Computing the Uncomputable Rado Sigma Function

An Automated, Symbolic Induction Prover for Nonhalting Turing Machines

Joachim Hertel

We discuss a new tool that is successfully used in an ongoing project to compute Σ(5): an automated symbolic induction prover (SIP). The SIP tool is written in *Mathematica* and provides a unified way to prove that large sets of Turing machines are nonhalters. In a way, an SIP enables certain Turing machines to provide their own proof of being a nonhalter.

■ Introduction

It is a classic result of computability theory that there exist uncomputable functions. A prime example is Tibor Rado's [1] function $\Sigma(n)$ (or the busy beaver function) that measures the productivity of n-state, binary Turing machines. Despite being uncomputable, we know [2] exact values of $\Sigma(n)$ for $n = 1, 2, 3, 4$ and a high lower bound for $n = 5$: $\Sigma(5) \geq 4098$ [3].

To compute the exact value of $\Sigma(n)$ we have to decide the halting question for $(4\,n+1)^{2\,n}$ n-state binary Turing machines. Fortunately, this huge number can be reduced by applying well-known techniques such as tree normalization, backtracking, and simple loop detection [2].

With few undecided Turing machines, we now have strong evidence that indeed $\Sigma(5) = 4098$. Final results will be reported in a forthcoming report [4].

□ Rado's Uncomputable Sigma Function

There are far too many functions from the positive integers to the positive integers for them all to be (Turing) computable. A Cantor diagonalization argument [5] shows that the set of all such functions is not enumerable, whereas the set of all Turing machines is denumerable [5]. Hence, there must exist functions that are uncomputable.

In 1962, Tibor Rado [1] presented the uncomputable function Σ (also known as the busy beaver function). Roughly speaking, $\Sigma(n)$ is the largest number of 1s left on the tape by a halting binary n-state Turing machine when started on an initially all 0 tape. The Σ function is uncomputable because otherwise it would solve the halting problem, which is known to be undecidable [5]. Even more, Rado [1] proved that Σ grows faster than any computable function for large enough values of n, that is, for any computable function $f : \mathbb{N} \to \mathbb{N}$, it holds that $\exists\, k_0\, \forall\, k \geq k_0\; f(k) < \Sigma(k)$.

In 1964 Green [6] established general lower bounds for the Σ function for larger values of n by explicitly constructing so-called class M Turing machines that generate long blocks of 1s. Julstrom [7] showed that Green's class M Turing machines are related to the Ackermann function [5], a function that is computable but not primitive recursive.

Various computationally based approaches to make progress on the Σ function have been taken: brute force searches, genetic algorithms, heuristic behavior analysis, and many more [8]. The Σ function has become a classic topic in the theory of computing [8]. For a comprehensive list of references on Rado's Σ function, see [9].

However, so far, exact values of $\Sigma(n)$ have only been computed for $n = 1, 2, 3, 4$ (see [2, 8]): $\Sigma(1) = 1$, $\Sigma(2) = 4$, $\Sigma(3) = 6$, $\Sigma(4) = 13$. These values are the first elements of the A028444 sequence of integers in The On-Line Encyclopedia of Integers [10].

In 1990 Marxen and Buntrock [3] established the lower bound $\Sigma(5) \geq 4098$, by publishing a record-holding binary 5-state Turing machine that halts after 47,176,870 steps and leaves 4098 1s on the tape.

Michel [11] shows a connection of most of the known low n record-holding Turing machines and the Collatz $3x + 1$ problem.

There are three other related uncomputable functions associated with binary n-state Turing machines:

- Shift(n): The maximum number of moves of a halting binary n-state Turing machine started on an all 0 tape.
- Num(n): The largest unary number that can be created by a halting binary n-state Turing machine started on an all 0 tape.
- Space(n): The largest number of different tape cells scanned by a halting binary n-state Turing machine started on an all 0 tape.

Ben-Amram et al. [12] prove that $\text{Num}(n) \leq \Sigma(n) \leq \text{Space}(n) \leq \text{Shift}(n), \forall\, n$.

□ Why Compute Values of Sigma?

Chaitin argues that the Σ function is of considerable meta-mathematical interest. According to Chaitin [13], let P be a computable predicate of a positive integer n so that for any specific n it is possible to compute if $P(n)$ is true or false. The Goldbach conjecture is an example for P. The Σ function provides a crucial upper bound, for if P has algorithmic information content p, it suffices to examine the first $\Sigma(p)$ natural numbers to decide whether P is true or false for all natural numbers [13]. Hence, the Σ function enables converting a large but finite number of calculations into a mathematical proof. Calculating $\Sigma(n)$ for specific values of n thus amounts to a systematic effort to settle all finitely refutable mathematical conjectures; that is, to determine all constructive mathematical truth [13].

□ Minds, Machines, and Sigma

There is an ongoing philosophical discussion as to whether the human mind can surpass the Turing limit or can be understood as a Turing machine. The famous logician Kurt Gödel discusses his point of view that the mind can indeed surpass the Turing limit in conjunction with his well-known incompleteness theorem. Gödel states in [14], that "although at each stage the number and precision of the abstract terms at our disposal may be finite, both may converge toward infinity in the course of the application of the procedure". Computing values of $\Sigma(n)$ for increasing n serves as a quantitative example for the situation Gödel is referring to. Indeed, Bringsjord et al. [15] present a "new Gödelian argument for hypercomputing minds" that is based on the Σ function. In their argument, the Σ function provides a discrete way to measure the achievement of the human mind, surpassing the Turing limit and establishing new constructive mathematical truth. In [15] the core assumption is that if the human mind can compute $\Sigma(n)$, it also can compute $\Sigma(n+1)$, although that advance might take quite a long time.

Experience from our ongoing project of computing Σ for 5-state binary Turing machines shows that the halting question of large subsets of Turing machines can be decided in a unified way. By using data mining to identify patterns and by the subsequent use of an SIP, we can prove that these machines are in fact nonhalters once they reach a certain pattern when started on an all 0 tape. If we succeed in identifying an induction base (i.e., some pattern), we then can use the SIP to operate as a kind of meta-Turing machine, enabling the underlying Turing machine to create its own proof of being a nonhalter. Nothing in that process is restricted to 5-state Turing machines. Hence we can imagine climbing beyond $n = 5$. The main challenge becomes the data mining part of identifying suitable induction base steps for the increasingly complex and "erratic" behavior of Turing machines.

We now turn in more detail to the computation of $\Sigma(5)$ and to our main computational tool for that task—the SIP for nonhalting Turing machines.

■ The Computation of Sigma for 5-State, Binary Turing Machines

□ Notations and Definitions

Binary n-state Turing machines conform to these conditions:

- The tape is infinite in both directions.
- The tape alphabet is $\Sigma = \{0, 1\}$.
- The all 0 tape is called the blank tape Ω.
- The Turing machine has n states, labeled $1, \ldots, n$, and a halt state, labeled 0.
- At each step the machine reads/writes the cell scanned by the Turing head and moves the Turing head one cell to the left/right.

Definition: The instruction table of a 5-state binary Turing machine M is a 5×2

table such that $M(\mathbf{s}, \mathbf{h}) := \{\text{ws}, \text{mv}, \text{ns}\}$, with $\mathbf{s} \in \{1, 2, 3, 4, 5\}$, $\mathbf{h} \in \{0, 1\}$, $\text{ws} \in \{0, 1\}$, $\text{mv} \in \{L, R\}$, $\text{ns} \in \{0, 1, 2, 3, 4, 5\}$.

We call **s** the current state of M and **h** the read symbol in the tape cell positioned under the Turing head H. The triple {ws, mv, ns} is called a Turing instruction, with the write symbol ws being written into the tape cell positioned under the Turing head H, mv the move direction of the Turing head H, and ns the next state of M. If ns = 0, M stops; otherwise, it continues executing instructions.

Without loss of generality, we assume the halt instruction to be {1, R, 0}.

That leaves us with $(2 \times 2 \times 5 + 1)^{2\times 5} = 21^{10}$ possible binary 5-state Turing machines. We call this finite set $\mathcal{TM}_5$.

Let $M \in \mathcal{TM}_5$. $M(\Omega)$ means Turing machine M is started in state 1 on tape Ω.

We define

$$\sigma(M) := \begin{cases} k & M(\Omega) \text{ halts and leaves } k \text{ 1s on the tape} \\ 0 & \text{otherwise.} \end{cases}$$

Here is an example of a a 5-state Turing machine first published by Marxen and Buntrock [3]:

$$M_{\text{MB}} = \begin{pmatrix} \{1, R, 2\} & \{1, L, 3\} \\ \{1, R, 3\} & \{1, R, 2\} \\ \{1, R, 4\} & \{0, L, 5\} \\ \{1, L, 1\} & \{1, L, 4\} \\ \{1, R, 0\} & \{0, L, 1\} \end{pmatrix}.$$

$M_{\text{MB}}(\Omega)$ halts after 47,176,870 steps and leaves 4098 1s on the tape; that is, $\sigma(M_{\text{MB}}) = 4098$:

$$\Sigma(5) := \operatorname*{Max}_{M \in \mathcal{TM}_5} \sigma(M).$$

Hence, $\Sigma(5) \geq 4098$ and we have to solve the halting question for a finite, but large number of Turing machines to compute the value of $\Sigma(5)$.

Configurations

Let Σ^* denote the set of finite words from the alphabet Σ. For $x \in \Sigma^*$ and $y \in \Sigma^*$ let x || y $\in \Sigma^*$ denote the concatenated word.

A machine configuration c is an expression

$$c = \{\mathbf{s}, \{\omega, \{\{x\}, 1\}, \mathbf{h}, \{\{y\}, 1\}, \omega\}\}, \ \mathbf{s} \in \{0, 1, 2, 3, 4, 5\}, \ \mathbf{h} \in \Sigma, \ x, y \in \Sigma^*,$$

where ω denotes the left/right infinite blank portion of the tape. In this notation $\Omega = \{\omega, \mathbf{0}, \omega\}$.

We use $\{\{x\}, n\} := \left\{\{x, \ldots, x\}, 1\right\}$ (n-times) for all $x \in \Sigma^*$ to define symbolic configurations with multipliers $n \geq 1$, as in $c = \{\mathbf{s}, \{\omega, \{\{x\}, n\}, \mathbf{h}, \{\{y\}, m\}, \omega\}\}$. For example, $c = \{3, \{\omega, 0, \{\{1, 0, 1\}, k\}, \omega\}\}$. That is, M is in state 3, the Turing head scans a 0 cell, and the subtape {1, 0, 1} occurs k times.

□ General Approach: Discover and Prove

By using well-known techniques [2], such as tree normalization, backtracking, or simple loop detection, the halting question can be decided for many of the 5-state binary Turing machines.

If we apply such techniques to the 21^{10} possible 5-state binary Turing machines, we find approximately 1,000,000 undecided machines (holdouts).

For each of the remaining undecided Turing machines:

- Iterate the holdout machine a "number of times", starting on the blank tape Ω.
- Save the configuration after each step.
- Try to discover recurring patterns and store them as machine configurations.

Here are some examples with $x, y \in \Sigma^*$ and the multiplier p_n satisfying a recurrence relation $p_{n+1} = a\ p_n + b$:

$$\begin{aligned} c_L(n) &:= \{\mathbf{s}, \{\omega, \mathbf{h}, \{\{x\}, a\ n + b\}, \omega\}\}, \\ c_R(n) &:= \{\mathbf{s}, \{\omega, \{\{x\}, a\ n + b\}, \mathbf{h}, \omega\}\}, \\ c_L(n) &:= \{\mathbf{s}, \{\omega, \mathbf{h}, \{\{x\}, 1\}, \{\{y\}, p_n\}, \omega\}\}, \\ c_R(n) &:= \{\mathbf{s}, \{\omega, \{\{x\}, 1\}, \{\{y\}, p_n\}, \mathbf{h}, \omega\}\}, \end{aligned}$$

where $n \geq 0$.
Here are some actual examples of configurations.

A simple linear configuration:

$$c(n) = \{\mathbf{2}, \{\omega, \mathbf{0}, \{\{1\}, 4\,n\}\}, \omega\}.$$

A simple exponential configuration:

$$c(n) = \{\mathbf{1}, \{\omega, \mathbf{1}, \{\{0\}, 3^n\}, \{\{1\}, 3\}, \omega\}\}.$$

A more complicated configuration:

$$\begin{aligned} c(n) = \{\mathbf{1}, \{\omega, \mathbf{0}, \{\{0, 0, 1\}, 2^{2\,n+1}\}, \{\{0, 1\}, 3\}, \{\{0, 0, 1\}, 2^{2\,n-1}\}, \{\{0, 1\}, 3\}, \\ \ldots, \{\{0, 0, 1\}, 2^3\}, \{\{0, 1\}, 3\}, \{\{0, 0, 1\}, 2^1\}, \{\{0, 1\}, 3\}, \omega\}\}. \end{aligned}$$

Once we have identified a reliable hypothesis for a recurring configuration $c(n)$ of a Turing machine M, we try to create an induction proof showing that M does indeed not halt.

The general scheme is as follows:

> **Step 1**: Establish the induction proposition: $M : \Omega \Longrightarrow c(n)$, $\forall\ n \geq 0$ (i.e., M transforms the blank tape into $c(n)$ in a countable number of steps).

Step 2: Verify the base case; that is, check that $M:\Omega \Longrightarrow c(n)$, $n = 0$ (or any finite number n).

Step 3: Formulate the induction hypothesis; that is, assume that $M:\Omega \Longrightarrow c(k)$, $\forall\ 0 \le k \le n$.

Step 4: Prove the induction step; that is, $M:\Omega \Longrightarrow c(n+1)$, or equivalently $M:c(n) \Longrightarrow c(n+1)$.

Step 4 involves the handling of symbolic configurations. This cannot be done by using the Turing machine's instruction table as it is, because the Turing instructions are defined for numeric tape configurations only, not for symbolic configurations. For this we need the SIP, a kind of meta-interpreter that enables the Turing machine to produce its own induction proof of being a nonhalter.

If the induction proof is successful, we know that the Turing machine M does not halt; hence, $\sigma(M) = 0$. The induction proof might fail because of the following.

- The hypothesis is wrong (it is just based on a finite amount of tape data).
- The proof does not terminate within the prespecified amount of CPU time (i.e., it might go through if we allow more CPU time).
- A situation is encountered beyond the implemented induction logic (we need to enhance the prover).
- A generalized Collatz $(3x+1)$ problem is encountered (see below).

We now turn to a more detailed discussion of the SIP.

■ The Symbolic Induction Prover

□ Meta-Instructions

The underlying idea is a unified and simple principle: analyze the Turing machine's instruction table and extract rules which describe meta-transactions on maximal invariant subtapes.

For example, let

$M =$

s / h	0	1
1	*	*
2	(0, *R*, 5)	*
3	(1, *L*, 3)	*
4	*	*
5	*	(0, *R*, 2)

Here is an example of a first-order meta-instruction:

$$\{\mathbf{3}, \{\text{gl}, \{\{0\}, n\}, \mathbf{0}, \text{gr}\}\} \to \{\mathbf{3}, \{\text{gl}, \mathbf{0}, \{\{1\}, n\}, \text{gr}\}\}\ \forall\ n \ge 1,$$

where gl is the symbol for a general left tape and gr is the symbol for a general right tape.

If at time t, M has configuration

$$c_t(n) = \{\mathbf{3}, \{\omega, \{\{x\}, 1\}, \{\{0\}, n\}, \mathbf{0}, \{\{y\}, 1\}, \omega\}\}$$

for some $x, y \in \Sigma^*$, we can apply the meta-instruction to get

$$c_{t+1}(n) = \{\mathbf{3}, \{\omega, \{\{x\}, 1\}, \mathbf{0}, \{\{1\}, n\}, \{\{y\}, 1\}, \omega\}\}.$$

Here is an example of a second-order meta-instruction:

$$\{\mathbf{2}, \{\text{gl}, \mathbf{0}, \{\{1, 0\}, n\}, \text{gr}\}\} \to \{\mathbf{2}, \{\text{gl}, \{\{0\}, 2\,n\}, \mathbf{0}, \text{gr}\}\}\ \forall\ n \geq 1.$$

Note that a meta-instruction modifies an arbitrarily large countable portion of the tape.

Induction Schemes

The SIP has a built-in library to support several induction proof schemes. Induction schemes are used in conjunction with meta-instructions to produce an induction proof for a nonhalting Turing machine. If necessary, the library can be enhanced with new induction schemes. We now discuss some of the implemented schemes.

Scheme: Commutation Relations at the Tape Boundary

Assume Turing machine M exhibits the symbolic machine configuration

$$c(k) = \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{q, 1\}, \omega\}\}$$

for some $x, q \in \Sigma^*$.

Verify: $M : \Omega \Longrightarrow c\,(k)$ for $k = 0$.

Assume: $M : c\,(k-1) \Longrightarrow c\,(k)$.

Prove by Induction: $M : c\,(k) \Longrightarrow c\,(k+1)$.

If true, M does not halt; hence, $\sigma\,(M) = 0$.

Induction Proof:
Let M be in the configuration

$$c(k) = \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{q, 1\}, \omega\}\}.$$

The SIP searches for maximal invariant boundary conditions and commutation relations.

Check If :

$$M : \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, 0\}, \{\tilde{q}, 1\}, \text{gr}\}\} \Longrightarrow \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, 1\}, \{\tilde{q}, 1\}, \text{gr}\}\},$$
$$\text{for some } \tilde{q} \in \Sigma^*, \text{ such that } \tilde{q} \text{ is } a \text{ prefix of } q.$$

If true, check further if $x \,||\, \tilde{q} = \tilde{q} \,||\, \tilde{x}$ for some $\tilde{x} \in \Sigma^*$; if this is also true, modify the induction assumption to read:

Extended Induction Hypothesis:

$$M : \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k-1\}, \{\tilde{q}, 1\}, \text{gr}\}\} \Longrightarrow \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{\tilde{q}, 1\}, \text{gr}\}\}.$$

The SIP proves this extended induction assumption as follows:

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{\tilde{q}, 1\}, \text{gr}\}\}$

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k-1\}, \{x, 1\}, \{\tilde{q}, 1\}, \text{gr}\}\}$

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k-1\}, \{\tilde{q}, 1\}, \{\tilde{x}, 1\}, \text{gr}\}\}$ (using the commutation relation)

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{\tilde{q}, 1\}, \{\tilde{x}, 1\}, \text{gr}\}\}$ (using the extended induction hypothesis for $(k-1) \mapsto (k)$ on any gr)

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{x, 1\}, \{\tilde{q}, 1\}, \text{gr}\}\}$ (using the commutation relation)

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k+1\}, \{\tilde{q}, 1\}, \text{gr}\}\}$ ■

Hence $M : c(k) \Longrightarrow c(k+1)$, $\forall\ k \geq 0$ and M does not halt, meaning that $\sigma(M) = 0$.

Scheme: Cyclic Conditions at the Tape Boundary

Assume Turing machine M exhibits the symbolic machine configuration

$$c(k) = \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{q, 1\}, \omega\}\},$$

for some $x, q \in \Sigma^*$.

Induction Base:
Verify $M : \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, 1\}, \text{gr}\}\} \Longrightarrow M : \{\mathbf{s}, \{\omega, \{y, 1\}, \mathbf{h}, \text{gr}\}\}$.

Induction Assumption:
$M : \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, j\}, \text{gr}\}\} \Longrightarrow M : \{\mathbf{s}, \{\omega, \{y, j\}, \mathbf{h}, \text{gr}\}\}$, $\forall\ j$ such that $1 \leq j \leq k-1$.

Assume further that the instruction table of M results in meta-instructions:

$\text{M1} : \{\mathbf{s}, \{\omega, \{y, n\}, \mathbf{h}, \{c_i, 1\}, \text{gr}\}\} \longrightarrow \{\mathbf{s}, \{\omega, \mathbf{h}, \{x, n\}, \{c_{i+1}, 1\}, \text{gr}\}\}$, $y, c_0, \ldots, c_m \in \Sigma^*$
$\text{M2} : \{\mathbf{s}, \{\omega, \{y, n\}, \mathbf{h}, \{c_m, 1\}, \text{gr}\}\} \longrightarrow \{\mathbf{s}, \{\omega, \{y, n+1\}, \mathbf{h}, \text{gr}\}\}$,

where $c_0, \ldots, c_m$ are the cyclic boundary conditions, with $c_0 = \{\}$.

Induction Proof:

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{q, 1\}, w\}\}$

$\{\mathbf{s}, \{\{\omega, \mathbf{h}, \{x, k-1\}, \{x, 1\}, \{q, 1\}, w\}\}$

$\{\mathbf{s}, \{\{\omega, \mathbf{h}, \{x, k-1\}, \mathrm{gr}\}\}$ (replacing $\{\{x, 1\}, \{q, 1\}, w\}$ by gr)

$\{\mathbf{s}, \{\omega, \{y, k-1\}, \mathbf{h}, \mathrm{gr}\}\}$ (first use of induction hypothesis)

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k-1\}, \{c_1, 1\}, \mathrm{gr}\}\}$ (using meta-instruction M1)

$\{\mathbf{s}, \{\omega, \{y, k-1\}, \mathbf{h}, \{c_1, 1\}, \mathrm{gr}\}\}$ (second use of induction hypothesis)

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k-1\}, \{c_2, 1\}, \mathrm{gr}\}\}$ (using meta-instruction M1)

$\vdots$

$\{\mathbf{s}, \{\omega, \{y, k-1\}, \mathbf{h}, \{c_{m-1}, 1\}, \mathrm{gr}\}\}$ $\left((m-1)^{\text{th}} \text{ use of induction hypothesis}\right)$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k-1\}, \{c_m, 1\}, \mathrm{gr}\}\}$ (using meta-instruction M1)

$\{\mathbf{s}, \{\omega, \{y, k\}, \mathbf{h}, \mathrm{gr}\}\}$ (use of induction hypothesis and use of M2)

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{c_1, 1\}, \mathrm{gr}\}\}$ (using meta-instruction)

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k\}, \{c_1, 1\}\{\{x, 1\}, \{q, 1\}, w\}\}$ (substituting back the explicit right tape portion)

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k+1\}, \{p, 1\}\{q, 1\}, w\}\}$ (this step might vary in other cases)

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k+1\}, \{q, 1\}\{0, p\}, w\}\}$ for some $p > 0$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, k+1\}, \{q, 1\}, w\}\}$ (subsuming the finite 0s into ω) ■

Scheme: Decreasing Cell Sequence at the Tape Boundary

Assume Turing machine M exhibits:

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n\}, \{\{B\}, 1\}, \{\{R\}, k\}, \mathrm{gr}\}\}$ $\quad k \geq 0,$ an integer

$\vdots$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n\}, \{\{B\}, 2\}, \{\{R\}, k-1\}, \mathrm{gr}\}\}$

$\vdots$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n\}, \{\{B\}, j\}, \{\{R\}, k-j\}, \mathrm{gr}\}\}$

Induction Base:

Verify that for some function f:

$$M : \{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n\}, \{\{B\}, 1\}, \{\{R\}, 1\}, \mathrm{gr}\}\} \longrightarrow \{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n + f(1)\}, \{\{B\}, 2\}, \mathrm{gr}\}\}$$

Verify the swap meta-instruction:

$$\{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n\}, \{\{B\}, k\}, \{\{R\}, 1\}, \mathrm{gr}\}\} \longrightarrow \{s, \{w, h, \{\{A\}, n + k_0\}, \{\{B\}, 1\}, \{\{R\}, k-1\}, \{\{B\}, 1\}, \mathrm{gr}\}\}$$

Induction Assumption:

$$M : \{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n\}, \{\{B\}, k-1\}, \{\{R\}, 1\}, \text{gr}\}\} \longrightarrow$$
$$\{s, \{w, h, \{\{A\}, n + f(k-1)\}, \{\{B\}, k\}, \text{gr}\}\}, \text{ for some function } f$$

Induction Proof:

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n\}, \{\{B\}, k\}, \{\{R\}, 1\}, \text{gr}\}\}$

$\downarrow$

$\{\mathbf{s}, \{\omega, \mathbf{h}, \{\{A\}, n + k_0\}, \{\{B\}, 1\}, \{\{R\}, k-1\}, \{\{B\}, 1\}, \text{gr}\}\}$ (apply swap meta-instruction)

$\downarrow$

$\left\{\mathbf{s}, \left\{\omega, \mathbf{h}, \left\{\{A\}, n + k_0 + \sum_{i=1}^{k-1} f(i)\right\}, \{\{B\}, k\}, \{\{R\}, 0\}, \{\{B\}, 1\}, \text{gr}\right\}\right\}$ (apply the induction assumption $k-1$ times)

$\downarrow$

$\left\{\mathbf{s}, \left\{\omega, \mathbf{h}, \left\{\{A\}, n + k_0 + \sum_{i=1}^{k-1} f(i)\right\}, \{\{B\}, k+1\}, \text{gr}\right\}\right\}$

Now, require that

$$k_0 + \sum_{i=1}^{k-1} f(i) = f(k)$$

and hence $f(k) = k_0\, 2^k$.

Similar schema are also supported:

$\{\mathbf{s}, \{\omega, \{\{A\}, n\}, \{\{B\}, k\}, \mathbf{h}, \{\{R\}, 1\}, \text{gr}\}\}$

$\downarrow$

$\{\mathbf{s}, \{\omega, \{\{A\}, n + k_0\}, \{\{B\}, 1\}, \mathbf{h}, \{\{R\}, k-1\}, \{\{X\}, 1\}, \text{gr}\}\}$ (apply swap meta-instruction)

and

$\{\mathbf{s}, \{\omega, \{\{A\}, n\}, \mathbf{h}, \{\{B\}, k\}, \{\{R\}, 1\}, \text{gr}\}\}$

$\downarrow$

$\{\mathbf{s}, \{\omega, \{\{A\}, n + k_0\}, \mathbf{h}, \{\{B\}, 1\}, \{\{R\}, k-1\}, \{\{X\}, 1\}, \text{gr}\}\}$ (apply swap meta-instruction)

Modular Arithmetic

Often meta-instructions are subject to some modular condition.

Here is an example:

$$M : \{\mathbf{s}, \{\text{gl}, \{1, n\}, \mathbf{h}, \text{gr}\}\} \longrightarrow$$
$$\left\{\boldsymbol{s}, \left\{\text{gl}, \{\{1\}, \text{Mod}[n, 3]\}, \mathbf{h}, \left\{\{0, 1, 0\}, 3\,\text{Floor}\left[\frac{n}{3}\right]\right\}, \text{gr}\right\}\right\}$$

If possible, the SIP calculates Mod[n, p] explicitly by using information about n.

If the variable is of the form $2n + 1$ and $p = 2$, then Mod[$2n + 1$, 2] = 1.

In general, the SIP uses modular arithmetic in the finite ring $\mathbb{Z}_n$, including Fermat's little theorem:

$$\text{Mod}[a^p, p] = a, \forall\ a \in \text{Integers}, \forall\ p \in \text{Primes}$$

and Euler's generalization:

$$\text{Mod}\left[a^{\varphi(n)}, n\right] = 1, \ \forall\ a, n \in \text{Integers}.$$

(Euler's phi function $\varphi(n)$ gives the number of positive integers less than or equal to n that are relatively prime to n.)

If nothing specific can be computed, the SIP picks a value $v \in \{0, \ldots, p - 1\}$ and generates p subproofs, one for each of the possible v values at each decision point in the general induction proof.

□ The Symbolic Induction Prover Package

The SIP includes the following major functional packages to support meta-instructions and symbolic induction proofs for Turing machine configurations.

`metaservices`	creates rules that represent meta-instructions
`swapservices`	handles all orders of meta-instructions
`shiftservices`	shifts configurations into normal form to make them comparable

Create and execute meta-instructions.

`inductionservices`	the main package for induction proof schemes
`tracebufferservices`	detects emerging schemes by tracing the steps of an induction proof, producing induction assumptions, and invokes the SIP recursively for proof

Handling of induction proofs.

`getgaugedvarservice`	handles modulo arithmetic for symbolic variables

Modulo arithmetic for symbolic variables.

`extremepointservices`	handles logic associated with maximal invariant tape boundaries

Handling boundary conditions.

`semaphoresupport`	provides services to serialize induction schemes to prevent "vicious circles"
`environmentservices`	specifies the proof environment (e.g., global parameters, etc.)

Technical services.

□ Results

Start with $\mathcal{TM}_5$, $\text{card}(\mathcal{TM}_5) = 21^{10}$.

Applying well-known [2] and fast techniques, such as tree normal form, backtracking, and simple loop detection, leaves about 1,000,000 Turing machines undecided.

- 850,000 Turing machines are of a linear type, for example, $\{\mathbf{s}, \{\omega, \mathbf{h}, \{x, a\ n + b\}, \omega\}\}$ for some $x \in \Sigma^*$, or slightly more complex and are proved by the SIP to not halt.
- 143,000 Turing machines are of polynomial or exponential configurations and are proved by the SIP to not halt.
- 1,000 Turing machines are currently too complex for the SIP.
  - 900 have been manually proven to not halt.
  - 100 cases are still holdouts and are currently under consideration, but there is strong evidence that they do not halt.
- 6,000 Turing machines halt within 50×10^6 steps.

Given that, we find that for the number of 1s (Σ) left on the tape, the number of cells scanned (Space), and the number of moves (Shift), the Marxen and Buntrock [3] machine is the champion

$$\begin{pmatrix} \{1, R, 2\} & \{1, L, 3\} \\ \{1, R, 3\} & \{1, R, 2\} \\ \{1, R, 4\} & \{0, L, 5\} \\ \{1, L, 1\} & \{1, L, 4\} \\ \{1, R, 0\} & \{0, L, 1\} \end{pmatrix}$$

with:

$$\begin{aligned} \Sigma(5) &= 4098 \\ \text{Shift}(5) &= 47{,}176{,}870 \\ \text{Space}(5) &= 12{,}289. \end{aligned}$$

The largest *unary* number (i.e., empty tape with a single block of consecutive 1s) produced by any halting binary 5-state Turing machine is Num(5) = 165.

The Num champion is:

$$\begin{pmatrix} \{1, R, 2\} & \{1, L, 1\} \\ \{1, R, 3\} & \{1, L, 5\} \\ \{1, R, 4\} & \{1, R, 5\} \\ \{0, L, 1\} & \{1, R, 3\} \\ \{1, R, 0\} & \{0, L, 2\} \end{pmatrix}$$

The possibility exists that the halting question for one or more of the remaining holdouts is linked to the Collatz $3x + 1$ problem (see also [11]). This is currently under investigation, and the final results will be reported in a forthcoming paper [4].

■ References

[1] T. Rado, "On Non-Computable Functions," *Bell System Technical Journal*, **41**, 1962 pp. 877-884.

[2] R. Machlin and Q. Strout, "The Complex Behavior of Simple Machines," *Physica D: Nonlinear Phenomena*, **42**, 1990 pp. 85-98, and references therein. www.eecs.umich.edu/~qstout/abs/busyb.html

[3] H. Marxen and J. Buntrock, "Attacking the Busy Beaver 5," *Bulletin of the European Association for Theoretical Computer Science*, **40**, 1990 pp. 247-251.

[4] J. Hertel, "The Computation of the Value of Rado's Non-Computable Σ Function for 5-State Turing Machines," work in progress.

[5] G. S. Boolos and R. C. Jeffrey, *Computability and Logic*, New York: Cambridge University Press, 1989.

[6] M. W. Green, "A Lower Bound on Rado's Sigma Function for Binary Turing Machines," in *Proceedings of the Fifth IEEE Annual Symposium on Switching Circuit Theory and Logical Design*, Princeton, NJ, 1964 pp. 91-94.

[7] B. A. Julstrom, "A Bound on the Shift Function in Terms of the Busy Beaver Function," *ACM Special Interest Group on Algorithms and Computation Theory*, **23**(3), 1992 pp. 100-106.

[8] E. W. Weisstein, "Busy Beaver" from Wolfram *MathWorld*—A Wolfram Web Resource. mathworld.wolfram.com/BusyBeaver.html

[9] H. J. M. Wijers, "Bibliography on the Busy Beaver Problem," 2004. www.win.tue.nl/~wijers/bbbibl.pdf

[10] N. J. A. Sloane. "The On-Line Encyclopedia of Integer Sequences, Sequence Id=A028444." oeis.org/search?q=A028444&language=english&go=Search

[11] P. Michel, "Busy Beaver Competition and Collatz-Like Problems," *Archive for Mathematical Logic*, **32**(5), 1993 pp. 351-367.

[12] A. M. Ben-Amrein, B. A. Julstrom, and K. Zwick, "A Note on Busy Beavers and Other Creatures," *Mathematical Systems Theory*, **29**(4), 1996 pp. 375-386.

[13] G. J. Chaitin, "Computing the Busy Beaver Function," *Open Problems in Communication and Computation* (T. M. Cover and B. Gopinath, eds.), New York: Springer-Verlag, 1987 pp. 108-112.

[14] K. Gödel, "Some Remarks on the Undecidability Results," *Collected Works Volume II, Publications 1938-1974*(S. Fefferman, J. W. Dawson, Jr., S. C. Kleene, G. H. Moore, and R. M. Soloway, eds.), New York: Oxford University Press, 1972/1990 pp. 305-306.

[15] S. Bringsjord, O. Kellett, A. Shilliday, J. Taylor, Bram van Heuvein, Y. Yang, J. Baumes, and K. Ross, "A New Gödelian Argument for Hypercomputing Minds Based on the Busy Beaver Problem," *Applied Mathematics and Computation*, **176**(2), 2006 pp. 516–530. DOI Link: dx.doi.org/10.1016/j.amc.2005.09.071

J. Hertel, "Computing the Uncomputable Rado Sigma Function," *The Mathematica Journal*, 2011. dx.doi.org/doi:10.3888/tmj.11.3–8.

About the Author

Joachim Hertel's main area of research includes quantum computation and the theory of computability. Hertel's special interest is in the general topic "Minds and Machines", particularly the question of what role Rado's Σ function and Chaitin's Ω number play in this context.

Joachim Hertel
50 Main St, STE 1000
White Plains, NY, 10606
jhertel@h-star.com