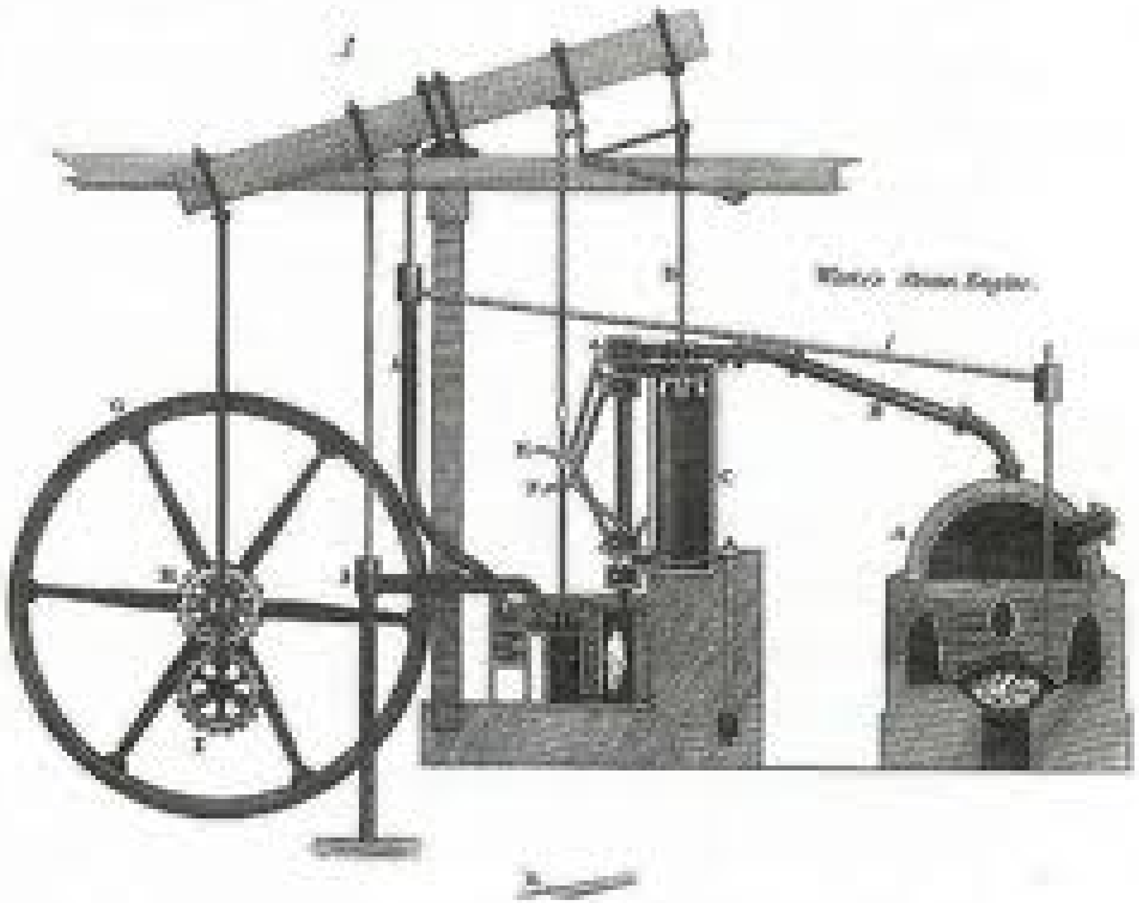
# Continuous transfinite Blum-Shub-Smale computations and a Church-like thesis for poly-time on infinite strings

*P.D.Welch University of Bristol* *NUS: Sets and Computations 16.iv.15*

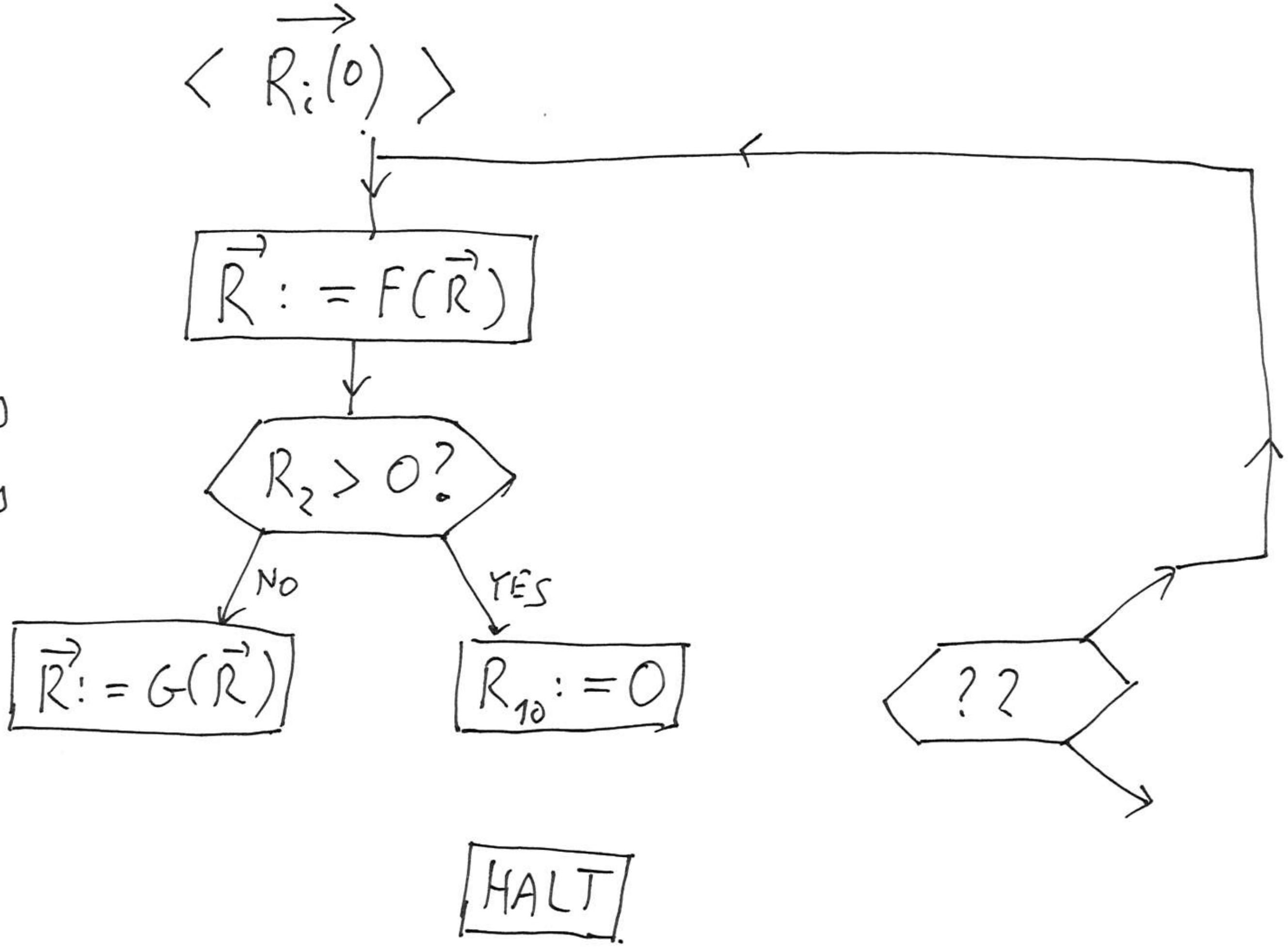


# Introduction: BSS Machines

*Idea*: A conceptual model for handling computations on reals from $\mathbb{R}$:

- *Registers* for reals $R_1, \ldots, R_N$ with contents $R_1(t), \ldots, R_N(t)$ at *time* or *stage* $t$.
- A finite *program* $P_e$ consisting of instructions $I_1, \ldots, I_K$ - with the current instruction $I(t)$.
- If, and when, the computation halts, at time say $\theta$, we say that the machine has *computed* $R_1(\theta)$: $P_e(\overrightarrow{R_i(0)}) \downarrow R_1(\theta)$.

Assume for simplicity only rational functions F, G at function nodes (Test for 0 first)



# Transfinite BSS machines: $I$BSS's[1]

Transfinite Action:

- **(Continuity)** If, at limit stage $\lambda$, any of the $\lim_{\alpha\to\lambda} R_i(\alpha)$ do not exist, then $P_e(\overrightarrow{R_i(0)})$ *crashes*.
- NB: This includes the case that $\lim_{\alpha\to\lambda} R_i(\alpha) = \infty$.
- *Limit instruction:* $I(\lambda) = \liminf_{t\to\lambda} I(t) =$
  the least instruction $\sharp$ performed cofinally in $\lambda$.

• **(Continuity)** is a stringent constraint.

---

[1] *Towards a theory of infinite time Blum-Shub-Smale Machines*, P. Koepke and B. Seyfferth, CiE2012 Proceedings, Springer LNCS, 2012.

# Transfinite BSS machines: *I*BSS's[1]

Transfinite Action:

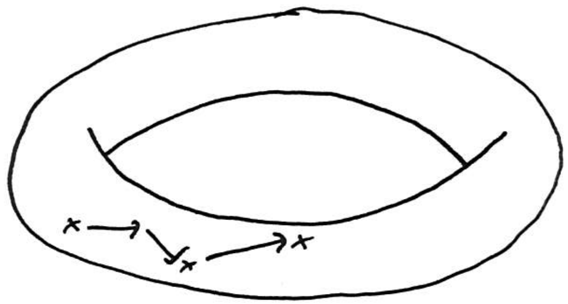
- **(Continuity)** If, at limit stage $\lambda$, any of the $\lim_{\alpha\to\lambda} R_i(\alpha)$ do not exist, then $P_e(\overrightarrow{R_i(0)})$ *crashes*.
- NB: This includes the case that $\lim_{\alpha\to\lambda} R_i(\alpha) = \infty$.
- *Limit instruction:* $I(\lambda) = \liminf_{t\to\lambda} I(t) =$
  the least instruction $\sharp$ performed cofinally in $\lambda$.

• **(Continuity)** is a stringent constraint.

• Koepke-Seyfferth ask:
Q1 How long do the machines compute before looping/crashing?
Q2 What do the machines compute?
Q3 How do the above depend on $N$, the no. of registers?
(They answer Q1& Q3. We have answered Q2.)

[1] *Towards a theory of infinite time Blum-Shub-Smale Machines*, P. Koepke and B. Seyfferth, CiE2012 Proceedings, Springer LNCS, 2012.

# A transfinite dynamical system



N-dim. torus $[0,1]^N$, identify $1 \to 0$.

Q4: For program $P_e$, what is the origin set $O_e := \{\vec{r} : P_e(\vec{r}) \downarrow 0\}$?

Q5: How complex is $\{e : O_e \neq \emptyset\}$?

# Q1: Looping times

**Theorem (Koepke-Seyfferth)**

*Any IBSS machine $P_e$ crashes, halts, or is looping, by time $\omega^{M+1}$ where $M$ is the $\sharp$ of function nodes in $P_e$.*

**Proof:** ($M = 1$) Suppose for a contradiction the point moves after stage $\omega^2$. Then the function $F(\vec{x})$ was applied at some stage $> \omega^2$ *but was not applied cofinally in* $\omega^2$ (by **(Continuity)**, since otherwise $\vec{R_i(\omega^2)}$ is a fixed point). So for some stage $\alpha < \omega^2$ the computation can be regarded as one starting from $\vec{R_i(\alpha)}$ using a program with one less function node. In this case (M=1) the program then use only the finitely many query nodes, and must be looping by stage $\omega^2$. Contradiction!
$M = l + 1$: repeat: argue by induction. $\square$

# Q1: Looping times

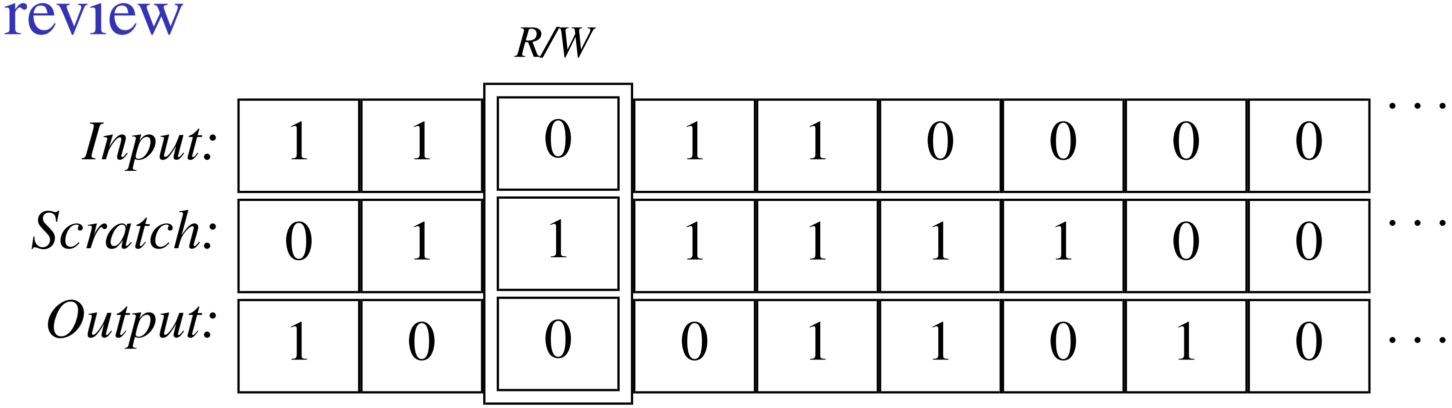
**Theorem (Koepke-Seyfferth)**

*Any IBSS machine $P_e$ crashes, halts, or is looping, by time $\omega^{M+1}$ where $M$ is the $\sharp$ of function nodes in $P_e$.*

**Proof:** ($M = 1$) Suppose for a contradiction the point moves after stage $\omega^2$. Then the function $F(\vec{x})$ was applied at some stage $> \omega^2$ *but was not applied cofinally in* $\omega^2$ (by **(Continuity)**, since otherwise $R_i(\vec{\omega^2})$ is a fixed point). So for some stage $\alpha < \omega^2$ the computation can be regarded as one starting from $R_i(\vec{\alpha})$ using a program with one less function node. In this case (M=1) the program then use only the finitely many query nodes, and must be looping by stage $\omega^2$. Contradiction!
$M = l + 1$: repeat: argue by induction. □

*Remark*:
(1) The number of registers was irrelevant. This answers Q1 and Q3.
(2) Since time $\omega^\omega$ is all that's needed, all IBSS computations can be run inside $L_{\omega^\omega}[\vec{R}_i]$ and by absoluteness are the same computations as in $V$. Hence the IBSS computable reals from inputs $\vec{R}_i$ are all in $L_{\omega^\omega}[\vec{R}_i]$. Later we shall see the converse.

# ITTM's: a review




- $P_e$: again a finite sequence of instructions.
- At limit stages $\lambda$ the $R/W$ Head, which is on cell $C_{c(t)}$ at time $t$, goes back to $C_{c(\lambda)}$ where $c(\lambda) := \liminf^*_{t\to\lambda} c(t) =_{df}$
  the least cell ♯ visited visited cofinally in $\lambda$.
  - *Limit instruction:* $I(\lambda) := \liminf_{t\to\lambda} I(t) =_{df}$
    the least instruction ♯ performed cofinally in $\lambda$.
  - *Cell update* $C_i(\lambda) := \liminf_{t\to\lambda} C_i(t)$ for $i < \omega$.
- We may consider running such on input $\vec{r} \in (2^\omega)^N$ restricting to $\omega^\omega$ steps only; as output we have precisely the $\Delta^0_{\omega^\omega}(\vec{r})$ reals.

# Confluence

**Theorem (W)**

*The following classes of functions of the form* $F : (2^{\mathbb{N}})^k \to 2^{\mathbb{N}}$ *are extensionally equivalent:*
(I) *Those functions computed by a continuous* IBSSM *machine;*
(II) *Those functions that are polynomial time* ITTM*;*
(III) *Those functions that are* SRSF, (PRSF,CRSF).

Proof:[2] (I) $\subseteq$ (II): We take $k = 1$. By Koepke-Seyfferth for any IBSSM computable $F$ there is $N < \omega$ so that $F(x)$ is computable in less than $\omega^N$ steps.
• Consider that computation to be performed inside $L_{\omega^N}[x]$.
• ITTM-compute a code for any $L_{\omega^N}[x]$, and its theory, uniformly in the input $x$ by time $\omega^{N+3}$. Since we have the theory, we have the digits of the final halting IBSSM-output (or otherwise the fact that it is looping or has crashed), since these are also part of the set theoretical truths of $L_{\omega^N}[x]$).
Thus (I) $\subseteq$ (II).

---

[2]See, *e.g.*, P. D Welch, *Turing's Legacy*, Ed. R. Downey, LNL, vol 42, CUP, 2012, pp 493-529.

Proof contd.)

**Theorem**

*The following classes of functions of the form $F:(2^{\mathbb{N}})^k \to 2^{\mathbb{N}}$ are extensionally equivalent:*
(I) *Those functions computed by a continuous* IBSSM *machine;*
(II) *Those functions that are polynomial time* ITTM*;*
(III) *Those functions that are* SRSF*,* (PRSF,CRSF).

(II) $\subseteq$ (III): If $F$ is in the class (II), then for some $N < \omega$, by absoluteness, $F(x)$ is ITTM-computable within $L_{\omega^N}[x]$. By setting up the definition of the ITTM program $P$ computing $F$, we may define some $\alpha$ such that the output of that program $P$ on $x$ (*i.e.* $F(x)$) is always the $\alpha$'th element in the natural wellorder of $L_{\omega^N}[x]$ uniformly in $x$. However the set $L_{\omega^N}[x]$ is SRSF-recursive from $\{x\}$ (again uniformly in $x$) as is a code for $\alpha$. This yields the conclusion that we may find uniformly the output of $P(x)$ using the code for $\alpha$, again as the output of an SRSF-recursive-in-$x$ function. This renders (II) $\subseteq$ (III).

Theorem

*The following classes of functions of the form* $F : (2^{\mathbb{N}})^k \to 2^{\mathbb{N}}$ *are extensionally equivalent:*
(I) *Those functions computed by a continuous* IBSSM *machine;*
(II) *Those functions that are polynomial time* ITTM*;*
(III) *Those functions that are* SRSF*,* (PRSF,CRSF).

Proof contd. (III) $\subseteq$ (I) Let $F(x/-)$ be in (III), then there is[3] $M < \omega$ and a $\Sigma_1$-formula $\varphi(v_0, v_1)$ so that

$$F(x/-) = z \text{ iff } L_{\omega^M}[x] \models \varphi[x, z]$$

• We have in turn another $\Sigma_1$ $\psi(v_0, v_1)$ (in $\mathcal{L}_{\dot{x},\dot{\in}}$) so that

$$F(x/-)(k) = z(k) = 1 \text{ iff } L_{\omega^M}[x] \models \psi[x, k].$$

• It thus suffices to be able to compute the $\Sigma_1$-truth sets for $L_\alpha[x]$ for all $\alpha < \omega^\omega$ by IBSSM's. There are a variety of ways one could do this, but it is well known that calculating the $\alpha$'th iterates of the Turing jump relativised to $x$ for $\alpha < \omega^\omega$ would suffice.

---

[3] A. Beckmann, S. Buss, S-D. Friedman, *"Safe Recursive Set Functions"*, Thm. 3.5)

# Coding truth sets into IBSSM comps.

• For $y \in \mathbb{R}$ let $y$ also denote the element of $2^{\mathbb{N}}$ coding the set of integers of $y$'s expansion as an infinite fraction and *vice versa*. Fix an $M < \omega$, to see that we may calculate $x^{(\beta)'}$ for $\beta < \omega^M$.
• Construct a counter to be used in general iterative processes, using registers $C_{M-1}, \ldots, C_0$ whose contents $C(j) = n_j =$ the multiple of $\omega^j$ in the Cantor normal form of $\omega^\beta$. Thus $\beta = \omega^{M-1} \cdot n_{M-1} + \cdots \omega \cdot n_1 + n_0 < \omega^M$ where we are at the $\beta$'th stage in the process.
• Effect this so that $C_0 = C_1 = \cdots = C_{M-1} = 0$ occurs first at stage $\omega^M$.
• Let $p_0 = 2, p_1 = 3$, etc., enumerates the primes. Code the characteristic function of $\{e \in \omega \mid e \in W_e^{x^{(\beta)'}}\}$ as $1/0$'s in the digits at the $s$'th-places after the decimal point of $R_1$ where $s$ is of the form $p_{M+e}.p_0^{n_0+1}. \cdots .p_{M-1}^{n_{M-1}+1}$.
• For limit stages $\lambda < \omega^M$, continuity of the register contents automatically ensures that this real in $R_1$ also codes the disjoint union of the $x^{(\beta)'}$ for $\beta < \lambda$, and at stage $\omega^M$ we have the whole sequence of jumps encoded as required. □

# A Church-like thesis

*Thesis: Any effective notion of computation on $\omega$-strings $x_1, \ldots, x_n$ that runs in less than $\omega^\omega$ steps is SRSF-computable from $(x_1, \ldots, x_n)$.*

- Any claim of justification for this, (as for the original CT-thesis) turns on what one means by 'effective notion'.
- But the case for this seems watertight:
  - Any such 'effective notion' has to be absolute between *ZF*-models, in particular be absolute to $L$.
  - Similarly it surely must be the case that such a notion for computing from $\vec{x}$, must be absolute between $L[\vec{x}]$ and $L_{\omega^\omega}[\vec{x}]$. For if this degree of absoluteness failed, then it would mean that the course of computation from $\vec{x}$ must be appealing to some additional information, some device, some agency, *extra* to $L_{\omega^\omega}[\vec{x}]$. How can one argue that any such external input is 'effective'?
  - And we have seen above that the theory of $L_{\omega^M}[\vec{x}]$ is essentially *SRSF*-computable.

- It would seem then that any notion of 'effective algorithm' for such computation should be coded in some way into $L_{\omega^\omega}$, or at the very worst have a description that is coded there.
- Compare this with arguments of Gandy [4] for standard computation on integers: notions of *effectivity* here must have a finite description, or have finite components which may be coded within the realm of the hereditarily finite sets. Thus a successful, *i.e.* halting, computation is an object in *HF* and relies on nothing outside of *HF*.
- Further compare with Kleene recursion, also a recursion on $2^{\mathbb{N}}$: a course of computation is coded as living on a well founded finite path tree, and is a hyperarithmetic object. Hence such computations belong precisely to the realm that they compute (since a set is Kleene-computable iff it is *HYP* and relies on nothing outside of its realm: $L_{\omega_1^{ck}}$.
- As above then, any effective notion of poly-time on strings should rely on nothing outside its realm: $L_{\omega^\omega}$.

---

[4] R.O. Gandy. *Church's thesis and principles for mechanisms*. In J. Barwise, H.J. Keisler, and K. Kunen, editors, The Kleene Symposium, Studies in Logic and the Foundations of Mathematics, pages 123-148, Amsterdam, 1980. North-Holland.

# Liminf for IBSSM's

- Relax **(Continuity)**. Instead use

**(Liminf)**: For $Lim(\lambda)$ define $R_i(\lambda) = \liminf_{t \to \lambda} R_i(t)$.

Interpret $P_e$ below in this sense.

# Liminf for IBSSM's

• Relax **(Continuity)**. Instead use

**(Liminf)**: For $Lim(\lambda)$ define $R_i(\lambda) = \liminf_{t \to \lambda} R_i(t)$.

Interpret $P_e$ below in this sense.

**Proposition**

KP + $\Pi_3$-Reflection $\vdash$"$\forall x \forall e (P_e(x)$ stabilizes ".

# Liminf for IBSSM's

- Relax **(Continuity)**. Instead use

**(Liminf)**: For $Lim(\lambda)$ define $R_i(\lambda) = \liminf_{t \to \lambda} R_i(t)$.

Interpret $P_e$ below in this sense.

**Proposition**

$\mathsf{KP} + \Pi_3$-Reflection $\vdash$"$\forall x \forall e (P_e(x)$ stabilizes ".

Q6 (Open) $\Delta^1_2$-$\mathsf{CA}_0 \vdash$ " $\forall x \forall e (P_e(x)$ stabilizes " ?