

A Generalised Dynamical System, Infinite Time Register Machines, and Π_1^1 -CA₀.

P. Koepke¹ and P.D. Welch²

¹ University of Bonn koepke@math.uni-bonn.de

² University of Bristol p.welch@bristol.ac.uk *

Abstract. We identify a number of theories of strength that of Π_1^1 -CA₀. In particular: (a) the theory that the set of points attracted to the origin in a generalised transfinite dynamical system of any n -dimensional integer torus exists; (b) the theory asserting that for any $Z \subseteq \omega$ and n , the halting set H_n^Z of infinite time n -register machine with oracle Z exists.

Suppose $f : \mathbb{N}^n \rightarrow \mathbb{N}^n$. We are going to consider transfinite iterations of such $f : \mathbb{N}^n \rightarrow \mathbb{N}^n$ as a *generalised dynamical system*. If one wishes, one may think of f acting on the points of an n -dimensional lattice torus where we identify ∞ with 0. We set this up as follows. Given a point $r = (r_1, \dots, r_n) \in \mathbb{N}^n$ set:

$$\begin{aligned} r^0 &= (r_1^0, \dots, r_n^0) = (r_1, \dots, r_n); \\ r^{\alpha+1} &= (r_1^{\alpha+1}, \dots, r_n^{\alpha+1}) = f((r_1^\alpha, \dots, r_n^\alpha)); \\ r^\lambda &= (r_1^\lambda, \dots, r_n^\lambda) = (\text{Liminf}_{\alpha \rightarrow \lambda}^* r_1^\alpha, \text{Liminf}_{\alpha \rightarrow \lambda}^* r_2^\alpha, \dots, \text{Liminf}_{\alpha \rightarrow \lambda}^* r_n^\alpha) \end{aligned}$$

where we define $\text{Liminf}_{\alpha \rightarrow \lambda}^* r_i^\alpha = \text{Liminf}_{\alpha \rightarrow \lambda} r_i^\alpha$ if the latter is $< \omega$, and set it to 0 otherwise, thus:

$$\begin{aligned} r_i^\lambda &= \text{Liminf}_{\alpha \rightarrow \lambda} r_i^\alpha && \text{if the latter is } < \omega \\ &= 0 && \text{otherwise.} \end{aligned}$$

We may ask after the behaviour of points under this dynamic. For example which points ultimately end up at the origin O ? As a more amusing example let $p = (p_0, p_1, p_2) \in (\mathbb{N}^n)^3$ be a triple of three points on the n -dimensional lattice. In general they thus form a proper triangle. Then define:

$$T_f = \{(p_0, p_1, p_2) \in \mathbb{N}^{n \cdot 3} \mid \exists \alpha \ p_0^\alpha = p_1^\alpha = p_2^\alpha\}.$$

T_f is thus the set of possible starting triangles, which at some point collapse and become coincident after iteration of their vertices (and remain collapsed of course from some point α_0 onwards).

* Keywords: dynamical system, Infinite Time Computability, Weak subsystems of analysis

Definition 1. Let GDS_T be the statement: “ $\forall n \forall f : \mathbb{N}^n \longrightarrow \mathbb{N}^n (T_f \text{ exists}).$ ”

Clearly a certain amount of analysis is needed to show that GDS_T holds. But how much? We use the common nomenclature for *subsystems of second order number theory* as taken, for example from [6]. Here “ ACA_0 ” and “ $\Pi_1^1\text{-CA}_0$ ” stand for arithmetical and Π_1^1 -Comprehension axioms respectively. “ ATR_0 ” is arithmetical transfinite recursion. Recall that these are boldface theories allowing real (*i.e.* set of integer) parameters (and that ACA_0 is included in ATR_0).

Theorem 1. Over ATR_0 the statements GDS_T and $\Pi_1^1\text{-CA}_0$ are equivalent.

Similar results hold for starting triangles which in particular collapse to the origin, or triangles which become collinear at some stage. As mentioned above, we could define

$$Z_f =_{\text{df}} \{p \in \mathbb{N}^n \mid \exists \alpha p^\alpha = O\}.$$

The appropriate statement GDS_Z would be “ $\forall n \forall f : \mathbb{N}^n \longrightarrow \mathbb{N}^n (Z_f \text{ exists}).$ ”

As VI.1 in [6] shows, a typical theory of the same strength as $\Pi_1^1\text{-CA}_0$ is: “For every tree $T \subseteq \omega^{<\omega}$ there is a perfect subtree $P \subseteq T$ such that the set of paths through T not a path through P forms a countable set.” This is thus Cantor’s result that any closed set in \mathbb{R} can be decomposed as a countable set together with a perfect set. There are many other statements known to be equivalent to $\Pi_1^1\text{-CA}_0$.

A feature of our dynamical systems is the use of the liminf process on the lattice of some arbitrary but finite dimension. The device used is to then redefine such a coordinate at a limit stage which has been sent out to ∞ to be zero. This method has also been employed in the *Infinite Time Register Machines* of Koepke and Miller [4] which we shall use to prove the theorem. We assume the notation from this paper or from [1].

Briefly such a machine consists of a standard Shepherdson-Sturgis Register Machine [5] (or see Cutland [2]), with a standard program for such a device. Such a program is a finite set of instructions, to do some basic register contents manipulation of the finitely many registers R_0, \dots, R_{N-1} of the machine. The infinitary behaviour is defined by letting register values at limit stages be liminf*’s of previous values.

Definition 2. Let N be a natural number. An N -register machine has registers R_0, R_1, \dots, R_{N-1} which can hold natural numbers. An N -register program is a finite list $P = I_0, I_1, \dots, I_{s-1}$ of instructions, each of which may be of one of five kinds where m, n range over the numbers $0, 1, \dots, N-1$:

1. the zero instruction $Z(n)$ changes the contents of R_n to 0;
2. the successor instruction $S(n)$ increases the natural number contained in R_n by 1;
3. the oracle instruction $O(n)$ replaces the content of the register R_n by the number 1 if the content is an element of the oracle, and by 0 otherwise;
4. the transfer instruction $T(m, n)$ replaces the contents of R_n by the natural number contained in R_m ;

5. the jump instruction $J(m, n, q)$ is carried out as follows: the contents r_m and r_n of the registers R_m and R_n are compared; then, if $r_m = r_n$, the machine proceeds to the q th instruction of P ; if $r_m \neq r_n$, the machine proceeds to the next instruction in P .

The instructions of the program can be addressed by their indices which are called program states. At ordinal time τ the machine will be in a configuration consisting of a program state $I(\tau) \in \omega$ and the register contents which can be viewed as a function $R(\tau) : N \rightarrow \omega$. $R(\tau)(n)$ is the content of register R_n at time τ . We also write $R_n(\tau)$ instead of $R(\tau)(n)$.

Definition 3. Let P be an N -register program. Let $R_0(0), \dots, R_{N-1}(0)$ be natural numbers and $Z \subseteq \omega$ be an oracle. These data determine the infinite time register computation

$$I : \theta \rightarrow \omega, R : \theta \rightarrow ({}^N\omega)$$

with program P , input $R_0(0), \dots, R_{N-1}(0)$ and oracle Z by recursion:

1. θ is an ordinal or $\theta = \text{Ord}$; θ is the length of the computation;
2. $I(0) = 0$; the machine starts in state 0;
3. If $\tau < \theta$ and $I(\tau) \notin s = \{0, 1, \dots, s-1\}$ then $\theta = \tau + 1$; the machine halts if the machine state is not a program state of P ;
4. If $\tau < \theta$ and $I(\tau) \in s$ then $\tau + 1 < \theta$; the next configuration is determined by the instruction $I_{I(\tau)}$ according to the previous definition;
5. If $\tau < \theta$ is a limit ordinal, then $I(\tau) = \liminf_{\sigma \rightarrow \tau} I(\sigma)$ and for all $k < \omega$

$$R_k(\tau) = \liminf_{\sigma \rightarrow \tau}^* R_k(\sigma).$$

So the register R_k is reset in case $\liminf_{\sigma \rightarrow \tau} R_k(\sigma) = \omega$.

If the computation halts then $\theta = \beta + 1$ is a successor ordinal and $R(\beta)$ is the final register content. In this case we say that P computes $R(\beta)(0)$ from $R(0)$ and the oracle Z , and we write $P : R(0), Z \mapsto R(\beta)(0)$.

Definition 4. A partial function $F : \omega^n \rightarrow \omega$ is computable if there is some N -register program P such that for every n -tuple $(a_0, \dots, a_{n-1}) \in \text{dom}(F)$,

$$P : (a_0, \dots, a_{n-1}, 0, 0, \dots, 0), \emptyset \mapsto F(a_0, \dots, a_{n-1}).$$

Using the \liminf process to define an instruction at a limit stage of time λ , rather neatly sets the machine to perform the beginning instruction of the outermost loop, (which we can think of as a 'subroutine') entered unboundedly often below λ . After executing the instruction $I(\alpha)$ the program may move on to execute the next line of instructions ($I(\alpha + 1)$ is the line following $I(\alpha)$), or else if a jump is involved, $I(\alpha + 1)$ may be elsewhere in the program. If the \liminf of the contents of a register becomes infinite, then we reset the contents value to 0: this allows the machine to continue.

Elementary arguments show that any register machine, working on any program, with any initial distribution of register contents, will either halt, or enter a permanently looping cycle by some countable ordinal stage. It is useful to add the capability of a machine to consult an *oracle* $Z \subseteq \omega$, and receive a 0/1 as to whether $R_i(\alpha)$ is in Z or not.

Again, however, immediately there is a question of, *how long* must one wait in order to see that the given machine with its program, oracle, and starting register values, is indeed looping?

Koepke and Miller give a criterion for a machine to be in a looping cycle. They dub a pair (I, R) of an instruction number (about to be performed) and a list of the current register contents R a *constellation*:

Lemma 1. *Let $I : \theta \rightarrow \omega, R : \theta \rightarrow {}^n\omega$ be a computation of the n register machine with program P and with oracle Z for order type θ many stages. Then if this computation has not halted by stage θ , then it will never do so if θ is sufficiently large so that there is some constellation (I', R') so that*

$$\text{otp}\{\{\beta \mid I(\beta) = I' \wedge R(\beta) = R'\}\} \geq \omega^\omega.$$

This gives then a simple criterion to check that we have gone far enough to test whether the computation will halt. Immediately:

Corollary 1. *Any such computation either halts, or cycles after a countable ordinal number of stages.*

It is also easy to sketch an argument that shows how *admissible ordinals* play a role. We let $\omega_k =_{\text{df}} \omega_k^{\text{CK}}$ be the the k^{th} *admissible ordinal* for $k < \omega$. (An ordinal α is *admissible* if $L_\alpha \models \text{KP}$ - the latter denotes Kripke-Platek set theory; ω_1^{CK} is thus the least non-recursive ordinal.) It is well known that any admissible ordinal is *Π_2 -reflecting*: that is if $\varphi(\mathbf{y})$ is any Π_2 formula of the language of set theory, with $\mathbf{y} \in L_\alpha$, then if $L_\alpha \models \varphi(\mathbf{y})$ then there is $\beta < \alpha$ with $L_\beta \models \varphi(\mathbf{y})$. Since the operations of a register machine are Δ_1 and so *absolute* to any transitive admissible set (containing the oracle set Z if there is one: note that a program P of length m is essentially an element of $\text{HF} = L_\omega$), it is clear that we may define the operations of any ITRM inside any admissible set. Imagine we have a virtually trivial machine with $Z = \emptyset$, and only one register R_0 ! As we run the computation, suppose that the program does not halt on its starting input by stage $\eta =_{\text{df}} \omega_1^{\text{CK}}$. Let $\delta_0 < \eta$ be any ordinal. Then trivially:

$$R_0(\eta) = \text{Liminf}_{\delta_0 < \beta \rightarrow \eta}^* R_0(\beta) = k \leq \omega$$

and

$$I(\eta) = \text{Liminf}_{\delta_0 < \beta \rightarrow \eta} I(\beta) = I_l \text{ where } l < m.$$

Let us first consider the case that $\text{Liminf}_{\delta_0 < \beta \rightarrow \eta} R_0(\beta) < \omega$. These values can be expressed by a conjunction of a Σ_2 and a Π_2 statement about δ_0 . Hence by Π_2 reflection, there is some $\delta_1 \in (\delta_0, \eta)$ so that

$$R_0(\delta_1) = R_0(\eta) = \text{Liminf}_{\delta_0 < \beta \rightarrow \delta_1} R_0(\beta) = k \wedge I(\delta_1) = I_l.$$

But as δ_0 was arbitrary, this can be repeated; in short we may define by a Σ_1 recursion a sequence of ordinals δ_ι for $0 < \iota < \eta$ with $R_0(\delta_\iota) = R' \wedge I(\delta_\iota) = I'$, with the constellation $(I', R') = (I_\iota, k)$ ready to be used in Lemma 1. Hence the computation never halts.

Now, suppose the other case holds, and that $\text{Liminf}_{\beta \rightarrow \eta} R_0(\beta) = \omega$; hence by fiat $R_0(\eta)$ is reset to 0. Suppose again $I_\iota = I(\eta)$. Now, suppose the computation continues to $\eta + \eta$ without halting. The segment of computation $[\eta, \eta + \eta]$ is *precisely* the same as that of the machine starting instead on the instruction I_ι ‘at time zero’ with register content $R_0(\eta)$ as initial value, and performing η many steps. We conclude again by the same Lemma 1 that either the machine is looping, or that $R_0(\eta + \eta) = 0$ due to a resetting of the register contents to 0 at time $\eta + \eta$. However now we may have that $I(\eta + \eta)$ is different from $I(\eta)$. Nevertheless we may iterate the argument. Let $\eta' =_{\text{df}} \omega_2^{\text{CK}}$ the second admissible ordinal. Suppose the computation continues as far as η' without halting. Recalling that admissible ordinals are multiplicatively closed, we conclude that for every ordinal $\gamma = \eta \cdot (\delta + 1)$ for some $\delta < \eta'$, that if the machine is not looping by stage γ then $R_0(\gamma) = 0$ again by the ‘resetting rule’, and the machine then will perform an instruction $I(\gamma)$. We claim that for some instruction number I_q for $q < m$ that the constellation $(I_q, 0)$ occurs with order type at least ω^ω before stage η' : for some q $(I_q, 0)$ occurs unboundedly in η' and if this only occurs with order type some $\beta < \omega^\omega$ this would afford a map of β cofinally into η' enumerating stages where this constellation occurs, all this in a Σ_1 -definable fashion over L_β . This would contradict the latter’s admissibility. The conclusion is that the computation either halts or enters an indefinite loop by the second admissible ordinal.

It is then a matter of induction, to use this argument again on machines with n registers, to see that they either halt or enter an infinite loop by stage ω_{n+1}^{CK} (the details of the induction are in Theorem 9 of [3]).

Definition 5. (i) (n -register halting set)

$$H_n =_{\text{df}} \{ \langle e, r_0, \dots, r_{n-1} \rangle \mid P_e(r_0, \dots, r_{n-1}) \downarrow \}$$

(ii) ITRM_n is the assertion: “The n -register halting set H_n exists.”

(iii) **ITRM** is the similar relativized statement that

“For any $Z \subseteq \omega$, for any $n < \omega$ the n -halting set H_n^Z exists.”

Theorem 2.

(i) ITRM_n can be proven in $\text{KP} +$ “there exist n admissible ordinals $> \omega$.”

(ii) **ITRM** can be proven in $\Pi_1^1\text{-CA}_0$.

Proof: (i) has essentially been outlined above where ITRM_1 was done in detail.

For (ii) $\Pi_1^1\text{-CA}_0$ proves that for any $Z \subseteq \omega$ the *hyperjump* $\text{HJ}(Z)$ of Z exists (in Simpson [6] VII.1.16). The hyperjump (as defined at [6] VII.1.5) is a complete $\Sigma_1^{1,Z}$ -set of integers. Now $\Pi_1^1\text{-CA}_0^{\text{set}}$ (recall that this is also a boldface theory as outlined in VII.3 *op. cit.*, and is, for the language of second order arithmetic, L^2 , a conservative extension of $\Pi_1^1\text{-CA}_0$) proves that for every $Z \subseteq \omega$ there is a class of constructible sets L^Z constructed from Z , which is a model of the Axiom

Beta. As L^Z is a model of Beta, we have that $\text{HJ}(Z)$ is a set in L^Z . Repeating this argument, we have that for every $k < \omega$ that $L^Z \models \text{HJ}(k, Z)$ exists (where $\text{HJ}(k, Z)$ is the k 'th hyperjump of Z). Then we may conclude that $L^Z \models \omega_k^{Z, \text{CK}}$ exists. But the arguments deployed above show then that for any k $L^Z \models H_k^Z$ exists. By absoluteness $H_k^Z = (H_k^Z)^{L^Z}$. This suffices. Q.E.D.

Now for the converse:

Theorem 3. (i) $\text{ATR}_0 + \text{ITRM} \vdash \Pi_1^1\text{-CA}_0$.

(ii) *There is a fixed $k < \omega$ so that for any $n < \omega$*
 $\text{ATR}_0 + \text{ITRM}_{n,k} \vdash \text{“HJ}(n, \emptyset) \text{ exists.”}$.

Proof: (Sketch (i)) Define $T \subseteq {}^{<\omega} \omega$ to be a *tree* in the usual way. A *path through T* is then a function $h : \omega \rightarrow T$ with $\forall i (h(i) \text{ is } T\text{-extended by } h(i+1))$. By Simpson [6] VI.1.1 it suffices to show that if $\langle T_k \mid k < \omega \rangle$ is a sequence of trees, that there exists the set $X = \{k \mid T_k \text{ has a path}\}$. Let then $\langle T_k \mid k < \omega \rangle$ be such a sequence. We may assume that it is coded in some recursive manner by $Z \subseteq \omega$. Modify the program of Lemma 1 of [1], to search for paths through trees coded into Z : this is straightforward, as that program checks for ill-foundedness anyway: all that is needed to do is to incorporate the recursive function decoding a T_k from Z . Suppose then that $P_f^Z(x)$ is the program searching for the answer to the ill-foundedness of T_x . Thus for any k $P_f^Z(k) \downarrow 1$ iff T_k is ill-founded, and $P_f^Z(k) \downarrow 0$ otherwise. Adjusting the program slightly just to diverge in the latter case, there is then a program $P_{f'}^Z$ so that

$$T_k \text{ is ill-founded iff } \langle f', k, 0, \dots, 0 \rangle \in H_m^Z$$

(where we have a sequence of $m - 2$ zeroes, where $P_{f'}$ uses $m - 1$ registers; the number of registers the latter actually uses will depend also on the number of registers used in the implementation of the code of Lemma 1 of [1]). **ITRM** tells us that H_m^Z exists as a set. Hence by ACA_0 so does $X = \{k \mid T_k \text{ has a path}\}$.

By, e.g., Simpson [6] VII.1.16, we may also show that $\text{ATR}_0 + \text{ITRM} \vdash \text{“For every } Z \subseteq \omega (\text{HJ}(Z) \text{ exists)”}$ to achieve the same conclusion. We first sketch this before detailing (ii).

Fix Z . Using Z as an oracle, we can, using the properties of ITRMs test for each given index e in turn, whether $\{e\}^Z$ is in WF or not. This again incorporates the argument of Theorem 1 of [1], which shows how given $Y \subseteq \omega$ there is a program P_e^Y (uniform in Y) that decides with a 0/1 output if $Y \in \text{WF}$. We wish to use this for each Y of the form $\{e\}^Z$. We thus extend the ‘pseudo-code’ of the program of Lemma 1 of [1], so that on input e with data Z it may compute membership questions about $\{e\}^Z$, and then run P_e^Y on $Y = \{e\}^Z$ as a sub-program. However this extension is straightforward (and also uniform in Z). This provides an index f so that $P_f^Z(e) \downarrow 1$ iff $\{e\}^Z \in \text{WF}$, and $P_f^Z(e) \downarrow 0$ otherwise. Adjusting this program to one with an index f' so that $P_{f'}^Z(e) \uparrow$ whenever $\{e\}^Z \notin \text{WF}$, we shall have $\{e\}^Z \in \text{WF}$ iff $\langle f', e, 0, \dots, 0 \rangle \in H_m^Z$ (where we have a sequence of $m - 2$ zeroes, where $P_{f'}$ uses $m - 1$ registers; again the number of registers the latter actually uses will depend also on the number of registers used in the implementation of the code of Lemma 1 of [1]).

ITRM tells us that H_m^Z exists as a set. Hence by ACA_0 so does \mathcal{O}^Z . However, there is (uniformly in Z) a (1-1) recursive function $h : \mathbb{N} \rightarrow \mathbb{N}$ with $m \in \text{HJ}(Z) \leftrightarrow h(m) \notin \mathcal{O}^Z$. (The complement of \mathcal{O}^Z and $\text{HJ}(Z)$ are both complete Σ_1^1 sets of integers and hence recursively isomorphic.) Hence $\text{HJ}(Z)$ exists. Hence assuming that we used overall k registers, what we actually have shown in the above is that $\text{ATR}_0 + \text{ITRM}_k \vdash \text{“HJ}(1, Z) = \text{HJ}(Z) \text{ exists.}”$

This argument can be repeated to get any n^{th} hyperjump of Z . Let $Z^{h(1)} =_{\text{df}} \text{HJ}(1, Z)$ as above. We then have argued that there is a k -register machine (for some $k < \omega$) so that it gives 0/1 answers to queries as to whether $n \in Z^{h(1)}$. In order to compute $Z^{h(2)} =_{\text{df}} \text{HJ}(2, Z)$ it suffices then to compute $\mathcal{O}^{Z^{h(1)}}$. We need thus to check answers to queries such as whether $\{e\}^{Z^{h(1)}} \in \text{WF}$ or not. In order to do this we employ again the routine to check for the e^{th} recursive function in a set Y and ask whether it is in WF; here whenever we have to query $?n \in Y?$ since we shall take $Y = Z^{h(1)}$, we must run (as a subroutine) the procedure to test if $?n \in Z^{h(1)}$. We may reserve the first k registers for this work (as described above), and the next k registers for computing the 0/1 answer to $?\{e\}^{Z^{h(1)}} \in \text{WF}$? In this fashion we may compute $\mathcal{O}^{Z^{h(1)}}$ and hence $Z^{h(2)}$. Inductively this means that the n^{th} hyperjump can be computed using $n \cdot k$ registers.

However, applying this to $Z = \emptyset$ is our (ii).

Q.E.D.

Remark 1. Using the registers in the algorithm for the iterated hyperjump more efficiently one could probably obtain a tighter bound for the number of registers needed, like $n + k$ instead of $n \cdot k$.

Remark 2. The paper [7] also contains results on certain infinite time computabilities and reverse mathematics.

Proof of Theorem 1. Assume GDS_T . We define a function $f : \mathbb{N}^m \rightarrow \mathbb{N}^m$ so that for any Z , $H_n^Z \leq_T T_{f'}$ for some $m > n$. We use the fact that the n -register machine halting set H_n^Z is in fact decidable by some n' -register machine with program $P_{e(n)}^Z$. This is one of the basic properties of ITRMs (*cf.* Theorem 4 of either [4] or [1]); recall that in particular this will require more registers than the n under observation. Let us assume $e(n)$ chosen so that $\langle e, r_0, \dots, r_{n-1} \rangle \in H_n^Z$ iff $P_{e(n)}^Z(e, r_0, \dots, r_{n-1}, 0, \dots, 0) \downarrow 1$. We'll make some further harmless modifications to $P_{e(n)}^Z$; this will describe a program $P_{e(n)'}^Z$: firstly, that if $P_{e(n)}^Z$ uses n_0 registers, then $P_{e(n)'}^Z$ uses a final extra register R_{n_0} which however is not addressed before the halting procedure shortly to be described.

Secondly that before halting, if $P_{e(n)}^Z$ is to set to halt with a 1 in R_0 , $P_{e(n)'}^Z$ sets R_{n_0} to 0; otherwise it leaves R_{n_0} unaltered. Now consider the register contents as the coordinates of points $p_i \in \mathbb{N}^{n_0+1}$ for $i < 3$ by letting the j^{th} coordinate of p_i be R_j , for $j \leq n_0$. The action of the program $P_{e(n)'}^Z$ defines a function $f : \mathbb{N}^{n_0+1} \rightarrow \mathbb{N}^{n_0+1}$ with each machine step corresponding to another iterative application of f . Suppose we start $P_{e(n)'}^Z$ with the register configuration $(e, r_0, \dots, r_{n-1}, 0, \dots, i)$; we set $p_i = p_i^0$ to have the coordinates of this starting

configuration. Then the register configuration at stage α of the computational process yields the coordinates of p_i^α .

Then we shall have $\langle e, r_0, \dots, r_{n-1} \rangle \in H_n^Z$ iff $(p_0^0, p_1^0, p_2^0) \in T_f$. By ACA_0 we thus have that H_n^Z is a set.

Conversely, assume **ITRM**. Let $f : \mathbb{N}^n \rightarrow \mathbb{N}^n$. By using repeated tupling functions, prime power coding and the usual devices, we may assume that f is coded by some $Z = Z_f \subseteq \omega$. In particular, as an oracle we may ‘query’ of Z given (r_0, \dots, r_{n-1}) , the value of $f(r_0, \dots, r_{n-1})$. We may thus define ITRM machines P^f that incorporate the function f . However then it is an elementary argument to write down an algorithm using such P^f on input a triple of points p_i each in \mathbb{N}^n , to compute the iterates of f on each p_i and see if they become coincident. If this algorithm is exemplified by P_e^f on n_0 registers then the halting set $H_{n_0}^{Z_f}$ tells us the answer to this question. We thus have for the triangle set T_f that $T_f \leq_T H_{n_0}^{Z_f}$. As **ITRM** asserts the existence of $H_{n_0}^{Z_f}$, by ACA_0 T_f exists.

Q.E.D. Theorem 1

References

- [1] M. Carl, T. Fischbach, P. Koepke, R. Miller, M. Nasfi, and G. Weckbecker. The basic theory of infinite time register machines. *Archive for Mathematical Logic*, 49(2):249–273, 2010.
- [2] N. Cutland. *Computability: an Introduction to Recursive Function Theory*. CUP, 1980.
- [3] P. Koepke. Ordinal computability. In K. Ambos-Spies *et al.*, editor, *Mathematical Theory and Computational Practice*, volume 5635 of *Lecture Notes in Computer Science*, pages 280–289. Springer, 2009.
- [4] P. Koepke and R. Miller. An enhanced theory of infinite time register machines. In A. Beckmann *et al.*, editor, *Logic and the Theory of Algorithms*, volume 5028 of *Springer Lecture Notes Computer Science*, pages 306–315. Springer, 2008.
- [5] J. Shepherdson and H. Sturgis. Computability of recursive functionals. *Journal of the Association of Computing Machinery*, 10:217–255, 1963.
- [6] S. Simpson. *Subsystems of second order arithmetic*. Perspectives in Mathematical Logic. Springer, January 1999.
- [7] P. D. Welch. Weak systems of determinacy and arithmetical quasi-inductive definitions. To appear in *Journal of Symbolic Logic*, 76, 2011.