INFORMATIQUE THÉORIQUE ET APPLICATIONS

KEIJO RUOHONEN

Undecidable event detection problems for ODEs of dimension one and two

Informatique théorique et applications, tome 31, nº 1 (1997), p. 67-79.

<http://www.numdam.org/item?id=ITA_1997__31_1_67_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 31, n° 1, 1997, pp. 67-79)

UNDECIDABLE EVENT DETECTION PROBLEMS FOR ODES OF DIMENSION ONE AND TWO (*)

Keijo RUOHONEN ([1])

Communicated by XAFFRAY

Abstract. – *The ability of dynamical systems of various kinds to simulate Turing machines and thus manifest a universal computation power (and beyond) has gathered a lot of interest lately,* see *e.g.* [16], [5], [6] *and* [4]. *A similar line of investigation for ordinary differential equations was started in* [11] *and continued in* [12] *and* [13]. *In this context the minimum dimension required for universal computation is of interest. The dynamical systems in* [5] *and* [6] *are of small dimension and the topic of* [4] *is to find the smallest dimension for certain types of dynamical systems. The results in this paper show that for ODEs dimension two can be reached and, allowing somewhat complicated events, even dimension one.*

Résumé. – *La capacité de systèmes dynamiques de différents types de simuler les machines de Turing et ainsi de posséder une puissance de calcul universel (et au-delà) a suscité beaucoup d'intérêt récemment, voir par exemple* [16], [5], [6] *et* [4]. *Une direction de recherche semblable sur les équations différentielles ordinaires a été amorcée en* [11] *et poursuivie en* [12] *et* [13]. *Connaître la dimension minimale exigée pour obtenir la puissance de calcul universel revêt un intérêt particulier. Les systèmes dynamiques de* [5] *et* [6] *sont de petite dimension et le sujet de* [4] *est de trouver la plus petite dimension pour certains types de système dynamiques. Les résultats de cet article montrent que pour les équations différentielles ordinaires la dimension deux est suffisante et même, si des événements légèrement plus compliqués sont autorisés, que l'on peut descendre en dimension un.*

1. INTRODUCTION

Many problems involving finitely given ordinary differential equations (ODEs) turn out to be algorithmically undecidable, something that probably has not been sufficiently appreciated. It is true that the classical noncomputability result of M.B. Pour-El and I. Richards [8] involves nonunique solutions and that these are not of much practical interest. There are however other undecidable problems, the event detection problems, which involve unique solutions and explicitly given ODEs. Indeed, event detection

(*) Received September 1995.

([1]) Department of Mathematics, Tampere University of Technology, 33101 Tampère, Finland.

is dynamically undecidable, i.e., the ODE simulates dynamically the steps of a Turing machine computation and its definition contains only the transition rules of the Turing machine and not results of whole computations, *see* [11].

The purpose of this paper is to extend the dynamical undecidability of event detection to ODEs of small dimension (that is, small number of dependent variables). This objective is of interest because the dimensions of the ODEs in [11],[12] and [13] are rather high. Although it is possible to get lower dimensions by using different types of machines in the simulation, to reach dimensions one and two requires separate constructs. These constructs are the subject of the present paper.

There has also been a recent interest in Turing machine simulation by low-dimensional dynamical systems, and the present paper may be considered as a contribution to this line of research, *see* e.g. [4],[5] and [6]. The conclusions reached here are similar to those in [4]: A rather natural two-dimensional dynamically undecidable event detection problem exists but to get to dimension one a much more complicated construct is needed. Somehow the smallest natural dimension for dynamical computation appears to be two, getting to dimension one strains naturality a lot.

Only some basic facts of computability, computable analysis and classical ODE theory are used. A good background is contained in [3],[9] and in [2] or [10].

2. PRELIMINARIES

An *event* of an ODE $\mathbf{y}' = \mathbf{f}(\mathbf{y}, t)$, with the initial value $\mathbf{y}(0) = \mathbf{y}_0$, occurs whenever at least one of the given equations

$$g_j(t, \mathbf{y}(t), \mathbf{y}'(t)) = 0 \ (j = 1, \dots, k)$$

is satisfied for some t in a given interval I. For aspects of numerical event detection see [15]. The *event detection problem* (*EDP*) is the problem of deciding for a given initial value problem and event on an interval I whether or not the event occurs.

If a quite general approach is taken then it is not very difficult to obtain low-dimensional undecidable event detection problems. Indeed, take a universal Turing machine $\mathcal{M}$ with nonnegative integer inputs and define the sequence $f_0, f_1, \dots$ of rationals by

$$f_n = \begin{cases} 2^{-m} & \text{if } \mathcal{M} \text{ stops in } m \text{ steps after receiving input } n \\ 0 & \text{if } \mathcal{M} \text{ does not stop after receiving input } n. \end{cases}$$

Following the nomenclature of [9], the sequence $f_0, f_1, \ldots$ is then not a computable sequence of rationals but it is a computable sequence of reals as it can be approximated by the computable double sequence f_{nk} $(n = 0, 1, \ldots; k = 0, 1, \ldots)$ of rationals where

$$f_{nk} = \begin{cases} 2^{-m} & \text{if } \mathcal{M} \text{ stops in } m \leq k \text{ steps after receiving input } n \\ 0 & \text{if } \mathcal{M} \text{ does not stop in } k \text{ steps after receiving input } n. \end{cases}$$

(A sequence of rationals is computable if there is an algorithm which on input n (resp. (n, k)) computes the denominator, the numerator and the sign of the nth (resp. the (n, k)th) term in the sequence. A sequence $x_0, x_1, \ldots$ of reals is computable if it can be approximated by a computable double sequence r_{nk} $(n = 0, 1, \ldots; k = 0, 1, \ldots)$ of rationals uniformly in n, i.e., $|x_n - r_{nk}| \leq 2^{-k}$ for all n and k.) Now, detection of the event $y(t) = 0$ for the ODE $y' = 0$, given n and the initial value $y(0) = f_n$, is undecidable on any interval containing 0, because $f_n = 0$ is undecidable. A further modification is obtained as follows. Define the smooth function

$$g(x) = f_{\lfloor x+1/2 \rfloor} e^{-\tan^2 \pi x}.$$

It is easy to see that g is computable on $[0, \infty)$. Detection of the event $y_1(t) = 0$ for the ODE

$$\begin{cases} y_1' = g(y_2) - 1 \\ y_2' = 0 \end{cases}$$

given an initial value $y_1(0) = 1, y_2(0) = n$ where n is a nonnegative integer is then undecidable on $[0, 1]$. There is a similar construct giving undecidability of two-dimensional symbolical event detection described in [11].

EDP is dynamically undecidable for time intervals of the form $[0, T)$ and $[0, \infty)$, as was shown in [11] through dynamical simulation of a universal Turing machine by an explicitly given ODE. Indeed, in [11] initial values $\mathbf{y}_0$ are n-tuples of nonnegative integers, $\mathbf{f}$ is a fixed explicitly given function and the event to be detected is of the simple form $y_i(t) = 0.5$. (No references to computability of reals or functions, or properties of symbolical expressions actually appear in [11].) Moreover, the solutions are computable. Even a smooth choice for $\mathbf{f}$ is possible. Extensions of the undecidability to parametric ODEs and to closed finite time intervals are given in [12] and [13].

The ODE used in [11] has a large dimension (that is, number of dependent variables). Reduction of the number of dependent variables depends heavily on the internal structure of the ODE. For this purpose some characteristics of the construct in [11] are given.

The central idea of [11] is to simulate a counter machine $\mathcal{M}$ with m counters and counter input (and no internal states) by a $2m+1$-dimensional autonomous ODE

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t))$$

with an initial value at $t = 0$. As is well known, counter machines have universal computing power. The following properties of this simulation will be needed:

(A) The simulation of the i-th step of the computation of $\mathcal{M}$ takes place in two stages, the first stage in the time interval $2i - 2 \leq t < 2i - 1$ and the second in $2i - 1 \leq t < 2i$.

(B) Two copies of counters of $\mathcal{M}$ are kept, the first in $q_1, \ldots, q_m$ and the second in $q_{m+1}, \ldots, q_{2m}$, giving the counts of symbols in the counters. The state q_{2m+1} is the time t (whence $Q_{2m+1} = 1$).

(C) During the first stage of simulation counter transition of $\mathcal{M}$ is performed on $q_1, \ldots, q_m$ using $q_{m+1}, \ldots, q_{2m+1}$, and $q_{m+1}, \ldots, q_{2m}$ will remain unchanged. During the second stage the states $q_{m+1}, \ldots, q_{2m}$ are updated using $q_1, \ldots, q_m, q_{2m+1}$, and $q_1, \ldots, q_m$ remain unchanged.

(D) Q_j is of the form $Q_j(\mathbf{q}) = P_j(q_{m+1}, \ldots, q_{2m})s(t)$ $(j = 1, \ldots, m)$ where the value of P_j is -1, 0 or 1,

$$\int_{2i-2}^{2i-1} s(t)dt = 1$$

and $s(t)$ is zero during the second stage. Denote for brevity

$$\mathbf{P} = (P_1, \ldots, P_m).$$

$\mathbf{P}$ and s are smooth, and so is $\mathbf{q}(t)$.

(E) If the input count of $\mathcal{M}$ is n then the initial value is

$$\mathbf{c} = (0, \ldots, 0, n, 0, \ldots, 0, n, 0)$$

where the n's are at the m-th and the $2m$-th positions.

(F) When $\mathcal{M}$ halts at the ith step, then the value of q_1, which hitherto has been 0, is raised to 1 during $2i - 2 \leq t \leq 2i - 1$, and will stay there for $t \geq 2i - 1$. The ODE does not "halt." Halting of $\mathcal{M}$ is thus signalled by the event $q_1(t) = 0.5$ which is undecidable.

Structural properties and explicit construction of $\mathbf{Q}$ are given in [11] (Theorem 1 and its proof).

3. ODEs OF DIMENSION TWO

During the first stage the values of $q_{m+1}, \ldots, q_{2m}$ are nonnegative integers. These are coded in the value of the second dependent variable z_2. Similarly during the second stage the values of $q_1, \ldots, q_m$ are nonnegative integers and these are coded in the value of the first dependent variable z_1. In both cases the coding scheme is the same so only z_2 is treated in what follows. To describe the scheme take m nonnegative integers $i_1, \ldots, i_m$. If $q_1 = i_1$ then the value of the variable z_2 is in the interval

$$\sum_{j=1}^{i_1} 2^{-j} \leq z_2 < \sum_{j=1}^{i_1+1} 2^{-j} \text{ i.e. } 1 - 2^{-i_1} \leq z_2 < 1 - 2^{-i_1-1}.$$

Similarly, if in addition $q_2 = i_2$ then the value z_2 is in the interval

$$1 - 2^{-i_1} + 2^{-i_1-1}(1 - 2^{-i_2}) \leq z_2 < 1 - 2^{-i_1} + 2^{-i_1-1}(1 - 2^{-i_2-1}),$$

and, in general, if $q_1 = i_1, \ldots, q_l = i_l$ then

$$\sum_{j=1}^{l} 2^{-i_1-\cdots-i_{j-1}-j+1}(1 - 2^{-i_j}) \leq z_2 < \sum_{j=1}^{l-1} 2^{-i_1-\cdots-i_{j-1}-j+1}(1 - 2^{-i_j})$$
$$+ 2^{-i_1-\cdots-i_{l-1}-l+1}(1 - 2^{-i_l-1}) \ (l = 1, \ldots, m).$$

To put it in another way, if $q_1 = i_1, \ldots, q_l = i_l$ then the binary representation of z_2 is of the form

$$z_2 = 0\,.\underbrace{1\cdots1}_{i_1 \text{ bits}}0\underbrace{1\cdots1}_{i_2 \text{ bits}}0\cdots0\underbrace{1\cdots1}_{i_l \text{ bits}}.$$

It is assumed that binary representations containing only finitely many 0's are not allowed. Thus, in general, any number z in the interval $[0, 1)$ has the binary representation

$$z = \sum_{j=1}^{\infty} 2^{-i_1-\cdots-i_{j-1}-j+1}(1 - 2^{-i_j}) = 0\,.\underbrace{1\cdots1}_{i_1 \text{ bits}}0\underbrace{1\cdots1}_{i_2 \text{ bits}}0\cdots0\underbrace{1\cdots1}_{i_j \text{ bits}}\cdots$$

corresponding to the infinite integer sequence $i_1, i_2, \ldots$. Denote then

$$g(z) = \lfloor -log_2(1 - z) \rfloor,$$

$$k(z) = 2(1 - (1 - z)2^{g(z)})$$

and

$$g_j(z) = g(k^{j-1}(z)) \; (j = 1, 2, \ldots)$$

where k^j denotes j-fold composition power of k. See Figures 1-4.

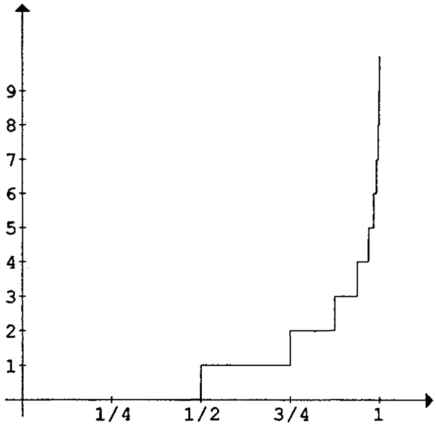

Figure 1. – The graph of $g(z)$.

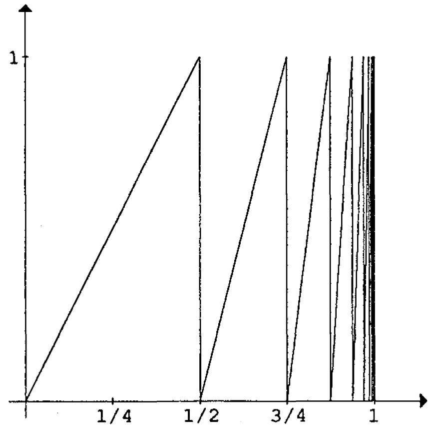

Figure 2. – The graph of $k(z)$.

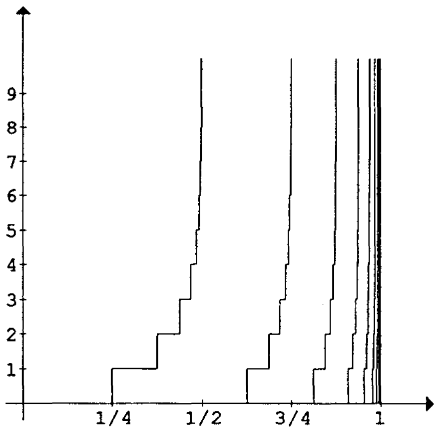

Figure 3. – The graph of $g(k(z))$.

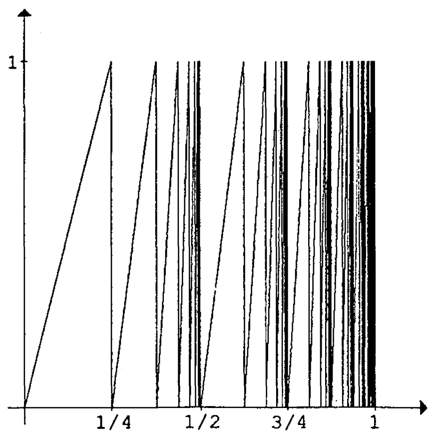

Figure 4. – The graph of $k(k(z))$.

LEMMA 1: $i_j = g_j(z)$ $(j = 1, 2, \ldots)$

Proof: The lemma is proved by induction on j. The case $j = 1$ is immediate. To proceed by induction it is first shown that if the integer sequence corresponding to z is $i_1, i_2, \ldots$ then the integer sequence corresponding to $k(z)$ is $i_2, i_3, \ldots$. Indeed if $g(z) = i_1$ and

$$z = \sum_{j=1}^{\infty} 2^{-i_1 - \cdots - i_{j-1} - j + 1}(1 - 2^{-i_j}) = 0\,.\underbrace{1\cdots1}_{i_1 \text{ bits}}0\underbrace{1\cdots1}_{i_2 \text{ bits}}0\cdots0\underbrace{1\cdots1}_{i_j \text{ bits}}\cdots$$

then

$$\begin{aligned} k(z) &= 2(1 - (1 - z)2^{i_1}) = 2^{i_1+1}(2^{-i_1} - 1 + z) \\ &= \sum_{j=2}^{\infty} 2^{-i_2 - \cdots - i_{j-1} - j + 2}(1 - 2^{-i_j}) = 0\,.\underbrace{1\cdots1}_{i_2 \text{ bits}}0\underbrace{1\cdots1}_{i_3 \text{ bits}}0\cdots0\underbrace{1\cdots1}_{i_j \text{ bits}}\cdots. \end{aligned}$$

It is easily checked that if $0 \leq z < 1$ then also $0 \leq k(z) < 1$. Replacing z by $k(z)$ repeatedly one sees inductively that the integer sequence corresponding to $k^{j-1}(z)$ is $i_j, i_{j+1}, \ldots$ whence $g_j(z) = g(k^{j-1}(z)) = i_j$. $\square$

It follows from Lemma 1 that $g_j(z_2) = i_j$ $(j = 1, \ldots, m)$ which is denoted simply by $\mathbf{i} = \mathrm{g}(z_2)$. By $h(\mathbf{i})$ a "typical" value of z_2 corresponding to the integers $i_1, \ldots, i_m$ is denoted and it is chosen to be the midpoint of the particular interval, *i.e.*,

$$h(\mathbf{i}) = \sum_{j=1}^{m-1} 2^{-i_1 - \cdots - i_{j-1} - j + 1}(1 - 2^{-i_j}) + 2^{-i_1 - \cdots - i_{m-1} - m}(2 - 3 \cdot 2^{-i_m - 1}).$$

Thus the binary representation of $h(\mathbf{i})$ is

$$h(\mathbf{i}) = 0\,.\underbrace{1\cdots1}_{i_1 \text{ bits}}0\underbrace{1\cdots1}_{i_2 \text{ bits}}0\cdots0\,\underbrace{1\cdots1}_{i_m \text{ bits}}\,0\,1.$$

and, by Lemma 1, $g(h(\mathbf{i})) = \mathbf{i}$. Now, to move the value of z_1 from one "typical" value to the next during first stage and using the value of z_2 one takes

$$\frac{dz_2}{dt} = (h(\mathrm{g}(z_2) + \mathbf{P}(\mathrm{g}(z_2))) - h(\mathrm{g}(z_2)))s(t) \text{ and } z_1(0) = h(0, \ldots, 0, n).$$

(The functions s and $\mathbf{P}$ appeared in property **(D)** in the previous section.) To update the value z_2 during second stage one then simply uses "dragging":

$$\frac{dz_2}{dt} = \frac{3}{2}\sqrt[3]{z_1 - z_2}\, s(t+1) \text{ and } z_2(0) = h(0, \ldots, 0, n).$$

During the operation z_2 starts from some value within $z_1 \pm 1$ and within a unit interval of time is dragged to the value z_1 where it stays. There are of course several ways to get such a dragging operation.

As in the case of the original ODE, if a nonreversible counter machine is simulated, then the solutions are necessarily nonunique in the backward direction and this nonuniqueness appears during the dragging. Indeed, whenever a counter machine configuration has several possible predecessors, information of which one of them actually appeared will be lost after the dragging. However, if a reversible counter machine is simulated then the present configuration uniquely determines the previous one and dragging can be replaced by a reversible operation similar to the one for z_1 and a unique solution is obtained. It should be mentioned that reversible counter machines can simulate reversible Turing machines, *see* [11], and the latter are known to be computationally universal (*see* [7] or [11]). In any case, as in [11], the solutions $z_1(t)$ and $z_2(t)$ are smooth and forward unique.

The behaviour of the original ODE

$$\frac{d\mathbf{q}}{dt} = \mathbf{Q}(\mathbf{q}(t))$$

is now faithfully emulated and occurrence of the event $z_1 = 0.5$ is undecidable. Of course the infinite time interval $[0, \infty)$ can be replaced by a finite half open one, say the interval $[0, \pi/2)$ obtained via the change of variable $t = \tan u$. Thus the following theorem is proved.

THEOREM 1: *There exists an explicitly given ODE pair*

$$\begin{cases} y_1' = f_1(y_1, y_2, t) \\ y_2' = f_2(y_1, y_2, t) \end{cases}$$

for which EDP is undecidable in the time interval $[0, \infty)$ *(or in a finite half open time interval* $[0, T)$*). The solutions* $y_1(t)$, $y_2(t)$ *are forward unique and smooth (and can be chosen to be unique at the expense of getting much more complicated* f_1 *and* f_2*).* $\square$

The right hand side of the ODE in Theorem 1 is bounded but discontinuous. To see this consider the fact that for $\mathbf{i} = \mathbf{i}_1 = (0, n, 0, i_4, \ldots, i_m)$ and $\mathbf{i} = \mathbf{i}_2 = (0, n, 1, i_4, \ldots, i_m)$ the typical values $h(\mathbf{i}_1)$ and $h(\mathbf{i}_2)$ can be arbitrarily close and yet the values of $h(\mathbf{i}_1 + \mathbf{P}(\mathbf{i}_1)) - h(\mathbf{i}_1)$ and $h(\mathbf{i}_2 + \mathbf{P}(\mathbf{i}_2)) - h(\mathbf{i}_2)$ may differ considerably depending on whether or not the value of i_1 is raised from 0 to 1. Let $K(x)$ be defined by

$$\begin{cases} e^{1-1/x} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0, \end{cases}$$

and take the smooth sigmoid

$$L(x) = 1 - K(1 - K(x)).$$

Then a somewhat "more continuous" replacement of $g_l(z)$ would be e.g. $d_l(z) = d(k^{l-1}(z))$ where

$$d(z) = \lfloor -\log_2(1 - z) \rfloor + L(2^m k(z)) - 1$$

(*see* Figures 5 and 6). It remains an open problem whether Theorem 1 is valid for continuous f_1 and f_2.

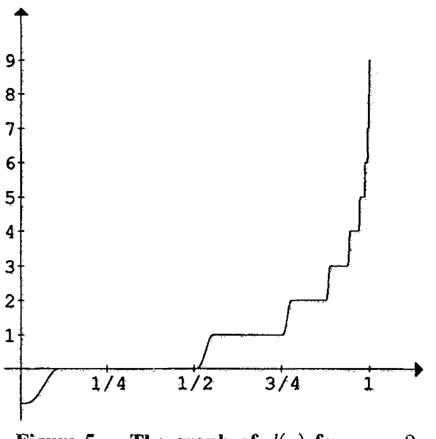

Figure 5. – The graph of $d(z)$ for $m = 2$

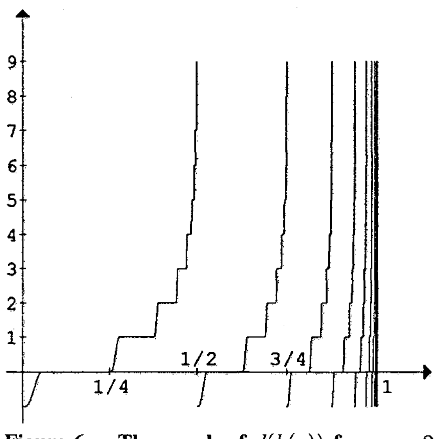

Figure 6. – The graph of $d(k(z))$ for $m = 2$

4. ODEs OF DIMENSION ONE

It is possible to still reduce dimension in the construct of the previous section. First it is noted that if $\mathbf{i}_1$ and $\mathbf{i}_2$ contain the counter counts of two computationally consecutive steps of the counter machine to be simulated then the sums of elements of $\mathbf{i}_1$ and $\mathbf{i}_2$ differ by at most m. Let then

$$\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_p$$

be a sequence of computationally consecutive counts where the initial count $\mathbf{i}_1$ is $(0, \ldots, 0, n)$. Using typical values this can be coded as the number

$$\begin{aligned} b(\mathbf{i}_1, \ldots, \mathbf{i}_p) &= \sum_{j=1}^{p} 2^{-(m+n+1)-(2m+n+1)-\cdots-((j-1)m+n+1)} h(\mathbf{i}_j) \\ &= \sum_{j=1}^{p} 2^{-l(n,j-1)} h(\mathbf{i}_j) \end{aligned}$$

where

$$l(n, j) = j(n + 1 + \frac{1}{2}m(j-1)).$$

Denote further

$$c(n, p, z) = 2^{l(n,p)} z - \lfloor 2^{l(n,p)} z \rfloor.$$

LEMMA 2: *If z is in the interval $b(\mathbf{i}_1, \ldots, \mathbf{i}_p) \leq z \leq b(\mathbf{i}_1, \ldots, \mathbf{i}_{p+1})$, then* $\mathbf{g}(c(n, p-1, z)) = \mathbf{i}_p$.

Proof: If $b(\mathbf{i}_1, \ldots, \mathbf{i}_p) \leq z \leq b(\mathbf{i}_1, \ldots, \mathbf{i}_{p+1})$, i.e.,

$$\sum_{j=1}^{p} 2^{-l(n,j-1)} h(\mathbf{i}_j) \leq z \leq \sum_{j=1}^{p+1} 2^{-l(n,j-1)} h(\mathbf{i}_j),$$

then

$$\sum_{j=1}^{p-1} 2^{-l(n,p-1)-l(n,j-1)} h(\mathbf{i}_j) + h(\mathbf{i}_p) \leq 2^{l(n,p-1)} z$$

$$\leq \sum_{j=1}^{p-1} 2^{-l(n,p-1)-l(n,j-1)} h(\mathbf{i}_j) + h(\mathbf{i}_p) + 2^{l(n,p-1)-l(n,p)} h(\mathbf{i}_{p+1})$$

and

$$\lfloor 2^{l(n,p-1)} z \rfloor = \sum_{j=1}^{p-1} 2^{l(n,p-1)-l(n,j-1)} h(\mathbf{i}_j).$$

So

$$h(\mathbf{i}_p) \le 2^{l(n,p-1)}z - \lfloor 2^{l(n,p-1)}z \rfloor \le h(\mathbf{i}_p) + 2^{-pm-n-1}h(\mathbf{i}_{p+1})$$

and, by Lemma 1,

$$\mathbf{g}(c(n, p-1, z)) = \mathbf{g}(2^{l(n,p-1)}z - \lfloor 2^{l(n,p-1)}z \rfloor) = \mathbf{i}_p. \ \square$$

Thus one gets the initial value problem faithfully simulating the one used in the previous section:

$$\frac{dz}{dt} = 2^{-l(n,\lfloor t \rfloor+1)}h(\mathbf{g}(c(n, \lfloor t \rfloor, z)) + \mathbf{P}(\mathbf{g}(c(n, \lfloor t \rfloor, z)))),$$

$$z(0) = h(\mathbf{i}_1) = h(0, \ldots, 0, n).$$

The values of z start from $h(\mathbf{i}_1)$ and then move piecewise linearly through the values

$$b(\mathbf{i}_1) = h(\mathbf{i}_1), b(\mathbf{i}_1, \mathbf{i}_2), b(\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3), \ldots$$

attained at $t = 0, 1, 2, \ldots$. The event to be detected is

$$g(c(n, \lfloor t \rfloor, z)) = 1.$$

The computational history is preserved in the value of z, so the value n, given in the initial value $z(0)$, can be recalled at any time during the simulation. Indeed, by Lemma 1,

$$n = g_m(z).$$

Thus the initial value problem aimed at here is

$$\frac{dz}{dt} = 2^{-l(g_m(z),\lfloor t \rfloor+1)}h(\mathbf{g}(c(g_m(z), \lfloor t \rfloor, z)) + \mathbf{P}(\mathbf{g}(c(g_m(z), \lfloor t \rfloor, z)))),$$

$$z(0) = h(\mathbf{i}_1) = h(0, \ldots, 0, n)$$

and the event is

$$g(c(g_m(z), \lfloor t \rfloor, z)) = 1.$$

There is no need for the dragging used in the previous section, as the history of the computation is preserved. Thus the solution $z(t)$ is unique. It might be noted that the idea of preserving computational history to obtain reversible computing is an old one. It appears first in the work of Y. Lecerf [7] and then independently in [1]. The construct used in [11] is from [14].

A smooth solution $z(t)$ is obtained when the change of variable

$$t = a \int_0^u K(\sin^2 \pi w) dw$$

is made where K is given in the previous section and the constant a is chosen to satisfy

$$a \int_0^1 K(\sin^2 \pi w) dw = 1.$$

Note that then $\lfloor t \rfloor = \lfloor u \rfloor$. Again the infinite time interval $[0, \infty)$ can be replaced by a finite half open one. Thus the following theorem is proved.

THEOREM 2: *There exists an explicitly given ODE $y' = f(y, t)$ for which the EDP is undecidable in the time interval $[0, \infty)$ (or in a finite half open time interval $[0, T)$). The solution $y(t)$ is unique and smooth.* □

Note that the price to be paid for getting Theorem 2 is the complicated structure of the event that must be used. As is easily seen, the event can be made simpler by adding one more dependent variable, and Theorem 1 follows.

The function f in the theorem is discontinuous, for the same reason as that in Theorem 1.

REFERENCES

1. C. H. BENNETT, Logical Reversibility of Computation. *IBM J. Res. Dev. 17*, 1973, pp. 525-523.
2. P. HARTMAN, *Ordinary Differential Equations.* Birkhäuser, 1982.
3. J. E. HOPCROFT and J. D. ULLMAN, *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley, 1979.
4. P. KOIRAN, P. COSNARD and M. GARZON, Computability with Low-Dimensional Dynamical Systems. *Theoret. Comput. Sci. 132*, 1994, pp. 113-128.
5. C. MOORE, Unpredictability and Undecidability in Dynamical Systems. *Phys. Rev. Lett. 64*, 1990, pp. 2354-2357.
6. C. MOORE, Generalized Shifts: Unpredictability and Undecidability in Dynamical Systems. *Nonlinearity 4*, 1991, pp. 199-230.
7. M. Y. LECERF, Machines de Turing réversibles. Récursive insolubilité en $n \in N$ de l'équation $u = \theta^n u$, où θ est un "isomorphisme de codes". *Comptes Rendus 257*, 1963, pp. 2597-2600.

8. M. B. Pour-El and I. Richards, A Computable Ordinary Differential Equation which Possesses No Computable Solutions. *Ann. Math. Logic 17*, 1979, pp. 61-90.
9. M. B. Pour-El and I. Richards, *Computability in Analysis and Physics.* Springer-Verlag, 1989.
10. E. O. Roxin, *Ordinary Differential Equations.* Wadsworth, 1972.
11. K. Ruohonen, Undecidability of Event Detection for ODEs. *J. Inform. Proc. Cybern. EIK*, *29*, 1993, pp. 101-113.
12. K. Ruohonen, Event Detection and Nonrecursive Hierarchies. *Results and Trends in Theoretical Computer Science* (J. Karhumäki, H. Maurer and G. Rozenberg, Eds.). *Lecture Notes in Computer Science 812.* Springer–Verlag, 1994, pp. 358-371.
13. K. Ruohonen, Decidability and Complexity of Event Detection Problems for ODEs. To appear in *Complexity.*
14. K. Ruohonen, Reversible Machines and Post's Correspondence Problem for Biprefix Morphisms. *Elektron. Inf.verarb. Kybern. EIK 21*, 1985, pp. 579-595.
15. L. F. Shampine, I. Gladwell and R. W. Brankin, Reliable Solution of Special Event Location Problems for ODEs. *ACM Trans. Math. Software 17*, 1991, pp. 11-25.
16. H. T. Siegelmann, Computation Beyond the Turing Limit. *Science 268*, 1995, pp. 545-548.