Decidability of Difference Logic over the Reals with Uninterpreted Unary Predicates

Bernard Boigelot[0009−0009−4721−3824], Pascal Fontaine[0000−0003−4700−6031], and Baptiste Vergain[✉][0009−0003−5545−4579]

Montefiore Institute, B28
Université de Liège, Belgium
{Bernard.Boigelot, Pascal.Fontaine, BVergain}@uliege.be

Abstract. First-order logic fragments mixing quantifiers, arithmetic, and uninterpreted predicates are often undecidable, as is, for instance, Presburger arithmetic extended with a single uninterpreted unary predicate. In the SMT world, difference logic is a quite popular fragment of linear arithmetic which is less expressive than Presburger arithmetic. Difference logic on integers with uninterpreted unary predicates is known to be decidable, even in the presence of quantifiers. We here show that (quantified) difference logic on real numbers with a single uninterpreted unary predicate is undecidable, quite surprisingly. Moreover, we prove that difference logic on integers, together with order on reals, combined with uninterpreted unary predicates, remains decidable.

Keywords: First-order logic · Decidability · SMT · Arithmetic · Uninterpreted predicates

1 Introduction

The success of satisfiability modulo theories (SMT) solvers in verification can be attributed to several things, but one of them is indisputably the omnipresence, in the combination of theories, of arithmetic reasoners. As SMT solvers get stronger in quantified reasoning, it becomes more interesting to get a clear picture of decidability frontiers when arithmetic is used in a quantified SMT context. Some pure arithmetic theories are already undecidable, even in their quantifier-free fragment, e.g., Peano arithmetic [12], i.e., a first-order theory of the natural numbers with addition and multiplication. However, Presburger arithmetic, somehow the linear restriction of Peano arithmetic, is decidable even in the quantified case [10], but augmenting Presburger arithmetic with a single unary uninterpreted predicate already yields undecidability [7, 11, 19]. To obtain a decidable fragment mixing arithmetic and uninterpreted predicates, one must further restrict the expressiveness.

In the SMT world, difference logic used to be a popular fragment of arithmetic, because of its low complexity in the quantifier-free case. In this fragment, arithmetic is limited to difference constraints of the form $x - y \bowtie c$ where x and y are variables, c is an integer constant and $\bowtie$ belongs to $\{<, \leq, =, \geq, >\}$.

Difference constraints can, e.g., express conditions on the distance between two variables, the atomic formula $x-y=2$ stating that the distance between the values of x and y must be exactly 2. Notice that since difference constraints involve only two variables (c is an integer constant) those constraints are strictly less expressive than linear constraints in Presburger arithmetic. The decidability of the logic mixing difference constraints and unary uninterpreted predicates, when interpreted over $\mathbb{N}$ (or similarly $\mathbb{Z}$) reduces to the decidability of the monadic second-order theory of one successor, usually referred to as S1S. The decidability of S1S has been established thanks to the concept of infinite-word automaton [4].

On the real domain, it is well known that the first-order theory of real-closed fields, which is in a sense the real counterpart of Peano arithmetic, is decidable [20] even in the presence of quantifiers. Whereas this might give the impression that decidability is more often obtained on the reals than on the integers, we here prove that the logic mixing difference constraints and unary uninterpreted predicates, when interpreted over $\mathbb{R}$, is undecidable.

Further restricting the arithmetic language, and considering order on the real domain only, it is known that the monadic second-order theory of order is undecidable [9, 17], but its universal fragment is decidable [5]. In this work, we establish that the fragment mixing unary uninterpreted predicates, difference constraints over integer variables, and order constraints over real variables is decidable.

Section 2 provides some prerequisites and the precise definition of the studied fragments. In Section 3, we prove the decidability of the fragment mixing unary uninterpreted predicates, difference constraints over integer variables, and order constraints over real variables. This was already the subject of a work-in-progress workshop paper [1]. In Section 4, we prove that the fragment of quantified difference constraints over real variables extended with a single unary uninterpreted predicate is undecidable.

2   Preliminaries

We refer to e.g., [8] for a general introduction to first-order logic with equality, and assume that the reader is familiar with the notions of signature, term, variable, and formula. We use the usual logical connectives ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) and first-order quantification $\exists x.\,\varphi$ and $\forall x.\,\varphi$, respectively equivalent to writing $\exists x\,(\varphi)$ and $\forall x\,(\varphi)$, i.e., the dot stands for an opening parenthesis that is closed at the end of the formula. Variable symbols are denoted by x, y, z, ... and are meant to be interpreted as real numbers.

Our signature contains the interpreted arithmetic symbols 0, 1, $+$, $-$, $<$, $\leq$, $\geq$, $>$, $=$, and other constants in $\mathbb{N}$ that stand for terms $1 + 1 + \cdots + 1$. We furthermore use a monadic (i.e., unary) interpreted predicate $x \in \mathbb{Z}$ to denote that x has an integer value. The signature also contains uninterpreted predicate symbols $P, Q, \ldots$ In the whole article, we only consider unary predicate symbols. Indeed, including binary uninterpreted predicates without restriction on first-order quantification directly yields undecidability. Our language is the set of all

well-formed formulas, in the usual sense, built using symbols from the signature. Further specific restrictions will be introduced later.

An interpretation specifies a domain (i.e., a set of elements), assigns a value in the domain to each free variable, and assigns relations of appropriate arity on the domain to predicate symbols in the signature. Throughout the article, the interpretation domain is always $\mathbb{R}$. The arithmetic symbols 0, 1, $+$, $-$, $<$, $\leq$, $\geq$, $>$, $=$ are interpreted as expected on $\mathbb{R}$, and $x \in \mathbb{Z}$ is true if and only if x has an integer value[2]. An interpretation assigns an arbitrary subset of the domain $\mathbb{R}$ to each unary predicate. By extension, an interpretation assigns a value in $\mathbb{R}$ to every term, and a truth value to every formula. We denote the interpretation I of a variable x by $I[x]$, and the interpretation of a predicate P by $I[P]$. A model of a formula is an interpretation that assigns true to this formula. A formula is satisfiable on a domain (here $\mathbb{R}$) if it has a model on that domain.

2.1 Difference Arithmetic with Unary Predicates

We consider several fragments where the language is restricted, in particular in the way that the arithmetic relations can be used. A fragment is decidable if there exists a procedure to check whether a given formula in this fragment is satisfiable.

In the various fragments introduced below, all arithmetic atoms are either *order constraints* of the form $x \bowtie y$, or *difference constraints* of the form $x - y \bowtie c$, where x and y are variables, c is a constant in $\mathbb{Z}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. As a reminder, the language of our formulas only contains *unary predicates*. The only atoms besides the arithmetic ones are of the form $P(x)$ where P is an uninterpreted predicate symbol and x is a variable, and $x \in \mathbb{Z}$ where x is a variable. Note that the addition of constraints of the form $x \bowtie c$, where x is a variable and c is an integer constant, to fragments that already admit difference constraints does not increase their expressive power: constraints $x \bowtie c$ can be replaced by difference constraints $x - v_0 \bowtie c$, where v_0 is a particular variable in $\mathbb{Z}$ intended to be interpreted as zero. Indeed, shifting an interpretation by a fixed integer j — i.e., the new interpretation of any variable x is the old value of x plus j, and the new value of any predicate P for a real number $d + j$ is the old value of P for d — preserves the assigned value of formulas in our fragments. Therefore any model where v_0 is an arbitrary integer can be shifted into a model where v_0 is zero.

As syntactic sugar, conjunctions of order constraints will be merged to improve readability, i.e., we will often write $x < y < z$ rather than $x < y \wedge y < z$. Finally, we use the shorthand $P(x + c)$ instead of $\exists y. \, y - x = c \wedge P(y)$, where x is a free variable and $c \in \mathbb{Z}$.

We now introduce our fragments of interest. Their names are inspired from the SMT-LIB nomenclature, where acronyms stand for the theories that appear in the combinations:

---

[2] In the current context, this choice of notation for mixed integer-real arithmetic is simpler than using a multi-sorted logic.

- UF1: the theory of uninterpreted functions, with the restriction that uninterpreted symbols may only correspond to monadic predicates;
- RO: the theory of order on the reals only;
- IRO: the theory of order on the reals and integers;
- IDL: difference logic on the integers;
- RDL: difference logic on the reals.

UF1·RO. The fragment UF1·RO is the fragment with unary uninterpreted predicates and order constraints between variables interpreted over $\mathbb{R}$. Difference logic constraints and atoms of the form $x \in \mathbb{Z}$ are not allowed.

Example: The formula $\forall x \, \exists y, z \,.\, y < x < z \wedge \forall t \,.\, (y < t < z \wedge P(t)) \Rightarrow t = x$ describes a predicate P that is true only on isolated real numbers.

UF1·IRO. The fragment UF1·IRO is the extension of UF1·RO where atoms of the form $x \in \mathbb{Z}$ are allowed. This fragment can express order relations between real and integer variables.

Example: The formula $\forall x, y. \, (x < y \wedge x \in \mathbb{Z} \wedge y \in \mathbb{Z}) \Rightarrow \exists v. \, x < v < y \wedge P(v)$ describes a predicate P that is true for at least one value located between any two integers.

UF1·IDL·IRO. The fragment UF1·IDL·IRO is an extension of the fragment UF1·IRO (and therefore of UF1·RO). It is also interpreted over $\mathbb{R}$. Order constraints between variables and atoms of the form $x \in \mathbb{Z}$ are allowed. Additionally, difference logic constraints are allowed, but they can only involve *integer-guarded* variables.

In order to enforce this integer-guard restriction on difference logic constraints, UF1·IDL·IRO formulas must be *well-guarded*, i.e., difference logic constraints can only appear in the two following contexts:

- $x \in \mathbb{Z} \wedge y \in \mathbb{Z} \wedge x - y \bowtie c$,
- $(x \in \mathbb{Z} \wedge y \in \mathbb{Z}) \Rightarrow x - y \bowtie c$,

where x and y are variables, $c \in \mathbb{Z}$ is a constant, and $\bowtie \in \{<, \leq, =, \geq, >\}$.

Example: The following formula describes a predicate that is either true on all odd numbers and false on all even numbers, or the opposite, as well as true on all non-integer numbers:
$$\big[\forall x, y. \, \big(x \in \mathbb{Z} \wedge y \in \mathbb{Z} \wedge y - x = 2\big) \Rightarrow \big(P(x) \Leftrightarrow P(y)\big)\big]$$
$$\wedge \big[\exists x, y. \, x \in \mathbb{Z} \wedge y \in \mathbb{Z} \wedge P(x) \wedge \neg P(y)\big] \wedge \big[\forall z. \, \neg(z \in \mathbb{Z}) \Rightarrow P(z)\big]$$

UF1·RDL. The fragment UF1·RDL is the fragment interpreted over $\mathbb{R}$, where order constraints, difference logic constraints and unary predicate atoms are allowed without any restriction. The use of atoms of the form $x \in \mathbb{Z}$ is forbidden. Since order constraints are a special case of difference logic constraints, the name of the fragment only refers to RDL and not RO.

Example: The formula $\forall x \, \exists y. \, 0 < y - x < 3 \wedge P(y)$ describes a predicate P such that any subinterval of $\mathbb{R}$ of length greater or equal to 3 contains a value for which P is true.

Note: It might appear to the reader that a missing logic in this nomenclature is UF1·IRDL, with difference logic constraints on both real and integer variables. We will later show that UF1·RDL is already undecidable, so it makes little sense to introduce any extension of it.

3   Decidability of UF1·IDL·IRO

The fragment UF1·RO is actually a restriction of the universal fragment of the monadic second-order theory of the real order $\mathbb{R}$, i.e., UF1·RO augmented with universal quantification of predicate variables. It has been established in [5] that the universal fragment of the monadic second-order theory of the real order $\mathbb{R}$ is decidable, which trivially implies the decidability of UF1·RO. We show here that its extension UF1·IDL·IRO (and therefore UF1·IRO) is also decidable, by a reduction to UF1·RO.

Theorem 1. *UF1·IDL·IRO and UF1·IRO are decidable.*

Note that the decidability of UF1·IRO is a direct consequence of the decidability of UF1·IDL·IRO, since UF1·IDL·IRO is an extension of UF1·IRO. The remaining of this section is thus dedicated to proving that UF1·IDL·IRO is decidable.

3.1   Recognizing Integer Values

We first show how to define in UF1·RO a predicate P_{int} over $\mathbb{R}$ that is $<$-isomorphic to $\mathbb{Z}$, i.e., such that there exists a bijection between the sets described by P_{int} and $\mathbb{Z}$ that preserves the order relation over their elements. Integer guards in UF1·IDL·IRO will later be translated using P_{int}. Intuitively, an integer-guarded variable in a UF1·IDL·IRO formula will correspond to a variable taking its value in the set described by P_{int} in the translated UF1·RO formula.

We axiomatize P_{int} in UF1·RO as follows:
- Every element of P_{int} is isolated:
  $\forall x \, \exists y, z. \ y < x < z \wedge \forall t. \ [y < t < z \wedge P_{int}(t)] \Rightarrow t = x.$
- Every point in $\mathbb{R}$ has a unique successor in P_{int}:
  $\forall x \, \exists y. \ x < y \wedge P_{int}(y) \wedge \forall t. \ x < t < y \Rightarrow \neg P_{int}(t).$
- Similarly, every point in $\mathbb{R}$ has a unique predecessor in P_{int}:
  $\forall x \, \exists y. \ y < x \wedge P_{int}(y) \wedge \forall t. \ y < t < x \Rightarrow \neg P_{int}(t).$

The set of all integers is a model for P_{int}, therefore the above axiomatization is consistent. The set of elements satisfying P_{int} is necessarily infinite and does not admit a maximal or a minimal element. This is a direct consequence of the successor and predecessor axioms. More interestingly, this set is also necessarily countable. Indeed, since each point is isolated, there exists an application that maps the elements satisfying P_{int} to disjoint open intervals. Any set of disjoint intervals in $\mathbb{R}$ with non-zero length is necessarily countable [18], since each of them contains a rational value that does not belong to the others.
It is now possible to define a successor relation on the real numbers satisfying P_{int} with the formula $Succ(x, y) = P_{int}(x) \wedge P_{int}(y) \wedge y < x \wedge \forall z. \ y < z < x \Rightarrow \neg P_{int}(z),$

i.e., x is the successor of y, or equivalently, y is the predecessor of x.

The axiomatization of P_{int} is, in fact, precise enough to have the following lemma.

Lemma 1. *For any model M of P_{int}, the set $M[P_{int}]$ is $<$-isomorphic to $\mathbb{Z}$.*

For convenience in the proof, we define 0_{int} as an arbitrary existentially quantified value that belongs to the set described by P_{int}.

Proof. Given a model M of the axiomatization of P_{int}, we need to define a bijection between the set $M[P_{int}]$ and $\mathbb{Z}$ that preserves order.

Let us define an application f from $M[P_{int}]$ to $\mathbb{Z}$. We set $f(0_{int}) = 0$, and then define recursively:

- $f(y) = f(x) + 1$ for each $x, y \in M[P_{int}]$ such that $y > 0_{int}$ and $Succ(y, x)$,
- $f(y) = f(x) - 1$ for each $x, y \in M[P_{int}]$ such that $y < 0_{int}$ and $Succ(x, y)$.

Thanks to the fact that every element of $M[P_{int}]$ has a unique predecessor and successor, it follows that f ranges over the whole set $\mathbb{Z}$, proving that f is surjective. Since it is clear that f preserves order, it follows that f is strictly increasing, and therefore injective. It remains to show that f is well defined for every element in $M[P_{int}]$.

If there exists some element $y \in M[P_{int}]$ for which f is not defined, it means that f is not well-defined, in the sense that there exists either an element $y > 0_{int}$ such that the interval $[0_{int}, y]$ contains an infinite number of elements satisfying P_{int}, or there exists an element $y < 0_{int}$ such that the interval $[y, 0_{int}]$ contains an infinite number of elements satisfying P_{int}. Since both cases are symmetric, we only address the former. There must exist a strictly increasing infinite series of elements in $M[P_{int}]$ bounded by y. Let us consider its limit $z \in \mathbb{R}$. Because there must exist an element of $M[P_{int}]$ smaller than z and arbitrarily close to z, it follows that z cannot have a predecessor, which contradicts an axiom. Therefore f is well-defined, and every element of $M[P_{int}]$ is associated to an integer number. The application f is therefore a bijection. □

3.2 Translating Formulas

We are now able to describe the satisfiability-preserving translation of formulas from UF1·IDL·IRO to UF1·RO. Consider a UF1·IDL·IRO formula φ. Without loss of generality, we assume that P_{int} does not appear in φ. The translation of φ is defined as

$$AXIOMS_{int}(P_{int}) \wedge [\![\varphi]\!]$$

where $AXIOMS_{int}(P_{int})$ is the conjunction of the axioms of P_{int}, and $[\![\cdot]\!]$ is a translation operator. This translation operator $[\![\cdot]\!]$ distributes over all Boolean operators and quantifiers, and corresponds to the identity transformation for most considered atoms, except in the following cases:

- $[\![x \in \mathbb{Z}]\!] = P_{int}(x)$;

– $[\![x - y \bowtie c]\!] = \exists z_0, \ldots z_c . \ (y = z_0) \wedge (x \bowtie z_c) \wedge \bigwedge_{0 \le i < c} Succ(z_{i+1}, z_i)$,
for $c \in \mathbb{N}$ and $\bowtie \in \{<, \le, =, \ge, >\}$. We assume that $z_0, \ldots z_c$ are fresh variables w.r.t. x and y.

Example: $[\![x - y \le 2]\!] = \exists z_0, z_1, z_2. \ y = z_0 \wedge Succ(z_1, z_0) \wedge Succ(z_2, z_1) \wedge x \le z_2$. Notice that we only deal with the case $c \in \mathbb{N}$ since every atom of the form $x - y \bowtie c$ with $c \in \mathbb{Z} \setminus \mathbb{N}$ and $\bowtie \in \{<, \le, =, \ge, >\}$ can be rewritten as $y - x \bowtie' -c$ with the following correspondences: $(\bowtie, \bowtie') \in \{(=, =), (<, >), (>, <), (\ge, \le), (\le, \ge)\}$.

3.3 Establishing Equisatisfiability

Given a UF1·IDL·IRO formula φ, the translation that we have introduced generates a corresponding UF1·RO formula ψ. To establish that they are equisatisfiable, we need to prove that if φ admits a model, then ψ also admits one, and reciprocally.

Lemma 2. *Given a* UF1·IDL·IRO *formula* φ, *consider its translation into* UF1·RO $\psi = AXIOMS_{int}(P_{int}) \wedge [\![\varphi]\!]$. *The formulas* φ *and* ψ *are equisatisfiable.*

Proof. If φ is satisfiable, let M be one of its models. Then, since ψ shares the same free variables and predicates than φ with the only addition of P_{int}, we can directly construct a model M' of ψ that is similar to M for the shared variables and predicates, and that interprets P_{int} so that $P_{int}(x)$ holds whenever $x \in \mathbb{Z}$. This is always possible since the only constraints on P_{int} generated by the construction of ψ are the axioms stated above.

If ψ is satisfiable, then there exists a model M of ψ. Let us construct a model M' of φ. Let $0_{int} \in \mathbb{R}$ be an arbitrary element of $M[P_{int}]$. We define an automorphism g of $\mathbb{R}$, such that $g(0_{int}) = 0$, and recursively $g(y) = g(x) + 1$ for $x, y \in M[P_{int}]$, $y > 0_{int}$ and $Succ(y, x)$, and $g(y) = g(x) - 1$ for $x, y \in M[P_{int}]$, $y < 0_{int}$ and $Succ(x, y)$. The automorphism g maps each open interval between the k-th and $(k+1)$-th successors (resp. predecessors) of 0_{int} in $M[P_{int}]$, onto the open interval $(k, k+1)$ (resp. $(-(k+1), -k)$) while preserving order.

M' is defined by $M'[x] = g(M[x])$ for each free variable x of the formula φ, and $M'[P] = \{g(x) \mid x \in M[P]\}$ for each uninterpreted predicate P of φ. No unary predicate atom can be violated by M' by definition. Furthermore, no order constraint can be violated by M' either since g preserves order. Regarding the difference logic constraints, the intermediate variables z_i introduced in the translation are necessarily mapped to values in $M[P_{int}]$ since the *Succ* relation enforces this property. Hence for each such variable, we have $g(M[z_i]) \in \mathbb{Z}$. Intuitively, this ensures that in M' the difference between the values taken by the integer variables is consistent with the difference logic constraints. It follows that M' is a model of φ. □

4 Undecidability of UF1·RDL

The result presented in the previous section establishes a lower bound for the decidability of our family of fragments. A natural follow-up problem is to establish

a corresponding upper bound, i.e., to find an extension of this logic that yields undecidability. We show here that, when combined with uninterpreted unary predicates, as soon as difference logic constraints on reals are allowed, the logic becomes undecidable.

We actually show a stronger result which is that a single unary predicate symbol is enough to yield undecidability. More precisely, we establish the undecidability of the restriction of UF1·RDL where only one predicate symbol is allowed, by reducing the halting problem of a Turing machine to the satisfiability problem over this restriction of UF1·RDL.

Theorem 2. *Satisfiability is undecidable for* UF1·RDL *with a single predicate.*

Corollary 1. *Satisfiability is undecidable for* UF1·RDL.

The remaining of this section is dedicated to proving Theorem 2. We consider w.l.o.g. Turing machines defined over an alphabet with only two symbols and no explicit blank symbol [16]. This choice leads to a simpler proof.

4.1 Definitions

The proof is by reduction from the halting problem for a Turing machine with a single bi-infinite tape, starting from a blank tape (i.e., a tape filled with the symbol 0). Consider a Turing machine $\mathcal{M} = (Q, \Sigma, q_I, q_F, \Delta)$, where

- Q is a finite nonempty set of states,
- Σ is the alphabet $\{0, 1\}$,
- $q_I \in Q$ is the initial state,
- $q_F \in Q$ is the halting state,
- $\Delta \subseteq \{(Q \setminus \{q_F\}) \times \Sigma \times Q \times \Sigma \times \{L, R\}\}$ is the transition relation, assumed to be total over its first two components, i.e., for any pair $(q, \alpha) \in (Q \setminus \{q_F\}) \times \Sigma$, there exists a tuple $(q, \alpha, q', \alpha', \lambda) \in \Delta$.

A *configuration* C of such a Turing machine is a triplet containing the current state q, the content of the tape $t \in \{0, 1\}^{\mathbb{Z}}$ and the position of the head $h \in \mathbb{Z}$. Since the machine starts from a blank tape, the initial configuration is $C_0 = (q_I, 0^{\mathbb{Z}}, 0)$.

A *run* ρ of length $n \in \mathbb{N}$ (resp. $n = +\infty$) of such a Turing machine is a finite (resp. infinite) sequence of configurations $(C_i)_{i \in [0, n]}$ (resp. $(C_i)_{i \in \mathbb{N}}$), such that for any two consecutive configurations $C_i = (q_i, t_i, h_i)$ and $C_{i+1} = (q_{i+1}, t_{i+1}, h_{i+1})$ there exists a transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$ such that:

- $q = q_i$ and $q' = q_{i+1}$,
- $t_i[h_i] = \alpha$, i.e., the tape cell at position h_i contains the symbol α,
- $t_{i+1}[h_i] = \alpha'$,
- $t_{i+1}[k] = t_i[k]$, for every $k \in \mathbb{Z}$, $k \neq h_i$,
- $h_{i+1} = h_i + 1$ if $\lambda = R$, and $h_{i+1} = h_i - 1$ if $\lambda = L$.

A *halting run* is a finite run such that the state of its last configuration is the halting state q_F.

4.2   Encoding Runs

Our goal is to encode a run of a Turing machine (as described before), i.e., encode the state, the tape content, and the position of the head for each configuration of such a run. Starting from the initial configuration, we must also ensure the coherence of the run w.r.t. the Turing machine transition relation, by connecting every two consecutive configurations. Our idea is to define an infinite sequence of intervals on the real line, such that each interval contains the encoding of its corresponding configuration (i.e., the first interval will contain the first configuration of the run, and so on). Difference constraints can then be used to connect consecutive configurations.

Let $N = \lceil \log_2(|Q|) \rceil$. Each state $q \in Q$ of $\mathcal{M}$ can therefore be uniquely encoded with N Boolean values $b_1^q, \ldots b_N^q$. We want to encode consecutive configurations of the Turing machine using a single predicate P over $\mathbb{R}$. In order to do so, we first need to describe a subset of $\mathbb{R}$ that will act as a grid supporting the encoding of the state, the tape content, and the head position of the current configuration.

We use the concept of linear ordering [15] to describe the shape of the grid. A *linear ordering* J is a totally ordered set, i.e., a set equipped with a binary relation $<$ which is irreflexive (for all j in J, $j \not< j$), asymmetric (for all j, k in J, if $j < k$, then $k \not< j$), transitive (for all i, j, k in J, if $i < j$ and $j < k$, then $i < k$), and complete (for all $j, k \in J$, either $j = k$, $j < k$, or $k < j$). The *order type* of a linear ordering J is the class of all linear orderings $<$-isomorphic to J. The order types of a singleton, the set composed of the N first natural numbers, $\mathbb{N}$, and $\mathbb{Z}$ are respectively denoted by 1, N, ω, and ζ. The concatenation of two linear orderings J and K (where their associated order relations are respectively $<_J$ and $<_K$) is denoted by $J + K$. It corresponds to the linear ordering composed of the set of pairs $\{(j, 1) \mid j \in J\} \cup \{(k, 2) \mid k \in K\}$, and equipped with the order relation $<$, defined by $(j_1, 1) < (j_2, 1)$ if $j_1 <_J j_2$, $(k_1, 2) < (k_2, 2)$ if $k_1 <_K k_2$, and $(j, 1) < (k, 2)$ for every $j \in J$ and $k \in K$. More generally, given two linear orderings J and K, the linear ordering $(J)^K$ is the set of pairs (j, k) with $j \in J$ and $k \in K$, with the order relation $<$ such that $(j_1, k_1) < (j_2, k_2)$ if either $k_1 <_K k_2$, or $k_1 =_K k_2$ and $j_1 <_J j_2$. These operators are naturally extended on order types. For instance, the order type $(\omega)^\omega$ is the class of all linear orderings $<$-isomorphic to $\mathbb{N}^2$.

The grid we consider is a linear ordering that is a subset of $\mathbb{R}$, of order type $(N + \zeta + 1 + \zeta)^\omega$. An ordering of order type $N + \zeta + 1 + \zeta$ within the interval $[0, 3)$ is depicted in Figure 1. Each dot corresponds to a natural number and each vertical line corresponds to an element of the linear ordering. The first N points will support the encoding of a state. The first subordering that is $<$-isomorphic to $\mathbb{Z}$ (i.e., of order type ζ) will be used to encode the position of the head, while the second one will support the encoding of the tape content. The whole grid is composed of an infinite repetition of the subordering $N + \zeta + 1 + \zeta$ (i.e., it is repeated on the intervals $[3k, 3k + 3)$ for all $k \in \mathbb{N}$), hence the ω exponent.
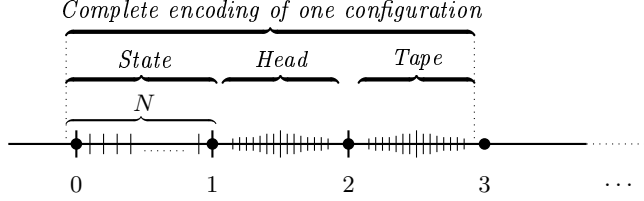
Complete encoding of one configuration

Fig. 1. A visual representation of a linear ordering of order type $N + \zeta + 1 + \zeta$.

4.3 Defining the Support of the Encoding

Let us first define concretely the support of the encoding of the Turing machine configurations. The difficulty lies in describing the grid using a single predicate P, without meddling with the actual encoding of the configurations afterwards. Our solution is to characterize the points that belong to the grid by enforcing that such a point is surrounded by an open interval where P is uniformly *true* on the left, and by an open interval where P is uniformly *false* on the right, such as depicted in Figure 2. We do not specify yet how P behaves on x, as this is how the configurations will actually be encoded later.
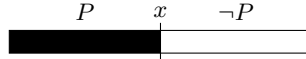

Fig. 2. The real number x belongs to the grid, since it is surrounded by a *true* (black) open interval on the left, and a *false* (white) open interval on the right.

Such a characterization is easy to express in our restriction of UF1·RDL:

$$Support(x) = (\exists y.\, y < x \wedge \forall z.\, y < z < x \Rightarrow P(z)) \wedge (\exists y.\, x < y \wedge \forall z.\, x < z < y \Rightarrow \neg P(z))$$

Let us now partially axiomatize the predicate P such that the set of *supporting* points constitutes a linear ordering of order type $\left(N + \zeta + 1 + \zeta\right)^{\omega}$:

(a) Let **0** be a variable and **1, 2** and **3** be respectively the +1-successor of **0, 1** and **2**:
$Axiom_1 = (\mathbf{1} = \mathbf{0} + 1) \wedge (\mathbf{2} = \mathbf{1} + 1) \wedge (\mathbf{3} = \mathbf{2} + 1)$
These free variables are implicitly existentially quantified in the final formula. Notice that the variable **0** can be interpreted as any real value, which only acts as a landmark for the beginning of the grid.

(b) **0**, **1** and **2** are *supporting* points:
$Axiom_2 = Support(\mathbf{0}) \wedge Support(\mathbf{1}) \wedge Support(\mathbf{2})$

(c) P is uniformly true before **0**, i.e., there are no supporting points before **0**:
$Axiom_3 = \forall x.\, x < \mathbf{0} \Rightarrow P(x)$

(d) There are exactly $N - 2$ *supporting* points within the interval $(\mathbf{0}, \mathbf{1})$:
$Axiom_4 = \exists x_1, x_2, \ldots x_N.\, x_1 = \mathbf{0} \wedge x_N = \mathbf{1}$
$\quad\quad \wedge \bigwedge_{1 \leq i < N} \left(\mathbf{0} \leq x_i < \mathbf{1} \wedge Succ_{Supp}(x_{i+1}, x_i) \right)$

10

where $Succ_{Supp}(x,y)$ is a formula that states that x is the first *supporting* real value that is strictly greater than y, i.e., x is the successor of y on the grid. It is defined as follows:

$Succ_{Supp}(x,y) = y < x \land Support(x) \land Support(y) \land \forall z.\, y < z < x \Rightarrow \neg Support(z)$

We also define an analogous formula to express that x is the predecessor of y: $Pred_{Supp}(x,y) = Succ_{Supp}(y,x)$.

(e) The set of *supporting* points within $(\mathbf{1}, \mathbf{2})$ is $<$-isomorphic to $\mathbb{Z}$. This is done similarly to the axiomatization of P_{int} (cf. Section 3.1). But because $\mathbf{1}$ (resp. $\mathbf{2}$) is a *supporting* point, there must exist a uniformly false (resp. true) interval of P at its right (resp. left) where no other *supporting* points can appear. All the *supporting* points will therefore be constrained to appear within a smaller interval (b_1, b_2) with $\mathbf{1} < b_1 < b_2 < \mathbf{2}$, as illustrated in Figure 3.

$$Axiom_5 = [\exists b_1, b_2.\, \mathbf{1} < b_1 < b_2 < \mathbf{2}] \tag{1}$$
$$\land\, [\forall x.\, (b_1 < x < b_2) \Rightarrow \exists y.\, x < y < b_2 \land Support(y)$$
$$\land\, \forall z.\, x < z < y \Rightarrow \neg Support(z)] \tag{2}$$
$$\land\, [\forall x.\, (b_1 < x < b_2) \Rightarrow \exists y.\, b_1 < y < x \land Support(y)$$
$$\land\, \forall z.\, y < z < x \Rightarrow \neg Support(z)] \tag{3}$$
$$\land\, [\forall x.\, (\mathbf{1} < x < \mathbf{2} \land Support(x)) \Rightarrow b_1 < x < b_2] \tag{4}$$

This axiom can be broken down into these elementary pieces:
(1) there exists an open interval (b_1, b_2) such that $\mathbf{1} < b_1 < b_2 < \mathbf{2}$,
(2) each real value in (b_1, b_2) has a *supporting* successor,
(3) each real value in (b_1, b_2) has a *supporting* predecessor,
(4) there are no *supporting* points within $(\mathbf{1}, b_1)$, nor within $(b_2, \mathbf{2})$.

(f) The pattern of *supporting* points within $(\mathbf{1}, \mathbf{2})$ is repeated onto the interval $(\mathbf{2}, \mathbf{3})$ with an exact offset of 1:

$Axiom_6 = \forall x.\, \mathbf{1} < x < \mathbf{2} \Rightarrow (Support(x) \Leftrightarrow Support(x+1))$

(g) The pattern of *supporting* points within $[\mathbf{0}, \mathbf{3})$ is repeated onto every interval $[\mathbf{3k}, \mathbf{3k+3})$ for $k \in \mathbb{N}$:

$Axiom_7 = \forall x.\, x \geq \mathbf{0} \Rightarrow (Support(x) \Leftrightarrow Support(x+3))$

Notice that for $Axiom_7$, it is not enough that a similar pattern appears within each interval $[\mathbf{3k}, \mathbf{3k+3})$: there must be an exact offset of 3 with the previous interval. This is mandatory to connect two consecutive configurations and ensure that they are coherent with the transition relation of the Turing machine, as defined later. The same goes for $Axiom_6$, where the exact offset of 1 will allow to connect the position of the head to the tape content within a single configuration.

Fig. 3. The points of the grid surrounded by open *true* (black) and *false* (white) intervals within $(\mathbf{1}, \mathbf{2})$.

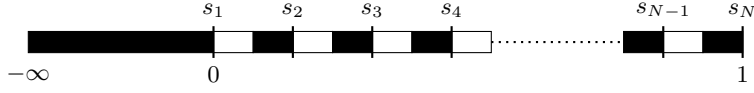The formula $AXIOMS_{Supp} = \bigwedge_{1 \leq k \leq 7} Axiom_k$ axiomatizes the predicate P.

11

Fig. 4. A model for the axiomatization of P over the interval $(-\infty, 1)$.

Lemma 3. *The formula AXIOMS$_{Supp}$ is consistent.*

The proof sketch below provides the key ideas to construct a model of $AXIOMS_{Supp}$. The complete construction is described in [2].

Proof. Let us construct a subset S of $\mathbb{R}$ that is a model of $AXIOMS_{Supp}$. Firstly, we make every negative number belong to S, which ensures that there do not exist negative supporting points. The interval $[0, 1]$ is then cut into $2N - 2$ intervals of equal length, which alternate between being included in S, and being disjoint from S. This ensures the existence of exactly $N - 1$ supporting points within the interval $(-\infty, 1)$, 0 being the first; 1 will be considered later. These $N - 1$ supporting points are referred to as $s_1, s_2, \ldots s_{N-1}$ and are depicted in Figure 4. Recall that the supporting points are exactly those surrounded by an interval of S (i.e., black on the figure) on the left, and an interval disjoint from S (i.e., white) on the right.

In order to make the real value 1 the N-th supporting point, it is enough to make an interval on its right disjoint from S, e.g., the interval $(1, 1 + \frac{1}{4})$. Symmetrically, we make the interval $(2 - \frac{1}{4}, 2)$ included in S, satisfying the left part of the requirement for the real value 2 to be a supporting point.

We further characterize S such that the set of supporting points within the interval $(1 + \frac{1}{4}, 2 - \frac{1}{4})$ is $<$-isomorphic to $\mathbb{Z}$. This can be done by partitioning the open interval $(1 + \frac{1}{4}, 2 - \frac{1}{4})$ into a bi-infinite sequence of open intervals alternating between being included and disjoint from S, as depicted in Figure 5.
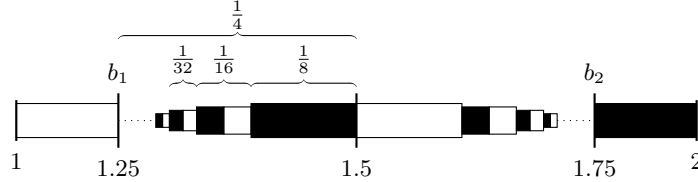

Fig. 5. A model for the axiomatization of P over the interval $(1, 2)$.

The whole pattern described on the interval $(1, 2)$ can be directly transposed onto the interval $(2, 3)$ with an exact offset of $+1$. Similarly, the distribution of S over the interval $(0, 3)$ can be transposed onto every interval $(3k, 3k + 3)$ with an offset of $+3k$, for $k > 0$. The only real values for which we do not describe their relation with S are the points surrounded by an interval included in S on one side, and an interval disjoint from S on the other side. These points never conflict with the axiomatization $AXIOMS_{Supp}$ which only deals with non-empty open intervals.

By construction, S satisfies each axiom of the formula $AXIOMS_{Supp}$, and is therefore a model of this formula. □

4.4 Encoding a Configuration of the Turing Machine

Now that the supporting grid has been properly defined, the actual encoding of a given configuration can be addressed. That is, the state, the tape content and the head position of the $(k+1)$-th configuration of a run are encoded on the supporting points contained within the interval $[3k, 3k+3)$.

Encoding the State Encoding the state of a given configuration is rather direct since we defined the grid to contain N consecutive supporting points within every interval $[3k, 3k+1]$ for $k \in \mathbb{N}$, that can support the encoding of a state. We only need to indicate that we start reading the encoding on a multiple of 3. However the logic UF1·RDL does not allow to express periodicity constraints on variables. Nevertheless, thanks to our axiomatization, 0 and every other positive multiple of 3 are the only points that simultaneously have no supporting predecessor, while admitting a supporting successor. These properties are expressible as follows:

$NoPred_{Supp}(x) = \forall z.\,(z < x \wedge Support(z)) \Rightarrow \exists y.\, z < y < x \wedge Support(y)$

$HasSucc_{Supp}(x) = \exists z.\, x < z \wedge Support(z) \wedge \forall y.\, x < y < z \Rightarrow \neg Support(y)$

For convenience, we introduce the formula $EncodingBegins$ to characterize a real value x on which the encoding of a state starts:

$EncodingBegins(x) = Support(x) \wedge NoPred_{Supp}(x) \wedge HasSucc_{Supp}(x)$

Furthermore, the formula $State_q$ expresses that a state $q \in Q$ is encoded on a given real number x and its $N-1$ supporting successors:

$$State_q(x) = EncodingBegins(x) \wedge \exists y_1, \ldots y_N.\, x = y_1$$
$$\wedge \bigwedge_{1 \leq i < N} Succ_{Sup}(y_{i+1}, y_i) \wedge \bigwedge_{1 \leq i \leq N} P(y_i) = b_i^q$$

where $P(y_i) = b_i^q$ is a shorthand for $P(y_i)$ if $b_i^q = \top$, and $\neg P(y_i)$ if $b_i^q = \bot$.

Encoding the Head Position The position of the head is encoded in the second part of the grid, that is, in the interval $(3k+1, 3k+2)$ for the $(k+1)$-th configuration (cf. Figure 1). The grid on this interval is $<$-isomorphic to $\mathbb{Z}$. Each element of this subordering will correspond to a position of the tape. When the predicate P is true at such a point, it means that the head points towards that cell. Since the Turing machines that we consider here have a single read/write head, it must point towards a unique cell for each configuration. Therefore P must be true only for a single element of that subordering.

Encoding the Tape Content Similarly, the tape content is encoded in the third part of the grid, that is, in the interval $(3k+2, 3k+3)$ for the $(k+1)$-th configuration (cf. Figure 1). Again, the grid on this interval is $<$-isomorphic to

$\mathbb{Z}$. And again, each element x of this subordering will correspond to a cell of the tape, matching the cell that corresponds to $x - 1$ in the head position interval. Figure 6 illustrates the connections between the suborderings, within a single configuration and with the next one. The idea of the encoding is to simply set the value of P to true on the elements of the subordering that correspond to cells containing a 1, and to false for cells containing a 0.
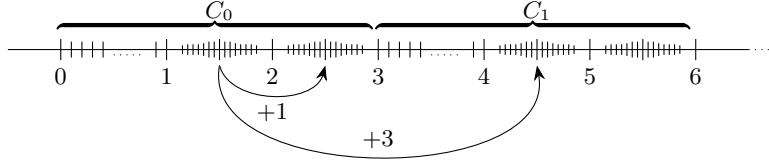

Fig. 6. The first two consecutive configuration encodings.

4.5 Enforcing a Valid Run

Let us now define formally the formulas characterizing an accepting run of $\mathcal{M}$. We will decompose the global formula into three main parts: the initial conditions $START_{\mathcal{M}}$, the conditions on the transitions $STEP_{\mathcal{M}}$ and the halting condition $END_{\mathcal{M}}$. For the sake of clarity, we use capital letters for these higher-level formulas.

The initial conditions of $\mathcal{M}$ are that the state encoded on $\mathbf{0}$ and its $N - 1$ supporting successors is the initial state q_0, that the head points towards a unique initial unspecified cell of the tape, and finally that the tape is initially filled with 0's. These conditions are expressed by the following formula:

$$START_{\mathcal{M}} = State_{q_0}(\mathbf{0}) \wedge \big[\exists y.\, \mathbf{1} < y < \mathbf{2} \wedge Support(y) \wedge P(y)$$
$$\wedge \forall x.\, (\mathbf{1} < x < \mathbf{2} \wedge Support(x) \wedge P(x)) \Rightarrow x = y\big]$$
$$\wedge \big[\forall y.\, (\mathbf{2} < y < \mathbf{3} \wedge Support(y)) \Rightarrow \neg P(y)\big]$$

The requirements on the transition are more complex. Intuitively, if before reaching the step $i \in \mathbb{N}$, we have not yet encountered the halting state q_F, then we must ensure that the configuration at Step i can be obtained from the configuration at the previous step $i - 1$ by following a transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$. The overall formula for this condition is the following:

$$STEP_{\mathcal{M}} = \forall y.\, (y > 0 \wedge EncodingBegins(y) \wedge NotEnded_{\mathcal{M}}(y)$$
$$\Rightarrow \exists x.\, y = x + 3 \wedge Transition_{\mathcal{M}}(x, y)$$

The subformula $NotEnded_{\mathcal{M}}(y)$ expresses that no valid real value prior to y (i.e., a positive multiple of 3 strictly smaller than y) encodes the halting state. This formula is defined by:

$$NotEnded_{\mathcal{M}}(y) = \forall x.\, (x < y \wedge EncodingBegins(x)) \Rightarrow \neg(State_{q_F}(x))$$

14

The subformula $Transition_{\mathcal{M}}(x, y)$ expresses that there exists a transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$ that allows to move in one step from the configuration encoded at x (i.e., that the encoding of the configuration starts exactly on x), to the configuration corresponding to y. To improve readability, we decompose the condition on the transition relation as follows:

$$Transition_{\mathcal{M}}(x, y) =$$
$$\bigvee_{(q,\alpha,q',\alpha',\lambda)\in\Delta} \left[State_q(x) \wedge State_{q'}(y) \wedge Tape_{\alpha,\alpha'}(x, y) \wedge Head_\lambda(x, y) \right]$$

For a given transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$, the conditions on the states, tape and head are expressed as follows:

- The state q must be encoded on the real value x, and the state q' on y: $State_q(x) \wedge State_{q'}(y)$
- The tape must contain $\alpha \in \{0, 1\}$ at the position of the head for the step corresponding to x. Additionally, for the step corresponding to y, the tape must contain α' at the previous position of the head, and remain unchanged at all other positions.

$$Tape_{\alpha,\alpha'}(x, y) = \big[\forall z. (x + 1 < z < x + 2 \wedge Support(z) \wedge P(z))$$
$$\Rightarrow P(z + 1) = \alpha \wedge P(z + 4) = \alpha' \big]$$
$$\wedge \big[\forall z. (x + 1 < z < x + 2 \wedge Support(z) \wedge \neg P(z)) \Rightarrow (P(z + 1) \Leftrightarrow P(z + 4)) \big]$$

Where $P(z + k) = \alpha$ is a shorthand for $\exists u. u = z + k \wedge P(u)$ if $\alpha = 1$, and $\exists u. u = z + k \wedge \neg P(u)$ if $\alpha = 0$. The "$+1$" operator allows us to connect the encoding of the head position with the encoding of the tape content within the same configuration. The "$+4$" operator does the same while jumping to the next configuration (cf. Figure 4). Notice that this formula does not involve y; it assumes (rightfully, given the formula $STEP_{\mathcal{M}}$) that the equality $y = x + 3$ holds.
- The head is moved in the direction specified by $\lambda \in \{L, R\}$, i.e., left for L and right for R. This can be expressed by exploiting the predecessor and successor relations defined for supporting real values.

$$Head_\lambda(x, y) = \forall z. (x + 1 < z < x + 2 \wedge Support(z) \wedge P(z))$$
$$\Rightarrow \exists v. f_\lambda(v, z + 3) \wedge P(v) \wedge \neg P(z)$$

where $f_R = Succ_{Supp}$ and $f_L = Pred_{Supp}$. Since in the initial configuration of the Turing machine the head points towards a single cell, the formula $Head_\lambda$ ensures that this remains the case throughout every run of the Turing machine.

Finally, the existence of a halting run is expressed by the formula:

$$END_{\mathcal{M}} = \exists x. State_{q_F}(x)$$

15

The global formula that expresses that the Turing machine $\mathcal{M}$ halts on some run encoded by the value of the predicate P is the following:

$$HALT_{\mathcal{M}} = START_{\mathcal{M}} \wedge STEP_{\mathcal{M}} \wedge END_{\mathcal{M}} \wedge AXIOMS_{Supp}$$

where $AXIOMS_{Supp}$ is the axiomatization of the supporting points as described in Section 4.3.

By construction, satisfiability of the global formula $HALT_{\mathcal{M}}$ is equivalent to the existence of a halting run for the Turing machine $\mathcal{M}$. It follows that the satisfiability problem for UF1·RDL is undecidable, which proves Theorem 2.

5   Conclusion

This work provides a lower and an upper bound for the decidability of first-order fragments with quantifiers mixing uninterpreted unary predicates and weak forms of real arithmetic. This draws a precise picture of the frontier of decidability in fragments mixing real arithmetic and uninterpreted predicates.

We proved the decidability of the fragment UF1·IDL·IRO, where uninterpreted unary predicates, order constraints between real and integer variables, and difference logic constraints between integer variables are allowed. This result is a consequence of the already established decidability of its restriction UF1·RO, where only uninterpreted unary predicates and order constraints between real values are allowed. To the best of our knowledge, there does not exist yet a practical decision procedure for UF1·RO.

There exist fragments of arithmetic that are more expressive than difference logic, but still weaker than full Presburger arithmetic. It would be interesting to investigate if decidability for these is preserved in presence of uninterpreted unary predicates. Note however that our proof of decidability strongly relies on the translation of the constraints into the first-order theory of order over $\mathbb{R}$, with unary predicates. This translation is not suitable for, e.g., constraints of the form $x + y \bowtie 0$, where x and y are variables, and $\bowtie \in \{<, \leq, =, \geq, >\}$.

In another result, we established the undecidability of the fragment UF1·RDL, where uninterpreted unary predicates and difference logic constraints between real variables are allowed. It is worth mentioning that this result can be adapted straightforwardly to the same logic interpreted over the domain $\mathbb{Q}$.

Our long term goal is to design an effective decision procedure for the decidable fragment. Complexity results have been established [13, 6, 14] for the temporal logic counterpart of the theory of order, to which we reduce the decidability of our fragment of interest. We are currently designing a decision procedure relying on the concept of automata on linear orderings introduced in [3]. We hope that the insight we obtained through this decision procedure will eventually guide the design of new powerful instantiation techniques for SMT in a more expressive context, and that these techniques will happen to be complete in particular for this decidable fragment.

References

1. Boigelot, B., Fontaine, P., Vergain, B.: Decidability of difference logics with unary predicates. Proceedings, 7th International Workshop on Satisfiability Checking and Symbolic Computation (2022)
2. Boigelot, B., Fontaine, P., Vergain, B.: Decidability of difference logic over the reals with uninterpreted unary predicates. arXiv preprint (2023)
3. Bruyère, V., Carton, O.: Automata on linear orderings. Journal of Computer and System Sciences **73**(1), 1–24 (2007)
4. Büchi, J.R.: On a decision method in restricted second order arithmetic, Logic, Methodology and Philosophy of Science (1962)
5. Burgess, J.P., Gurevich, Y.: The decision problem for linear temporal logic. Notre Dame Journal of Formal Logic **26**(2), 115–128 (1985)
6. Cristau, J.: Automata and temporal logic over arbitrary linear time. In: FSTTCS 2009. LIPIcs, vol. 4, pp. 133–144 (2009)
7. Downey, P.J.: Undecidability of Presburger arithmetic with a single monadic predicate letter. Tech. rep., Center for Research in Computer Technology, Harvard University (1972)
8. Enderton, H.B.: A mathematical introduction to logic (2nd edition). Academic Press, Boston (2001)
9. Gurevich, Y., Shelah, S.: Monadic theory of order and topology in ZFC. Annals of Mathematical Logic **23**(2-3), 179–198 (1982)
10. Haase, C.: A survival guide to Presburger arithmetic. ACM SIGLOG News **5**(3), 67–82 (2018)
11. Halpern, J.Y.: Presburger arithmetic with unary predicates is Π_1^1 complete. The Journal of Symbolic Logic **56**(2), 637–642 (Jun 1991)
12. Matiyasevich, Y.V.: Hilbert's Tenth Problem. MIT Press, Cambridge, MA, USA (1993)
13. Rabinovich, A.: Temporal logics over linear time domains are in PSPACE. Information and Computation **210**, 40–67 (2012)
14. Reynolds, M.: The complexity of temporal logic over the reals. Annals of Pure and Applied Logic **161**(8), 1063–1096 (2010)
15. Rosenstein, J.G.: Linear orderings. Academic press (1982)
16. Shannon, C.E.: A universal Turing machine with two internal states. Automata studies **34**, 157–165 (1956)
17. Shelah, S.: The monadic theory of order. Annals of Mathematics **102**(3), 379–419 (1975)
18. Sierpiński, W.: Cardinal and ordinal numbers (2nd edition). PWN, Warszawa (1965)
19. Speranski, S.O.: A note on definability in fragments of arithmetic with free unary predicates. Arch. Math. Log. **52**(5-6), 507–516 (2013)
20. Tarski, A.: A Decision Method for Elementary Algebra and Geometry. Univ. of California Press, 2nd edn. (1951)