An Arbitrarily Self-validating Configuration of Turing Machines

Mark Inman, Ph.D.
mark.inman@egs.edu

November 26, 2016

Abstract

We present a top level description for the configuration of self-validating Turing machines modified with parallel inputs and based on Turing's $\mathscr{H}$ Machine as he describes in his paper, "On Computable Numbers, with an Application to the Entscheidungsproblem." This novel description allows such a machine configuration to solve for β' by introducing a parallel computing component and memory check procedure to Turing's original conception of an $\mathscr{H}$ Machine. Solving for β' is RE-complete and has implications for complexity class relationships.

1 Overview

1.1 Context

This article presents a top level description for the configuration of modified $\mathscr{H}$ Machines with parallel inputs and the ability to self-verify. While this article is not entirely self-contained, we have limited our outside references of significance to just Turing's original paper, "On Computable Numbers, with an Application to the Entscheidungsproblem," which is also the only reference necessary to understand this article. So, to avoid any confusions that might arise from material written since Turing's source document, and to eliminate any iota of external influence and bias, we have chosen to use original terminology and reference descriptions as they are found in Turing's paper. This means that, while we accept his work as true, we have reason to believe it is incomplete; that it does not account for the machine configuration we describe in this paper. Thus, we will consider Turing's primary results on the Entschiedungsproblem as if they have not yet been universally accepted as true for all cases until the novel case presented in this article is reviewed. If we find no discrepency between his results and ours, this will be a double confirmation of his results through our unique method employing the concept of Turing's machine. If a discrepency is found, then we should spend time to re-evaluate our notions of computability and related arguments such as Cantor's Diagonal Method and countability.

1.2 Preliminary Considerations

By convention, the terms *Circular Machine* and *Circle-free Machine* are not typically used, as we conveniently may refer to the halting or infinite loops of programs, where infinite loops are unsatisfactory and halting is satisfactory. However, in our circumstance, it is convenient to note here that a *Circular Machine* is deemed by Turing to be unsatisfactory due to forever looping, redundantly over a repeating pattern. Also, a *Circle-free Machine* is deemed by Turing to be satisfactory because it's ability to continue deciding indefinitely, without entering an infinite loop. We have chosen to keep Turing's original terminology for the sake of clarity when comparing the work of this article to that of Turing's original paper. This same reasoning for our choice of nomenclature, is employed for all subsequent terminology which may no longer be used by only a matter of convention as well. [2]

A *Standard Description* or S.D. is the rule set for any given Turing Machine $\mathscr{M}$ in a standard form. By creating a standard, the rule sets themselves can be used to create a *Description Number* or D.N. which itself may be readable by a Universal Turing Machine, $\mathscr{U}$, as an instruction set. [2]

From Turing's paper: "Let $\mathscr{D}$ be the Turing Machine which when supplied with the Standard Description (S.D.) of any computing machine $\mathscr{M}$ will test this S.D. and if $\mathscr{M}$ is circular will mark the S.D. with the symbol "*u*" and if it is circle free, will mark it with '*s*' for 'unsatisfactory' and 'satisfactory' respectively. By combining machines $\mathscr{D}$ and $\mathscr{U}$, we could construct a machine $\mathscr{H}$ to compute the sequence of $\beta^{'}$" [2]

$\mathscr{H}$ is circle free by construction and if it weren't for Turing's findings, we could accept the machine as circle free intutively and right away. However, there is nothing intuitive about the Entscheidungsproblem, nor it's solution. [2]

The only additional prerequisite needed for this article is that the reader accepts the Church-Turing thesis that says every function which is calculable, is a computable function. It is not in the scope of this paper to re-invent the wheel and give an exhaustive description of Turing's machines. We could very well choose to use a "black box machine" that has the same expressive power and limitations as the Universal Turing Machine and this article should still hold. There are many online and free resources with in depth and clear descriptons, as well asvideos of working physical models of Turing Machines which are widely available for the public to view and understand their expresive power. For an in depth understanding of Turing Machines, the Universal Turing Machine and the history behind it's development, I recomend one reads *The Annotated Turing* by Charles Petzold. [1]

1.3 Turing's Claim

In the eighth section of Turing's paper on the Entscheidungsproblem, Turing claims that $\beta^{'}$ can not be determined vis a vis the following reason:

"The instructions for calculating the R(K)-th [figure] would amount to 'calculate the first R(K)-th figures computed by $\mathscr{H}$ and write down the R(K)-th'. This R(K)-th would never be found. I.e. $\mathscr{H}$ is circular..." [2]

This is because, since $\mathscr{H}$ relies on certain subroutines to make it's determination, when it reaches and tries to evaluate K, it must call itself, which provides instructions on reading inputs from 1 to K-1 in order to call the R(K)-th figure, but it can never get

there, because it keeps repeating it's own instruction loop. [2] The question is, however, is there a Turing Machine which can recognize itself arbitrarily[1] when it reaches it's own Description Number (D.N.), such that some $\mathscr{H}$ machine configuration prints $\beta^{'}$?

2 Self-validating Computers

2.1 Supermachine

Now, let us consider that $\mathscr{H}^{'}$ is a controller machine with a D.N. of $K^{'}$. It controls two different $\mathscr{H}$ machines: $\mathscr{H}_0$ and $\mathscr{H}_1$. $\mathscr{H}_0$ and $\mathscr{H}_1$ each have the ability to determine "u" or "s" on a D.N. input, except $\mathscr{H}_0$ tests as Turing describes, from D.N. 1 counting upwards (Each D.N. is a natural number) and $\mathscr{H}_1$ tests from a certain twos complement of whatever number is being tested by $\mathscr{H}_0$ as a simultaneous parallel input, such that it's subsequent D.N. is one less than the previously tested D.N. Let us represent each D.N. by some integer i. $\mathscr{H}_0$ and $\mathscr{H}_1$ have a unique D.N. of K_0 or K_1 respectively.[2]

Upon input of i_0 to be read by $\mathscr{H}_0$, let $\mathscr{H}^{'}$ store the value pair (i_0, z) until i_0 is determined to be satisfactory or unsatisfactory. When the output is determined, let $\mathscr{H}^{'}$ replace the (i_0, z) with the respective (i_0, s) or (i_0, u) in the data store, such that there is no longer a data store of (i_0, z). Let the same process occur for i_1, such that $\mathscr{H}^{'}$ also initially stores each D.N. input with (i_1, z) and $\mathscr{H}_1$ reads i_1 to determine satisfactory or unsatsifactory, subsequently replacing the initial value pair with the respective value pair (i_1, s) or (i_1, u) depending on the output of $\mathscr{H}_1$. A *redundancy* occurs when some $i_0 = i_1$.

Let $\mathscr{H}^{'}$ have the ability to compare value pairs such that the machine may recognize a redundancy when it occurs, and may also recognize when a value pair contains a z value on the condition of such a redundancy. Let's call this a *z-check* ability.

Let $\mathscr{H}_s$ be the supermachine that is the configuration of all three $\mathscr{H}$ Machines as described above and let K_s be the D.N. for the supermachine.

Initialize the identifier strings such that $K_1 < K_0$.

Let the number of bits in $K_0 = n$. Let the twos complement of the first D.N. input to $\mathscr{H}_0$, which is 1, be determined by n such that it satisifies the equation $c = 2^n - 1$.

Lemma. *K_1 is decided by $\mathscr{H}_0$, K_0 is decided by $\mathscr{H}_1$, and $\mathscr{H}_s$ proceeds taking inputs circle free, or at least until it reads K_s.* If $c - K_0 > K_1$, then re-initialize the D.N.[3] in either K_0 or K_1 such that $c - K_0 < K_1$. This guarantees that $\mathscr{H}_0$ will read K_1 before $\mathscr{H}_1$ reads K_1 and also guarantees $\mathscr{H}_1$ will read K_0 before $\mathscr{H}_0$ reads K_0. Let the controller $\mathscr{H}^{'}$ contain a memory command which stores the decision value pairs of $\mathscr{H}_0$ and $\mathscr{H}_1$. The controller may routinely check for a redundancy on the next input.

[1] by recognizing itself arbitrarily, we mean that it can recognize it's own program or description number even if K is not fixed as the program continues to run. Also, there may not exist an initializer that feeds a fixed K to be recognized by a single read instruction that skips K and just rubber stamps approval. Such "rubber stamping" is considered a trivial case and is not of concern to this article.

[2] This can be determined through a unique identifier string, which does not effect The machine's function or performance, but differentiates the two machines from each other giving them each a unique D.N.

[3] one may re-initialize, if necessay, the D.N. by adding irrelevent description information into some S.D. yielding a different D.N. provided such information does not affect the integrity of the original S.D.

Now consider when $\mathscr{H}_0$ reads K_1, and K_1 calls the D.N. for $\mathscr{H}_0$: $\mathscr{H}_0$ will call K_1 which will result in a z-check, recognizing that the value pair (K_1, z) is already stored in memory, and therefore, since $K_1 < c$, we know that K_1 is the description number for the other $\mathscr{H}$ machine, which is $\mathscr{H}_1$, and we can mark it as satisfactory. This same reasoning can be applied for when $\mathscr{H}_1$ reads K_0.

If however, the machine has determined a redundancy occured on a value pair where the value is either (i, s) or (i, u) (i.e., the z-check is negative, but the redundancy check is positive), then we have already evaluated this D.N. from the other $\mathscr{H}$ machine, and we no longer have to continue within the range 1 to c, since they will all have been decided. The supermachine, at this point proceeds to utilize machine $\mathscr{H}_0$ and proceeds from D.N. input value $c+1$, and continues through the rest of all Description Numbers, $c+2$, $c+3$, etc... at least until it reaches it's own D.N., K_s, for no other D.N. should be problematic in determining the output decision. Thus, $\mathscr{H}_s$ proceeds circle free, at least until it reaches K_s which is easily constructed to be larger than c. ∎

2.2 β' is Decidable

Proof. β' *is Decidable.* At the point K' is received as an input, it is determined satisfactory by either $\mathscr{H}_0$ or $\mathscr{H}_1$. Neither $\mathscr{H}_0$ nor $\mathscr{H}_1$ are called during this phase of the process.

By lemma, K_0 is decided by $\mathscr{H}_1$, K_1 is decided by $\mathscr{H}_0$ and $\mathscr{H}_s$ continues indefinitely until we reach K_s, which describes $\mathscr{H}_s$. K_s is read by $\mathscr{H}'$ and as before, it's Description Number is stored along with it's temporary pair value of z until $\mathscr{H}_0$ or $\mathscr{H}_1$ returns a value for β' at that location. K_s is sent to be verified by $\mathscr{H}_0$, which when $\mathscr{H}'$ calls K_s for a second time, under the given recursive property of K_s which will eventually call itself, the z-check for value pair (K_s, z) is recognized as both redundant and with a z value, stored by $\mathscr{H}'$ in the data store, but because the associated value is z,the z-check ability tells us this process has already occurred, sends K_s to $\mathscr{H}_1$, which self-verifies repeated z-check values. By construction, the only value K_i which can provide this multiple z-check values where $K_i > c$ is K_s, so $\mathscr{H}_s$ now self-verifies the input K_s as it's own D.N., provides a value of "s" for satisfactory, and moves to evaluate $K_s + 1$ to continue indefinitely as a *Circle-free Turing Machine.*

Therefore, β' is decidable over the set of all Description Numbers given some $\mathscr{H}_s$ Machine. As a direct consequence,

...

$RE \subseteq PSPACE$ ∎

References

[1] Charles Petzold *The Annotated Turing* 2008: Wiley Publishing, Indianapolis, IN.

[2] A. M. Turing *"On Computable Numbers, with an Applcation to the Entscheidungsproblem"* 1936-1937: Proceedings of the London Mathematical Society, Series 2, 42, pp 230?265.