

An injection from $\mathbb{N}^{\mathbb{N}}$ to \mathbb{N}

Andrej Bauer

Faculty of Mathematics and Physics

University of Ljubljana

Andrej.Bauer@andrej.com

June 15, 2011

Abstract

We provide a realizability model in which there is an injection from the internal Baire space $\mathbb{N}^{\mathbb{N}}$ to the the natural numbers object \mathbb{N} .

1 Introduction

At the Mathematical Foundations of Programming Semantics meeting in May 2011, Paulo Oliva and Martín Escardó showed a program which witnessed the fact that there was no injection from the Baire space $\mathbb{N}^{\mathbb{N}}$ to natural numbers \mathbb{N} . The program took as input a function $h : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ and produced two sequences $x, y \in \mathbb{N}^{\mathbb{N}}$ such that $x \neq y$ and $h(x) = h(y)$. Martín Escardó popularized the program as interesting example of extraction of computational content from classical proofs, which lead me to wonder whether there was a constructive proof of the statement

$$\forall h : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N} . \exists x, y \in \mathbb{N}^{\mathbb{N}} . (x \neq y \wedge h(x) = h(y))$$

that would yield such programs more directly. Fred Richman asked for a constructive proof of the weaker statement that there was no injection $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$, and nobody could come up with a proof.

Classically there is no such injection, of course. Constructively, it is easy to see that it must be wildly discontinuous, if it exists. Thus we cannot hope to find one in any of the usual varieties of constructive mathematics, as they all satisfy some kind of continuity principle.

The main and only result of this note is that there is a realizability topos based on infinite time Turing machines [2] in which there is an injection $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$. It is likely that the topos can be used for other ominous purposes. For example, it validates the principle LPO but its logic is not classical.

Acknowledgment. I thank Joel Hamkins for explaining infinite time Turing machines to me, to Alex Simpson for helpful discussions, and to Jaap van Oosten for doubting that such a model exists.

2 Infinite time Turing machines

For details about infinite time Turing machines we recommend [2], here we only give a brief overview.

An infinite time Turing machine, or just *machine*, is like a Turing machine which is allowed to run infinitely long, where the computation steps are counted by ordinals. The machine has a finite program, an input tape, work tapes, an output tape, etc. We assume that the tape cells contain 0's and 1's. At successor ordinals the machine acts like an ordinary Turing machine. At limit ordinals it enters a special “limit” state, its heads are placed at the beginnings of the tapes, and the content of each tape cell is computed as the limsup of the values written in the cell at earlier stages. More precisely, if c_α denotes the value of the cell c at step α , then for a limit ordinal β we have

$$c_\beta = \begin{cases} 0 & \text{if } \exists \alpha < \beta. \forall \gamma. (\alpha \leq \gamma < \beta \Rightarrow c_\alpha = 0), \\ 1 & \text{otherwise.} \end{cases}$$

The machine terminates by entering a special halt state, or it may run forever. It turns out that a machine which has not terminated by step ω_1 runs forever.

We can think of machines as computing partial functions $2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$: we initialize the input tape with an infinite binary sequence $x \in 2^{\mathbb{N}}$, run the machine, and observe the contents of the output tape if and when the machine terminates. We can also consider infinite time computation of partial functions $\mathbb{N} \rightarrow \mathbb{N}$: we initialize the input tape with the input number, run the machine, and interpret the contents of the output tape as a natural number, where we ignore anything that is beyond the position of the output head. By performing the usual encoding tricks, we can feed the machines more complicated inputs and outputs, such as pairs, finite lists, and even infinite lists of numbers or sequences. We say that a function is *infinite time computable* if there is a machine that computes it.

The power of infinite time Turing machines is vast and extends far beyond the halting problem for ordinary Turing machines, although of course they cannot solve their own halting problem. For example, for every Π_1^1 -

subset $S \subseteq 2^{\mathbb{N}}$ there is a machine which, given $x \in 2^{\mathbb{N}}$ on its input tape, terminates and decides whether $x \in S$.

There is a standard enumeration t_0, t_1, t_2, \dots of infinite time Turing machines, where t_n is the machine whose program is encoded by the number n in some reasonable manner. The associated enumeration $\psi_0, \psi_1, \psi_2, \dots$ of infinite time computable partial functions $\mathbb{N} \rightarrow \mathbb{N}$ is defined as

$$\psi_n(k) = \begin{cases} m & \text{if } t_n \text{ on input } k \text{ terminates and outputs } m, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The enumeration ψ satisfies the s-m-n and u-t-m theorems.

Theorem 2.1 (s-m-n) *There is a total infinite time computable map $s : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ such that $\psi_{s(m,i)}(j) = \psi_m(\langle i, j \rangle)$ for all $m, i, j \in \mathbb{N}$.*

Theorem 2.2 (u-t-m) *There is a partial infinite time computable map $u : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ such that $\psi_n(k) = u(n, k)$ for all $n, k \in \mathbb{N}$.*

To convince ourselves that the u-t-m theorem holds, we think a bit how a universal infinite time Turing machine works. It accepts the description of a machine n and the initial input tape x . At successor steps the simulation of machine t_n on input x proceeds much like it does for the ordinary Turing machines. Thus it takes finitely many successor steps to simulate one successor step of t_n . Each limit step of t_n is simulated by one limit step of the universal machine, followed by finitely many successor steps. Indeed, whenever the universal machine finds itself in the special limit state, it updates the state of the simulated machine to reflect the fact that a limit state has been reached, which takes finitely many steps. Also, at limit steps the contents of the simulated tapes get updated just as they would if we actually ran t_n .

To see what sort of tasks can be performed by infinite time Turing machines, we consider several examples that will be useful later on.

There is a machine which decides whether two infinite sequences $x, y \in \mathbb{N}^{\mathbb{N}}$ are equal. It first initializes a fresh work cell with 0, and then for each k , it compares x_k and y_k . If they differ, it sets the work cell to 1. After ω steps the work cell will be 1 if, and only if, $x \neq y$.

A more complicated problem is to *semidecide* whether a given machine t_n computes a given sequence $x \in 2^{\mathbb{N}}$. The machine which performs such a task accepts n and x as inputs and begins by writing down the sequence $y_k = \psi_n(k)$ onto a work tape. This it can do by simulating t_n successively on inputs $0, 1, 2, \dots$ and writing down the values y_k as they are obtained. If

any of the y_k 's is undefined, the machine will run forever. Otherwise it will be able to detect that the entire sequence y has been computed and written down, simply by checking in ω steps that each one of them is really there on the tape. After that, the machine verifies that $x_k = y_k$ for all $k \in \mathbb{N}$, as described previously.

Suppose we have a machine t which takes as input an infinite sequence x and a number n . We would like to construct another machine which accepts an infinite sequence x and outputs a number n such that $t(x, n)$ terminates, if one exists. We use the familiar dovetailing technique to tackle the problem. Given $x \in 2^{\mathbb{N}}$ as input, we simulate in parallel the executions of machine t on inputs of the form (x, n) , one for each n :

$$t(x, 0), \quad t(x, 1), \quad t(x, 2), \quad \dots$$

Each of these requires several infinite tapes, but since we only need countably many of them, they may be interleaved into a single tape. At successor steps the simulation performs the usual dovetailing technique. At limit steps the simulation inserts extra ω bookkeeping steps, during which it places the simulated machines in the “limit” state and moves their head positions. The extra steps do not ruin the limits of the simulated tapes, because those are left untouched. After the extra steps are performed, dovetailing starts over again. As soon as one of the simulations $t(x, n)$ terminates, we return the results n . Note that n is computed from x in a deterministic fashion, although a different simulation technique may yield a different n .

3 Realizability over infinite time Turing machines

For background on realizability theory we refer to [3]. To build a realizability model from infinite time Turing machines, we first need to turn them into a partial combinatory algebra (PCA). In view of the s-m-n and u-t-m theorems this is no problem at all. The combinator K is obtained by an application of the s-m-n theorem to the first projection $\langle i, j \rangle \mapsto i$, while the combinator S requires a bit more work and the use of the u-t-m theorem. In honor of the inventor of infinite time Turing machines we denote by \mathbb{J} the PCA whose underlying set is \mathbb{N} and the partial application is defined by $m \cdot n = \psi_m(n)$.

In the next section we will show that the realizability topos $\text{RT}(\mathbb{J})$ contains an injection $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$. In fact, we only need to consider a much simpler realizability model of *numbered sets* over \mathbb{J} . These are equivalent to a full subcategory of $\text{RT}(\mathbb{J})$ which contains the natural numbers object \mathbb{N} and the internal Baire space $\mathbb{N}^{\mathbb{N}}$.

Recall that a numbered set (S, δ) is a set S with a partial surjection $\delta : \mathbb{N} \rightarrow S$. If (T, η) is another numbered set, we say that a map $f : S \rightarrow T$ is infinite time computable with respect to δ and η if there exists an infinite time computable map $r : \mathbb{N} \rightarrow \mathbb{N}$ such that $\text{dom}(\delta) \subseteq \text{dom}(r)$ and $f(\delta(n)) = \eta(r(n))$ for all $n \in \text{dom}(\delta)$. We say that r *tracks* f .

The natural numbers object in $\text{RT}(\mathbb{J})$ is the numbered set $\mathbb{N} = (\mathbb{N}, \text{id}_{\mathbb{N}})$, while the internal exponential $\mathbb{N}^{\mathbb{N}}$ is the set

$$\{f : \mathbb{N} \rightarrow \mathbb{N} \mid f \text{ is infinite-time computable}\}$$

of infinite time computable total function, with the numbering ψ restricted to the codes of total maps.

4 An injection $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ in $\text{RT}(\mathbb{J})$

To show that our realizability model has an injection $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$, we first formulate a constructive plan of attack.

Proposition 4.1 *Suppose the following hold:*

1. *choice from functions to numbers, and*
2. *$\mathbb{N}^{\mathbb{N}}$ is the image of a subcountable set.*

Then there is an injection $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$, constructively.

Proof. The second condition means that there is a partial surjection $s : \mathbb{N} \rightarrow \mathbb{N}^{\mathbb{N}}$. For all $f \in \mathbb{N}^{\mathbb{N}}$ there exists $n \in \mathbb{N}$ such that $s(n) = f$. By function choice there exists $h : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ such that $s(h(f)) = f$ for all $f \in \mathbb{N}^{\mathbb{N}}$. Clearly, h is injective because it is a section of s . \square

The second condition is satisfied in $\text{RT}(\mathbb{J})$. The partial surjection $e : \mathbb{N} \rightarrow \mathbb{N}^{\mathbb{N}}$ is defined as

$$e(n) = \begin{cases} \psi_n & \text{if } \psi_n \text{ is total} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

and is tracked by the identity map. Its domain of definition is the set of codes of total infinite time computable maps. The map e is surjective in the internal logic of the topos because it is surjective and tracked by the identity.

Choice from functions to numbers is also known as $AC_{1,0}$. It states that any total relation between $\mathbb{N}^{\mathbb{N}}$ and \mathbb{N} contains a function. We show that $\text{RT}(\mathbb{J})$ satisfies an even stronger principle, namely general choice for functions: every total relation on $\mathbb{N}^{\mathbb{N}}$ contains a function. In categorical terms this amounts to $\mathbb{N}^{\mathbb{N}}$ being internally projective, see [3, 3.2.3].

Proposition 4.2 *The object $\mathbb{N}^{\mathbb{N}}$ is internally projective if, and only if, there exists an infinite time computable map $r : \mathbb{N} \rightarrow \mathbb{N}$ such that:*

1. *if ψ_k is total then $r(k)$ is defined and $\psi_k = \psi_{r(k)}$, and*
2. *if ψ_k is total then $r(r(k)) = r(k)$.*

Proof. See e.g. [3, 3.2.3] or [1, 1.3.4]. □

We describe informally how a machine computing r works. Suppose k is the code of a total function ψ_k (our machine will diverge if k is not the code of a total function). The machine first writes down the sequence $x_i = \psi_k(i)$ onto a tape. Because x_k is total, it will eventually detect that x has been written down completely. After that it searches in parallel for $m \in \mathbb{N}$ such that ψ_m computes x . This can be done because it is semidecidable whether a given m computes x . One such m will be found eventually because x is computed by ψ_k . The number m depends only on x , thus it is a canonical realizer for ψ_k .

Notice that the map r just described tracks an injection $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$, so we could have constructed such an injection directly, without knowing anything about internally projective sets. Nevertheless, it is still interesting to know that $\text{RT}(\mathbb{J})$ validates function choice, while at the same time $\mathbb{N}^{\mathbb{N}}$ is the image of a subcountable set.

References

- [1] A. Bauer. *The realizability approach to computable analysis and topology*. PhD thesis, Carnegie Mellon University, 2000.
- [2] J. D. Hamkins and A. Lewis. Infinite time turing machines. *Journal of Symbolic Logic*, 65(2):567–604, 2000.
- [3] Jaap van Oosten. *Realizability: An Introduction to its Categorical Side*, volume 152 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2008.