Extended subdomains: a solution to a problem of Hernández-Orallo and Dowe

Samuel Allen Alexander[1]

The U.S. Securities and Exchange Commission samuelallenalexander@gmail.com
https://philpeople.org/profiles/samuel-alexander/publications

Abstract. This is a paper about the general theory of measuring or estimating social intelligence via benchmarks. Hernández-Orallo and Dowe described a problem with certain proposed intelligence measures. The problem suggests that those intelligence measures might not accurately capture social intelligence. We argue that Hernández-Orallo and Dowe's problem is even more general than how they stated it, applying to many subdomains of AGI, not just the one subdomain in which they stated it. We then propose a solution. In our solution, instead of using test-cases within the given AGI subdomain to estimate an AI's intelligence, one would use test-cases in an extended subdomain where test-cases have the ability to simulate the AI being tested. Surprisingly, AIs only designed for the original subdomain can be tested with test-cases in the extended subdomain anyway. By extending the subdomain in this way, we might avoid Hernández-Orallo and Dowe's problem.

Keywords: Social intelligence · Intelligence measurement · Universal intelligence.

1 Introduction

The problem of designing AGI goes hand-in-hand with the problem of measuring the intelligence of artificial agents. After all, without the ability to measure intelligence, it would be hard to even know whether progress is being made toward AGI. For the diverse and wide-ranging types of intelligent agents considered by AGI researchers as a whole, the intelligence-measurement problem is quite difficult (it is not clear to what extent objective intelligence measurement is even possible in such a general context). Concrete progress can be made by restricting attention to narrow, well-defined subdomains of AGI. Within a narrow subdomain of AGI, one can measure (or at least estimate) intelligence by using benchmarks: how well does the agent perform on such-and-such test-cases? For example, how well does a given AI perform at Atari games? Such subdomains of AGI can be considered as (in Goertzel's words) "idealized case[s] of AGI, similar to assumptions like the frictionless plane in physics" [4].

Hernández-Orallo and Dowe pointed out [6] a problem in certain theoretical intelligence-measurement benchmarks. We will argue that the problem they point out is actually much more general: they posed it in the context of one particular subdomain of AGI but it is not limited to that subdomain. A benchmark generally consists of a battery of simple test-cases, or a simple procedural method for randomly generating test-cases. But this seems to prevent the test-cases from having genuine social aspects, for the following reason. To include genuine social aspects in a test-case would (apparently) require that genuine intelligence be somehow built into that test-case. For example, in an Atari game, enemy (or ally) Non-Player Characters (NPCs) are simplistic automatons. Simplistic automatons are not genuinely social. To add genuine social aspects to an Atari game, one would need to replace those automatons with genuinely intelligent agents. But the complexity of such agents would far exceed the complexity of the Atari game! Or, if test-cases are generated procedurally, perhaps the procedure could randomly generate test-cases with genuine social aspects, but the odds of this would be extremely small. One could replace an Atari NPC's script with a randomly-generated script, but the odds are negligible that the NPC would thus become genuinely intelligent. So then, how can our benchmarks capture social intelligence?

We will propose a general solution where AIs in one subdomain are benchmarked against test-cases in an extended subdomain. In the extended subdomain, test-cases have the ability to secretly simulate the agent being measured. For example, to measure AIs in the subdomain of Atari games, we would run those AIs against *extended Atari games*. An extended Atari game is just like an Atari game, except that the game's mechanics are allowed to use an oracle to query what the AI playing the game would do in arbitrary situations. We will argue that in the extended subdomain, genuine social aspects can be built into simple test-cases. Furthermore, this solution is surprisingly quite practical. In the Atari subdomain, for example, if we have an AI's source-code, we can use that source-code to realize the oracle needed to run the AI in an extended Atari game. By contrast, it would be virtually impossible for a human to play extended Atari games in general, because it would be virtually impossible to realize an oracle that could predict the human's actions in arbitrary situations[1].

2 Background: The Hernández-Orallo and Dowe Problem

> 'How can we create environments so that they have intelligent agents inside? It is enlightening (but perhaps of little practical use) to think that some extremely complex infinite environments we consider as possible in the test could contain "life". In some of them, we could even find "intelligent beings" ... When we say that it is perhaps of little practical use, it is because the complexity of these environments is extremely high and the probability of one of them appearing by chance is almost zero.

[1] Even a perfect genetic clone of the human player would not be enough, since a human player's actions are not determined by genetics alone but depend on a whole lifetime of previous learning and interaction with the world.

Therefore, we cannot bind the evaluation of social intelligence to this remote chance. However, this a priori remote probability is in fact a much higher a posteriori probability if we think in terms of evolution. ... Consequently, we require inserting these other agents into the environments'
—Hernández-Orallo and Dowe [6]

Hernández-Orallo and Dowe did not state their problem in its full generality. They stated it [6] in the universal intelligence context of [10], essentially a very formal, theoretical version of the reinforcement learning (RL) context. We will avoid spelling everything out in full detail as the details are verbose and unimportant. Roughly speaking, in the universal intelligence context, *agents* interact with *environments.* They take turns. On the agent's turn, the agent takes an *action.* On the environment's turn, the environment gives the agent an *observation* and a *reward.* Certain technical constraints are placed on the rewards which the environment can output, in order to ensure certain convergence properties. The agent is considered to perform better or worse in a given environment if the rewards it receives are bigger or smaller, respectively.

Legg and Hutter proposed [10] defining the numerical intelligence level of such an agent to be the average total reward the agent receives across the space of all computable environments, weighting environments with some distribution. A uniform distribution would be no good because No-Free Lunch theorems imply all agents would end up with the same exact intelligence measure, see [8]. Legg and Hutter instead proposed giving each environment μ a weight of $2^{-K(\mu)}$ where $K(\mu)$ is the Kolmogorov complexity of μ (the length of the shortest computer program for μ). Note that $K(\mu)$ depends implicitly on the choice of a background Universal Turing Machine (UTM). Intuitively one can think of a UTM as a programming language, so the choice amounts to choosing: which programming language should environments be programmed in? This choice is non-trivial, see [12][2].

Other methods have also been proposed for measuring intelligence in the universal intelligence context. The Legg-Hutter intelligence definition is impractical because, mathematically, the average performance of the agent across the whole space of computable environments, is an infinite sum, each term of which involves the Kolmogorov complexity function (itself already non-computable). More practical methods involve running the agent for bounded numbers of turns against randomly-generated environments. Universal intelligence measures of this type are proposed by Legg and Veness [11] and by Hernández-Orallo and Dowe themselves [6]. Hernández-Orallo and Dowe further refine the idea, proposing to dynamically adjust the complexity of the randomly-generated environments based on the agent's performance, an idea motivated by human psychometrics.

All these methods of measuring universal intelligence are highly susceptible to Hernández-Orallo and Dowe's problem. Environments containing genuinely-intelligent built-in NPCs must be highly complicated. So in the Legg-Hutter infinite sum, any such environment would contribute very little, because its weight

[2] Some progress on UTM-choice was presented at last year's AGI conference [3].

$2^{-K(\mu)}$ would be extremely small. In intelligence measures based on running the agent in randomly-generated environments, the odds are quite small that a randomly-generated environment would contain a genuinely intelligent NPC. So all these intelligence measures would seem to poorly capture social intelligence.

Hernández-Orallo et al [7] proposed using multi-agent environments to solve the problem. In their proposal, in order to quantitatively estimate the intelligence of an agent, one would randomly generate multi-agent environments, and also randomly select agents for each multi-agent role (except for the role to be filled by the agent being measured). Despite this solution's inherent beauty, it is not very practical. Either the randomly-generated agents are generated completely at random (e.g., they have random source-codes), in which case the odds of such an agent being genuinely intelligent are extremely small; or, they are generated in some way such that with non-negligible probability they are genuinely intelligent. But the latter seems almost as difficult as creating AGI in the first place, so an intelligence measure dependent on it might not be very helpful as a step toward AGI. We will propose a different solution to Hernández-Orallo and Dowe's problem, which does not involve randomly generating agents.

2.1 The Generalized Hernández-Orallo and Dowe Problem

Nothing about the Hernández-Orallo and Dowe Problem inherently depends on the particular background of universal intelligence in which they stated it. The problem applies any time we would use simple test-cases (or a simple procedure for generating test-cases) to benchmark AIs in any subdomain of AGI. We would state the general problem as follows:

Problem 1 *(The Generalized Hernández-Orallo and Dowe Problem) Assume we are working in some subdomain of AGI where we want to benchmark AIs against test-cases. Any test-case with genuine intelligence built into it must necessarily be highly complex. Thus, no such test-case can occur in any fixed library of simple test-cases, and no such test-case can be generated with non-negligible probability by any simple procedure for generating test-cases. Thus, no such library or procedure can be used to reliably benchmark social intelligence (since social intelligence requires interaction with other genuine intelligences).*

Note that when we say "no such test-case can be generated with non-negligible probability..." we assume a certain sparseness condition. For any n, if S_1 is the set of all length-n computer programs, and S_2 is the set of all length-n computer programs of AGIs, then presumably $|S_2|/|S_1| \approx 0$. Given an algorithm for an AGI, one could contrive a programming language falsifying this (e.g., contrive the language so that said algorithm can be written in just 1 character). We believe this sparseness assumption is plausible for natural programming languages whose semantics do not depend on any already-known AGI.

Example 2 *(Image classification) Consider a subdomain of AGI where image classifiers can be trained on labeled images and asked to predict labels of unlabeled* *images. Fix some genuinely intelligent classifier A_0 and some finite set T of images with labels from $\{0,1\}$. Assume A_0 has been trained on T. Suppose A is a classifier whose intelligence we are trying to measure. As one test-case, we could systematically investigate how well A learns to classify images as either "images A_0 classifies as 0" or "images A_0 classifies as 1".*

The test-case in Example 2 is presumably complicated, because it depends on the genuinely intelligent classifier A_0. Thus, the test-case would never be included in any simple test-case library, and there is low probability it would be generated by any simple test-case-generating procedure. Thus, any estimate of a classifier's intelligence based on a simple test-case library, or on test-cases generated by a simple procedure, would fail to reliably capture the classifier's performance on the test-case in question. One could argue that the test-case in question is a social intelligence test-case, because it tests how well the classifier learns to anticipate its colleague A_0.

3 Extending Subdomains to Solve the Hernández-Orallo and Dowe Problem

We propose to solve Problem 1 by extending the subdomain in question so as to admit simple new test-cases capable of incorporating social aspects, in such a way that a given AI (only designed for the original subdomain) can still attempt test-cases in the extended subdomain. The intuitive idea is that, in the extended domain, when an AI is being tested on a test-case, the test-case is allowed to query an oracle which tells the test-case what the AI would output in response to arbitrary inputs. This allows for self-play to be incorporated into the test-cases (below, we address the anticipated objection that self-play is not genuinely social). Certainly we are not claiming that self-play is a new innovation. It has been widely used to train agents for specific individual environments, from Backgammon all the way to StarCraft II, and in a sense it is also used in adversarial techniques such as GAN (see [5]). What is new in our proposal is that we suggest self-play can be applied to general social intelligence measurement, where, instead of having a specific environment in mind, we are interested in an agent's general performance over the whole space of all environments.

Definition 3 *A* subdomain of AGI *is a tuple $\mathscr{D} = (\mathscr{A}, U_{\mathscr{A}}, \mathscr{T}, U_{\mathscr{T}}, L)$ where:*

1. *$\mathscr{A}$ is a set of computer programs (called* AIs*) in programming language $U_{\mathscr{A}}$;*
2. *$\mathscr{T}$ is a set of computer programs (called* test-cases*) in programming language $U_{\mathscr{T}}$ which extends $U_{\mathscr{A}}$ (possibly including some oracles);*
3. *For all $U_{\mathscr{A}}$-programs a_1, a_2, if a_1 and a_2 both compute the same function, then $a_1 \in \mathscr{A}$ iff $a_2 \in \mathscr{A}$.*
4. *For all $U_{\mathscr{T}}$-programs t_1, t_2, if t_1 and t_2 both compute the same function, then $t_1 \in \mathscr{T}$ iff $t_2 \in \mathscr{T}$.*
5. *L is a computable function which takes $a \in \mathscr{A}$, $t \in \mathscr{T}$, $n \in \mathbb{N}$, and outputs a rational number $L(a,t,n) \in \mathbb{Q}$ which we call a* measure of a's performance on t at step n.

We say $\mathscr{D}$ is code-independent *if the following requirement holds: for all $a_1, a_2 \in \mathscr{A}$, for all $t_1, t_2 \in \mathscr{T}$, if a_1 and a_2 compute the same function, and t_1 and t_2 compute the same function, then for all $n \in \mathbb{N}$, $L(a_1, t_1, n) = L(a_2, t_2, n)$.*

Example 4 *Take $\mathscr{A}$ to be the set of programs defining RL agents (in some formalization of RL) and $\mathscr{T}$ to be the set of programs defining RL environments, both in some common language $U_{\mathscr{A}} = U_{\mathscr{T}}$. Let $L(a, t, n)$ be the nth reward a gets in an interaction with t. The resulting code-independent subdomain $\mathscr{D}$ could be called the* RL subdomain of AGI.

Definition 5 *Suppose $\mathscr{D} = (\mathscr{A}, U_{\mathscr{A}}, \mathscr{T}, U_{\mathscr{T}}, L)$ is a subdomain of AGI. The* extension *of $\mathscr{D}$ is the subdomain $\mathscr{D}' = (\mathscr{A}, U_{\mathscr{A}}, \mathscr{T}', U_{\mathscr{T}'}, L')$ where:*

1. *$U_{\mathscr{T}'}$ is the extension of $U_{\mathscr{T}}$ by a new oracle $\mathbf{a}$.*
2. *$\mathscr{T}'$ is the set of all $U_{\mathscr{T}'}$ programs t with the following property: for each $a \in \mathscr{A}$, if t_a is the $U_{\mathscr{T}}$ program obtained from t by replacing all instances of $\mathbf{a}$ by a, then $t_a \in \mathscr{T}$.*
3. *L' is the computable function which, on input $a \in \mathscr{A}$, $t \in \mathscr{T}'$, and $n \in \mathbb{N}$, outputs $L'(a, t, n) = L(a, t_a, n)$, where t_a is as above.*

If $\mathscr{D}$ is the RL subdomain of AGI (Example 4), then $\mathscr{D}'$ is a variation of RL in which environments can simulate agents in order to base their rewards and observations not only on what actions the agent has actually taken, but also on what actions the agent would hypothetically take in counterfactual scenarios[3].

Lemma 6 *If $\mathscr{D}$ is a subdomain of AGI, then $\mathscr{D}'$ really is a subdomain of AGI.*

Proof. The only nontrivial part of the claim is that L' is a computable function which, given $a \in \mathscr{A}$, $t \in \mathscr{T}'$, $n \in \mathbb{N}$, outputs $L'(a, t, n) \in \mathbb{Q}$. Clearly the operation of replacing instances of oracle $\mathbf{a}$ by a, is computable. So the computability of L' follows from the computability of L. By definition, $t \in \mathscr{T}'$ means $t_a \in \mathscr{T}$, so $L'(a, t, n) = L(a, t_a, n)$ exists and is in $\mathbb{Q}$ since L satisfies Definition 3. □

Even though Lemma 6 is trivial, it has profound implications. It says that even though an AI is designed for the original, un-extended subdomain of AGI, that AI can nevertheless be tested using test-cases in the extended subdomain.

While clearly not a perfect solution, the following theorem at least partly solves Problem 1.

Theorem 7 *(Deparametrization Theorem) Let $\mathscr{D} = (\mathscr{A}, U_{\mathscr{A}}, \mathscr{T}, U_{\mathscr{T}}, L)$ be a code-independent subdomain of AGI. Suppose F is a $U_{\mathscr{T}}$ program which takes as input an AI $a \in \mathscr{A}$ and outputs a test-case $F(a) \in \mathscr{T}$. In the extended subdomain $\mathscr{D}' = (\mathscr{A}, U_{\mathscr{A}}, \mathscr{T}', U_{\mathscr{T}'}, L')$, there is a test-case F^*, of approximately the same complexity as F, such that for all $a \in \mathscr{A}$ and $n \in \mathbb{N}$, $L'(a, F^*, n) = L(a, F(a), n)$.*

[3] Alexander et al explore this RL variation in [1], suggesting a variation of the Legg-Hutter intelligence measure that might measure an agent's self-reflection intelligence via its performance in extended RL environments (if these environments could be further pared down to just those of a social nature, the same idea could lead to a formal measure of RL agent social intelligence, but we do not currently know how to so pare them down).

Proof. Let F^* be the $U_{\mathscr{T}'}$ program:

1. Take input $\vec{X}$.
2. Output the result of running $F(\mathbf{a})$ on $\vec{X}$.

Clearly F^* has approximately the same complexity as F. For each $a \in \mathscr{A}$, F^*_a is the $\mathscr{T}$-program:

1. Take input $\vec{X}$.
2. Output the result of running $F(a)$ on $\vec{X}$.

Clearly F^*_a and $F(a)$ compute the same function. Thus by condition 4 of Definition 3, $F^*_a \in \mathscr{T}$. By arbitrariness of a, this shows $F^* \in \mathscr{T}'$. For any $a \in \mathscr{A}$ and $n \in \mathbb{N}$, $L'(a, F^*, n) = L(a, F^*_a, n)$ by Definition 5, which equals $L(a, F(a), n)$ since F^*_a and $F(a)$ compute the same function and $\mathscr{D}$ is code-independent. $\square$

Theorem 7 says that any AI-parametrized procedure for generating test-cases in the original subdomain can be replaced by a *single* test-case, in the extended subdomain, roughly as complex as the original procedure. In the single test-case, the AI-parameter is replaced by a simulated copy of the very AI we are trying to test. For example, suppose we want to test an Atari-playing AI's social intelligence. We could take $F(a)$ to be an Atari game in which the player plays "Super Breakout" with a as partner. Then F^* is a *single* extended Atari game in which the player plays "Super Breakout" with a *clone of herself* as her partner. Thus, the infinite test-case family, "Play Super Breakout with partner a" (each one of whose complexity is approximately the complexity of Super Breakout plus the complexity of a), is replaced by the single test-case, "Play Super Breakout with a clone of yourself as partner", roughly as complex as Super Breakout.

We would argue that test-cases produced by Theorem 7 are appropriate for benchmarking intelligence in a super-general context[4]. If we design an NPC opponent using a huge neural network, the resulting test-case has an inherent bias toward neural networks. That would be inappropriate for measuring alien intelligences based on some other technology. It would be rather arbitrary to judge a Martian life form by how well it can raise a *human* baby, or to judge a human by how well he can raise a *Martian* baby. But it would be quite appropriate to judge each by how well it can raise *a baby version of itself.*

The reader might object that there is nothing social about interacting with one's own clone. But in general, AIs act based not only on immediate stimulus, but on the whole history of prior stimuli. In short: AIs train. This is abstracted away in Definition 3. One should not think of the AIs in Definition 3 as taking immediate observations as sole inputs, but rather as taking entire histories as inputs. In Theorem 7, $F(a)$ might output a test-case where one plays chess against an instance of a that has been trained on, say, 50 years of random stimuli (generated dynamically, to keep F simple). Then F^* is a test-case where one

[4] Provided the test-cases $F(a)$ are nontrivial; Tic-Tac-Toe would be a poor social intelligence benchmark regardless of the opponent's intelligence.

plays chess against a clone of oneself trained with 50 years of random stimuli[5]. This could be quite different than playing against oneself directly. When Silver et al declare that

> "The agent consists solely of the decision-making entity; anything outside of that entity (including its body, if it has one) is considered part of the environment," [13]

that *body* would certainly include the brain and the hippocampus. So if I am being driven by an agent in Silver et al's sense, then a clone of that agent needn't share my memories. And to the extent that my personality depends on my memories (including what I was taught in school, etc.), said clone needn't share my personality. Indeed, if personality is a function of training, the following Paper-Rock-Scissors example illustrates how one could apply different training to a self-play opponent, encouraging the opponent to differ in personality from the agent.

Example 8 *(Paper-Rock-Scissors Python Example, see Listing 1.1) Consider a concrete formalization of RL in which environments are instances of Python environment-classes and agents are instances of agent-classes. An environment-class is required to implement a "start" method (outputting an initial observation) and a "step" method (which takes the agent's latest action and outputs an observation and a reward). An agent-class is required to implement an "act" method (which takes an observation and outputs an action) and a "train" method (which takes a prior observation, an action, a reward, and a next observation, the intent being that the agent should update its neural net, Q-table, etc., based on the fact that it took the given action in response to the given prior observation and this resulted in the given reward and the given next observation). This is a subdomain of AGI. The extended subdomain is identical except that an extended environment-class's methods have access to an oracle* AgentClass *(the* $\mathbf{a}$ *in Definition 5) for the agent-class used to instantiate the agent. Its methods can thus instantiate independent agent-clones for use in its reward-observation calculations. Listing 1.1 defines an extended environment-class where the agent plays Paper-Rock-Scissors against a clone of itself, but every move, the clone gets trained* twice *instead of once.*

Example 8 gives a single test-case in an extended subdomain. It corresponds to an infinite family of unextended test-cases, indexed by AgentClass. It tests the player's performance in the task[6]: "Play Paper-Rock-Scissors against a clone of yourself that trains twice as much as you." The extended environment has low complexity (≈ 10 or 20 lines of code), far simpler than a non-extended version

[5] If the player is human-like, 50 years of such training might even make the clone so different that the player doesn't realize the opponent *is* a clone.

[6] The example is not trivialized by the random strategy. A good RL agent should balance *exploitation* of known good strategies (like random play) against *exploration*. Otherwise the agent would be suboptimal against certain flawed opponents. Indeed, this line of thought leads to Hibbard's hierarchical intelligence measures [9] [2].

Listing 1.1. An extended environment in which the agent plays Paper-Rock-Scissors against a clone of itself, but the clone trains twice as much.

```
class PaperRockScissors_DoubleTrainingEnemy:
  def start(self):
    # Instantiate a clone of the agent. This clone will play
    # the role of the agent's enemy.
    self.sim = AgentClass()
    # Start interaction with both player & enemy seeing paper
    self.prev_player_action = PAPER
    return {'obs': PAPER}

  def step(self, player_action):
    # Figure out which action the agent's enemy takes
    enemy_action = self.sim.act(obs=self.prev_player_action)

    player_reward = compute_reward(player_action, enemy_action)

    # Train the enemy based on how the enemy sees things
    # (the enemy gets the opposite reward as the player, etc.)
    self.sim.train(prev_obs=self.prev_player_action,
        act=enemy_action, reward=-player_reward,
        next_obs=player_action)
    # Train again, so the enemy trains twice as much
    self.sim.train(prev_obs=self.prev_player_action,
        act=enemy_action, reward=-player_reward,
        next_obs=player_action)

    self.prev_player_action = player_action
    return {'reward': player_reward, 'obs': enemy_action}
```

with a fixed genuinely intelligent enemy built-in. Because the opponent is trained differently than the player, we would expect the opponent to develop a different personality than the player (except in some degenerate cases)—this gives the test-case a social aspect. The reader can easily imagine more ambitious examples where entire communities (or even civilizations) of entities interact with themselves and the player, each entity instantiated as AgentClass(), but different entities trained differently and therefore having distinct personalities. With some creativity, such ambitious extended environments could be programmed with relatively low complexity: the most complicated part (how the entities behave) is delegated away.

4 Conclusion

Hernández-Orallo and Dowe described [6] a problem which may prevent certain intelligence measures from measuring social intelligence. They stated the problem in the universal intelligence context [10]. We pointed out that the problem is more general. It arises any time we try to use simple test-cases (or a simple procedure for generating test-cases) to estimate intelligence in any AGI subdomain. The problem is that building genuine intelligence into a test-case (apparently necessary for the test-case to measure social intelligence) would make that test-case complicated, not simple. We propose a high-level solution. Instead of designing test-cases in the subdomain in question, design test-cases in an extended subdomain where test-cases can simulate the AI being tested. Such extended test-cases can incorporate social interaction by delegating competitors' or collaborators' intelligence to a clone (or clones) of the AI being tested. For example, instead of testing, "How well can the AI negotiate with such-and-such human?" (a question involving a complex arbitrary parameter), instead, test: "How well can the AI negotiate with its clone?" (a simple non-parametrized question).

Acknowledgments

We acknowledge José Hernández-Orallo and the reviewers for valuable feedback.

References

1. Alexander, S.A., Castaneda, M., Compher, K., Martinez, O.: Extending environments to measure self-reflection in reinforcement learning. Preprint (2022)
2. Alexander, S.A., Hibbard, B.: Measuring intelligence and growth rate: Variations on Hibbard's intelligence measure. JAGI **12**(1), 1–25 (2021)
3. Alexander, S.A., Hutter, M.: Reward-punishment symmetric universal intelligence. In: CAGI (2021)
4. Goertzel, B.: Artificial general intelligence: concept, state of the art, and future prospects. JAGI **5**, 1–48 (2014)
5. Hernández-Orallo, J.: Twenty years beyond the Turing test: Moving beyond the human judges too. Minds and machines **30**(4), 533–562 (2020)
6. Hernández-Orallo, J., Dowe, D.L.: Measuring universal intelligence: Towards an anytime intelligence test. AI **174**(18), 1508–1539 (2010)
7. Hernández-Orallo, J., Dowe, D.L., Espana-Cubillo, S., Hernández-Lloreda, M.V., Insa-Cabrera, J.: On more realistic environment distributions for defining, evaluating and developing intelligence. In: CAGI (2011)
8. Hibbard, B.: Bias and no free lunch in formal measures of intelligence. JAGI **1**(1), 54 (2009)
9. Hibbard, B.: Measuring agent intelligence via hierarchies of environments. In: CAGI (2011)
10. Legg, S., Hutter, M.: Universal intelligence: A definition of machine intelligence. Minds and machines **17**(4), 391–444 (2007)
11. Legg, S., Veness, J.: An approximation of the universal intelligence measure. In: Algorithmic Probability and Friends, pp. 236–249. Springer (2013)
12. Leike, J., Hutter, M.: Bad universal priors and notions of optimality. In: Conference on Learning Theory. pp. 1244–1259. PMLR (2015)
13. Silver, D., Singh, S., Precup, D., Sutton, R.: Reward is enough. AI (2021)