Extending Hibbard's intelligence measure by delegating hard choices to the agent

Samuel Alexander*

2020

Abstract

When studying the theory of agents with Artificial General Intelligence (the theory of AGIs), one can sometimes avoid arbitrary decisions by delegating those decisions to the AGIs under investigation. In 2011, Hibbard suggested an intelligence measure for AGIs acting as predictors in adversarial sequence prediction games. Hibbard's measure depends arbitrarily on a function-sequence imported from a paper by Liu. We argue that Hibbard's measure is trivial: any genuine AGI should have Hibbard intelligence ∞, because the functions in Liu's paper are trivial to a genuine AGI. Our key insight is this: rather than arbitrarily choosing a function-sequence, we can delegate that choice to an AGI. In this way, we propose non-trivial variations of Hibbard's original measure, which we call simple Hibbard measures. However, simple Hibbard measures are still flawed. To repair their flaws, we further propose what we call transfinite Hibbard measures, which depend on an AGI choosing a slow-growing hierarchy (from mathematical logic) rather than a function-sequence.

1 Introduction

Todo:

1. Fix the subscripts in generalintelligencemeasuredefn to start at 0.
2. Better formal encoding of slow-growing hierarchies.
3. Formal theorem suggesting THM/IOI connection.
4. References about adversarial sequence prediction etc (including Gamez).
5. More background about fast- and slow-growing hierarchies.
6. In Example bruteforcenottotallyoptimalexample, we choose a function which the AGI does not know. Add a lemma showing that such functions must exist.

*The U.S. Securities and Exchange Commission

7. Rewrite 'caution' in such a way as to emphasize Reinhardt-style ignorance.
8. A lemma to the effect that if X knows M is a Turing machine which computes f at all but finitely many points, then there exists a TM M' which computes f everywhere and such that X knows M' is total.
9. Fix the definition of t_e to work for predictors as well as evaders.
10. Add a remark about temporal nature of command responses.
11. Redefine simple Hibbard measures so that the agent is commanded to enumerate faster-and-faster growing sequences. Point out a different flaw, related to non-Archimedeanness. This will better justify the slow-growing hierarchy. Put more emphasis on the fact that if X knows Y's predictor is total then X is smarter than Y.

At a high-level, this paper is about a meta-technique in theoretical research on agents with Artificial General Intelligence (that is, on AGIs), namely: avoiding difficult and arbitrary decisions by delegating those decisions to the very AGIs we are studying. To speak metaphorically, suppose we were fighting about what clothes the Oracle of Delphi should wear: such fighting could be avoided by asking the oracle herself what clothes an Oracle of Delphi should wear. At a low-level, this paper is about a certain specific intelligence measure, and improvements thereon: these serve as a kind of proving ground for the higher-level meta-technique.

In his insightful paper [6], Bill Hibbard introduces a novel intelligence measure (which we will here refer to as the *original Hibbard measure*) for AGIs. Hibbard's measure is based on "the game of adversarial sequence prediction against a hierarchy of increasingly difficult sets of" evaders (environments that attempt to emit 1s and 0s in such a way as to evade prediction). The levels of Hibbard's hierarchy are labelled by natural numbers[1], and an agent's original Hibbard measure is the maximum $n \in \mathbb{N}$ such that said agent learns to predict all the evaders in the nth level of the hierarchy, or implicitly[2] an agent's original Hibbard measure is ∞ if said agent learns to predict all the evaders in all levels of Hibbard's hierarchy.

In this paper, we will argue that the original Hibbard measure is trivial: we will argue that *all* suitably idealized AGIs should have original Hibbard intelligence ∞. This is because Hibbard's hierarchy depends on a function-sequence which is (we would argue) arbitrarily chosen, and which happens to consist of functions which should be trivial to a genuine AGI.

[1] Technically, Hibbard's hierarchy begins at level 1 and he separately defines what it means for an agent to have intelligence 0, but that definition is equivalent to what would result by declaring that the 0th level of the hierarchy consists of the empty set of evaders.

[2] Hibbard does not explicitly include the ∞ case in his definition, but in his Proposition 3 he refers to agents having "finite intelligence", and it is clear from context that by this he means agents who fail to predict some evader somewhere in the hierarchy.

We will suggest two different methods to repair Hibbard's intelligence measure.

- The first method we describe will be elementary, in that it will essentially involve no mathematics more advanced than the mathematics used in Hibbard's original approach. This will yield a non-trivial family of intelligence measures, which we will call *simple Hibbard measures*, but these will still suffer a flaw.
- The second method we describe will make use of *slow-growing hierarchies* [13] (an advanced notion from mathematical logic) and will produce a family of ordinal-number-valued intelligence measures, which we will call *transfinite Hibbard measures*, to avoid the above-mentioned flaw.

Both of these methods will, rather than producing a single intelligence measure, rather, produce a family of intelligence measures indexed by a parameter X where X is an AGI. Every AGI X will have a corresponding simple Hibbard measure $|\bullet|_X$ (for any AGI Y, $|Y|_X$ will be a natural-number-valued measurement of Y's intelligence "according to X"). Every AGI X will also have a corresponding transfinite Hibbard measure $\|\bullet\|_X$ (for any AGI Y, $\|Y\|_X$ will be a computable-ordinal-valued measurement of Y's intelligence "according to X").

The structure of the paper is as follows.

- In Section 2, we present an idealized viewpoint of AGIs.
- In Section 3, we review the original Hibbard measure.
- In Section 4, argue that under certain idealizing assumptions, *every* AGI probably has original Hibbard intelligence ∞.
- In Section 5, we delegate an arbitrary choice (which Hibbard originally made by importing from a paper by Liu) to an AGI, thereby obtaining what we call *simple Hibbard measures*.
- In Section 6, we review the *slow-growing hierarchy*.
- In Section 7, we use slow-growing hierarchies to define what we call *transfinite Hibbard measures* (delegating the choice of a slow-growing hierarchy to an AGI).
- In Section 8, we summarize and make concluding remarks.

2 An idealized viewpoint of AGIs

> "Thus it is a property of man[-like intelligence] to be capable of learning grammar; for if he is a man[-like intelligence], then he is capable of learning grammar, and if he is capable of learning grammar, then he is a man[-like intelligence]." —Aristotle [4] (modified)

As yet, we are unaware of any formal mathematical definition of what an AGI actually is. We take the stance that an AGI should be thought of like a faithful employee which can understand practical conversation in a common human language (such as English) and capable of (and faithfully willing to) obey commands expressed by its employer in said language. So if the AGI's employer said, "Recite the digits of pi until I tell you to stop," then the AGI would begin computing and reciting digits of pi, until the employer told it to stop (if ever). Unlike a human employee, who would eventually get fed up with such a task, an AGI should have no fear of tedium, and should never grow tired of performing whatever task it has been assigned (we assume the AGI never runs out of memory and can indeed run forever, just like a Turing machine is routinely assumed to be able to run forever and have infinite tape).

We think assumptions similar to the above are implicit in many attempts to study AGIs. Many authors who set out to study AGIs actually study not the full AGI, but rather the AGI's persona when it is performing some general type of task. For example:

- In the Turing Test, the AGI is reduced to a conversation partner pretending to be human. An AGI is more than a pretend human conversation partner; but an AGI would act as a pretend human conversation partner if commanded to do so.
- In Hibbard's paper [6], the AGI is reduced to a predictor in an adversarial sequence prediction game. An AGI is more than a predictor; but an AGI would act as such a predictor if commanded to do so.
- Some authors [8] [5] reduce the AGI to a reinforcement learning agent. An AGI is more than a reinforcement learning agent; but an AGI would act as a reinforcement learning agent if commanded to do so.
- Some philosophers reduce the AGI to a mere *knower* who performs no actions but simply knows things in some formal language, or, equivalently, to a formal theory (namely the list of all the things it knows). An AGI is more than a mere knower or theory [10]; but an AGI would act as a mere knower (and give rise to a resulting theory) if commanded to do so.
- In the Non-Axiomatic Reasoning System research program (NARS) [11], it is hoped that an AGI can be realized via the implementation of what is, essentially, a special type of database capable of being bombarded by many (possibly conflicting) assertions in a certain formal language and of outputting probabilistic answers to realtime queries about said assertions. An AGI is more than a mere database; but an AGI would act as such a database if commanded to do so.

In the same way that IQ tests are intended to measure human intelligence despite the fact that humans are more than professional IQ-test-takers, likewise, the above cross-sections of AGI serve to illuminate AGI even though none of them capture the full AGI.

Assumption 1. *We make the following idealizing assumptions about every AGI X.*

- *(Basic mathematical exposure) X knows the axioms of ZFC (the axiomatic system underlying mainstream mathematics).*
- *(Mathematical truthfulness) If X knows some statement in the language of ZFC, then that statement is true.*
- *(Mathematical reasoning ability) If X knows some set S of mathematical axioms then, given sufficient time to think about it, X would eventually become aware of all the consequences of S.*
- *(Caution) X avoids infinite loops: If X is commanded to carry out any task which involves outputting things forever, and if the task is not impossible (e.g., does not require solving the Halting Problem or doing other non-computable or nonsensical things), then X will continue outputting things (perhaps after long intervals of time between outputs) indefinitely.*

3 Hibbard's original measure

Hibbard's intelligence measure is based on the AGI's performance against many opponents in a game of adversarial sequence prediction (we define this formally below). The AGI acts as a predictor p which competes against evaders e. In each step of the game, both predictor and evader simultaneoulsy choose a binary digit, 1 or 0. Only after both of them have made their choice do they see which choice the other one made, and then the game proceeds to the next step. The predictor's goal in each step of the game is to choose the same digit that the evader will choose; the evader's goal is to choose a different digit than the predictor. The predictor wins the game (and is said to *learn to predict* e) if, after finitely many initial steps, eventually the predictor always chooses the same digit as the evader.

Definition 2. *By B, we mean the binary alphabet $\{0,1\}$. By B^*, we mean the set of all finite binary sequences. By $\langle\rangle$ we mean the empty binary sequence.*

Definition 3. *(Predictors and evaders)*

1. *By a* predictor*, we mean a Turing machine p which takes as input a finite (possibly empty) binary sequence $(x_1, \ldots, x_n) \in B^*$ (thought of as a sequence of* evasions*) and outputs* 0 *or* 1 *(thought of as a* prediction*), which output we write as $p(x_1, \ldots, x_n)$.*
2. *By an* evader*, we mean a Turing machine e which takes as input a finite (possibly empty) binary sequence $(y_1, \ldots, y_n) \in B^*$ (thought of as a sequence of* predictions*) and outputs* 0 *or* 1 *(thought of as an* evasion*), which output we write as $e(y_1, \ldots, y_n)$.*

3. *For any predictor p and evader e, the* result of p playing the game of adversarial prediction against e *(or more simply, the* result of p playing against e*) is the infinite binary sequence $(x_1, y_1, x_2, y_2, \ldots)$ defined as follows:*

   (a) *The first evasion $x_1 = e(\langle\rangle)$ is the output of e when run on the empty prediction-sequence.*

   (b) *The first prediction $y_1 = p(\langle\rangle)$ is the result of applying p to the empty evasion-sequence.*

   (c) *For all $n > 0$, the $(n+1)$th evasion $x_{n+1} = e(y_1, \ldots, y_n)$ is the output of e on the sequence of the first n predictions.*

   (d) *For all $n > 0$, the $(n+1)$th prediction $y_{n+1} = p(x_1, \ldots, x_n)$ is the result of applying p to the first n evasions.*

4. *Suppose $r = (x_1, y_1, x_2, y_2, \ldots)$ is the result of a predictor p playing against an evader e. For every $n \geq 1$, we say* the predictor wins round n in r *if $x_n = y_n$; otherwise,* the evader wins round n in r*. We say that p* learns to predict e *(or simply that p* learns e*) if there is some $N \in \mathbb{N}$ such that for all $n > N$, p is the winner of round n in r.*

The following definition makes explicit a subtlety which is implicit in Hibbard's original paper.

Definition 4. *Suppose X is an AGI. By X's* predictor*, we mean the predictor which X would act as if X were commanded to act as a predictor as in Definition 3.*

In the following definition, we differ from Hibbard's original paper because of a minor (and fortunately, easy-to-fix) error there.

Definition 5. *For any evader e and $n \in \mathbb{N}$, let $t_e(n)$ be the maximum number of steps that e takes to run on any length-n sequence of binary digits. In other words, $t_e(0)$ is the number of steps e takes to run on $\langle\rangle$, and for all $n > 0$,*

$$t_e(n) = \max_{b_1,\ldots,b_n \in \{0,1\}} (\textit{number of steps } e \textit{ takes to run on } (b_1, \ldots, b_n)).$$

Example 6. *Let e be an evader. Then $t_e(2)$ is equal to the number of steps e takes to run on input $(0,0)$, or to run on input $(0,1)$, or to run on input $(1,0)$, or to run on input $(1,1)$—whichever of these four possibilities is largest.*

Definition 7. *(The evader-set bounded by a function) Suppose $f : \mathbb{N} \to \mathbb{N}$. We define the* evader-set bounded by f*, written E_f, to be the set of all evaders e such that there is some $k \in \mathbb{N}$ such that $\forall n > k$, $t_e(n) < f(n)$.*

In the following broad definition, for every function-sequence of functions from $\mathbb{N}$ to $\mathbb{N}$, we define a corresponding intelligence measure for AGIs. The original Hibbard measure is a special case for one specific such function-sequence.

Definition 8. *(The general Hibbard measure given by a function-sequence) Suppose $(g_1, g_2, \ldots)$ is a function-sequence, each $g_i : \mathbb{N} \to \mathbb{N}$. For each $m \in \mathbb{N}$, define $f_m : \mathbb{N} \to \mathbb{N}$ by*

$$f_m(k) = \max_{0<i\leq m} \max_{j\leq k} g_i(j).$$

For any AGI X, we define the Hibbard intelligence of X given by $(g_1, g_2, \ldots)$*, written $|X|_{(g_1,g_2,\ldots)}$, as follows. Let p be X's predictor (Definition 4). We define $|X|_{(g_1,g_2,\ldots)}$ to be the maximum $m > 0$ such that p learns to predict e for every $e \in E_{f_m}$ (or 0 if there is no such m, or ∞ if p learns to predict e for every $e \in E_{f_m}$ for every m).*

Definition 9. *The* original Hibbard measure *is defined to be the Hibbard intelligence measure given by one specific family $(g_1, g_2, \ldots)$ of functions, namely: Liu's [9] enumeration of the primitive recursive functions.*

4 Do all AGIs have original Hibbard intelligence ∞?

In this section, we will argue that under certain idealizing assumptions, every AGI should have original Hibbard intelligence ∞. Thus, we would suggest that Definition 9 is really only appropriate for certain weak AIs who are strong enough that they can be induced to participate in adversarial sequence prediction games, but who are too weak to qualify as genuine AGIs.

4.1 A semi-optimal predictor

The most obvious way to try to predict an evader would be to enumerate some list $M_1, M_2, \ldots$ of Turing machines and sequentially try to predict e by assuming e computes the same function as M_1, and if that ever fails, then predict e by assuming e computes the same function as M_2, and if that ever fails, then predict e by assuming e computes the same function as M_3, and so on. In doing so, we should limit ourselves to Turing machines M_i which define total functions, lest we end up waiting forever for a non-terminating computation and never getting around to making our next prediction. Unfortunately, the set of total Turing machines is not computably enumerable, so the best we can hope for is to enumerate some subset of the total Turing machines. But which subset? This is a tricky and arbitrary decision—thankfully, we do not need to decide it ourselves, we can let the AGI decide for us.

Definition 10. *(Brute-Force Predictors) Suppose X is an AGI. Let $M_1, M_2, \ldots$ be the list of Turing machines which X would enumerate if X were commanded: "Enumerate all the Turing machines which you know define total functions from B^* to B". By X*'s brute-force predictor*, we mean the predictor p defined as follows.*

1. *Initially, p shall attempt to predict the evader e by assuming that e defines the same function as M_1. When (if ever) this fails, say that* M_1 is ruled out from being the evader.

2. *Once M_1 has been ruled out from being the evader (if ever), p shall attempt to predict e by assuming that e defines the same function as M_2. When (if ever) this fails, say that* M_2 is ruled out from being the evader.

3. *Once M_2 has been ruled out from being the evader (if ever), p shall attempt to predict e by assuming that e defines the same function as M_3. When (if ever) this fails, say that* M_3 is ruled out from being the evader.

4. *And so on forever...*

Note that X's brute-force predictor is total precisely because of the mathematical truthfulness assumption (from Definition 1): whenever X knows that M_i defines a total function from B^* to B, M_i really *does* define a total function from B^* to B (hereafter, we will suppress remarks like this and routinely use the mathematical truthfulness assumption without explicit mention).

Lemma 11. *Let X be an AGI and let e be an evader (so e is a Turing machine which computes a function from B^* to B). Let M be any Turing machine which computes the same function from B^* to B as e computes. If X knows that M defines a total function from B^* to B, then X's brute-force predictor learns to predict e.*

Proof. Let p be X's brute-force predictor. Let $M_1, M_2, \ldots$ be as in Definition 10. Since X knows that M defines a total function from B^* to B, it follows that $M = M_k$ for some k. When p plays against e, it cannot occur that M_k is ruled out from being the evader, because in order for that to occur, p would have to fail at predicting e when p assumes that e computes the same function as M_k, but that cannot fail because that assumption is true. Since M_k cannot be ruled out from being the evader, it follows that step $k+1$ of Definition 10 will never be reached, which in turn implies that p stops failing at predicting e after finitely many initial failures, in other words, p learns to predict e. □

For any AGI X, we would like to claim that X's brute-force predictor is optimal among all predictors X could act as; unfortunately this is not true in the strongest sense it possibly could be true, as the following example shows.

Example 12. *Let X be an AGI. Let $f : \mathbb{N} \to B$ be a total computable function which is not computed*[3] *by any Turing machine M such that X knows X computes a total function from $\mathbb{N}$ to B. For each $i \in \{0, 1\}$, let e_i be an evader such*

[3]Such an f must exist because otherwise, X could enumerate all the total computable functions from $\mathbb{N}$ to B, which is absurd because those functions are not computably enumerable.

that:

$$e_i(\langle\rangle) = i,$$
$$e_i(y_1, \ldots, y_n) = \begin{cases} i & \text{if } y_1 = \cdots = y_n = i, \\ f(n) & \text{otherwise.} \end{cases}$$

For each $i \in \{0, 1\}$, let p_i be a constantly-i predictor, $p_i(x_1, \ldots, x_n) = i$. Then p_0 learns to predict e_0 and p_1 learns to predict e_1, but X's brute-force predictor learns to predict at most one of e_0 or e_1.

Proof. Clearly the result of p_0 playing against e_0 is $(x_0, y_0, x_1, y_1, \ldots) = (0, 0, 0, 0, \ldots)$, so p_0 learns to predict e_0. Likewise, the result of p_1 playing against e_1 is $(x_0, y_0, x_1, y_1, \ldots) = (1, 1, 1, 1, \ldots)$, so p_1 learns to predict e_1. It remains to show that X's brute-force predictor p cannot learn to predict both e_0 and e_1.

Let $M_1, M_2, \ldots$ be as in Definition 10. Let $g : B^* \to B$ be the function computed by M_1.

Case 1: $g(\langle\rangle) = 1$. Then when p plays against e_0, after the first step, it will never be the case that $y_1 = \cdots = y_n = 0$ (because $y_1 = g(\langle\rangle) = 1$). Thus, for all $n > 0$, $e_0(y_1, \ldots, y_n) = f(n)$. I claim that p does not learn e_0. To see this, assume (for the sake of contradiction) that p learns e_0. It follows that there is some k (which we may take as small as possible) such that M_k never gets ruled out from being the evader (Definition 10). Let $h : B^* \to B$ be the function defined by M_k. Let $(x_0, y_0, x_1, y_1, \ldots)$ be the result of p playing against e_0. It follows that there is some j such that for all $i > j$, $h(y_1, \ldots, y_{i-1}) = x_i = y_i = f(i)$. This shows that for all but finitely many $i \in \mathbb{N}$, $f(i) = h(y_1, \ldots, y_{i-1})$. Since X knows that M_k defines a total function from B^* to B, it follows that there is a Turing machine M such that M computes f and X knows M computes a total function from $\mathbb{N}$ to B. Absurd, this contradicts our choice of f.

Case 2: $g(\langle\rangle) = 0$. Then by similar reasoning as in Case 1, it can be shown that p does not learn e_1. □

Example 12 shows that we cannot hope for the brute-force to be totally optimal in the most extreme possible sense: there will always be evaders that the AGI's brute-force predictor fails to learn, but which other predictors (which the AGI is capable of acting as) do nevertheless learn. Example 12 involves highly contrived evaders which are custom-made to act stupidly in one specific case (allowing them to be learned by stupid predictors), while being highly sophisticated in other cases. In the following definition, we rule out situations where a less sophisticated predictor manages to learn a more sophisticated evader due to the evader concealing its true sophistication from the predictor.

Definition 13. *Suppose p is a predictor and e is an evader. We say that p* learns *e* but-not-without-a-fight *if the following conditions hold:*

1. *p learns e.*
2. *For each i, $t_p(i) \geq t_e(i)$.*

Armed with Definition 13, we can now state a theorem which shows the semi-optimality of the brute-force predictor.

Theorem 14. *(Semi-optimality of brute force) Let X be an AGI and let p be any predictor. If X knows that p computes a total function from B^* to B, and if p learns an evader e but-not-without-a-fight, then X's brute-force predictor learns e.*

Proof. Since X knows p computes a total function from B^* to B, X knows that e' is a total computable function from B^* to B, where e' is the Turing machine which takes an input $b \in B^*$ and operates as follows:

1. Calculate $t_p(i)$ (by running p on all length-i binary sequences).
2. Run e on b for up to $t_p(i)$ steps. If e outputs a result x within that time, then output x. Otherwise, output 0.

Since p learns e but-not-without-a-fight, each $t_p(i) \geq t_e(i)$, so in fact e' computes the same function as e. By Lemma 11, X's brute-force predictor learns e'. Since e' and e compute the same function, this implies X's brute-force predictor learns e. □

4.2 Triviality of the original Hibbard measure

Although Example 12 showed that an AGI X's brute-force predictor is not optimal in the strongest possible sense, Theorem 14 shows that X's brute-force predictor is still semi-optimal. In some sense, X's brute-force predictor learns every evader which any other predictor p (that X knows is a predictor) would learn, except possibly for cases where p only learns an evader e because e conceals its full sophistication from p.

Since we only used basic mathematics to prove Theorem 14, any genuine AGI should also eventually figure out Theorem 14. Thus, when an AGI is commanded to compete as a predictor in the adversarial sequence prediction game, the resulting predictor which the AGI acts as should be at least as good as the brute-force predictor, because why would the AGI act as a worse predictor if it can figure out that brute-force is semi-optimal? We formalize this in the following assumption.

Assumption 15. *For every AGI X, X's predictor is at least as good as X's brute-force predictor, by which we mean that for every evader e, if X's brute-force predictor learns e, then X's predictor learns e.*

Based on Assumption 15, we are now ready to argue that the original Hibbard measure is trivial: that it assigns intelligence level ∞ to every AGI.

Theorem 16. *Every AGI X has original Hibbard intelligence ∞.*

Proof. Let p be X's brute-force predictor, and let q be X's predictor. By Definition 8, in order to show X has original Hibbard intelligence ∞, we must

show that q learns to predict e for every e in E_{f_m} for every $m \in \mathbb{N}$, where each $f_m : \mathbb{N} \to \mathbb{N}$ is defined by $f_m(k) = \max_{0<i\leq m} \max_{j\leq k} g_i(j)$, where g_i is the ith primitive recursive function in Liu's enumeration (Definition 9). By Assumption 15, it suffices to show that q learns all such e, for then Assumption 15 says that p learns all such e as well.

Let $m \in \mathbb{N}$ be arbitrary and let $e \in E_{f_m}$ be arbitrary. By Definition 7, for all but finitely many $n \in \mathbb{N}$, $t_e(n) < f_m(n)$. We will show that p learns to predict e.

Let M be a Turing machine which takes an input $(y_1, \ldots, y_n) \in B^*$ and operates as follows:

- Spend up to $f_m(n)$ steps computing $e(y_1, \ldots, y_n)$. If e halts (with output x) during that time, then output x. Otherwise, output 0.

Since, for all but finitely many $n \in \mathbb{N}$, $t_e(n) < f_m(n)$, it follows that M computes the same function as e except for finitely many exceptions.

It is well-known that for every primitive recursive function g, Peano arithmetic proves that g is total. Thus, by the *basic mathematical exposure* and *mathematical reasoning ability* assumptions from Assumption 1, it follows that X knows that M is total. Since M computes the same function as e except for finitely many exceptions, it follows that there is a Turing machine M' which computes the same function as e, and such that X knows that M' is total. By Lemma 11, p learns to predict e. □

5 An elementary method to non-trivialize Hibbard's intelligence measure

In Subsection 4.2, we argued that every AGI probably has intelligence ∞ according to the original Hibbard measure. In this section, we will present a family of variations of the original Hibbard measure which do not assign intelligence ∞ to every AGI. This family will contain one intelligence measure for each AGI X, which measure one might think of informally as a measure of "intelligence as judged by X".

The original Hibbard measure depends on a particular function-sequence. Hibbard chose one specific function-sequence, namely the enumeration from Liu [9]. Our method here is not to choose a function-sequence by ourselves, but instead to delegate that choice to an AGI.

Definition 17. *If X is an AGI, let $(C_1^X, C_2^X, \ldots)$ be the sequence of Turing machines which would result if X were commanded: "Until further notice, list all the Turing machines C that you can think of such that you know C computes a total function from $\mathbb{N}$ to $\mathbb{N}$." For each $i > 0$, let $g_i^X : \mathbb{N} \to \mathbb{N}$ be the function computed by C_i^X.*

For example, if X is an AGI, then in response to the command in Definition 17, X might think for a while, figure out a Turing machine C_1^X which computes

the constant function $g_1^X(n) = 0$. Then X might think some more and figure out a Turing machine C_2^X which computes the identity function $g_2^X(n) = n$. Then X might think some more and figure out a Turing machine C_3^X which computes the prime number function, so that $g_3^X(n)$ is the $(n+1)$th prime number. And so on.

Definition 18. *For any AGI X, the* simple Hibbard measure given by X *is defined to be the Hibbard measure $|\bullet|_{(g_1^X, g_2^X, \ldots)}$ given by $(g_1^X, g_2^X, \ldots)$ (see Definition 8). For any AGI Y, we will write $|Y|_X$ for $|Y|_{(g_1^X, g_2^X, \ldots)}$.*

5.1 Non-triviality of the simple Hibbard measures

We will now argue that at least for certain AGIs X, the simple Hibbard measure given by X does not assign intelligence ∞ to every AGI: but that, at least for certain AGIs X, there exist AGIs Y such that $|Y|_X < \infty$.

Theorem 19. *(Non-triviality of simple Hibbard measures) Suppose X and Y are AGIs. Let p be Y's predictor. If X knows that p defines a total function from B^* to B, then $|Y|_X < \infty$.*

Proof. Let e be the evader which attempts to evade the predictor by assuming that the predictor is p and always outputting the opposite of what the predictor will output based on that assumption. Clearly, p does not learn to predict e.

Since X knows that p defines a total function from B^* to B, it follows that there is a Turing machine C such that:

1. C computes the function $g : \mathbb{N} \to \mathbb{N}$ defined by $g(n) = t_e(n) + 1$, where t_e is as in Definition 5.

2. X knows that C computes a total function from $\mathbb{N}$ to $\mathbb{N}$.

By (2), $C = C_m^X$ for some $m > 0$, and thus $g_m^X = g$. So if $f_m : \mathbb{N} \to \mathbb{N}$ is defined (as in Definition 8) as $f_m(k) = \max_{0<i\leq m} \max_{j\leq k} g_i^X(j)$, then for every k, $f_m(k) \geq g_m^X(k) = g(k) = t_e(k) + 1 > t_e(k)$. This shows that $e \in E_{f_m}$ (from Definition 7). Since p does not learn to predict e, it is not the case that p learns to predict every evader in E_{f_m}. Therefore, $|Y|_X < m$. □

5.2 The problem with simple Hibbard measures

In Theorem 19 we proved more than was necessary, yielding the following stronger result.

Corollary 20. *Suppose X and Y are AGIs. Let p be Y's predictor. Suppose that X knows that p defines a total function from B^* to B. Then there is a Turing machine C such that:*

1. *C computes the function $g : \mathbb{N} \to \mathbb{N}$ defined by $g(n) = t_e(n) + 1$, where t_e is as in Definition 5.*

2. X knows that C computes a total function from $\mathbb{N}$ to $\mathbb{N}$.

Furthermore: $|Y|_X < m$ where m is such that $C = C_m^X$.

On the other hand, the following result also holds.

Lemma 21. *Suppose X and Y are AGIs. Suppose that for all $i = 1, \ldots, k$, g_i^X is identically zero: $g_i^X(n) = 0$. Then $|Y|_X \geq k$.*

Proof. If $f_k : \mathbb{N} \to \mathbb{N}$ is defined (as in Definition 8) by $f_k(n) = \max_{0<i\leq k} \max_{j\leq n} g_i^X(j)$, then, by assumption, $f_k(n) = \max_{0<i\leq k} \max_{j\leq n} 0 = 0$. Thus E_{f_k} is empty, and so, vacuously, Y's predictor learns to predict every evader in E_{f_k}. This shows $|Y|_X \geq m$. □

By the *mathematical reasoning ability* and *basic mathematical exposure* assumptions from Assumption 1, for any AGI X, there are infinitely many Turing machines C such that

1. C defines the zero function $g(n) = 0$ defined on all of $\mathbb{N}$, and
2. X knows C defines a total function from $\mathbb{N}$ to $\mathbb{N}$.

Thus, combining Corollary 20 and Lemma 21, we see that if X knows that Y's predictor defines a total function from B^* to B, then we can make $|Y|_{X'}$ have any value we like, by letting X' be an AGI identical to X in every way except that X' outputs Turing machines in a different order than X when commanded as in Definition 17. If we want $|Y|_{X'}$ to be exactly equal to, say, k, we need only contrive that $C_1^{X'}, \ldots, C_k^{X'}$ compute the zero function and $C_{k+1}^{X'}$ computes Y's predictor.

The above is a crucial flaw in the simple Hibbard measures, because it shows that $|Y|_X$ depends just as much on the arbitrary order in which X outputs Turing machines as it does on how intelligent Y is.

In Section 7, we will define a more sophisticated type of Hibbard measure which avoids the above problem. First, though, we need to borrow some machinery from mathematical logic.

6 The slow-growing hierarchy

The original Hibbard measure (Definition 9) depends on a specific list of quickly-growing computable functions from $\mathbb{N}$ to $\mathbb{N}$, imported from [9]. In Section 4, we argued that the original Hibbard measure is trivial for AGIs because the functions thus imported are too slow-growing. In Section 5 we generalized the original Hibbard measure by delegating the choice of a list of quickly-growing computable functions to an AGI rather than to a single external paper. The resulting family of simple Hibbard measures is non-trivial (Theorem 19), but in Subsection 5.2 we argued that these measures still have a crucial flaw: they depend too heavily on the order in which the AGI supplies its list of computable functions.

In order to solve the problem of dependence on computable functions, we will take a different strategy.

- In mathematical logic, there is a general recipe for obtaining quickly-growing computable functions, through the usage of so-called *slow-growing hierarchies*.
- The recipe takes as input a computable ordinal number α along with some so-called *fundamental sequences* for certain ordinals $\lambda < \alpha$ (called *limit ordinals*), and outputs a family of functions: one function $G_\beta : \mathbb{N} \to \mathbb{N}$ for every ordinal $\beta < \alpha$, and for large values of β, G_β grows very quickly.
- Thus, instead of commanding our AGI to directly produce a list of quickly-growing computable functions, we can instead command our AGI to produce computable ordinal numbers and fundamental sequences, and then obtain a list of quickly-growing computable functions indirectly, via the resulting slow-growing hierarchy.

This solves the problem of order-dependence: we can completely ignore the order in which the AGI outputs the computable ordinals, because computable ordinals are *themselves* ordered. In other words, if the AGI outputs a small computable ordinal and a large computable ordinal, we can consider the small computable ordinal to precede the large computable ordinal, regardless of in which order the AGI outputs them.

Definition 22. *(Classification of ordinal numbers) Let α be an ordinal number.*

1. *If $\alpha = \beta + 1$ for some ordinal number β, then α is called a* successor ordinal*.*
2. *If α is not a successor ordinal and $\alpha \neq 0$, then α is called a* limit ordinal*.*

Definition 23. *(Fundamental sequences) If λ is a limit ordinal, a* fundamental sequence *for λ is a sequence $(\lambda[0], \lambda[1], \lambda[2], \ldots)$ of ordinals, such that the following requirements hold:*

1. *Each $\lambda[i] < \lambda[i+1] < \lambda$.*
2. *The supremum of the $\lambda[i]$s is λ. Or equivalently: for each ordinal $\alpha < \lambda$, there is some $N \in \mathbb{N}$ such that for all $i > N$, $\lambda[i] > \alpha$.*

Definition 24. *(Slow-Growing Hierarchies) Suppose α is an ordinal and suppose $\mathcal{F}$ is a function which takes as input a limit ordinal $\lambda < \alpha$ and outputs a fundamental sequence $\mathcal{F}(\lambda) = (\lambda[0], \lambda[1], \lambda[2], \ldots)$ for λ. The* slow-growing hierarchy given by α and $\mathcal{F}$ *is the family $\{G_\beta\}_{\beta<\alpha}$ of functions consisting of a function $G_\beta : \mathbb{N} \to \mathbb{N}$ for every ordinal $\beta < \alpha$, defined by transfinite induction according to following cases:*

1. $G_0(x) = 0$.
2. $G_{\beta+1}(x) = G_\beta(x) + 1$.

3. $G_\lambda(x) = G_{\lambda[x]}(x)$ *(for every limit ordinal* $\lambda < \alpha$*), where* $(\lambda[0], \lambda[1], \lambda[2], \ldots)$ *is the fundamental sequence* $\mathcal{F}(\lambda)$ *of* λ*.*

Lemma 25. *Suppose* $\{G_\beta\}_{\beta<\alpha}$ *is the slow-growing hierarchy given by an ordinal* α *and function* $\mathcal{F}$*. If* α *is computable and* $\mathcal{F}$ *is computable, then each* G_β *is computable.*

Proof. By Church's thesis. □

Example 26. *Consider the ordinal* $\alpha = \omega^2$ *(where* ω *is the smallest infinite ordinal). The limit ordinals* $\lambda < \omega^2$ *are exactly the ordinals* $\{\omega\cdot(n+1) : n \in \mathbb{N}, \}$*. A natural way to assign fundamental sequences to these limit ordinals is to define, for all* $n, m \in \mathbb{N}$*,*

$$\mathcal{F}(\omega \cdot (n+1))(m) = (\omega \cdot (n+1))[m] = (\omega \cdot n) + m.$$

Let us compute some of the functions $\{G_\beta : \beta < \alpha\}$ *given by* α *and* $\mathcal{F}$*. Below,* x *and* n *denote arbitrary natural numbers.*

1. *By the base case,* $G_0(x) = 0$ *for all* $x \in \mathbb{N}$*.*
2. *By the successor case,* $G_1(x) = G_0(x) + 1 = 0 + 1 = 1$*.*
3. *By the successor case,* $G_2(x) = G_1(x) + 1 = 1 + 1 = 2$*.*
4. *Generalizing the above pattern,* $G_n(x) = n$*.*
5. *By the limit case,* $G_\omega(x) = G_{\omega\cdot(0+1)}(x) = G_{(\omega\cdot(0+1))[x]}(x) = G_{(\omega\cdot 0)+x}(x) = G_x(x) = x$*.*
6. *By the successor case,* $G_{\omega+1}(x) = G_\omega(x) + 1 = x + 1$*.*
7. *By the successor case,* $G_{\omega+2}(x) = G_{\omega+1}(x) + 1 = x + 2$*.*
8. *Generalizing the above pattern,* $G_{\omega+n}(x) = x + n$*.*
9. *By the limit case,* $G_{\omega\cdot 2}(x) = G_{\omega\cdot(1+1)}(x) = G_{(\omega\cdot(1+1))[x]}(x) = G_{(\omega\cdot 1)+x}(x) = x + x$*.*
10. *Similarly, the reader can check that* $G_{\omega\cdot 3}(x) = x + x + x$*.*
11. *Generalizing the above pattern,* $G_{\omega\cdot n}(x) = x \cdot n$*.*

Example 27. *The ordinal* ϵ_0 *(pronounced "epsilon-nought") is the smallest ordinal larger than the ordinals* $\omega, \omega^\omega, \omega^{\omega^\omega}, \ldots$*. It satisfies the equation* $\epsilon_0 = \omega^{\epsilon_0}$*, and can be intuitively thought of as*

$$\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}.$$

Examples of ordinals below ϵ_0 *include such ordinals as* ω^ω*,* $\omega^{\omega^\omega+\omega}$*,*

$$\omega^{\omega^{\omega^{\omega}+\omega^{3}+5}+\omega^{\omega^{2}}+\omega^{\omega}+1} + \omega^{\omega^{\omega^{2}+\omega+1}+\omega^{800}} + \omega^{\omega^{5}\cdot 12+\omega^{4}\cdot 9+1000} + 1,$$

and so on. There is a natural way to assign fundamental sequences to limit ordinals $< \epsilon_0$, yielding a corresponding slow-growing hierarchy where, for example,

$$G_{\omega^\omega}(x) = x^x,$$
$$G_{\omega^{\omega}+5}(x) = x^{x^x} + 5,$$
$$G_{\omega^{\omega^\omega}}(x) = x^{x^{x^x}},$$

and so on. Likewise, $(\omega, \omega^\omega, \omega^{\omega^\omega}, \ldots)$ is itself a fundamental sequence for ϵ_0. What could $G_{\epsilon_0}(x)$ be? Thinking of ϵ_0 as $\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}$, one might initially think $G_{\epsilon_0}(x) = x^{x^{\cdot^{\cdot^{\cdot}}}}$, but such an infinite tower of natural numbers makes no sense for $x > 1$. Instead, the answer defies familiar mathematical notation:

$$G_{\epsilon_0}(0) = 0$$
$$G_{\epsilon_0}(1) = 1^1$$
$$G_{\epsilon_0}(2) = 2^{2^2}$$
$$G_{\epsilon_0}(3) = 3^{3^{3^3}},$$

and so on.

Example 27 illustrates how slow-growing hierarchies reduce the open-ended problem

"Invent a quickly-growing computable function"

to a slightly less open-ended and more manageable problem:

"Invent fundamental sequences up to a large computable ordinal."

Slow-growing hierarchies provide an elegant means of obtaining quickly-growing computable functions from $\mathbb{N}$ to $\mathbb{N}$. The larger the computable ordinal, the larger the resulting computable function. However, inventing large computable ordinals is a non-trivial task—in [2] we argue that it might be a task that exhausts the full range of intelligence (i.e., that for any possible intelligence level, there is a corresponding computable ordinal whose invention requires at least that much intelligence). Even having invented a large computable ordinal, the task of assigning fundamental sequences to the limit ordinals below it is another difficult task in general[4]. Fortunately, this is a theoretical paper about AGI, so we can avoid all these decisions by delegating them to an AGI. In order to do so in a sensible way, we need a certain technical lemma which will allow us to command the AGI to give us a slow-growing hierarchy piece-by-piece.

Definition 28. *Suppose $\mathcal{H}_1$ is the slow-growing hierarchy given by ordinal α_1 and function $\mathcal{F}_1$, and suppose $\mathcal{H}_2$ is the slow-growing hierarchy given by ordinal α_2 and function $\mathcal{F}_2$. We say $\mathcal{H}_2$* extends *$\mathcal{H}_1$, written $\mathcal{H}_1 \prec \mathcal{H}_2$, if the following requirements hold:*

[4]The choice of fundamental sequences can dramatically change the growth rate of the resulting slow-growing hierarchy—see [12].

1. $\alpha_1 < \alpha_2$.

2. *For every limit ordinal* $\lambda < \alpha_1$, $\mathcal{F}_1(\lambda) = \mathcal{F}_2(\lambda)$.

In the following lemma and its proof, we will be dealing with unions of functions and with unions of ordinals, so it is important to recall the following:

- A *relation* is a set R of pairs (x, y).
- A *function* is a relation f which satisfies the requirement (sometimes called the "vertical line test") that whenever $(x, y_1) \in f$ and $(x, y_2) \in f$, then $y_1 = y_2$. Whenever $(x, y) \in f$, we write $f(x)$ to denote y.
- Every ordinal α is equal to the set of all ordinals $\beta < \alpha$.

Lemma 29. *Suppose for every* $i \in \mathbb{N}$, $\mathcal{H}_i$ *is the slow-growing hierarchy given by an ordinal* α_i *and function* $\mathcal{F}_i$. *If* $\mathcal{H}_0 \prec \mathcal{H}_1 \prec \mathcal{H}_2 \prec \cdots$, *then* $\mathcal{F} = \cup_i \mathcal{F}_i$ *is a function which assigns fundamental sequences to all limit ordinals* $\lambda < \alpha = \cup_i \alpha_i$ *(so that together,* α *and* $\mathcal{F}$ *give a slow-growing hierarchy).*

Proof. To show that $\mathcal{F}$ is a function, which must show that for any two pairs $(\lambda, y_1) \in \mathcal{F}$ and $(\lambda, y_2) \in \mathcal{F}$, $y_1 = y_2$ (i.e., that $\mathcal{F}$ "passes the vertical line test"). Assume $(\lambda, y_1) \in \mathcal{F}$ and $(\lambda, y_2) \in \mathcal{F}$. This means there are some $i_1, i_2 \in \mathbb{N}$ such that $\mathcal{F}_{i_1}(\lambda) = y_1$ and $\mathcal{F}_{i_2}(\lambda) = y_2$. If $i_1 = i_2$ then $y_1 = y_2$ because $\mathcal{F}_{i_1} = \mathcal{F}_{i_2}$ is a function, but assume $i_1 \neq i_2$. Without loss of generality we may assume $i_1 < i_2$ (the other case is similar). Since $\mathcal{H}_{i_1} \prec \cdots \prec \mathcal{H}_{i_2}$, it follows by a simple inductive argument that $\mathcal{H}_{i_1} \prec \mathcal{H}_{i_2}$. Thus by condition 2 of Definition 28, $y_1 = y_2$, as desired.

It remains to let $\lambda < \alpha$ be a limit ordinal and show that conditions 1 and 2 of Definition 23 hold for $\mathcal{F}(\lambda)$. Since $\lambda < \alpha$, that means $\lambda < \alpha_i$ for some $i \in \mathbb{N}$. Let $(\lambda[0], \lambda[1], \ldots) = \mathcal{F}(\lambda)$. Then $(\lambda[0], \lambda[1], \ldots) = \mathcal{F}_i(\lambda)$, so conditions 1 and 2 of Definition 23 hold because $\mathcal{F}_i$ is a function which assigns fundamental sequences to all limit ordinals $< \alpha_i$. □

7 The Transfinite Hibbard measures

We would like to delegate the problem of inventing large computable ordinals and fundamental sequences to an AGI. In order to do so, we need a systematic way for the AGI to encode those things. Since we do not care about the specific details of the encoding, we can get away with the following high-level definition. A more concrete definition would be possible by using, say, the ordinal notation system known as Kleene's $\mathcal{O}$ [7], but the details would be long and tedious.

Definition 30. *By a* slow-growing hierarchy notation*, we mean a definition (in the language of ZFC) of a pair* $(\alpha, \mathcal{F})$ *such that* α *is a computable ordinal and* $\mathcal{F}$ *is a computable function that assigns fundamental sequences to limit ordinals* $\lambda < \alpha$.

Definition 31. *Let X be an AGI. Let $d_0, d_1, \ldots$ be the slow-growing hierarchy notations which would result if X were commanded as follows:*

> *"Until further notice, output slow-growing hierarchy notations of pairs $(\alpha_i, \mathcal{F}_i)$, such that the corresponding slow-growing hierarchies $\mathcal{H}_i$ satisfy the hypotheses of Lemma 29. Do this in such a way that the α_is include computable ordinals as large as possible."*

For each $i \in \mathbb{N}$, let α_i and $\mathcal{F}_i$ be the computable ordinal and computable function defined by d_i. The slow-growing hierarchy of X *is defined to be the slow-growing hierarchy given by $\alpha = \cup_i \alpha_i$ and $\mathcal{F} = \cup_i \mathcal{F}_i$ (see Lemma 29).*

Definition 32. *Suppose that X is an AGI, and that α and $\mathcal{F}$ are as in Definition 31, and that $\{G_\beta\}_{\beta<\alpha_\infty}$ is the slow-growing hierarchy given by α and $\mathcal{F}$ (Definition 24). The* transfinite Hibbard measure given by X *is the function which assigns to each AGI Y a computable ordinal (or ∞) $\|Y\|_X$ defined as follows. We define $\|Y\|_X$ to be the smallest ordinal $\beta < \alpha$ such that there is some evader $e \in E_{G_\beta}$ such that Y's predictor does not learn to predict e (where E_{G_β} is as in Definition 7), or we define $\|Y\|_X = \infty$ if there is no such $\beta < \alpha$.*

Definition 32 avoids the order-dependency problem which we saw in Subsection 5.2, because the functions G_α which are used to define evader-sets come with their own intrinsic ordering (G_α comes before G_β if and only if $\alpha < \beta$) rather than depending on the order in which the AGI thinks of things.

If X is an AGI, consider the set S of all AGIs Y such that $\|Y\|_X < \infty$. The transfinite Hibbard measure given by X is a computable-ordinal-valued intelligence measure defined on S. This should be contrasted with intelligence measures that are real-number-valued. In [3] we point out that because the real numbers are constrained by a certain generalized Archimedean property, real numbers are inappropriate for measuring things which are non-Archimedean in a certain technical sense. Arguably, if X itself is sufficiently intelligent, then the intelligences of the AGIs in S are probably non-Archimedean, and so the usage of computable ordinals to measure their intelligence is more appropriate than using real numbers.

If AGI Y plays the adversarial sequence prediction game using Y's brute-force predictor, Y enumerates all the Turing machines $M_1, M_2, \ldots$ such that Y knows that M_i computes a total function from B^* to B, and Y systematically tries to predict the evader e by first assuming that e computes the same function as M_1, and if that fails, then assuming that e computes the same function as M_2, and so on. In particular, suppose Y knows that a certain computable ordinal α_i and function $\mathcal{F}_i$ give a slow-growing hierarchy $\{G_\beta\}_{\beta<\alpha_i}$. For any $\beta < \alpha_i$, since Y is capable of mathematical reasoning, for every Turing machine M which computes a total function from B^* to B and which has runtime bounded by G_β, there should be a Turing machine M' such that M computes the same function as M' and such that Y knows that M' computes a total function from B^* to B. Thus, $M_1, M_2, \ldots$ should include Turing machines that compute all the functions from B^* to B that can be so computed with runtime bounded by

any such G_β. Thus, by Assumption 15, if $\|Y\|_X < \infty$, then it seems that $\|Y\|_X$ should in some sense measure the size of the supremum of the ordinals α such that Y knows a fundamental sequence up to α. This suggests a close relation between the transfinite Hibbard measure and our Intuitive Ordinal Intelligence measure (an AGI Y's Intuitive Ordinal Intelligence is defined as the supremum of the ordinals α such that α has any code c such that Y knows that c is the code of a computable ordinal) [1] [2].

8 Summary and Conclusion

To summarize:

- Hibbard proposed [6] an intelligence measure for predictors (and, implicitly, for AGIs capable of acting as predictors) in games of adversarial sequence prediction.
- Hibbard's measure depends on one specific sequence of computable functions from $\mathbb{N}$ to $\mathbb{N}$.
- In Section 2, we argued that all AGIs probably have intelligence ∞ according to Hibbard's definition. Essentially, this is because the specific functions that Hibbard's definition is based on should all be trivially well-known by any genuine AGI.
- In Section 5, we introduced what we call *simple Hibbard measures*. For each AGI X, there is a corresponding simple Hibbard measure which assigns an intelligence measurement $|Y|_X$ to every AGI Y. The simple Hibbard measure corresponding to X might be thought of as a measure of "intelligence as judged by X". The key insight needed to define the simple Hibbard measure corresponding to X is as follows: rather than depend on one fixed sequence of functions (as in Hibbard's original definition), we can delegate the task of choosing a function-sequence to X.
- We showed that simple Hibbard measures are less trivial than Hibbard's original measure. However, they still suffer a crucial flaw: the Hibbard measure corresponding to an AGI X depends unnaturally on the *order* in which X can think of computable functions.
- To remedy the above-mentioned flaw in the simple Hibbard measures, we reviewed *slow-growing hierarchies* from mathematical logic and used them to define (in Section 7) what we call *transfinite Hibbard measures*, which lead to computable-ordinal-number-valued intelligence measures rather than real-number-valued intelligence measures.

This paper exemplifies a useful technique for theoretical AGI research. By viewing an AGI as an employee who is capable of engaging in casual human language communication with its employer, and which can be commanded (using said casual human language) to perform mathematical tasks, the AGI becomes

a useful black box to which many arbitrary decisions can be delegated. For example, instead of arbitrarily choosing one particular function-sequence (as Hibbard did in order to define his original measure), we can short-circuit the task of function-sequence-selection by delegating it to an AGI.

References

[1] Samuel Allen Alexander. Measuring the intelligence of an idealized mechanical knowing agent. In *CIFMA*, 2019.

[2] Samuel Allen Alexander. AGI and the Knight-Darwin law: why idealized AGI reproduction requires collaboration. In *International Conference on Artificial General Intelligence*, 2020.

[3] Samuel Allen Alexander. The Archimedean trap: Why traditional reinforcement learning will probably not yield AGI. *arXiv preprint arXiv:2002.10221*, 2020.

[4] Aristotle. Topics. In Jonathan Barnes et al., editors, *The Complete Works of Aristotle*. Princeton University Press, 1984.

[5] José Hernández-Orallo and David L Dowe. Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence*, 174(18):1508–1539, 2010.

[6] Bill Hibbard. Measuring agent intelligence via hierarchies of environments. In *ICAGI*, pages 303–308, 2011.

[7] Stephen Cole Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938.

[8] Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and machines*, 17(4):391–444, 2007.

[9] Shih Chao Liu. An enumeration of the primitive recursive functions without repetition. *Tohoku Mathematical Journal*, 12(3):400–402, 1960.

[10] Pei Wang. Three fundamental misconceptions of artificial intelligence. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(3):249–268, 2007.

[11] Pei Wang, Peter Isaev, Patrick Hammer, et al. Nars tutorial and workshop. In *International Conference on Artificial General Intelligence*, 2020.

[12] Andreas Weiermann. Sometimes slow growing is fast growing. *Annals of Pure and Applied Logic*, 90(1-3):91–99, 1997.

[13] Andreas Weiermann. Slow versus fast growing. *Synthese*, 133:13–29, 2002.