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

Amir M. Ben-Amram

June 27, 2013

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 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. We show that also over integers, undecidability (moreover, Π^0_2 completeness) begins at two dimensions. 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 complexity is PTIME for affine functions, but for piecewise-affine ones it is PSPACE-complete. The undecidability proofs use some variants of the Collatz problem, which may be of independent interest.

1 Introduction

The purpose of this paper is to study some computational problems regarding the asymptotic behaviour of discrete-time dynamical systems, in particular problems related to questions about the termination of simple programs. 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 is called *a trajectory.* A class of functions much studied in the Dynamical System literature is piecewise affine functions (Definition 1.3 at the end of this section). Some of the central problems regarding such systems involve their asymptotic behavior, among which are *global convergence* and *mortality*, defined next.

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., $\exists t \geq 0 : x_t = 0$).

These problems were studied from a Computability viewpoint by Blondel et al. [5]. 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., [8]).

THEOREM 1.2. [5] *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 is decidable for $n = 1$ when the function is continuous.

Among the many decision problems studied for dynamical systems, mortality is the most appealing in its discrete setting since it is a restricted halting problem for a very simple type of program (more on the connection to program termination problems below). The global convergence problem 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) and we will treat both problems throughout the following sections.

The main contribution of this paper is to establish for the integer setting results similar to Theorem 1.2. The proofs in [5] 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 [5] 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 ([5] left this question open). Furthermore, we prove Π_2^0 completeness, which is a sharper result than mere undecidability.

Next, we consider some other restrictions on the space of functions. Restriction of the number of regions is discussed in Section 4. Section 5 shows undecidability for two-dimensional functions of a very simple form, $f(\vec{x}) = f(x_1, x_2) = (x_{i_1} + b_1, x_{i_2} + b_2)$. This result is also applicable to the rationals, implying an improvement of the characterization of [5].

Sections 6–7 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.

A problem similar to global convergence to zero, also found in the dynamical system literature, is *global convergence to (any) fixed point* [22]. 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. This problem is taken up in Section 8.

To conclude this introduction, 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 [33]).

Taking classical (discrete) computability to continuous dynamics, Moore [30] discusses the significance of undecidability (in the Turing sense) to dynamical systems. He shows that a TM 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 already in two dimensions, but not in one [22]. Blondel and Tsiklitis [6] 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). 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 [3, 7, 11, 31]). 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 [37], 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 [8] 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 [36], but not for integers [41]. Another classical example of an integer-specific problem is the Collatz problem (or "$3x + 1$ problem") [24]. Lagarias' excellent volume shows clearly that this problem is related both to Dynamical System Theory and to Computability Theory. Regarding Computability, Conway [12, 13] and several subsequent works [21, 10, 14, 26, 23] proved undecidability results for *generalized Collatz problems* by showing how to simulate a counter machine. We shall make essential use of this idea, building on the reductions in [23], which were the only ones (known to the author) to derive Π^0_2 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., [34]. 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 [9].

Part of this work has been previously presented at STACS 2013 [2].

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* 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 [5] and other related publications. Section 3 discusses an implication of this restriction (and propose a relaxation).

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 [29], 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 [5]. 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 $[i, b_1, \ldots, b_n, D_1, \ldots, D_n, k]$ where $1 \leq i \leq \ell$ is the internal state number, $b_j \in \{Z, P\}$ 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.

The class Π^0_2.

Definition 1.4. 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 RE (characterized by formulas $(\exists y)P(x,y)$). Standard Π_2^0-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. Note that even undecidability (let alone Π_2^0-completeness) of mortality is non-trivial since many programs, while halting from all initial states, still diverge if started in a configuration that is not reachable in a proper computation (i.e., from an initial state)[1]. For Turing machines, Hooper [19] and Herman [18] proved the undecidability of mortality (under two different definitions of the state space). Kurtz and Simon proved the following

THEOREM 1.5 ([23])**.** *The mortality problem for counter machines (CMs) with $n \geq 2$ counters is Π_2^0-complete.*

2 Undecidability in Two Dimensions

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 [23], Kurtz and Simon reduce the CM mortality problem to a Generalized Collatz Problem. We shall use this problem to prove our result for piecewise affine functions.

The definition below is based on [23], with a non-essential modification for convenience in the sequel.

Definition 2.1. A function $g : \mathbb{N}_+ \to \mathbb{N}_+$ is called a *generalized 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 $m, a_0, b_0, \ldots, a_{m-1}, b_{m-1}$ (say 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. In our notation, it is given by $m = 2$, $a_0 = 1, b_0 = 0$, $a_1 = 6, b_1 = 4$.

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

THEOREM 2.3. [23] *GCP is Π_2^0-complete.*

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

[1]An anecdotal evidence of this difference is that for Petri nets, halting is EXPSPACE-hard [15, 25], while mortality is PTIME [28, 16].

THEOREM 2.4. *Global convergence and mortality over $\mathbb{Z}^2$ of piecewise affine functions with integer coefficients is a Π_2^0-complete problem.*

Proof. Like the GCP, our problem is clearly a "$\forall\exists$" problem (more specifically: for all $\vec{x} \in \mathbb{Z}^2$, there is an n such that $f^n(\vec{x}) = 0$), hence belonging to Π_2^0. For Π_2^0-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 generalized 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	$x>m,\ y \geq 0$	$(x-m, y+1)$
R_0	$x=0\,,\ y > 0$	$(a_0y+b_0, 0)$
R_1	$x=1\,,\ y > 0$	$(a_1y+b_1, 0)$
R_2	$x=2\,,\ y \geq 0$	$(a_2y+b_2, 0)$
$\vdots$		
R_{m-1}	$x=m,\ y \geq 0$	$(a_my+b_m, 0)$
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 the origin. Thus, f globally converges to zero if and only if it is mortal if and only if g satisfies the GCP. $\square$

In the following sections we prove results which are strictly stronger than Theorem 2.4. However their proofs are also more involved, so it seemed worthwhile to begin by presenting a simple proof, which is also re-used in the next result.

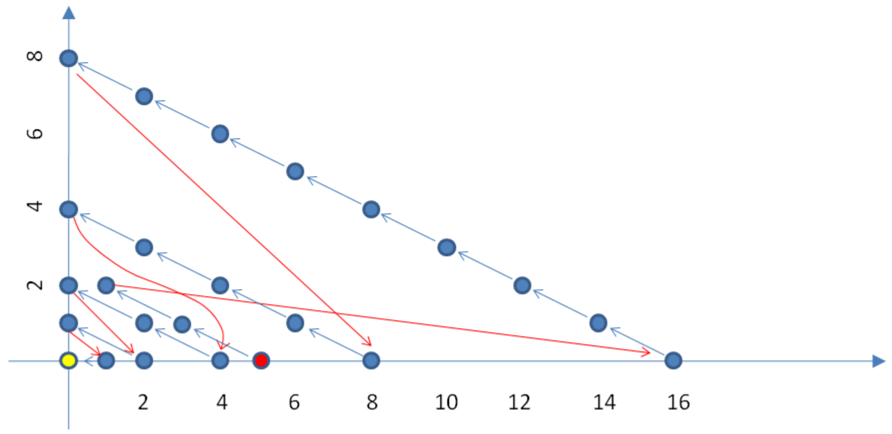

Figure 1: Simulating the Collatz sequence from $x = 5$ (the initial point is $(5, 0)$).

3 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[2]. In [23], 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—the number of instructions plus 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.

A reduction of a CM halting problem to mortality appears both in [23] and in [5]. Since the latter reduction (henceforth: the BBKPT reduction) is simpler, we will build upon it. A weakness of BBKPT is that they reduce from halting on a given input, not from totality, but we will change that. First, we review their 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 "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

[2]Undecidability, though not Π_2^0 completeness, has been shown to hold for a fixed modulus in [21, 14, 26]. Their constructions could perhaps be used to give an alternative proof of the result.

$2\max(1,V)$ in W and 0 in V. After having done that, it moves into state 1 (the initial state of M).

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.

The reader may want to reflect on why this ensures mortality if and only if M halts on the specified initial state (or turn to [5] for explanations).

Let U be 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. We assume, with no loss of generality, that U operates so that it never increases R_1 beyond its initial value.

Since counter machines in themselves are not our goal, we can grant ourselves the convenience of using a slightly stronger computational model.

Definition 3.1. An *enhanced CM* is a counter machine, as specified in Section 1, except that in an instruction $[i, b_1, \ldots, b_n, D_1, \ldots, D_n, k]$, the D_j component is an integer $d \geq -1$, which is added to R_j. Hence, one may increase the value of a register by more than 1.

THEOREM 3.2. *From an (ordinary) counter machine M, 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 $(1, \langle x, 0, \ldots, 0\rangle)$. The number of registers and instructions of U_M is independent of M.*

Proof. U_M is obtained by applying the BBKPT reduction to the universal machine U, with the following modifications. Suppose that U has n registers. We add a register R_{n+1}. We modify the instructions so that this register is incremented whenever R_1 is decremented and vice versa. We modify the effect of the reset state (more precisely, the group of states that perform the reset, starting at q_0) so that it sets $R_1, R_2, \ldots, R_{n+1}$ to $R_1+R_{n+1}, \widehat{M}, 0, \ldots, 0$ respectively. It sets $\langle R_1, R_{n+1}\rangle$ to $\langle R_1+R_{n+1}, 0\rangle$ using increments and decrements—this requires a constant number of states—sets $R_2, \ldots, R_n$ to zero using decrements, and then sets R_2 to $\widehat{M}$ in a single step (here we use an "enhanced" instruction). It proceeds to update V and W as in the BBKPT reduction, also a constant number of states.

We now claim that U_M halts on all configurations in which $R_1 + R_{n+1} = x$ 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 $R_1 + R_{n+1} = x$, the latter invariant will be kept 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 halting will be reached. 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.

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

THEOREM 3.3. *From an (ordinary) counter machine M, one can compute a generalized 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 is independent of M.*

Proof. We first map M to U_M and then use a translation from counter machines to generalized Collatz functions based on [23], extended to represent enhanced CM instructions (in fact, there is only one such instruction in our machine). The translation of [23] (using an idea which goes back to Conway [12]) represents a machine state s by an integer $\widehat{s}$. Each register R_k is associated with a prime number p_k. The contents of R_k are encoded as the exponent of p_k in the prime-number factorization of $\widehat{s}$. Control states are represented in the same way, that is, state j is associated with a prime q_j; its exponent in $\widehat{s}$ is supposed to be 1 if the machine is in that state, and 0 otherwise. The modulus m is chosen as $\prod_i p_i \prod_j q_j$, which ensures that every remainder modulo m determines the control state and, for every register whether it is null. Thus there is a unique instruction corresponding to each remainder i (except for remainders which represent no valid state). For the "valid" remainders, the function $x \mapsto a_i(x-i)/m + b_i$ implements the corresponding instruction. To encode an instruction that increments R_k, a_i is chosen so that the exponent of p_k is incremented (specifically, a_i will be a multiple of $(p_k)^2$, since m includes a p_k factor). An enhanced instruction which adds d to R_k can be similarly encoded, choosing a_i as to increase the exponent of p_k by d.

Apart from the above, we reuse the translation of [23], and therefore it is unnecessary to repeat it in detail (in particular, we avoid the discussion of how "invalid remainders" are handled). The interested reader will enjoy reading its description in Kurtz and Simon's nice paper. In the next section, we present a reduction which deviates more significantly from Kurtz and Simon's, and so will be explicated more fully. $\square$

COROLLARY 3.4. *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 3.5. *There is a constant r such that global convergence and mortality over $\mathbb{Z}$ of piecewise affine functions f with integer coefficients and r regions are Π^0_2-complete problems.*

4 A Challenge: the Threshold Number of Regions

Section 2, along with the decidability of the one-dimensional case (Sect. 7), pinpoint the threshold for decidability of the problems considered in terms of dimensionality. In light of Section 3, it is natural to ask what the threshold is regarding the number of regions. One may expect to further develop a tradeoff between the number of regions and dimension, of the kind investigated for a number of undecidable problems, see [27]. At this stage, we only discuss the question of minimizing the number of regions (irrespective of dimension).

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. It seems natural to 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 are interested in 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, 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 (specifically, for the homogenous case) by Braverman [8].

For $\mathcal{R} = 2$ the problem has been recently shown to be undecidable [4]. There, the dimension was not bounded. In fact [4] reduces from mortality of counter machines, and the dimension of the dynamical system constructed is related to the size of the counter machine. Using Theorem 3.2, we can deduce that the undecidability result holds under a certain fixed bound on the dimension.

5 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 5.1. A *monic* piecewise affine function[3] 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 $\pm x_i \le d$.

The main point is that the definition of the function f does not allow for terms ax_j with $a \ne 1$ (or sums like $x_1 + x_2$)[4].

5.1 Two-counter machines

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

LEMMA 5.2. *2CM mortality is Π^0_2-complete.*

[3] which we abbreviate to "monic PAF'

[4] 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.

Proof. Blondel et al. point out that the standard translation of n-counter machines to 2CM preserves mortality. Combining this with the Π_2^0-completeness result of Simon and Kurtz, we obtain the lemma. $\square$

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

LEMMA 5.3. *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 normalized.

Proof. Arranging for (1) to hold is not hard to do 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). In fact, the deviation will, in general, be 0 or -1, except when the simulated register is null, where we may be forced to increase it to 1 in order to maintain the goal of always modifying the register.

To verify this idea in more detail, we present, in Table 1 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 $i^{(d)}$ with $d \in \{-1, 0, 1\}$ represents the deviation of the counter (that is, $d = 1$ means that the M' counter is one larger than the M counter, which we only use if M's counter is zero). Note that this invariant means that certain situations cannot arise in a simulation; specifically, a state $i^{(1)}$ cannot be entered unless the counter is null. To preserve mortality, we add instructions to map such 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)). $\square$

5.2 Compass Collatz-like functions

The proof will require a specially-adapted variant of the Collatz problem, presented next (the definition may seem a little forced; keep in mind that it is just an auxiliary tool in this work). In describing this kind of Collatz-like functions 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.*

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 1: modifying instructions to ensure the counter is always modified.

Definition 5.4. 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 $i = 0, \dots, m-1$, 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 $0 \leq i < p$):

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

The function could be represented as an ordinary generalized 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 visualize 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 5.5. 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 5.6. *CCP is Π^0_2-complete.*

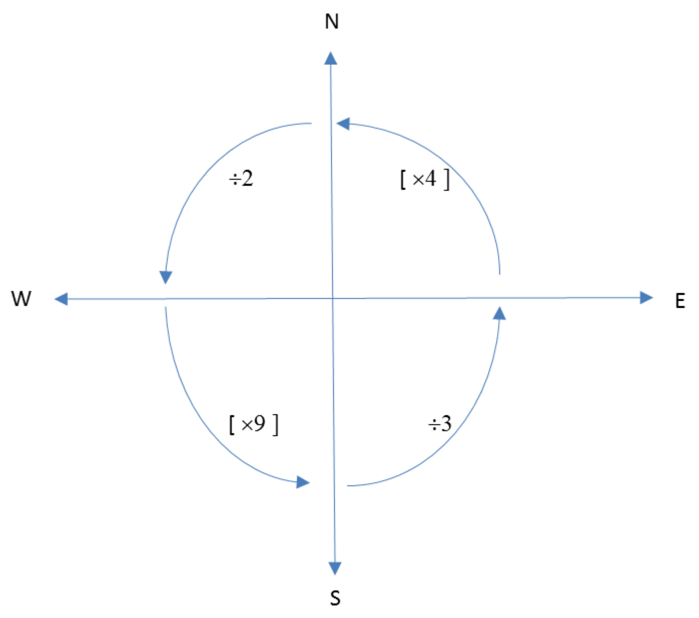

Figure 2: The dynamics of a Compass Collatz-like function.

Proof. The proof is a reduction from 2CM mortality. Given a normal 2-counter machine M (Lemma 5.3), 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 state $s = (i, \langle r_1, r_2 \rangle)$ 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 state to s.

Let $\widehat{s} = 2^{r_1} 3^{r_2} p + i = mx + rp + i$, where $0 \leq r < 6$. The remainder r determines whether each of r_1 and r_2 is zero or positive (since it is $2^{r_1} 3^{r_2} p \bmod 6$); hence, r and i determine the instruction $[i, b_1, b_2, D_1, D_2, k]$ 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 5.4; 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):

$$g(mx + rp + i, E) \quad = \begin{cases} (mx + rp + i, N) & D_1 = (-1) \\ (4(mx + rp) + i, N) & D_1 = (+1) \end{cases} \tag{3}$$

- 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:

$$g(mx + rp + i, W) \quad = \begin{cases} (mx + rp + k, S) & D_2 = (-1) \\ (9(mx + rp) + k, S) & D_2 = (+1)\,. \end{cases} \tag{4}$$

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 sent 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*, represented by a number of the form $(rp + 0, E)$ 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

$$g(mx + rp + 0, E) = (mx + rp + 0, N), \tag{5}$$

and

$$g(mx + rp + 0, W) = (mx + rp + 0, S) \tag{6}$$

(For the North and South axes the definition follows (2)).

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 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 (5) and (6), it will progress as follows:

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

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 $2^a3^bd'p$; 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 state 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 this state, 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. □

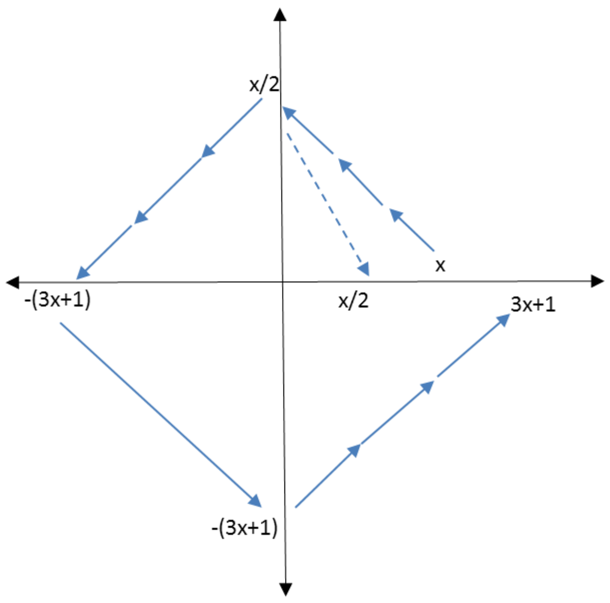

Figure 3: Simulating the $3x + 1$ function by a 2-dimensional monic PAF.

5.3 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 $(3x + 1, 0)$.

Here is the complete definition of the PAF:

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	$x<0$, $y<0$	$(x+1, y-1)$
South to East	$x \geq 0$, $y<0$	$(x+1, y+1)$
Final zone	$0 \leq x \leq 1$, $y=0$	(x, y)

Next, we handle the CCP in all generality.

LEMMA 5.7. *A CCP can be effectively reduced to mortality (or global convergence) of a monic PAF on $\mathbb{Z}^2$.*

Proof. Given a description of a Compass Collatz function g, our reduction produces the function f defined by Table 5.3, where for convenience every region has a label. Explanations follow; see also Figure 4. Note that the 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 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 maintains 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.

region label	constraints	$f(x,y)$
NW	$x \leq 0,\ y \geq 2m$	$(x-m,\ y-2m)$
$W^{\text{in}}_{r,i}$ $_{0\leq r<12,\ 0\leq i<p}$	$x \leq 0,\ y = rp+i > 0$	$(x-\lfloor r/2 \rfloor p - i,\ -\lfloor r/2 \rfloor p - i)$
$W^{\text{out}}_{r,i}$ $_{rp+i \in R_S}$	$x \leq -m,\ y = -rp-i$	$(0,\ x+i-w_{rp+i})$
$W^{\text{out}}_{r,i}$ $_{rp+i \notin R_S}$	$x \leq -m,\ y = -rp-i$	$(x+m,\ y-9m+rp+i-w_{rp+i})$
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 2: Simulating the Compass Collatz function by a 2-dimensional monic PAF.

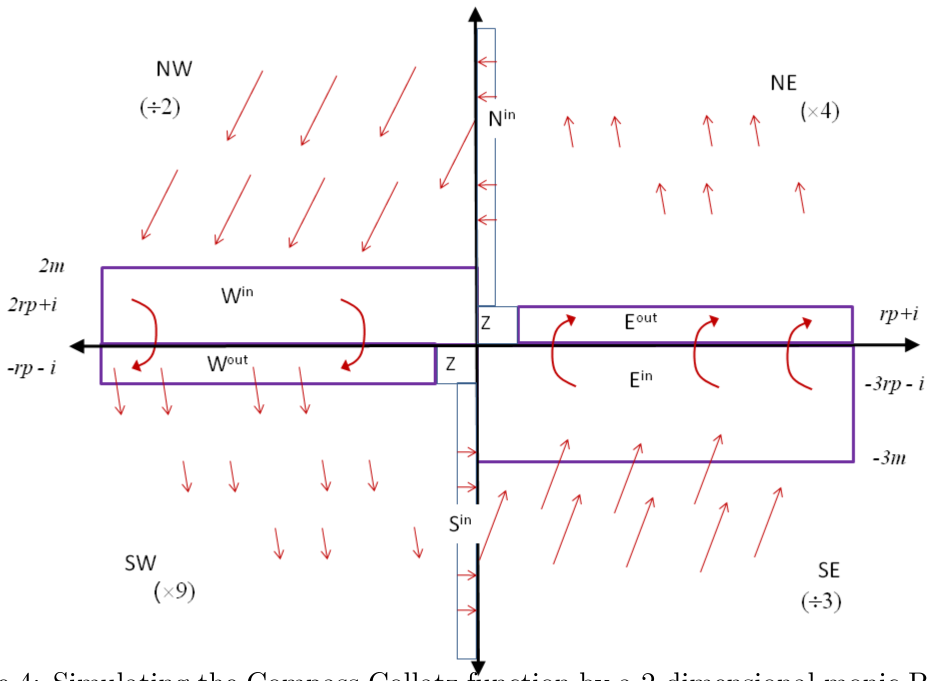

Figure 4: Simulating the Compass Collatz function by a 2-dimensional monic PAF.

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 $(mx + z, N)$.

   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 5.4, $g(c_N) = (\frac{1}{2}mx+\lfloor\frac{1}{2}r\rfloor p+i, W)$. If we write $g(c_N)$ as $(mx'+z', W)$, this compass point 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:

- Clearly, g is not simulated for points (x, Δ) with $x < m$. This is OK, since such points are the stopping condition for the CCP, and indeed they are mapped by our PAF to $(0, 0)$.
- 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.

□

Combining the lemmas, we obtain

THEOREM 5.8. *Global convergence and mortality of monic piecewise-affine functions over $\mathbb{Z}^2$ are Π^0_2-complete problems.*

Open Problems. In view of Section 3, 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^0_2 hard?

I have not been able to apply the method of Section 3 to this class of functions. But the fact that the $3x + 1$ problem is representable with 7 regions may be interpreted as evidence for the hardness of this problem, even with a constrained number of regions. We may also note that in three dimensions, Π^0_2 hardness follows quite easily by simulating an enhanced CM (Section 3), using the three coordinates to simulate the two counters and the program counter. In one dimension the problem is decidable (Section 7). It is, therefore, all the more intriguing that the situation in two dimensions seems so hard to settle.

One can also ask about an even smaller class of functions, specifically 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. What is the situation for functions over $\mathbb{Z}^n$ and the mortality problem?

5.4 An application to functions over the rationals

Recall that [5] proved that global convergence and mortality of piecewise-affine functions over $\mathbb{Q}^2$ are undecidable. As a corollary of the last theorem, we can obtain Π_2^0-completeness for their problems. Briefly, the reason is that results on *monic* functions are easy to transfer to the rationals. We need to slightly extend the class of functions, as follows.

Definition 5.9. A *quasi-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 $\{i_1, i_2\} = \{1, 2\}$, or $f(\vec{x}) = (0, 0)$. In addition, the halfspaces which define the space partition are given by inequalities of the form $\pm x_i \le d$.

THEOREM 5.10. *Global convergence and mortality of quasi-monic piecewise-affine functions over $\mathbb{Q}^2$ are Π_2^0-complete problems.*

Proof. We claim that the asymptotic behaviour of the function f constructed in Lemma 5.7, when converted to a quasi-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) equals an integer-valued trajectory from $(\lfloor x \rfloor, \lfloor y \rfloor)$, shifted by a fixed fractional part. The extension to the quasi-monic class has the role of allowing the definition on the region Z to be the constant $(0, 0)$, so that a trajectory that reaches Z is mapped to the origin, forgetting the fractional part. Table 5.4 gives the details.

□

6 Decidability for affine functions

Affine-linear transformations have been much studied in Dynamical System Theory, e.g., [17]. The setting there is that of a continuous state space, but the techniques are hardly affected by the restriction to $\mathbb{Z}^n$. I have chosen to include full proofs for the theorems below, however, as they are sufficiently short, at least 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 [35].

THEOREM 6.1. *There is an 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 $e_1, \ldots, e_n$ be the basis vectors. Suppose that for every e_i there is a k_i such that $A^{k_i} e_i = 0$. Let $k = \max(k_1, \ldots, k_n)$. Then, for general $\vec{x} = a_1 e_1 + \cdots + a_n e_n$, we have

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

region label	constraints	$f(x,y)$
NW	$x < 1,\ y \geq 2m$	$(x-m,\ y-2m)$
$W^{\text{in}}_{r,i}$ $0 \leq r < 12,\ 0 \leq i < p$	$x < 1,$ $0 < rp+i \leq y < rp+i+1$	$(x - \lfloor r/2 \rfloor p - i,\ y - \lfloor \frac{3}{2} r \rfloor p - 2i)$
$W^{\text{out}}_{r,i}$ $rp+i \in R_S$	$x \leq -m,$ $-rp-i \leq y < -rp-i+1$	$(0,\ x+i-w_{rp+i})$
$W^{\text{out}}_{r,i}$ $rp+i \notin R_S$	$x \leq -m,$ $-rp-i \leq y < -rp-i+1$	$(x+m,\ y-9m+rp+i-w_{rp+i})$
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 3: The definition of the monic PAF, rewritten for the rationals.

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 $A^n = \mathbf{0}$ ([35, Theorem 1.13]). Hence, this is all we have to check. □

THEOREM 6.2. *There is an 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

$$f^{(r)}(\vec{x}) = A^r\vec{x} + (A^{r-1} + \cdots + A + I)b = A^r\vec{x} + f^{(r-1)}(b)\,. \tag{7}$$

Suppose that f is mortal. Then, in particular, the sequence $b, f(b), f^{(2)}(b), \ldots$ 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 values 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 (7), $f^{(n)}(b) = f^{(n-1)}(b)$, 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 $b = 0$. □

Note that the decision procedures described in this section are clearly polynomial-time.

7 Decidability and Complexity in One Dimension

Blondel et al. prove that global convergence is decidable for $n = 1$ when the function is continuous. We prove the result for the integers, where continuity is irrelevant. Moreover we shall classify the problem's complexity.

The reader is invited to study the following examples and notice 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. Similarly, the second function has a fixed point at 4/3.

$$f_1(x) = \begin{cases} -2x+10 & x > 2 \\ 4 & x < 0 \\ 0 & x = 0 \\ 4-2x & 1 \le x \le 2 \end{cases} \tag{8}$$

$$f_2(x) = \begin{cases} x-3 & x > 2 \\ 4 & x < 0 \\ 0 & x = 0 \\ 4-2x & 1 \le x \le 2 \end{cases} \tag{9}$$

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.,

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

where each interval is associated with an affine function. 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 $a^+x + b^+$.

We may also assume $f(0) = 0$ since, for the convergence problem, the answer is negative if this does not hold, while for mortality, we may as well modify the function so that $f(0) = 0$.

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

$$\rho_0 = \max(\{R\} \cup \{f(x), \text{ where } L \le x \le R\})$$
$$\lambda_0 = \min(\{L\} \cup \{f(x), \text{ where } L \le x \le R\})$$

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 dynamic 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 7.1. *Suppose that at least one of a^+, a^- is non-negative. If either a^+ or a^- is bigger than 1; or $a^+ = 1$ and $b^+ \ge 0$; or $a^- = 1$ and $b^- \le 0$, then f is divergent, and not mortal. Otherwise, it has the finite attractor $A = [\lambda_0, \rho_0]$.*

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 $b^- \le 0$, 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

$$\forall x_0 \in \mathbb{Z} \, . \, \exists \, t > 0 \, .x_t = f^{(t)}(x_0) \in A \, . \tag{10}$$

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, (**??**) 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 $f(x) = x + b^+ < x$ on $[R, \infty)$. As long as $x_i > R$, we have $x_{i+1} < x_i$. Hence, the trajectory is a descending sequence which must reach a point in $(R + b^+, R]$. Note that by definition, $\lambda_0 \leq f(R) = R + b^+$, 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 $f(x_0) > \rho_0$ or $f(x_0) < \lambda_0$. In the first case, this means $a^+ x_0 + b^+ > \rho_0 \geq f(R) = a^+ R + b^+$, 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 will proceed as an ascending sequence which must reach a point in $[L, L + b^-)$, hence in A.
   (b) $a^- = 0$. In this cae, $f(f(x_0)) = b^- = f(L) \in A$.

□

THEOREM 7.2. *Global convergence and mortality over $\mathbb{Z}$ of a piecewise affine function f with integer coefficients are PSPACE problems.*

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 7.2 is applicable. By testing the conditions stated in the lemma, we either conclude immediately that f is not mortal, or get the attactor-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: here is a pseudo-code to convince the skeptic.

```
for x := $\lambda_0, \ldots, \rho_0$ 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"
```

□

It may be interesting to note that the decision procedure for the one-dimensional case in [5] is much simpler, and in fact polynomial-time (for rational coefficients in a standard representation). Keep in mind, however, that it solves a different problem (a continuous state space and a continuous function), which is neither a sub-problem nor a generalization of the problem we consider. In fact, for our problem, PSPACE turns out to be a tight classification. We shall prove that global convergence over $\mathbb{Z}$ of a piecewise affine function f with integer coefficients is PSPACE-hard. To this end, we next introduce several Turing-machine variants as intermediate representations, and show a sequence of reductions, starting with a standard PSPACE-hard problem and culminating in our mortality problem. First, we define a few machine models. The first machine is a restricted form of *linearly bounded automaton* (LBA) [40].

Definition 7.3. 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. The tape alphabet is 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 7.4. 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.

Definition 7.5. 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.

LEMMA 7.6. *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 7.7. *The computation of a tally LBA on input 0^n can be simulated by a fixed-space machine with oblivious queue access, starting on a blank queue of capacity $2n$.*

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 used to simulate both endmarkers, by setting them to 11 (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. $\square$

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 7.8. *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, and from that point on, faithfully simulate the given input-free machine. □

LEMMA 7.9. *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 complete 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 transitting 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.

□

THEOREM 7.10. *Global convergence over $\mathbb{Z}$ of a piecewise affine function f 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:

$$\langle q, w \rangle = q \cdot 2^n + w \,. \tag{11}$$

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

$$f(\langle q, bz \rangle) = \langle q', zb' \rangle \,. \tag{12}$$

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

$$f(\langle q, bz \rangle) = 0. \tag{13}$$

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 (11) 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,
   $x \in R_{q,b} \Rightarrow f(x) = 0.$

$\square$

Given Theorem 7.10, we can ask for the complexity of restrictions of the one-dimensional problem. Of course, many restrictions can be invented. By looking at the proof of Theorem 7.1, 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 5.8 and Corollary 3.5):

- *Open problem* 1. Consider one-dimensional integer piecewise affine functions defined by
  $$f(x) = x + b_i \quad \text{for } x \in H_i, \tag{14}$$
  where the sets $H_1, \ldots, H_p$ are an exhaustive partition of the $\mathbb{Z}$ 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?

8 Convergence to any Fixed Point

Definition 8.1. Function f is *globally convergent to a fixed point* if for every initial point x_0, the sequence $x_{t+1} = f(x_t)$ includes a fixed point of f.

Note that the loops that "iterate until reaching a fixed point" (not known in advance) are ubiquitous in Computer Science, so deciding the termination of such loops has an obvious interest.

THEOREM 8.2. Global convergence to a fixed point *of piecewise affine functions f with integer coefficients over $\mathbb{Z}^2$ is a Π_2^0-complete problem. Moreover, there is a constant such that the hardness result holds when the number of regions in the definition of f is bounded by that constant.*

Proof. We look back at the reductions used in Section 3 to prove Theorems 3.2 and 3.3. Their sum total is a reduction from the CM totality problem to mortality. We argue that the same reduction works for the above problem. This is entailed by the fact that the function constructed either globally converges to zero, or diverges unboundedly.

Indeed, if the given machine M halts for all x, the corresponding Collatz-like function g (according to Theorem 3.3) converges to 1, and the piecewise-affine function f constructed form it by the method of Theorem 2.4 converges to $(0, 0)$. In the complementary case, since M does not halt for a certain input x, the machine U_M of Theorem 3.2m when started at the configuration $(1, \langle x, 0, \dots, 0\rangle)$, performs infinitely many iterations from the reset state, each time setting W to a bigger number. Since this number is a component of the machine state, represented in the argument of f, these values are also unbounded. $\square$

THEOREM 8.3. Global convergence to a fixed point *of a piecewise affine function f with integer coefficients over $\mathbb{Z}$ is PSPACE-complete.*

Proof. The PSPACE algorithm illustrated in the proof of Theorem 7.1 is easily adapted to decide *global convergence to a fixed point*. For hardness, inspecting the reduction (Section 7) reveals that the only fixed point of the function constructed is the origin. $\square$

THEOREM 8.4. *There is a polynomial-time algorithm for deciding* global convergence to a fixed point *over $\mathbb{Z}^n$ of an affine function $f(x) = Ax + b$.*

Proof. We first consider the homogenous case, $f(x) = Ax$. The set of fixed points of f (in $\mathbb{Q}^n$) is the solution set of the equation $Ax = x$, or, equivalently, the eigenspace E_1 associated with the eigenvalue 1. For every sequence to converge to a fixed point, we want $A^r x$ to belong to E_1 for some r, for all $x \in \mathbb{Z}^n$. It is necessary and sufficient to have $A^{r_i} e_i \in E_1$ for the vectors e_i of the standard basis, and then it follows that there is a k such that $A^k(\mathbb{Z}^n) \subseteq E_1$. If such a k exists, then there is such a $k \le n$ (apply [35, Theorem 1.13] to a matrix representing the effect of A on $E_1^\perp$). We conclude that it suffices to check whether $A^n e_i \subseteq E_1$ for all e_i.

Next, let $f(x) = Ax + b$ for any b. Suppose that f globally converges to a fixed point. Thus for any initial value x_0, the sequence $x_{t+1} = f(x_t)$ reaches a fixed point.

This will also be true for the sequence of differences $x_t - x'_t$ obtained from two initial points x_0 and x'_0 (note that the limit difference may be non-zero, if there are many fixed points). By the definition of the sequence,

$$x_{t+1} - x'_{t+1} = A(x_t - x'_t) .$$

Thus global convergence to a fixed point of f requires global convergence to a fixed point of $g(x) = Ax$, which we know how to verify. Suppose that g satisfies the requirement; then, letting $x'_0 = 0$, we see that the sequence from x_0 converges if and only if the sequence from 0 converges. This latter sequence is $x'_{t+1} = (A^t + A^{t-1} + \cdots + I)b$, and the equation $x'_{t+1} = x'_t$ implies that $A^t b = 0$ for some t. This is, again, the case if and only if $A^n b = 0$. □

For dessert, here is another variant, for which essentially the same results and techniques apply, except that in the one-dimensional case, the problem is PTIME and not PSPACE-complete. Verifying this is proposed as an exercise to the energetic reader.
Convergence to a Finite Set (In Dynamical System parlance: *Finite Attractor.* In Program Analysis parlance: *Finite Tail-Invariant*).

Input: f, as previously. Question: is there a finite set S such that every sequence $x_{t+1} = f(x_t)$ stays within S for all t large enough?

9 Conclusion

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 [37, 8, 4] and on the analysis of the mortality problem in a continuous domain [5]. 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.

We have shown that mortality, convergence (to zero) and *convergence to any fixed point* are undecidable in general for iterated piecewise-affine functions. 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 focus of this work is on deciding global properties of systems, so we have not dwelled 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 was mentioned as an open problem in [22, 5]. For affine 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 for linear transformations over the rationals in [20].

I 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. There is certainly much terrain yet to be explored in this zone of interaction of Dynamical System Theory and discrete computation. 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^0_2 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?

Another famous open problem which seems related is known as *Skolem's problem* (and sometimes *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, the problem is to decide whether the sequence $(M^i u)_{i\geq 0}$ ever hits a given hyperplane $a \cdot x = 0$. Vyalyi and Tarasov [39] 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.* Chonev et al. [**?**] give decidability and complexity results for the problem in case that the "chamber" is a linear subspace of dimension up to 3. See also the survey [**?**].

Peter van Emde Boas and Marek Karpinski [38] describe the "mouse in an octant" problem, attributed to Lothar Budach, a problem which has the flavour of a Collatz-like problem (intuitively, closer to our CCP than to Conway's GCP), and whose decidability status is apparently still open (see also [32, **?**]).

Finally, a possible direction for extending this study is to look at polynomials and piecewise-polynomial functions.

Acknowledgement

I thank Jakob Grue Simonsen for a good discussion, prompting me to extend and refine my initial results.

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] 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.

[3] Amir M. Ben-Amram and Samir Genaim. On the linear ranking problem for integer linear-constraint loops. In *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM press, 2013.

[4] Amir M. Ben-Amram, Samir Genaim, and Abu Naser Masud. On the termination of integer loops. *ACM Transactions on Programming Languages and Systems*, 2012. to appear.

[5] 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.

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

[7] 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.

[8] Mark Braverman. Termination of integer linear programs. In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA*, volume 4144 of *Lecture Notes in Computer Science*, pages 372–385. Springer, 2006.

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

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

[11] 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.

[12] John. H. Conway. Unpredictable iterations. In *Proc. 1972 Number Theory Conf., Univ. Colorado, Boulder*, pages 49–52. 1972. Reprinted with historical commentary in [24].

[13] 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 [24].

[14] 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.

[15] 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 (Volume I), pages 374–428. Springer-Verlag (New York), Dagstuhl, Germany, September 1996, revised paper 1998.

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

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

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

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

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

[21] František Kascák. 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.

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

[23] 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 *Lecture Notes in Computer Science*, pages 542–553. Springer, 2007.

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

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

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

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

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

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

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

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

[32] Aleš Pultr and Josef Úlehla. On two problems of mice. *Proceedings of the 10th Winter School on Abstract Analysis*, pages 249–262, 1982.

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

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

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

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

[37] 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.

[38] 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.

[39] 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.

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

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