Hypercomputing Minds

New Numerical Evidence

Joachim Hertel

H-Star, Inc.

TKG 2020, Brno, January 13-15, 2020

Turing Machines vs. Gödel Minds

- **MIND = Turing Machine (TM) ?**

Turing Machines vs. Gödel Minds

- **MIND = Turing Machine (TM) ?**
- Let's recall some historical context....:

Turing Machines vs. Gödel Minds

- **MIND = Turing Machine (TM) ?**
- Let's recall some historical context....:
- Gödel, in his 1951 Gibbs lecture, delivered an opinion on Minds and Machines [1]

Turing Machines vs. Gödel Minds

- **MIND = Turing Machine (TM) ?**
- Let's recall some historical context....:
- Gödel, in his 1951 Gibbs lecture, delivered an opinion on Minds and Machines [1]

Turing Machines vs. Gödel Minds

- **MIND = Turing Machine (TM) ?**
- Let's recall some historical context....:
- Gödel, in his 1951 Gibbs lecture, delivered an opinion on Minds and Machines [1]

> *The human mind is incapable of mechanizing all its mathematical intuitions, i.e. if it has succeeded in formulating some of them, this very fact yields new knowledge.*

Turing Machines vs. Gödel Minds

- Gödel on Turing's proof that every mental procedure [....] is equivalent to a mechanical one, [2] :

Turing Machines vs. Gödel Minds

- Gödel on Turing's proof that every mental procedure [....] is equivalent to a mechanical one, [2] :
  - *Turing gives an argument which is supposed to show that mental procedures cannot carry any farther than mechanical procedures. However, this argument is inconclusive, because it depends on the supposition that a* **finite mind** *is capable of only a* **finite number of distinguishable** *states [...] although at each stage of the mind's development the number of* **possible states is finite**, *there is no reason why this number should not* **converge to infinity** *in the course its development.*

Turing Machines vs. Gödel Minds

Hao Wang: Gödel's notion of "*the number of mind's states converging to infinity*" is a complicated requirement [...] , [3] :

Problem

- *How can this be made more precise ?*

- There are too many functions f: N$\rightarrow$ N for them all to be (Turing) computable [4].

- There are too many functions f: N$\rightarrow$ N for them all to be (Turing) computable [4].
- In 1962, Rado [5] presented the uncomputable function Σ (aka the Busy Beaver function). Σ(n) is the largest number of 1 s left on the tape by a halting binary n-state Turing machine when started on an all 0-tape.

- There are too many functions f: N$\rightarrow$ N for them all to be (Turing) computable [4].
- In 1962, Rado [5] presented the uncomputable function Σ (aka the Busy Beaver function). Σ(n) is the largest number of 1 s left on the tape by a halting binary n-state Turing machine when started on an all 0-tape.
- The Σ function is uncomputable, because otherwise it would solve the Halting problem [4], which is known to be undecidable [4].

- There are too many functions f: N$\rightarrow$ N for them all to be (Turing) computable [4].
- In 1962, Rado [5] presented the uncomputable function Σ (aka the Busy Beaver function). Σ(n) is the largest number of 1 s left on the tape by a halting binary n-state Turing machine when started on an all 0-tape.
- The Σ function is uncomputable, because otherwise it would solve the Halting problem [4], which is known to be undecidable [4].
- It is known [6], [7] that: $\Sigma(1) = 1$ $\Sigma(2) = 4$ $\Sigma(3) = 6$ $\Sigma(4) = 13$ and $\Sigma(5) \geq 4098$

Bringsjord: A New Gödelian Argument

A quantified, measurable Gödelian Argument

- Let p denote Persons, m Turing Machines and let $cpl(m)$ measure the complexity of a Turing Machine in terms of states and transitions and let k be an Integer. According to Bringsjord we may state the **thesis of Computationalism** as: [8]

- Let p denote Persons, m Turing Machines and let $cpl(m)$ measure the complexity of a Turing Machine in terms of states and transitions and let k be an Integer. According to Bringsjord we may state the **thesis of Computationalism** as: [8]
- $\mathbf{[C]}\forall p \exists m(p = m \wedge cpl(m) \leq k)$

Bringsjord: A New Gödelian Argument

A quantified, measurable Gödelian Argument

- Let p denote Persons, m Turing Machines and let $cpl(m)$ measure the complexity of a Turing Machine in terms of states and transitions and let k be an Integer. According to Bringsjord we may state the **thesis of Computationalism** as: [8]
- $\mathbf{[C]}\forall p \exists m(p = m \wedge cpl(m) \leq k)$
- A quantified, measurable **New Gödelian Argument** was recently given by Bringsjord et al.,[8] and it is *based on the Rado* $\mathbf{\Sigma}$ – *Function*:

- Let p denote Persons, m Turing Machines and let $cpl(m)$ measure the complexity of a Turing Machine in terms of states and transitions and let k be an Integer. According to Bringsjord we may state the **thesis of Computationalism** as: [8]
- **[C]** $\forall p \exists m(p = m \wedge cpl(m) \leq k)$
- A quantified, measurable **New Gödelian Argument** was recently given by Bringsjord et al.,[8] and it is *based on the Rado* $\mathbf{\Sigma}$ *– Function*:
- **[A] If the human mind is able to compute $\mathbf{\Sigma(n)}$ it is able to eventually compute $\mathbf{\Sigma(n+1)}$.**

Bringsjord: A New Gödelian Argument

A quantified, measurable Gödelian Argument

- Bringsjord et al.,[8] have shown, that:

- Bringsjord et al.,[8] have shown, that:
- **[A]** $\Rightarrow \neg$**[C]** i.e. if assumption [A] holds, then Computationalism cannot hold

- Bringsjord et al.,[8] have shown, that:
- $\mathbf{[A]} \Rightarrow \neg\mathbf{[C]}$ i.e. if assumption [A] holds, then Computationalism cannot hold
- This concludes the philosophical context of our work

- Bringsjord et al.,[8] have shown, that:
- $\mathbf{[A]} \Rightarrow \neg\mathbf{[C]}$ i.e. if assumption [A] holds, then Computationalism cannot hold
- This concludes the philosophical context of our work
- We now proceed to present new **numerical evidence for Hypercomputing (Gödel) Minds**

Progress Report on Computing $\Sigma(5)$

Overview: What is known

- To keep notation simple we represent a 5-state binary Turing machine as a 5-by-2 matrix, the matrix elements are the transitional instructions that control the operation of the Turing head H on any given tape T.

Progress Report on Computing $\Sigma(5)$

Overview: What is known

- To keep notation simple we represent a 5-state binary Turing machine as a 5-by-2 matrix, the matrix elements are the transitional instructions that control the operation of the Turing head H on any given tape T.
- A 5-state binary Turing machine M is a 5-by-2 matrix M, such that $M(s, h) = (ws, mv, ns)$, with $s \in \{1, 2, 3, 4, 5\}$, $h \in \{0, 1\}$, $ws \in \{0, 1\}$, $mv \in \{L, R\}$, $ns \in \{0, 1, 2, 3, 4, 5\}$

Progress Report on Computing $\Sigma(5)$

Overview: What is known

- To keep notation simple we represent a 5-state binary Turing machine as a 5-by-2 matrix, the matrix elements are the transitional instructions that control the operation of the Turing head H on any given tape T.
- A 5-state binary Turing machine M is a 5-by-2 matrix M, such that $M(s, h) = (ws, mv, ns)$, with $s \in \{1, 2, 3, 4, 5\}$, $h \in \{0, 1\}$, $ws \in \{0, 1\}$, $mv \in \{L, R\}$, $ns \in \{0, 1, 2, 3, 4, 5\}$

- We call s the current state of M and h the current read symbol in the tape cell positioned under the Turing head H. The triple (ws, mv, ns) is called a Turing instruction with ws the write symbol 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.

Progress Report on Computing $\Sigma(5)$

Overview: What is known

- To keep notation simple we represent a 5-state binary Turing machine as a 5-by-2 matrix, the matrix elements are the transitional instructions that control the operation of the Turing head H on any given tape T.
- A 5-state binary Turing machine M is a 5-by-2 matrix M, such that $M(s,h) = (ws, mv, ns)$, with $s \in \{1,2,3,4,5\}$, $h \in \{0,1\}$, $ws \in \{0,1\}$, $mv \in \{L,R\}$, $ns \in \{0,1,2,3,4,5\}$

- We call s the current state of M and h the current read symbol in the tape cell positioned under the Turing head H. The triple (ws, mv, ns) is called a Turing instruction with ws the write symbol 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.
- That leaves us with 24^{10} possible binary 5-state Turing machines.

Progress Report on Computing $\Sigma(5)$

The Marxen-Buntrock Lower Bound

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

  is a 5-state Turing machine, published by Marxen and Buntrock, [9]. When started on an all-0-tape it halts after **47,176,870** steps and leaves **4098** 1's on the tape. Hence $\mathbf{\Sigma(5) \geq 4098}$

Progress Report on Computing $\Sigma(5)$

Known Reduction Methods

- Well-known methods exist, [6],[7] to reduce the search space and decide large number of TMs

Progress Report on Computing $\Sigma(5)$

Known Reduction Methods

- Well-known methods exist, [6],[7] to reduce the search space and decide large number of TMs
- Tree Normalform

Progress Report on Computing $\Sigma(5)$

Known Reduction Methods

- Well-known methods exist, [6],[7] to reduce the search space and decide large number of TMs
- Tree Normalform
- Back Tracking

Progress Report on Computing $\Sigma(5)$

Known Reduction Methods

- Well-known methods exist, [6],[7] to reduce the search space and decide large number of TMs
- Tree Normalform
- Back Tracking
- Simple Loop Detection

Progress Report on Computing $\Sigma(5)$

Known Reduction Methods

- Well-known methods exist, [6],[7] to reduce the search space and decide large number of TMs
- Tree Normalform
- Back Tracking
- Simple Loop Detection
- That leaves $1{,}676{,}482$ undecided 5-state binary TMs

Progress Report on Computing $\Sigma(5)$

New Approach: Tape Number Method

- Tape Numbers [4] are a way to encode the infinite Turing tape as two integers, the **l**eft **t**ap**e** **n**umber (ltpn) and the **r**ight **t**ap**e** **n**umber (rtpn)

Progress Report on Computing $\Sigma(5)$

New Approach: Tape Number Method

- Tape Numbers [4] are a way to encode the infinite Turing tape as two integers, the **l**eft **tap**e **n**umber (ltpn) and the **r**ight **tap**e **n**umber (rtpn)
- The binary representation of **ltpn** is given by the infinite portion of the tape to the **left** of the scanned cell

Progress Report on Computing $\Sigma(5)$

New Approach: Tape Number Method

- Tape Numbers [4] are a way to encode the infinite Turing tape as two integers, the **l**eft **tap**e **n**umber (ltpn) and the **r**ight **tap**e **n**umber (rtpn)
- The binary representation of **ltpn** is given by the infinite portion of the tape to the **left** of the scanned cell
- The binary representation of **rtpn** is given by the infinite portion of the tape **including the scanned cell** and to its **right** , written **backwards**

Progress Report on Computing $\Sigma(5)$

New Approach: Tape Number Method

- Tape Numbers [4] are a way to encode the infinite Turing tape as two integers, the **l**eft **tap**e **n**umber (ltpn) and the **r**ight **tap**e **n**umber (rtpn)
- The binary representation of **ltpn** is given by the infinite portion of the tape to the **left** of the scanned cell
- The binary representation of **rtpn** is given by the infinite portion of the tape **including the scanned cell** and to its **right** , written **backwards**
- 

Progress Report on Computing $\Sigma(5)$

New Approach: Tape Number Method

- Tape Numbers [4] are a way to encode the infinite Turing tape as two integers, the **l**eft **tap**e **n**umber (ltpn) and the **r**ight **tap**e **n**umber (rtpn)
- The binary representation of **ltpn** is given by the infinite portion of the tape to the **left** of the scanned cell
- The binary representation of **rtpn** is given by the infinite portion of the tape **including the scanned cell** and to its **right** , written **backwards**
(0)	0	1	1	0	1	0	**1**	1	0	1	0	1	1	1	(0)

- ltpn = '11010' = 26 rtpn = '1110101**1**' = 235 hence : T = (26,235)

Progress Report on Computing $\Sigma(5)$

Using TPNs: First Results

- Recording TPNs for many steps and applying pattern recognition we identify TM-specific recurrence relations for TPNs

Progress Report on Computing $\Sigma(5)$

Using TPNs: First Results

- Recording TPNs for many steps and applying pattern recognition we identify TM-specific recurrence relations for TPNs
- Using TM-specific recurrence relations allow us to perform automated Induction proofs to show a TM is a non-HALTER.

Progress Report on Computing $\Sigma(5)$

Using TPNs: First Results

- Recording TPNs for many steps and applying pattern recognition we identify TM-specific recurrence relations for TPNs
- Using TM-specific recurrence relations allow us to perform automated Induction proofs to show a TM is a non-HALTER.
- This way we decided **1,468,620** (out of **1,676,482**, **88%**) TMs to be non-HALTER

Progress Report on Computing $\Sigma(5)$

Using TPNs: First Results

- Recording TPNs for many steps and applying pattern recognition we identify TM-specific recurrence relations for TPNs
- Using TM-specific recurrence relations allow us to perform automated Induction proofs to show a TM is a non-HALTER.
- This way we decided **1,468,620** (out of **1,676,482**, **88%**) TMs to be non-HALTER
- We provide an example as follows:

Progress Report on Computing $\Sigma(5)$

Using TPNs: An Example

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

  this TM shows: in s = 3 tape = (4^{n+1} - 1,0);

  shorthand: <3, 4^{n+1} - 1, 0>

Progress Report on Computing $\Sigma(5)$

Using TPNs: An Example

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

  this TM shows: in s = 3 tape = (4^{n+1} - 1,0);
  shorthand: <3, 4^{n+1} - 1, 0>

- When started in s = 1 on tape = (0,0) , this TM reaches s =3 with tape = (4^1 - 1, 0) = (3,0) after 2 steps:
  Step 0 : <1,0,0>
  Step 1 : <2,1,0>
  Step 2 : **<3,3,0>, which establishes the case for n = 0.**

Progress Report on Computing $\Sigma(5)$

Using TPNs: Rules for Calculating Tape Numbers

- **Left Move**

  l odd : $l_{new} = \frac{l-1}{2}$, $r_{new} = 2r + 1$

  l even: $l_{new} = \frac{l}{2}$, $r_{new} = 2r$

Progress Report on Computing $\Sigma(5)$

Using TPNs: Rules for Calculating Tape Numbers

- **Left Move**

  l odd : $l_{new} = \frac{l-1}{2}$, $r_{new} = 2r + 1$

  l even: $l_{new} = \frac{l}{2}$, $r_{new} = 2r$

- **Right Move**

  r odd : $l_{new} = 2l + 1$, $r_{new} = \frac{r-1}{2}$

  r even: $l_{new} = 2l$, $r_{new} = \frac{r}{2}$

Progress Report on Computing $\Sigma(5)$

Using TPNs: The Induction Proof (1 of 2)

- **Induction Hypothesis**: after some finite number of steps, TM has reached $<\mathbf{3, 4^n - 1 , 0}>$ for $n > 0$

Progress Report on Computing $\Sigma(5)$

Using TPNs: The Induction Proof (1 of 2)

- **Induction Hypothesis**: after some finite number of steps, TM has reached $<\mathbf{3, 4^n - 1 , 0}>$ for $n > 0$
- **Induction Proof**: we have to show, that after some finite steps TM reaches $<\mathbf{3, 4^{n+1}-1, 0}>$

Progress Report on Computing $\Sigma(5)$

Using TPNs: The Induction Proof (2 of 2)

- $< 3, 4^{\mathbf{n}}. - 1, 0 >$

  $\downarrow$ fixed, 20 Steps

  $<2, 4^{n\text{-}k}\ \text{-}1, 2^{2k+1}\ \text{-}5>$

  $\downarrow$ 10 Steps for k $\rightarrow k+1$.

  $<2,\ 0,\ 2^{2n+1}\ \text{-}5>$

  $\downarrow$fixed, 5 Steps

  $<2,\ 0,\ 4^{n+1}\ \text{-}6>$

  $\downarrow$ *fixed*,15 Steps

  $<3,\ 4^{k+1-1}\ \text{-}1,\ 4^{n-k}\ \text{-}1>$

  $\downarrow$ *fixed*,4 Steps for $k \rightarrow k+1$

  $<\mathbf{3,\ 4^{n+1}\ \text{-}1,\ 0}>$

  q.e.d.

  Note: Length of Proof: 14n + 40 Steps

Progress Report on Computing $\Sigma(5)$

Using TPNs: Double Exponential Growth

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

Progress Report on Computing $\Sigma(5)$

Using TPNs: Double Exponential Growth

- $$\begin{pmatrix} (1,R,2) & (1,L,1) \\ (0,L,1) & (0,\mathbf{R},3) \\ (0,R,4) & (1,R,0) \\ (0,R,5) & (0,R,4) \\ (1,L,5) & (0,L,2) \end{pmatrix}$$
- **This TM exhibits a double exponential growth in the left TPN**

$$(1, 2^{2^{n+1}-4} - 1, 2)$$

Progress Report on Computing $\Sigma(5)$

Using TPNs: "Collatz-Type" Sequences in the Exponent

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

Progress Report on Computing $\Sigma(5)$

Using TPNs: "Collatz-Type" Sequences in the Exponent

- $$\begin{pmatrix} (1,R,2) & (1,R,4) \\ (1,L,3) & (1,R,2) \\ (0,L,4) & (0,L,4) \\ (1,R,5) & (0,R,2) \\ (1,R,1) & (1,R,0) \end{pmatrix}$$
- This TM exhibits the following recurrence relation for the right TPN:

Progress Report on Computing $\Sigma(5)$

Using TPNs: "Collatz-Type" Sequences in the Exponent

- $\begin{pmatrix} (1,R,2) & (1,R,4) \\ (1,L,3) & (1,R,2) \\ (0,L,4) & (0,L,4) \\ (1,R,5) & (0,R,2) \\ (1,R,1) & (1,R,0) \end{pmatrix}$
- This TM exhibits the following recurrence relation for the right TPN:
  - $(5, 1, 2^{a_n} - 2)$

Progress Report on Computing $\Sigma(5)$

Using TPNs: "Collatz-Type" Sequences in the Exponent

- $$\begin{pmatrix} (1,R,2) & (1,R,4) \\ (1,L,3) & (1,R,2) \\ (0,L,4) & (0,L,4) \\ (1,R,5) & (0,R,2) \\ (1,R,1) & (1,R,0) \end{pmatrix}$$
- This TM exhibits the following recurrence relation for the right TPN:
  - $(5, 1, 2^{a_n} - 2)$
- where the exponent follows

Progress Report on Computing $\Sigma(5)$

Using TPNs: "Collatz-Type" Sequences in the Exponent

- $\begin{pmatrix} (1,R,2) & (1,R,4) \\ (1,L,3) & (1,R,2) \\ (0,L,4) & (0,L,4) \\ (1,R,5) & (0,R,2) \\ (1,R,1) & (1,R,0) \end{pmatrix}$
- This TM exhibits the following recurrence relation for the right TPN:
  - $(5, 1, 2^{a_n} - 2)$
- where the exponent follows
  - $a_{n+1} = \begin{cases} 3*\frac{a_n}{2}+4 \text{ if } a_n \text{ even} \\ 3*\frac{a_n-1}{2}+2 \text{ if } a_n \text{ odd} \end{cases}$

Progress Report on Computing $\Sigma(5)$

Using TPNs: "Collatz-Type" Sequences in the Exponent

- $\begin{pmatrix} (1,R,2) & (1,R,4) \\ (1,L,3) & (1,R,2) \\ (0,L,4) & (0,L,4) \\ (1,R,5) & (0,R,2) \\ (1,R,1) & (1,R,0) \end{pmatrix}$
- This TM exhibits the following recurrence relation for the right TPN:
  - $(5, 1, 2^{a_n} - 2)$
- where the exponent follows
  - $a_{n+1} = \begin{cases} 3*\frac{a_n}{2}+4 \text{ if } a_n \text{ even} \\ 3*\frac{a_n-1}{2}+2 \text{ if } a_n \text{ odd} \end{cases}$
- with initial value

$$a_0 = 2$$

Progress Report on Computing $\Sigma(5)$

Summary

- Summary
  Using **tapenumbers** , we reported progress in calculation of $\Sigma(5)$ ($=$ 4098 most likely!)

Progress Report on Computing $\Sigma(5)$

Summary

- Summary
  Using **tapenumbers** , we reported progress in calculation of $\Sigma(5)$ (= 4098 most likely!)
- thus, we provided new numerical evidence and a tiny step towards validating Bringsjord **New Gödelian Argument,** [8]

Progress Report on Computing $\Sigma(5)$

Summary

- Summary
  Using **tapenumbers** , we reported progress in calculation of $\Sigma(5)$ ($=$ 4098 most likely!)
- thus, we provided new numerical evidence and a tiny step towards validating Bringsjord **New Gödelian Argument,** [8]
- ... and hence a bit more evidence for **Gödel's Hypercomputing Minds**.

Progress Report on Computing $\Sigma(5)$

Summary

- Summary
  Using **tapenumbers** , we reported progress in calculation of $\Sigma(5)$ ($=$ 4098 most likely!)
- thus, we provided new numerical evidence and a tiny step towards validating Bringsjord **New Gödelian Argument,** [8]
- ... and hence a bit more evidence for **Gödel's Hypercomputing Minds**.
- **Thank You!**

References

[1] S.Feferman et al. (eds.), Kurt Gödel Collected Works Vol. III,304-323, Oxford, 1995

[2] S.Feferman et al. (eds.), Kurt Gödel Collected Works Vol. II,306, Oxford, 1995

[3] H. Wang, A Logical Journey, Cambridge,1996,200-201

[4] S.Boolos, R.C.Jeffrey, Computability and Logic, Cambridge, 1989

[5] T.Rado, Bell System Technical Journal, 41, 1962, 877-884

[6] P.Michel, Arch. Math. Logic, 32, 1993, 351-367

[7] R.Machlin, Q.Strout, Physica D, 42, 1990, 85-98, and references therein

[8] S.Bringsjord et al., Applied Mathematics and Computation, 176,2006,516-530

[9] H.Marxen, J.Buntrock, Bull. EATACS, 40, 1990, 247-251