distinct. Then α can also be seen as a map $\alpha : V \rightharpoonup \mathbb{Q}$. The interpretation $[\![\vec{x}]\!]^\alpha$ of a vector of variables $\vec{x}$ under α is defined in a standard way as component-wise application of α. Define the notation $[\![t]\!]^\alpha$ as evaluation of term t under assignment α, that can be partial, in which case $[\![t]\!]^\alpha$ is treated symbolically. We extend $[\![\cdot]\!]^\alpha$ to predicates, clauses and CNF in the usual way and true, false denote the constants of the Boolean domain. The evaluation $[\![t \diamond 0]\!]^\alpha$ for a predicate $\diamond$ and a term t results in true or false only if all variables in t are assigned by α.

3.2. Arithmetical resolution and linearisation

In order to formally restate the calculus, the notions of linear resolvent and linearisation are essential.

Definition 3.1. A *(generalised) resolvent* $R_{\alpha,\mathcal{L},z}$ on a variable z is a set of linear constraints that do not contain z, are implied by the formula $\mathcal{L}$ and which evaluate to false under the partial assignment α.

As with resolution in the propositional setting,

$$\frac{A \vee p \quad B \vee \neg p}{A \vee B}$$

two clauses over linear constraints can be resolved as well. This gives rise to the inference rule

$$\frac{A \vee (az + b \leq 0) \quad B \vee (-cz + d \leq 0)}{A \vee B \vee (cb + ad \leq 0)}$$

where a, c are positive rational constants and b, d are linear terms. We refer to this rule as *arithmetical resolution* [23,8].

Given a linear formula $\mathcal{L}$ and partial assignment α together with an unassigned variable z, a *linear conflict (for z)* refers to the situation that $[\![\mathcal{L}]\!]^{\alpha::z\mapsto q} = \text{false}$ for any $q \in \mathbb{Q}$. In `ksmt` resolvents are generally obtained through arithmetical resolution; note however, that a single application of this rule does not necessarily result in a resolvent as the clause can still contain the variable z. General methods to obtain resolvents of linear conflicts based on arithmetical resolution exist [8, Section 3.3]. Here, we will just give an example.

Example 3.1. Consider the assignment $\alpha = (x \mapsto 5 :: y \mapsto 7)$ and $\mathcal{L}$ consisting of the three linear clauses $C_1 : (x - y > 4 \vee x + 2z \leq 0)$, $C_2 : (y < 0 \vee z \leq 0 \vee y - z \leq 0)$, and $C_3 : (1 - z \leq 0)$. Under α there is a linear conflict for z in $\mathcal{L}$. In this case the generalised resolvent can be obtained by first applying arithmetical resolution between C_1 and C_2 on z resulting in clause C_4:

$$\frac{C_1 \quad C_2}{C_4 : (x - y > 4 \vee y < 0 \vee z \leq 0 \vee x + 2y \leq 0)}$$

As this clause C_4 still contains z, the second step is another arithmetical resolution of C_4 with C_3 on z resulting in clause C_5:

$$\frac{C_4 \quad C_3}{C_5 : (x - y > 4 \vee y < 0 \vee x + 2y \leq 0 \vee 1 \leq 0)}$$

As C_5 does not contain z anymore, and it evaluates to false under α, $\{C_5\}$ satisfies the conditions of the generalised resolvent $R_{\alpha,\mathcal{L},z}$. Note that $R_{\alpha,\mathcal{L},z}$ is not necessarily unique as neither the set of clauses chosen for arithmetical resolution nor the sequence of resolutions are fixed.

Definition 3.2. Let P be a non-linear constraint and let α be an assignment with $[\![P]\!]^\alpha = \text{false}$. A *linearisation of P at α* is a linear clause C with the properties:

1. $\forall \beta : [\![P]\!]^\beta = \text{true} \implies [\![C]\!]^\beta = \text{true}$, and
2. $[\![C]\!]^\alpha = \text{false}$.

Without loss of generality we can assume that the variables of C are a subset of the variables of P. Let us note that any linear clause C represents the complement of a rational polytope R and we will use both interchangeably. Thus for a rational polytope R, $\vec{x} \notin R$ also stands for a linear clause. In particular, any linearisation excludes a rational polytope containing the conflicting assignment from the search space.

3.3. Transition rules of ksmt

For a formula $\mathcal{L}_0 \cup \mathcal{N}$ in separated linear form, the initial ksmt state is $(\mathsf{nil}, \mathcal{L}_0, \mathcal{N})$. The ksmt calculus consists of the following transition rules from a state $S = (\alpha, \mathcal{L}, \mathcal{N})$ to S':

(A) *Assignment.* $S' = (\alpha :: z \mapsto q, \mathcal{L}, \mathcal{N})$ iff $[\![\mathcal{L}]\!]^{\alpha} \neq \mathsf{false}$ and there is a variable z unassigned in α and $q \in \mathbb{Q}$ with $[\![\mathcal{L}]\!]^{\alpha::z\mapsto q} \neq \mathsf{false}$.
(R) *Resolution.* $S' = (\alpha, \mathcal{L} \cup R_{\alpha,\mathcal{L},z}, \mathcal{N})$ iff $[\![\mathcal{L}]\!]^{\alpha} \neq \mathsf{false}$ and there is a variable z unassigned in α with $\forall q \in \mathbb{Q} : [\![\mathcal{L}]\!]^{\alpha::z\mapsto q} = \mathsf{false}$ and $R_{\alpha,\mathcal{L},z}$ is a resolvent.
(B) *Backjump.* $S' = (\gamma, \mathcal{L}, \mathcal{N})$ iff $[\![\mathcal{L}]\!]^{\alpha} = \mathsf{false}$ and there is a maximal prefix γ of α such that $[\![\mathcal{L}]\!]^{\gamma} \neq \mathsf{false}$.
(L) *Linearisation.* $S' = (\alpha, \mathcal{L} \cup \{L_{\alpha,P}\}, \mathcal{N})$ iff $[\![\mathcal{L}]\!]^{\alpha} \neq \mathsf{false}$, there is P in $\mathcal{N}$ with $[\![P]\!]^{\alpha} = \mathsf{false}$ and $L_{\alpha,P}$ is a linearisation of P at α.
($F^{sat}_{)}$ *Final sat.* $S' = \mathsf{sat}$ if all variables are assigned in α, $[\![\mathcal{L}]\!]^{\alpha} = \mathsf{true}$ and rule (L) is not applicable.
(F^{unsat}) *Final unsat.* $S' = \mathsf{unsat}$ if $[\![\mathcal{L}]\!]^{\mathsf{nil}} = \mathsf{false}$. In other words a trivial contradiction, e.g., $0 > 1$ is in $\mathcal{L}$.

A path (or a run) is a derivation in the ksmt calculus, that is, given an initial state S_0, a derivation denotes a sequence $S_0, S_1, \ldots, S_n, \ldots$ of states where each S_{i+1} is obtained from S_i by application of one of the ksmt rules. A ksmt procedure is an effective (possibly non-deterministic) way to construct a path.

If no transition rule is applicable, the derivation terminates. For clarity, we added the explicit rules (F^{sat}) and (F^{unsat}) which lead to the final states.

The ksmt calculus defined by these rules is sound [8, Lemma 2]: if the final transition is (F^{sat}), then α is a solution to the original formula, or it is (F^{unsat}), in which case a trivial contradiction (e.g., $0 > 1$) was derived and the original formula is unsatisfiable. The calculus also makes progress by reducing the search space [8, Corollary 1]. For clarity, we restate the corresponding lemmas.

Lemma 3.1 *(Soundness). For $n \in \mathbb{N}$ let $(S_i)_{i\leq n}$ be a sequence of ksmt states such that S_{i+1} is derived from $S_i = (\alpha_i, \mathcal{L}_i, \mathcal{N})$ by application of one of the ksmt rules for all $i < n$. The following hold.*

1. *For all $0 < i < n$ and total assignments α: $[\![\mathcal{L}_{i-1} \wedge \mathcal{N}]\!]^{\alpha} = [\![\mathcal{L}_i \wedge \mathcal{N}]\!]^{\alpha}$.*
2. *If no rule is applicable to S_n and S_0 is the initial ksmt state, the following are equivalent:*

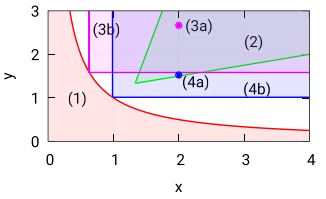
   - *$\mathcal{L}_0 \wedge \mathcal{N}$ is satisfiable.*
   - *α_{n-1} is a solution to $\mathcal{L}_0 \wedge \mathcal{N}$.*
   - *$S_n = \mathsf{sat}$.*
   - *A trivial contradiction is not in $\mathcal{L}_{n-1}$.*

Lemma 3.2 *(Progress). Let $S = (\alpha, \mathcal{L}, \mathcal{N})$ be a ksmt state and let $n \in \mathbb{N}$ be the number of variables in $\mathcal{L} \wedge \mathcal{N}$. Then in any ksmt derivation from S of length greater than $n + 1$ the search space is reduced.*

Example 3.2. An example run of the ksmt calculus is presented in Fig. 2. We start in a state with a non-linear part $\mathcal{N} = \{P\}$, which defines the red area (1) and the linear part $\mathcal{L} = \{(x/4 + 1 \leq y), (y \leq 4 \cdot (x - 1))\}$, shaded and printed in green (2). In this example, the first two assignments are performed consecutively as there are no conflicts yet and result in point (3a). After that, a non-linear conflict appears since $\frac{8}{3} \leq 1/2$ does not hold. The linearisation procedure in this example is given at the top right and invoked, producing the lemma printed in pink. This lemma excludes the pink region (3b) in the graph, which contains the conflicting assignment. This step is followed by an invocation of the backjumping rule and a new assignment to y (4a). Then we continue to successively apply ksmt rules excluding the blue region (4b) around the candidate solution by another linearisation, until we completely separate the red from the green area. This is done by generalised resolutions, first between blue and green lemmas producing the next two clauses printed in black, and then by a resolution between those, thus deriving the final contradiction.

Remark 3.1. In general a derivation may not terminate. The only cause of non-termination is the linearisation rule which adds new linear constraints and can be applied infinitely many times. To see this, observe that ksmt with only the rules $(A), (R), (B)$ corresponds to the conflict resolution calculus which is known to be terminating [23,24]. Thus, in infinite ksmt runs the linearisation rule (L) is applied infinitely often. This argument is used in the proof of Theorem 3.3 below. Let us note that during a run the ksmt calculus neither conflicts nor lemmas (new constraints produced by rules (R) and (L)) can

$$\underbrace{(y \le 1/x)}_{P} \wedge (x/4+1 \le y) \wedge (y \le 4\cdot(x-1)) \wedge \left((x \le \tfrac{12}{19}) \vee (y \le \tfrac{19}{12})\right) \wedge \left((x \le \tfrac{220}{223}) \vee (y \le \tfrac{223}{220})\right) \wedge (\tfrac{4}{3} \le x) \wedge (x \le \tfrac{220}{223}) \wedge (\tfrac{4}{3} \le \tfrac{220}{223})$$



Linearisation of P on conflicts (x, y) at α here:

- choose $d \coloneqq (1/[\![x]\!]^\alpha + [\![y]\!]^\alpha)/2$,
- linearisation lemma $\left(x \le 1/d \vee y \le d\right)$

rule	α	note
(A)	$x \mapsto 2$	
(A)	$x \mapsto 2,\ y \mapsto \frac{8}{3}$	(3a)
(L)	$x \mapsto 2,\ y \mapsto \frac{8}{3}$	(3b)
(B)	$x \mapsto 2$	
(A)	$x \mapsto 2,\ y \mapsto \frac{84}{55}$	(4a)
(L)	$x \mapsto 2,\ y \mapsto \frac{84}{55}$	(4b)
(B)	$x \mapsto 2$	
(R)	$x \mapsto 2$	on y
(B)		
(R)		on x
(F^{unsat})		`unsat`

Fig. 2. An example of an `unsat` run of `ksmt` on initial state $(\alpha, \mathcal{L}, \mathcal{N})$ using interval linearisation [8]. Here, the constraints $\mathcal{L}$ are shaded in green and $\mathcal{N}$ in red. See Example 3.2 for details.

be generated more than once. In fact, any generated linearisation is not implied by the linear part restricted to constraints over variables in α, prior to adding this linearisation.

3.4. Sufficient termination conditions

In this section we will assume that $(\alpha, \mathcal{L}, \mathcal{N})$ is a `ksmt` state obtained by applying `ksmt` inference rules to an initial state. As in [14] we only consider bounded instances. In many applications this is a natural assumption as variables usually range within some (possibly large) bounds. We can assume that these bounds are made explicit as linear constraints in the system.

Definition 3.3. Let F be the formula $\mathcal{L}_0 \wedge \mathcal{N}$ in separated linear form over variables $x_1, \ldots, x_n$ and let B_i be the set defined by the conjunction of all clauses in $\mathcal{L}_0$ univariate in x_i, for $i = 1, \ldots, n$; in particular, if there are no univariate linear constraints over x_i then $B_i = \mathbb{R}$. We call F a *bounded instance* if:

- $D_F \coloneqq \bigtimes_{i=1}^{n} B_i$ is bounded, and
- for each non-linear constraint $P : f(x_{i_1}, \ldots, x_{i_k}) \diamond 0$ in $\mathcal{N}$ with $i_j \in \{1, \ldots, n\}$ for $j \in \{1, \ldots, k\}$ it holds that $\overline{D_{P,F}} \subseteq \operatorname{dom} f$ where $D_{P,F} \coloneqq \bigtimes_{j=1}^{k} B_{i_j}$.

When the formula F is clear from the context, we will drop the index F from $D_{P,F}$.

By this definition, already the linear part of bounded instances explicitly defines a bounded set by univariate constraints. Consequently, the set of solutions of F is bounded as well.

In Theorem 3.3 we show that when we consider bounded instances and restrict linearisations to so-called ϵ-full linearisations, then the procedure terminates. We use this to show that the `ksmt`-based decision procedure we introduce in Section 5 is δ-complete.

Definition 3.4. Let $\epsilon > 0$, P be a non-linear constraint over variables $\vec{x}$ and let α be an assignment of $\vec{x}$. A linearisation C of P at α is called *ϵ-full* iff for all assignments β of $\vec{x}$ with $[\![\vec{x}]\!]^\beta \in B([\![\vec{x}]\!]^\alpha, \epsilon)$, $[\![C]\!]^\beta = \mathsf{false}$.

A `ksmt` run is called ϵ-full for some $\epsilon > 0$, if all but finitely many linearisations in this run are ϵ-full.

The next theorem provides a basis for termination of `ksmt`-based decision procedures for satisfiability.

Theorem 3.3. *Let $\epsilon > 0$. On bounded instances, ϵ-full `ksmt` runs are terminating.*

Proof. Let $F : \mathcal{L}_0 \wedge \mathcal{N}$ be a bounded instance and $\epsilon > 0$. Towards a contradiction assume there is an infinite ϵ-full derivation $(\alpha_0, \mathcal{L}_0, \mathcal{N}), \ldots, (\alpha_n, \mathcal{L}_n, \mathcal{N}), \ldots$ in the `ksmt` calculus. Then, by definition of the transition rules, $\mathcal{L}_k \subseteq \mathcal{L}_l$ for all k, l with $0 \le k \le l$. According to Remark 3.1 in any infinite derivation the linearisation rule must be applied infinitely many times. During any run of `ksmt` the set of non-linear constraints $\mathcal{N}$ is fixed and therefore there is a non-linear constraint P in $\mathcal{N}$ over variables $\vec{x}$ to which linearisation is applied infinitely often. Let $(\alpha_{i_1}, \mathcal{L}_{i_1}, \mathcal{N}), \ldots, (\alpha_{i_n}, \mathcal{L}_{i_n}, \mathcal{N}), \ldots$ be a corresponding

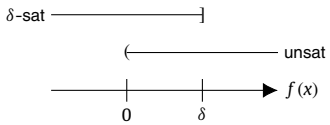


Fig. 3. The overlapping cases in the δ-SMT problem $f(x) \leq 0$.

subsequence in the derivation such that $C_{i_1} \in \mathcal{L}_{i_1+1}, \ldots, C_{i_n} \in \mathcal{L}_{i_n+1}, \ldots$ are ϵ-full linearisations of P. Consider two different linearisation steps $k, \ell \in \{i_j : j \in \mathbb{N}\}$ in the derivation where $k < \ell$. By the precondition of rule (L) applied in step ℓ we have $[\![\mathcal{L}_\ell]\!]^{\alpha_\ell} \neq \mathsf{false}$. In particular the linearisation $C_k \in \mathcal{L}_{k+1} \subseteq \mathcal{L}_\ell$ of P constructed in step k does not evaluate to false under α_ℓ. Since the set of variables in C_k is a subset of those in P, $[\![C_k]\!]^{\alpha_\ell} \neq \mathsf{false}$ implies $[\![C_k]\!]^{\alpha_\ell} = \mathsf{true}$. By assumption, the linearisation C_k is ϵ-full, thus from Definition 3.4 it follows that $[\![\vec{x}]\!]^{\alpha_\ell} \notin B([\![\vec{x}]\!]^{\alpha_k}, \epsilon)$. Therefore the distance between $[\![\vec{x}]\!]^{\alpha_k}$ and $[\![\vec{x}]\!]^{\alpha_\ell}$ is at least ϵ. However, every conflict satisfies the variable bounds defining D_F, so there could be only finitely many conflicts with pairwise distance at least ϵ. This contradicts the above. □

Concrete algorithms to compute ϵ-full linearisations are presented in Sections 5 and 6.

4. δ-decidability

In the last section, we proved termination of the `ksmt` calculus on bounded instances when linearisations are ϵ-full. Let us now investigate how ϵ-full linearisations of constraints involving non-linear computable functions can be constructed. To that end, we assume that all non-linear functions are defined on the closure of the bounded space D_F defined by the bounded instance F.

So far we described an approach which gives exact results but at the same time is necessarily incomplete due to undecidability of non-linear constraints in general. On the other hand, non-linear constraints usually can be approximated using numerical methods allowing to obtain approximate solutions to the problem. This gives rise to the bounded δ-SMT problem [14] which allows an overlap between the properties δ-sat and unsat of formulas as illustrated by Fig. 3. It is precisely this overlap that enables δ-decidability of bounded instances.

Let us recall the notion of δ-decidability, adapted from [14].

Definition 4.1. Let F be a formula in separated linear form and let $\delta \in \mathbb{Q}_{>0}$. We inductively define the δ-weakening F_δ of F.

- If F is linear, let $F_\delta := F$.
- If F is a non-linear constraint $f(\vec{x}) \diamond 0$, let

$$F_\delta := \begin{cases} f(\vec{x}) - \delta \diamond 0, & \text{if } \diamond \in \{<, \leq\} \\ f(\vec{x}) + \delta \diamond 0, & \text{if } \diamond \in \{>, \geq\} \\ |f(\vec{x})| - \delta \leq 0, & \text{if } \diamond \in \{=\} \\ (f(\vec{x}) < 0 \vee f(\vec{x}) > 0)_\delta, & \text{if } \diamond \in \{\neq\}. \end{cases}$$

- Otherwise, F is $A \circ B$ with $\circ \in \{\wedge, \vee\}$. Let $F_\delta := (A_\delta \circ B_\delta)$.

δ-deciding F designates computing

$$\begin{cases} \mathsf{unsat}, & \text{if } [\![F]\!]^\alpha = \mathsf{false} \text{ for all } \alpha \\ \delta\text{-sat}, & \text{if } [\![F_\delta]\!]^\alpha = \mathsf{true} \text{ for some } \alpha. \end{cases}$$

In cases when both answers are valid, the algorithm may output any.

An assignment α with $[\![F_\delta]\!]^\alpha = \mathsf{true}$ we call a *δ-satisfying assignment* for F.

For non-linear constraints P this definition of the δ-weakening P_δ corresponds exactly to the notion of δ-weakening $P^{-\delta}$ used in the introduction of δ-decidability [15, Definition 4.1].

Remark 4.1. The δ-weakening of a non-linear constraint $f(\vec{x}) \neq 0$ is a tautology.

We now consider the problem of δ-deciding quantifier-free formulas in separated linear form. The notion of δ-decidability is slightly stronger than in [14] in the sense that we do not weaken linear constraints. Consider a formula F in separated linear form. As before, we assume variables $\vec{x}$ to be bounded by linear constraints $\vec{x} \in D_F$. We additionally assume that for all non-linear constraints $P : f(\vec{x}) \diamond 0$ in $\mathcal{N}$, f is defined on $\overline{D_P}$ and, in order to simplify the presentation, throughout the rest of paper we will assume only the predicates $\diamond \in \{>, \geq\}$ are part of formulas, since the remaining ones $<, \leq, =$ can

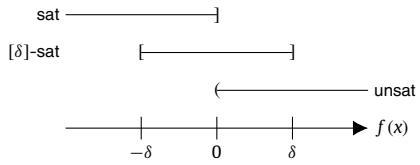


Fig. 4. The overlapping cases in the $[\delta]$-decision problem $f(x) \leq 0$.

F	$F_{-\delta}$	$[\delta]$-decision of F
unsat	unsat	unsat
δ-sat α	unsat	$[\delta]$-sat α
–	δ-sat β	sat β

Fig. 5. $[\delta]$-decision of F obtained by combining results of δ-deciding both, formula F and its δ-strengthening $F_{-\delta}$. Here, α, β are final assignments. Note that in case $F_{-\delta}$ is δ-sat with assignment β, Lemma 4.1 shows that β also satisfies F.

easily be expressed by the former using simple arithmetic transformations, and by Remark 4.1 predicates $\neq$ are irrelevant for δ-deciding formulas.

An algorithm is *δ-complete*, if it δ-decides bounded instances [14].

4.1. The $[\delta]$-decision problem

The notion of δ-deciding a formula F introduced above is asymmetrical in the sense that δ-sat enlarges the set of satisfiable formulas while the unsatisfiable ones remain the same. The δ-sat result also is weaker than a sat result. However, in this section we show how the asymmetry can be avoided and a large part of the stronger sat result can be obtained given any δ-complete decision procedure. The problem solved in this way we call the *$[\delta]$-decision problem* and is summarised in Fig. 4.

In order to define this problem, we recall the notion of a δ-strengthening of a formula F adapted to our setting [15, Definition 4.1].

Definition 4.2. Let $\delta > 0$. The *δ-strengthening* $F_{-\delta}$ of a formula F in separated linear form is a obtained by the substitution of $-\delta$ for δ in Definition 4.1.

Remark 4.2. Similar to the δ-weakening of non-linear disequality constraints, the δ-strengthening of non-linear equalities is trivially a contradiction.

Now we can state the $[\delta]$-decision problem: Compute one of

$$\begin{cases} \text{sat}, & \text{if } F \text{ is satisfiable} \\ [\delta]\text{-sat}, & \text{if } F_\delta \text{ is satisfiable and } F_{-\delta} \text{ is not} \\ \text{unsat}, & \text{if } F \text{ is unsatisfiable} \end{cases}$$

The below properties follow directly from the definition.

Lemma 4.1. *Let F be formula and let $\delta > 0$. The following hold:*

1. *F is equivalent to $(F_{-\delta})_\delta$.*
2. *$F_{-\delta}$ implies F.*

Given a δ-complete decision procedure for some $\delta > 0$, we can approach the $[\delta]$-decision problem as follows. First, δ-decide F and in case it is unsat, return unsat as the result. Otherwise F is δ-sat with assignment α. Then, secondly, δ-decide the δ-strengthening of F. If $F_{-\delta}$ is δ-sat with assignment β, return sat with assignment β, otherwise return $[\delta]$-sat with assignment α. Fig. 5 provides a summary of this construction and additionally mentions the contradictory combination of F being unsat and $F_{-\delta}$ being δ-sat, which in turn implies that F is sat.

Corollary 4.2. *Let $\delta > 0$ and F be a formula in separated linear form. The $[\delta]$-decision problem can be solved by δ-deciding both, F and $F_{-\delta}$, separately.*

5. `δ-ksmt`

Since δ-decidability as introduced above adapts the condition when a formula is considered to be satisfied to δ-sat, this condition has to be reflected in the calculus, which in this section we show solves the bounded δ-SMT problem. Adding the

Rules	$(\alpha, \mathcal{L}, \mathcal{N}) \Rightarrow *$ where $*$ is:	applicability conditions
(A) assign	$(\alpha :: z \mapsto q, \mathcal{L}, \mathcal{N})$	z unassigned, $q \in \mathbb{Q}$ and no linear conflict
(R) resolve	$(\alpha, \mathcal{L} \cup R, \mathcal{N})$	resolvent R excludes the linear conflict
(B) backjump	$(\gamma, \mathcal{L}, \mathcal{N})$	to γ maximal conflict-free prefix of α
(L) linearise	$(\alpha, \mathcal{L} \cup L, \mathcal{N})$	linearisation L excludes the non-linear conflict
(F^{sat})	sat	all variables are assigned and neither a linear nor non-linear conflict exists
(F^{unsat})	unsat	$[\![\mathcal{L}]\!]^{\text{nil}} = \text{false}$
(F^{sat}_{δ})	δ-sat	all variables are assigned and $[\![\mathcal{L} \wedge \mathcal{N}_\delta]\!]^{\alpha} = \text{true}$

Fig. 6. Summary of the rules of the δ-ksmt calculus.

following rule (F^{sat}_{δ}) together with the new final state δ-sat to ksmt relaxes the termination conditions and turns it into the extended calculus we call δ-ksmt. A summary of its rules is provided in Fig. 6.

(F^{sat}_{δ}) *Final δ-sat.* If $(\alpha, \mathcal{L}, \mathcal{N})$ is a δ-ksmt state where α is a total assignment and $[\![\mathcal{L} \wedge \mathcal{N}_\delta]\!]^{\alpha} = \text{true}$, transition to the δ-sat state.

The applicability conditions on the rules (L) and (F^{sat}_{δ}) individually are not decidable [33,6], however, when we compute them simultaneously, we can effectively apply one of these rules, as we will show in Lemma 5.4. In combination with ϵ-fullness of the computed linearisations (Lemma 5.5), this leads to Theorem 5.6, showing that δ-ksmt is a δ-complete decision procedure.

Let us note that if we assume $\delta = 0$ then δ-ksmt would just reduce to ksmt as (F^{sat}) and (F^{sat}_{δ}) become indistinguishable, but in the following we always assume $\delta > 0$.

In the following sub-section, we prove that terminating derivations of the δ-ksmt calculus lead to correct results. Then, in Section 5.2, we present a concrete algorithm for applying rules (L) and (F^{sat}_{δ}) and show its linearisations to be ϵ-full, which is sufficient to ensure termination, as shown in Theorem 3.3. These properties lead to a δ-complete decision procedure. In Section 6 we develop a more practical algorithm for ϵ-full linearisations that does not require computing a uniform modulus of continuity.

5.1. Soundness

In this section we show soundness of the δ-ksmt calculus, that is, validity of its derivations. In particular, this implies that derivability of the final states unsat, δ-sat and sat directly corresponds to unsatisfiability, δ-satisfiability and satisfiability of the original formula, respectively.

Lemma 5.1. *For all δ-ksmt derivations of $S' = (\alpha', \mathcal{L}', \mathcal{N})$ from a state $S = (\alpha, \mathcal{L}, \mathcal{N})$ and for all total assignments β, $[\![\mathcal{L} \wedge \mathcal{N}]\!]^{\beta} = [\![\mathcal{L}' \wedge \mathcal{N}]\!]^{\beta}$.*

Proof. Let β be a total assignment of the variables in $\mathcal{L} \wedge \mathcal{N}$. Since the set of variables remains unchanged by δ-ksmt derivations, β is a total assignment for $\mathcal{L}' \wedge \mathcal{N}$ as well. Let $S' = (\alpha', \mathcal{L}', \mathcal{N})$ be derived from $S = (\alpha, \mathcal{L}, \mathcal{N})$ by a single application of one of δ-ksmt rules. By the structure of S', its derivation was not caused by neither $(F^{unsat}), (F^{sat})$ or (F^{sat}_{δ}). For rules (A) and (B) there is nothing to show since $\mathcal{L} = \mathcal{L}'$. If (R) caused $S \mapsto S'$, the claim holds by soundness of arithmetical resolution. Otherwise (L) caused $S \mapsto S'$ in which case the direction $\Rightarrow$ follows from the definition of a linearisation (condition 1 in Definition 3.2) while the other direction trivially holds since $\mathcal{L} \subseteq \mathcal{L}'$.

The condition on derivations of arbitrary lengths then follows by induction. □

Lemma 5.2. *Let $\delta \in \mathbb{Q}_{>0}$. Consider a formula $G = \mathcal{L}_0 \wedge \mathcal{N}$ in separated linear form and let $S = (\alpha, \mathcal{L}, \mathcal{N})$ be a δ-ksmt state derivable from the initial state $S_0 = (\mathit{nil}, \mathcal{L}_0, \mathcal{N})$. The following hold.*

- *If rule (F^{unsat}) is applicable to S then G is unsatisfiable.*
- *If rule (F^{sat}_{δ}) is applicable to S then α is a δ-satisfying assignment for G, hence G is δ-satisfiable.*
- *If rule (F^{sat}) is applicable to S then α is a satisfying assignment for G, hence G is satisfiable.*

Proof. Let formula G and states S_0, S be as in the premise. As S is not final in δ-ksmt, only ksmt rules have been applied in deriving it. The statements for rules (F^{unsat}) and (F^{sat}) thus hold by soundness of ksmt [8, Lemma 2].

Assume (F^{sat}_{δ}) is applicable to S, that is, $[\![\mathcal{L} \wedge \mathcal{N}_\delta]\!]^{\alpha}$ is true. Since $\mathcal{L}$ contains all constraints from $\mathcal{L}_0$, α also satisfies $\mathcal{L}_0$. Hence, α satisfies $\mathcal{L}_0 \wedge \mathcal{N}_\delta$, which, according to Definition 4.1, equals G_δ. Therefore α is a δ-satisfying assignment for G. □

Since the only way to derive one of the final states unsat, δ-sat and sat from the initial state in δ-ksmt is by application of the rule (F^{unsat}), (F^{sat}_{δ}) and (F^{sat}), respectively, as corollary of Lemmas 5.1 and 5.2 we obtain validity of the δ-ksmt derivations, thus soundness of the calculus.

Theorem 5.3 *(Soundness). Let $\delta \in \mathbb{Q}_{>0}$. The δ-ksmt calculus is sound.*

5.2. δ-completeness

We proceed by introducing Algorithm 1 computing linearisations and deciding which of the rules (F^{sat}_{δ}) and (L) to apply. These linearisations are then shown to be ϵ-full for some $\epsilon > 0$ depending on the bounded instance. By Theorem 3.3, this property implies termination, showing that δ-ksmt is a δ-complete decision procedure.

Given a non-final δ-ksmt state, the function NLINSTEP_{δ} in Algorithm 1 computes a δ-ksmt state derivable from it by application of (F^{sat}_{δ}) or (L). This is done by evaluating the non-linear functions and adding a linearisation C based on their uniform moduli of continuity as needed. To simplify the algorithm, it assumes total assignments as input. It is possible to relax this requirement to partial assignments, e.g., by returning a state obtained by invoking rules (A) or (R) instead of returning δ-sat.

Algorithm 1 (NLINSTEP_{δ}) Algorithm computing a δ-ksmt derivation according to either rule (L) or (F^{sat}_{δ}) from a state $(\alpha, \mathcal{L}, \mathcal{N})$ where α is total. The function f in non-linear constraint $P \in \mathcal{N}$ is assumed to be computed by machine $M^{?}_{f}$ and $\mu_f = \mu_{f,\overline{D_{P,F}}}$ to be a computable uniform modulus of continuity of f.

function $\text{LINEARISE}_{\delta}(f, \vec{x}, \diamond, \alpha)$
  pick $p \in \mathbb{N}$ with $2^{-p} \leq \delta/4$
  $\varphi \leftarrow (n \mapsto [\![\vec{x}]\!]^{\alpha})$
  $\epsilon \leftarrow 2^{-\mu_f(p)}$
  $y \leftarrow M^{\varphi}_{f}(p)$
  if $y \diamond -\delta/2$ **then**
    return None
  end if
  $C \leftarrow (\vec{x} \notin B([\![\vec{x}]\!]^{\alpha}, \epsilon))$
  return C ▷ linearisation at α
end function

function $\text{NLINSTEP}_{\delta}(\alpha, \mathcal{L}, \mathcal{N})$
  for each $(f(\vec{x}) \diamond 0)$ **in** $\mathcal{N}$ **do**
    $C \leftarrow \text{LINEARISE}_{\delta}(f, \vec{x}, \diamond, \alpha)$
    if $C \neq$ None **then** ▷ (L)
      return $(\alpha, \mathcal{L} \cup \{C\}, \mathcal{N})$
    end if
  end for
  return δ-sat ▷ (F^{sat}_{δ})
end function

Lemma 5.4. *Let $\delta \in \mathbb{Q}_{>0}$, $\mathcal{L} \wedge \mathcal{N}$ be a bounded instance and let $S = (\alpha, \mathcal{L}, \mathcal{N})$ be a δ-ksmt state where α is total and $[\![\mathcal{L}]\!]^{\alpha} = true$. Then $\text{NLINSTEP}_{\delta}(\alpha, \mathcal{L}, \mathcal{N})$ computes a state derivable by application of either (L) or (F^{sat}_{δ}) to S.*

Proof. In the proof we will use notions from computable analysis, as defined in Section 2.1. Let $(\alpha, \mathcal{L}, \mathcal{N})$ be a state as in the premise, $F = \mathcal{L} \wedge \mathcal{N}$, and let $P : f(\vec{x}) \diamond 0$ be a non-linear constraint in $\mathcal{N}$. Let $M^{?}_{f}$ compute f as in Algorithm 1. The algorithm computes a rational approximation $y = M^{([\![\vec{x}]\!]^{\alpha})_i}_{f}(p)$ of $f([\![\vec{x}]\!]^{\alpha})$ where $p \in \mathbb{N}$ with $2^{-p} \leq \delta/4$. $[\![\mathcal{L}]\!]^{\alpha} = \text{true}$ implies $[\![\vec{x}]\!]^{\alpha} \in D_{P,F} \subseteq \text{dom}\, f$, thus the computation of y terminates. Since $M^{?}_{f}$ computes f, y is accurate up to $2^{-p} \leq \delta/4$, that is, $y \in [f([\![\vec{x}]\!]^{\alpha}) \pm \delta/4]$. By assumption $\diamond \in \{>, \geq\}$, thus

1. $y \diamond -\delta/2$ implies $f([\![\vec{x}]\!]^{\alpha}) \diamond -\delta$, which is equivalent to $[\![P_{\delta}]\!]^{\alpha} = \text{true}$, and
2. $\neg(y \diamond -\delta/2)$ implies $\neg(f([\![\vec{x}]\!]^{\alpha}) \diamond -\delta/2 + \delta/4)$, which in turn implies $[\![P]\!]^{\alpha} = \text{false}$ and the applicability of rule (L).

For Item 1 no linearisation is necessary and indeed the algorithm does not linearise P. Otherwise (Item 2), it adds the linearisation $C : (\vec{x} \notin B([\![\vec{x}]\!]^{\alpha}, \epsilon))$ to the linear clauses. Since $[\![\vec{x}]\!]^{\alpha} \in D_{P,F}$ by Eq. (2.1) we obtain that $0 \notin B(f(\vec{z}), \delta/4)$ holds, implying $\neg(f(\vec{z}) \diamond 0)$, for all $\vec{z} \in B([\![\vec{x}]\!]^{\alpha}, \epsilon) \cap \overline{D_{P,F}}$. Hence, the formula C is a linearisation of P at α.

In case $\text{NLINSTEP}_{\delta}(\alpha, \mathcal{L}, \mathcal{N})$ returns δ-sat, the premise of Item 1 holds for every non-linear constraint in $\mathcal{N}$, that is, $[\![\mathcal{N}_{\delta}]\!]^{\alpha} = \text{true}$. By assumption $[\![\mathcal{L}]\!]^{\alpha} = \text{true}$, hence the application of the (F^{sat}_{δ}) rule deriving δ-sat is possible in δ-ksmt. □

Lemma 5.5. *For any bounded instance $\mathcal{L}_0 \wedge \mathcal{N}$ there is a computable $\epsilon \in \mathbb{Q}_{>0}$ such that any δ-ksmt run starting in $(nil, \mathcal{L}_0, \mathcal{N})$, where applications of (L) and (F^{sat}_{δ}) are performed by NLINSTEP_{δ}, is ϵ-full.*

Proof. Let $P : f(\vec{x}) \diamond 0$ be a non-linear constraint in $\mathcal{N}$. Since $F = \mathcal{L}_0 \wedge \mathcal{N}$ is a bounded instance, $D_{P,F} \subseteq \mathbb{R}^n$ is also bounded. Let $\epsilon_P := 2^{-\mu_f(p)}$ where $p \in \mathbb{N}$ with $2^{-p} \leq \delta/4$ as in Algorithm 1. As μ_f is a uniform modulus of continuity of f valid on $\overline{D_{P,F}}$, the inequalities in the following construction hold on the whole domain $\overline{D_{P,F}}$ of f and do not depend on the concrete assignment α where the linearisation is performed. Since $\log_2$ and μ_f are computable, so are p and ϵ_P. There are

finitely many non-linear constraints P in $\mathcal{N}$, therefore the linearisations the algorithm $\textsc{nlinStep}_\delta$ computes are ϵ-full with $\epsilon = \min\{\epsilon_P : P \text{ in } \mathcal{N}\} > 0$. □

We call δ-ksmt derivations when linearisations are computed using Algorithm 1 δ-ksmt with full-box linearisations, or δ-*ksmt-fb* for short. As the runs computed by it are ϵ-full for $\epsilon > 0$, by Theorem 3.3 they terminate.

Theorem 5.6. *δ-ksmt-fb is a δ-complete decision procedure.*

Proof. δ-ksmt-fb is sound (Theorem 5.3) and terminates on bounded instances (Theorem 3.3 and Lemma 5.5). □

Remark 5.1. The properties function $\textsc{linearise}_\delta(f, \vec{x}, \diamond, \alpha)$ in Algorithm 1 needs to satisfy can be summed up as follows. Given $\delta > 0$, a non-linear constraint $P : f(\vec{x}) \diamond 0$ and α with $[\![\vec{x}]\!]^\alpha \in \operatorname{dom} f$, $\textsc{linearise}_\delta(f, \vec{x}, \diamond, \alpha)$ computes one of

- None, if $[\![P_\delta]\!]^\alpha = \mathsf{true}$, or
- a linearisation C of P at α, if $[\![P]\!]^\alpha = \mathsf{false}$.

In case there is $\epsilon > 0$ such that all computed linearisations are ϵ-full, so are the corresponding δ-ksmt runs and thus they terminate on bounded instances.

In the following section, we will introduce another kind of linearisation procedure based on effective local moduli of continuity satisfying these conditions and prove existence of a lower bound $\epsilon > 0$ on the ϵ-fullness for bounded instances.

6. Local ϵ-full linearisations

In practice, when the algorithm computing ϵ-full linearisations described in the previous section is going to be implemented, the question arises of how to get a good uniform modulus of continuity μ_f for a computable function f. Depending on how f is given, there may be several ways of computing it. Implementations of exact real arithmetic, e.g., iRRAM [30], AERN2 [20], Ariadne [2], CDAR [3], are usually based on the formalism of function-oracle Turing machines (see Definition 2.1) which allow to compute with representations of computable functions [11] including implicit representations of functions as solutions of ODEs/PDEs [32,9]. If f is only available as a function-oracle Turing machine $M_f^?$ computing it, a modulus μ_f valid on a compact domain can be computed, however, in general this is not possible without exploring the behaviour of the function on the whole domain, which in many cases is computationally expensive. Moreover, since μ_f is uniform, $\mu_f(n)$ is constant on the entire domain, independent of the actual assignment α determining where f is evaluated. Yet, computable functions admit *local* moduli of continuity that additionally depend on the concrete point in their domain. In most cases these would provide linearisations with ϵ larger than that determined by μ_f leading to larger regions being excluded, ultimately resulting in fewer linearisation steps and general speed-up. In many cases local moduli of continuity can be obtained, e.g., from an explicit representation of the function or the concrete approximation as a term. In that case, this local modulus of f can be used directly instead of μ_f in Algorithm 1.

However, even if there is no knowledge about the function except its computability by a machine, a local modulus of continuity can still be obtained. In fact, most of the prominent implementations of computable analysis evaluate user-defined functions by searching for a local modulus of continuity at the point of evaluation. This method has been formalised as a representation of continuous functions [11] in the second-order complexity framework of Kawamura and Cook [18]. Though unlike the representation they suggest for $C([0,1])$, implementations typically choose to not provide polynomial-time access to a uniform modulus of continuity. This is due to the fact that no representation of $C([0,1])$ allows for both, polynomial-time computability of a uniform modulus and fast computation of the evaluation operator [11, Theorem 2.4]. Nonetheless, as shown in this section, even from algorithms adhering to the Cauchy representation (as implemented in most systems for exact real arithmetic) ϵ-full linearisations can be obtained, with two benefits: (a) local moduli and thus ϵ can be bounded globally, which allows us to prove termination of δ-ksmt even when no additional knowledge about the function is available, and (b) only a single run of the machine is needed per linearisation.Indeed, machines producing finite approximations of $f(x)$ from finite approximations of x internally have to compute some form of local modulus to guarantee correctness. In this section, we explore this approach of obtaining local linearisations covering a larger part of the function's domain.

In order to guarantee a positive bound on the local modulus of continuity extracted directly from the run of the machine $M_f^?$ computing f, it is necessary to employ a restriction on the names of real numbers $M_f^?$ computes on. The set of names should in a very precise sense be "small", i.e., it has to be compact. The very general notion of names used in Definition 2.1 is too broad to satisfy this criterion since the space of rational approximations is not even locally compact. Here, we present an approach using practical names of real numbers as sequences of dyadic rationals of lengths restricted by accuracy. For that purpose, we introduce another representation of $\mathbb{R}$, that is, the surjective mapping $\xi : \mathbb{D}_\omega \rightarrowtail \mathbb{R}$ called *Cauchy-compatible compact representation*. This terminology is motivated by Lemma 6.1 and Theorem 6.4 stated later. Here, $\mathbb{D}_\omega$ denotes the set of infinite sequences $(\varphi_k)_k$ of dyadic rationals where for each k the denominator of φ_k is bounded by $2^{-(k+1)}$, see

Definition 6.1. If φ has a limit (in $\mathbb{R}$), we write $\lim\varphi$. Algorithms operating on such sequences of approximations can be implemented using interval arithmetic or generalisations thereof [30,10,21]. The Cauchy-compatible compact representation is also close to implementations as for efficiency reasons they tend to cut unnecessary precision from approximations.

Definition 6.1. We introduce a representation of reals based on sequences of dyadics of restricted lengths, which we call *Cauchy-compatible compact representation*, or CCC for short, as follows.

- For $k \in \omega$ let $\mathbb{D}_k \coloneqq \mathbb{Z} \cdot 2^{-(k+1)} = \{m/2^{k+1} : m \in \mathbb{Z}\} \subset \mathbb{Q}$ and let $\mathbb{D}_\omega \coloneqq \bigtimes_{k\in\omega} \mathbb{D}_k$ be the set of all sequences $(\varphi_k)_k$ with $\varphi_k \in \mathbb{D}_k$ for all $k \in \omega$. By default, $\mathbb{D}_\omega$ is endowed with the Baire space topology, which corresponds to that induced by the metric
$$d : (\varphi, \psi) \mapsto \begin{cases} 0 & \text{if } \varphi = \psi \\ 2^{-n} & \text{if } \varphi_n \neq \psi_n \text{ with } n \in \omega \text{ minimal} \end{cases}$$
- Define $\xi : \mathbb{D}_\omega \rightarrowtail \mathbb{R}$ as the partial function mapping $\varphi \in \mathbb{D}_\omega$ to $\lim\varphi$ iff $\forall i, j : |\varphi_i - \varphi_{i+j}| \leq 2^{-(i+1)}$. Any $\varphi \in \xi^{-1}(x)$ is called a *ξ-name* of $x \in \mathbb{R}$. We also refer to ξ as the *Cauchy-compatible compact representation*.
- The representation $\rho : (x_k)_k \mapsto x$ mapping names $(x_k)_k$ of $x \in \mathbb{R}$ to x as per Definition 2.1 is called *Cauchy representation*.

Using a standard product construction we can easily generalise the notion of ξ-names to ξ^n-names of $\mathbb{R}^n$. When clear from the context, we will drop n and just write ξ to denote the corresponding generalised representation $\mathbb{D}_\omega^n \rightarrowtail \mathbb{R}^n$. Note that in the definition of $\mathbb{D}_k$ the denominators are set to 2^{k+1} in order to align ξ-names with Cauchy-names, see Lemma 6.1.

Computable equivalence between two representations not only implies that there are continuous maps between them but also that names can computably be transformed [37]. Since the Cauchy representation itself is continuous [5] we derive continuity of ξ, which is used below to show compactness of preimages $\xi^{-1}(X)$ of compact sets $X \subseteq \mathbb{R}$ under ξ.

Lemma 6.1. *The following properties hold for the Cauchy-compatible compact representation ξ.*

1. *ξ is a representation of $\mathbb{R}^n$.*
2. *Any ξ-name of $\vec{x} \in \mathbb{R}^n$ is a Cauchy-name of $\vec{x}$.*
3. *ξ is computably equivalent to the Cauchy representation.*
4. *ξ is continuous.*
5. *The domain of ξ is closed.*

Proof. 1. We only need to show that $\xi : \mathbb{D}_\omega \rightarrowtail \mathbb{R}^n$ is surjective. This property is implied by (computable) equivalence to the Cauchy representation, see Item 3.

2. We prove that for any $\vec{x} \in \mathbb{R}^n$ and ξ-name φ of $\vec{x}$, $\forall k : |\varphi_k - \vec{x}| \leq 2^{-k}$ holds. For simplicity we assume a dimension of $n = 1$. The general case can be proved similarly.

   Let $x \in \mathbb{R}$ and φ be a ξ-name of x and let $k \in \omega$. By construction $x = \lim\varphi$, hence there is $n_0 \in \omega$ such that for every $n \geq n_0$ the bound $|\varphi_n - x| < 2^{-(k+1)}$ holds. If $n_0 \leq k$, the previous bound already gives the required property. Otherwise, $n_0 > k$, and $|\varphi_k - x| \leq |\varphi_k - \varphi_{n_0}| + |\varphi_{n_0} - x|$ holds. Since $\varphi \in \operatorname{dom}\xi$, the first summand is bounded by $2^{-(\min(k,n_0)+1)} = 2^{-(k+1)}$. By the property above, so is the second. Ergo $|\varphi_k - x| \leq 2^{-k}$.

3. For simplicity we consider the case of dimension 1. The general case can be proved similarly.

   $\Rightarrow$) Let ψ be a ξ-name of $x \in \mathbb{R}$. By Item 2, ψ is a Cauchy-name of x.

   $\Leftarrow$) Given $\varphi \in \operatorname{dom}\rho$ and $n \in \omega$. Compute $\psi_n \coloneqq \lfloor \varphi_{n+4} \cdot 2^{n+1} \rceil / 2^{n+1} \in \mathbb{D}_n$ where $\lfloor\cdot\rceil : \mathbb{Q} \to \mathbb{Z}$ is a computable rounding operation with $|\lfloor q \rceil - q| \leq 1/2$ for all $q \in \mathbb{Q}$.

   Then with $x \coloneqq \lim\varphi$:
$$\begin{aligned} |\psi_n - x| &= |\lfloor \varphi_{n+4} \cdot 2^{n+1} \rceil / 2^{n+1} - x| \\ &\leq |\lfloor \varphi_{n+4} \cdot 2^{n+1} \rceil - \varphi_{n+4} \cdot 2^{n+1}| / 2^{n+1} + |\varphi_{n+4} - x| \\ &\leq 2^{-(n+2)} + 2^{-(n+4)} \end{aligned}$$
   We show $\psi \coloneqq (\psi_n)_n$ is a ξ-name of x. Let $n, k \in \omega$ with $k > 0$.
$$\begin{aligned} |\psi_n - \psi_{n+k}| &\leq |\psi_n - x| + |\psi_{n+k} - x| \\ &\leq 2^{-(n+2)} + 2^{-(n+4)} + 2^{-(n+k+2)} + 2^{-(n+k+4)} \\ &\leq 2^{-(n+2)} + 2^{-(n+4)} + 2^{-(n+3)} + 2^{-(n+5)} \\ &\leq 2^{-(n+1)} \end{aligned}$$
   Thus, $\psi \in \operatorname{dom}\xi$ and therefore ψ is a ξ-name of $\lim\psi = x$.

4. Computable equivalence between two representations implies there are continuous maps between them. Since the Cauchy representation is continuous itself [5], so is ξ.
5. We prove that $\operatorname{dom}\xi \subseteq \mathbb{D}_\omega$ is closed. For that we show that $\mathbb{D}_\omega \setminus \operatorname{dom}\xi$ is open. Let $\varphi \in \mathbb{D}_\omega$ with $\varphi \notin \operatorname{dom}\xi$. By definition of ξ, there are i, j such that $|\varphi_i - \varphi_{i+j}| > 2^{-(i+1)}$. Let $\psi \in \mathbb{D}_\omega$ with $\forall k \leq i+j : \varphi_k = \psi_k$. Then $\psi \notin \operatorname{dom}\xi$. Thus the open ball with prefix $(\varphi_k)_{k=0}^{i+j}$ is outside of $\operatorname{dom}\xi$. □

The converse of Item 2 does not hold. An example for a Cauchy-name of $0 \in \mathbb{R}$ is the sequence $(x_n)_n$ with $x_n = (-2)^{-n}$ for all $n \in \omega$, which does not satisfy $\forall i, j : |x_i - x_{i+j}| \leq 2^{-(i+1)}$. However, given a name of a real number, we can compute a corresponding ξ-name, this is one direction of the property in Item 3.

As a consequence of Item 2 a function-oracle machine $M^?$ computing $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ according to Definition 2.1 can be run on ξ-names of $\vec{x} \in \mathbb{R}^n$ leading to valid Cauchy-names of $f(\vec{x})$. Note that this proposition does not require $M_f^?$ to compute a ξ-name of $f(\vec{x})$. Any rational sequence rapidly converging to $f(\vec{x})$ is a valid output. This means, that the model of computation remains unchanged with respect to the earlier parts of this paper. It is the set of names the machines are operated on, which is restricted. This is reflected in Algorithm 2 by computing dyadic rational approximations φ_m of $[\![\vec{x}]\!]^\alpha$ such that $\varphi_m \in \mathbb{D}_m^n$ instead of keeping the name of $[\![\vec{x}]\!]^\alpha$ constant as has been done in Algorithm 1.

Algorithm 2 (Local linearisation) Algorithm δ-deciding $P : f(\vec{x}) \diamond 0$ and – in case unsat – computing a linearisation at α or returning "None", and in this case α satisfies P_δ. The function f is computed by machine $M_f^?$.

```
function LineariseLocal_δ(f, x⃗, ⋄, α)
  φ ← (m ↦ approx([[x⃗]]^α, m))                ▷ then φ is a ξ-name of [[x⃗]]^α
  pick p ∈ ℕ with 2^{-p} ≤ δ/4
  run M_f^φ(p+2), record its output y and its maximum query k ∈ ω to φ
  if y ⋄ −δ/2 then
    return None
  else
    return (x⃗ ∉ B([[x⃗]]^α, 2^{-k}))
  end if
end function
```

In particular, in Theorem 6.5 we show that linearisations for the (L) rule can be computed by Algorithm 2, which – in contrast to LINEARISE$_\delta$ in Algorithm 1 – does not require access to a procedure computing an upper bound μ_f on the uniform modulus of continuity of the non-linear function $f \in \mathcal{F}_{\text{nl}}$ valid on the entire bounded domain. It not just runs the machine $M_f^?$, but also observes the queries M_f^φ poses to its oracle in order to obtain a local modulus of continuity of f at the point of evaluation. The function $\operatorname{approx}(\vec{x}, m) := \lfloor \vec{x} \cdot 2^{m+1} \rceil / 2^{m+1}$ used to define Algorithm 2 computes a dyadic approximation of $\vec{x}$, with $\lfloor \cdot \rceil : \mathbb{Q}^n \to \mathbb{Z}^n$ denoting a rounding operation (computable by a classical Turing machine), that is, it satisfies $\forall \vec{q} : \|\lfloor \vec{q} \rceil - \vec{q}\| \leq \frac{1}{2}$.

Definition 6.2 *([37, Definition 6.2.6]).* Let $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ and $\vec{x} \in \operatorname{dom} f$. A function $\gamma : \mathbb{N} \to \mathbb{N}$ is called *a (local) modulus of continuity of f at $\vec{x}$* if for all $p \in \mathbb{N}$ and $\vec{y} \in \operatorname{dom} f$, $\|\vec{x} - \vec{y}\| \leq 2^{-\gamma(p)} \implies |f(\vec{x}) - f(\vec{y})| \leq 2^{-p}$ holds.

We note that in most cases a local modulus of continuity of f at $\vec{x}$ is smaller than the best uniform modulus of f on its domain, since it only depends on the local behaviour of f around x. One way of computing a local modulus of f at $\vec{x}$ is using the function-oracle machine $M_f^?$ as defined next.

Definition 6.3. Let $M_f^?$ compute $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ and let $\vec{x} \in \operatorname{dom} f$ have Cauchy-name φ. The function $\gamma_{M_f^?,\varphi} : p \mapsto \max\{0, k : M_f^\varphi(p+2)$ queries index k of $\varphi\}$ is called *the effective local modulus of continuity induced by $M_f^?$ at φ*.

The effective local modulus of continuity of f at a name φ of $\vec{x} \in \operatorname{dom} f$ indeed is a local modulus of continuity of f at $\vec{x}$ [19, Theorem 2.13]. Algorithm 2 computes ϵ-full linearisations by means of the effective local modulus as stated next.

Lemma 6.2. *Let $P : f(\vec{x}) \diamond 0$ be a non-linear constraint in $\mathcal{N}$ and α be an assignment of $\vec{x}$ to rationals in $\operatorname{dom} f$. Whenever $C =$ LINEARISELOCAL$_\delta(f, \vec{x}, \diamond, \alpha)$ and $C \neq$ None, C is an ϵ-full linearisation of P at α, with ϵ corresponding to the effective local modulus of continuity induced by $M_f^?$ at a ξ-name of $[\![\vec{x}]\!]^\alpha$.*

Proof. Let $P : f(\vec{x}) \diamond 0$, α and $C \neq$ None be as in the premise and let p, y and φ as in Algorithm 2. Since $C \neq$ None by construction $C = (\vec{x} \notin B([\![\vec{x}]\!]^\alpha, 2^{-k}))$ where $k = \gamma_{M_f^?,\varphi}(p)$ is the maximum query $M_f^\varphi(p+2)$ poses to its oracle. Thus, by definition, C is ϵ-full with $\epsilon = 2^{-k}$. In order to show that C indeed is a linearisation of P at α, let β be a total assignment with $\vec{z} := [\![\vec{x}]\!]^\beta \in B([\![\vec{x}]\!]^\alpha, 2^{-k})$. If $\vec{z} \notin \operatorname{dom} f$ there is nothing to show since $[\![f(\vec{x}) \diamond 0]\!]^\beta \neq$ true. Otherwise, since $\gamma_{M_f^?,\varphi}$ is a local

modulus of continuity of f at $[\![\vec{x}]\!]^\alpha$ [19, Theorem 2.13], $f(\vec{z})$ is within distance $2^{-p} \leq \delta/4$ of $f([\![\vec{x}]\!]^\alpha)$, which, by definition of $M_f^?$, is at most $2^{-(p+2)} < \delta/4$ away from y. By construction of C in Algorithm 2 the property $\neg(y \diamond -\delta/2)$ holds. As in case 2 in the proof of Lemma 5.4, this property implies $\neg(f(\vec{z}) \diamond 0)$. Therefore, according to Definition 3.4, C is an ϵ-full linearisation of P at α. □

Thus, the function LINEARISELOCAL$_\delta$ in Algorithm 2 is a drop-in replacement for LINEARISE$_\delta$ in Algorithm 1 since the condition on returning a linearisation of P versus accepting P_δ is identical. The linearisations however differ in the radius ϵ, which now, according to Lemma 6.2, corresponds to the effective local modulus of continuity. The resulting procedure we call NLINSTEPLOCAL$_\delta$. One of its advantages over NLINSTEP$_\delta$ is running $M_f^?$ on ξ-names instead of Cauchy-names. ξ-names form a compact set for bounded instances, unlike the Cauchy-names. This allows us to globally bound $\epsilon > 0$ for all computed ϵ-full local linearisations, see Theorem 6.5.

The following example shows that for some realisers of a computable function f no bound on the effective local modulus can be obtained that is valid for all Cauchy-names of $x \in \operatorname{dom} f$. In particular, there are realisers $\hat{M}^?$ of f such that the effective local modulus of continuity induced by $\hat{M}^?$ is unbounded on $\rho^{-1}(\{x\})$ for any precision n. More formally, the following holds.

$$\forall f \text{ computable}, x \in \operatorname{dom} f, \text{ precision } n \in \mathbb{N}:$$
$$\exists \hat{M}^? \text{ realising } f : \sup\{\gamma_{\hat{M}^?,\varphi}(n) : \varphi \in \rho^{-1}(\{x\})\} = \infty$$

Indeed, consider a realiser $M^?$ of a computable function $f : \mathbb{R} \rightarrowtail \mathbb{R}$.Then construct another realiser $\hat{M}^?$ of f that, upon input of a name $\varphi \in \mathbb{Q}^\omega$ of $x \in \mathbb{R}$ and $n \in \mathbb{N}$, performs the following steps.

1. Query $\varphi_0 = \frac{p}{q}$.
2. Query φ_q and discard the result.
3. Execute $M^\varphi(n)$.

Obviously, $\hat{M}^?$ computes the same function as $M^?$ but with potentially different queries to its oracle. By using the denominator of the (reduced) fraction $\frac{p}{q}$ as index, the query is unbounded on the set $\rho^{-1}(\{x\})$ of names of any $x \in \operatorname{dom} f$, since this set contains names that start with arbitrarily large denominators. Consequently, one can only obtain the bound $\epsilon \geq 0$ but not the required $\epsilon > 0$.

However, by removing such redundant names φ from the representation, (a) we obtain a way to bound $\epsilon > 0$ for the computed ϵ-full local linearisations of otherwise arbitrary δ-`ksmt` runs, and (b) we are not required to change the implementation of realisers of non-linear functions. The compatibility of ξ with the Cauchy-representation facilitates interoperability with different implementations of realisers, as long as they accept dyadic ξ-names. Prominent examples of such implementations include IRRAM [30], AERN2 [20], ARIADNE [2], CDAR [3]. This property of ξ is crucial to the feasibility of local linearisations and can be characterised as compactness of the set of names of a compact set, as shown in the following Lemma and Theorem. In a nutshell, the argument can be summarised as showing that the set $\{\varphi_p : \varphi \in \xi^{-1}(X)\}$ is compact, for any $p \in \mathbb{N}$ and any compact $X \subset \mathbb{R}^n$. A similar result has been shown to hold for the signed-digit representation ρ_{sd} [29], though it is important to note that ρ_{sd} operates on Cantor space which, unlike Baire space, already is compact. By virtue of Lemma 6.1, Item 2 the Cauchy-compatible compact representation ξ also matches implementations more closely than the signed-digit representation.

We prove compactness of preimages $\xi^{-1}(X)$ of compact sets X by first showing that they are closed.

Lemma 6.3. *Let $X \subset \mathbb{R}^n$ be closed. Then the set $\xi^{-1}(X)$ of ξ-names of elements in X is closed in $\mathbb{D}_\omega^n$.*

Proof. By definition, the complement $\mathbb{R}^n \setminus X$ is open in $\mathbb{R}^n$. By Lemma 6.1 Item 4 ξ is continuous, hence the preimage $\xi^{-1}(\mathbb{R}^n \setminus X)$ is an open subset of $\operatorname{dom}\xi$ as well. Therefore $\operatorname{dom}\xi \setminus \xi^{-1}(\mathbb{R}^n \setminus X) = \xi^{-1}(X)$ is closed in $\operatorname{dom}\xi$ which by Lemma 6.1 Item 5 is closed in $\mathbb{D}_\omega^n$ as well. Thus, $\xi^{-1}(X)$ is a closed subset of $\mathbb{D}_\omega^n$. □

Now, compactness of preimages $\xi^{-1}(X)$ follows from Tychonoff's theorem, which states that arbitrary products of non-empty compact spaces again are compact [36].

Theorem 6.4. *Let $X \subset \mathbb{R}^n$ be compact and let ξ denote the Cauchy-compatible compact representation of $\mathbb{R}^n$. Then the set $\xi^{-1}(X) \subset \mathbb{D}_\omega^n$ of ξ-names of elements in X is compact as well.*

Proof. By Item 2 in Lemma 6.1 $\xi^{-1}(X)$ is a subset of the product of the finite and therefore compact spaces

$$\{\vec{y} \in \mathbb{D}_k^n : \|\vec{x} - \vec{y}\| \leq 2^{-k}, \vec{x} \in X\}$$

over $k \in \omega$. As a closed (Lemma 6.3) subset of a compact space, $\xi^{-1}(X)$ is compact as well. □

This proof relies on the fact that for each component φ_k of ξ-names $(\varphi_k)_k$ of $\vec{x} \in X$ there are just finitely many choices from $\mathbb{D}_k^n$ due to the restriction of the length of the dyadics. This is not the case for the Cauchy representation used in Definition 2.1 and it is the key for deriving existence of a strictly positive lower bound ϵ on the ϵ-fullness of linearisations in the following Theorem.

Theorem 6.5. *Let $\delta \in \mathbb{Q}_{>0}$. For any bounded instance $\mathcal{L}_0 \wedge \mathcal{N}$ there is $\epsilon > 0$ such that any δ-ksmt run starting in $(nil, \mathcal{L}_0, \mathcal{N})$, where applications of (L) and (F_δ^{sat}) are performed according to* nlinStepLocal$_\delta$*, is ϵ-full.*

Proof. Assume $F = \mathcal{L}_0 \wedge \mathcal{N}$ is a bounded instance. Set $\epsilon := \min\{\epsilon_P : P \in \mathcal{N}\}$, where ϵ_P is defined as follows. Let $P : f(\vec{x}) \diamond 0$ in $\mathcal{N}$. Then the closure $\overline{D_{P,F}}$ of the bounded set $D_{P,F}$ is compact. Let E be the set of ξ-names of elements of $\overline{D_{P,F}} \subseteq \operatorname{dom} f$ (see Definition 6.1) and for any $\varphi \in E$ let k_φ be defined as $\gamma_{M_f^?,\varphi}(p)$ (see Definition 6.3) where p is computed from δ as in Algorithm 2 and is independent of φ. In order to find ϵ_P, consider the function $\varphi \mapsto k_\varphi$ mapping ξ-names $\varphi \in E$ to the effective local modulus of continuity induced by $M_f^?$ at φ evaluated at the precision p required for an accuracy of at least δ. For any $i \in \omega$, the preimage of $\{i\}$ under this function from $\mathbb{D}_\omega^n$ to ω is the set of all names $\varphi \in E$ where $M_f^\varphi(p+2)$ queries its oracle φ exactly up to i. This set is open in $\mathbb{D}_\omega^n$ as it corresponds to the union of basic open sets $\mathbb{D}_\omega^n[w]$ over all finite sequences $w \in \omega^i$ such that $w\varphi \in E$ for some $\varphi \in \omega^\omega$ and $k_{w\varphi} = i$, where $\mathbb{D}_\omega^n[w] \subset \mathbb{D}_\omega^n$ corresponds to 'the ball' around w as in the proof of Lemma 6.3. Therefore, $\varphi \mapsto k_\varphi$ is continuous on E, as is the composition $\varphi \mapsto 2^{-k_\varphi}$. By Theorem 6.4 the set E is compact, thus, there is $\psi \in E$ such that $2^{-k_\psi} = \inf\{2^{-k_\varphi} : \varphi \in E\}$. Set $\epsilon_P := 2^{-k_\psi}$. The claim then follows by Lemma 6.2. □

Thus we can conclude.

Corollary 6.6. *δ-ksmt with local linearisations is a δ-complete decision procedure.*

Remark 6.1. As a consequence of Theorem 6.5, for each δ and $P : f(x) \diamond 0$ there is a corresponding bound $\epsilon(\delta) > 0$ on ϵ for the ϵ-full linearisations. Though in our setting δ is fixed when considering δ-ksmt, these bounds implicitly define a uniform modulus of continuity of f on $\overline{D_P}$, as the following holds:

$$\forall p \in \mathbb{N}\ \forall x, x' \in \overline{D_P} : \|x - x'\| \leq \epsilon(2^{-p}) \implies |f(x) - f(x')| \leq 2^{-p}$$

Hence, μ_f can be defined as $p \mapsto \mu_f(p)$ where $2^{-\mu_f(p)} \leq \epsilon(2^{-p})$.

However, the benefit of local linearisations is that the lower bound $\epsilon(\delta)$ valid on $\overline{D_P}$ does *not* need to be computed (an exponential-time problem) and that most cases $\epsilon(\delta)$ is worse than the local ϵ obtained by local linearisations, which is valid at a particular point in $\overline{D_P}$. If fast access to both, uniform and local, moduli of continuity is available, their minimum can be used.

7. Conclusion

In this paper we extended the ksmt calculus to the δ-satisfiability setting and proved that the resulting δ-ksmt calculus is a δ-complete decision procedure for solving non-linear constraints over computable functions which include polynomials, exponentials, logarithms, trigonometric and many other functions used in applications. We presented algorithms for constructing ϵ-full linearisations ensuring termination of δ-ksmt. Based on methods from computable analysis we also presented an algorithm for constructing local linearisations. Local linearisations exclude larger regions from the search space and can be used to avoid computationally expensive global analyses of non-linear functions.

For future work we plan on implementing the δ-ksmt calculus in the existing ksmt solver by leveraging libraries for exact real computations. Its new rule requires examining all non-linear constraints at once in contrast to the more local linearisation rule, which is likely to pose challenges with respect to efficiency. We also plan to compare its performance using global and local linearisations to that of other δ-complete decision procedures.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable remarks and suggestions, which led to clarifications and some simplifications.

References

[1] J. Bard, H. Becker, E. Darulova, Formally verified roundoff errors using SMT-based certificates and subdivisions, in: M.H. ter Beek, A. McIver, J.N. Oliveira (Eds.), Formal Methods - The Next 30 Years - Third World Congress, FM 2019, Proceedings, Springer, 2019, pp. 38–44.
[2] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, T. Villa, Assume–guarantee verification of nonlinear hybrid systems with Ariadne, Int. J. Robust Nonlinear Control 24 (2014) 699–724.
[3] J. Blanck, CDAR: implementation of computable real numbers, https://github.com/jensblanck/cdar, 2021.
[4] M.P. Bonacina, S. Graham-Lengrand, N. Shankar, Conflict-driven satisfiability for theory combination: transition system and completeness, J. Autom. Reason. 64 (2020) 579–609.
[5] V. Brattka, P. Hertling, Topological properties of real number representations, Theor. Comput. Sci. 284 (2002) 241–257.
[6] V. Brattka, P. Hertling, K. Weihrauch, A Tutorial on Computable Analysis, Springer New York, New York, NY, 2008, pp. 425–491.
[7] F. Braußе, Z. Khasidashvili, K. Korovin, Selecting stable safe configurations for systems modelled by neural networks with ReLU activation, in: A. Ivrii, O. Strichman (Eds.), 2020 Formal Methods in Computer Aided Design, FMCAD 2020, IEEE, 2020, pp. 119–127.
[8] F. Braußе, K. Korovin, M.V. Korovina, N.T. Müller, A CDCL-style calculus for solving non-linear constraints, in: A. Herzig, A. Popescu (Eds.), Frontiers of Combining Systems - 12th International Symposium, FroCoS 2019, Proceedings, Springer, 2019, pp. 131–148.
[9] F. Braußе, M.V. Korovina, N.T. Müller, Towards using exact real arithmetic for initial value problems, in: M. Mazzara, A. Voronkov (Eds.), Perspectives of System Informatics - 10th International Andrei Ershov Informatics Conference, PSI 2015, in Memory of Helmut Veith, Revised Selected Papers, Springer, 2015, pp. 61–74.
[10] F. Braußе, M.V. Korovina, N.T. Müller, Using Taylor models in exact real arithmetic, in: I.S. Kotsireas, S.M. Rump, C.K. Yap (Eds.), Mathematical Aspects of Computer and Information Sciences - 6th International Conference, MACIS 2015, Berlin, Germany, November 11-13, 2015, Revised Selected Papers, Springer, 2015, pp. 474–488.
[11] F. Braußе, F. Steinberg, A minimal representation for continuous functions, CoRR, arXiv:1703.10044, 2017.
[12] A. Cimatti, A. Griggio, A. Irfan, M. Roveri, R. Sebastiani, Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions, ACM Trans. Comput. Log. 19 (2018) 19.
[13] P. Fontaine, M. Ogawa, T. Sturm, X. Vu, Subtropical satisfiability, in: C. Dixon, M. Finger (Eds.), Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Proceedings, Springer, 2017, pp. 189–206.
[14] S. Gao, J. Avigad, E.M. Clarke, δ-complete decision procedures for satisfiability over the reals, in: B. Gramlich, D. Miller, U. Sattler (Eds.), Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Proceedings, Springer, 2012, pp. 286–300.
[15] S. Gao, J. Avigad, E.M. Clarke, Delta-decidability over the reals, in: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, IEEE Computer Society, 2012, pp. 305–314.
[16] T.C. Hales, M. Adams, G. Bauer, D.T. Dang, J. Harrison, T.L. Hoang, C. Kaliszyk, V. Magron, S. McLaughlin, T.T. Nguyen, T.Q. Nguyen, T. Nipkow, S. Obua, J. Pleso, J.M. Rute, A. Solovyev, A.H.T. Ta, T.N. Tran, D.T. Trieu, J. Urban, K.K. Vu, R. Zumkeller, A formal proof of the Kepler conjecture, CoRR, arXiv:1501.02155, 2015.
[17] D. Jovanovic, L. de Moura, Solving non-linear arithmetic, ACM Commun. Comput. Algebra 46 (2012) 104–105.
[18] A. Kawamura, S.A. Cook, Complexity theory for operators in analysis, ACM Trans. Comput. Theory 4 (2012) 5.
[19] K. Ko, Complexity Theory of Real Functions, Birkhäuser/Springer, 1991.
[20] M. Konecnỳ, aern2-real: a Haskell library for exact real number computation, https://hackage.haskell.org/package/aern2-real, 2021.
[21] M. Konecný, E. Neumann, Implementing evaluation strategies for continuous real functions, CoRR, arXiv:1910.04891, 2019.
[22] K. Korovin, M. Kosta, T. Sturm, Towards conflict-driven learning for virtual substitution, in: V.P. Gerdt, W. Koepf, W.M. Seiler, E.V. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing - 16th International Workshop, CASC 2014, Proceedings, Springer, 2014, pp. 256–270.
[23] K. Korovin, N. Tsiskaridze, A. Voronkov, Conflict resolution, in: I.P. Gent (Ed.), Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Proceedings, Springer, 2009, pp. 509–523.
[24] K. Korovin, A. Voronkov, Solving systems of linear inequalities by bound propagation, in: N. Bjørner, V. Sofronie-Stokkermans (Eds.), Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Proceedings, Springer, 2011, pp. 369–383.
[25] J. Kurátko, S. Ratschan, Combined global and local search for the falsification of hybrid systems, in: A. Legay, M. Bozga (Eds.), Formal Modeling and Analysis of Timed Systems - 12th International Conference, FORMATS 2014, Proceedings, Springer, 2014, pp. 146–160.
[26] J.P. Marques-Silva, K.A. Sakallah, GRASP - a new search algorithm for satisfiability, in: R.A. Rutenbar, R.H.J.M. Otten (Eds.), Proceedings of the International Conference on Computer-Aided Design, ICCAD 1996, IEEE Computer Society/ACM, 1996, pp. 220–227.
[27] L.M. de Moura, D. Jovanovic, A model-constructing satisfiability calculus, in: R. Giacobazzi, J. Berdine, I. Mastroeni (Eds.), Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Proceedings, Springer, 2013, pp. 1–12.
[28] L.M. de Moura, G.O. Passmore, Computation in real closed infinitesimal and transcendental extensions of the rationals, in: M.P. Bonacina (Ed.), Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Proceedings, Springer, 2013, pp. 178–192.
[29] N.T. Müller, Subpolynomial complexity classes of real functions and real numbers, in: L. Kott (Ed.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 1986, pp. 284–293.
[30] N.T. Müller, The iRRAM: exact arithmetic in C++, in: J. Blanck, V. Brattka, P. Hertling (Eds.), Computability and Complexity in Analysis, 4th International Workshop, CCA 2000, Selected Papers, Springer, 2000, pp. 222–252.
[31] A. Platzer, Logical Foundations of Cyber-Physical Systems, Springer, 2018.
[32] M.B. Pour-El, J.I. Richards, Computability in Analysis and Physics, Perspectives in Mathematical Logic, Springer, 1989.
[33] D. Richardson, Some undecidable problems involving elementary functions of a real variable, J. Symb. Log. 33 (1968) 514–520.
[34] A. Tiwari, P. Lincoln, A search-based procedure for nonlinear real arithmetic, Form. Methods Syst. Des. 48 (2016) 257–273.
[35] V.X. Tung, T.V. Khanh, M. Ogawa, raSAT: an SMT solver for polynomial constraints, in: N. Olivetti, A. Tiwari (Eds.), Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Proceedings, Springer, 2016, pp. 228–237.
[36] A.N. Tychonoff, Über die topologische Erweiterung von Räumen, Math. Ann. 102 (1930) 544–561.
[37] K. Weihrauch, Computable Analysis – An Introduction, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2000.
[38] S. Willard, General Topology, Addison-Wesly, 1970.