Mathematical definition of "intelligence" (abbreviated version)

Warren D. Smith*
warren.wds@gmail.com

June 18, 2006

Abstract — **This is an abbreviated version of a 6-times longer work. (1) We propose a mathematical definition of intelligence, i.e. know what intelligence is and can prove theorems about it. (2) The most important theorem is our construction of a UACI (uniformly asymptotically competitive intelligence) which is at least as intelligent in the long run as any competing entity (on the same hardware) as measured by any intelligence test. (3) Psychological experiments and facts provide evidence that human intelligence actually is constructed in essentially the same manner as the UACI construction. No one piece of that evidence is very convincing but the combination is fairly imposing. (4) It is now possible and desirable to set up an annual fully mechanized "intelligence contest" with standardized, reproducible intelligence measurements made for each contestant. That would virtually assure increased measured intelligence every year. It took about 50 years for computers to become superior to humans in chess and that probably would not have happened without the annual computer chess contests and "chess ratings" keeping it sane and measurable. We suggest that the same will be true of AI.**

There is some similarity between our definition of "intelligence" and our "UACI" construction, versus Turing's definition of "algorithm" and his construction of a "universal Turing machine."

Unusually, a box at the end gives a short proposed "consensus statement" about intelligence and we seek people willing to sign that statement.

The longer work on which this is based, is [41].

1 Motivation

Here are four reasons to study "what is intelligence?":

1. A numerical comparison of the crudest possible upper bounds on the raw information processing speed of human brains and 2005-era computers (based on comparing the CMOS transistor-pair counts in recent Intel processors with the synapse counts in the human brain, and multiplying by their frequencies of operation) shows the latter are superior: 286×10^6 transistors times 3.6GHz$= 5.1 \times 10^{17}$ bit-ops/second for an Intel Xeon, versus under 10^{15} synapses [29][5] times 400Hz$\leq 4 \times 10^{17}$ bit-ops/second for a human brain.

2. People want to build AIs.

3. The undefined notion of a "conscious intelligent observer" appears important to interpreting quantum mechanics.

4. Previous AI workers have either admitted failure at providing a definition of intelligence [52] or have provided clearly wrong or inadequate [31] attempts; and the psychometricians have been even worse, e.g. A.R.Jensen ([16] p.48) commenting that "My study of these two symposia [in 1921 and 1986 aiming to define intelligence]... has convinced me that psychologists are incapable of reaching a consensus... It has proved to be a hopeless quest."

The "Turing test" for intelligence [50] is inadequate because it is not useful for attempted software development of an AI: It is extremely inefficient and cannot be automated; it does not produce a numerical measure of intelligence or even a $<$, $=$, or $>$ comparison between two intelligences; it is not a mathematical definition; it has no intrinsic meaning in the absence of humans; has little or no predictive power about the nature of intelligent entities; and provides no understanding of what "intelligence" really is.

2 Preparatory thoughts

Our basic ideas are that

1. a usefully intelligent entity is one that can supply good "answers" to "questions" (both are general bitstrings), and what matters is not initial competence but rather asymptotic final competence at answering questions of that type.

2. we recognize that only NP questions and P-verifiable answers are needed (up to some technical issues concerning randomness) and then permit entirely general such questions,

3. we argue that any entity that cannot score well on certain such tests is *not* intelligent, while any entity that can score optimally well on all such tests is extremely intelligent even if it can do nothing besides scoring well on the test (thus "proving" that our definition is "correct")

4. somewhat sneakily implicit in that reasoning is a "lookahead" to §4 where we shall construct a "UACI," i.e. an entity which *does* score optimally well (asymptotically in a competitive sense) on every such intelligence test, and to §7 where it is conjectured that the working mechanism behind human intelligence *is* such a UACI.

Final but not initial competence is plainly what people have in mind when they claim "humans are intelligent but animals

*Non-electronic mail to: 21 Shore Oaks Drive, Stony Brook NY 11790.

not" since human infants are clearly less competent than most other newborn animals in essentially every way, and also even adult humans are initially far less competent at various tasks than birds (flying, memorizing hiding places for seeds), even though with effort and time humans eventually can become superior to birds.

P-verifiable answers are necessary because otherwise it is not possible to justify the validity of the scoring of intelligence test answers to some skeptic, nor it is necessarily feasible to produce those scores at all. Once that is insisted upon, NP [12] questions are an automatic consequence. Entities which cannot ever learn to answer such questions as "find a factor of 9797" or "find an escape route from this maze" are clearly *not* intelligent. Entities which can answer arbitrary NP questions – or merely answer them as well as any polytime randomized algorithm possibly can – are clearly intelligent. Why? Because even if they can do nothing other than this, that still suffices to have superior ability to any human mathematician ("find a proof of the Riemann hypothesis less than 200 pages long?") theoretical physicist ("find a theory, with proof of validity all less than 200 pages long, that explains the following experimental data XXXX?"), computer programmer ("find an algorithm, with formal correctness and polynomial runtime proofs, all less than 200 pages long, and the smallest possible polynomial bound, for the following formally specified task XXXX?")[1] etc.

3 Formal definition of Intelligence

(Alternative "deluxe versions" of our definition, aiming for improved performance, also are possible and are discussed in the nonabbreviated version.)

We employ the usual (since the days of Church and Turing) underlying computational model – a machine polynomially equivalent to a Turing machine. As is well known, an "algorithm" is a computer program that terminates no matter what the input, and it is a "polynomial time" algorithm if it does so in time bounded by a polynomial function of the input's bit-length. Let us now define something less familiar.

A **reent-algorithm** means a computer program that *never* terminates, and which keeps soliciting and accepting input bitstrings from one or several parties, outputting bitstrings in between. A reent-algorithm is "polynomial time" if the time it consumes to produce each output is bounded by a polynomial function of the total bit-length of all the inputs it has received so far. ("Reent" stands for "reentrant"; the concept is of a program that carries on an *interactive* dialogue [or several dialogues], as opposed to the old concept of "algorithm" which is a *batch* concept.) We shall also sometimes permit reent-algorithms to access a source of random bits.

Cast of characters:
PG: Problem generator
SC: Solution checker
ET: Entity under test

An **intelligence test** consists of one polynomial-time reent-algorithm PG and one polynomial time ordinary algorithm SC. The first reent-algorithm, called the "problem generator" (PG), uses random bits, and outputs an infinite sequence P_1, P_2,..., of output bitstrings called "problems;" the kth time the entity under test tells the problem generator "ready" it (beginning the next "*cycle*") outputs the next problem P_k, and it also outputs a second bitstring D_k called the "secret associated data" – but the entity under test (ET) is only allowed to see P_k and is never allowed to see any of the D_k. The second algorithm is called the "solution checker" (SC). As its input, it reads the problem P_k spit out by PG, and it *also* gets to read the secret associated data D_k. Finally, it reads as its third input, a communication from the entity under test called the "answer" A_k. It then outputs a "score" integer $S_k \geq 0$ which is a function of P_k, D_k, and A_k. It is essential that ET is never told what the underlying algorithm inside PG is, although it is probably permissible to make the algorithm inside SC public.

ET, after receiving P_k, is allowed to submit any number of trial answers A_k to SC for scoring, i.e. is allowed to invoke SC at any time to score any proposed trial answer for the current P_k. Only the *final* answer ET submits for P_k before requesting the next problem P_{k+1} from PG, corresponds to the final score S_k it gets on problem k. ET's *cumulative score* after time T, is $\sum_{k=1}^{K} S_k$ where K is the number of cycles completed before T.

There could be many possible (PG,SC) pairs, each one generating a different intelligence test.

Entities that get higher cumulative test scores as a function of time T are "more intelligent" at least as far as that test is concerned. If some entity 1 is at least as intelligent with respect to every test (or at least every test from some set under consideration) than entity 2 (and more intelligent on some tests) in the limit $T \to \infty$, then it is simply "more intelligent."

We have allowed ET to be literally any entity. However, for the purpose of studying Artificial Intelligence it is convenient to consider "entities" which in fact are *reent-algorithms* – probably random-bit-using ones – and preferably *polynomial-time* random-bit-using reent-algorithms.

4 The UACI Theorem

Having a formal mathematical definition of intelligence enables proving theorems about intelligence. The most important one is our construction of a UACI – uniformly asymptotically competitive intelligence – which, we show, is asymptotically as intelligent (up to a constant factor, which in fact in suitable models of computation is just 1) as *any* other entity, and this is true on every intelligence test simultaneously. Unfortunately this UACI consumes time exponential (2^ℓ) in the codelength ℓ of the competitor entity, but another theorem shows (under widely believed computational complexity conjectures) that is unavoidable, i.e. best possible.

1. As a warm-up, let us first consider a slightly altered (and not recommended!) version of our "definition of intelligence"

[1] Our point in these three examples is that (1) all are NP-questions, (2) an optimally competitive algorithm for answering them would by definition be superior to any human; and (3) any entity that *could* accomplish these feats would clearly be a superior mathematician, physicist etc versus any human, because all human mathematicians, physicists, etc are presently incapable of these feats.

in which PG and SC are the same entity and both are fully deterministic.

Let an "*asymptotically* godlike superintelligence" be a polynomial time reent-algorithm which asymptotically on a long sequence of intelligence test cycles generated by the same *deterministic* polynomial time test-problem-and-answer generator reent-algorithm, always achieves *asymptotically* the maximum achieveable cumulative score (provided that answers that would yield unboundedly large cumulative score totals exist).

Theorem: *Asymptotically godlike superintelligence* is *possible.*

Proof: The idea is to guess the test-problem generation algorithm[2] by successively systematically trying all possible algorithms. Each cycle a new guess is tried until one is found that agrees with all problem-solution pairs so far, then we just stay with it until a disagreement occurs, then resume guessing. Eventually (i.e. after some very large but finite number of cycles) the right guess will be found and stayed with forever after, causing optimally godlike superintelligence from then on.

There are a few tricks we need to explain in order to justify this:

1. We need to know that every polynomial time algorithm may be written in a "self proving" fashion which is a priori *known* to be a polynomial time algorithm with a *known polynomial* as its runtime bound. My favorite way to do that is to make the algorithm have a standardized straight-line-code *preface* that (1) reads its input and (2) computes a polynomial P of its bitlength N; (and does so in a clear, straightforward, standardized manner so that it is obvious that is what it is doing) and then the *remainder* of the algorithm decrements P as a side-effect of every step it takes, self-terminating as soon as P hits zero. [Note: similar remarks instead may be made about *exponential-time* algorithms, but *not* about *all* algorithms.]
2. In that way we can generate all polynomial time algorithms in, say, lexicographic order, but without generating any super-polynomial-time programs.
3. It also is possible in numerous ways, in our intelligence test problem-answer-cycle framework, to systematically do "time sharing" among all possible such algorithms in such a way as still to keep the combined creature a polynomial-time reent-algorithm. For example, if an algorithm has worst-case time bound KN^D, then we can run it for N steps each cycle (and if not yet done, continue running it N steps the next cycle but still using the old data, and so on, until it gets done). If each cycle we add a new trial algorithm to our collection, the next effect is that N cycles get completed in $O(N^3)$ time so that we plainly have a polynomial-time reent algorithm, but with the property that every trial algorithm eventually is run on an unboundedly large number of P_k, thus assuring that one with 100% success rate eventually will be found.

Q.E.D.

Extension: Indeed, by continuing to explore all algorithms permanently with 50% of one's computer time, but using the other 50% to run the currently-best algorithm, we not only can obtain asymptotically godlike superintelligence in the above scenario, but in fact we can do so while staying within an asymptotic *factor of two* of optimizing a measure of *computational efficiency.* Even better one can run it a fraction $\max\{1/2, 1-10^{10}/\sqrt{n}\}$ of the time on the nth test-cycle, thus getting *100% efficiency* asymptotically in an appropriate computational model.

Warning: the preceding theorem and extension depended heavily on the wrong assumptions that the test-problem generator is *deterministic* and also generates the *answer.* In reality PG is randomized and a separate scoring routine SC evaluates answers (which need not be unique and which quite possibly cannot be deduced from the problems in polynomia time). In the preceding theorem, using a cryptographically strong pseudorandom number generator would have been fine, *but* the proof breaks if the test-problem generator is allowed access to a *true* random-bit generator. Indeed,

2. Theorem: *An asymptotically godlike superintelligence is* not *possible if the polynomial time test-problem-generation-and-test reent-algorithm instead has access to a true random bit generator.*

The **proof** is trivial: on the Nth cycle, demand an N-bit answer and award score 1 if the answer matches a sequence of N freshly-generated coin tosses, otherwise award score 0.

Then the total expected score for any intelligence whatever, even cumulated over an infinitely long intelligence test, is ≤ 1, but the maximum possible score is infinite. Q.E.D.

We now state our **most important theorem:**

3. *A "uniformly asymptotically competitive intelligence" (UACI)* is a randomized reent-algorithm C which, when repeatedly given any intelligence test, achieves an *expected* cumulative score at least asymptotically equal to that achieved by *any* particular other randomized polytime reent-algorithm X on the same test sequence, while consuming compute time at most a constant factor larger than those consumed by X, and memory resources growing at most linearly with time.

The UACI Theorem: *A UACI* is *possible.*

Proof: The idea is to guess the competitor's algorithm by successively systematically considering all possible polynomial time reent-algorithms. Each cycle a new guess is considered. If one is found that would have yielded a larger expected cumulative score throughout past test problem-solution-pair history, then we switch to using it to generate our test answers.

Notes: we estimate cumulative expected scores as follows: each cycle, we, for all candidate-algorithms in our current collection, try another set of random input bits on all of their past history, and update that candidate algorithm's expected-score-estimate appropriately. By the law of large numbers, ultimately the expected-score estimates will (with probability 1) approach their true values for any particular candidate algorithm up to any particular time. By switching, at some point, to exhaustive enumeration of all 2^n bitstrings with n bits rather than Monte Carlo sampling, we can in fact determine the *exact* expected-score up to the earliest point of

[2]For brevity, we shall often use the word "algorithm" when we mean "reent-algorithm."

consumption of the nth random bit, not merely an estimate (while still consuming only polynomial space). So eventually the right guess for X will be found (or something as good or better) and stayed with forever after, resulting in at least competitive performance from then on.

Note 2: to make this all work with at most polynomial slowdown, we need to use the same tricks as in the preceding proof, plus a few more. Since the candidate algorithms are eating data at different rates, we of course need to keep track of all their "cumulative scores" as well as their consumed "times" in order that we may compare apples with apples. Ultimately asymptotically all algorithms consume the same amount of "time" so the comparisons will be asymptotically fair.

Note 3: The reader might worry that, on some task, there might be some sequence of polynomial-time algorithms, say with runtime N^k for the kth algorithm, which achieve greater and greater scores, e.g. cumulative score proportional to k^2 after k cycles if we switch to algorithm k at cycle k. Therefore, our UACI might find these algorithms successively, with the net effect of finding an algorithm that really has super-polynomial runtime.

Avoiding that issue is in fact precisely why in §3 we defined the cumulative score to be a function of *time* T and *not* of the number of test cycles so far. If the score-producing beast SC pre-transforms its scores by some appropriate monotonic pre-transformation function it can encourage the intelligence to prefer just *one* of the algorithms in that sequence in order to get good scores without taking too much runtime to do it – while if a super-polynomial-time reent algorithm then yields superior asymptotic performance per unit time to any polynomial algorithm, it indeed will be preferred, but that is then a feature, not a bug.

Note 4: We have discussed "provable polytime algorithms" and their generation in a previous proof; we of course also reuse that trick here.

Note 5: We will never be *sure* that duplication of the best possible competitor has occurred, hence will need to continue experimenting forever, causing a slowdown by a possibly-large, even though polynomially bounded, asymptotic factor. But by devoting 50% of runtime to the best currently-known candidate algoritm and 50% of runtime to the ongoing search for improved ones, the asymptotic slowdown factor can be made to be 2, and indeed (as we explained last proof) even $1+\epsilon$. Q.E.D.

Despite whatever theoretical claims of grandeur this existence theorem has, from a practical point of view it is not terribly useful because the proof technique – even though constructive – takes a very long time (exponential in the code-length of the competitor algorithm) before the competitor algorithm is duplicated allowing asymptopia to set in.

It is possible to address this criticism to some extent as we shall see in §9 and 6. Before doing so we also point out

4. The proofs of the preceding theorems also show that the code for an UACI can without loss of generality and without loss of performance (except for polynomially bounded factors) be required to be *short*. I.e. the "Kolmogorov complexity" (code length) of a universally asymptotically competitive intelligence is remarkably small.[3]

Indeed, it is not difficult to write down in full and complete detail, a program for an UACI, in some standard computer language such as `C` or `Scheme`.

5. Although the strategy of "searching over all possible algorithms" employed in theorem 3's construction of a UACI may seem (and is) very crude and inefficient, there are good reasons to believe that it is **not possible to do better**. More precisely: We can **prove** under standard computational complexity conjectures that it is *not possible* to find the best algorithm (or even one merely *comparable* to the best one), even if among *quadatic-time* algorithms describable in N bits, in worst case time below exponential (2^N) in the description length N of that algorithm. Indeed, it is a standard conjecture that "breaking the AES secret key cryptosystem" (i.e. given plaintext-ciphertext pairs for an N-bit long secret-key cryptosystem of the same ilk as AES [7], find the N-bit-long secret key that encodes the encryption algorithm) cannot be done in subexponential (below 2^N) time.[4]

Therefore, the only hope for improving theorem 3's crude UACI construction is to ignore the UACI's worst case performance (which already is optimal) and try instead to improve its performance on *good* cases while still not diminishing performance too greatly in bad cases. But a limit on the ability to do that is set by the

Two-way UACI simulation theorem: *Any two UACIs A, B are equivalent in the sense that A can (and will) simulate B and B will simulate A with at most a constant additive slowdown (note: this constant may depend on A and B and may be very large) plus $\le (1+\epsilon)$ multiplicative-factor slowdown (this is valid for any $\epsilon > 0$).*

(This theorem is an immediate consequence of the UACI theorem.)

Our UACI is a natural outgrowth of previous "universality" ideas [20][14] in computer science dating back to Turing [51] in 1936.

[3] And indeed a companion paper [40] analyses the description-length of the biological "blueprint" for *human* intelligence and concludes that either 6 or 81 Kbytes suffice, up to a factor of 3 worth of imprecision in the estimate, in two different models (specifically: it depends whether you believe "exons and introns" are important for regulation or not). This is not very large. Solo humans have written considerably larger computer programs.

[4] "AES-like cryptosystems" work as follows: each "stage," the plaintext is transformed by one of two reversible transformations T_0 or T_1 each with highly-bit-scrambling effects. At the kth stage one performs T_b where b is the kth bit of the secret key. The net result of composing all N of these transformations (arising from an N-bit key) is the "ciphertext." Note that the "key" here really is an N-bit-long description of an encryption (and the corresponding decryption) "algorithm" and anything capable of guessing that algorithm is capable of "breaking the cryptosystem," i.e. of rapidly producing plaintexts corresponding to given ciphertexts. So far, we have described "Feistel cryptosystems." However, as a description of the AES, it has been oversimplified; it is an important *protective modification* that the transformations T_b actually incorporate the *whole* key not just one bit, because otherwise AES would be attackable by "fast Gray code update" and "meet in the middle" attacks. It is a very widely believed conjecture that it is not possible to break such cryptosystems in less than 2^N steps on average.

5 Correct treatment of complexity classes

We have adopted the oversimplification of only talking about the deterministic computational complexity classes P and NP, even though really, we should have been permitting *randomization*. We now give the appropriate replacement classes and some additional discussion.

	randomized	deterministic
1	BPP	P
2	N(BPP)	NP
3	ME(FP)	NP
4	PSPACE	PSPACE
5		EXPTIME

Figure 5.1. Important intelligence-related computational complexity classes (explained in the text). ME(FP) appears not to have been studied before and is defined here for the first time. **Notation:** "N" is an *operator* which converts a class T of tasks into the class NT whose answers are *verifiable* by a computation in the class T. "#" is an operator such that #T is the class of problems of *counting the number of solutions* of some problem in the class T. "ME" is defined below. ▲

ME(FP): Our notation regards "ME" as an *operator* which converts a class T of functions (here T=FP, which is the class of polynomial-time functions f that convert bitstrings to binary integers) into the class of problems of the form "maximize the expected value of $f(x)$ by appropriately choosing its input bits x" where some known *subset* of those input bits are choosable whereas the complement subset are chosen randomly by coin tosses and are not controllable by us.

The problem faced by an intelligence whose answers are scored with a binary integer by a polynomial time scoring device that employs random bits, is: maximize your expected score! That is a ME(FP) problem; but if no random bits are used it is just NP. Of course that was assuming that SC and PG are known – but if they are not known, then the problem of guessing them given the known data (with the aim, e.g. of maximizing the correctness probability of the guess) is also an NP or ME(FP) problem.

Our main theorems about these complexity classes (which are proved in the full paper, here we only state them) are:

1. ME(FP) completeness Theorem. *The following problem "probability-maximization SAT" is complete over ME(FP), i.e. any ME(FP) problem can be solved in polynomial time if we have access to an oracle for solving probability-maximization SAT instances.*

INSTANCE: There is a known N-bit-input, 1-bit-output polytime forward-only boolean logic circuit, which also accepts N more inputs from random coin-toss bits, for $2N$ inputs in total.

PROBLEM: to find the N-bit input which maximizes the probability the output bit is "on."

2. Complexity class inclusion Theorem.

$$\mathrm{P}\subseteq\mathrm{NP}\subseteq\mathrm{PH}\subseteq\#\mathrm{P}\subseteq\mathrm{ME(FP)}\subseteq\mathrm{N}(\mathrm{P}^{\#P})\subseteq\mathrm{PSPACE}$$

where "$A \subseteq B$" here is taken to mean[5] *that problems in class A can be solved easily (in polynomial time) if we have an oracle that will solve problems in class B on demand. Also (Sipser [38]) P⊆BPP⊆PH and (Savitch) N(PSPACE)=PSPACE.*

Notation: P is polynomial time. "SAT" is standard computer science lingo for the "boolean satisfiability problem," which is Cook's standard NP-complete problem [12]; "probability-maximization SAT" is the name of *our* new class of problems which however is highly related to Cook's original SAT class. PH is the "polynomial hierarchy" of problems soluble in polynomial time by a machine that has access to an NP oracle (this is P^{NP}), in polytime by a higher-level machine with access to an oracle for *that*, in polytime by a still-higher-level machine with access to an oracle for *that*, etc. (These are the successive levels of the hierarchy.) Two problems known to be #P-complete are "counting SAT" and "permanent of an integer matrix." Supserscripting A^B denotes the class of problems A *but* allowing the solver access to an *oracle* for solving problems in class B on demand.

6 Exhaustive search that is faster than brute force

The naive method for trying out A algorithms, each one running for time T, takes at least AT steps. We shall now give theorems showing that, at least if we restrict ourselves to exhaustive searches over certain important subsets of algorithms, this search-over-algorithms can instead be accomplished in $O(A)$ time, i.e. $O(1)$ steps per algorithm.

Faster-than-brute-force Theorem. *Tree-structured loopless algorithms with N tree nodes each selected from a finite palette of possible functions of their children (and with each node having a bounded number of children), may be exhaustively generated and run on fixed input, in $O(1)$ average time per algorithm.*

Proof sketch. The idea is to generate all the trees in a manner such that each tree differs from the previous by $O(1)$, i.e. a bounded number of alterations. Here the "$O(1)$" includes counting every node which has an altered descendent, as itself "altered." Then we re-evaluate the new tree (i.e. run the new algorithm) by only updating the stored intermediate quantities corresponding to the updated tree nodes.

Lucas's "Gray code for binary trees" [23] makes it possible to visit all N-node binary trees exactly once by performing a single tree "edge rotation" each time to move from one tree to the next. (There is a well known "planar duality" bijection between N-node binary trees and triangulations of a convex $(N+2)$-gon, in which each "edge rotation" in the tree corresponds to a "quadrilateral diagonal flip" in the triangulated polygon.) We propose to use a modification resembling [24] of Lucas's original scheme. Our modified generator actually sometimes performs *more* than one edge-rotation between generated trees, but still only performs a constant number of them *on average*. (For non-binary trees see [18]; everything

[5] This convention is convenient but slightly nonstandard.

we say about binary trees can be generalized to them.[6]) This modification is simple to program, and involves only $O(1)$ computational work per tree generated. Furthermore[7] there is a modified version of the [24] tree-generation algorithm features *average depth* (i.e. distance to the tree root) ≤ 3 to each rotated edge. Q.E.D.

The full paper discusses more results of this ilk, and see also [34][26].

7 Human intelligence vis-a-vis our definition

Let the **Human UACI Hypothesis (HUH)** denote the (intentionally vaguely phrased) assertion that human intelligence is built in essentially the same way as our UACI construction in §4.

There are four main lines of evidence for HUH, each line containing numerous[8] items of experimental evidence. While none of these evidence-items by itself is very convincing, the combination is. (It is rather like seeing 1 square inch of the skin of an elephant and trying to prove you have an elephant. Not very convincing – but if you do this for 50 different independent square inches, then it becomes so.) We now discuss the four lines extremely briefly; the full paper goes into much greater detail.

1. Spearman g and the 1-dimensional nature of human IQ. Spearman in 1904 found [48], and other experimental research confirmed [16][49][25][8] (to a large degree) two principles about human intelligence: **(a)** the "positive correlation principle" that human performance on any two mental tests is *positively correlated* (i.e. if you are better than average at French, then you tend to be better than average at Algebra), **(b)** the "one dimensionality principle" that human intelligence scatterplotted in high-dimensional space as performances on a large number of mental tests, is distributed like a multidimensional ellipsoidal-shaped Gaussian, where the ellipsoid is "needlelike," i.e. "one dimensional" in the sense that *one* of its axes (dubbed "Spearman's g") is much longer than the others.

In the actual psychometric literature this all is far less well and less concisely described – and the psychometricians are unaware of many fundamental theorems from linear algebra [13] that their subject rests on – and the experimental evidence for the Spearman principles is less convincing than usually claimed. (All that is reviewed in the full paper.) Nevertheless, to the (considerable) extent that they are true, both Spearman principles, and the observed highly general, adaptible, and universal nature of human intelligence, seem compatible with the HUH.

2. Piaget's notions of mental development of children. Piaget and successors found [37] that many aspects of intelligence in children develop in a fairly predictable sequence of stages. The sequence happens in the same order regardless of culture and consists of successive transformations of less-adequate into more-adequate notions. This seems compatible with the HUH assuming the UACI employs "Piagetian search" over algorithms in which successive *refinements* of good-performing algorithms are systematically explored with replacement in an attempt to get better performance.

3. Forgetfulness. The fact that humans forget things [2][22][47][53] is astounding considering the incredible ease with which computer programmers can prevent forgetting. How can human intelligence be so far ahead of computer programmers *but* so incredibly bad when it comes to memory? (And animals remember things forever, such as salmon remembering their birthplace stream, so there clearly is nothing biological to inherently prevent permanent memory.) Assume the HUH with a UACI based on a randomized Piagetian search. Then the UACI if trained to solve (say) SAT problems will eventually become good at that. If then suddenly the goal changes to proving theorems from axioms, it will perform badly but would eventually become good at that too. *But* if PG then switches back to posing SAT problems, the UACI will *not* immediately regain its old competence, although it will be expected to relearn faster than it originally learned. It is hard to prevent the UACI from thus "forgetting" how to solve SAT problems. Yes, one could clone it as soon as the SAT-to-theorem-proving transition occurred, but there is no obvious transition (the new problems might be just a new form of the old problems, all the UACI sees is bitstrings) and the UACI might for all it knows be about to see some more

[6] Indeed we remark that the *general* rooted ordered trees with N nodes can be represented as, i.e. are in 1-to-1 correspondence with, the *binary* rooted ordered trees with $N-1$ parent-to-left-child arcs by making the rightward paths in the binary tree correspond to the nodes in the general tree. This remark actually is not quite sufficient for our purposes, but it goes a long way.

[7] This remarkable fact was not stated in [24] but was realized by its author Frank Ruskey and myself in private emails. The proof is: The [24] tree generation algorithm is based on the fact that the N-node binary trees can be got from the $(N-1)$-node trees by adding an Nth node somewhere on the path of successive right-children of the root, and then the rest of that path (below the now-inserted Nth node) needs to be made a left-child of the new node. This allows us to generate all N-node (rooted ordered) binary trees by recursively generating all $(N-1)$-node trees – with the new Nth node being irrelevant to that since it just passively hangs off the end of the right-child-path as the trees fluctuate. And then, in between fluctuations, for each $(N-1)$-node tree, we "walk" the Nth node up and then back down the right-child path by means of rotations. The crucial lemma, an early form of which was pointed out to me by Ruskey, then is that the average length of this right-child-path is ≤ 3. That is because the average number of nodes in the path from the root to the rightmost leaf node in a random N-node (rooted ordered) binary tree is precisely $3N/(N+2)$. This follows from the fact that the number $T(N,k)$ of binary trees with a k-node right-child path, $0 \le k \le N$, is $k\binom{2N-k-1}{N-k}/N$ and then

$$\frac{\sum_{k=0}^{N} k^2 \binom{2N-k-1}{N-k}/N}{\sum_{k=0}^{N} k \binom{2N-k-1}{N-k}/N} = \frac{3N}{N+2} < 3.$$

This and the easier fact that the number of N-node binary (rooted ordered) trees is $T(N) = \binom{2N}{N}/(N+1)$ both may be proven by consideration of recurrences such as $T(n) = \sum_{a,b\ge 0,\ a+b+1=n} T(a)T(b)$ and $T(n,k) = \sum_{a,b\ge 0,\ a+b+1=n} T(a)T(b,k-1)$ if $k \ge 1$ and $T(n,k) = 0$ if $k > n$ with $T(n,n) = 1$ for all $n \ge 0$. They also may be attacked via generating function identities concerning $F(x) = \sum_{n\ge 0} T(n)x^n$ and $G_k(x) = \sum_{n\ge 0} T(n,k)x^n$ e.g. $F(x) = 1 + xF(x)^2$ so that $F(x) = (1-\sqrt{1-4x})/(2x)$, and and $G_k(x) = T(0,k) + xF(x)G_{k-1}(x)$ so that $G_k(x) = [xF(x)]^k$.

[8] Between 12 and thousands.

SAT problems soon. Also we conjecture that human hardware limitations prevent easy copying of large programs anyhow. And for very complicated programs that solve varieties of problems and reuse each other's subroutines, it is very hard to tell what *is* the "SAT-solving part" of the program, so that you can "save it in a box" to prevent forgetting. We look at this in more detail and hopefully clarity in the full paper, but suffice it to say that this kind of human phenomenon seems entirely compatible with, indeed predicted by, the HUH.

4. Human time-consumption behavior. Humans take enormous amounts of time to invent/discover things, but then use them to solve problems at a far more rapid rate. For example learning to walk, learning to solve Rubik's cube, and learning to multiply numbers. This behavior is entirely compatible with our UACI construction's "exponential roll out." Also humans exhibit "power law learning" curves of improvement with "practice" [28][32]. The full paper proves a theorem saying that for a certain explicit and reasonably large-seeming class of problems (under standard computational complexity assumptions) UACIs will also exhibit power law learning curves.

8 Consciousness – still a mystery?

Although psychologists had experienced immense difficulty trying to devise a consensus definition of "intelligence" (a quest which hopefully has now ended), "consciousness" is an even *more* murky and elusive concept. Because of that murk, we cannot confidently provide a consensus-inspiring definition. Nevertheless we try by providing a "tentative proposal" of a definition:

Tentative Proposal: A consciousness is "an intelligent entity which interacts with some law-obeying randomized *external environment* in an effort to increase some kind of numerical *reward*."

If it is accepted, then consciousness is trivialized because any "intelligent" entity by our definition automatically is conscious! (The full paper provides a negative discussion that refutes all the most commonly-heard alternative notions about consciousness.)

Sleep. The most dramatic feature of human consciousness is the fact that it is turned off one-third of the time. The full paper provides a long discussion of sleep. What is our rationale for including that discussion? We claim that sleep is a logically-crucial issue for the following reason. We have provided a large amount of confirmatory evidence for the Human UACI Hypothesis (HUH) extracted from the experimental psychology literature. A critic might now carp that perhaps that evidence was "cherrypicked," i.e. that I looked through the psychology literature seeking confirmatory evidence but ignoring evidence that mitigated against the HUH. That is not the case – my search simply did not uncover any countervailing evidence. However, the closest thing I know to countervailing evidence, is sleep! That is because the HUH does *not* predict sleep. HUH also does not forbid sleep, but conceivably some other hypothesis about how human intelligence works, *would* predict sleep – and if it also predicted all the same phenomena that HUH predicts, then the experimental confirmation of that other hypothesis would have to be judged superior. Furthermore, the full paper exhibits two kinds of computer programs whose performance is inherently increased by "sleep," causing us to worry that there indeed *might* be such an alternative hypothesis lurking. That is the full paper's rationale for examining known facts about sleep in considerable detail, and it finds (in agreement with previous workers [35][36]) that the most obvious guesses about sleep all are known to be false. The only proposed sleep explanations that currently appear still to be standing are the hypotheses that sleep is either an evolutionary accident or merely intended to keep animals "quiet and out of trouble," and for many animals – to a large extent including for humans – sleep is in no way necessary nor even helpful for any known facet of our intelligence. Indeed, it appears likely that quite-intelligent animals exist that never sleep. In view of this (pending any increases in our understanding of sleep) we conclude that sleep *cannot* constitute a challenge to the HUH.

9 Toward practicality

The full paper contains a long discussion, including many ideas and thought experiments, of how to make a more practical and "less brute force" UACI. There is immense scope for such improvements and it is entirely unclear how far and fast they can be pushed. However the analysis in the full paper makes it at least plausible that at least a mild degree of competence by human standards may be attainable.

This effort could easily require 50 years of work by an entire community. To keep such an effort sane and measure its progress, we propose setting up standardized "intelligence contests." Specifically, there could be a web site on which is found a standard intelligence-test protocol, with both intelligent entities (ET) and additional intelligence tests (PG and SC) being provided continually by anybody in the world, and with record performance on each kind of test being recorded.

To make an analogy, it took about 50 years of work by an entire community, starting from the earliest experiments with computer chess players, before reaching the point where computers surpassed human chess strength. This effort included many "false starts" (i.e. directions of research which ultimately seemed not useful), misconceptions, and wrong estimates. Many would argue that this effort would not have succeeded without annual computer chess tournaments and chess ratings which quantitatively measured progress and thus kept eveything sane. The problem of creating an artificial intelligence is far more difficult and up to the present moment there has not been any corresponding mechanism for keeping the area sane and for quantitatively measuring progress. We are saying that now, such a mechanism can be created, fairly easily. The difficulty of creating it would be approximately comparable to the difficulty of creating internet "game servers" such as the "generic game server" [6] and the "internet chess club." The full paper gives a more precise descritpion of how to do this.

10 Previous work by Hutter

This work can be viewed as a rediscovery of ideas by Marcus Hutter published in a 2004 book [15]. The full paper includes a section giving an extensive survey of the relationships between our two works. Hutter's and our terminology, topics, and attitudes differ but we both have essentially the same few core ideas – which are actually quite simple – namely we both have similar "definitions of intelligence" and we both have versions of "the UACI theorem." Hutter actually goes further in the sense that he defines a 2-parameter continuum (he calls it "AiXi($t\ell$)") of "intelligences" while we only concern ourselves with the "lowest point" on that continuum; and Hutter employs more sophisticated techniques to accomplish that. Also Hutter, building on work of Solomonoff, has interesting "convergence theorems" whose impact is unclear to me. (I.e. I find it unclear whether these are really advances over our own simpler theorems and if so how much that matters.) But the full paper gives arguments that Hutter's extra generality is ultimately undesirable. Hutter unlike us did not examine the human-psychology literature nor did he find our faster-than-brute-force search theorem. Finally, in some papers Hutter wrote with Legg [19] after his book, he made what in our view is a mistake by advancing the notion that there is a "universal intelligence test." This directly contradicts our idea that there are an infinite number of intelligence tests, although there is a universal intelligent entity (UACI) that performs competitively-optimally on them all. It would be wonderful if there really were a universal intelligence test, but, at least with our definition of intelligence, we can prove there is no such thing:

Theorem (no universal IQ test exists): *Any (putatively universal) intelligence test U of the form in §3 has the property that some entity ET exists, that performs asymptotically optimally on it,* but *performs pessimally on some other intelligence test B.*

Proof. To get asymptotically competitively optimal behavior on U, simply make ET be a UACI as in theorem 3 of §4. To create B so that the ET will always score *zero* on B, add a special modification to ET to detect test problems of B's form, and to deliver appropriately bad answers whenever that detection happens. (We can design B to always output problems in a special easy-to-detect format, such as an all-1s bitstring, and B's scorer will demand a certain easily-evadable kind of answer to get a nonzero score, such as an all-0s bitstring.)

To complete the proof, we need to argue that a suitable B and ET-modification both exist such that the UACI's asymptotically optimal cumulative score on test U, is only negligibly diminished asymptotically. We can accomplish that by a "Cantor diagonalization" argument. The successive problems P_k output by U (or the successive *probability distributions* of the P_k if U is randomized) are considered. We can easily see ala Cantor that there necessarily will exist an alternative sequence of $\widetilde{P}_k$ that will necessarily *not* be generated by U – or only generated with negligible probability, indeed with *total* expected number of equalities $\widetilde{P}_j = P_k|_{1\le j<\infty,\, 1\le k\le j^3}$ upper bounded by an arbitarily small *constant* c, since, e.g, the expected count of $\widetilde{P}_j$ equalities is $\le 0.5cj^{-2}$. In fact, were U deterministic one could construct B by simply making B *be* U but with a postprocessing step added to alter U's output away from U's (and away from all of U's previous outputs) whereas for randomized U one could try $2k^2/c$ Monte-Carlo experiments (for each running U up to $k = j^3$) and then add a postprocessing step to pick an easy-to-recognize output not in the resulting $2j^9/c$-element set. Q.E.D.

The full paper goes on to see that there is (conjecturally) a way to partially salvage the Legg-Hutter idea for a universal intelligence test, albeit by going beyond the bounds of what is a permissible intelligence test under our definition – but it is probably of no practical interest. Thus the "universal IQ test" controversy has been essentially completely resolved.

11 Multiresearcher Consensus

I believe it would be useful – in the sense that it would prevent several years from being wasted – to get a number of prominent human- and artificial-intelligence researchers to sign the following short "consensus statement":

1. "Intelligence" and "intelligence test" both have mathematical definitions, which (perhaps up to minor alterations) can be taken to be the ones in §3 of the present work.
2. It is already known how – easily – to build a "universal artificial intelligence" that would meet this definition, but which unfortunately would perform poorly in practice.
3. The AI community should adopt the previous two points as the foundation for future research.
4. The AI community should organize a perpetually ongoing "intelligence contest" open to both human and computer intelligences as contestants, accepting standardized "intelligence tests" contributed by anyone, and posting scoring records of all contestants on all tests. This modus operandi should ensure that clear definable and measurable gains in machine intelligence happen every year.

So far (22 May 2006) nobody has signed it besides me.

References

[1] Some of these references are not cited in the text. This is a useful subset of the larger bibliography in [41].

[2] Alan Baddeley: Your memory, a user's guide, Firefly Books (new illustrated ed. 2004).

[3] Eric B. Baum: What is thought?, MIT Press 2004.

[4] Allan Borodin & Ran El-Yaniv: Online computation and competitive analysis, Cambridge University Press 1998.

[5] V.Braitenberg & A.Schüz: Anatomy of the cortex, Springer 1991.

[6] Michael Buro & Igor Durdanovic: An overview of NECI's generic game server, `http://www.cs.ualberta.ca/~mburo/ps/ggs.pdf`, `http://www.cs.ualberta.ca/~mburo/ggsa/`.

[7] Joan Daemen & Vincent Rijmen: The design of Rijndael, the Advanced Encryption Standard, Springer-Verlag 2003.

[8] Ian J. Deary: Intelligence, a very short introduction, Oxford Univ. Press 2001.

[9] Ian J. Deary: Looking down on human Intelligence, Oxford Univ. Press (psychology series #34) 2000.

[10] D-Z. Du & K-I. Ko: Theory of Computational Complexity, John Wiley & Sons 2000.

[11] S. Even & R.E. Tarjan: A combinatorial problem which is complete in polynomial space, J. Assoc. Computing Machinery 23 (1976) 710-719.

[12] M.R. Garey & D.S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.

[13] Roger A. Horn & Charles R. Johnson:, Matrix Analysis, Cambridge Univ. press, 1985.

[14] Marcus Hutter: The Fastest and Shortest Algorithm for All Well-Defined Problems, Int'l J. Foundations of Computer Science 13,3 (June 2002) 431-443.

[15] Marcus Hutter: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability, Springer, 300 pages, Berlin 2004. ISBN=3-540-22139-5. Most of Hutter's papers are cited in and/or incorporated into his book [15] and are available on his web page `http://www.idsia.ch/∼marcus/ai`.

[16] Arthur R. Jensen: The g factor: The science of mental ability. Westport, CT: Praeger 1998

[17] Arthur R. Jensen: Straight talk about mental tests, Free Press, New York 1981.

[18] J.F. Korsh & P. LaFollette: Multiset Permutations and Loopless Generation of Ordered Trees with Specified Degree Sequence, J. Algorithms 34,2 (2000) 309-336.

[19] Shane Legg & Marcus Hutter: A Universal Measure of Intelligence for Artificial Agents, IDSIA technical report 04-05 (Galleria 2, CH-6928 Manno-Lugano, Switzerland, April 2005); A Formal Measure of Machine Intelligence, IDSIA TR 10-06 April 2006 (8 pages) presented at Annual Machine Learning Conference of Belgium and The Netherlands (Benelearn-2006) and both are available on Hutter's web page `http://www.idsia.ch/∼marcus/official/publ.htm`.

[20] Leonid A. Levin: Universal sequential search problems, Problems of Information Transmission 9 (1973) 265-266; original Russian version: Problemy Peredaci Informacii 9,3 (1973) 115-116.

[21] M. Li & P.M.B. Vitanyi: An introduction to Kolmogorov complexity and its applications, Springer (2nd edition) 1997.

[22] E.L.Loftus: Eyewitness testimony, Harvard University Press 1996.

[23] Joan M. Lucas: The rotation graph of binary trees is hamiltonian, J. Algorithms 8,4 (1987) 503-535.

[24] J.M. Lucas, D. Roelants van Baronaigien, F. Ruskey: On Rotations and the Generation of Binary Trees, J. Algorithms 15,3 (1993) 343-366.

[25] N.J.Mackintosh: IQ and human Intelligence, Oxford University Press 1998.

[26] B.D. McKay: Isomorph-free exhaustive generation, J. Algorithms 26,2 (1998) 306-324.

[27] Marvin L. Minsky: Finite and infinute machines, Prentice-Hall 1967.

[28] A. Newell & P.S. Rosenbloom: Mechanisms of skill acquisition and the law of practice, 1-55 in J. Anderson (ed.) Cognitive Skills and Their Acquisition, Lawrence Erlbaum Associates, Hillsdale, NJ 1981.

[29] B.Pakkenberg & 6 others: Aging and the human neocortex, Exp'l. Gerontology 38 (2003) 95-99; B.Pakkenberg & H.J.G.Gundersen: Neocortical neuron number in humans: effect of sex and age, J. Comparative Neurology 384 (1997) 312-320.

[30] Christos H. Papadimitriou: Computational Complexity, Addison Wesley 1994.

[31] Elaine Rich & Kevin Knight: Artificial Intelligence (2nd ed.) McGraw Hill 1991.

[32] P.S. Rosenbloom, J.E. Laird, A. Newell: The chunking of skill and knowledge, 391-410 in Working models of human perception (B.A.G. Elsendoorn & H. Bouma eds.) Academic Press 1989.

[33] Stuart J, Russell & Peter Norvig: Artificial Intelligence: A Modern Approach, (2nd Edition) Prentice-Hall 2003.

[34] Carla Savage: A survey of combinatorial Gray codes, SIAM Review 39,4 (1997) 605-629.

[35] Jerome M. Siegel: Why we sleep, Scientific American (Nov. 2003) 92-97.

[36] Jerome M. Siegel: Clues to the functions of mammalian sleep, Nature 437 (2005) 1264-1271.

[37] R.S. Siegler & M.W.Alibali: Children's thinking (4th ed.), Prentice-Hall 2005.

[38] M.Sipser: A Complexity theoretic approach to randomness, Proceedings 15th ACM Symposium on Theory of Computing STOC (1983) 330-335.

[39] Michael Sipser: Introduction to the Theory of Computation, PWS publishers 1997.

[40] W.D.Smith: Information content of human intelligence and life, `http://math.temple.edu/∼wds/homepage/works.html` #94.

[41] W.D.Smith: Mathematical definition of intelligence (and consequences), `http://math.temple.edu/∼wds/homepage/works.html` #93.

[42] Ray Solomonoff: The Universal Distribution and Machine Learning, The Computer Journal 46,6 (Nov. 2003) 598-601.

[43] Ray Solomonoff: Three Kinds of Probabilistic Induction: Universal Distributions and Convergence Theorems, To appear in Festschrift for Chris Wallace.

[44] Ray Solomonoff: Progress in Incremental Machine Learning; Revision 2.0, 30 Oct. 2003, Given at NIPS Workshop on Universal Learning Algorithms and Optimal Search, Dec. 14, 2002, Whistler, B.C., Canada.

[45] Ray Solomonoff: Two Kinds of Probabilistic Induction, The Computer Journal 42,4 (1999) 256-259.

[46] Ray Solomonoff: A Formal Theory of Inductive Inference, I: Information and Control 7,1 (1964) 1-22; II: 7,2 (1964) 224-254. [All of these Solomonoff papers are available on his web page `http://world.std.com/∼rjs/pubs.html`.]

[47] Norman E. Spear & David C. Riccio: Memory phenomena and principles, Allyn & Bacon 1994.

[48] Charles Spearman: "General Intelligence," objectively determined and measured, American J. Psychology 15 (1904) 201-293.

[49] Louis L. Thurstone: Vectors of the mind, 1935, Thurstone later redid and expanded this book as Multiple Factor Analysis 1947 (both University of Chicago Press).

[50] A.M.Turing: Computing machinery and intelligence, Mind 59 (Oct. 1950) 433-460.

[51] A.M.Turing: On Computable Numbers, With an Application to the Entscheidungsproblem, Proc. London Math. Soc. 2,42 (1936) 230-265; 43 (1937) 544-546.

[52] Patrick H. Winston: Artificial Intelligence, Addison-Wesley (now 3rd Edition) 1992.

[53] Eugene B. Zechmeister & Stanley E. Nyberg: Human memory, Brooks/Cole 1982.