A CDCL-style calculus for solving non-linear constraints[⋆]

F. Brauße[1], K. Korovin[2], M. Korovina[3], and N. Müller[1]

[1] Abteilung Informatikwissenschaften, Universität Trier, Germany
[2] The University of Manchester, UK
[3] A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russia

Abstract. In this paper we propose a novel approach for checking satisfiability of non-linear constraints over the reals, called ksmt. The procedure is based on conflict resolution in CDCL-style calculus, using a composition of symbolical and numerical methods. To deal with the non-linear components in case of conflicts we use numerically constructed restricted linearisations. This approach covers a large number of computable non-linear real functions such as polynomials, rational or trigonometrical functions and beyond. A prototypical implementation has been evaluated on several non-linear SMT-LIB examples and the results have been compared with state-of-the-art SMT solvers.

1 Introduction

Continuous constraints occur naturally in many areas of computer science such as verification of safety-critical systems, program analysis and theorem proving. Historically, there have been two major approaches to solving continuous constraints. One of them is the symbolic approach, originated by the Tarski's decision procedure for the real closed fields [31], and developed further in procedures based on cylindrical decomposition (CAD) [5], Gröbner basis [3,14], and virtual substitution [20,7]. Another one is the numerical approach, based on interval computations, where the technique of interval constraint propagations have been explored to deal with continuous constraints on compact intervals, e.g., [1,10,12,11]. It is well known that both approaches have their strengths and weaknesses concerning completeness, efficiency and expressiveness.

Recently, a number of methods has been developed aimed at merging strengths of symbolical and numerical methods, e.g. [27,9,4,28]. In particular, the approach developed in this paper is motivated by extensions of CDCL-style reasoning into domains beyond propositional logic such as linear [17,23,16,15] and polynomial constraints [13]. In this paper we develop a conflict-driven framework called ksmt

---

for solving non-linear constraints over large class of functions including polynomial, exponential and trigonometric functions. Our approach combines model guided search for a satisfying solution and constraint learning as the result of failed attempts to extend the candidate solution.

In the nutshell, our ksmt algorithm works as follows. Given a set of non-linear constraints, we first separate the set into linear and non-linear parts. Then we incrementally extend a candidate solution into a solution of the system and when such extension fails we resolve the conflict by generating a lemma that excludes a region which includes the falsifying assignment. There are two types of conflicts: between linear constraints which are resolved in a similar way as in [17] and non-linear conflicts which are resolved by local linearisations developed in this paper. One of the important properties of our algorithm is that all generated lemmas are linear and hence the non-linear part of the problem remains unchanged during the search. In other words, our algorithm can be seen as applying gradual linear approximations of non-linear constraints by local linearisations guided by solution search in the CDCL-style.

The quantifier-free theory of reals with transcendental functions is well known to be undecidable [30] and already problems with few variables pose considerable challenge for automated systems. In this paper we focus on a practical algorithm for solving non-linear constraints applicable to problems with large number of variables rather than on completeness results. Our ksmt algorithm can be used for both finding a solution and proving that no solution exist. In addition to a general framework we discuss how our algorithm works in a number of important cases such as polynomials, transcendental and some discontinuous functions. In this paper we combine solution guided search in the style of conflict resolution, bound propagation and MCSAT [6] with linearisations of real computable functions. The theory of computable functions has been developed in Computable Analysis [32] with implementations provided by exact real arithmetic [24]. Linearisations have been employed in different SMT theories before, including NRA and a recently considered one with transcendental functions [21,29,4], however, not for the broad class we consider here. We define a general class of functions called *functions with decidable rational approximations* to which our approach is applicable. This class includes common transcendental functions, exponentials, logarithms but also some discontinuous functions.

We implemented the ksmt algorithm and evaluated it on SMT benchmarks. Our implementation is at an early stage and lacking many features but already outperforms many state-of-the-art SMT solvers on certain classes of problems.

2 Preliminaries

We consider the reals extended with non-linear functions $\mathbb{R}_{nl} = (\mathbb{R}, \langle \mathcal{F}_{lin} \cup \mathcal{F}_{nl}, \mathcal{P} \rangle)$, where $\mathcal{F}_{lin}$ consists of rational constants, addition and multiplication by rational constants; $\mathcal{F}_{nl}$ consists of a selection of non-linear functions including multiplication, trigonometric, exponential and logarithmic functions; $\mathcal{P} = \{<, \leq, >, \geq\}$ are predicates.

We consider a set of variables V. We will use x, y, z possibly with indexes for variables in V, similar we will use q, a, b, c, d for rationals, f, g for non-linear functions in $\mathcal{F}_{nl}$. Terms, predicates and formulas over X are defined in the standard way. We will also use predicates $\neq, =$, which can be defined using predicates in $\mathcal{P}$. An atomic formula is a formula of the form $t \diamond 0$ where $\diamond \in \mathcal{P}$. A literal is either an atomic formula or its negation. In this paper we consider only quantifier-free formulas in conjunctive normal form. We will use conjunctions and sets of formulas interchangeably.

We assume that terms are suitably normalised. A linear term is a term of the form $q_1 x_1 + \ldots + q_n x_n + q_0$. A linear inequality is an atomic formula of the form $q_1 x_1 + \ldots + q_n x_n + q_0 \diamond 0$. A linear clause is a disjunction of linear inequalities and a formula is in linear CNF if it is a conjunction of linear clauses.

2.1 Separated linear form

In this paper we consider the satisfiability problem of quantifier-free formulas in CNF over $\mathbb{R}_{nl}$, where the linear part is separated from the non-linear part which we call separated linear form.

Definition 1. *A formula F is in* separated linear form *if it is of the form $F = \mathcal{L} \cup \mathcal{N}$ where $\mathcal{L}$ is a set of clauses containing predicates only over linear terms and $\mathcal{N}$ is a set of unit-clauses each containing only non-linear literals of the form $x \diamond f(\boldsymbol{t})$, where $f \in \mathcal{F}_{nl}$, $\boldsymbol{t}$ is a vector of terms and $\diamond \in \mathcal{P}$.*

Lemma 1 (Monotonic flattening). *Any quantifier-free formula F in CNF over $\mathbb{R}_{nl}$ can be transformed into an equi-satisfiable separated linear form in polynomial time.*

Proof. Consider a clause C in F which contains a linear combination of non-linear terms, i.e., is of the form $C = qf(\boldsymbol{t}) + p \diamond 0 \vee D$, where $f \in \mathcal{F}_{nl}$ and $q \neq 0$. Then we introduce a fresh variable x, add $x \diamond' f(\boldsymbol{t})$ into S and replace C with $qx + p \diamond 0 \vee D$. Here, $\diamond'$ is $\geq$, if either $q > 0$ and $\diamond \in \{\leq, <\}$ or $q < 0$ and $\diamond \in \{\geq, >\}$; and $\diamond'$ is $\leq$ otherwise. The resulting formula is equi-satisfiable to F. The claim follows by induction on the non-linear monomials.

Let us remark that monotonic flattening avoids introducing equality predicates, which is based on the monotonicity of linear functions. In some cases we need to flatten non-linear terms further (in particular to be able to represent terms as functions in the $\mathcal{F}_{\mathrm{DA}}$ class introduced in Section 5). In most cases this can be done in the same way as in Lemma 1 based on monotonicity of functions in corresponding arguments, but we may need to introduce linear conditions expressing regions of monotonicity. For simplicity of the exposition we will not consider such cases here.

2.2 Trails and Assignments

Any sequence of single variable assignments $\alpha \subset (V \times \mathbb{Q})^*$ such that a variable is assigned at most once is called a *trail*. By ignoring the order of assignments in

α, we will regard α as a (partial) assignment of the real variables in V and use $V(\alpha) \subseteq V$ to denote the set of variables assigned in α. We use the notation $[\![t]\!]^\alpha$ to denote the (partial) application of α to a term t, that is, the term resulting from replacing every free variable x in t such that $x \in V(\alpha)$ by $\alpha(x)$ and evaluating term operations on constants in their domains. We extend $[\![\cdot]\!]^\alpha$ to predicates over terms and to CNF in the usual way. An evaluation of a formula results in true or false, if all variables in the formula are assigned, or else in a partially evaluated formula. A *solution* to a CNF $\mathcal{C}$ is a total assignment α such that each term in $\mathcal{C}$ is defined under α and for each clause $C \in \mathcal{C}$ there is (at least) one literal $l \in C$ with $[\![l]\!]^\alpha = $ true.

Any triple $(\alpha, \mathcal{L}, \mathcal{N})$ when α is a trail, $\mathcal{L}$ is a set of clauses over linear predicates and $\mathcal{N}$ is a set of unit clauses over non-linear predicates is called *state*. A state is called *linearly conflict-free* if $[\![\mathcal{L}]\!]^\alpha \neq $ false. It is called *conflict-free* if it is linearly conflict-free and $[\![\mathcal{N}]\!]^\alpha \neq $ false.

The main problem we consider in this paper is finding a solution to $\mathcal{L} \wedge \mathcal{N}$ or showing that no solution exists.

3   The ksmt algorithm

Our ksmt algorithm will be based on a CDCL-type calculus [22,25] and is in the spirit of Conflict Resolution [17,16], Bound Propagation [18,8], GDPLL [23], MCSAT [6] and related algorithms.

The ksmt calculus will be presented as a set of transition rules that operate on the states introduced previously. The *initial state* is a state of the form $(\text{nil}, \mathcal{L}, \mathcal{N})$. A final state will be reached when no further ksmt transition rules (defined below) are applicable.

Informally, the ksmt algorithm starts with a formula in separated linear form and the empty trail, and extends the trail until the solution is found or a trivial inconsistency is derived by applying the ksmt transition rules. During the extension process the algorithm may encounter conflicts which are resolved by deriving lemmas which will be linear clauses. These lemmas are either derived by resolution between two linear clauses or by linearisation of non-linear conflicts, which is described in detail in Section 3.4. One of the important properties of our calculus is that we only generate linear lemmas during the run of the algorithm and the non-linear part $\mathcal{N}$ remains fixed.

3.1   General procedure

Let $(\alpha, \mathcal{L}, \mathcal{N})$ be a conflict-free state and $z \in V \setminus V(\alpha)$ be a variable unassigned in α. Assume there is no $q \in \mathbb{Q}$ such that $(\alpha :: z \mapsto q, \mathcal{L}, \mathcal{N})$ is linearly conflict-free. That means that for any potential assignment q there is a clause $D \in \mathcal{L}$ not satisfied under $\alpha :: z \mapsto q$. Another way of viewing this situation, called a *conflict*, is that there are clauses consisting under α only of predicates linear in and only depending on z that contradict each other. Analogously to resolution

in propositional logic,

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

the following inference rule we call *arithmetical resolution on x* is sound [17,23] on clauses over linear predicates:

$$\frac{A \vee (cx + d \leq 0) \quad B \vee (-c'x + d' \leq 0)}{A \vee B \vee (c'd + cd' \leq 0)}$$

where c, c' are positive rational constants and d, d' are linear terms. Similar rules exist for strict comparisons. We denote by $R_{\alpha,\mathcal{L},z}$ a set of resolvents of clauses in $\mathcal{L}$ upon variable z such that $[\![R_{\alpha,\mathcal{L},z}]\!]^\alpha = \mathsf{false}$. In Section 3.3 we discuss how to obtain such a set.

We consider the following rules for transforming states into states under some preconditions, i.e., the binary relation $\Rightarrow$ on states.

Assignment refinement: In order to refine an existing partial assignment α by assigning $z \in V$ to $q \in \mathbb{Q}$ in a state $(\alpha, \mathcal{L}, \mathcal{N})$, the state needs to be linearly conflict-free, that is, no clause over linear predicates in $\mathcal{L}$ must be false under α. Additionally, under this assignment the clauses over linear predicates in $\mathcal{L}$ must be valid under the new assignment, formally: For any state $(\alpha, \mathcal{L}, \mathcal{N})$, $z \in V$ and $q \in \mathbb{Q}$

$$(\alpha, \mathcal{L}, \mathcal{N}) \Rightarrow (\alpha :: z \mapsto q, \mathcal{L}, \mathcal{N}) \tag{A}$$

whenever $[\![\mathcal{L}]\!]^\alpha \neq \mathsf{false}$, $z \notin V(\alpha)$, and $[\![\mathcal{L}]\!]^{\alpha::z\mapsto q} \neq \mathsf{false}$. In the linear setting of [17], this rule exactly corresponds to "assignment refinement".

Conflict resolution: Assume despite state $(\alpha, \mathcal{L}, \mathcal{N})$ being linearly conflict-free and $z \in V$ unassigned in α there is no rational value to assign to z that makes the resulting state linearly conflict-free. This means, that for any $q \in \mathbb{Q}$ there is a *conflict*, i.e., a clause in $\mathcal{L}$ that is false under $\alpha :: z \mapsto q$. In order to progress in determining $\mathtt{sat}$ or $\mathtt{unsat}$, the partial assignment α needs to be excluded from the search space. Arithmetical resolution $R_{\alpha,\mathcal{L},z}$ provides exactly that: a set of clauses preventing any $\beta \supseteq \alpha$ from being linearly conflict-free. For any state $(\alpha, \mathcal{L}, \mathcal{N})$ and $z \in V$

$$(\alpha, \mathcal{L}, \mathcal{N}) \Rightarrow (\alpha, \mathcal{L} \cup R_{\alpha,\mathcal{L},z}, \mathcal{N}) \tag{R}$$

whenever $[\![\mathcal{L}]\!]^\alpha \neq \mathsf{false}$, $z \notin V(\alpha)$ and $\forall q \in \mathbb{Q} : [\![\mathcal{L}]\!]^{\alpha::z\mapsto q} = \mathsf{false}$. In the linear setting of [17], this rule corresponds to "conflict resolution".

Backjumping: In case the state $(\alpha, \mathcal{L}, \mathcal{N})$ contains one or more top-level assignments that make it not linearly conflict-free, these assignments are removed. This is commonly known as backjumping. Indeed, when transitioning to applying this rule, the information on the size of the suffix of assignments to remove is already available, as is detailed in Section 3.2. Formally, for a state $(\alpha, \mathcal{L}, \mathcal{N})$ such that $[\![\mathcal{L}]\!]^\alpha = \mathsf{false}$, let γ be the maximal prefix of α such that $[\![\mathcal{L}]\!]^\gamma \neq \mathsf{false}$. Then, Backjumping is defined as follows:

$$(\alpha, \mathcal{L}, \mathcal{N}) \Rightarrow (\gamma, \mathcal{L}, \mathcal{N}) \tag{B}$$

Linearisation: The above rules are only concerned with keeping the (partial) assignment linearly conflict-free. This rule extends the calculus to ensure that the non-linear clauses in $\mathcal{N}$ are conflict-free as well. In essence, the variables involved in a non-linear conflict are "lifted" into the linear domain by a linearisation of the conflict local to α. The resulting state will not be linearly conflict-free as is shown in Lemma 4. Formally, if $(\alpha, \mathcal{L}, \mathcal{N})$ is a state and $L_{\alpha, \mathcal{N}}$ a non-empty set of linearisation clauses as detailed in Section 3.4, then the rule reads as

$$(\alpha, \mathcal{L}, \mathcal{N}) \Rightarrow (\alpha, \mathcal{L} \cup L_{\alpha, \mathcal{N}}, \mathcal{N}) \tag{L}$$

whenever $[\![\mathcal{L}]\!]^\alpha \neq \mathsf{false}$ and $[\![\mathcal{N}]\!]^\alpha = \mathsf{false}$.

Let us note that the set $\mathcal{N}$ remains unchanged over any sequence of states obtained by successive application of the above rules.

Lemma 2 (Soundness). *Let I be an input instance in separated linear form. Let $(S_0, S_1, \ldots, S_n)$ be a sequence of states $S_i = (\alpha_i, \mathcal{L}_i, \mathcal{N})$ where S_0 is the initial state and each S_{i+1} is derived from S_i by application of one of the rules (A), (R), (B), (L).*

1. *For all $i < n$ and total assignments $\alpha : V \to \mathbb{Q}$: $[\![\mathcal{L}_i \wedge \mathcal{N}]\!]^\alpha = [\![\mathcal{L}_{i+1} \wedge \mathcal{N}]\!]^\alpha$.*
2. *If no rule is applicable to S_n then the following are equivalent:*
   - *I is satisfiable,*
   - *α_n is a solution to I,*
   - *S_n is linearly conflict-free,*
   - *the trivial conflict clause $(1 \leq 0)$ is not in $\mathcal{L}_n$.*

Lemma 3 (Progress). *Let $(S_i)_i$ be a sequence of states $S_i = (\alpha_i, \mathcal{L}_i, \mathcal{N})$ produced from initial state S_0 by the $\mathtt{ksmt}$ rules, n be the number of variables and*

$$\Lambda_i := \{\alpha : (A) \text{ cannot be applied to } (\alpha, \mathcal{L}_i, \mathcal{N}) \text{ linearly conflict-free}\}.$$

Then $\Lambda_i \supseteq \Lambda_{i+1}$ and $\Lambda_i \neq \Lambda_{i+n+2}$ hold for all i.

The proofs follow from the following:

1. (A) does not change Λ and can be applied consecutively at most n times,
2. after application of (R) or (L) the set Λ is reduced which follows from the properties of the resolvent, and Corollary 2 respectively, and
3. (B) does not change Λ and can be applied only after (R) or (L).

Corollary 1. *After at most $n + 2$ steps the search space is reduced.*

3.2 Concrete algorithm

The algorithm transforms the initial state by applying $\mathtt{ksmt}$ transition rules exhaustively. The rule applicability graph is shown in Figure 1. The rule (B) is applicable whenever the linear part is false in the current assignment. This is always the case after applications of either (R) or (L). In order to check applicability of remaining rules (A), (R) and (L) the following conditions need to be checked.
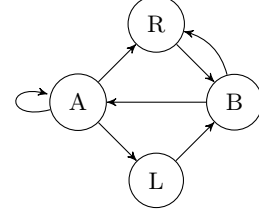
1. Is the state conflict-free? In particular, we need to check whether the non-linear part evaluates to false under the current assignment. Decidability of this problem for the broad class of functions $\mathcal{F}_{\mathrm{DA}}$ is shown in Section 5.1, along with concrete algorithms for common classes of non-linear functions.
2. If the state is linearly conflict-free and a variable is chosen, can it be assigned in a way that the linear part remains conflict-free? A polynomial-time procedure is described in Section 3.3.

These computations determine whether (A), (R) or (L) is applicable next. Item 2 has to be checked after each application of (A) and (B). Note that in case of transitioning to an application of rule (B) the size of the suffix of assignments to revoke is syntactically available in form of the highest position in α of a variable in $R_{\alpha,\mathcal{L},z}$ or the linearisation $L_{\alpha,\mathcal{N}}$, respectively.

Fig. 1. Transitions between applicability of rules.

Let us note that the calculus allows for flexibility in the choices of:

1. The variable z and value q to assign to z when applying rule (A).
2. Which arithmetical resolutions to perform when applying rule (R).
3. Which linearisations to perform when applying rule (L). We describe the general conditions in Section 3.4 and our approach in Section 5.2.

Many of the heuristics presented in [16,8] are applicable to items 1 and 2 as well.

3.3 Determining bounds and resolvents

In this section we consider the problem of checking whether we can extend the trail of a linearly conflict-free state in such a way that the linear part remains conflict-free after the extension and in this case we apply rule (A), or otherwise there is a conflict which should be resolved by applying rule (R).

Given a linearly conflict-free state $(\alpha, \mathcal{L}, \mathcal{N})$ and a variable z unassigned in α, the problem

$$\exists q \in \mathbb{Q} : [\![\mathcal{L}]\!]^{\alpha::z \mapsto q} \neq \mathsf{false}$$

can be solved efficiently by the following algorithm. Let $\mathcal{L}_{z,\alpha}$ be those partially applied (by α) clauses from $\mathcal{L}$ that only depend on z. The other clauses are either already satisfied or depend on a further unassigned variable. So each $D \in \mathcal{L}_{z,\alpha}$ is 'univariate', i.e. just a set of $z \diamond c_i$. The disjunction of these simple predicates in D is equivalent to a clause of the form (i) $z < a \lor z > b$, perhaps with non-strict inequalities, giving an alternative between a lower and an upper bound or (ii) a unit clause for a lower bound, or (iii) a unit clause for an upper bound, or (iv) an arithmetic tautology. So each clause is equivalent to the union of at most two half-bounded rational intervals. The conjunction of two such clauses corresponds to the intersection of sets of intervals, which is again a set of intervals. This intersection can be computed easily and can also be checked for emptiness. In case the intersection is not empty, it even gives us intervals to choose an assignment q for z with $[\![\mathcal{L}]\!]^{\alpha::z \mapsto q} \neq \mathsf{false}$. If the intersection is empty,

we know there is no such q and we can use arithmetical resolution to resolve this conflict to obtain $R_{\alpha,\mathcal{L},z}$.

3.4 Non-linear predicates

While resolution is a well-established and efficient symbolic technique for dealing with the linear part of the CNF under consideration, there seem to be no similarly easy techniques for non-linear predicates. The approach presented here is based on numerical approximations instead.

Given a linearly conflict-free state $(\alpha, \mathcal{L}, \mathcal{N})$, in order to decide on the applicability of (L), the non-linear unit clauses in $\mathcal{N}$ have to be checked for validity under α. If all are valid, then, by definition, $(\alpha, \mathcal{L}, \mathcal{N})$ is conflict-free. Lemma 5 gives sufficient conditions on the non-linear functions in $\mathcal{F}_{nl}$ in order to make this problem decidable. In this section, we will describe how we deal with the case that some unit clause $\{P\} \in \mathcal{N}$ is false under α, where according to (L) we construct a linearisation of P with respect to α. We will not need the order of variables given in the trail α, so we will only use α as a partial assignment.

Definition 2. *Let P be a non-linear predicate and let α be a partial assignment with $[\![P]\!]^\alpha = \mathsf{false}$. An (α, P)-linearisation is a clause $L_{\alpha,P} = \{L_i : i \in I\}$ consisting of finitely many rational linear predicates $(L_i)_{i \in I}$ with the properties*

1. *$\{\beta : [\![P]\!]^\beta = \mathsf{true}\} \subseteq \{\beta : [\![L_{\alpha,P}]\!]^\beta = \mathsf{true}\}$, and*
2. *$[\![L_{\alpha,P}]\!]^\alpha = \mathsf{false}$.*

If we let $\boldsymbol{c}_\alpha$ denote the values assigned in α and $\boldsymbol{x}$ the vector of assigned variables, we can reformulate the properties of $L_{\alpha,P}$ as a formula:

$$\left(P \implies \bigvee_{i \in I} L_i \right) \wedge \left(\boldsymbol{x} = \boldsymbol{c}_\alpha \implies \neg \bigvee_{i \in I} L_i \right)$$

This formula will not be added to the system but is just used as a basis for discussions. Later we will use a similar formalism to define linearisation clauses.

A central idea of our approach is to add $L_{\alpha,P}$ as a new clause to the CNF, as well as the predicates L_i. Adding $L_{\alpha,P}$ is sound, as the following lemma shows:
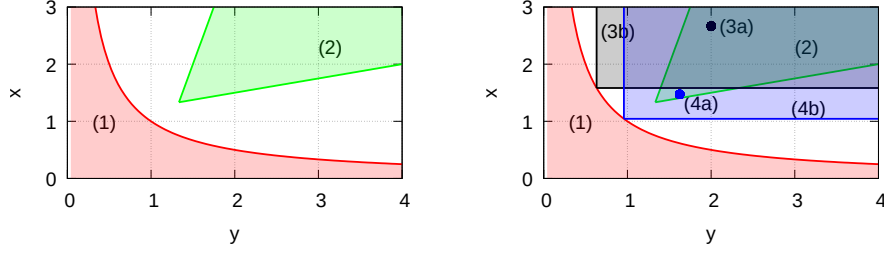
Lemma 4. *Suppose a partial assignment α violates a predicate P with $\{P\} \in \mathcal{N}$, so $[\![P]\!]^\alpha = \mathsf{false}$. Further suppose $L_{\alpha,P}$ is an (α, P)-linearisation.*

1. *Any β, which is a solution for $\mathcal{L} \cup \mathcal{N}$, is also a solution for $\mathcal{L} \cup \{L_{\alpha,P}\} \cup \mathcal{N}$.*
2. *$(\alpha, \mathcal{L} \cup \{L_{\alpha,P}\}, \mathcal{N})$ is not linearly conflict-free.*

Corollary 2. *Whenever (L) is applied, the search space is reduced.*

Hence at least the partial assignment α (and all extensions thereof) are removed from the search space for the linear part of our CNF, at the cost of adding the clause $L_{\alpha,P}$ usually containing several new linear predicates L_i. In general, our linearisations will not just remove single points but rather polytopes from the search space.

Fig. 2. Initial system and linearisations constructed.

We should emphasise several remarks on the linearisations: There is a high degree of freedom when choosing a linearisation for a given pair (α, P). Techniques for constructing these will be discussed in Section 5.2. They will all be based on numerical approximations.

Furthermore we are allowed to add more than one clause in one step, so we can construct several linearisations for different (α, P') as long as $[\![P']\!]^\alpha = \mathsf{false}$, and then add all of them. This has already been formulated in (L) as a *set* of linearisation clauses $L_{\alpha, \mathcal{N}}$ instead of a single clause $L_{\alpha, P}$.

4 Example

As a basic example describing our method we consider the conjunction of the non-linear predicate $P : (x \leq \frac{1}{y})$, and linear constraints $L_1 : (x \geq y/4 + 1)$ and $L_2 : (x \leq 4 \cdot (y - 1))$, shown on Figure 2. We will first detail on how linearisations can be constructed numerically for P. In Section 5.2 we will detail on how linearisations can be constructed in general.

Linearisation of P. Assume $[\![P]\!]^\alpha = \mathsf{false}$ under assignment α. By definition, α assigns (x, y) to some values (c_x, c_y) such that $c_x > 1/c_y$, (point (3a), at $(8/3, 2)$). Here we will only discuss the case $c_y > 0$ needed below. The other cases can be dealt with in a similar way. To construct an (α, P)-linearization, first we compute the rational number d such that $1/c_y < d < c_x$. In this example, we take $d := (c_x + 1/c_y)/2$, that is, for this linearisation $19/12 \approx 1.58$. In general, such values are computed by numerical approximations to the function value. Then the clause $L_{\alpha, P} = \{x \leq d, y \leq 1/d\}$ is the required linearisation (which excludes region (3b) containing the conflicting assignment). Indeed, $L_{\alpha, P}$ is implied by P and $[\![L_{\alpha, P}]\!]^\alpha = \mathsf{false}$.

After adding $L_{\alpha, P}$ to the linear constraints, region (3b) is excluded from the search space and backjumping to the empty assignment is performed (since $8/3$ is not a linearly conflict-free assignment to x anymore). The system again is linearly conflict-free. In the next iteration we obtain a solution (4a) roughly at $(1.47, 1.63)$ to the new linear system, linearisation at (4a) results in linear lemma excluding region (4b) where $d \approx 1.04$. Finally, the resulting linear constraints are unsatisfiable and therefore the original system is proven to be also unsatisfiable. This example is based on an actual run of our system.

5 Schemes for local linearisations

A successful linearisation scheme has to fulfil two tasks: (a) deciding whether a trail α is in conflict with a non-linear predicate P and then, if there is a conflict, (b) finding reasonable linearisations $L_{\alpha,P}$. We first address task (a).

5.1 Deciding non-linear conflicts

By Definition 1, P is of the form $x \diamond f(\boldsymbol{t})$, where f is a function symbol, $\boldsymbol{t}$ is a vector of terms, and $\diamond \in \{<, \leq, >, \geq\}$. In the following assume that the terms in $\boldsymbol{t}$ use the variables $(y_1, \ldots, y_k) = \boldsymbol{y} \in V^k$. So the semantical interpretation $[\![f(\boldsymbol{t})]\!]$ of the syntactical term $f(\boldsymbol{t})$ is a function $g : \mathbb{R}^k \to \mathbb{R}$.

In order to introduce the class $\mathcal{F}_{\mathrm{DA}}$ we use the following notion of approximable function.

Definition 3. *We call a partial function $g : \mathbb{R} \to \mathbb{R}$ approximable if the set*

$$\square_g := \{(p, q, s, t) : g([p, q]) \subset (s, t),\ p, q, s, t \in \mathbb{Q}\}$$

is computably enumerable. Here, $g(I)$ denotes the set-evaluation of g on I, that is, $\{g(x) : x \in I \cap \mathrm{dom}\, g\}$.

This definition can easily be generalized to the multi-variate case by taking boxes $[p_1, q_1] \times \cdots \times [p_k, q_k]$ with $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{Q}^k$. For total continuous real functions, approximability coincides with the notion of computability known from Computable Analysis (TTE) [32,2].

Given a number $d \in \mathbb{Q}$ and a vector $\boldsymbol{c} \in \mathbb{Q}^k$ with $d \neq g(\boldsymbol{c})$ we can always decide whether $d \diamond g(\boldsymbol{c})$ holds if $g : \mathbb{R}^k \to \mathbb{R}$ is a total approximable function. However, in general we cannot decide the premise $d \neq g(\boldsymbol{c})$. Therefore we restrict our considerations to a general class of functions where this problem is decidable.

Definition 4. *A partial function $g : \mathbb{R}^k \to \mathbb{R}$ is called a* function with decidable rational approximations, *denoted $g \in \mathcal{F}_{\mathrm{DA}}$, if the following holds.*

- *$\mathrm{dom}(g)$ is decidable on $\mathbb{Q}^k$,*
- *$\mathrm{graph}(g)$ is decidable on $\mathbb{Q}^k \times \mathbb{Q}$, and*
- *g is approximable.*

The following important classes of functions belong to $\mathcal{F}_{\mathrm{DA}}$.

Multivariate polynomials. For multivariate polynomials g with rational coefficients, rational arguments are mapped to rational results using rational arithmetic and the relations $\diamond$ under consideration are decidable on $\mathbb{Q}^2$.

Selected elementary transcendental functions. Let $g \in \{\exp, \ln, \log_b, \sin, \cos, \tan, \arctan\}$, where in the case of $\log_b$, $b \in \mathbb{Q}$. Let us show that $g \in \mathcal{F}_{\mathrm{DA}}$. Indeed, it is well known that $g : \mathbb{R} \to \mathbb{R}$ is computable [32]. Since emptiness of $[p, q] \setminus \mathrm{dom}\, g$ is decidable, g is also approximable. In addition, $X_g := \mathrm{graph}(g) \cap \mathbb{Q}^2$ either consists of a single point, or in the case of $\log_b$, is of the form $X_g = \{(b^n, n) : n \in \mathbb{Z}\}$ [26] and therefore is decidable, as is the respective domain.

Selected discontinuous functions. Additionally, $\mathcal{F}_{\text{DA}}$ includes some discontinuous functions like e.g. the step-functions taking rational values with discontinuities at finitely many rational points and more generally piecewise polynomials defined over intervals with a decidable set of rational endpoints. Multi-variate piecewise defined functions with non-axis-aligned discontinuities are included as well.

Lemma 5. *Let P be a predicate over reals and let α be a trail assigning all variables used in P. If P is linear or $P : (x \diamond f(\boldsymbol{t}))$ with $[\![f(\boldsymbol{t})]\!] \in \mathcal{F}_{\text{DA}}$ then $[\![P]\!]^{\alpha}$ is computable.*

Proof. By definition, trails α contain rational assignments. If P is linear, there is nothing to show. Let $P : (x \diamond f(\boldsymbol{t}))$ with $g(\boldsymbol{y}) = [\![f(\boldsymbol{t})]\!] \in \mathcal{F}_{\text{DA}}$ where $\boldsymbol{y}$ is the vector of free variables in terms $\boldsymbol{t}$. The cases $[\![\boldsymbol{y}]\!]^{\alpha} \notin \operatorname{dom} g$ and $[\![(\boldsymbol{y}, x)]\!]^{\alpha} \in \operatorname{graph}(g)$ are decidable by the definition of $\mathcal{F}_{\text{DA}}$. The remaining case is $\boldsymbol{z} := [\![\boldsymbol{y}]\!]^{\alpha} \in \operatorname{dom} g$ and $[\![(\boldsymbol{y}, x)]\!]^{\alpha} \notin \operatorname{graph}(g)$. Perform a parallel search for 1) $q \in \mathbb{Q}$ with $(\boldsymbol{z}, q) \in \operatorname{graph}(g)$ and for 2) a rational interval box $\boldsymbol{I} \times J$ in $\square_{\tilde{g}}$ with $\boldsymbol{z} \in \boldsymbol{I}$ and $[\![x]\!]^{\alpha} \notin J$. We now show that this search terminates. Either $g(\boldsymbol{z}) \in \mathbb{Q}$, then $q = g(\boldsymbol{z})$ can be found in the graph of g, or $g(\boldsymbol{z}) \notin \mathbb{Q}$, then $|[\![x]\!]^{\alpha} - g(\boldsymbol{z})| > 0$, thus there is a rational interval box $\boldsymbol{I} \times (s, t) \in \square_g$ with $\boldsymbol{z} \in \boldsymbol{I}$ and $s, t \in \mathbb{Q}$ such that $[\![x]\!]^{\alpha} \notin (s, t)$. Note that $\boldsymbol{I}$ can be the point-interval $[\boldsymbol{z}]$ since $\boldsymbol{z} \in \operatorname{dom} g$. $\qquad\square$

In particular, if all predicates $P : (x \diamond f(\boldsymbol{t}))$ appearing in a given problem instance are such that the function $[\![f(\boldsymbol{t})]\!]$ used in this instance are from $\mathcal{F}_{\text{DA}}$, we can decide if a `ksmt` state is conflict-free as required in Section 3.2.

5.2   Linearisations for functions in $\mathcal{F}_{\text{DA}}$

This section addresses task (b), namely finding reasonable linearisations $L_{\alpha, P}$ in case a trail α is in conflict with a non-linear predicate P, that is, $[\![P]\!]^{\alpha} = \mathsf{false}$. In order to reduce the number of cases, we assume that the comparison operator $\diamond$ in $P : x \diamond f(\boldsymbol{t})$ is from $\{<, \leq\}$. The other two cases $\{>, \geq\}$ are symmetric.

Again let $g = [\![f(\boldsymbol{t})]\!] : \mathbb{R}^k \to \mathbb{R}$ be the function represented by the term $f(\boldsymbol{t})$. We assume that $g \in \mathcal{F}_{\text{DA}}$. Let $c_x = [\![x]\!]^{\alpha} \in \mathbb{Q}$ and $\boldsymbol{c_y} = [\![\boldsymbol{y}]\!]^{\alpha} \in \mathbb{Q}^k$ be the values assigned by α to the free variables $\boldsymbol{y}$ in $\boldsymbol{t}$, additionally, let $\boldsymbol{c_y} \in \operatorname{dom} g$. Furthermore let $c_g = g(\boldsymbol{c_y}) = [\![f(\boldsymbol{t})]\!]^{\alpha} \in \mathbb{R}$ be the value resulting from an exact evaluation of g. Note that c_g will only be used in the discussion and will not be added to the constraints, since in general $c_g \notin \mathbb{Q}$. Then our assumption of an existing conflict $[\![P]\!]^{\alpha} = \mathsf{false}$ can be read as $c_x > c_g$ for $\diamond \in \{\leq\}$, and as $c_x \geq c_g$ for $\diamond \in \{<\}$. Let us note that c_x and $\boldsymbol{c_y}$ are rational, but c_g is a real number and usually irrational. Since $g \in \mathcal{F}_{\text{DA}}$ we can compute approximations $\bar{c}_g \in \mathbb{Q}$ to c_g with $|\bar{c}_g - c_g| \leq \varepsilon$ for any rational $\varepsilon > 0$ using Lemma 5.

We now give a list of possible linearisations of g, starting from trivial versions where we exclude just the conflicting point $(c_x, \boldsymbol{c_y})$ to more general linearisations excluding larger regions containing this point.

Point Linearisation: A trivial (α, P)-linearisation excluding the point $(c_x, \boldsymbol{c_y})$ is

$$(\boldsymbol{y} = \boldsymbol{c_y} \implies x \neq c_x)$$

Half-Line Linearisation: An (α, P)-linearisation excluding a closed half-line starting in c_x is

$$(\boldsymbol{y} = \boldsymbol{c_y} \implies x < c_x)$$

In the following we will develop more powerful linearisations with the aim to exclude larger regions of the search space.

For better linearisations, we can exploit additional information about the predicate P and the trail α, especially about the behaviour of g in a region around $\boldsymbol{c_y}$. This information could be obtained by a per-case analysis on $\mathcal{F}_{nl}$, or during run time using external algebra systems or libraries for exact real arithmetic or interval arithmetic on the extended real numbers $\mathbb{R} \cup \{-\infty, +\infty\}$. Our focus, however, is on the numerical and not the symbolical approach.

As we aim at linearisations, the regions should have linear rational boundaries, so we concentrate on finite intersections of half-spaces:

Definition 5. *An (open or closed) rational half-space $H \subseteq \mathbb{R}^k$ is the solution set of a linear predicate $\boldsymbol{a} \cdot \boldsymbol{y} \leq b$ or $\boldsymbol{a} \cdot \boldsymbol{y} < b$ for some $\boldsymbol{a} \in \mathbb{Q}^k$, $b \in \mathbb{Q}$. A rational polytope $R \subseteq \mathbb{R}^k$ is a finite intersection of rational half-spaces.*

Any such polytope R is a convex and possibly unbounded set and can be represented as the conjunction of linear predicates over the variables $\boldsymbol{y}$. Therefore the complement $\mathbb{R}^k \setminus R$ can be represented as a linear clause $\{L_i : i \in I\}$ denoting the predicate $\boldsymbol{y} \notin R$. For the ease of reading, instead of writing clauses like $\bigvee_{i \in I} L_i \vee D$ we will use $\boldsymbol{y} \in R \implies D$ in the following.

Since $g \in \mathcal{F}_{\mathrm{DA}}$ and approximable it follows that for any bounded rational polytope $R \subseteq \mathbb{R}^k$ in the domain of g we can find arbitrarily precise rational over-approximations (a, b) such that $g(R) \subset (a, b)$.

Interval Linearisation: Suppose we have $c_x \neq c_g$. By approximating c_g we compute $d \in \mathbb{Q}$ with $c_g < d < c_x$. The proof of Lemma 5 provides an initial rational polytope $R \in \mathbb{R}^k$ with $\boldsymbol{c_y} \in R$ such that $d \notin g(R)$. Then
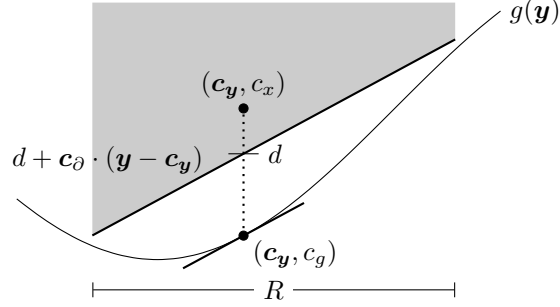
$$\boldsymbol{y} \in R \implies x \leq d \tag{5.1}$$

is an (α, P)-linearisation. Using specific properties of g, e.g., monotonicity, we can extend the polytope R to an unbounded one.

This linearisation excludes the set $\{x : x > d\} \times R$ from the search space which is a polytope, now in $\mathbb{R} \times \mathbb{R}^k$, containing the point $(c_x, \boldsymbol{c_y})$.

Linearisations in Example 4 are of this type, there $c_g = 1/c_y$ and $R = (1/d, \infty)$ defined by $y > 1/d$ which is the negation of $y \leq 1/d$, the second literal in the linear lemma $L_{\alpha,P}$ is the right hand side of the implication (5.1).

The univariate predicate $x \leq d$ corresponds to a very special half-space in $\mathbb{R} \times \mathbb{R}^k$, as it is independent from the variables in $\boldsymbol{y}$. Usually, using partial derivatives gives better linearisations:

Tangent Space Linearisation: Suppose we again have $c_x \neq c_g$. Assume the partial derivatives of g at $\boldsymbol{c_y}$ exist and we are able to compute a vector

Fig. 3. Tangent Space Linearisation shown for univariate g. The shaded area will be excluded from the search space.

$c_\partial = (c_1, \ldots, c_k)$ of rational approximations, that is, $c_i \approx \frac{\partial g}{\partial y_i}(c_y)$. As before we construct $d \in \mathbb{Q}$ with $c_g < d < c_x$ and search for a rational polytope $R \in \mathbb{R}^k$ with $c_y \in R$. But instead of just $d \notin g(R)$ now R has to fulfil the constraint

$$\forall r \in R : g(r) \leq d + c_\partial \cdot (r - c_y)$$

using the dot product of c_∂ and $(r - c_y)$. Again, R can be found using interval computation. Then

$$y \in R \implies x \leq d + c_\partial \cdot (y - c_y)$$

is an (α, P)-linearisation, since the dot-product is a linear and rational operation. This situation is schematically depicted in Figure 3.

Using the tangent space, we are able to get a much better 'fit' of $d + \sum c_i y_i$ to g than just using the naive interval evaluations. This allows to choose d closer to c_g for given R, or to choose a bigger polytope R for a given d. Some examples of Tangent Space Linearisations are available.[4]

Lemma 6. *By construction, the above procedures indeed provide linearisations as stated in Definition 2.*

For the rest of this section, we briefly discuss more specific linearisations for some important cases when we can perform a by-case analysis on g and exploit further properties like (piecewise) monotonicity, convexity or boundedness, which cannot be deduced by naive interval arithmetic, see Section 6 for details.

- $g(y) = y^{2n}$ is convex, with polytope $R = (-\infty, +\infty)$ for $\diamond \in \{>, \geq\}$.
- $g(y) = y^{2n+1}$ is monotonically increasing, with polytopes R of the form $(-\infty, c]$, similar to the linearisation in Section 4.
- Polynomials can be decomposed into monomials.
- Piecewise convex/concave functions g like $\sin, \cos, \tan$ allow polytopes covering a convex area in their domain.
- More direct ways of computing linearisations of the elementary transcendental functions can be obtained e.g. by bounding the distance of the image of specific g to algebraic numbers, such bounds are given in [19, section 4.3].

6 Evaluation

We implemented our approach in the ksmt system, which is open source and publicly available [4]. The ksmt system supports a subset of QF_LRA and QF_NRA logics as defined in the SMT-LIB standard. As with Z3, when no logic is specified in the input script, our extended signature $\mathbb{R}_{nl}$ is the default.

Choices made in the implementation include:

- Selecting a rational value in a non-empty interval as smallest dyadic or by continued fractions.
- The decision which clauses to resolve on conflict is guided by an internal SAT-solver.
- Heuristic about reusing existing constraints when computing polytope R, leading to piecewise linear approximations of g.
- Specialised linearisation algorithms for specific combinations of subclasses of functions $g \in \mathcal{F}_{\text{DA}}$ and $\boldsymbol{c_y}$:

  differentiable: Use Tangent Space Linearisation.

  convex/concave: Derive the polytope R from computability of unique intersections between g and the linear bound on $\boldsymbol{y}$.

  piecewise: This is a meta-class in the sense that $\text{dom}\, g$ is partitioned into $(P_i)_{i \in I}$ where the P_i are linear or non-linear predicates in $\boldsymbol{y}$, and for each $i \in I$ there is a linearisation algorithm, then the decision which linearisation to use is based on membership of the exact rational value $\boldsymbol{c_y}$ in one of the P_i.

  rational: Evaluate c_g exactly in order to decide on linearisation to use.

  transcendental: Bound $|c_x - c_g|$ by a rational from below by approximating c_g by the TTE implementation iRRAM[5] [24] in order to compute d.

We evaluated our approach over higher dimensional sphere packing benchmarks which are available at [4]. Sphere packing is a well known problem which goes back to Kepler's conjecture, and in higher dimensions is also of practical importance e.g., in error correcting codes. The purpose of this evaluation is to exemplify that our approach is viable and can contribute to the current state-of-the-art, extensive evaluation is left for future work.

The solvers[6] were compiled with GCC-8.2 according to their respective documentation (except for mathsat, which is not open-source). Experiments were run on a machine with 32 GiB RAM, 3.6 GHz Core i7 processor and Linux 3.18.

Example 1 (Sphere packing). Let $n, d \in \mathbb{N}$ and let

$$K_{n,d} := \exists \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d : \bigwedge_{1 \le i \le n} \|\boldsymbol{x}_i\|_\infty \le 1 \wedge \bigwedge_{1 \le i < j \le n} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 > 2$$

---

[4] http://informatik.uni-trier.de/~brausse/ksmt

[5] http://irram.uni-trier.de

[6] ksmt-0.1.3, cvc4-1.6+gmp, z3-4.7.1+gmp, mathsat-5.5.2, yices-2.6+lpoly-1.7, dreal-v3.16.08.01, rasat-0.3

s: 'sat'
δ: 'δ-sat', $\delta = 10^{-3}$
u: 'unsat'
?: 'unknown'
>: timeout

d	n	ksmt	cvc4	z3	mathsat	yices	dreal	rasat
2	2	s 0.01s	? 0.03s	s 0.01s	s 0.02s	s 0.01s	δ 0.01	s 0.02
	3	s 0.03s	? 0.08s	> 60m	s 0.24s	s 0.03s	δ 0.02	> 8h
	4	> 8h	u 1474.16s	> 60m	u 8.11s	> 17h	δ 0.05	> 8h
	5	u 1.43s	u 0.45s	> 8h	u 0.28s	> 8h	u 3581.96	> 8h
	6	u 5.00s	u 0.75s	> 8h	u 0.40s	> 166m	> 8h	> 8h
3	5	s 0.93s	? 465.45s	> 8h	s 0.12s	s 0.06s	> 8h	> 8h
	6	s 6.02s	> 143m	> 7h	> 8h	> 6h	> 8h	> 8h
4	5	s 0.38s	? 1544.87s	s 2165.78s	s 0.10s	s 7.34s	> 8h	> 8h
	6	s 0.57s	> 91m	> 8h	s 0.23s	s 0.38s	> 8h	> 8h
	7	s 14.27s	> 160m	> 8h	s 0.18s	> 8h	> 8h	> 8h

Table 1. Benchmarks of $K_{n,d}$ for different n, d.

r	ksmt	cvc4	z3	mathsat	yices	dReal	raSAT
$\sqrt{37}$	u 0.07s	u 0.76s	u 510.67s	u 40.55s	u 0.07s	u 0.01	> 8h
$\sqrt{49}$	u 0.40s	u 2.46s	u 23211.20s	u 6307.18s	u 0.11s	u 0.03	> 8h
$\sqrt{62}$	u 11.61s	u 5.07s	u 210.16s	> 14.5h	u 76.82s	u 2.00	> 8h
$\sqrt{63}$	u 55.84s	? 0.48s	u 3925.65s	> 14.5h	u 0.10s	u 12.38	> 8h
$\sqrt{64}$	s 0.01s	? 0.01s	s 0.00s	> 21.6h	s 0.00s	δ 0.01	> 8h

Table 2. Benchmarks of C_r for different r.

An instance $K_{n,d}$ is sat iff n balls fit into a d-dimensional box of radius 2 without touching each other. In the SMT-Lib language the $\|\cdot\|_\infty$ norms in these instances are formulated using per-component comparisons to the lower and upper endpoints of the range, while the euclidean norms $\|\boldsymbol{s}\|_2 > t$ are expressed by the equivalent squared variant $\sum_i \boldsymbol{s}_i^2 > t^2$. Table 1 provides a comparison of different solvers on instances of this kind.

Example 2. Let $r \in \mathbb{Q}$, then

$$C_r := \exists \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3 : \|\boldsymbol{x}\|_2^2 \le r^2 \wedge \|\boldsymbol{y}\|_2^2 \ge 8^2 \wedge \|\boldsymbol{x} - \boldsymbol{y}\|_\infty \le \tfrac{1}{100}.$$

C_r is sat for some $r \in [0, 8]$ iff there is a translation of the center $\boldsymbol{x}$ of the 3-dimensional ball $B_r(\boldsymbol{x})$ in a box of radius $\frac{1}{100}$ such that it intersects the complement of $B_8(\boldsymbol{y})$. Since the constraints are expressed as square-root-free expressions, obviously for $r \ge 8 - \frac{1}{100}$, there is a solution. Table 2 list running times for various r of our solver and other solvers of non-linear real arithmetic.

Noteworthy about these benchmarks is the monotonicity of the running times of ksmt in contrast to e.g. yices in conjunction with unlimited precision, which seems to be what prevents cvc4 from deciding the instance for $r = \sqrt{63}$ and even $r = \sqrt{64}$.

These experiments show that already in the early stage of the implementation, our system can handle high dimensional non-linear problems which are challenging for most SMT solvers.

7 Conclusions and future work

In this paper we presented a new approach for solving non-linear constraints over the reals. Our `ksmt` calculus combines model-guided solution search with targeted linearisations for resolving non-linear conflicts. We implemented our approach in the `ksmt` system, our preliminary evaluation shows promising results demonstrating viability of the proposed approach.

For future work we are developing more precise linearisations for specific trigonometric functions and are analyzing the complexity of deciding conflicts in general. We are working on extending the applicability of our implementation and a more extensive evaluation. We are also investigating theoretical properties of our calculus, such completeness in restricted settings and δ-completeness.

References

1. F. Benhamou and L. Granvilliers. Continuous and interval constraints. In *Handbook of Constraint Programming*, pages 571–603. Elsevier, 2006.
2. V. Brattka, P. Hertling, and K. Weihrauch. A tutorial on computable analysis. In *New Computational Paradigms*, pages 425–491. Springer, 2008.
3. B. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *ACM SIGSAM Bulletin*, 10(3):19–29, 1976.
4. A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani. Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. *ACM Trans. Comput. Log.*, 19(3):19:1–19:52, 2018.
5. G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, pages 134–183, 1975.
6. L. M. de Moura and D. Jovanovic. A model-constructing satisfiability calculus. In *Proc. VMCAI'2013, LNCS v. 7737*, pages 1–12, 2013.
7. A. Dolzmann and T. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.
8. I. Dragan, K. Korovin, L. Kovács, and A. Voronkov. Bound propagation for arithmetic reasoning in Vampire. In *Proc. SYNASC' 2013*, pages 169–176. IEEE, 2013.
9. P. Fontaine, M. Ogawa, T. Sturm, V. Khanh To, and X. Tung Vu. Wrapping Computer Algebra is Surprisingly Successful for Non-Linear SMT. In *SC-square 2018*, Oxford, United Kingdom, July 2018.
10. M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT*, 1(3-4):209–236, 2007.
11. S. Gao, J. Avigad, and E. M. Clarke. δ-complete decision procedures for satisfiability over the reals. In *IJCAR' 2012, LNCS v. 7364*, pages 286–300, 2012.
12. M. Hladík and S. Ratschan. Efficient solution of a class of quantified constraints with quantifier prefix exists-forall. *Math. in Comp. Sc.*, 8(3-4):329–340, 2014.
13. D. Jovanovic and L. de Moura. Solving non-linear arithmetic. In *IJCAR'2012, LNCS v. 7364*, pages 339–354, 2012.

14. D. Kapur, Y. Sun, and D. Wang. A new algorithm for computing comprehensive gröbner systems. In *Proc. ISSAC '10*, pages 29–36, New York, USA, 2010. ACM.

15. K. Korovin, M. Kosta, and T. Sturm. Towards conflict-driven learning for virtual substitution. In *Proc. CASC 2014*, volume 8660 of *LNCS*, pages 256–270, 2014.

16. K. Korovin, N.Tsiskaridze, and A. Voronkov. Implementing conflict resolution. In *Proc. PSI'2011*, volume 7162 of *LNCS*, pages 362–376, 2012.

17. K. Korovin, N. Tsiskaridze, and A. Voronkov. Conflict resolution. In *CP'09, LNCS v.5732*, pages 509–523, 2009.

18. K. Korovin and A. Voronkov. Solving systems of linear inequalities by bound propagation. In *CADE-23*, volume 6803 of *LNCS*, pages 369–383, 2011.

19. V. Lefévre. *Moyens arithmetiques pour un calcul fiable*. PhD thesis, École normale supérieure de Lyon, 2000.

20. R. Loos and V. Weispfenning. Applying linear quantifier elimination. *THE Computer Journal*, 36(5):450–462, 1993.

21. A. Maréchal, A. Fouilhé, T. King, D. Monniaux, and M. Périn. Polyhedral approximation of multivariate polynomials using Handelman's theorem. In *Proc. VMCAI'2016, LNCS v. 9583*, pages 166–184, 2016.

22. J. P. Marques-Silva and K. A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5):506–521, 1999.

23. K. L. McMillan, A. Kuehlmann, and M. Sagiv. Generalizing DPLL to richer logics. In *Proc. CAV'09 LNCS v. 5643*, pages 462–476. Springer-Verlag, 2009.

24. N. T. Müller. The iRRAM: Exact arithmetic in C++. In *Proc. CCA'2000, LNCS v. 2064*, pages 222–252, 2001.

25. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, 2006.

26. I. Niven. *Irrational Numbers*. Mathematical Association of America, 1956.

27. G. O. Passmore, L. C. Paulson, and L. M. de Moura. Real algebraic strategies for MetiTarski proofs. In *AISC/MKM/Calculemus*, LNCS v. 7362, pages 358–370, 2012.

28. G. Reger, N. Bjorner, M. Suda, and A. Voronkov. AVATAR modulo theories. In C. Benzmüller, G. Sutcliffe, and R. Rojas, editors, *2nd Global Conference on Artificial Intelligence*, EPiC Series in Computing vol. 41, pages 39–52. EasyChair, 2016.

29. A. Reynolds, C. Tinelli, D. Jovanović, and C. Barrett. Designing theory solvers with extensions. In *Proc. FroCoS '17, LNAI v. 10483*, pages 22–40, 2017.

30. D. Richardson. Some undecidable problems involving elementary functions of a real variable. *J. Symb. Log.*, 33(4):514–520, 1968.

31. A. Tarski. A decision method for elementary algebra and geometry. In *2nd ed. Univ. Cal.*, 1951.

32. K. Weihrauch. *Computable analysis: an introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.