A Formalization of the Turing Test[*]

Phillip G. Bradford and Michael Wollowski

(812) 855-2136

Indiana University
Department of Computer Science
215 Lindley Hall
Bloomington, Indiana 47405

{ bradford, wollowsk }@cs.indiana.edu

February 1994

Abstract

Alan Turing proposed an interactive test to replace the question "Can machines think?" This test has become known as the Turing Test and its validity for determining intelligence or thinking is still in question.

Struggling with the validity of long proofs, program correctness, computational complexity and cryptography, theoreticians developed interactive proof systems. By formalizing the Turing Test as an interactive proof system and by employing results from complexity theory, this paper investigates the power and limitations of the Turing Test. In particular, if human intelligence subsumes machine intelligence, and human intelligence is not simulatable by any bounded machine, then the Turing Test can distinguish humans and machines to within arbitrarily high probability.

This paper makes no claim about the Turing Test's sufficiency to distinguish humans and machines. Rather, through its formalization this paper gives several ramifications involving the acceptance or rejection of the Turing Test as sufficient for making any such distinction.

Keywords: Turing Test, Interactive Proofs, Complexity Theory
Running Head: A Formalization of the Turing Test

[*]The authors names are in alphabetical order. An extended abstract of this paper appeared in the Proceedings of the 5th Midwest Artificial Intelligence and Cognitive Science Conference, T. E. Ahlswede Editor, 83-87, April 1993. This is Technical Report #399, Indiana University. Hardcopy is available or a postscript file is obtainable via internet: cs.indiana.edu:/usr/ftp/pub/techreports/TR399.ps.Z.

1 Introduction

Circumventing a discussion of what it means to "think," Alan Turing proposed what has become the *Turing Test* (Turing, 1950). This test has a human, the *interrogator*, alternately converse with another human and a computer, the *agents*. Starting without knowing which agent is the human or which is the computer, the interrogator's task is to distinguish the human and the computer through conversation by symbolic interaction. To facilitate this task the interrogator chooses the topics of conversation and leads the conversation with both agents. If the interrogator cannot distinguish the human and computer, then the computer has passed the test and some might even say that it is "intelligent."

This paper does not intend to define intelligence, rather it gives a more precise definition of the Turing Test.

1.1 Motivation

The formalization of the Turing Test given in this paper is primarily of theoretical and philosophical interest. This formalization lies right at the foundations of mathematics, a place that has recently seen the convergence of proofs, computability theory, and complexity theory, see for example (Rawlins, 1992). In addition, knowing Alan Turing's theoretical work in computability, this formalization constitutes an interesting twist for the Turing Test.

This paper attempts to naturally extend the Turing Test towards making distinctions between humans and machines through their interactions. In doing so, we go from asking computability questions about human intelligence to asking complexity theory questions about symbolic interactions.

Problem solving ability is central to most assessments of intelligence. At the same time, complexity theory has given well accepted characterizations of the relative hardness of problems. This paper combines these two notions to examine the Turing Test, using recent results tying proofs, problems and symbolic interaction together.

Non-rote testing generally gives probabilistic information. During an interactive examination, weak or unusual answers can be challenged to expose inconsistencies. Generally, there is only time to ask a small fraction of all applicable and reasonable questions. So, at best, such testing gives probabilistic information.

1.2 Main Results of this Paper

Can we use the Turing Test to distinguish humans and machines? To address this question, we will examine how interactive proofs quantify what theoretical and real machines can *solve* through interaction. In order to do this we must, in some sense, characterize what humans can do through symbolic interaction. This paper does not argue that machine are or are not "intelligent," rather it gives an interpretation of both possibilities in the context of interactive proof systems.

We are assuming that human intelligence subsumes machine intelligence. That is not to say all human functionality subsumes all machine functionality—for instance machines typically have "better" arithmetic computation speed and accuracy than humans. But, we are assuming human intelligence subsumes machine intelligence in the spirit of the Turing Test. The spirit of the Turing Test dictates that machines should interact like humans to convince the interrogator they are human. That is, the machine is doing its best to simulate human intelligence.

The main results of this paper are from theorems about interactive proof systems and the complexity classes they characterize. In particular, we show that if human intelligence subsumes machine intelligence but isn't simulatable by any bounded machine, then the Turing Test can distinguish humans and machines to within arbitrarily high probability.

1.3 Previous Work

To our knowledge there have been no prior formalizations of the Turing Test. However, there is a large literature on the philosophical implications of the Turing Test, for instance see (Searle, 1990; Epstein, 1992; Johnson, Harnad and Shapiro, 1992). While in a very different context complexity theory has become prominent in the study of artificial intelligence, (Bylander, 1991).

1.4 The Structure of this Paper

This paper assumes basic familiarity with complexity theory as in (Garey and Johnson, 1979). Also, throughout this paper we offer the same caveats regarding the size of constants hidden in asymptotic notation. That is, say the best we can do to solve some problem is to use an algorithm that requires $3n^{100}$ steps for any input of size n. Although we can "solve" this problem in polynomial time using our algorithm, the problem is infeasible. Even for inputs of size five, that is $n = 5$, no modern day computer run the above algorithm to completion in the next handful of centuries. Therefore, such solutions are typically not considered in complexity theory (Garey and Johnson, 1979). We assume this situation holds for problems in other complexity classes as well. That is, take a problem that is "technically in" some complexity class, but is very costly relative to other problems in this class—in this case we may choose not to consider such a problem.

Section 2 briefly describes interactive proof systems and relevant complexity theory, characterizing problems that may distinguish humans and machines through their symbolic and interactive solution. Section 3 gives a model of the Turing Test using interactive proof systems and discusses the formal systems this paper uses. Finally, section 4 presents some ramifications of this paper's model of the Turing Test.

2 Interactive Proof Systems

This section discusses some of the recent results tying together the notions of computation, proof and interaction.

Interactive proofs are *probabilistic* proofs of validity that are done by repeatedly and interactively checking consistency (Goldwasser, Micali, and Rackoff 1989). An interactive proof system, if used properly, allows an exponential increase in the probability that a statment is valid in a polynomial number of interactions. In the rest of this paper all polynomials are in the size of the problem instance or theorem statement given, see (Garey and Johnson, 1979; Goldwasser, Micali, and Rackoff 1989) for more information. An interactive proof consists of two machines: $\mathcal{I}$, the interrogator and $\mathcal{A}$, the agent. Machine $\mathcal{A}$ makes a claim and attempts to convince $\mathcal{I}$ of the validity of the claim with exponentially increasing probability. For example, a theorem π has an interactive (probabilistic) proof if $\mathcal{A}$ can convince $\mathcal{I}$ that π is valid with probability $1 - \frac{1}{2^n}$ in n interactions. $\mathcal{I}$ is *convinced* that π is valid if $\mathcal{I}$ could not find any inconsistencies in $\mathcal{A}$'s claim through their interactions.

A problem π is in the class $\mathcal{IP}$ if, *there exists* an interactive proof system that proves π is valid with exponentially increasing probability in a polynomial number of interactions. Intuitively, if $\mathcal{I}$ is an arbitrarily bounded machine and $\mathcal{A}$ is a polynomially bounded machine that form an interactive proof system, then in a polynomial number of interactions, the probability of the validity of a claim approaches a limit of 1 exponentially fast.

Formally, an interactive proof system (Goldwasser, Micali, and Rackoff 1989; Babai, 1988) is a pair of computing machines $(\mathcal{I}, \mathcal{A})$ where $\mathcal{I}$ is an arbitrarily bounded machine and $\mathcal{A}$ is polynomially bounded. Each of these machines has a *one-way* tape containing an infinite random string of binary bits. $\mathcal{I}$'s one-way random tape cannot be accessed by $\mathcal{A}$ and once $\mathcal{I}$ reads a bit from this tape it can never reverse the read-write head and re-read it. The symmetric case holds for $\mathcal{A}$'s one-way random tape. These "random tapes" represent unbiased coin tosses. In addition, each machine $\mathcal{I}$ and $\mathcal{A}$ has one (infinite) work tape and there are two *communication* tapes connecting $\mathcal{I}$ and $\mathcal{A}$. One communication tape goes from $\mathcal{I}$ to $\mathcal{A}$ and the other goes from $\mathcal{A}$ to $\mathcal{I}$. These communication tapes are for the interactions.

Next is an example of an interactive proof system for the Graph_Isomorphism problem based on (Goldreich, Micali and Wigderson, 1986). Here the agent, $\mathcal{A}$, claims to have an algorithm for determining whether or not any two graphs are isomorphic. The interrogator, $\mathcal{I}$, finds two graphs and gives a copy of them to the agent, for example consider the two non-isomorphic graphs in figure 1.

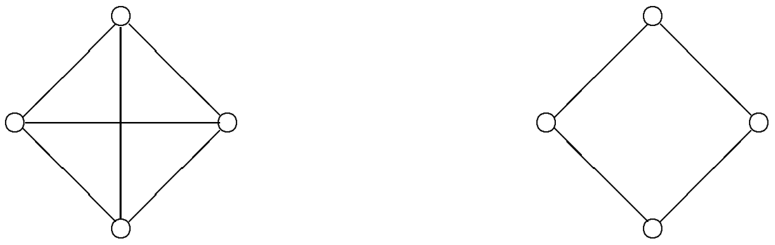

Figure 1: Two Non-Isomorphic Graphs

Let this interactive proof have n rounds of interactions. In each round, the interrogator randomly selects one of two graphs and secretly permutes its vertices and edges to make the graph unrecognizable to $\mathcal{A}$ unless the $\mathcal{A}$ can solve the Graph_Isomorphism problem. Next $\mathcal{I}$ sends the permuted graph to $\mathcal{A}$. $\mathcal{A}$'s job is to determine which of the two graphs it just received. Using a graph isomorphism algorithm, $\mathcal{A}$ can determine whether or not the permuted graph is isomorphic to either the first or to the second graph. $\mathcal{A}$ could do this, for instance, if it had a polynomial time algorithm that solves the graph isomorphism problem. It can be argued that we cannot guess with probability of greater than 1/2 the correct answer without a graph isomorphism algorithm. Further, there are problems that have been shown to have no good approximations without certain complexity classes collapsing. Such problems are ideal for interactive proofs.

$\mathcal{A}$ communicates its result back to $\mathcal{I}$. If $\mathcal{A}$ does not have an algorithm that solves the Graph_Isomorphism problem, then assume $\mathcal{A}$ may guess the correct solution with probability 1/2. If $\mathcal{A}$ is wrong, then $\mathcal{I}$ stops the interactive proof having determined that $\mathcal{A}$ does not have an algorithm for determining graph isomorphism. However, if $\mathcal{A}$ is correct through n such interactions, then $\mathcal{I}$

determines, with a probability of $1 - \frac{1}{2^n}$, that the prover has a Graph_Isomorphism algorithm.

An arbitrarily bounded machine (denoted ∞-CM) may simply be thought to have "an edge" over other bounded machines. We may view it as being able to generate extremely good heuristics, or as knowing the solution to some famous problem (such as $\mathcal{P} \overset{?}{=} \mathcal{NP}$). In fact, it has been shown that many challenging problems have no "good" heuristics (Arora and Safra, 1992). In any event, it does all of its "arbitrarily bounded computations" in polynomial time for convenience. One of the first questions interactive proof systems addressed is: What classes of problems can an arbitrarily bounded machine and a polynomially bounded machine interactively solve?

2.1 Interaction

What can two agents exchange in some number of purely symbolic interactions? To attempt to answer this, we first agree that the exchanges between these agents must alternate and consist of symbols from some predefined alphabet. Also, a bound on the number of exchanges should be predetermined somehow.

Definition 1 *Let $\mathcal{HC}$ be all interpretations of all strings representing two way human interactions using a finite alphabet.*

In interaction there may be many possible interpretations for any string. It is feasible that human interpretations of the strings can be generated by non-formal means. But these *symbolic interactions* are based on some formal system, because, the interaction uses only concrete formal symbols in some number of exchanges. Therefore, for any interaction there is a finite number of strings representing all possible interactions. Whatever interpretations of these strings one chooses, collectively they form a (large) formal system. Furthermore, if one string does not make sense or is unclear it may be possible to clarify it through more interaction.

This paper allows the possibility that in a task like interpreting poetry there may be no formal system *in* the human brain doing the interpretation. But during a Turing Test for the symbolically written interpretation to be *verified* by others there must be some formal system to communicate that interpretation. For instance, verifying analogies is done by illustrating commonalities or similarities. Also, stating an interpretation of the meaning of a line of poetry is done with symbolic references. Naturally, there may be "grounding problems" to discuss here, but we go to say that it is possible that the grounding of certain symbols and references can be done through other interactive proofs.

The interrogator and the agents may not even be aware of any interpretation of the Turing Test as an interactive proof. Hence, we don't claim there is any relation in the building of formal proofs and "intelligence." But, the probabilistic nature of interactive proofs seem to model casual human interaction quite well. For instance, in oral exams, debates, conversations, and discussions people use the consistency of their statements try to establish validity of their position. Such human interactions can be modeled by randomly generated questions in a particular domain.

2.2 Complexity and Interactive Proofs

Many problems have *certificates* for verifying their solution. A classical mathematical proof of an open problem is a certificate that a particular problem has been "solved." Certificates act as records showing that the claimed result has been achieved.

Certificates can also document that an algorithm has solved a particular instance of a problem. It is taken for granted in complexity theory (Garey and Johnson, 1979) that given an algorithm to solve a problem, this algorithm is known to correctly solve the problem at hand. That is, we have a proof of correctness for this algorithm. For example, someone may want to prove that an algorithm has found a shortest path in some given graph. Given the graph and an algorithm that can (provably) solve the shortest path problem in polynomial time, a certificate can be created by both of the steps in Figure 2.

- provide a proof of correctness for the algorithm
- provide a trace of the algorithm running over the given data

Figure 2: Acceptable Steps for Building Verification Certificates

Since the algorithm runs in polynomial time we can verify that the algorithm can produce the given trace in polynomial time. In addition, the proof of the algorithm's correctness will be of constant size relative to the variable size of the input. Since the proof of correctness of the algorithm does not vary with the input size of the algorithm. Therefore we can verify, in polynomial time, whether or not this algorithm provided a shortest path for a given input.

The class of computable problems $\mathcal{P}$ is all decision problems π such that there exists a polynomial time computing machine that given an instance π of polynomial size (in the number of symbols input) produces the correct answer in polynomial time in the size of π. Therefore, if we know that a problem π is in the class $\mathcal{P}$, then *there exists* a polynomial time bounded computing machine that solves π. In addition, given the appropriate algorithm we can create a certificate of polynomial size for the solution of π. Generally problems in the class $\mathcal{P}$ are considered to be "practical" for solution by computer (Garey and Johnson, 1979). Therefore, this paper takes all computers as capable of only solving problems in $\mathcal{P}$.

The class of computable problems $\mathcal{NP}$ is all decision problems π for which there exists a non-deterministic polynomially bounded computing machine that solves a given instance of π of polynomial size. Perhaps the greatest open question in theoretical computer science is "does $\mathcal{NP}$ equal $\mathcal{P}$?"

$\mathcal{P} \subseteq \mathcal{NP}$ because a non-deterministic computing machine can solve any problem in $\mathcal{P}$ in a polynomial number of steps. All problems in $\mathcal{NP}$ have certificates of polynomial size which also can be generated as in figure 2. In some sense $\mathcal{NP}$ contains all theorems with proofs of polynomial size in the number of symbols of their theorem's statements, given a specific type of formal system (Hartmanis, et al., 1990). Following (Hartmanis, et al., 1990) we say that a theorem is in $\mathcal{NP}$ if it has a certificate or proof of polynomial size and a non-deterministic machine can generate the certificate or proof in polynomial time. From here on let the terms problem and theorem be synonymous, in addition to solution and certificate.

In particular, to prove that a computing machine has solved a problem π in $\mathcal{NP}$, the machine can provide a certificate of polynomial size. If a problem π is in the class $\mathcal{NP}$, then *there exists* a non-deterministic computing machine that can solve π in polynomial time. There is a great variety of popular problems in $\mathcal{NP}$, see for example (Garey and Johnson, 1979; Johnson, 1990).

If a problem π is in $\mathcal{P}$SPACE, then there exists a deterministic or non-deterministic computing machine that can solve π in polynomial space. This does *not* imply that π is solvable in polynomial time.

In the worst case, some problems in $\mathcal{P}$SPACE, unlike those in $\mathcal{NP}$, seem to require certificates of exponential size. Worst case exponential size certificates are generated by a polynomial space bounded machine since the hardest problems in $\mathcal{P}$SPACE seem to take exponential time. Therefore, in the exponential number of steps an algorithm takes to solve such a problem only polynomial space is used at any instance, but the total space used during the entire computation is exponential. This means there are theorems in $\mathcal{P}$SPACE that we can state with n symbols whose proofs seem to require exponential size in n. We can *generate* these proofs using only polynomial space. Some of these proofs cannot be verified by a human or machine, since just reading the symbols in such a proof would take millenia. We say that a theorem is in $\mathcal{P}$SPACE if its certificate or proof is of exponential size or smaller and its certificate or proof can be generated by a polynomially space bounded computing machine.

$\mathcal{NEXP}$TIME is the class of problems solvable in exponential time using a non-deterministic computing machine. This is the class of problems solvable by a non-deterministic exponentially time bounded computing machine.

We say that a theorem is in $\mathcal{NEXP}$TIME if its certificate or proof is of exponential size or smaller and its certificate or proof can be generated by a non-deterministic computing machine in exponential time. Some of the theorems or problems in $\mathcal{NEXP}$TIME require proofs of exponential size relative to their theorem statements. Examples of problems in $\mathcal{NEXP}$TIME can be found in (Johnson, 1990).

The class $\mathcal{P}$SPACE *seems* to contain problems that require exponentially long proofs in the size of a theorem's statement in a given formal system. On the other hand, the class $\mathcal{NEXP}$TIME contains problems that *definitely* require exponentially long proofs in the size of a theorem's statement in a given formal system. Just verifying a proof of exponential size is very costly. This is where recent results about interactive proof systems come in. Such probabilistic proof checkers are useful for cases where the entire proof can never be written down. With this in mind, we call a probabilistic certificate where a polynomial number of interactions verify a statement to an exponentially high probability a *diploma*.

Class of Problems	*Best* Known *Certificates in the* Worst Case
$\mathcal{P}$	polynomial space certificate
$\mathcal{NP}$	polynomial space certificate
$\mathcal{P}$SPACE	likely exponential space certificate
$\mathcal{NEXP}$TIME	definite exponential space certificate
$\mathcal{IP}$	polynomial space diploma
$\mathcal{MIP}$	multi-prover polynomial space diploma

Figure 3: Certificates and Diplomas

Now some results of complexity theory are given.

Many people consider it likely that $\mathcal{P} \neq \mathcal{NP}$ and it is conjectured that $\mathcal{NP} \neq \mathcal{P}\textsc{Space}$. So we have the open question,

$$\mathcal{NP} \stackrel{?}{=} \mathcal{P}\textsc{Space}.$$

Since all problems in $\mathcal{NP}$ are solvable with a polynomial amount of space the inclusion $\mathcal{NP} \subseteq \mathcal{P}\textsc{Space}$ is straightforward, but the question

$$\mathcal{P}\textsc{Space} \stackrel{?}{=} \mathcal{NEXP}\textsc{Time}$$

persists. On the other hand it is known that

$$\mathcal{NP} \neq \mathcal{NEXP}\textsc{Time}.$$

Recently, it has been shown that all problems in $\mathcal{NP}$ are interactively provable using a diploma of $O(\lg n)$ bits (Arora and Safra, 1992). An instance of a problem is verifiable in $\mathcal{IP}$ iff it is computable in $\mathcal{P}\textsc{Space}$.

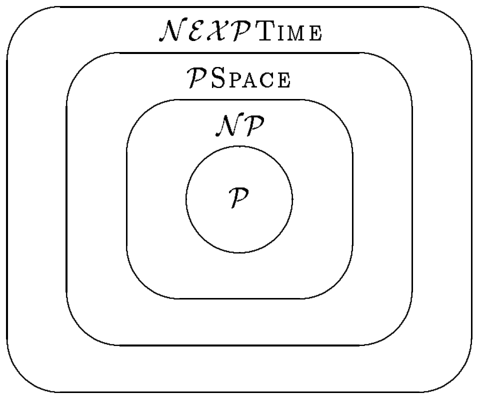

Figure 4: Possible Complexity Hierarchy

Theorem 1 (Shamir, 1992) $\mathcal{IP} = \mathcal{P}\textsc{Space}$

The class of problems that can be interactively proved using two provers and one verifier is $\mathcal{MIP}$, where the two provers are arbitrarily bounded machines (∞-CMs). Technically, these two provers don't communicate with each other, since the verifier "plays them off each other."

For example, given two graphs G and H, say one prover claims to have a polynomial time algorithm for producing up to a polynomial number subgraphs of G isomorphic to H. The judge could interactively verify these subgraph isomorphisms independently with the other prover using interactive proofs, provided this other prover could solve the graph isomorphism problem.

Theorem 2 (Babai, et al. 1990) $\mathcal{MIP} = \mathcal{NEXP}\text{TIME}$

In summary,

$$\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{P}\text{SPACE} = \mathcal{IP} \subseteq \mathcal{NEXP}\text{TIME} = \mathcal{MIP}.$$

In particular note that it is currently possible that $\mathcal{P}\text{SPACE} = \mathcal{NEXP}\text{TIME}$, although $\mathcal{NP} \neq \mathcal{NEXP}\text{TIME}$. Figures 3 and 4 summarize some of the discussion so far.

A nice characteristic of these complexity classes is reducibility. In particular, if we can solve a problem that is among the "hardest" in $\mathcal{NP}$, then we can solve all problems in $\mathcal{NP}$ using roughly the same work. This means if an agent can solve a problem that is among the hardest in $\mathcal{NP}$, then it may be best for the interrogator to try problems that are among the hardest in $\mathcal{P}\text{SPACE}$, etc.

Although random numbers are used in interactive proofs, we do not analyze any random complexity classes. The complexity results outlined in this section are all *worst case* results. In addition, the interrogator acts as a *worst case adversary* against the agents.

3 The Turing Test as an Interactive Proof System

This section models the Turing Test as an interactive proof system.

According to Babai, the class $\mathcal{IP}$ models "teacher-student" interactions (Babai, 1988). For instance, when an apparently omnipotent instructor relates a theorem to disbelieving and possibly perplexed students, the students ask questions about it in an apparently probabilistic manner and each of their questions builds their confidence in the validity of the Theorem. We argue that this teacher-student interaction is not limited to math classes, since in general teachers build the confidence of their students through their "consistency." In the same way, we take the Turing Test to be an interactive proof system. Now we formalize the Turing Test as an interactive proof system.

A Turing Test extends over a limited period of time and in some instances its duration is set beforehand (Epstein, 1992; Turing, 1950). There are interactive proof systems that use only a constant number of interactions (Cai, Condon and Lipton, 1991) to prove problems in $\mathcal{P}\text{SPACE}$. But, it's conceivable to let a Turing Test go until the interrogator makes a judgment or gives up. In addition, the case in which an interrogator has a set of prepared questions is subsumed by the fact that the questions are asked at random.

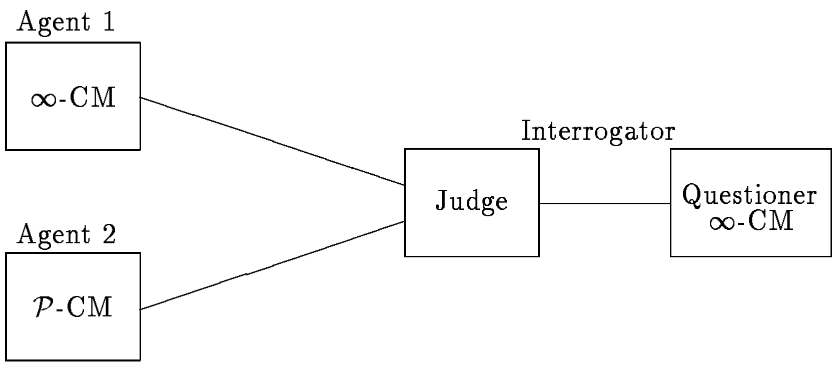

Figure 5: The Turing Test as an Interactive Proof System

How can we formalize the fact that a Turing Test attempts to distinguish humans and machines? As we have seen, interactive proofs assess the complexity of problems that can be *interactively* and *symbolically* solved. A Turing Test is not formalized as a theorem of whether a machine could pass a Turing Test, but it is formalized in terms of figure 5.

First, call the proofs that a formalized Turing Test performs *exams*. Exams may consist of solving games, puzzles, interpreting poetry, recall of historical facts, etc. Of course, this assumes all of these domains have some formal system that characterizes them so they can be written down and exchanged. Therefore, such problems can be verified with either a certificate or a diploma.

After the interactive session, the interrogator must judge which agent is a computer and which is human. This means the interrogator has two functions, that of interrogating and that of judging. This is just like the interrogator in a Turing test. Since the class of problems in $\mathcal{MIP}$ requires the two ∞-CMs (provers) not to be directly connected, we separate the function of *judge* and *questioner*. Furthermore, the judge is between the questioner and agents, and the judge is a polynomial bounded machine. The judge builds a certificate or diploma from the interactions that take place. Once the exam is over, the judge has the diplomas or certificates made during the exam. Such a diploma or certificate is the only determining factor for deciding which agent is human.

The questioner serves as the prover and the agents assume the role of verifiers. This reversal of roles is justified next, see figure 5.

The interrogator asks the agents questions in such a way as to make them build a diploma verifying a theorem *known to the interrogator*. For example, take an interactive proof system of the graph isomorphism problem. The judge might ask the agents to repeatedly and randomly try to "outsmart" the questioner by playing a game illustrating that the questioner can solve the graph isomorphism problem. If the questioner and the agents are computationally equivalent to polynomially bounded computers, then the judge will not be able to distinguish the two. Of course, the domain of such an exam does not have to be in mathematics, since problems from many fields have been shown to be among the hardest problems in the complexity classes of figure 4. Again, just because such problems are among the hardest problems in a certain complexity class does not imply that they are the hardest in other ways.

Now, say a judge first tries to interactively generate a diploma for a problem in $\mathcal{P}$SPACE with both agents. Assume the questioner subsumes the machine intelligence of an ∞-CM in the spirit of the Turing Test. In this case, take the human to simply be an ∞-CM although it is feasible that a machine that consults oracles could prove things outside of $\mathcal{P}$SPACE in a polynomial number of interactions. But, the interrogator *wants* to distinguish the computer and human, therefore assume the interrogator will only act as an ∞-CM. This allows the judge to play the human agent and the questioner off each other to prove theorems to the that are in $\mathcal{MIP}$. At the same time, the questioner and the computer can at most (probabilistically) prove theorems in $\mathcal{IP}$. Therefore, in the situation just given, the Turing Test can distinguish humans and machines if $\mathcal{IP} \neq \mathcal{MIP}$, that is $\mathcal{P}$SPACE $\neq \mathcal{NEXP}$TIME.

Of course, the human agent may not be aware of the interrogator's intentions or knowledge of this formalization of the Turing Test. In this case, if human intelligence cannot be simulated by an arbitrarily machine—that is infinite machines or "stronger notions" are required for human intelligence, then such an interactive proof may solve problems that an arbitrarily bounded computing machine can't solve. Hence, the human-human interaction may solve problems outside of $\mathcal{MIP}$. Further the human-machine interaction may solve problems outside of $\mathcal{IP}$. In this case it is not clear that we can distinguish humans and machines with this formalization.

Let a polynomially bounded judge attempt to interactively generate a diploma for a problem in $\mathcal{NEXP}$Time with both agents, see figure 5. Assuming the interrogator and *one* of the agents has intelligence that subsumes that of an ∞-CM and since one of the agents is a polynomially bounded machine it is very unlikely that it will be able to help generate a diploma for the problem in $\mathcal{NEXP}$Time exposing it as the computer! In fact, using an interactive proof system one can make it exponentially unlikely that a machine could pass as a human in this case. After the exam is over the judge will have a diploma to document which agent passed the Turing Test.

4 The Power of the Turing Test

This section gives some ramifications of taking the Turing Test to be an interactive proof system.

Suppose, human intelligence subsumes machine intelligence that can be generated by a nondeterministic polynomially bounded computer, then an interrogator could choose theorems or problems among the hardest in $\mathcal{NP}$ but not likely to be in $\mathcal{P}$. Next, the interrogator has both agents interactively prove such a theorem or solve such a problem (Arora and Safra, 1992; Brassard and Crepeau, 1986). In particular, there are problems in $\mathcal{NP}$, which we think are not in $\mathcal{P}$ that are unlikely to have "good approximations"; these problems are ideal for such interactive proofs (Arora and Safra, 1992; Garey and Johnson, 1979; Johnson, 1990). In fact, we may just hand the agents "hard" problems to solve and wait for the agents to give solutions. The agent that solves the problem "wins" the contest in two interactions. For example, different game puzzles or mystery stories could be such "hard" problems.

We want to insure the interaction is not trivial, otherwise we may lose the flavor of the Turing Test. In addition, the Turing Test as an interactive proof system becomes most interesting when applied to a variety of problems, and especially ones that have exponential certificates. If $\mathcal{P} = \mathcal{NP}$, then we can choose problems in $\mathcal{P}$Space for a two interaction Turing Test. (A two interaction Turing Test would be one where only two interactions are allowed between the interrogator and each agent.) It is also possible that $\mathcal{NP} = \mathcal{P}$Space, but in this case since $\mathcal{NP} \neq \mathcal{NEXP}$Time it must be that $\mathcal{NEXP}$Time $\neq \mathcal{P}$Space making the $\mathcal{IP}$ versus $\mathcal{MIP}$ Turing Test most desirable. In particular, problems in $\mathcal{P}$Space would then have polynomially sized certificates, and we know some problems in $\mathcal{NEXP}$Time have exponential sized certificates. Therefore, interactively separating the agents by $\mathcal{IP}$ and $\mathcal{MIP}$ as in figure 6 would be ideal. If $\mathcal{NEXP}$Time $= \mathcal{P}$Space, then the $\mathcal{IP}$ versus $\mathcal{MIP}$ Turing Test may be of no use. But, since $\mathcal{NEXP}$Time $\neq \mathcal{NP}$ we could interactively distinguish the human and machine using problems in $\mathcal{NEXP}$Time.

Polynomial bounded machines can't (non-probabilistically) prove the hardest theorems in $\mathcal{NEXP}$Time, therefore, they can't interactively prove the hardest theorems in $\mathcal{NEXP}$Time. In addition, we know that $\mathcal{P} \neq \mathcal{NEXP}$Time, so the hardest problems in $\mathcal{NEXP}$Time are of interest. The hardest problems in $\mathcal{NEXP}$Time *require* exponential certificates in the worst case. This means for large enough instances of such problems the entire certificate may never be interactively exchanged in a Turing Test. So we would have to resort to generating diplomas for solving such problems during the Turing Test. Therefore, if humans could interactively solve problems in $\mathcal{NEXP}$Time, then the Turing Test would distinguish humans and computers.
The results in this section assume the setup in figure 6.

Definition 2 *Let $\mathcal{PHC}$ be the problems interactively solvable by humans using $\mathcal{HC}$.*

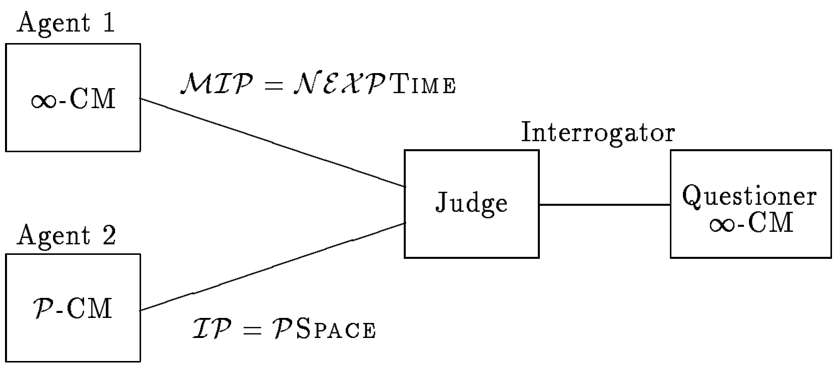

Figure 6: An Interactive Proof System

$\mathcal{PHC}$ contains problems that are solvable by symbolic interactions given that the interpretations at each end of the interactions are done by humans. The human interpretations of these symbolic interactions may be driven by non-formal systems.

The only problems that $\mathcal{PHC}$ must contain for the results in this paper to hold are those with interactive proofs.

Theorem 3 *If the formalized Turing Test is not sufficient to distinguish humans and computers, then $\mathcal{PHC} = \mathcal{P}$Space or $\mathcal{NEXP}$Time $= \mathcal{P}$Space or both.*

A proof is elementary and comes from the two cases:

Case i: In the first case say $\mathcal{P}$Space $\neq \mathcal{NEXP}$Time, then since $\mathcal{MIP} = \mathcal{NEXP}$Time, $\mathcal{PHC}$ must be $\mathcal{IP}$.

Case ii: On the other hand, if $\mathcal{P}$Space $= \mathcal{NEXP}$Time, then we cannot distinguish $\mathcal{IP}$ and $\mathcal{MIP}$. Therefore this also gives $\mathcal{PHC} = \mathcal{P}$Space $= \mathcal{NEXP}$Time.

From this come the following corollaries, that are about two interactive ∞-CMs: the questioner and one of the agents. These corollaries hold for machines stronger than arbitrarily bounded machines provided the stronger machines can simulate the arbitrarily bounded machines in a reasonable fashion.

Corollary 1 *If we assume that the formalized Turing Test is sufficient for distinguishing humans and computers and all problems solvable in $\mathcal{PHC}$ are interactively solvable by two ∞-CMs, then $\mathcal{P}$Space $\neq \mathcal{NEXP}$Time.*

A proof of this corollary follows from the fact that the Turing Test must be able to separate the problems solvable by $\mathcal{PHC}$ from $\mathcal{IP}$. Since this corollary assumes all problems solvable by $\mathcal{PHC}$ can be solved by two interacting ∞-CMs, it must be that $\mathcal{MIP} = \mathcal{PHC}$. This means $\mathcal{IP} \neq \mathcal{MIP}$, hence $\mathcal{P}$Space $\neq \mathcal{NEXP}$time.

Corollary 2 *Assuming the formalized Turing Test is not sufficient for distinguishing humans and computers and all problems in $\mathcal{PHC}$ can be solved interactively by two ∞-CMs, then $\mathcal{P}$Space $= \mathcal{NEXP}$Time.*

Assuming that computers can converse like intelligent human beings means that symbolic interactive human behavior can be characterized by problems in $\mathcal{P}$.

Alternatively, rejecting the claim that intelligent behavior can be generated by computers implies problems encapsulating "intelligent behavior" are not in $\mathcal{P}$. This endorses the Turing Test as a valid means for distinguishing humans from computers. This does not imply that proving theorems are among the hardest problems in the class $\mathcal{PHC}$.

Assuming the Turing Test is a valid means of distinguishing humans and machines implies that human and machine intelligence are comparable at least in terms of the complexity and interactive proof theory. The Turing Test essentially boils down to the question of how complex it is to computationally attain or simulate intelligent behavior. If simulating intelligent behavior turns out to be more than a polynomially bound machine can handle, then humans and computers can be distinguished using the Turing Test.

5 Conclusions and Further Directions

Recently, symbolic interaction has been formalized in a way that allows the Turing Test to be modeled as an interactive proof system. By formalizing the Turing Test in such a way, then by employing results from complexity theory, we investigate the power and limitations of the Turing Test. This paper makes no claim about the Turing Test's sufficiency to distinguish humans and machines and assumes human intelligence subsumes machine intelligence in the spirit of the Turing Test. In particular, this paper makes several strong statements about assumptions of the sufficiency or insufficiency of the Turing Test for distinguishing humans and machines. In addition, this formalization seems to lend some formal credence to the Turing Test.

We believe it would be interesting to quantify the complexity of interaction in additional classes of problems to shed more light on the Turing Test.

In addition, more should be said about the consistency of humans in terms of the Turing Test (Epstein, 1992). And for this reason "typing foibles" have been built into programs that take the Turing Test (Epstein, 1992). In fact, according to Epstein, such foibles may have helped a program win a recent variant of the Turing Test. This leads immediately to variations of interactive proof systems.

6 Acknowledgments

We would like to thank the following people for fruitful comments on this paper: Angie Allen, Kenneth Chiu, Michael Chui, Cornelia Davis, Ruth Eberle, Andy Hanson, Sushil Louis, John MacCuish, Jennifer McDaniel, Larry Moss, Gregory J. E. Rawlins, Gregory E. Shannon, Peter Shirley, Dirk Van Gucht, Perry Wagle and Beata Winnicka. In addition we would like to thank the Mind AI and Cognition group at Indiana University as well as the participants of the Mid-West Artificial Intelligence and Cognitive Science Conference for enlightening discussions.

References

Arora, A. and S. Safra, (1992) "Probabilistic Checking of Proofs; A New Characterization of NP," Proceedings of the 33rd Annual IEEE Symposium on the Foundations of Computer Science, 2-13.

Babai, L. (1988) "Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes," *J. of Computer and System Sciences*, Vol. 36, 254-276.

Babai, L., L. Fortnow, and C. Lund, (1990) "Non-Deterministic Exponential Time has Two-Prover Interactive Proofs," Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science, 16-25.

Brassard, G. and C. Crepeau, (1986) "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond," Proceedings of the 27th Annual IEEE Symposium on the Foundations of Computer Science, 188-195.

Bylander, T. (1991) "Tractability and Artificial Intelligence," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 3, 171-178.

Cai, J., A. Condon and R. Lipton, (1994) "PSPACE is Provable by Two Provers in One Round," Journal of Computer and Systems Sciences, Vol. 48, 183-193, 1994.

Dreyfus, H. (1992) *What Computers* Still *Can't Do*, MIT Press.

Editorial and Commentary, by Johnson, W. L., S. Harnad, and S. C. Shapiro (1992) in *SIGART Bulletin*, Vol. 3 No. 4, 7-11.

Epstein, R. (1992) "Can Machines Think?," *AI Magazine*, Vol. 13, No. 2, 80-95.

Garey, M. R. and Johnson, D. S. (1979) *Computers and Intractability*, W. H. Freeman.

Goldreich O., S. Micali and A. Wigderson: "Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design," Proceedings of the 27th IEEE Foundations of Computer Science Conference, 174-187, 1986.

Goldwasser, S., S. Micali, and C. Rackoff, (1989) "The Knowledge Complexity of Interactive Proof Systems," *SIAM J. on Computing*, Vol. 18, No. 1, 186-208.

Hartmanis, J., R. Chang, D. Ranjan, and P. Rohatgi, (1990) "On IP = PSPACE and Theorems with Narrow Proofs," in The Structural Complexity Column, *EACTS Bulletin*, No. 41, 166-174.

Johnson, D. S. (1990) "A Catalog of Complexity Classes," Chapter 2 in *Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity*, V. Van Leeuwen—editor, Elsevier, 67-161.

Rawlins, Gregory J. E. (1992) *Compared To What ?* Computer Science Press/W. H. Freeman.

Searle, J. R. (1980) "Minds, Brains and Programs," *Behavorial and Brain Sciences*, Vol. 3, No. 3, 417-457.

Searle, J. R. (January 1990) "Is the Brain's Mind a Computer Program?" *Scientific American*, Vol. 262, No. 1, 26-31.

Shamir, A. (1992) "IP = PSPACE," *Journal of the ACM*, Vol. 39, No. 4, 869-877.

Turing, A. M. (1950) "Computing Machinery and Intelligence," *Mind*, Vol. 59, No. 236, 433-460.