Symbolic Computation for All the Fun

Chad E. Brown[1], Mikoláš Janota[1] and Mirek Olšák[2]

[1]*Czech Technical University in Prague, CIIRC, Czechia*
[2]*University of Cambridge*

Abstract
Motivated by the recent 10 million dollar AIMO challenge, this paper targets the problem of finding all functions conforming to a given specification. This is a popular problem at mathematical competitions and it brings about a number of challenges, primarily, synthesizing the possible solutions and proving that no other solutions exist. Often, there are infinitely many solutions and then the set of solutions has to be captured symbolically. We propose an approach to solving this problem and evaluate it on a set of problems that appeared in mathematical competitions and olympics.

Keywords
AIMO, synthesis, quantifier elimination, SMT

1. Introduction

The *AIMO challenge*[1] promises to award \$10 million to an AI model that will be able to win a gold medal at the *International Mathematical Olympiad (IMO)*[2]. In this paper we draw attention to the fact that symbolic computation tools can give a significant boost to a system aiming at this challenge. We target a popular problem encountered in mathematical competitions, which is finding *all* functions adhering to a certain specification. The task in its generality is nearly impossible but the problems in the competitions are constructed so that they have nice solutions. As a motivational example, consider the problem that asks us to find all functions f from $\mathbb{R}$ to $\mathbb{R}$ satisfying the following identity.

$$\forall xy : \mathbb{R}. f(x+y) = xf(y) + yf(x) \tag{1}$$

Substituting y with 0 gives $\forall x : \mathbb{R}. f(x) = f(0)x$, which tells us that f must be linear and that it is fully determined by the value of $f(0)$. Further, substituting x with 0 shows that f has to be 0 everywhere, i.e., the solution is $fx \triangleq 0$. A solution is expected to give a clear description of how all possible f are calculated, possibly parametrized by constants. To verify the correctness of the solution, one needs to prove the following biimplication.

$$(\forall xy : \mathbb{R}. f(x+y) = xf(y) + yf(x)) \Leftrightarrow (\forall x : \mathbb{R}. fx = 0) \tag{2}$$

Note that the problem of finding all solutions is not fully formal in the problem statement, that is, we expect a "reasonable description" of the set of all the solutions. For example, the following would *not* be considered a valid answer: The solution consists of all the functions satisfying $f(x+y) = xf(y)+yf(x)$ for all $x, y \in \mathbb{R}$.

A problem of such a form "find all solutions satisfying X" sparked some debate within the community about how they should be formally handled. How exactly we handle such question is explained in Sections 3.

SC2 Workshop 2024: Satisfiability Checking and Symbolic Computation
mikolas.janota@gmail.com (M. Janota)
http://people.ciirc.cvut.cz/~janotmik/ (M. Janota)
0000-0003-3487-784X (M. Janota); 0000-0002-9361-1921 (M. Olšák)

[1]https://aimoprize.com/
[2]https://www.imo-official.org/

2. Problems Description

As a source for our problems we have used a collection compiled by Vít Musil [1] comprising problems from several sources, including international competition and some (easier) problems from the The Prague Seminar in Mathematics[3]. We have transcribed the problems into SMT2, where each original problem is divided into 3 type of queries.

- The satisfiability of the specification if there is a solution (problems `find`).
- The unsatisfiability of the specification together with the negation of all proposed solutions (problems `prove`).
- The check that every proposed solution is indeed a solution to the specifications (problems `check`).

This way, we are asking SMT solvers to solve particular steps, although these steps are not necessarily covering the full solution of a functional equation. We are not providing a formal description of what a valid solution is allowed to consist of (that could be further work using for example SyGuS). Rather, we use our own method for finding the set of all solutions of a particular form, described in Section 3. The SMT2 problems are made available as a github repository[4] describing 87 problems. In some cases, the `check` query is split into multiple SMT2 files, if there are multiple distinct solutions.

We remark that during this work we uncovered several bugs in our translation to SMT from the source material. But also, we have found an issue in the original source material. Namely the problem U9 (Cvičení 9) was unsatisfiable in the original [1] and we created two versions of the problem as two different ways of correcting it (`C9` and `C9a`).

2.1. Solutions of Selected Problems

We show here three more example problems together with their original (human) solutions:

- Problem U24, which is among the most interesting ones that we were able to fully solve;
- Problem C12, which we were not able to solve but it does not require any advanced technique, so it could be feasible to solve it;
- Problem U2, which showcases a problem requiring induction to be solved, and we assume is currently out of reach.

Example Problem U24 (Baltic Way 1998-7): Find all functions $f : \mathbb{R} \to \mathbb{R}$ such that for any pair of real numbers x, y, the following identity holds

$$f(x) + f(y) = f(f(x)f(y)).$$

Solution: First we notice that the image of f is closed under addition: if a, b are the values of f at points x, y, then also $a + b$ is a value of f, in particular at point $f(x)f(y) = ab$. Fix arbitrary x_1, and denote $a = f(x_1)$. By the previous observation, we can find also x_2 and x_4 such that $f(x_2) = 2a$, and $f(x_4) = 4a$.

Now, we plug into the functional equation the assignments $(x = x_2, y = x_2)$, and $(x = x_1, y = x_4)$.

$$\begin{aligned} 4a &= f(x_2) + f(x_2) = f(f(x_2)f(x_2)) = f(4a^2) \\ 5a &= f(x_1) + f(x_4) = f(f(x_1)f(x_4)) = f(4a^2) \end{aligned}$$

We conclude $4a = 5a$, and consequently $a = 0$. Since we started with arbitrary $f(x) = a$, the function must be constant zero, which indeed satisfies the equation.

[3]https://prase.cz/info/info_en.php?lang=1

[4]https://github.com/MikolasJanota/FuncProbs

Example Problem C12 (IMO 1992-2): Find all functions $f : \mathbb{R} \to \mathbb{R}$ such that for any pair of real numbers x, y, the following identity holds

$$f(x^2 + f(y)) = y + f(x)^2.$$

Solution: Setting $x = 0$ gives $f(f(y)) = y + f(0)^2$. The right-hand side $y + f(0)^2$ is a bijection, so also the left hand side $f \circ f$ is a bijection $\mathbb{R} \to \mathbb{R}$, and that can only happen when f itself is a bijection.

Since f is a bijection, we can consider an argument b such that $f(b) = 0$. Let us substitute $(x = b, y = 0)$ and $(x = -b, y = 0)$ to the original equation.

$$0 = 0 + f(b)^2 = f(b^2 + f(0)) = f((-b)^2 + f(0)) = 0 + f(-b)^2 = f(-b)^2.$$

We obtained $f(-b) = 0$. By injectivity, $-b = b$, so $b = 0$, and $f(0) = 0$. This also simplifies our first observation to $f(f(y)) = y$.

Now we prove that the function is increasing. Any pair of numbers $a < b$ can be expressed as $a = f(y), b = x^2 + f(y)$ for some $x, y, x \neq 0$. Combining the original equation with the same equation having substituted $x = 0$, we get

$$f(a) = f(f(y)) = y < y + f(x)^2 = f(x^2 + f(y)) = f(b),$$

concluding that f is increasing.

Consider any y where $y \leq f(y)$. Since f is increasing, also $f(y) \leq f(f(y)) = y$. Analogously, if $f(y) \leq y$, we obtain $y = f(f(y)) \leq f(y)$. Thus the statements $y \leq f(y)$ and $f(y) \leq y$ are equivalent for any y, leaving the only option $f(y) = y$, which is a function satisfying the given equation.

Example Problem U2 (Cauchy equation): Find all increasing functions $f : \mathbb{R} \to \mathbb{R}$ such that for any pair of real numbers x, y, the following identity holds

$$f(x) + f(y) = f(x + y).$$

Proof sketch: By fixing $x = 1$, and setting $y = 0, 1, 2, \ldots$, we obtain by induction that $f(nx) = n \cdot f(x)$ for every non-negative integer n, in particular $f(0) = 0$, and if we denote $f(1) = c$, we obtain the following equation for all non-negative integers x.

$$f(x) = cx. \tag{$\clubsuit$}$$

We gradually extend the domain for which we know this equation. First, substitution $(x = x, y = -x)$ into the original equation gives $f(x) = -f(-x)$ which extends ($\clubsuit$) to all integers x. Let x be an arbitrary number satisfying ($\clubsuit$), and n be a positive integer. Then

$$cx = f(x) = f\left(n \cdot \frac{x}{n}\right) = n \cdot f\left(\frac{x}{n}\right)$$

leading to

$$c \cdot \frac{x}{n} = f\left(\frac{x}{n}\right).$$

Therefore, ($\clubsuit$) holds for any rational number x.

Finally, we show that ($\clubsuit$) must be satisfied by every real number. Suppose on the contrary that for some x, we have for example $f(x) < cx$, or $f(x) > cx$. We discuss here the first option, the other one is analogous. If $f(x) < cx$, there is a rational number q such that $\frac{f(x)}{c} < q < x$ (note $c > 0$ since f is increasing). Then $q < x$ but $f(x) < qc = f(q)$ contradicting that f is increasing.

3. Template-and-QE

One could see the problem as quantifier elimination in the theory of uninterpreted functions, accompanied by the theory required for describing the problem itself — in our case, the theory of reals or rationals. Due to the undecidability of such a problem, quantifier elimination algorithms cannot be used directly. A possible approach would be to *synthesize* one function adhering to the specification and then strengthen the specification so that the individual solution is not admitted anymore. Then, one would have to repeat this process with the hope that there are finitely many solutions, or, observe the solutions and generalize them into a more compact description. All these steps are highly nontrivial. Instead, we propose an approach that tries a fixed template and then performs quantifier elimination: *template-and-QE*, which comprises the following subtasks.

1. *Identify* a template for the solution, e.g. $fx \triangleq ax + b$.
2. *Prove* that all solutions must fall into this template (*template verification*)
3. Perform *quantifier elimination* over input variables of the function(s).

Note that in the case of reals, all tasks are computable, except for task 2. The remainder of this section elaborates on the individual steps.

3.1. Template Verification

We run cvc5 [2], z3 [3], Vampire [4] and Waldmeister [5] to attempt to prove all solutions to a problem necessarily fit a certain template. Since Waldmeister does not support theories directly, it needs to special treatment and we elaborate on this below. The other solvers support natively the SMT2 syntax [6] enabling us to use the combination of quantified non-linear reals with unintepreted functions.[5]

We consider the following set of templates for solutions: *constant*, *(monomial)linear*, *(monomial) quadratic*. All these are subclasses of the quadratic form but we consider these to simplify the task for the solvers—it is conceivable that it is easier to prove that the function must be constant than to prove that it must be arbitrary quadratic. Since the set of templates is small, we always try all of them.

In order to ask an SMT2 solver if all solutions must be linear, we add the assertion that the function is not linear. An obvious way to state that f is linear is via the formula $\exists ab : \mathbb{R}.\forall x : \mathbb{R}.f(x) = ax + b$. We call the problem resulting from including the properties of the original problem along with the negation of $\exists ab : \mathbb{R}.\forall x : \mathbb{R}.f(x) = ax + b$ the *first variant of the linear template verification*. We had more success using a different test for linearity. A function f is linear if and only if $\forall x : \mathbb{R}.f(x) = (f(1) - f(0))x + f(0)$. Note that this formulation avoids the use of existential quantifiers. We call the problem resulting from including the properties of the original problem along with the negation of $\forall x : \mathbb{R}.f(x) = (f(1) - f(0))x + f(0)$ *the second variant of the linear template verification*. If any SMT2 solver can determine either variant is unsatisfiable, then we know all solutions must be linear.

We likewise have two variants for each of the other templates. The formulations for the first variants simply use existential quantification for the coefficients. For the second variants we use the following properties:

- *constant*: $\forall x : \mathbb{R}.f(x) = f(0)$
- *monomial linear*: $\forall x : \mathbb{R}.f(x) = f(1)x$
- *linear*: $\forall x : \mathbb{R}.f(x) = (f(1) - f(0))x + f(0)$
- *monomial quadratic*: $\forall x : \mathbb{R}.f(x) = f(1)x^2$
- *quadratic*: $\forall x : \mathbb{R}.2f(x) = ((f(1) + f(-1)) - 2f(0))x^2 + (f(1) - f(-1))x + 2f(0)$

3.1.1. Using Waldmeister

In addition to using SMT2 solvers, we also used Waldmeister [5]. Waldmeister is an automated theorem prover specializing in first-order unit equality. Unlike SMT solvers, Waldmeister has no information

[5]The standard does not list UFNRA as one of the logics, therefore the files indicate AUFNIRA as the closest superset.

about integers or reals. Consequently, in order to ask Waldmeister if all functions f satisfying some properties must be contained within the class given by a template, we first declare a sort R along with constants $0, 1 : R$ and operations $+, -, \cdot$ satisfying properties of a commutative ring with identity. Each of these properties is given by a unit equation. We purposely use rings instead of fields (although $\mathbb{R}$ is a field). This allows us to ignore division since the side condition that the denominator is not zero would result in a clause that is not a unit equation. We only generate such a problem for Waldmeister when all the properties stated for f in the problem are unit equations, only quantify over reals (as opposed to integers), and do not use division. Furthermore, we require that all specific real numbers mentioned in the properties correspond to integers (e.g., 0.0, 1.0, -1.0, etc.). These restrictions allow us to formulate a unit equation to give Waldmeister corresponding to each property of f given in the problem.

In addition to the assumed unit equations, Waldmeister expects a unit equation a as goal to prove. The goal varies based on the template we are targeting and corresponds to the second variant of the problems for the SMT2 solver. In each case we fix a constant d of type R (not mentioned in the assumptions). For each template, the goal unit equation is as follows:

- *constant*: $f(d) = f(0)$
- *monomial linear*: $f(d) = f(1)d$
- *linear*: $f(d) = (f(1) - f(0))d + f(0)$
- *monomial quadratic*: $f(d) = f(1)d^2$
- *quadratic*: $2f(d) = ((f(1) + f(-1)) - 2f(0))d^2 + (f(1) - f(-1))d + 2f(0)$

Waldmeister uses Knuth-Bendix style completion [7] on the assumed unit equations finding a proof of the goal when both sides of the goal rewrite to a common term.

Waldmeister can prove 8 of the problems only have solutions within the expected class. Two of these problems were not covered by SMT solvers: U25 and U87. In both of these problems, the least template class is linear monomials. We briefly discuss these two problems.

The problem U25 asks for all real functions f satisfying $f(xf(x) + f(y)) = y + f(x)^2$. Musil [1] gives a proof that the only two solutions are $f(x) = x$ and $f(x) = -x$. Waldmeister's goal is to prove $f(d) = f(1)d$ from the ring identities and the equation above. It is difficult to directly compare the aforementioned human proof to the proof found by Waldmeister. However, there are some interesting common intermediate results. Both proofs prove f is involutive (i.e., $f(f(x)) = x$), $f(x)^2 = x^2$ and $f(0) = 0$. On the other hand, there are several identities in each proof that do not appear in the other. Musil's proof [1] proves f is surjective and uses this fact, while the Waldmeister proof cannot directly represent the concept of surjectivity.

The other problem Waldmeister can prove only has linear monomial solutions is U87. The problem U87 asks for all real functions f satisfying $f(x + y^2 + z) = f(f(x)) + yf(y) + f(z)$. The (only) two solutions are $f(x) = x$ and $f(x) = 0$. In this case there is no corresponding proof in [1] as a point of comparison. A number of intermediate identities derived by Waldmeister stand out, e.g., $f(f(0)) = 0$ and $f(f(x)) = f(x)$.

3.2. Quantifier Elimination

Quantifier elimination (QE) is a method to take a general formula in a theory and produce an equivalent quantifier-free formula. So for instance, in $\exists x \in \mathbb{R}. a < x < b$, the quantified variable x is eliminated as $a < b$. In many cases, quantifier elimination serves as a theoretical tool to show that a theory is decidable but also has a long tradition of improvements at the algorithmic level [8, 9, 10, 11, 12, 13, 14]. In our work, quantifier elimination is used in a practical way to reveal the particular solutions within a class given by a template.

We assume the original problem only has one uninterpreted function, f, and it is a unary function from reals to reals. If we want to determine all functions f of the form $f(x) = ax + b$, we can simply inline f in the properties, replacing each occurrence $f(t)$ by $at + b$. The resulting problem has no uninterpreted functions and is usually in the theory of the reals. (Exceptional cases are when the properties mention integers as well as reals.) When the problem and template result in such an inlined problem in the theory of reals, we can apply quantifier elimination.

We have made use of two implementations of quantifier elimination for the reals: z3 [3] and Tarski [15]. In z3, QE is invoked by applying the `qe` tactic, with the `qe-nonlinear` parameter turned on. The Tarski system is highly configurable but since QE is not a bottle-neck for us, we use the `qepcad-qe` [16] function in its default setting. For each system, quantifier elimination should result in a quantifier-free formula involving the uninterpreted constants (e.g., the coefficients a and b of the linear template $ax+b$). One might expect this formula to be in a solved form, e.g., $a = 1 \vee a = 0 \wedge b = 1$, from which one can read of the precise class of linear functions satisfying the original property. However, this is generally not the case. A separate postprocessing step attempts to convert the quantifier-free formula into such a solved form. In practice the postprocessing does not always succeed. In the case of z3, we also call the tactics simplify and propagate-values to have z3 simplify the formula before applying the postprocessing to attempt to find a solved form.

Problem U91 asks for all functions satisfying $f((x-y)^2) = f(x)^2 - 2xf(y) + y^2$. Using the techniques of Section 3.1, we can prove all solutions f must be linear. After replacing $f(x)$ with $ax + b$ and calling z3 to do quantifier elimination, we obtain the quantifier-free formulas $a = 1$ and $b = 0 \vee b = 1$. In general, z3 returns several formulas whose conjunction should be equivalent to the original formula. The formula $a = 1 \wedge (b = 0 \vee b = 1)$ is almost in solved form, and we can easily obtain the two solutions $f(x) = x$ and $f(x) = x + 1$ via postprocessing. Tarski's quantifier elimination returns the more complicated formula

$$-1 + a = 0 \wedge b \geq 0 \wedge -1 + b \leq 0 \wedge (b = 0 \vee -1 + b = 0).$$

The postprocessor is also able to obtain the two solutions from this formula.

Sometimes there is a parameterized class of solutions. Problem U3, for example, asks for all functions satisfying $f(x + y) = f(x) + y$. The techniques of Section 3.1 determine all solutions are linear. After inlining $f(x) = ax + b$ we call quantifier elimination. Both Z3 and Tarski produce the formula $-1 + a = 0$. From this the postprocesser can determine $a = 1$ and b is unconstrained. This gives the class of solutions $f(x) = x + b$ where b is a real number. Indeed, this is the class of all solutions, as desired.

3.2.1. Lazy Verification

An alternative to the approach outlined above, the template of the solution does not have to be verified all at once. Instead, we could use QE to find a class of solutions for a fixed template and then prove there are no other solutions. The advantage of this approach is illustrated by the following example.

Problem C1 asks for all functions f such that

$$f(x + y) + 2f(x - y) - 4f(x) + xf(y) = 3y^2 - x^2 - 2xy + xy^2.$$

The only solution is $f(x) = x^2$.

In this case, the solvers listed in Section 3.1 are unable to prove all solutions are a monomial quadratic (or that they must be quadratic). However, we can still apply quantifier elimination to find all monomial quadratic solutions. For this example, z3 timed out even when given 10 minutes. On the other hand, Tarski returned the formula $-1 + a = 0$ (yielding the solution $a = 1$, i.e., x^2) in less than a second. Since we were unable to prove all solutions must belong to the monomial quadratic template class, it is still possible there are more solutions outside the monomial quadratic template class. However, now that we have the specific solution $f(x) = x^2$ we can ask an SMT2 solver to prove this is the only solution by assuming the equation and also the negation of $\forall x : \mathbb{R}. f(x) = x^2$. The solver cvc5 is easily able to show this is unsatisfiable, so that we know $f(x) = x^2$ is the only solution.

Table 1
Result summary. Problems without any result are omitted.

(a) Templates of solutions. Proven template is denoted by ✓; disproven template is denoted by $\times$; dash means that no solver provided an answer.

Problem	c	ax	$ax+b$	ax^2	ax^2+bx+c
C2	×	×	✓	×	✓
C6	×	×	-	×	-
C9a	-	-	-	✓	-
C10	×	✓	✓	×	✓
C12	×	-	-	×	-
C13	×	×	×	×	×
U2	×	-	-	×	-
U3	×	×	✓	×	✓
U5	×	✓	✓	×	✓
U7	×	×	-	×	-
U9	×	×	✓	×	✓
U13	×	✓	✓	×	✓
U16	×	×	✓	×	-
U20	×	×	-	×	-
U23	-	×	-	×	-
U24	✓	✓	✓	✓	✓
U25	×	✓	✓	×	-
U26	×	-	-	×	-
U27	×	×	×	×	×
U41	-	×	-	×	-
U42	×	×	-	×	-
U44	×	-	-	-	-
U45	×	-	-	×	-
U48	×	-	-	×	-
U49	×	-	-	×	-
U50	×	-	-	×	-
U51	×	×	-	×	-
U54	×	×	×	×	×
U56	×	×	-	×	-
U62	×	-	-	×	-
U64	×	-	-	×	-
U67	×	×	×	×	×
U68	×	-	-	×	-
U71	✓	✓	✓	✓	✓
U72	-	×	-	×	-
U87	×	✓	✓	×	✓
U90	×	×	×	×	-
U91	×	×	✓	×	✓
U92	×	-	-	×	-

(b) Prove/check

Problem	Prove	Check
C1	✓	✓
C2	✓	✓
C3	-	✓
C4	-	✓
C5	✓	✓
C6	-	✓
C9	✓	✓
C9a	✓	✓
C10	✓	✓
C12	-	✓
C13	-	✓
U2	-	✓
U3	✓	✓
U4	✓	✓
U5	✓	✓
U6	-	✓
U7	-	✓
U8	-	✓
U9	✓	✓
U10	-	✓
U11	✓	✓
U12	-	✓
U13	✓	✓
U14	-	✓
U16	✓	✓
U17	✓	✓
U19	-	✓
U20	-	✓
U23	-	✓
U24	✓	✓
U25	-	✓
U26	-	✓
U27	-	✓
U39	-	✓
U41	-	✓
U42	-	✓
U44	-	✓
U45	-	✓
U46	-	✓
U48	-	✓
U49	-	✓
U50	-	✓
U51	-	✓
U53	-	✓
U54	-	✓
U56	-	✓
U57	✓	✓
U61	-	✓
U62	-	✓
U64	-	✓
U66	-	✓
U67	-	✓
U68	-	✓
U71	✓	✓
U72	-	✓
U75	-	✓
U76	-	✓
U79	✓	✓
U81	✓	✓
U82	✓	✓
U87	-	✓
U89	-	✓
U90	-	✓
U91	-	✓
U92	-	✓
U93	-	✓

4. Experiments

We report results of the template-and-qe method (Section 3) on the benchmark presented in Section 2. The lazy extension of our approach (Section 3.2.1) was left for future work.

The template verification (Section 3.1) has proven to be the most difficult task. Table 1a summarizes the obtained results for template verification. For all these we were able to solve the QE task. Thirteen instances were solved completely by the automated method. Out of these thirteen, two can be traced to a competition event. Problem U24 comes from Baltic Way competition—see Section 2 for "human" solution. Problem U71 comes from The Prague seminar (PraSe-18-6-1). In both cases, the only solution is a function that is constantly 0.

The verification tasks results often in satisfiable problems, namely if there exists a solution outside of the proposed template (the $\times$ symbol in Table 1a). These satisfiable problems are nontrivial because they involve creating a counterexample to the template. For this purpose, we also ran the SyGuS approach[6] of cvc5 and its linear model builder [17].

We also report on the verification of the correctness of the handwritten solutions, which comprises two components: *prove* — show that all possible solutions are covered by the suggested solutions; *check* — check that all suggested solutions are indeed solutions to the problem. Table 1b summarizes the obtained results. Sixty-five of the provided handwritten solutions were successfully checked, i.e., the individual solutions satisfy the original specifications. Twenty of the handwritten solutions were shown to cover all the individual solutions. Unsurprisingly, the proving task is harder than the checking task. We remark that checking solutions for `U79` is trivial because there are no individual solutions. More detailed results can be found on the authors' web page.[7]

5. Conclusions and Future Work

This paper joins the effort of rising up to the challenge of making computers as powerful as are the golden medalist at the International Math Olympics (IMO). Efforts such as AlphaGeometry [18], show that machine learning models are useful for the task but at the same time, further research shows that they benefit from existing symbolic methods [19]. Here we show that symbolic methods are indeed already powerful in solving a highly nontrivial task of finding *all* functions fulfilling a certain specification. Besides the task leading to undecidable questions, its difficulty also lies in the fact it is *not* a decision problem but the response is a description of a class of mathematical objects.

The method we employ is anchored in templates, which is a well-known technique in function and program synthesis [20]. The templates we consider are rather simple and they could be made more powerful by adding conditionals (if-then-else). However, the templates need to be kept simple if we wish to be able to apply quantifier elimination.

An alternative to templates would be to attempt extending different synthesis approaches, e.g., such as those that are realized in saturation-based solvers [21, 22] or inside SMT solvers [23, 24, 25] or inductive logic programming [26]. The challenge here would be how to come up with all the solutions.

Acknowledgments

The results were supported by the Ministry of Education, Youth and Sports within the dedicated program ERC CZ under the project *POSTMAN* no. LL1902 and co-funded by the European Union under the project *ROBOPROX* (reg. no. CZ.02.01.01/00/22_008/0004590). C. Brown was supported by *CORESENSE*: the European Union's Horizon Europe research and innovation programme under grant agreement no. 101070254 CORESENSE. This article is part of the *RICAIP* project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857306.

References

[1] V. Musil, Funkcionální rovnice, 2024. URL: https://prase.cz/library/FunkcionalniRovniceVM/FunkcionalniRovniceVM.pdf, downloaded 8 March 2024.

[2] H. Barbosa, C. W. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, Y. Zohar, cvc5: A versatile and industrial-strength SMT solver, in: Tools and Algorithms for the Construction and Analysis of Systems, TACAS, volume 13243 of *LNCS*, Springer, 2022, pp. 415–442. doi:10.1007/978-3-030-99524-9_24.

[6] option `--sygus-inference`

[7] https://sat.inesc-id.pt/~mikolas/sc2_2024/

[3] L. M. de Moura, N. Bjørner, Z3: an efficient SMT solver, in: Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, volume 4963, Springer, 2008, pp. 337–340. doi:10.1007/978-3-540-78800-3_24.

[4] L. Kovács, A. Voronkov, First-order theorem proving and Vampire, in: N. Sharygina, H. Veith (Eds.), Computer Aided Verification - 25th International Conference, CAV, volume 8044 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 1–35. doi:10.1007/978-3-642-39799-8_1.

[5] T. Hillenbrand, B. Löchner, The next Waldmeister loop, in: A. Voronkov (Ed.), Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, volume 2392 of *Lecture Notes in Computer Science*, 2002, pp. 486–500. doi:10.1007/978-3-540-45085-6_27.

[6] C. Barrett, P. Fontaine, C. Tinelli, The SMT-LIB Standard: Version 2.6, Technical Report, Department of Computer Science, The University of Iowa, 2017. Available at www.SMT-LIB.org.

[7] L. Bachmair, N. Dershowitz, D. A. Plaisted, Completion without failure, in: Rewriting Techniques, Elsevier, 1989, pp. 1–30.

[8] G. E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decompostion, in: H. Brakhage (Ed.), Automata Theory and Formal Languages, Springer Berlin Heidelberg, Berlin, Heidelberg, 1975, pp. 134–183.

[9] D. C. Cooper, Theorem proving in arithmetic without multiplication, Machine intelligence 7 (1972) 300.

[10] J. Ferrante, C. Rackoff, A decision procedure for the first order theory of real addition with order, SIAM Journal on Computing 4 (1975) 69–76. URL: https://doi.org/10.1137/0204006. doi:10.1137/0204006.

[11] R. Loos, V. Weispfenning, Applying linear quantifier elimination, Comput. J. 36 (1993) 450–462. doi:10.1093/COMJNL/36.5.450.

[12] J. H. Davenport, Y. Siret, E. Tournier, Computer algebra - systems and algorithms for algebraic computation (2. ed.), Academic Press, 1993.

[13] A. R. Bradley, Z. Manna, The calculus of computation - decision procedures with applications to verification, Springer, 2007. doi:10.1007/978-3-540-74113-8.

[14] N. S. Bjørner, M. Janota, Playing with quantified satisfaction, in: LPAR, EasyChair, 2015, pp. 15–27. doi:10.29007/VV21.

[15] F. Vale-Enriquez, C. W. Brown, Polynomial constraints and unsat cores in Tarski, in: J. H. Davenport, M. Kauers, G. Labahn, J. Urban (Eds.), Mathematical Software - ICMS 2018 - 6th International Conference, South Bend, IN, USA, July 24-27, 2018, Proceedings, volume 10931 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 466–474. doi:10.1007/978-3-319-96418-8_55.

[16] G. E. Collins, H. Hong, Partial cylindrical algebraic decomposition for quantifier elimination, J. Symb. Comput. 12 (1991) 299–328. doi:10.1016/S0747-7171(08)80152-6.

[17] M. Janota, B. Piotrowski, K. Chvalovský, Towards learning infinite SMT models, 2023. URL: http://people.ciirc.cvut.cz/~janotmik/, 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC.

[18] T. H. Trinh, Y. Wu, Q. V. Le, H. He, T. Luong, Solving olympiad geometry without human demonstrations, Nature 625 (2024) 476–482. doi:10.1038/s41586-023-06747-5.

[19] S. Sinha, A. Prabhu, P. Kumaraguru, S. Bhat, M. Bethge, Wu's method can boost symbolic AI to rival silver medalists and AlphaGeometry to outperform gold medalists at IMO geometry, 2024. arXiv:2404.06405.

[20] S. Srivastava, S. Gulwani, J. S. Foster, Template-based program verification and program synthesis, Int. J. Softw. Tools Technol. Transf. 15 (2013) 497–518. doi:10.1007/S10009-012-0223-4.

[21] P. Hozzová, L. Kovács, C. Norman, A. Voronkov, Program synthesis in saturation, in: B. Pientka, C. Tinelli (Eds.), Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings, volume 14132 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 307–324. doi:10.1007/978-3-031-38499-8_18.

[22] P. Hozzová, D. Amrollahi, M. Hajdu, L. Kovács, A. Voronkov, E. M. Wagner, Synthesis of recursive programs in saturation, in: International Joint Conference on Automated Reasoning IJCAR, 2024.

[23] A. Reynolds, V. Kuncak, C. Tinelli, C. W. Barrett, M. Deters, Refutation-based synthesis in SMT,

Formal Methods Syst. Des. 55 (2019) 73–102. doi:10.1007/S10703-017-0270-2.

[24] A. Abate, H. Barbosa, C. W. Barrett, C. David, P. Kesseli, D. Kroening, E. Polgreen, A. Reynolds, C. Tinelli, Synthesising programs with non-trivial constants, J. Autom. Reason. 67 (2023) 19. doi:10.1007/S10817-023-09664-4.

[25] H. Barbosa, A. Reynolds, D. Larraz, C. Tinelli, Extending enumerative function synthesis via SMT-driven classification, in: C. W. Barrett, J. Yang (Eds.), 2019 Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA, October 22-25, 2019, IEEE, 2019, pp. 212–220. doi:10.23919/FMCAD.2019.8894267.

[26] D. M. Cerna, A. Cropper, Generalisation through negation and predicate invention, Proceedings of the AAAI Conference on Artificial Intelligence 38 (2024) 10467–10475. URL: https://ojs.aaai.org/index.php/AAAI/article/view/28915. doi:10.1609/aaai.v38i9.28915.