# On Consistently Guessing the Output of Algorithms

Andrew Drucker*

Scott Aaronson [Aar] suggested the study of the following computational problem:

**Given:** a description $\langle Q \rangle$ of an (input-free) Turing machine $Q$, syntactically restricted to produce at most 1 output bit;

**Output:** 1 if $Q$ outputs 1; 0 if $Q$ outputs 0. Otherwise (if $Q$ loops), we may output either 0 or 1.

Aaronson calls this the *consistent guessing (CG)* problem; a familiar diagonalization argument shows that it is incomputable [Aar]. Aaronson asked whether any oracle solving CG can be used to solve the Halting problem (HALT). In this note we give a negative answer to Aaronson's question. In fact, the answer is negative even if HALT is replaced by *any* incomputable language.

This result is not really new; it is a special case of a more general result. The only property of CG that we need in our proof, is that the collection of solutions to CG can be expressed as the set of *infinite paths* in a *computable binary tree* (definitions will follow). Such a collection is referred to as a $\Pi_1^0$ *class*. It is known that, for any $\Pi_1^0$ class $P$ and any incomputable language $L$, there is a solution to P that cannot be used to compute $L$. The techniques we use in this note (which are well-known) also yield a proof of this more general result with no additional effort. See the survey [Cen] for various strengthenings and variants of this result, and for more information on $\Pi_1^0$ classes.

First we need to review a bit of standard notation and formalize our assertion. All Turing machines we consider will be syntactically restricted to output at most 1 bit. We will use the letter $Q$ to denote an input-free Turing machine, and $M$ to denote an input-accepting one. Say that

*CSAIL, MIT. Email: adrucker@mit.edu

a language $L \subseteq \{0,1\}^*$ is a *solution to CG* if, for all input-free, *halting* machines $Q$,

$$\langle Q \rangle \in L \quad \iff \quad Q \text{ outputs 1.}$$

Thus, we place no constraint on $\chi_L(\langle Q \rangle)$ when $Q$ is looping. Let $CG^{sol}$ denote the set of solutions to CG.

Let $L(M)$ denote the set of strings accepted by $M$. Let $M^B$ denote the machine $M$ equipped with oracle access to $B$. For languages $A, B$, write $A \leq_T B$ if $A$ Turing-reduces to $B$. That is,

$$A \leq_T B \quad \iff \quad \exists M : L(M^B) = A.$$

We show the following:

**Theorem 1.** *Let $L$ be incomputable; then there exists a set $A \in CG^{sol}$, such that $L \not\leq_T A$.*

*Proof.* By a *(binary) tree*, we mean a subset $T \subseteq \{0,1\}^*$ that is closed under prefixes. That is, if $x \in T$ and $x'$ is a prefix of $x$, then $x' \in T$. A *computable tree* is just one whose characteristic function is computable.

An *infinite path* in $T$ is an infinite string $p \in \{0,1\}^\omega$ whose finite prefixes are all in $T$. We use an important, simple fact about trees:

**Fact 1** (König's Lemma)**.** *If $T$ is an infinite binary tree, then $T$ contains an infinite path.*

Now let $L$ be incomputable; we will construct $A \in CG^{sol}$ such that $L \not\leq_T A$. Our construction will proceed in stages. On each stage $i \geq 0$ we will define a computable tree $T_i$, such that the following requirements are met:

**R0:** All infinite paths $p$ in $T_0$ satisfy $p \in CG^{sol}$;

**R1:** $T_i \subseteq T_{i-1}$, for $i = 1, 2, \ldots$;

**R2:** Each $T_i$ contains an infinite path;

**R3:** For $i \geq 1$, let $M_i$ be the $i^{th}$ machine in a standard universal enumeration of (input-accepting) Turing machines. Then we have that for all infinite paths $p$ in $T_i$, $L(M_i^p) \neq L$.

Under our requirements, it is not hard to see that the intersection $T_\omega := \bigcap_{i \geq 0} T_i$ is an infinite tree. Letting $p$ be an infinite path in $T_\omega$, **R0** guarantees that $p \in CG^{sol}$, and **R3** guarantees that $L \not\leq_T p$. Thus, constructing $T_0, T_1, \ldots$ will prove the Theorem.[1]

Let $Q_1, Q_2, \ldots$ be a standard enumeration of input-free Turing machines. Building $T_0$ is simple: for a string $v \in \{0,1\}^n$, let $v \in T_0$ iff the following condition holds: for all $j \leq n$, if $Q_j$ halts with output $b$ in at most $n$ steps, then $v_j = b$.

$T_0$ is clearly a computable tree. We verify **R0** and **R2** (the other two requirements don't apply to $T_0$). It is easily checked that **R0** holds. That **R2** holds is easy to see: for any $p \in CG^{sol}$, all prefixes of $p$ lie in $T_0$, so $p$ is an infinite path in $T_0$.

Now let $i \geq 1$, and assume $T_0, \ldots, T_{i-1}$ have been constructed satisfying **R0**-**R3**. To construct $T_i$, we will need the following notion. A finite string $v$ can be considered as an "incomplete oracle", giving values for oracle queries to strings up to a certain index. For an input-accepting oracle machine $M$ and string $x$, let $M^v(x) := b$ if on input $x$ and oracle $v$, $M$ outputs $b$ without ever querying a string whose oracle-value is left undefined by $v$. Otherwise, we let $M^v(x)$ be undefined.

To define $T_i$, we will first define an infinite set of "candidate trees"

$$\{\ T_{i,y}\ \}_{y \in \{0,1\}^*}\ .$$

$T_i$ will be chosen as one of these. For any $y \in \{0,1\}^*$, let $T_{i,y}$ be defined as follows. Let $v \in T_{i,y}$ iff the following conditions both hold:

(a) $v \in T_{i-1}$;

(b) The computation $M_i^v(y)$ either is undefined, or does not halt within $n = |v|$ steps, *or*, halts within $n$ steps with output $1 - \chi_L(y)$.

Each $T_{i,y}$ is clearly a computable tree. It is also immediate that **R1** is satisfied for any choice of $T_i = T_{i,y}$.

If we can choose $y$ so that $T_i = T_{i,y}$ satisfies **R2**, then **R3** will hold as well, since for any infinite path $p$ in $T_{i,y}$ we will have $M_i^p(y) \neq \chi_L(y)$. We claim that a $y$ must exist; proving this assertion will give us a suitable choice of $T_i = T_{i,y}$, extending our construction to stage $i$ and completing the proof of the Theorem.

[1] (Note: although each $T_i$ will be individually computable, the sequence $\{T_i\}$ we construct will not be *uniformly* computable: that is, there will not be an algorithm to decide whether $v \in T_i$, given $i$ and $v$ as inputs.)

Suppose for contradiction's sake that no such $y$ exists. We claim that then $L$ is computable, contrary to our initial assumption. Let $P_{i-1}$ be an algorithm to decide membership in $T_{i-1}$. Here is our algorithm to decide if $y \in L$:

**Algorithm $P_L(y)$:**

1. Set $n := 1$.
2. Enumerate all length-$n$ strings in $T_{i-1}$, using $P_{i-1}$. Let $v[n,1], \ldots, v[n,m]$ be these strings. For each $j \leq m$, simulate $M_i^{v[n,j]}(y)$ for $n$ steps. If all of these simulations halt with a common output $b$ (without ever querying a value left undefined by the oracle), halt and output $b$; otherwise, set $n := n + 1$ and repeat Step 2.

We claim that for any $y$, $P_L(y)$ halts with output $\chi_L(y)$. To see this, let $p$ be an infinite path in $T_{i-1}$; such a $p$ exists, since $T_{i-1}$ satisfies **R2**. By our assumption, $p$ is not an infinite path in $T_{i,y}$; suppose that for the length-$N$ prefix $p[1,\ldots,N]$ we have $p[1,\ldots,N] \notin T_{i,y}$. By definition of $p$, this implies that $M_i^{p[1,\ldots,N]}(y)$ halts in at most $N$ steps with output $\chi_L(y)$.

Now $T_{i-1}$ is a tree, so all prefixes of $p$ are also in $T_{i-1}$. For no such prefix $p'$ can the computation $M_i^{p'}(y)$ halt with output $1 - \chi_L(y)$. Thus, on no stage $n$ can $P_L(y)$ halt with output $1 - \chi_L(y)$.

The tree $T_{i,y}$ contains no infinite path, so by König's Lemma it is finite. Say that it contains no strings of length $N' > 0$. Reasoning as above, we find that for each string $v \in T_{i-1} \cap \{0,1\}^{N'}$ we have $M_i^v(y) = \chi_L(y)$. Thus, after at most $N'$ stages, $P_L(y)$ halts with output $\chi_L(y)$. So $P_L$ computes $L$ as claimed. As argued earlier, this completes the proof of the Theorem. $\square$

# References


[Aar] Scott Aaronson. Rosser's Theorem via Turing Machines (blog post). `http://www.scottaaronson.com/blog/?p=710`

[Cen] Douglas Cenzer. $\Pi_1^0$ classes in Computability Theory. Handbook of Computability (ed. E. Griffor), North-Holland Studies in Logic 140 (1999), 37-85. Draft available at `http://www.math.ufl.edu/~cenzer/research_html/n42.ps`