preprint. To appear in *Computability* (IOS Press)

Mortality of Iterated Piecewise Affine Functions over the Integers: Decidability and Complexity

Amir M. Ben-Amram

November 26, 2014

Abstract

In the theory of discrete-time dynamical systems one studies the limiting behaviour of processes defined by iterating a fixed function f over a given space. A much-studied case involves piecewise affine functions on $\mathbb{R}^n$. Blondel et al. (2001) studied the decidability of questions such as *global convergence* and *mortality* for such functions with rational coefficients. *Mortality* means that every trajectory (namely a sequence $x_0, f(x_0), f(f(x_0)), \dots$) includes a 0; if the iteration is implemented as a loop `while` $(x \neq 0)$ x `:=` $f(x)$, mortality means that the loop is guaranteed to terminate. Checking the termination of simple loops (under various restrictions of the guard and the update function) is a much-studied topic in automated program analysis.

Blondel et al. proved that the problems are undecidable when the state space is $\mathbb{R}^n$ (or $\mathbb{Q}^n$), and the dimension n is at least two. From a program analysis (and discrete computability) viewpoint, it is more natural to consider functions over the integers.

This paper establishes (un)decidability results for the *integer* setting, while strengthening results for the continuous setting. We show that also over integers, undecidability begins at two dimensions. In both settings, we shown Π^0_2 completeness. We further investigate the effect of several restrictions on the iterated functions. Specifically, we consider bounding the size of the partition defining f, and restricting the coefficients of the linear components. In the decidable cases, we give complexity results: The problem is PSPACE-complete for piecewise-affine functions in one dimension; it is polynomial-time for affine functions in any dimension. The undecidability proofs use some variants of the Collatz problem, which may be of independent interest.

1 Introduction

In the context of this paper, an (n-dimensional) *discrete-time dynamical system* on a set X is defined by $x_{t+1} = f(x_t)$, where $f : X \to X$. For a given initial point x_0, the sequence so generated (sometimes denoted by $f^{(t)}(x_0)$) is called *a trajectory*, and some of the central problems regarding such systems involve the asymptotic properties shared by all trajectories of the system. Three such properties, on which we shall focus, are called *global convergence*, *mortality* and *global convergence to a fixed point.* A class of functions much studied in the Dynamical System literature is *piecewise affine functions* (its precise definition appears in Section 1.3). The purpose of this paper is to study the complexity of deciding the above-named asymptotic properties for piecewise affine functions over the integers. The next subsection states what these problems are exactly, what has been known about them (forming the departure point of this study) and what the new contributions are.

1.1 The problems studied, known and new results

The asymptotic properties studied in this work are *global convergence*, *mortality* and *global convergence to a fixed point.*

Definition 1.1. Let f be an arbitrary map on a metric space X with distinguished origin 0.
f is *globally convergent to zero* if for every initial point $x_0 \in X$, the trajectory $x_{t+1} = f(x_t)$ converges to 0 (the words "to zero" and sometimes "globally" may be omitted in the sequel, as long as no confusion can arise).
f is *mortal* if for every initial point $x_0 \in X$, the trajectory $x_{t+1} = f(x_t)$ reaches the origin (i.e., $\exists t \geq 0 : x_t = 0$).
f is *globally convergent to a fixed point* if for every initial point $x_0 \in X$, the trajectory $x_{t+1} = f(x_t)$ reaches a fixed point of f (i.e., $\exists t \geq 0 : f(x_t) = x_t$).

Decidability of the first two problems was studied by Blondel et al. [9]. They considered piecewise affine functions f, where the coefficients are all rational (this is important since we are considering computability in the traditional, discrete sense). The domain of the functions is defined as $\mathbb{R}^n$, though in this particular case, results would not be different if it were restricted to $\mathbb{Q}^n$ (there are problems where such a restriction is significant, even when studying functions with rational coefficients; e.g., [14]).

THEOREM 1.2. [9] *The following problems are undecidable for all $n \geq 2$: Given a piecewise affine function $f : \mathbb{R}^n \to \mathbb{R}^n$ (with rational coefficients), is it globally convergent? Is it mortal?*

Global convergence and mortality[1] *are decidable for $n = 1$ when the function is continuous.*

The domain of the functions can be restricted to $[0, 1]^n$.

Global convergence to a fixed point is discussed by Koiran et al. [38], who note that the simulation of Turing machines by two-dimensional dynamical systems leads to an undecidability result. They do not provide a result on one-dimensional systems.

Among the many decision problems studied for dynamical systems, mortality and convergence to a fixed point are perhaps the most appealing in their discrete setting, since they are the halting problems for very simple types of programs (more on the connection to program termination problems below). The global convergence problem

[1]The statement in [9] regarding the one-dimensional case omits mortality. However, it is easy to see that a decision procedure for mortality follows, see Section 3.1.

	$\mathbb{Q}^2$	$\mathbb{Z}^2$	$\mathbb{N}^2$
PAF without further restrictions	Π^2_0-complete (for continous functions, open)	Π^2_0-complete	Π^2_0-complete
PAF with a restriction on the number of regions	as for $\mathbb{Z}^2$	Π^2_0-complete for N regions, where $N \simeq 7000$	as for $\mathbb{Z}^2$
Coefficient-restricted PAF	Π^2_0-complete for nearly-monic functions	Π^2_0-complete for monic functions	decidability for monic functions is open

Table 1: Summary of main results on mortality of 2-dimensional piecewise-affine functions (abbreviated PAF). For integers, the complexity of global convergence (to 0 or to any fixed point) is the same as that of mortality, while for the rationals these problems are at least as hard. Monic functions are just one kind of coefficient-restricted PAF, see Section 2.4 for further discussion.

as defined above is very closely related (in a discrete setting, a sequence x_t converges to 0 if and only if it is eventually constantly zero).

The main contribution of this paper is to establish for the integer setting results similar to Theorem 1.2. The proofs in [9] do not apply to this setting, since they are based on encoding the state of a computation (say, a Turing-machine tape) in the fractional digits of a number; unlimited precision is essential. To handle a discrete (or limited precision) setting, different proof techniques are necessary. As in the continuous case, we obtain decidability for one dimension and undecidability for two or more. The new undecidability result is stronger than the one in [9] in that it is achieved for a class of functions where there is a fixed bound on the number of regions in the partition defining f ([9] left this question open). Furthermore, we prove Π^0_2 completeness, which is a sharper result than mere undecidability. Since our results are also applicable to the rationals, we obtain an improvement of the characterization in [9].

Next, we consider some other restrictions on the space of functions. Restriction of the number of regions is discussed in Sections 2.2 and 2.3. Section 2.4 shows undecidability for monic functions—these are two-dimensional functions of a very simple form, $f(\vec{x}) = f(x_1, x_2) = (x_{i_1} + b_1, x_{i_2} + b_2)$. The precise definition of the class is quite important for the result. Some variants will be discussed in Section 2.4.5 (but mostly leading to open problems). Table 1 summarises the results on 2-dimensional piecewise-affine functions and indicates some open problems; for details, additional results and open problems see the corresponding sections.

Sections 3–4 present decidability results: first we show that the special case where there is only one region (that is, f is affine) is decidable (in PTIME) in any dimension. Secondly, that the one-dimensional piecewise-affine case is PSPACE-complete. The final section, Section 5, reviews the contributions of this paper, and discusses some related directions that have not been pursued in this work.

1.2 Background and motivation

Here are a few comments on the background to this research. The connection of Dynamical System Theory to the Theory of Computation is obvious—any traditional model of computation is a discrete-time dynamical system. However, most literature in Dynamical System Theory refers to a continuous state-space, whereas in the Theory of Computation, most models are discrete (but analog models *are* studied and cross-fertilization with Dynamical System Theory is evident; see for example [54]).

Taking classical (discrete) computability to continuous dynamics, Moore [47] discusses the significance of undecidability (in the Turing sense) to dynamical systems. He shows that a Turing machine can be simulated by a piecewise-affine map on the plane—the method is quite similar to the one used by Blondel et al. He concludes that for such a map, the set of points on which the sequence converges (to a particular zone) is not recursive. Koiran, Cosnard, and Garzon showed that such simulations can be done in two dimensions, but not in one [38]. Blondel and Tsiklitis [10] survey applications of Discrete Computability and Complexity to dynamical systems, including the above-cited results.

The author's interest in the mortality problem and its variants arose from their interpretation as special cases of the program termination problem (the latter term often refers to termination for any input, that is, a global property as those studied here). Note that the mortality problem can be seen as a question on the termination of the loop `while` $(x \neq 0)$ x `:=` $f(x)$, and global convergence to a fixed point asks about the termination of the "fixed-point seeking" loop `while` $(f(x) \neq x)$ x `:=` $f(x)$. In fact, the pattern of "iterate until stable" (rather than until a known value, such as 0, is reached) is ubiquitous in Computer Science and the problem of termination is of obvious interest.

Decision procedures for the termination of *simple loops*, where a fixed (loop-free) computation is iterated until an end-condition is met, have gained much interest in program analysis and several heuristic approaches have been proposed (e.g., various constructions of ranking functions [6, 13, 19, 52]). Note that these works concentrate on integer data, and on functions which are linear, piecewise linear, or defined by linear constraints (which is a wider class). In [58], Tiwari draws on inspiration from Dynamical System Theory to solve a termination problem for loops with an affine-linear update function—however, over the reals. Consequently, Braverman [14] tackled the problem for the rationals and integers. Passing from the real-number world to the integers is sometimes quite a challenge as the theory of integers has many surprises of its own. A notorious example is the solution of multivariate polynomial equations—or, more generally, quantifier elimination—decidable for the reals [57], but not for integers [62]. Another classical example of an integer-specific problem is the Collatz problem (or "$3x + 1$ problem") [41]. Lagarias' excellent volume shows clearly that this problem is related both to Dynamical System Theory and to Computability Theory. Regarding Computability, Conway [20, 21] and several subsequent works [37, 16, 24, 43, 40] proved undecidability results for *generalised Collatz problems* by showing how to simulate a counter machine. We shall make essential use of this idea, building on the reductions in [40], which were the only ones (known to the author) to derive Π_2^0 hardness of a mortality problem. Novel variations of the constructions will be presented in order to obtain stronger results (hardness of mortality for restricted classes of functions).

For completeness, we should also mention the works on dynamics of algebraic or rational maps over the integers (or more general number fields), e.g., [55]. These represent a different direction of transferring dynamical-system style problems to a discrete setting. There are kinds of dynamical systems even more remote from our subject, which the interested reader may find in (e.g.) Brin and Stuck's book [15]. There is also a decision problem called "mortality" which is quite different from our problem; it deals with a set of matrices, and asks whether the zero matrix is a product of some sequence of matrices from this set. See [50, 33, 3]. One may phrase the essence of the difference between the problems in that our mortality problem ask whether all trajectories lead to zero, while their problem asks whether there is a trajectory leading to zero (in a sort of non-deterministic dynamical system).

Part of this work has been previously presented at STACS 2013 [5].

1.3 Preliminary definitions

A closed (respectively open) *half-space* of $\mathbb{R}^n$ is the set defined by $\{\vec{x} \in \mathbb{R}^n : c\vec{x} + d \geq 0\}$ (respectively > 0) where $c \in \mathbb{R}^n, d \in \mathbb{R}$. We are interested in *rational* half-spaces, where the components of c, d are rational. A (rational) *convex polyhedral region* is the intersection of a finite number of (closed or open) half-spaces, which we sometimes call *the constraints.*

Definition 1.3. A *piecewise affine function* (PAF) on $\mathbb{R}^n$ (respectively $\mathbb{Z}^n$) is a function defined by

$$f(\vec{x}) = A_i\vec{x} + b_i \quad \text{for } \vec{x} \in H_i \text{ (respectively, } H_i \cap \mathbb{Z}^n) \tag{1}$$

where the sets $H_1, \ldots, H_p$ are an exhaustive partition of $\mathbb{R}^n$ into p convex rational polyhedral regions, and for $i = 1, \ldots, p$, $A_i \in \mathbb{Q}^{n\times n}$ and $b_i \in \mathbb{Q}^n$ (respectively, $\mathbb{Z}^{n\times n}$ and $\mathbb{Z}^n$).

The restriction to convex regions is somewhat arbitrary, but follows the definition in [9] and other related publications. Section 2.2 discusses an implication of this restriction (and proposes a relaxation). If f maps $\mathbb{N}^n$ into $\mathbb{N}^n$, we can speak of mortality over the naturals. While this research initially focused on $\mathbb{Z}^n$, comments about the natural-number setting have been added (thanks to a suggestion by a referee).

We use the notation $[a, b]$ for an interval of integers, namely $\{a, a + 1, \ldots, b\}$.

Counter Machines The counter machine model, due to Minsky [46], is well known. The details of the definition vary in the literature, but the differences are rarely essential. The following description conforms (up to non-critical details) with [9]. A counter machine M is specified by the number n of registers and the number ℓ of internal (control) states, plus a list of instructions. An instruction is an $(2n+2)$-tuple $[i, \beta_1, \ldots, \beta_n, D_1, \ldots, D_n, k]$ where $1 \leq i \leq \ell$ is the internal state number, $\beta_j \in \{Z, P\}$ represents whether register R_j is Zero or Positive, D_j is either $+1$, -1 or 0 and represents the change to R_j, and k is the new internal state, or 0, which signifies halting. Instructions have to be valid: they never decrement a null register. A configuration of the machine is written as $(i, \langle r_1, \ldots, r_n \rangle)$ where i is the internal state and r_j the contents of R_j. The machine is assumed to be deterministic, meaning that exactly one instruction is enabled at any configuration. State 1 is the *initial state* of the machine. Note that the machine does not receive any input while computing, so when used to compute a function, the argument is supposed to be present in some register upon commencement say R_1; the *initial configuration* for input x is thus $(1, \langle x, 0, \ldots, 0 \rangle)$. Occasionally, one considers functions of k arguments, which are supplied in the first k registers.

Since counter machines in themselves are not our goal, we grant ourselves, in some constructions, the convenience of using a slightly stronger computational model, namely the enhanced counter machine defined next.

Definition 1.4. An *enhanced CM* is a counter machine, as specified in Section 1, except that in an instruction $[i, \beta_1, \ldots, \beta_n, D_1, \ldots, D_n, k]$, each D_j component is an integer in $[-1, \infty)$, which is added to R_j. Hence, an increase to the value of a register can be any positive constant.

The class Π^0_2.

Definition 1.5. Π^0_2 is the class of decision problems that can be expressed by a formula of the form $(\forall z)(\exists y)P(x, y, z)$ with P recursive.

This class properly contains Σ^0_1 (characterised by formulas $(\exists y)P(x, y)$), also known as RE or the recursively enumerable problems (sometimes called computationally enumerable). Standard Π^0_2-complete sets are the totality problems (termination on all inputs) for Turing-equivalent machines, such as counter machines. Kurtz and Simon extended the hardness result to mortality of counter machines, in the following sense:

Definition 1.6. For a class of deterministic abstract machines (equivalently, programs with an operational semantics), the mortality problem is the problem of deciding, given such a machine (or program), whether it is the case that for every configuration C in the configuration-space of this machine (or program), its computation from C is finite (i.e., reaches a halting configuration).

For example, given a counter machine as described above, the question is whether the computation from any configuration $(i, \langle r_1, \ldots, r_n \rangle)$ leads to halting; the computation is unique because the machine is deterministic. Note the difference from the totality problem, which only concerns halting for all input values—i.e., halting of computations from initial configurations $(1, \langle x, 0, \ldots, 0 \rangle)$.

While undecidability of totality is a basic result in computability that applies to all computational models effectively equivalent to a Turing machine, undecidability (let alone Π^0_2-completeness) of mortality is non-trivial to prove since many programs, while halting from all initial states, still diverge if started in a configuration that is not reachable in a proper computation.[2] For Turing machines, Hooper [35] and Herman [34] proved the undecidability of mortality (under two different definitions of the state space). Kurtz and Simon proved the following

THEOREM 1.7 ([40]). *The mortality problem for counter machines (CMs) with $n \geq 2$ counters is Π_2^0-complete.*

2 Undecidability in Two Dimensions

This section is dedicated to undecidability results. The first subsection proves results corresponding to the undecidability results of Blondel et al.—but over the integers, and proving Π_2^0 hardness. The next subsections discusses the question: do certain restrictions on the space of functions turn the problems to decidable ones? First, we consider constant bounds on the number of regions, and similar parameters. Next, we consider severe restrictions on the coefficients of the functions, defining the class of *monic* functions and proving undecidability for this class. This also yields a new result for functions over the rational numbers (Theorem 2.24).

2.1 The general case

Blondel et al. prove the undecidability results by reducing from the mortality problem for counter machines. This reduction encodes counter-machine states in rational numbers, making essential use of fractions 2^{-n} in representing a counter of value n (there are a few other constructions in the literature which use a similar encoding). For the integer setting, we need a more integer-oriented technique. In [40], Kurtz and Simon reduce the CM mortality problem to a Generalised Collatz Problem. We shall use this problem to prove our result for piecewise affine functions.

The definition below is based on [40], with a non-essential modification for convenience in the sequel.

Definition 2.1. A function $g : \mathbb{N}_+ \to \mathbb{N}_+$ is called a *generalised Collatz function* if there is an integer $m > 0$, positive integers $\{a_0, \ldots, a_{m-1}\}$ and non-negative integers $\{b_0, \ldots, b_{m-1}\}$, such that whenever $x \equiv i \bmod m$, $g(x) = a_i(x - i)/m + b_i$.

A *standard representation* of g is the list $m, a_0, b_0, \ldots, a_{m-1}, b_{m-1}$ (in binary notation).

The standard Collatz function is usually described by $g(x) = 3x + 1$ if x is odd, $g(x) = x/2$ if x is even (the *Collatz conjecture* is that the trajectories of this function all reach 1). In our notation, g is given by $m = 2$, $a_0 = 1, b_0 = 0$, $a_1 = 6, b_1 = 4$.

Definition 2.2. GCP (for Generalised Collatz Problem) is the problem of deciding, from a standard representation of g, whether every trajectory of g reaches 1.

THEOREM 2.3. [40] *GCP is Π_2^0-complete.*

[2] An anecdotal evidence of this difference is that for Petri nets, halting is EXPSPACE-complete [25, 42, 8, 23], while mortality is PTIME [45, 26].

Note that the GCP is really a mortality problem (with the end-state defined as 1 instead of 0), but the functions considered in the GCP are not piecewise affine; their expression involves division and remainders, which make it easier to encode computations and simulate counter machines.

Our first result is

THEOREM 2.4. *Global convergence, mortality and global convergence to a fixed point of piecewise affine functions with integer coefficients over $\mathbb{Z}^2$ (or $\mathbb{N}^2$) are Π^0_2-complete problems.*

Proof. Like the GCP, mortality and convergence (to zero) are clearly "$\forall\exists$" problem, as they can be formulated as follows: for all $\vec{x} \in \mathbb{Z}^2$, there is an n such that $f^n(\vec{x}) = 0$. For global convergence to a fixed point, we formulate it as: for all $\vec{x} \in \mathbb{Z}^2$, there is an n such that $f^n(\vec{x}) = f^{n+1}(\vec{x})$. Hence, all these problems clearly belong to Π^0_2. For Π^0_2-hardness, we reduce from the GCP.

Given a description $\langle m, a_0, b_0, \ldots, a_{m-1}, b_{m-1}\rangle$ of a generalised Collatz function g, our reduction produces the function f defined by the following table, where for convenience every region has a label.

region label	constraints	$f(x, y)$
D	$x \geq m+1$, $y \geq 0$	$(x - m, y + 1)$
R_0	$x = 0$, $y \geq 1$	$(a_0 y + b_0, 0)$
R_1	$x = 1$, $y \geq 1$	$(a_1 y + b_1, 0)$
R_2	$x = 2$, $y \geq 0$	$(a_2 y + b_2, 0)$
$\vdots$		
R_{m-1}	$x = m - 1$, $y \geq 0$	$(a_{m-1} y + b_{m-1}, 0)$
Z	elsewhere	$(0, 0)$

To simulate a Collatz sequence generated by g, we repeatedly apply f starting from $(x_0, 0)$. Observe that started at $(x, 0)$ for $x > 1$, the computation will stay in Region D (the *division* region) until obtaining the result $(i, (x - i)/m)$ where $x \equiv i \bmod m$. Computation will then reach one of regions R_0 through R_{m-1} and apply the appropriate part of g, producing $(g(x), 0)$. This process only stops if it arrives at $(1, 0)$, which is the final point of the Collatz sequence and is mapped by our function to $(0, 0)$, our final state. For example: the standard Collatz sequence, started with $x_0 = 5$, is $5, 16, 8, 4, 2, 1$. The simulation of this computation by our two-dimensional function is shown in Figure 1; note that the points on the x-axis correspond to the Collatz sequence (except for the final step jumping to $(0, 0)$). The points above the x-axis correspond to the process of division.

Note that every point in the regions D through R_{m-1} represents an intermediate state of a valid simulation of a g sequence, while all other points map immediately to

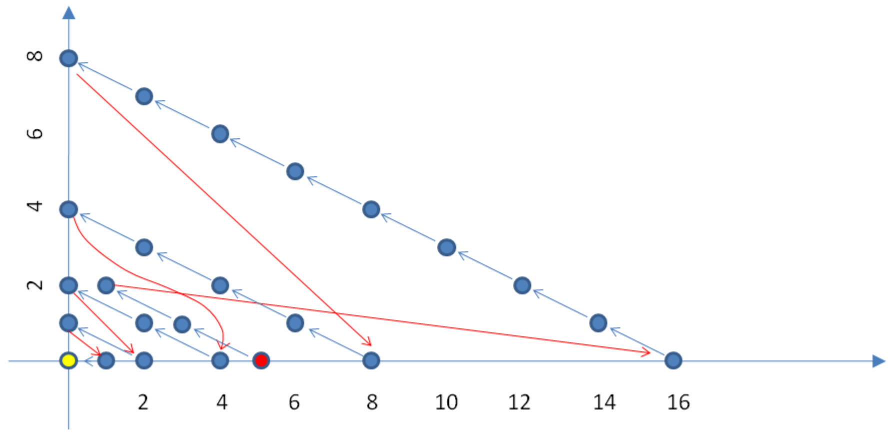


Figure 1: Simulating the Collatz sequence from $x = 5$ (the initial point is $(5, 0)$).

the origin. Thus, f globally converges to zero if and only if it is mortal if and only if g satisfies the GCP. It is furthermore easy to verify that the function has no fixed point besides the origin. Therefore, it globally converges to a fixed point if and only if g satisfies the GCP. □

We observe that this reduction also works over the rationals, which will allow us to solve an open problem of [9] in the next subsection. More precisely:

OBSERVATION 2.5. *Function f as constructed in the last proof is mortal over the rationals if and only if it is mortal over the integers.*

Proof. It is clear that if f is not mortal over the integers, it is not mortal over the rationals. In the other direction, suppose that f is mortal over integers. We consider a trajectory that starts with a point (x, y) where at least one of x, y is non-integral. We have the following cases:

1. $(x, y) \in Z$. In this case it is mapped to $(0, 0)$.
2. $(x, y) \in D$ and x is integral and y is not, the trajectory inevitably leads to one of the regions R_i, hence to the next case.
3. (x, y) is in one of the regions R_0 through R_{m-1}. In this case the second coordinate of $f(x, y)$ is 0. Write $f(x, y) = (x', 0)$. If x' is integral, the rest of the trajectory is integer-valued, and presumed mortal. If x' is non-integral, but smaller than m, then $(x', 0) \in Z$. Otherwise, it falls under the following case.
4. $(x, y) \in D$ and x is non-integral. Then the trajectory inevitably leads to Z.

□

In the following sections we prove results which are strictly stronger than Theorem 2.4. However some of their proofs will reuse the above reduction.

2.2 Undecidability with a Bounded Number of Regions

The number of regions in the definition of function f above depends on the modulus m of the Collatz function. In this section, we establish that the mortality problem is also hard when the number of regions is bounded by some *a priori* constant (if it is large enough). This will follow immediately from proving that a result like Theorem 2.3 holds for a fixed modulus (which may be interesting in itself). For this purpose, we modify the reductions leading up to this theorem[3]. In [40], the mortality problem for counter machines is proved Π_2^0-hard by reduction from the standard complete problem for this class, the totality problem (does a given CM halt on every input?). Then, CM mortality is reduced (via an intermediate form, called FracTran) to the GCP. The modulus of the resulting Collatz function is determined by the size of the counter machine—more precisely, by the number of instructions and the number of counters. For any given bound B, mortality of counter machines whose size is bounded by B is decidable (there are only finitely many such machines). Therefore one cannot simply impose a bound and use the same reduction. Our proof is a new reduction that uses an extended type of CM instructions and a universal machine (note, however, that while the halting problem for a universal machine is undecidable, a universal machine is never mortal).

A reduction of a CM halting problem to mortality appears both in [40] and in [9]. Since the latter reduction (henceforth: the BBKPT reduction) is simpler, we will build upon it, however with certain modifications. First, they reduce from halting on a given input, not from totality. Secondly, we make some changes for the purpose of improving the bounds we can give on the size of the machine, with an eye towards bounding the modulus in the GCP.

First, we review the BBKPT reduction. Suppose that we are given a counter machine M with n counters $R_1, \ldots, R_n$ so that we are to determine if it halts on the initial state $(1, \langle r_1, \ldots, r_n \rangle)$. They construct a machine M' with $n+2$ counters $R_1, \ldots, R_n, V, W$.

The machine M' has a special new "reset" state q_0. Once M' enters q_0, it executes a sequence of instructions whose effect is to set $R_1, \ldots, R_n$ to $r_1, \ldots, r_n$ respectively, store $2V+1$ in W and 0 in V. After having done that, it moves into state 1, the initial state of M. Note: in the BBKPT reduction, $r_1, \ldots, r_n$ are fixed (as input to the reduction), so it is possible to set R_i to r_i by a hard-coded sequence of increments. As a consequence of our higher goal, to prove Π_2^0 hardness and not just RE-hardness, thus to reduce from the totality problem and not from the halting problem, we will not have the input hard-coded. The solution will be explained below.

The operation of M' in the states taken from M is such that it simulates M while also performing the following operations: for every step, it increments V and decrements W. It only performs the next instruction of M if $W > 0$. If $W = 0$, it returns to the reset state.

Blondel et al. show that this ensures mortality if and only if M halts on the specified initial state. This follows from the following observations.

[3]Undecidability, though not Π_2^0 completeness, has been shown to hold for a fixed modulus in [37, 24, 43]. Their constructions could perhaps be used to give an alternative proof of the result.

1. For M' to halt, it must halt while simulating M, since the added instructions do not halt in themselves. If M has a non-terminating computation from initial state $(1, \langle r_1, \dots, r_n \rangle)$, then M' cannot halt when starting from such state, and is not mortal.

2. Suppose that M halts on the given state. Then, when M' is started with an *arbitrary* initial state, one of two things can happen: either M' halts without reaching a state where $W = 0$, or it reduces W to 0. When this happens, M' re-initialises itself to start a simulation of M from $(1, \langle r_1, \dots, r_n \rangle)$, which can run for $2V+1$ steps. It will either halt, or exhaust the "budget" and re-initialise. However, since the budget grows whenever it is re-computed, eventually the simulation of M to its halting point will be successful.

Next, we apply this technique to a universal machine, in a way that yields a reduction of the totality problem to morality, and moreover, for machines with a bounded number of registers and control states.

Definition 2.6. By U we shall denote a universal CM, such that when started with a value x in R_1, and an encoding $\widehat{M}$ of a machine M in R_2 (the choice of an encoding is immaterial to our discussion), it simulates the computation of M when started with x in its first register and 0 in the rest.

The notion of simulation that we need is weak in the sense that we are not interested in output; it suffices that U halts if and only if M halts on $\langle x, 0, \dots, 0 \rangle$. We can also restrict M to any class of counter machines that is sufficiently strong to make the totality problem Π_2^0 complete; for example, two-counter machines. The details of the machines are not very important, unless one wants to calculate certain constants explicitly. At the request of one of the referees that have reviewed this paper, such an explicit calculation will be given. But at this stage, we proceed with the non-specific description that should make it easier to see the essential ideas.

Since counter machines in themselves are not our goal, we can grant ourselves the convenience of using a slightly stronger computational model, namely the enhanced counter machine (Definition 1.4), where a constant of any size can be added to a register at one step.

PROPOSITION 2.7. *From an (ordinary) counter machine M and a universal machine U, one can compute an enhanced counter machine U_M such that U_M is mortal if and only if M halts on every initial state $(1, \langle x, 0, \dots, 0 \rangle)$. The number of registers and control states of U_M are independent of M.*

Proof. U_M is obtained by applying the BBKPT reduction to the universal machine U, with the following modifications. Let n be the number of registers in U. We add a register R_{n+1}. It will be used to keep an input value on which we simulate M. Further, we add two registers fulfilling the roles of V and W, as previously described. The reset state, besides the operation already described above, will set $R_1, R_2, \dots, R_{n+1}$ to $R_{n+1}, \widehat{M}, 0, \dots, 0$ respectively. Copying R_{n+1} into R_1 requires a loop, of course; or, more precisely, two loops—one to copy R_{n+1} to R_1, while decrementing R_{n+1}; we copy it at the same time also to an auxiliary register, say R_3; the second loop recovers R_{n+1} from R_3. Let us call the intermediate configurations encountered during these loops *fragile*, because in them, the invariant value of R_{n+1} is temporarily modified.

We now claim that U_M halts on all non-fragile configurations in which $R_{n+1} = x$ if and only if M halts on initial state $(1, \langle x, 0, \ldots, 0\rangle)$. First, suppose that M does halt on this initial state. When U_M is started in a configuration where $R_{n+1} = x$, which is not fragile, the value of this register will be maintained until the machine either halts or W is decremented down to zero. In the latter case, the reset operations prepare U_M to correctly simulate a computation of M on input x. The simulation will either halt or time out and reset again, but since the time limit is increased each time through, eventually the machine will halt. Conversely, suppose that M does *not* halt on input x; then starting U_M in the state $(1, \langle x, \widehat{M}, 0, \ldots, 0\rangle)$ we get a non-terminating computation.

In a fragile configuration, the value of R_{n+1} is not x, as it may be partially transferred into R_1; but x can nonetheless be determined from the configuration, and besides this subtlety, the above analysis holds.

We conclude that U_M is mortal if and only if M halts for every x. $\square$

THEOREM 2.8. *From a counter machine M (of the type interpreted by U), one can compute a generalised Collatz function g that satisfies the GCP if and only if M halts on every initial state $(1, \langle x, 0, \ldots, 0\rangle)$. The modulus of g depends only on the size of U; in particular, it is independent of M.*

Proof. We first map M to U_M as shown above, and then use a translation from counter machines to generalised Collatz functions based on [20, 40], extended to represent enhanced CM instructions and slightly modified to reduce the modulus value. The translation represents a machine state s by an integer $\widehat{s}$ as follows. Each register R_k (up to R_{n+1}, since we follow the numbering in the previous proof) is associated with a distinct prime number p_k. The contents of R_k are encoded as the exponent of p_k in the prime-number factorization of $\widehat{s}$. Let q be the number of control states in U_M (including 0, the halting state). In what follows, we assume the states have been renumbered so that the halting state is now number 1; this is done in order to obtain a stopping condition that suits the GCP problem, where the sequence is supposed to end at 1. Without loss of generality we assume that there is no other kind of halting (i.e., no control states with missing instructions). Moreover, we add to the machine instructions from state 1 to itself, conditional on a register being non-null, which decrement it; thus the machine will truly halt only after clearing all registers. Now, we encode a configuration $s = (\ell, \langle r_1, \ldots, r_{n+1}\rangle)$ by

$$\widehat{s} = q(\prod_{k \le n+1} p_k^{r_k}) + \ell\,.$$

The modulus m is defined to be $q \prod_{k=1}^{n+1} p_k$, which ensures that the remainder of $\widehat{s}$ modulo m determines the control state and, for every register, whether it is null. Specifically, the control state is $\widehat{s} \bmod q = (\widehat{s} \bmod m) \bmod q$; to test R_K for nullity, we consider whether $(\widehat{s} \bmod m)$ is a multiple of $p_k q$; if it is, R_k is non-null.

Thus there is a unique instruction corresponding to each remainder i. We define the function $x \mapsto a_i(x-i)/m + b_i$ so that it implements the corresponding instruction. To encode an instruction that adds D_k to R_k, a_i is chosen to be a multiple of $(p_k)^{D_k+1}$. Since m includes a p_k factor, the exponent of p_k will be increased by D_k. To change the control state to r, we choose b_i to make r the new remainder modulo q. To sum up, the instruction:

$$[\ell, \beta_1, \ldots, \beta_{n+1}, D_1, \ldots, D_{n+1}, r] \tag{2}$$

is simulated by letting

$$a_i = q\left(\prod_{i=1}^{n+1}(p_k)^{D_k+1}\right); \quad b_i = r\,,$$

for each $0 \le i < m$ where $i \bmod q = \ell$, and $(p_k | \lfloor i/q \rfloor) \iff (\beta_k = P)$.

Now, the halting state (with all registers null) is represented by $x \bmod m = q \cdot 1 + 1$. Let $i = q + 1$, then we let

$$a_i = q; \quad b_i = 1\,. \tag{3}$$

Thus, if $x \equiv i \pmod{m}$, the application of our Generalised Collatz function to x results in

$$a_i(x-i)/m + b_i = q(x-i)/m + 1$$

which is less than x, unless $x = i = q+1$; thus, simulating a halting transition triggers a chain of descending numbers ending up at 1. The definition (3) is also applied to any other $q+1 < i < m$ such that $i \bmod q = 1$ and $(i-1)/q$ is not divisible by any p_k. Such values may exist, for example, suppose that $q = 4$, $n = 1$, $p_1 = 2$, $p_2 = 3$; then $m = 24$, and $i = 21$ satisfies $i \bmod q = 1$, and $(i-1)/q = 5$ is indivisible by $2, 3$.

As in [40], we note that we are not satisfied by the Generalised Collatz function being able to simulate a computation of U_M, but we want its global behaviour to represent the mortality of U_M; it is therefore necessary to pay attention to the fact that some numbers do not represent any state according to the encoding function. Those are, to be exact, the numbers x such that $\lfloor x/q \rfloor$ has a prime factor larger than p_{n+1} (let us call it a *noise* prime). We should guarantee that starting from such a number will not cause divergence. This is indeed the case for the following reason: consider such a number

$$x = q\left(\prod_{i=1}^{n+1}(p_k)^{r_k}\right) \cdot d + \ell\,,$$

where d is the noise factor (a product of noise primes). Simulating an instruction (2) that applies, we reach

$$x' = q\left(\prod_{i=1}^{n+1}(p_k)^{r_k+D_k}\right) \cdot d + r\,,$$

so, in essence, the simulation is correct, maintaining the same noise factor d. We can call it a *d-trajectory* of the Generalised Collatz function (note that trajectories made out of proper encodings form 1-trajectories). An exception is when the halting configuration is reached; then (3) applies, so we have

$$x = qd + 1$$

and

$$x' = q \cdot \lfloor x/m \rfloor + 1 = q \cdot \lfloor \frac{q}{m} d \rfloor + 1 \,.$$

If $\lfloor (q/m)d \rfloor = 0$, we obtain 1; otherwise, since $\lfloor (q/m)d \rfloor < d$, this results in a number that belongs to an e-trajectory, for $e < d$; therefore the number of such events is finite (and we must end up at 1 or at a 1-trajectory). Revisiting our example, if $x = 21$ (with $d = 5$), then $x' = 1$; if $x = 45$ (also with $d = 5$), we have $x' = 5$, which is a proper encoding (i.e., in a 1-trajectory), as $5 = q \cdot 1 + 1$. □

COROLLARY 2.9. *There is a constant m such that GCP restricted to modulus m is Π^0_2-complete.*

Applying now the reduction in the previous section, we have

COROLLARY 2.10. *There is a constant r such that global convergence, mortality and global convergence to a fixed point of piecewise affine functions f over $\mathbb{Z}^2$ (or $\mathbb{N}^2$) with integer coefficients and r regions are Π^0_2-complete problems. This is also true for mortality of piecewise affine functions over $\mathbb{Q}^2$.*

It is natural to ask how large this constant bound has to be for the problems to be undecidable. Moreover, one may expect to develop a tradeoff between the number of regions and dimension, of the kind investigated for a number of undecidable problems, regarding their characteristic parameters, see [44]. This is, however, a plan for future research. Presently, we only provide an estimate on the size of the constants m and r in the two last propositions. Then, we discuss alternative parameters for the descriptional complexity of the function: the number of non-zero regions, and the number of defining hyperplanes.

Bounds for the above constructions. We give some bounds for the constants m and r appearing in the last two propositions. These bounds are probably far from lowest; but they are not the worst that can come out of the above proof. The main factor in the determination of the bound is the size of the machine U_M. The construction of the modulus of the GCP function in the proof of Theorem 2.8 shows that this value grows exponentially in the number of registers, but only linearly in the number of states of the machine. Consequently, in order to obtain a good bound, we should try to optimise the machine mostly in terms of the number of registers.

LEMMA 2.11. *The construction of Proposition 2.7 can be implemented in a way that results in a machine of three registers and 233 control states.*

The proof of the lemma is just a lot of technical details, and is deferred to Appendix A. By setting these parameters in the proof of Theorem 2.8, we get the following value for the modulus:

$$m = 233 \cdot 2 \cdot 3 \cdot 5 = 6990.$$

The number of regions, r in the second proposition, is of a similar magnitude.

Remark: as already mentioned, in [20, 40], the construction of the GCP was slightly different. Significantly, their modulus grows exponentially in the number of states as well. The bound on m that followed from their construction was huge (roughly in the order of 10^{35}), hence the motivation to improve the construction.

Discussion The results of this section can be seen as an investigation of the question: how complicated need the function be to make its behaviour computationally unpredictable. One can come up with other measures for the descriptional complexity of the function, for example, the size of a linear decision tree that computes the function. Such a tree is an ordered finite binary tree each of whose internal nodes represents a "decision", asking for a certain affine linear function h whether $h(x) > 0$. Every leaf represents a region on which the defined function is affine linear. The class of functions defined by up to B regions, for some constant B, also has a constant bound on the size of the decision tree (defined as the number of leaves), though the constant may differ.

As the estimate of circa 7000 regions is probably way beyond the true decidability threshold, one can look at another indication of hardness: the connection to the $3x + 1$ problem. In fact, this problem can be encoded as a mortality problem with just three regions (or a decision tree with four leaves), as follows (this is, essentially, an ad-hoc implementation of the method of Theorem 2.4, economizing on regions):

$$f(x, y) = \begin{cases} (x - 2, y + 1) & \text{when } x > 0 \\ (y - 1, 0) & \text{when } x = 0 \\ (3y, 0) & \text{when } x < 0\,. \end{cases}$$

PROPOSITION 2.12. *Function f above is mortal if and only if the Collatz conjecture holds.*

Proof. First, we note that the Collatz conjecture is equivalent to stating that any sequence starting from $x_0 \geq 2$ will reach 2. We thus simulate the sequence only for such values, and 2 is our stopping point. By "shifting the axis," namely representing a number x in the sequence by the point $(x - 2, 0)$, arrival of all sequences at 2 coincides with mortality. Also, we note that for odd x we can map it to $(3x + 1)/2$ (shortcutting a step of the original sequence).

Consider a point $(x - 2, 0)$ for $x \geq 2$. We claim that iterating f leads to $(x' - 2, 0)$ where x' is the element following x in the Collatz sequence:

If x is even (and greater than 2, the case 2 is trivial),

$$(x - 2, 0) \mapsto^* (0, (x - 2)/2) \mapsto (((x - 2)/2) - 1, 0) = ((x/2) - 2, 0)\,.$$

If $x > 2$ is odd,

$$(x-2,0) \mapsto^* (-1,(x-1)/2) \mapsto ((3(x-1)/2),0) = (((3x+1)/2)-2,0)\,.$$

What about other points of the plane?

1. All points in the closed NE quadrant participate in orbits as above.
2. A point $(x,0)$ with $x < 0$ is mapped to $(0,0)$.
3. A point in the open SE quadrant will reach either the positive x axis (then it falls back to case 1) or the negative y axis.
4. A point $(0,y)$ with $y < 0$ is mapped to $(y-1,0)$, a point on the negative x axis.
5. A point (x,y) with $x < 0$ and $y \neq 0$ is mapped to $(3y,0)$, which is either on the positive x axis (hence OK) or on the negative x axis.

□

Here, too, f can be interpreted as a function either over $\mathbb{Z}^2$ or over $\mathbb{Q}^2$. We conclude that, in each of these settings, by finding a decision procedure for two regions we would get as far as possible as one can get, in terms of this parameter, without implicitly solving the Collatz problem. A complete solution for two regions is conjectured possible but left out of the scope of this work. For a single region the problem is settled in Section 4.

2.3 The number of non-zero regions

Consider the function f defined in Section 2. How many regions does it have? There are $m+2$ rows in the table. But the last one does not count as a single region according to Definition 1.3. The problem is that it is not convex, and therefore has to be split in a "meaningless" way into convex polyhedra. Suppose that we adjust the way we count regions by allowing the function to be defined explicitly on a certain number of convex polyhedral regions, and *zero elsewhere*. We can now look for the smallest number $\mathcal{R}$ of defined regions (not counting the "elsewhere") where mortality or convergence become undecidable. Under this definition, the case $\mathcal{R} = 1$ coincides with an important open problem from the area of static program analysis, known as the termination of affine integer loops. Such loops are defined by an affine function over a convex polyhedron, and terminate whenever the value leaves that region. In our setting, assuming that the region is shifted (if necessary) to exclude the origin, mortality becomes equivalent to leaving the active region. This problem has been settled in part by Braverman [14], who showed decidability in the homogenous case (i.e., using functions whose constant term is zero; in other words, linear functions rather than affine-linear). For a recent progress (solving the problem under a different restriction), see [51].

In the program-analysis context, the case $\mathcal{R} = 2$ represents loops with a single "if statement" in their bodies (where the test is a linear inequality). The termination problem of such loops has been recently shown to be undecidable [7]. This work does not consider a fixed dimension. However, the undecidability proof is a reduction from mortality of counter machines, and the dimension of the dynamical system constructed is related to the size of the counter machine. Using Proposition 2.7, we can deduce that undecidability (Π_2^0 hardness, in fact) holds under a certain fixed bound on the dimension.[4]

2.4 Undecidability for Monic Functions

The result of the last section may be interpreted as showing that the hardness of the problem does not depend on allowing an unbounded number of regions. So, it must come from allowing an unbounded set of possible affine functions for each region. In this section we will show that even when restricting the coefficients in the linear components of these functions to 1, we still have undecidability.

Definition 2.13. A *monic*[5] piecewise affine function on $\mathbb{Z}^2$ is a piecewise affine function where each of the defining affine components has the form $f(\vec{x}) = (x_{i_1} + b_1, x_{i_2} + b_2)$, where $\{i_1, i_2\} = \{1, 2\}$. In addition, the halfspaces which define the space partition are given by inequalities of the form $\pm x_i \leq d$.

The main point of this definition is to exclude multiplication by a constant, in the form ax_j with $a \neq 1$ (or sums like $x_1 + x_2$, which can be used to quickly double a value). The reason for this choice was that the previous reduction relied significantly on multiplication, thus it seemed interesting to see what happens without this option. For consistency, the option to multiply by a constant was also removed from the expressions that define the boundaries of regions in the function's definition. But these choices are quite arbitrary, and one can certainly find similar phenomena when considering other function classes, slightly more or slightly less restricted. Such variations will be discussed in Section 2.4.5.

2.4.1 Two-counter machines

In this proof we will use 2-counter machines (2CM). A 2CM is a counter machine as previously described, just with two counters.

LEMMA 2.14. *2CM mortality is Π_2^0-complete.*

Proof. Blondel et al. [9] show that Minsky's well-known translation of n-counter machines to 2CM preserves mortality. Combining this with the Π_2^0-completeness result of Kurtz and Simon [40], we obtain the lemma. $\square$

[4]Some back-of-the envelope calculations in the style of those leading to Lemma 2.11 suggest a bound in the order of a few hundreds. The tedious calculation has not been carried out in full. This may be far from the true edge of decidability, suggesting a problem for further research.

[5]Usually, a "monic polynomial" is univariate. Our definition requires each coordinate of the result to be a univariate monic linear function of one of the variables.

It will be useful to restrict the machines under consideration to a certain class of "normal forms", as described in the following proposition.

LEMMA 2.15. *Every 2-counter machine M can be effectively transformed into a machine M' such that M' is mortal if and only if M is, and, in addition,*

1. *In every transition of M', all the components D_j are non-zero. That is, all registers are modified.*
2. *Whenever control is transferred to the halting label, 0, the values of the counters are either $\langle 0,0\rangle$, $\langle 0,1\rangle$ or $\langle 1,0\rangle$.*

We shall call such a machine normalised.

Proof. Arranging for (1) to hold is done by increasing the number of control states, so that as M' is simulating M, the value of each register may deviate by 1 from its value in M; the deviation is recorded in the finite control. We also record, in the finite control, information on whether the counter is currently null (in the simulated state). The deviation will, in general, be 0 or -1 (that is, the value of the M' register may be one less than the simulated register), except when the simulated register is null, where the M' register may be set to 1, to maintain the goal of always modifying the register.

To verify this idea in more detail, we present in Table 2 the translation applied to a single-register machine. The construction for a two-register machine consists of applying the translation, independently, to both parts of each instruction. In the translation showed, each original state i becomes a set of states $i^{(d)}$ where $d \in \{-1,0,1\}$ represents the deviation of the M' register from the simulated regiseter. Note that certain situations cannot arise in a simulation; specifically, a state $i^{(1)}$ cannot be entered unless the counter is null. To preserve mortality, besides translating the instructions of M according to the table, we also add instructions to map the "invalid" configurations to the halting state.

For part (2) of the lemma, a halting transition is changed into a transition into a new set of states that (almost) empty the counters (one cannot always nullify both counters because of (1)). □

2.4.2 Compass Collatz-like functions

The proof will use a specially-adapted variant of the Collatz problem, presented next (the definition may seem a little forced; keep in mind that it was conceived as a tool for reductions). In describing this kind of Collatz-like function we will make use of the set $\mathbf{C} = \{E, N, W, S\}$ of Compass Directions. A pair (x, Δ), where $x \in \mathbb{N}$, may be depicted as a point on one of the axes in the Cartesian plane, in the direction Δ (we may also refer to such a point as lying *on the Δ axis*). We refer to it as *a compass point.*

A *Compass Collatz-like function*, defined next, differs from the previously-used generalised Collatz function in several respects. First, it operates on compass points. Consequently, the definition of $f(x, \Delta)$ is split into cases both according to the direction

Original instruction	Translation
$[i, Z, 1, k]$	$[i^{(0)}, Z, 1, k^{(0)}]$,
	$[i^{(1)}, P, -1, k^{(-1)}]$
$[i, Z, 0, k]$	$[i^{(0)}, Z, 1, k^{(1)}]$
	$[i^{(1)}, P, -1, k^{(0)}]$
$[i, P, a, k]$, $a \neq 0$	$[i^{(0)}, P, a, k^{(0)}]$
	$[i^{(-1)}, P, a, k^{(-1)}]$
	$[i^{(-1)}, Z, 1, k^{(-a)}]$
$[i, P, 0, k]$	$[i^{(0)}, P, -1, k^{(-1)}]$
	$[i^{(-1)}, P, 1, k^{(0)}]$
	$[i^{(-1)}, Z, 1, k^{(0)}]$

Table 2: modifying instructions to ensure the counter is always modified.

component Δ, and according to the remainder of the integer x modulo a certain number m. However, we do not specify the function by listing its action for every conjugacy class modulo m, because our functions operate quite uniformly, so in fact the specification requires less data than the specification of a generalised Collatz function of the same modulus. For points in the North and South directions, the action is expressed by a single, simple formula; for points in the East and West directions, the conjugacy classes are split between two formulas, so a description of the function includes listing the sets of remainders that are handled by each of the formulas. The formal definition follows; Figure 2 is a pictorial representation.

Definition 2.16. A function $g : \mathbb{N}_+ \times \mathbf{C} \to \mathbb{N}_+ \times \mathbf{C}$ is called a *Compass Collatz-like function* if there is a number $m = 6p$ with $p \geq 5$ a prime, sets $R_N, R_S \subseteq [0, m-1]$ and integers $w_i \in [0, m-1]$ for $i = 0, \ldots, m-1$, such that g satisfies the following equations (for convenience we represent its argument in the form $mx + rp + i$, where $x \geq 0$, $0 \leq r < 6$ and $0 \leq i < p$):

$$\begin{aligned}
g(mx+rp+i, E) &= \begin{cases} (mx+rp+i, N) & rp+i \in R_N \\ (4(mx+rp)+i, N) & rp+i \notin R_N \end{cases} \\
g(mx+rp+i, N) &= (\frac{1}{2}mx + \lfloor \frac{1}{2} r \rfloor \cdot p + i, W) \\
g(mx+rp+i, W) &= \begin{cases} (mx+rp+w_{rp+i}, S) & rp+i \in R_S \\ (9(mx+rp)+w_{rp+i}, S) & rp+i \notin R_S \end{cases} \\
g(mx+rp+i, S) &= (\frac{1}{3}mx + \lfloor \frac{1}{3} \cdot r \rfloor p + i, E)
\end{aligned} \tag{4}$$

As an example, let $p = 5$, so $m = 30$, and let us compute $g(42, S)$. First, we write

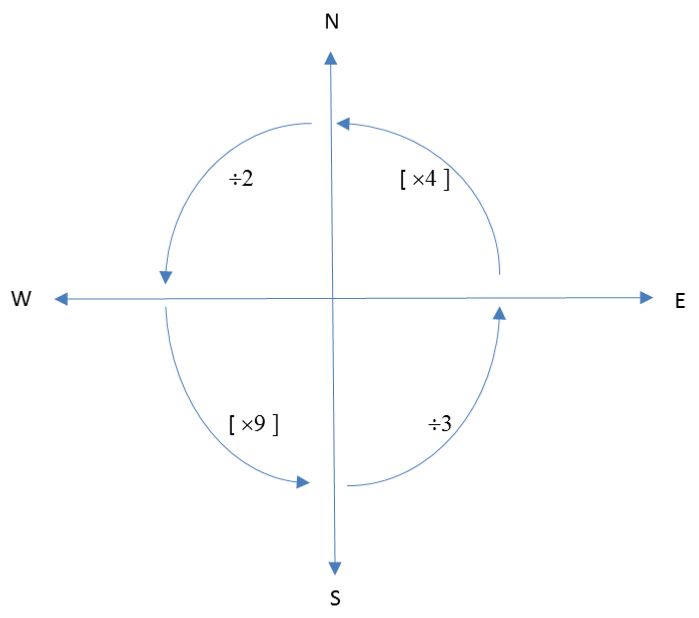


Figure 2: The dynamics of a Compass Collatz-like function.

$42 = m + 2p + 2$. Then, by definition,

$$g(42, S) = (\frac{1}{3}m + \lfloor\frac{2}{3}\rfloor p + 2, E) = (10 + 0 + 2, E) = (12, E)\,.$$

Next, consider $g(42, E)$: here we branch according to the remainder modulo m, namely 12. If $12 \in R_N$, we have $g(42, E) = (42, N)$, while if $12 \notin R_N$, then $g(42, E) = (4 \cdot 40 + 2, N) = (162, N)$.

The function could be represented as an ordinary generalised Collatz function by encoding the "direction" in the remainder modulo a prime $q \neq 2, 3, p$, replacing $m = 6p$ by $6pq$. For every congruence class modulo $6pq$, the function is then linear. The "compass" representation is useful for the transition to a 2-dimensional dynamical system (and also makes it easier to visualise the dynamics). Another "twist" that makes the next proof easier is the definition of the final zone, below, as the set $\{(x, E) \mid x < m\}$ rather than just the point 1.

Definition 2.17. CCP (for Compass Collatz-like Problem) is the problem of deciding, given g, whether every g-trajectory eventually reaches the final zone $\{(x, E) \mid x < m\}$.

LEMMA 2.18. *CCP is Π^0_2-complete.*

Proof. We reduce from 2CM mortality. Given a normalised 2-counter machine M (Lemma 2.15), we show how to simulate it by a Compass Collatz-like function.

Let $m = 6p$ where $p \geq 5$ is a prime such that the number of internal states of M is $p - 1$ (there is no loss of generality, since extra states can always be added). The idea is to represent a configuration $s = (i, \langle r_1, r_2\rangle)$ by $\widehat{s} = (2^{r_1}3^{r_2}p + i)$. For a number of this form, $(\widehat{s}, E)$ will be called *a proper state*. A pair (y, E) with $y > 0$ which is not a proper state is an *improper state*.

We design a CCP that takes every proper state $(\widehat{s}, E)$, in a bounded number of steps, to a $(\widehat{s'}, E)$ where s' is the successor configuration to s.

Let $\widehat{s} = 2^{r_1}3^{r_2}p + i = mx + rp + i$, where $0 \le r < 6$. The remainder r determines whether each of r_1 and r_2 is zero or positive; hence, r and i determine the instruction $[i, \beta_1, \beta_2,\ D_1, D_2, k]$ to be simulated.

We now describe how to define g to perform this simulation. The definition for points on the N and S axes is fixed by Definition 2.16; it rests to define the function on the E and W axes.

- For a point $(mx + rp + i, E)$, we define g according to the intended effect on R_1 (represented by D_1 being $+1$ or -1):

$$g(mx + rp + i, E) \quad = \begin{cases} (mx + rp + i, N) & D_1 = (-1) \\ (4(mx + rp) + i, N) & D_1 = (+1) \end{cases} \tag{5}$$

- For a point $(mx + rp + i, W)$, we define g according to the intended effect on R_2 (represented by D_2 being $+1$ or -1), as well as the new internal state, k:

$$g(mx + rp + i, W) \quad = \begin{cases} (mx + rp + k, S) & D_2 = (-1) \\ (9(mx + rp) + k, S) & D_2 = (+1)\,. \end{cases} \tag{6}$$

Some explanations: starting with $\widehat{s} = (2^{r_1}3^{r_2}p + i, E)$, g produces a point on the N axis, specifically $(2^{r_1+1+D_1}3^{r_2}p+i, N)$. This point is mapped by g to $(2^{r_1+D_1}3^{r_2}p+i, W)$, so the exponent of 2 has been incremented or decremented, as desired. Moving to the S direction, we get $(2^{r_1+D_1}3^{r_2+1+D_2}p + k, S)$, and finally we reach $(2^{r_1+D_1}3^{r_2+D_2}p + k, E)$, completing the simulation of the transition.

We conclude that, if M is mortal, every trajectory starting at a proper state will arrive at a *halting configuration*, which by Lemma 2.15 is either $(0, \langle 0, 0\rangle)$, $(0, \langle 0, 1\rangle)$ or $(0, \langle 1, 0\rangle)$. Its encoding as a compass point is $(rp + 0, E)$ with $r \in \{1, 2, 3\}$, so the CCP is satisfied.

Since g has to be total, we have to define g for points where the "control state" is 0 as well; as we shall see below, the following definitions are useful

$$g(mx + rp + 0, E) = (mx + rp + 0, N), \tag{7}$$

and

$$g(mx + rp + 0, W) = (mx + rp + 0, S) \tag{8}$$

(for the North and South axes the definition follows (4)).

Now, we consider also improper states. Consider any point (y, E) with $y > m$. Write y as $2^{r_1}3^{r_2}dp + i$ where $d > 0$ is indivisible by 2 and 3, and $0 \le i < p$. Call (y, E) a d-state (so proper states are 1-states). Suppose that (y, E) is improper. If M is mortal, a trajectory similar to the one beginning with $2^{r_1}3^{r_2}p + i$ will be followed, since the presence of d affects neither the remainder modulo p, nor divisibility by 2 or

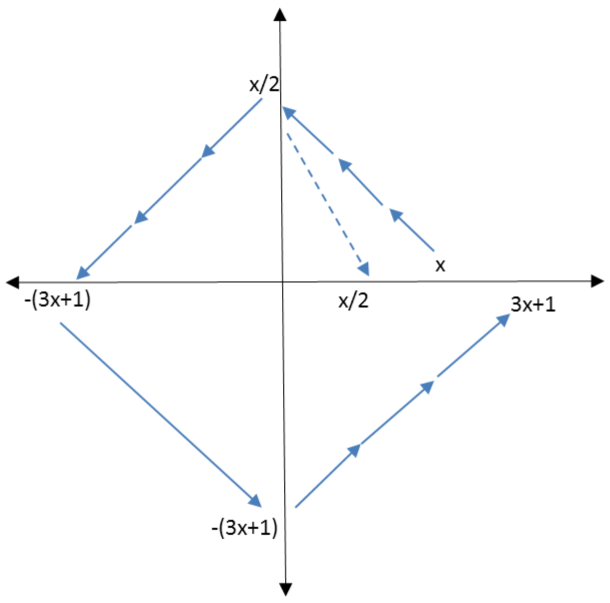


Figure 3: Simulating the $3x + 1$ function by a 2-dimensional monic PAF.

3. This trajectory (call it a d-trajectory) will end at a point of the form $(2^{e_1}3^{e_2}dp, E)$ with $e_1 + e_2 \le 1$. Thanks to definitions (7) and (8), it will progress as follows:

$$(2^{e_1}3^{e_2}dp, E) = (mx + rp + 0, E) \mapsto (mx + rp + 0, N) \mapsto (\frac{1}{2}mx + \lfloor\frac{1}{2}r\rfloor p, W)$$
$$\mapsto (\frac{1}{2}mx + \lfloor\frac{1}{2}r\rfloor p, S) \mapsto (\frac{1}{6}mx + \lfloor\frac{1}{6}r\rfloor p, E) = (\frac{1}{6}mx, E) .$$

Observe that $\frac{1}{6}mx \le \frac{1}{6}(mx + rp) < dp$. Hence, we obtained a state (y, E) with $y < dp$. Note also that y is divisible by p. Express y as $2^a 3^b d'p$; then $d' < d$. Thus, a trajectory of d-states (that does not reach the final zone) eventually "jumps" to a d'-state, with $d' < d$. Clearly this process must be finite.

If M is *not mortal*, there will be an infinite computation, starting with some configuration s with $\widehat{s} = 2^{r_1}3^{r_2}p + i$. Now, let us choose (for a reason explained below) an improper representation of s, specifically $2^{r_1}3^{r_2}dp + i$, with d some prime greater than 6. The infinite computation of M is simulated by a d-trajectory of g, staying within d-states (since the simulation does not halt, the situation where a truncated division breaks divisibility by d will not arise). Note that every d-state has the form $ydp + i$ and that $ydp > m$. So the trajectory will never reach the final zone. The reason for choosing a d-state is to ensure this; a trajectory of proper states can "accidentally" reach the final zone.

We conclude that our construction produces a CCP instance that is positive if and only if M is mortal. $\square$

role of region	constraints	$f(x,y)$
Div by 2	$x \geq 2$, $y \geq 0$	$(x-2, y+1)$
$x \bmod 2 = 0$	$x = 0$, $y \geq 0$	(y, x)
$x \bmod 2 = 1$	$x = 1$, $y \geq 1$	$(x-5, y)$
Compute $3x+1$	$x < 0$, $y \geq 1$	$(x-6, y-1)$
West to South	$x < 0$, $y \leq 0$	$(x+1, y-1)$
South to East	$x \geq 0$, $y < 0$	$(x+1, y+1)$
Final zone	$0 \leq x \leq 1$, $y = 0$	(x, y)

Table 3: A monic PAF simulating the $3x+1$ function.

2.4.3 Reducing CCP to mortality of monic PAFs

The main theorem of this section is proved by reducing the CCP to the mortality problem for monic PAFs. As the details of this construction are somewhat involved, it may be useful to first look at a simple example, where the classic $3x+1$ problem is represented by a monic 2-dimensional PAF, whose iteration reaches $(1,0)$ from initial point $(x,0)$ if and only if the Collatz sequence from x reaches 1.

Recall that in the $3x+1$ problem there are just two possible "updates," division by 2 and the mapping $x \mapsto 3x+1$, which are selected according to $x \bmod 2$. This makes it simpler than the Compass problem, where there are four "updates" and a large modulus. However, we still make use of trajectories that orbit around the origin, starting with the Cartesian point $(x,0)$ (Figure 3). The NE quadrant carries $(x,0)$ to $(\lfloor x/2 \rfloor, x \bmod 2)$; if the division was even, the point is mapped to $(x/2, 0)$, ready for the next iteration; otherwise, proceeding counter-clockwise, this point is carried first to $(-(3x+1), 0)$, then to $(0, -(3x+1))$ and finally to $(3x+1, 0)$.

For the complete definition of the PAF, see Table 3.

Next, we handle the CCP in all generality.

LEMMA 2.19. *A CCP can be effectively reduced to mortality of a monic PAF on $\mathbb{Z}^2$ (alternatively, to global convergence or global convergence to a fixed point of such a function).*

Proof. Given a description of a Compass Collatz function g, our reduction produces the function f defined by Table 4, where for convenience every region has a label. Explanations follow; see also Figure 4. Note that a row may be a concise presentation of several regions, distinguished by indices (e.g., $W^{\text{out}}_{r,i}$). The table also deviates from the definition of the class of monic functions by including values that have a constant component in certain regions; e.g., in region S^{in}, $f(x,y) = (0,y)$. This can be corrected by decomposing the region into smaller ones. In the example of S^{in}, we define one region

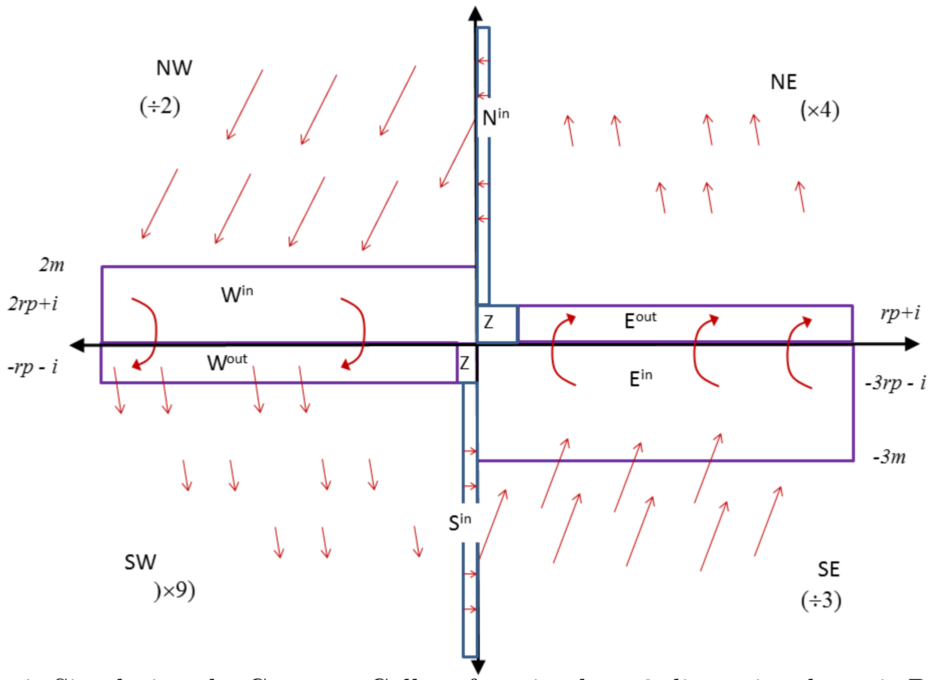


Figure 4: Simulating the Compass Collatz function by a 2-dimensional monic PAF.

for every x value (of which there are $p-1$); then, for the region with $x = c$, we define $f(x, y) = (x - c, y)$.

To see how this PAF simulates g, we first describe the mapping of compass points (x, Δ) to points in $\mathbb{Z}^2$ (Cartesian points):

1. A compass point $(mx + z, N)$, for $x > 0$, $z < m$, is represented by a Cartesian point on the positive y axis, specifically $(0, mx + z)$.

2. A point $(mx + z, S)$ is represented by a point on the negative y axis, specifically $(0, -mx - z)$.

3. A point $(mx + z, E)$ is represented by a point close to the positive x axis, specifically, $(mx + z, z)$. Note that the y coordinate is the remainder modulo m.

4. A point $(mx+z, W)$ is represented by a point close to the negative x axis, specifically, $(-mx - z, -z)$.

Next, we explain how dynamics of f simulate g. The handling of the West/South axes is almost symmetric to that of the East/North axes so we only describe the latter.

1. A compass point $(mx + z, E)$ is represented by $(mx + z, z)$. According to the definition of g, its action on this point depends on z:

   If $z = rp + i \in R_N$, $(mx + z, z)$ falls under the first definition of $E^{\text{out}}_{r,i}$, and is mapped immediately to $(0, mx + z)$, which is the representation of compass point $(mx + z, N)$.

region label	constraints	$f(x,y)$
NW	$x\leq 0,\ y\geq 2m$	$(x-m,\ y-2m)$
$W^{\text{in}}_{r,i}$ $_{0\leq r<12,\ 0\leq i<p}$	$x\leq 0,\ y=rp+i>0$	$(x-\lfloor r/2\rfloor p-i,\ -\lfloor r/2\rfloor p-i)$
$W^{\text{out}}_{r,i}$ $_{rp+i\in R_S}$	$x\leq -m/2,\ y=-rp-i$	$(0,\ x+i-w_{rp+i})$
$W^{\text{out}}_{r,i}$ $_{rp+i\notin R_S}$	$x\leq -m/2,\ y=-rp-i$	$(x+m,\ y-9m+rp+i-w_{rp+i})$
SW	$x\leq -p,\ y\leq -m$	$(x+p,\ y-9p)$
S^{in}	$-p<x<0,\ y\leq -m$	$(0,\ y)$
SE	$x\geq 0,\ y\leq -3m$	$(x+m,\ y+3m)$
$E^{\text{in}}_{r,i}$ $_{0\leq r<18,\ 0\leq i<p}$	$x\geq 0,\ y=-rp-i<0$	$(x+\lfloor r/3\rfloor p+i,\ \lfloor r/3\rfloor p+i)$
$E^{\text{out}}_{r,i}$ $_{rp+i\in R_N}$	$x\geq m,\ y=rp+i$	$(0,\ x)$
$E^{\text{out}}_{r,i}$ $_{rp+i\notin R_N}$	$x\geq m,\ y=rp+i$	$(x-m,\ y+4m-rp)$
NE	$x\geq p,\ y\geq m$	$(x-p,\ y+4p)$
N^{in}	$0<x<p,\ y\geq m$	$(0,\ y)$
Z	elsewhere	$(0,0)$

Table 4: Simulating the Compass Collatz function by a 2-dimensional monic PAF.

If $z = rp + i \notin R_N$, $(mx+z, z)$ falls under the second definition of $E^{\text{out}}_{r,i}$, and is mapped to $(m(x-1)+z,\ 4m+i)$. This will (in the typical case) fall into the NE zone, where, repeatedly, p is subtracted from the first coordinate while $4p$ is added to the second, until finally we get $(i, 4(mx+rp)+i)$. This is now in the N^{in} zone, and is mapped to $(0, 4(mx+rp)+i)$, the representation of compass point $(4(mx+rp)+i, N)$.

2. A compass point $c_N = (mx+rp+i, N)$ is represented by the Euclidean point $\vec{x}_N = (0, mx+rp+i)$. According to Definition 2.16, $g(c_N) = (\frac{1}{2}mx + \lfloor\frac{1}{2}r\rfloor p + i, W)$. If we write $g(c_N)$ as $(mx'+z', W)$, then $g(c_N)$ is represented by $(-mx'-z', -z')$.

   The action of f on $\vec{x}_N$ is as follows. According to the action at region NW, m is subtracted from the first coordinate while $2m$ is subtracted from the second, until one of the W^{in} regions is reached; it is not hard to see that the point reached is $(-mx', m(x-2x')+rp+i)$, where $0 \le x-2x' < 2$, so that $x' = \lfloor x/2 \rfloor$. Now, the specific region this point falls in is $W^{\text{in}}_{r',i}$ where $r'p = m(x-2x')+rp$ (with a single exception, when $r' = i = 0$, see below). The point is thus mapped into

$$(-mx' - \lfloor r'/2 \rfloor p - i,\ -\lfloor r'/2 \rfloor p - i)\,.$$

   Now, $-mx' - \lfloor r'/2 \rfloor p = -(\frac{1}{2}mx + \lfloor\frac{1}{2}r\rfloor p)$, so the point obtained has as x coordinate $-(\frac{1}{2}mx + \lfloor\frac{1}{2}r\rfloor p) - i = -mx'-z'$, and as y coordinate it has $-\lfloor r'/2 \rfloor p - i$, which is exactly $-z'$. We conclude that g is correctly simulated. Note that the last point is in $W^{\text{out}}_{\lfloor r'/2 \rfloor, i}$.

   In the exceptional case $r' = i = 0$, the point $(-mx', m(x-2x')+rp+i) = (-mx', 0)$ is already in $W^{\text{out}}_{\lfloor r'/2 \rfloor, i}$, that is, W^{in} is skipped.

A few other points are in order:

- Some points (z, Δ) in the vicinity of the origin are mapped immediately to $(0,0)$, specifically, $\{(z,E) \mid z < m\}$, (p, N), $\{(z,W) \mid z < m/2\}$ and $\{(y,S) \mid y < 3m\}$. This agrees with the correctness of the reduction for the following reasons:
  - Points (z, E) with $z < m$ are the stopping condition for the CCP.
  - Point (p, N) cannot be $g(z, E)$ for any $z \ge m$ (this can be seen by inspecting Definition 2.16). Thus, it cannot participate in an immortal trajectory, except possibly as its initial point. But then there would be an immortal trajectory without this point, too.
  - A point (z, W) with $z < m/2$ cannot be $g(y, N)$ for any $y \ge m$ (this can be seen by inspecting Definition 2.16). Thus, they cannot participate in an immortal trajectory, except possibly as its initial point. But then there would be an immortal trajectory without this point, too.
  - Any point (y, S) with $y < 3m$ is mapped by g to the final zone.

role of region	constraints	$f(x,y)$
Div by 2	$x>1$, $y \geq 0$	$(x-2, y+1)$
$x \bmod 2 = 0$	$x=1$, $y \geq 0$	$(y, x-1)$
Compute $3x+1$	$x<0$, $y \geq 0$	$(x-6, y-1)$
West to South	$x \leq (-2)$, $y < 0$	$(x+1, y-1)$
South to East	$x > (-2)$, $y < 0$	$(x+1, y+1)$

Table 5: A monic PAF simulating the $3x+1$ function with five regions.

- There are also Cartesian points that do not represent any Compass point according to the correspondence set above. For example, points (x, y) where $x, y > 0$ and $x \not\equiv y \pmod{m}$; and similar points in the other quadrants. It is not hard, however, to verify that such points are mapped, after a finite number of steps, to some point on the y axis; the latter kind are all "proper," that is, they represent a Compass point, hence the trajectory they subsequently follow represents the dynamics of g.
- The function has no fixed point except the origin.

This concludes the proof of Lemma 2.19 . □

Combining the lemmas, we obtain

THEOREM 2.20. *Global convergence, mortality as well as global convergence to a fixed point, all of monic piecewise-affine functions over $\mathbb{Z}^2$ are Π_2^0-complete problems.*

Bounding the number of regions. In view of Section 2.2, it is natural to ask whether the hardness of the above problems survives some (large enough) constant bound on the number of regions. Is there a constant bound for which they are undecidable? Π_2^0 hard?

The method of Section 2.2 does not seem to apply to this class of functions. But the fact that the $3x+1$ problem is representable with seven regions may be interpreted as evidence for the hardness of this problem, even with a constrained number of regions: it means that a decision procedure that works for seven regions only would tell us the answer to the Collatz problem. This even holds for five regions, since it is possible to optimise the representation of the problem to a 5-region function, shown in Table 5. This encoding is based on representing an element x of the sequence as $(0, x-1)$ rather than $(0, x)$.

We also note that in three dimensions, Π_2^0 hardness with a constant number of regions follows relatively easily (details are deferred to Section 2.4.4). In one dimension the problem is decidable (Section 3). It is, therefore, all the more intriguing that the situation in two dimensions seems so hard to settle.

2.4.4 Monic PAFs in three dimensions

We now consider functions over $\mathbb{Z}^3$ (or $\mathbb{N}^3$), and show a Π^0_2-hardness result with a bounded number of regions. It turns out that we can actually restrict the class of functions: the natural adaptation of the definition of monic PAFs would allow for permuting the variables, as in

$$f(\vec{x}) = (x_{i_1} + b_1, x_{i_2} + b_2, x_{i_3} + b_3)$$

where $\{i_1, i_2, i_3\} = \{1, 2, 3\}$. But we will only use functions of the form

$$f(\vec{x}) = (x_1 + b_1, x_2 + b_2, x_3 + b_3)\,. \tag{9}$$

The proof idea is to simulate an enhanced two-counter CM, using the three coordinates to simulate the two counters and the program counter. Hardness with a bounded number of regions follows from an adaptation of Proposition 2.7.

PROPOSITION 2.21. *From an (ordinary) counter machine M and a universal machine U, one can compute an enhanced two-counter machine U^2_M such that U^2_M is mortal if and only if M halts on every initial state $(1, \langle x, 0, \ldots, 0\rangle)$. The number of registers and control states of U^2_M are independent of M.*

Proof. U^2_M is obtained by converting the machine U_M, presented in the proof of Proposition 2.7, to a two-counter machine. This is done using the standard encoding of several registers in one, going back to Minsky [46]. We recall it briefly. Suppose that U_M has $n+1$ registers (for consistency with its construction in Section 2.2). Their contents are encoded as the exponents of prime numbers $p_1, \ldots, p_{n+1}$, so that state $(k, \langle x_1, x_2, \ldots, x_{n+1}\rangle)$ of U_M is represented by a state $(k, p_1^{x_1} \cdots p_{n+1}^{x_{n+1}}, 0)$ of the two-counter machine. Simulation of a step of U_M involves the operations of: testing if the contents of the first register is divisible by a certain constant, dividing or multiplying it by a constant; all can be implemented by loops, using the second register for temporary storage. Importantly, multiplication by c can be effected by a loop that uses the ability of an enhanced CM to add c at a time; thus the number of control states does not depend on the magnitude of constants in enhanced instructions of U_M. It only depends on the number of registers and control states of U_M, known to be independent of M. Finally, we recall from [9] that this translation to two-counter machines preserves mortality. $\square$

THEOREM 2.22. *There is a constant t such that mortality of piecewise-affine functions of the form* (9)*, whose definition consists of at most t regions, is Π^0_2-complete.*

Proof. We reduce from mortality of counter machines by first applying the transformation of Proposition 2.21, and then simulating the two-counter machine by a 3-dimensional PAF (operating on a vector (x, y, z)). This is simple: one dimension is used for the program counter and two others for the counters. Every enhanced CM instruction, $[i, \beta_1, \beta_2, D_1, D_2, k]$, corresponds to a rectangular region (intersecting $\{x = i\}$ with $\{y = 0\}$ or $\{y > 0\}$, according to β_1, and with $\{z = 0\}$ or $\{z > 0\}$, according to β_2).

	Class of affine functions	region boundaries	$\mathbb{Z}^2$	$\mathbb{Z}^2$, partition size bounded	$\mathbb{Z}^3$, partition size bounded
1	$f(\vec{x}) = (x_{i_1} + b_1, x_{i_2} + b_2)$, where $i_1 \neq i_2$	$\pm x_i \leq d$ $d \in \mathbb{Z}$	Π^0_2	Π^0_1?	Π^0_2
2	$f(\vec{x}) = (a_1 x_{i_1} + b_1, a_2 x_{i_2} + b_2)$, where $i_1 \neq i_2$ and $a_{1,2} \in \{\pm 1\}$	$\pm x_i \leq d$ $d \in \mathbb{Z}$	Π^0_2	Π^0_2?	Π^0_2
3	$f(\vec{x}) = (x_{i_1} + b_1, x_{i_2} + b_2)$, where $i_1 \neq i_2$	$cx_i \leq d$ $c, d \in \mathbb{Z}$	Π^0_2	Π^0_2?	Π^0_2
4	$f(\vec{x}) = (x_{i_1} + b_1, x_{i_2} + b_2)$, where $\{i_1, i_2\} \subseteq \{1, 2\}$	$\pm x_i \leq d$ $d \in \mathbb{Z}$	Π^0_2	Π^0_2?	Π^0_2
5	$f(\vec{x}) = (x_1 + b_1, x_2 + b_2)$	$\pm x_i \leq d$	decidable?	decidable?	Π^0_2-complete

Table 6: Some simple function classes and results known or conjectured about the decidability of Mortality (the conjectures are marked with "?"). The first row refers to our class of monic PAFs. When referring to bounded partition size, the expression "for a bound large enough" should be tacitly understood. When a class Π^0_i appears, the problem is (or is conjectured to be) complete for that class.

Its effect is clearly expressed by the monic function $f(x, y, z) = (x+k-i, y+D_1, z+D_2)$. For all the points with x greater than the highest instruction, we define the function so that it reduces x to 0, and for $x = 0$ (which signifies halting) we define it so it reduces y and z to zero. Thus mortality of the PAF is equivalent to mortality of the CM. $\square$

2.4.5 Some function classes similar to monic PAFs

As previously stated, the choice of studying specifically the monic PAF class is arbitrary (though not random). It turned out to be an interesting class, as some results could be achieved (but required a non-trivial proof), and some open problems remain. Some similar classes are suggested in the following paragraphs, and summarised in Table 6, along with results about them and also some conjectures (the table also includes the monic PAF class studied above, to give the complete picture).

Larger classes of functions. Classes 2–4 in Table 6 are larger than our monic PAFs. Hence, results proved for our class transfer easily. But it is conjectured that stronger results can be proved. Note that the first extends the range of coefficients of the affine parts; the second removes the restriction on the coefficients in the region definitions; and the last one allows to duplicate a value, that is, to map (x, y) to $(x + a, x + b)$ (this class and the conjecture about it have been proposed by a referee).

A smaller class of functions. The last class in the table includes functions which, in each region, take the form $f(\vec{x}) = (x_1 + b_1, x_2 + b_2)$. Such functions resemble the class studied by Asarin, Maler and Pnueli [1] in a continuous setting. They showed that undecidability of the *reachability* problem begins at three dimensions. Over the integers, undecidability of reachability as well as Π_2^0-completeness of mortality at three dimensions follow easily from simulation of two-counter machines (as shown in Section 2.4.4). Over $\mathbb{Z}^2$, decidability has not yet been established.

Functions over $\mathbb{N}^n$. Unlike the reduction used to prove Theorem 2.4, the construction using monic functions makes essential use of negative numbers. While in three dimensions, Π_2^0 hardness follows from Theorem 2.22 (one just has to review its proof to see that only natural numbers are necessary), we leave the situation in two dimensions as an open problem.

2.4.6 An application to functions over the rationals

Recall that [9] proved that global convergence and mortality of piecewise-affine functions over $\mathbb{Q}^2$ are undecidable, a result strengthened by Corolloary 2.10 to Π_2^0-completeness for mortality (for global convergence we can only deduce Π_2^0-hardness, since the problem is not clearly in Π_2^0). Next, we show that the result on monic functions can also be adapted to the rationals. We need to slightly extend the class of functions, as follows.

Definition 2.23. A *nearly-monic* piecewise affine function (PAF) on $\mathbb{Q}^2$ is a piecewise affine function where each of the defining affine components has either the form $f(\vec{x}) = (x_{i_1} + b_1, x_{i_2} + b_2)$, where $\{i_1, i_2\} = \{1, 2\}$ and $b_1, b_2 \in \mathbb{Z}$, or $f(\vec{x}) = (0, 0)$. In addition, the halfspaces which define the space partition are given by inequalities of the form $\pm x_i \le d$.

Thus, the extension to the class of monic functions is the availability of the option $f(\vec{x}) = (0, 0)$. This extension is important: without it, such a function cannot be mortal, since one cannot map non-integral values to integers.

THEOREM 2.24. *Global convergence, mortality and global convergence to a fixed-point of nearly-monic piecewise-affine functions over $\mathbb{Q}^2$ are Π_2^0-complete problems.*

Proof. We claim that the behaviour of the function f constructed in Lemma 2.19, when converted to a nearly-monic function on $\mathbb{Q}^2$, is the same as for the integers. The basic idea is quite simple: we make sure that the definition of the regions is such that a rational point (x, y) falls into the same region as $(\lfloor x \rfloor, \lfloor y \rfloor)$. Note that the function $f(\vec{x}) = (x_{i_1} + b_1, x_{i_2} + b_2)$ leaves the fractional parts fixed. So the trajectory from (x, y) is an integer-valued trajectory from $(\lfloor x \rfloor, \lfloor y \rfloor)$, shifted by a fixed fractional part. Table 7 gives the details. The changes in comparison to Table 4 are two: first, the definition of regions has been corrected, where necessary, to cover the adjacent unit squares; for example, in the definition of $W_{r,i}^{\text{in}}$, instead of $y = -rp - i$, we now have $-rp - i \le y < -rp - i + 1$. Secondly, cases where, in the integer case, a component of the value of a function was written as a constant, have been rewritten to conform to the monic function class; for example, whereas in Table 4 we find that in region S^{in}, $f(x, y) = (0, y)$, Table 7 breaks S^{in} into sets S_i^{in} with $f(x, y) = (x + i, y)$. This is not really a change (i.e., this is how the function was supposed to be also in the integer case), but by making it explicit it becomes easier to verify that nothing breaks when adapting to the continuous case.

□

3 Decidability and Complexity in One Dimension

Blondel et al. [9] prove that global convergence is decidable in polynomial time for $n = 1$ when the function is continuous. The result can be easily extended to mortality, as shown in the next subsection. Our main result concerns the integers, where continuity is irrelevant. We show decidability and classify the problem's complexity as PSPACE-complete[6].

3.1 The continuous case

We extend the solution of Blondel et al. for global convergence, to show that mortality can also be decided in polynomial time for continuous piecewise-affine functions with rational coefficients.

The decision procedure follows form the following lemma.

LEMMA 3.1. [9] *For a continuous piecewise affine function $f : \mathbb{R} \to \mathbb{R}$ (with rational coefficients), f is globally convergent if and only if the following hold:*
$(*)$ *For every $x > 0$, $f(x) < x$ and $f^{(2)}(x) < x$, and for every $x < 0$, $f(x) > x$ and $f^{(2)}(x) > x$.*

We prove

LEMMA 3.2. *For a continuous piecewise affine function $f : \mathbb{R} \to \mathbb{R}$ (with rational coefficients), f is mortal if and only if the conditions $(*)$ above hold, and in addition, there is some $\varepsilon > 0$ such that either $f(x) = 0$ on $0 \le x \le \varepsilon$, and $f(x) \ge 0$ on $-\varepsilon \le x \le 0$, or $f(x) = 0$ on $-\varepsilon \le x \le 0$, and $f(x) \le 0$ on $0 \le x \le \varepsilon$.*

Proof. Assume that f is mortal. Then it is globally convergent. It must have $f(0) = 0$. By Lemma 3.1, for any $x > 0$, $f(x) < x$. We can consider ε small enough so that on $[0, \varepsilon]$ our function is linear, and similarly on $[-\varepsilon, 0]$. Suppose that in $[0, \varepsilon]$ we have $f(x) > 0$; then starting from $0 < x_0 < \varepsilon$, we get an infinite trajectory, limiting at zero but not reaching it. So we must have $f(x) \le 0$. We similarly show that for $-\varepsilon \le x \le 0$ we require $f(x) \ge 0$.

In addition, if $f(x) < 0$ for small positive x, and $f(x) > 0$ for small negative x, again we have an infinite trajectory. Thus one can conclude that at least one of these conditions must not hold; i.e., $f(x) = 0$ either on $[0, \varepsilon]$ or on $[-\varepsilon, 0]$. □

For the problem convergence to a fixed point, its complexity in the 1-dimensional, continuous case remains open.

[6]Recent work by Finkel, Göller and Haase [27] includes a PSPACE algorithm for a more general problem: machines with a single register subject to polynomial updates.

region label	constraints	$f(x,y)$
NW	$x < 1,\, y \geq 2m$	$(x - m,\ y - 2m)$
$W^{\text{in}}_{r,i}$ $_{0\leq r<12,\ 0\leq i<p}$	$x < 1,$ $0 < rp + i \leq y < rp + i + 1$	$(x - \lfloor r/2 \rfloor p - i,\ y - \lfloor \frac{3}{2} r \rfloor p - 2i)$
$W^{\text{out}}_{r,i}$ $_{rp+i \in R_S}$	$x \leq -m/2,$ $-rp - i \leq y < -rp - i + 1$	$(0,\ x + i - w_{rp+i})$
$W^{\text{out}}_{r,i}$ $_{rp+i \notin R_S}$	$x \leq -m/2,$ $-rp - i \leq y < -rp - i + 1$	$(x + m,\ y - 9m + rp + i - w_{rp+i})$
SW	$x < -p + 1,\, y < -m + 1$	$(x + p,\ y - 9p)$
S^{in}_{i} $_{0<i<p}$	$-i \leq x < -i + 1,\, y < -m + 1$	$(x + i,\ y)$
SE	$x \geq 0,\, y < -3m + 1$	$(x + m,\ y + 3m)$
$E^{\text{in}}_{r,i}$ $_{0\leq r<18,\ 0\leq i<p}$	$x \geq 0,$ $-rp - i \leq y < -rp - i + 1 \leq 0$	$(x + \lfloor r/3 \rfloor p + i,\ y + \lfloor \frac{4}{3} r \rfloor p + 2i)$
$E^{\text{out}}_{r,i}$ $_{rp+i \in R_N}$	$x \geq m,$ $rp + i \leq y < rp + i + 1$	$(y - rp - i,\ x)$
$E^{\text{out}}_{r,i}$ $_{rp+i \notin R_N}$	$x \geq m,$ $rp + i \leq y < rp + i + 1$	$(x - m,\ y + 4m - rp)$
NE	$x \geq p,\, y \geq m$	$(x - p,\ y + 4p)$
N^{in}_{i} $_{0<i<p}$	$i \leq x < i + 1,\, y \geq m$	$(x - i,\ y)$
Z	elsewhere	$(0, 0)$

Table 7: The definition of the monic PAF, rewritten for the rationals.

3.2 Understanding the integer problem

The following examples illustrate the difference between mortality over the integers and over the reals. The first function has a fixed point at 10/3 and therefore is not mortal over the reals (or rationals); but it is over the integers, and we explain later how this is ascertained by our algorithm. Similarly, the second function has a fixed point at 4/3, which breaks its mortality if the reals or rationals are considered.

$$f_1(x) = \begin{cases} -2x+10 & x > 2 \\ 4 & x < 0 \\ 0 & x = 0 \\ 4-2x & 1 \leq x \leq 2 \end{cases} \tag{10}$$

$$f_2(x) = \begin{cases} x-3 & x > 2 \\ 4 & x < 0 \\ 0 & x = 0 \\ 4-2x & 1 \leq x \leq 2 \end{cases} \tag{11}$$

Note that in dimension one, the regions are just a partition of $\mathbb{Z}$ into a finite number of intervals. We may assume that these are given explicitly as the list of the end points of closed intervals, e.g.,

$$(-\infty, -3], [-2, 0], [1, +\infty)$$

plus the coefficients of the affine function associated with each interval. This list forms the *standard description*, in terms of which we make considerations of complexity. There will always be one interval infinite to the left, which we denote by $(-\infty, L]$, and one infinite to the right, denoted by $[R, +\infty)$; and by breaking intervals into parts if necessary we can ensure $L < 0$ and $R > 0$. We denote the function in the negative infinite interval by $a^- x + b^-$ and the function in the positive infinite interval by $a^+ x + b^+$.

3.3 A decision algorithm

The input to our decision algorithm is a standard description $\langle f \rangle$ of the subject function f. We define

$$\rho = \max(\{R\} \cup \{f(x), \text{ where } L \leq x \leq R\})$$
$$\lambda = \min(\{L\} \cup \{f(x), \text{ where } L \leq x \leq R\})$$

Note that these values can be easily calculated in polynomial time, since for each finite interval, $f(x)$ assumes the maximum (or minimum) at one of its ends. These values show how far away from 0 one can get without using the infinite regions.

Next, we show a simple (polynomial-time) algorithm that either determines that the dynamical system is *divergent*, which means that there is an unbounded trajectory; or returns a finite interval A such that all trajectories visit it infinitely many times.

LEMMA 3.3. *Suppose that at least one of a^+, a^- is non-negative. If either a^+ or a^- is bigger than 1; or $a^+ = 1$ and $b^+ \geq 0$; or $a^- = 1$ and $b^- \leq 0$, then f is divergent, hence* *not mortal. Otherwise, it holds that all trajectories visit the set $A = [\lambda, \rho]$ infinitely often.*

Proof. If either a^- or a^+ is bigger than 1, it is easy to see that for x of sufficiently large absolute value, iterating f from initial point x diverges. Similarly, if $a^- = 1$ and $b^- \leq 0$, or $a^+ = 1$ and $b^+ \geq 0$. Hence, let us assume that none of these cases holds. We then have to prove that all trajectories visit A infinitely often, which boils down to proving

$$\forall x_0 \in \mathbb{Z} \ . \ \exists \ t > 0 \ . \ x_t = f^{(t)}(x_0) \in A \, . \tag{12}$$

There are a few sub-cases to consider, depending on the values of a^+, a^-, b^+, b^-. For symmetry, it suffices to consider $x_0 > 0$. By the definition of A, (12) clearly holds for $x_0 \in [L, R]$. So we consider $x_0 > R$, and perform a case analysis on a^+.

1. Suppose that $a^+ = 1$ and $b^+ < 0$ (the case with our example f_2). Then, as long as $x_i > R$, we have $x_{i+1} = x_i + b^+ < x_i$. Hence, the trajectory is a descending sequence which must reach a point in $(R + b^+, R]$. Note that by definition, $\lambda \leq f(R) = R + b^+$, so we see that the trajectory must enter A.
2. If $a^+ = 0$, then the function is $f(x) = b^+$ on $[R, +\infty)$, so $x_1 = b^+ = f(R) \in A$.
3. This leaves the case $a^+ < 0$. Now $f(x_0) = a^+ x_0 + b^+$. If $f(x_0) \in A$, we are done. Suppose it is not. That is, either $f(x_0) > \rho$ or $f(x_0) < \lambda$. In the first case, this means $a^+ x_0 + b^+ > \rho \geq f(R) = a^+ R + b^+$, implying $x_0 < R$, a contradiction. In the second case, $f(x_0) < L$, so we consider the definition of f in this region. Since we are not in the divergent case, there are, again, two sub-cases.
   (a) $a^- = 1$ and $b^- > 0$. Thus $f(y) > y$ for $y \leq L$ and the trajectory proceeds as an ascending sequence which must reach a point in $[L, L + b^-)$, hence in A.
   (b) $a^- = 0$. In this case, $f(f(x_0)) = b^- = f(L) \in A$.

□

Example 3.1. Consider f_1 from (10). By its definition for $x < 0$, we see that $a^- = 0$. By its definition for $x > 2$, we see that $a^+ = (-2)$. Hence, none of the cases that immediately imply divergence apply. Note that $[L, R] = [-1, 3]$. We then compute

$$\lambda = \min(\{-1\} \cup \{f(x), \text{ where } -1 \leq x \leq 3\}) = -1;$$
$$\rho = \max(\{3\} \cup \{f(x), \text{ where } -1 \leq x \leq 3\}) = 4 \, .$$

So we test the trajectory from each point in $[-1, 4]$; this interval is actually covered by the mortal trajectories shown next

$$\begin{tikzcd}
 & 1 \arrow[dr] & & \\
-1 \arrow[r] & 4 \arrow[r] & 2 \arrow[r] & 0 \\
3 \arrow[ur] & & &
\end{tikzcd}$$

```
for x := $\lambda, \ldots, -1, 1, \ldots, \rho$ do {
        a := x;
        b := $f$(x);
        while a $\neq$ b do {
                if b = 0    begin next iteration of the for loop
                b := $f$(b);
                if b = 0    begin next iteration of the for loop
                b := $f$(b);
                a := $f$(a);
        }
        return "immortal trajectory starting at" x
}
return $f$ "is mortal"
```

Figure 5: Algorithm to test for immortal trajectory starting in the set A.

We conclude that the function is mortal.

THEOREM 3.4. *Global convergence, mortality and convergence to a fixed point, all for a piecewise affine function $f : \mathbb{Z} \to \mathbb{Z}$ with integer coefficients, are PSPACE problems.*

As we later prove, they are actually PSPACE-complete.

Proof. The algorithm for analysing mortality is as follows. First, if both a^+ an a^- are negative, we construct (in polynomial time) a representation of $f \circ f$, whose asymptotic behaviour is the same, and has positive coefficients in the infinite regions; we thus assume that Lemma 3.3 is applicable. By testing the conditions stated in the lemma, we either conclude immediately that f is not mortal, or get the attractor-like set A. In the latter case, we now proceed to trace, for each point $x_0 \in A$, the trajectory from x_0, until either finding that it reaches zero, or that it cycles without meeting the origin—so we know if f is mortal. Note that this can be accomplished in polynomial space: a pseudo-code of the procedure appears in Figure 5. To verify it, note that on the ith entrance to the loop, we have

$$\mathtt{a} = f^{(i)}(\mathtt{x}), \quad \mathtt{b} = f^{(2i+1)}(\mathtt{x}), \quad \forall j < 2i+1 \,.\, f^{(j)}(\mathtt{x}) \neq 0, \quad \forall k < i \,.\, f^{(k)}(\mathtt{x}) \neq f^{(2i+1)}(\mathtt{x})\,.$$

It follows that if the sequence from $\mathtt{x}$ has an eventual period of p, the loop will stop. To see this, choose i so that i is big enough to enter the periodic part, and $i+1$ a multiple of p. Then $f^{(i)}(\mathtt{x}) = f^{(2i+1)}(\mathtt{x})$, i.e., $\mathtt{a} = \mathtt{b}$, and the loop guard is not satisfied.

A somewhat simpler (but slower to stop) algorithm is to compute each trajectory while counting how many times a point in A is visited; the count will exceed $|A|$ if and only if there is a cycle.

Finally we consider the other problems. For the convergence problem, the answer is negative if $f(0) \neq 0$, while if $f(0) = 0$ it is equivalent to mortality. For convergence to a fixed point, we can use essentially the same algorithm as for mortality, except for the following changes:

- First, comparing to Lemma 3.3, we note that the response when $a^+ = 1$ and $b^+ = 0$, or $a^- = 1$ and $b^- = 0$, changes: in these cases, the trajectories starting at the corresponding infinite region do reach a fixed point (immediately); so we should proceed to the main algorithm (exhaustively testing trajectories starting at A).
- In the exhaustive-search algorithm, instead of looking for the arrival at zero, we are looking for a point where $f(x) = x$.

□

3.4 PSPACE hardness

In this subsection we shall prove that global convergence and mortality over $\mathbb{Z}$ of a piecewise affine function f with integer coefficients are PSPACE-hard. To this end, we next introduce several Turing-machine variants as intermediate representations, and show a sequence of reductions, culminating in our decision problem.

3.4.1 Some abstract machines

We define the machine models used in the series of reductions. The first machine is a restricted form of *linearly bounded automaton* (LBA) [61].

Definition 3.5. A *tally LBA* is a single-tape machine that receives as input a string of the form 0^n for some $n \geq 0$; this string is initially written on its work tape, delimited by endmarkers. The machine is guaranteed to never move beyond the endmarkers. In our machines, the tape alphabet is always binary.

In the next definition, we consider the tally string to be part of the machine's description: hence, a given machine only performs a single computation.

Definition 3.6. A *fixed-space machine with oblivious queue access* is a Turing machine with the following features:

1. The work tape is a *queue* of a fixed capacity n (which we consider to be given, in tally form, as part of the standard description of such a machine). In every step the machine strips a symbol from the front of the queue and adds one to the rear. Hence, an instruction of the machine is given by a 4-tuple (q, b, q', b'), where:
   q is the current control state, q' the next;
   b the symbol read off the queue, b' the symbol appended to the queue.

   We use the customary symbol δ for the set of these 4-tuples.

2. The work-tape (queue) alphabet is binary.

3. The initial contents of the queue are 0^n.

$$\begin{tikzcd}
\text{Tally LBA halting} \arrow[r] & \text{FSOQM halting} \arrow[dr] & \\
& & \text{FSOQ\&CM mortality} \arrow[dl] \\
\text{1-dim PAF mortality} & \text{FSOQM mortality} \arrow[l] &
\end{tikzcd}$$

Figure 6: Reductions for proving PSPACE-hardness. Here, FSOQM stands for fixed-space queue machine (Definition 3.6) and FSOQ&CM stand for fixed-space queue machine with clock (Definition 3.7).

Definition 3.7. A *fixed-space machine with oblivious queue access and a clock*, briefly a fixed-space Q&C machine, is a Turing machine with the following features:

1. The work tape is a *queue* of a fixed capacity n (which we consider to be given, in tally form, as part of the standard description of such a machine). In every step the machine strips a symbol from the front of the queue and adds one to the rear. Hence, an instruction of the machine is given by a 4-tuple (q, b, q', b'), as in the last definition.

2. The work-tape (queue) alphabet is binary.

3. The machine also has a *clock*. This device is a counter (a register of non-negative integer value) that is automatically decremented each time the machine has completed another cycle through the queue (that is, exactly n transitions). If the clock reaches zero, the machine is reset: the queue is cleared, the control state is reset to an initial state (0) and the clock is reset to its initial value.

4. The initial contents of the queue are 0^n; the initial clock value is given—in binary—as part of the standard description of such a machine.

3.4.2 Simulations and hardness results

We now proceed to a sequence of reductions, that starting from the text-book result that halting of tally LBA is PSPACE-hard, culminates in our problem of interest. A road map is provided by Figure 6.

LEMMA 3.8. *The halting problem of tally LBAs is a PSPACE-hard problem.*

This is a folklore result, which can be proved directly from the definition of PSPACE-hardness with little effort.

LEMMA 3.9. *The computation of a tally LBA on input 0^n can be simulated by a fixed-space machine with oblivious queue access, whose queue capacity is $2n$.*

Proof. This is a rather simple construction with a little twist. Suppose (momentarily) that the queue capacity is $2n + 2$ bits. The tape is simulated by the queue, using an encoding of bits as pairs of bits, for instance $0 \mapsto 00,\ 1 \mapsto 01$. The additional 2 bits are set to 11 and used to simulate he endmarkers (one pair simulates both endmarkers because the queue is cyclic!). The machine marks the current head position on the tape by changing the first digit of the pair "under the head" from a 0 to a 1. Every transition of M is simulated by cycling through the queue. During each cycle, when the head position is encountered, the appropriate local action can be taken (but see the next paragraph).

The reader may wonder how a transition that moves a head *backwards* can be simulated as we cycle through the queue. The reader might also wonder why the queue capacity has been defined as $2n$ rather than $2n + 2$. Both questions are answered by revealing that we keep one symbol (the one to the left of the head) only in the finite control. In every "instruction cycle," the machine takes a pair of bits off the queue, which corresponds to the next symbol (say, the ith) on the simulated tape, while enqueuing the pair representing the $i - 1$st symbol. If a left-move is to be simulated, when the machine reads a 1-bit as the first bit of a pair (which signals that the head position has been reached), it will write a 1-bit, thus marking the *previous* encoded symbol as being scanned by the head. □

Mortality of fixed-space machines is defined as usual: halting when started with any possible configuration. Note that there are finitely many such configurations, due to the fixed space.

LEMMA 3.10. *The halting problem for fixed-space Turing machines with oblivious queue access can be reduced, in logarithmic space, to mortality of a fixed-space Q&C machine.*

Proof. Observe that if the given machine ever halts, the length of its computation must be bounded by $2^n \cdot m$, where n is the given queue size and m the number of control states (argument: this is the number of distinct configurations of the machine). We let this be the initial value of the clock, so that if the machine does halt, it will halt before the clock times out. If the machine does not halt, it will eventually time out, then restart, ad infinitum. To see that a halting machine is transformed into a mortal one, note that regardless of its initial configuration, the Q&C machine will either halt or time out. If timed out, it resests, and subsequently carries on a correct simulation the given input-free machine. □

LEMMA 3.11. *Mortality of fixed-space Turing machines with oblivious queue access is PSPACE-hard.*

Proof. We reduce from mortality of fixed-space Q&C machines. Given such a machine, our job is to eliminate the clock. We do it by the following steps.

1. Let the machine have m control states $q \in Q$. We create a set of mn control states, $Q \times [0, n - 1]$, and change every transition from state q to q' to a transition from

(q, i) to $(q', i + 1 \bmod n)$ for all $i < n$. The effect is that the second component of the control state indicates the position in the queue (it returns to zero each time that a cycle over the queue is completed).

2. We now change the queue size from n to $n+c$, where the additional c bits suffice to represent the clock. Accordingly, we increase the set of control states so that when transitioning from $(q, n-1)$, the machine first moves over the space dedicated to the clock, decrementing its value (this is quite easy to do by representing the binary value with its LSB in front), and also checking whether zero has been reached. According to the result, either $(q', 0)$ will be entered (completing a simulation of the original transition), or a special set of states will be entered which resets the machine.

□

Next, we reduce to the PAF mortality problem. It suffices to consider functions of natural numbers.

THEOREM 3.12. *Mortality, as well as global convergence, of a piecewise affine function $f : \mathbb{N} \to \mathbb{N}$ with integer coefficients is PSPACE-hard (under logspace reductions).*

Proof. We reduce from mortality of fixed-space machines with oblivious queue access.

The reduction transforms a machine M, given with its space bound n, into a representation of a piecewise-affine function f that simulates M and is mortal if and only if M is.

Let us first describe the essence of this simulation. Suppose that M has m states. A configuration of M is specified by (q, w) where $q \in [0, m-1]$ is the control state, and $w \in \{0, 1\}^n$ is the contents of the queue.

By identifying w with the integer that it represents in binary notation, we define an integer that encodes a configuration:

$$\langle q, w \rangle = q \cdot 2^n + w \,. \tag{13}$$

Next, we define f as a function that maps a configuration to the next, simulating the machine; we present the definition by cases.
For each $(q, b, q', b') \in \delta$, we define

$$f(\langle q, bz \rangle) = \langle q', zb' \rangle \,. \tag{14}$$

If there is no transition starting with (q, b),

$$f(\langle q, bz \rangle) = 0. \tag{15}$$

We now verify that the above definitions yield a piecewise-affine function. Since this is a one-dimensional PAF, its regions of definition are intervals and we use the notation $a + [b, c]$ for $[a + b, a + c]$. Let $R_{q,b}$ be the interval $q \cdot 2^n + b \cdot 2^{n-1} + [0, 2^{n-1} - 1]$. The function f is defined on each such interval according to the applicable case:

1. Every transition described by (14) is translated into:
   $x \in R_{q,b} \Rightarrow\; f(x) = \; 2(x - q \cdot 2^n - b \cdot 2^{n-1}) + q' \cdot 2^n + b'$
2. If there is no such transition,
   $x \in R_{q,b} \Rightarrow f(x) = 0.$

$\square$

Inspecting the machines constructed in the proof of Lemma 3.10, we see that every transition changes the control state. In the last construction, this means that there will be no fixed point other than zero. Hence, we also have

COROLLARY 3.13. *Mortality, as well as global convergence, of a piecewise affine function $f : \mathbb{N} \to \mathbb{N}$ with integer coefficients is PSPACE-hard (under logspace reductions).*

It may be interesting to note that the set of coefficients of x in the function produced by the reduction is rather limited: 0 or 2.

3.4.3 Discussion

It now becomes natural to ask for the complexity of our decision problems for restrictions of the one-dimensional problem. Of course, many restrictions can be invented. By looking at the proof of Theorem 3.4, we see that if a^+ or a^- are greater than 1, it is an easy case. I find the following questions particularly interesting (compare Theorem 2.20 and Corollary 2.10):

- *Open problem* 1. Consider one-dimensional integer piecewise affine functions defined by
  $$f(x) = x + b_i \quad \text{for } x \in H_i, \tag{16}$$
  where the sets $H_1, \ldots, H_p$ are an exhaustive partition of $\mathbb{Z}$ (or just $\mathbb{N}$) into p intervals, and the constants $b_1, \ldots, b_p$ arbitrary integers. What is the complexity of the mortality problem for such functions?
- *Open problem* 2. Is there a polynomial algorithm for the mortality problem of one-dimensional PAFs when the number of intervals is considered a constant?
- *Open problem* 3. Is there an algorithm for the mortality problem of PAFs over $\mathbb{Q}$ (with rational coefficients), when continuity is not assumed?

4 Decidability for affine functions

Affine-linear transformations have been much studied in Dynamical System Theory, e.g., [31]. The setting there is that of a continuous state space, but the techniques are hardly affected by the restriction to $\mathbb{Z}^n$. Full proofs for the theorems below have been nonetheless included for the sake of completeness, but also to make it easier to verify that they do hold in our setting.

We rely on some properties of matrices and matrix powers. They may be found in textbooks such as [56].

THEOREM 4.1. *There is a polynomial-time algorithm for deciding global convergence over $\mathbb{Z}^n$ of an affine function $f(\vec{x}) = A\vec{x} + b$.*

Proof. Note, first, that $f(0) = b$. Hence, for global convergence we must have $b = 0$. So the function is $f(\vec{x}) = A\vec{x}$ and the question is whether $\forall \vec{x} \exists k : A^k \vec{x} = 0$. In fact, it suffices to find such a k for every vector in the standard basis of $\mathbb{Q}^n$. To see this, let $e_1, \dots, e_n$ be the basis vectors. Suppose that for every e_i there is a k_i such that $A^{k_i} e_i = 0$. Let $k = \max(k_1, \dots, k_n)$. Then, for general $\vec{x} = a_1 e_1 + \cdots + a_n e_n$, we have

$$A^k \vec{x} = \sum_i a_i A^k e_i = \sum_i a_i A^{k-k_i} A^{k_i} e_i = \sum_i a_i A^{k-k_i} 0 = 0.$$

Note that for k as above we actually have $A^k = \mathbf{0}$, that is, matrix A is nilpotent. This is known to be true if and only if $A^n = \mathbf{0}$ ([56, Theorem 1.13]). Hence, this is all we have to check. □

THEOREM 4.2. *There is a polynomial-time algorithm for deciding mortality over $\mathbb{Z}^n$ of an affine function $f(\vec{x}) = A\vec{x} + b$.*

Proof. We shall prove that mortality is only possible if $b = 0$, so it coincides with global convergence to zero.

It is easily proved by induction that for any $r > 0$, we have

$$f^{(r)}(\vec{x}) = A^r \vec{x} + (A^{r-1} + \cdots + A + I)b = A^r \vec{x} + f^{(r-1)}(b)\,. \tag{17}$$

Suppose that f is mortal. Then, in particular, the sequence $b, f(b), f^{(2)}(b), \dots$ reaches 0, that is, $f^k(b) = 0$ for some $k > 0$. Note that $b = f(0)$, so we conclude that this sequence is periodic. Let S be the (finite) set of vectors appearing in this sequence. Now, for any $\vec{x} \neq 0$,

$$f^{(r)}(\vec{x}) - A^r \vec{x} \in S$$

so, if f is mortal, we must have $A^r \vec{x} \in -S$ for some r. Suppose that for all r, $A^r \vec{x} \neq 0$. That is, the vector $A^r \vec{x}$ contains a non-zero component. Then, for integer λ large enough, we shall have $A^r(\lambda \vec{x}) \notin -S$ for all r (simply choose λ larger than the maximum absolute value of components of vectors in S), a contradiction; we deduce that $A^r \vec{x}$ must be 0 for some r. Thus, as in the previous proof, A is nilpotent.

Now, $A^n = \mathbf{0}$, and by (17), $f^{(n)}(b) = f^{(n-1)}(b)$, that is, f has a fixed point at $f^{(n-1)}(b)$. For mortality we require the only fixed point to be 0, which means $b = 0$. □

5 Conclusion

We summarize the work presented and follow it with some open problems and research directions, including some pointers to the literature.

5.1 Summary of results

This article presents work on the border between Dynamical System Theory (much of which refers to continuous state-spaces) and the Theory of Computation in discrete models. In particular, this work connects Dynamical System Theory to problems of program termination, inspired by previous work on the termination of affine and piecewise-affine loops [58, 14, 7] and on the analysis of the mortality problem in a continuous domain [9]. I'd like to argue that such results may be of interest also in the context of continuous-space dynamical systems (e.g., as models of some kinds of physical systems), since, unlike previous proofs, the undecidability is not derived from the encoding of information into the fractional bits of a value, which boils down to assuming unlimited precision in measurement[7].

We have shown that mortality, convergence (to zero) and *convergence to any fixed point* are undecidable in general for iterated piecewise-affine functions over $\mathbb{Z}^2$. The undecidability is affected by several properties of the functions: we get decidable problems if we restrict the dimension to one, or the functions to affine ones. Some other restrictions preserve undecidability, for example, the problems are undecidable in two dimensions for a large enough, but fixed, number of affine pieces. The undecidability results have been refined to Π^0_2 completeness; the decidable problems have been investigated for their complexity, which ranges from PTIME to PSPACE-complete. The strengthened undecidability results also apply to the continuous setting, improving on previous work.

It is the author's hope that the results presented are of interest, but also the techniques and connections made to Collatz-like problems and to automata that capture PSPACE. Undecidability for a very restricted class of functions, the monic functions, may be a useful result for future work in the sense that it may be easier to translate the restricted class into other problems of interest.

5.2 Reachability-type problems

The focus of this work is on deciding global properties of systems, so we have not dwelt on the problem that corresponds to the common Turing-machine *halting problem*, namely: given a function and an initial value x_0, does the sequence beginning with x_0 reach zero. However it is easy to see, from our proofs, that this problem is RE-complete in two dimensions and PSPACE-complete in one. Interestingly, for functions over $\mathbb{R}$, this "halting problem" was mentioned as an open problem in [38, 9]. For affine-linear functions, the problem: does the sequence beginning with x_0 reach a given value y? Is known as the *orbit problem* and was solved in polynomial time over the rationals in [36] (technically, they refer to linear transformations, but the problem for any affine-linear transformation is easily reducible to the former).

[7]There are works that address this criticism in another way, namely by considering robust constructions that work in dynamical systems subject to some noise. There is research about the computational power of robust systems [11, 32] and about "robust" versions of decision problems about dynamical systems [12, 28, 29].

Some extensions, or variants, of the orbit problem prove harder (in terms of computational complexity, or in terms of establishing results about them). A celebrated open problem is the decidability of *Skolem's problem* (sometimes referred to as *Pisot's problem*): consider a function defined by $f(x) = Mx$ where M is a square ($k \times k$) integer matrix; given an initial value u, decide whether the sequence $(M^i u)_{i \geq 0}$ ever hits a given hyperplane $a \cdot x = 0$. Vyalyi and Tarasov [60] point out that this problem and Kannan and Lipton's orbit problem can both be seen as special cases of the problem of reaching a given convex region, which they call *the chamber hitting problem*, and which they relate to questions of formal-language theory. Chonev et al. [17] give decidability and complexity results for the problem in case that the "chamber" is a linear subspace of dimension up to 3. Recent progress on this problem is reported in [18]. The case where the target is a closed half-space is considered in [49]. See also the survey [48].

In a variant studied by Cortier [22], a point in $\mathbb{N}^n$ is subjected to an arbitrary number of iterations of an affine-linear function f_1, and then to iterations of another function f_2; the reachability problem (for a single target point) is shown to be decidable. A generalization involving a larger number of stages $f_1, f_2, \ldots, f_p$ is shown to be undecidable once p is beyond some (unspecified) constant. Over $\mathbb{Z}$ (in one dimension), decidability for any number of functions is claimed in [30], while undecidability for $\mathbb{Z}^n$, $n \geq 2$, follows from results on the matrix mortality problem (see below).

A beautiful paper by Bell and Potapov [2] shows how various reachability problems are related to questions about sets of matrices and to Post's correspondence problem. In particular, show that given 5 integer matrices, $M_1, \ldots, M_5$, of dimension at least four, an initial (integer) vector x_0 and a final vector y, it is undecidable whether y can be reached by applying some sequence of these linear transformation, i.e., $y = M_{i_1} M_{i_2} \ldots M_{i_k}$ for some finite sequence $i_1, \ldots, i_k$. They also show an undecidability result for five matrices in two dimensions, but only over the rationals. The punch of this work is in showing that undecidability holds for very small instances. One naturally wonders whether, perhaps using similar techniques, undecidability bounds for our problems (such as the number of regions, discussed in Section 2.2 could be made much smaller.

Peter van Emde Boas and Marek Karpinski [59] describe the "mouse in an octant" problem, attributed to Lothar Budach, a reachability problem for a very simple kind of 2-counter programs, whose decidability status is apparently still open. Its analysis [59] shows that number-theoretic considerations, somewhat in the flavour of the Collatz problem, are involved in understanding such programs. See also [53, 4].

5.3 Open problems

Here is a recap of some open problems—all concern piecewise-affine functions over the integers, and have been described in more detail in previous sections.

1. Is there any constant bound on the number of regions which suffices to make mortality of monic piecewise-affine functions over $\mathbb{Z}^2$ as hard as the general problem, i.e., Π_2^0 hard? Or just undecidable?

2. (Braverman's problem) Is mortality decidable for functions which are zero outside a single convex region (and affine within)?

3. What is the complexity of the mortality problem in the one-dimensional case, when the coefficient of x is always 1?

4. What is the complexity of the mortality problem in the one-dimensional case, when the number of intervals is considered a constant?

We also note some open problems regarding the continuous case (i.e., functions with rational coefficients over either $\mathbb{Q}^n$ or $\mathbb{R}^n$):

1. What is the computability class of mortality for continuous piecewise-affine functions over $\mathbb{Q}^2$?

2. What is the computability class of mortality for (not necessarily continuous) piecewise-affine functions over $\mathbb{Q}$?

3. What is the computability class of global convergence, and of global convergence to a fixed point, of piecewise-affine functions over $\mathbb{Q}^2$? We only know they are Π_2^0-hard (when the functions are not assumed continuous).

Finally, a possible direction for extending this study is to look at polynomials and piecewise-polynomial functions (a recent paper [27] considers a related model—register machines with polynomial updates. It shows PSPACE complexity for a machines with a single register).

A Proof of Lemma 2.11

To obtain the bounds stated in Lemma 2.11, the following steps will be taken. First, we recall a particular universal counter machine constructed by Korec [39]. Then, we redo the construction of Theorem 2.7 in a more ad-hoc fashion, to get optimised results for this particular universal machine. Finally, we reduce the number of registers in the machine.

Korec's machine In [39], Ivan Korec presents several universal counter machines, differing in details of their operation and the instruction sets they use. Among them, we use one which is called $\mathcal{U}_{19}$ (because it has 19 states). This machine has the following features:

1. It simulates two-counter machines. This suffices for our purposes, because the totality problem for 2CM is Π_2^0-complete.

2. It has seven registers.

   - $R1$ is where the code $\widehat{M}$ is kept.

- $R7$ is used as a back-up copy of $R1$ when decoding instructions. This is done so that the sum $R1 + R7$ is invariant. Hence, when restarting the simulation, as needed in the construction of Theorem 2.7, all we need is to transfer the contents of $R7$ to $R1$ and we are sure to recover $\widehat{M}$.
- Five registers are used for the actual simulation; they are called $R0, R2, R4, R5$ and $R6$ (the name $R3$ is not used). The first, $R0$, should hold the input in the beginning (corresponding to R_2 in Definition 2.6).

3. Importantly for our purpose, it has a restricted instruction set. Each state is associated with just a single instruction, out of three instruction types: $ZM(X, i, j)$ tests whether register X is null; if it is not, it is decremented; execution proceeds at state i in the first case, or j in the other. Instruction $P(X, i)$ increments X and proceeds to state i. Instruction $Z(X, i, j)$ just test X, jumping to state i if it is null and to j otherwise.

When applying the construction of Theorem 2.7, we add the counters V and W as previously described, but do not modify them on every step, just at a single point in the program, that has the property that it cuts every potentially non-terminating cycle; this accomplishes the same goal. We also include a register Y to keep the input (what was denoted by R_{n+1} in the proof of Theorem 2.7).

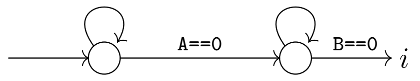
Reducing the number of registers. We use the standard encoding trick used to simulate many registers on two. Specifically, registers $R0, R2, R4, R5, R6, R7$, V, W and Y will be replaced by two registers only, called A and B. The values of the eight registers are all represented by a single number, normally placed in A, using exponents of primes. For this purpose, we assign primes as follows:

register	$R0$	$R2$	$R4$	$R5$	$R6$	$R7$	V	W	Y
prime	11	7	3	2	5	23	17	13	19

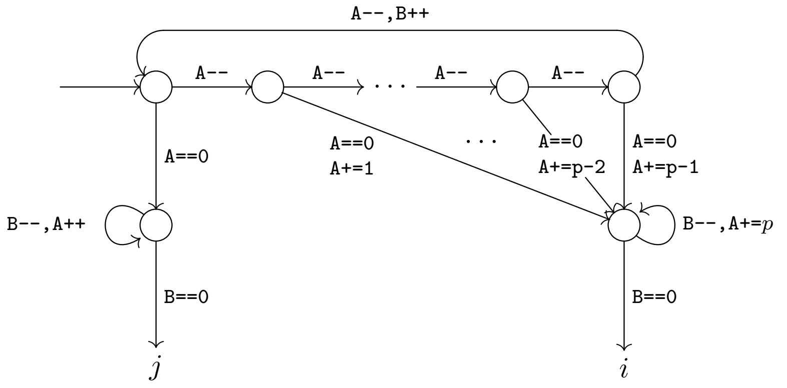
This assignment is not (entirely) random. It is intended to reduce the number of control states in the machine.

Now, we should explain how the two instruction types of Korec's machine are simulated. It is important to note here that we are simulating Korec's restricted instructions with the richer instruction set available to an enhanced counter machine as defined in this work.

- To simulate an instruction of the type $P(X, i)$ we should multiply A by p, the prime number associated with register X. This is accomplished by two states, as depicted in the next diagram, which uses a C-like syntax to express the updates and conditions for passing from state to state.



- To simulate an instruction of the type $ZM(X, i, j)$ we should divide A by p, where p is the prime associated with register X. If the division is even, we should return the quotient to A and move to state j; if it is not, we should restore A to its previous value, and move to state i. This is accomplished by $p + 2$ states, as shown in the following diagram. The loop in the upper part of the diagram consists of p states and accomplishes a division by p.



- Finally, the $Z(X, i, j)$ instruction is simulated by a loop as in ZM, however always recovering the old value of A on exit.

It is important to note that the loop above cannot be a cause of non-mortality.

We are now ready to calculate the number of states in the resulting machine, shown in Table 8.

The state numbers correspond to Figure 6 of [39], and the reason that they are not consecutive is that this is how they are given there. States 1–32 represent Korec's machine; states 40–41 perform the update of V, W; and states 50–63 perform the reset operations when W is zero. This includes resetting several registers to zero, copying Y into $R0$ setting $R1$ to $\widehat{M}$ and W to $2V + 1$.

The "cost" column lists the number of 3CM states used to simulate the instruction. Note that $R1$ is not encoded, therefore an operation on $R1$ costs 1; moreover, we can use an enhanced counter-machine instruction to perform the addition at 62. At state 63, we use an ad-hoc implementation, based on the implementation of a ZM instruction, to repeatedly divide the encoded number by 17 (the prime for V) and multiply by 169 (the code for W, squared). We also multiply by 13 once again at the exit node.

While constructing this table, I have included a "peephole optimisation": when an instruction (of any type) is succeeded by a P instruction, it is possible to use the exit state of that previous instruction to perform the multiplication that simulates the P. This optimisation reduces the cost of the subsequent P instruction to zero.

By summing the costs along the table, the sum of 216 states is obtained.

state	instruction	cost	comments
1	ZM(R1,6,3)	1	
3	P(R7,1)	0	
4	ZM(R5,40,6)	4	originally ZM(R5,7,6)
6	P(R6,4)	0	
7	ZM(R6,4,9)	7	
9	P(R5,10)	0	
10	ZM(R7,13,12)	25	
12	P(R1,7)	0	
13	Z(R6,1,14)	7	
14	ZM(R4,16,1)	5	
16	ZM(R5,23,18)	4	
18	ZM(R5,27,20)	4	
20	ZM(R5,30,22)	4	
22	P(R4,16)	0	
23	ZM(R0,1,32)	13	
27	ZM(R2,1,32)	9	
30	P(R0,31)	0	
31	P(R2,32)	2	
32	ZM(R4,0,1)	5	jump to 0 is exit
40	P(V,41)	0	
41	ZM(W,50,7)	15	
50	ZM(R1,51,50)	1	
51	ZM(R2,52,51)	9	
52	ZM(R4,53,52)	5	
53	ZM(R5,54,53)	4	
54	ZM(R6,55,54)	7	
55	ZM(R7,56,55)	25	
56	ZM(R0,57,56)	13	
57	ZM(Y,60,58)	21	
58	P(R0,59)	0	
59	P(R5,57)	2	
60	ZM(R5,62,61)	4	
61	P(Y,60)	0	
62	$R1{+}=\widehat{M}$, goto 63	1	
63	$W{+}=2V+1$, goto 1	19	

Table 8: Korec's machine with the additions for the mortality proof, and details for calculation of the machine's size.

Acknowledgement

I thank Jakob Grue Simonsen for a good discussion, prompting me to extend and refine my initial results. Elaborate comments and suggestions by the anonymous referees have also contributed many improvements to this paper.

References

[1] Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theor. Comput. Sci*, 138(1):35–65, 1995.

[2] Paul Bell and Igor Potapov. On undecidability bounds for matrix decision problems. *Theoretical Computer Science*, 391(1-2):3–13, 2008.

[3] Paul C. Bell, Mika Hirvensalo, and Igor Potapov. Mortality for 2×2 matrices is NP-hard. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012*, volume 7464 of *Lecture Notes in Computer Science*, pages 148–159. Springer, 2012.

[4] Amir M. Ben-Amram. A comment on Budach's mouse-in-an-octant problem. Technical Report arxiv.org/abs/1305.0911, 2013.

[5] Amir M. Ben-Amram. Mortality of Iterated Piecewise Affine Functions over the Integers: Decidability and Complexity. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 514–525, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[6] Amir M. Ben-Amram and Samir Genaim. Ranking functions for linear-constraint loops. *Journal of the ACM*, 61(4), 2014.

[7] Amir M. Ben-Amram, Samir Genaim, and Abu Naser Masud. On the termination of integer loops. *ACM Transactions on Programming Languages and Systems*, 34(4):16:1–16:24, December 2012.

[8] Michel Blockelet and Sylvain Schmitz. Model-checking coverability graphs of vector addition systems. In Filip Murlak and Piotr Sankowski, editors, *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11)*, volume 6907 of *Lecture Notes in Computer Science*, pages 108–119. Springer, August 2011.

[9] Vincent D. Blondel, Olivier Bournez, Pascal Koiran, Christos H. Papadimitriou, and John N. Tsitsiklis. Deciding stability and mortality of piecewise affine dynamical systems. *Theor. Comput. Sci.*, 255(1-2):687–696, 2001.

[10] Vincent D. Blondel and John N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

[11] Olivier Bournez, Daniel S. Graça, and Emmanuel Hainry. Computation with perturbed dynamical systems. *Journal of Computer and System Sciences*, 79(5):714–724, 2013.

[12] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In Parosh Aziz Abdulla and Igor Potapov, editors, *Reachability Problems, 7th International Workshop, RP 2013, Uppsala, Sweden, September 24-26, 2013 Proceedings*, volume 8169 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[13] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. The polyranking principle. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proc. 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1349–1361. Springer Verlag, 2005.

[14] Mark Braverman. Termination of integer linear programs. In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification, CAV'06*, volume 4144 of *LNCS*, pages 372–385. Springer, 2006.

[15] Michael Brin and Garrett Stuck. *Introduction to dynamical systems.* Cambridge University Press, Cambridge, UK, 2002.

[16] Serge Burckel. Functional equations associated with congruential functions. *Theoretical Computer Science*, 123(2):397–406, January 1994. Note.

[17] Ventsislav Chonev, Joël Ouaknine, and James Worrell. The orbit problem in higher dimensions. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 941–950, New York, NY, USA, 2013. ACM. extended version in CoRR, `http://arxiv.org/abs/1303.2981`.

[18] Ventsislav Chonev, Joël Ouaknine, and James Worrell. The polyhedron-hitting problem. Technical Report arxiv.org/abs/1407.1889, 2014. to appear in the proceedings of SODA 2015.

[19] Michael Colón and Henny Sipma. Synthesis of linear ranking functions. In *7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2031 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2001.

[20] John. H. Conway. Unpredictable iterations. In *Proc. 1972 Number Theory Conf., Univ. Colorado, Boulder*, pages 49–52. 1972. Reprinted with historical commentary in [41].

[21] John. H. Conway. FRACTRAN: a simple universal programming language for arithmetic. In T. M. Cover and B. Gopinath, editor, *Open Problems in Communication and Computation*, pages 3–27. Springer-Verlag, 1987. Reprinted with historical commentary in [41].

[22] Véronique Cortier. About the decision of reachability for register machines. *Theoretical Informatics and Applications*, 36(4):341–358, 2002.

[23] Stéphane Demri. On selective unboundedness of VASS. *Journal of Computer and System Sciences*, 79(5):689–713, August 2013.

[24] Philippe Devienne, Patrick Lebegue, Anne Parrain, Jean-Christophe Routier, and Jörg Würtz. Smallest horn clause programs. *Journal of Logic Programming*, 27(3):227–267, 1996.

[25] Javier Esparza. Decidability and complexity of Petri net problems—an introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets, Vol. I: Basic Models*, volume 1491 of *LNCS*, pages 374–428. Springer-Verlag (New York), Dagstuhl, Germany, September 1996, revised paper 1998.

[26] Javier Esparza and Mogens Nielsen. Decidability issues for petri nets. Technical Report RS-94-8, BRICS, Department of Computer Science, University of Aarhus, 1994.

[27] Alain Finkel, Stefan Göller, and Christoph Haase. Reachability in register machines with polynomial updates. In Krishnendu Chatterjee and Jiří Sgall, editors, *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS'13)*, volume 8087 of *Lecture Notes in Computer Science*, pages 409–420. Springer, August 2013.

[28] Martin Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In *Computer Science Logic, 13th International Workshop, CSL99, Madrid, Spain, September 2025, 1999, Proceedings*, volume 1683 of *Lecture Notes in Computer Science*, pages 126–139. Springer, 1999.

[29] Martin Fränzle. What will be eventually true of polynomial hybrid automata? In Benjamin C. Pierce Naoki Kobayashi, editor, *Theoretical Aspects of Computer Software, 4th International Symposium, TACS 2001, Sendai, Japan, October 2931, 2001, Proceedings*, volume 2215 of *Lecture Notes in Computer Science*, pages 340–359. Springer, 2001.

[30] Daniel Fremont. The reachability problem for affine functions on the integers. Technical Report arxiv.org/abs/1304.2639, 2013.

[31] Oded Galor. *Discrete Dynamical Systems.* Springer, 2007.

[32] Daniel S. Graça, Manuel Lameiras Campagnolo, and Jorge Buescu. Robust simulations of Turing machines with analytic maps and flows. In S. Barry Cooper,

Benedikt Löwe, and Leen Torenvliet, editors, *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005, Proceedings*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179. Springer, 2005.

[33] Tero Harju. Post correspondence problem and small dimensional matrices. In Volker Diekert and Dirk Nowotka, editors, *Developments in Language Theory, 13th International Conference, DLT 2009, Proceedings*, volume 5583 of *Lecture Notes in Computer Science*, pages 39–46. Springer, 2009.

[34] Gabor T. Herman. A simple solution of the uniform halting problem. *The Journal of Symbolic Logic*, 34(4):pp. 639–640, 1969.

[35] Philip Hooper. The undecidability of the Turing machine immortality problem. *The Journal of Symbolic Logic*, 31(2):219–234, June 1966.

[36] R. Kannan and R. J. Lipton. Polynomial-time algorithm for the orbit problem. *J. ACM*, 33(4):808–821, October 1986.

[37] František Kaščák. Small universal one–state linear operator algorithm. In Ivan M. Havel and Vaclav Koubek, editors, *Proceedings of MFCS '92. Mathematical Foundations of Computer Science (MFCS '92)*, volume 629 of *LNCS*, pages 327–335, Berlin, Germany, August 1992. Springer.

[38] Pascal Koiran, Michel Cosnard, and Max Garzon. Computability with low-dimensional dynamical systems. *Theor. Comput. Sci.*, 132:113–128, September 1994.

[39] Ivan Korec. Small universal register machines. *Theoretical Computer Science*, 168(2):267–301, 1996.

[40] Stuart A. Kurtz and Janos Simon. The undecidability of the generalized Collatz problem. In Jin-Yi Cai, S. Barry Cooper, and Hong Zhu, editors, *Theory and Applications of Models of Computation, 4th International Conference, TAMC 2007, Shanghai, China, May 22-25, 2007*, volume 4484 of *LNCS*, pages 542–553. Springer, 2007.

[41] Jeffrey C. Lagarias. *The Ultimate Challenge: The* $3x + 1$ *Problem.* American Mathematical Society, 1st edition, 2010.

[42] Richard J. Lipton. The reachability problem requires exponential space. Technical Report 63, Yale University, 1976.

[43] J. Marcinkowski. Achilles, turtle, and undecidable boundedness problems for small DATALOG programs. *SIAM Journal on Computing*, 29(1):231–257, 1999.

[44] Maurice Margenstern. Frontier between decidability and undecidability: a survey. *Theoretical Computer Science*, 231(2):217 – 251, 2000.

[45] G. Memmi and G. Roucairol. Linear algebra in net theory. In Wilfried Brauer, editor, *Net Theory and Applications*, volume 84 of *LNCS*, pages 213–223. Springer Berlin / Heidelberg, 1980.

[46] Marvin L. Minsky. *Computation: finite and infinite machines.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

[47] Cristopher Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, 64(20):2354–2357, May 1990.

[48] Joël Ouaknine and James Worrell. Decision problems for linear recurrence sequences. In Alain Finkel, Jérôme Leroux, and Igor Potapov, editors, *Reachability Problems - 6th International Workshop, RP 2012, Proceedings*, volume 7550 of *Lecture Notes in Computer Science*, pages 21–28. Springer, 2012.

[49] Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14. SIAM, 2014.

[50] Michael S Paterson. Unsolvability in 3 by 3 matrices. *Studies in Applied Mathematics*, 49(1):105–107, 1970.

[51] João Sousa Pinto, Joël Ouaknine, and James Worrell. On termination of integer linear loops. Technical Report arxiv.org/abs/1407.1891, 2014. to appear in the proceedings of SODA 2015.

[52] Andreas Podelski and Andrey Rybalchenko. A complete method for the synthesis of linear ranking functions. In *VMCAI*, pages 239–251, 2004.

[53] A. Pultr and J. Úlehla. On two problems of mice. In *Proceedings of the 10th Winter School on Abstract Analysis*, pages 249–262, 1982.

[54] H.T. Siegelmann and E.D. Sontag. Analog computation, neural networks, and circuits,. *Theor. Comp. Sci.*, 131:331–360, 1994.

[55] Joseph H. Silverman. Integer points, Diophantine approximation, and iteration of rational maps. *Duke Math. J.*, 71(3):793–829, 1993.

[56] G. W. Stewart. *Matrix Algorithms, Volume II : Eigensystems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[57] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* Univ. of California Press, 2nd edition, 1951.

[58] Ashish Tiwari. Termination of linear programs. In Rajeev Alur and Doron Peled, editors, *Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 387–390. Springer Berlin / Heidelberg, 2004.

[59] Peter van Emde Boas and Marek Karpinski. A number theoretic problem arising from a problem in automata theory. *Bulletin of the EATCS*, 12:50–53, 1980.

[60] M. Vyalyi and S. Tarasov. Orbits of linear maps and regular languages. *Journal of Applied and Industrial Mathematics*, 5:448–465, 2011. Original Russian Text published in Diskretnyi Analiz i Issledovanie Operatsii, 2010, Vol. 17, No. 6, pp. 2049.

[61] K. Wagner and G. Wechsung. *Computational Complexity.* D. Reidel, Dordrecht, 1986.

[62] Yuri V. Matiyasevich. *Hilbert's Tenth Problem.* MIT Press, Cambridge, Massachusetts, 1993.