Turing Machines and Semantic Symbol Processing
Why Real Computers Don't Mind Chinese Emperors

Richard Yee

Debate over the computer metaphor of mind appears endless. At issue is the prospect of a computer ever having what could properly be considered a mind. Many proponents of the view that computers could have minds are persuaded in large part by the power of information processing. The essence of mind seems to lie in the processing of various forms of information including goals, perceptions, plans, and actions. Moreover, the **theory of computation** provides powerful support for the view that any sufficiently complete scientific theory of the mind, e.g., a theory relating neurological processes to psychological ones, would be **computable**. This essentially reflects the view that the **Church-Turing thesis** covers mind-brain processes. If human minds were understood in sufficient detail, then it would certainly be possible, in principle at least, to build computers that have real minds.

In contrast, many critics view computers as mere mindless automatons. In the rote execution of a program, how could a computer ever come in contact with any intrinsic meaning in its actions? How could a computer's symbols ever represent anything **to the computer**? Because the ability to process symbols semantically is a key mental trait, the **formal symbol processing** that a computer performs could never be sufficient to endow it with a mind.

The resulting debate between proponents and critics often revolves around the possibility of a computer's having any of a number of key mental qualities including consciousness, understanding, semantics, intentionality, qualia, creativity, and insight. Most such phenomena currently have no completely satisfying or agreed-upon characterizations (cf. Sloman, 1985). However this lack does not inhibit the formulation of arguments and refutations that reflect personal intuitions rooted in diverse backgrounds and biases. It is therefore inevitable that divergent and strongly held views should arise over many central questions, and equally inevitable that recurring attempts should be

made to convince the other side of the compelling force of one's own intuitions. Few are ever swayed, of course, and the debate rages on.

Disagreement over poorly defined mental phenomena is one thing, but too often what divides proponents and critics are their differing intuitions about computers, programs, and computation. For example, consider a computer running a program, which produces some computation of interest. In analyzing this phenomenon, do the key philosophical questions center on the program or on the computer? Which entity has primary responsibility for the computation? If it should happen that proponents focus on programs while critics focus on computers, then the ensuing debate might be vigorous indeed, being fueled by arguments that are largely at cross-purposes. Such a situation, however, should not be tolerated for long because, unlike mental phenomena, computers, programs, and computation should have reasonably precise definitions.

Unfortunately, in most cases the pivotal concept: **computer** is **not** precise. The common meaning of the term denotes programmable machines such as personal computers, workstations, and mainframes. Such machines are physical instances of **universal Turing machines** (UTM's). Often, however, the term is used to indicate any **Turing machine** (TM), not just universal ones. Although in many contexts the distinction between UTM's and non-universal TM's is not stressed, in the debate over the "computer metaphor" of mind, the distinction is crucial.

Countless critiques of the idea that mind is computable stem from the view that "computers" are only formal symbol processors. While such a view might hold for UTM's, it does not hold for all TM's. Shifting the debate to a more rigorous basis—i.e., focusing upon Turing machines proper rather than upon "computers" or "formal systems"—has devastating consequences for two of the most widely known critiques of the computer metaphor: Searle's **Chinese room** argument (Searle, 1980) and the family of arguments based upon Gödel's **Incompleteness theorems** (e.g., Nagel & Newman, 1958; Lucas, 1961; Rucker, 1982; Penrose 1989; Tymoczko, 1990). The Chinese room argument only attacks UTM's, which constitute an exceptional **subclass** of all TM's. The

"Gödelian" arguments attack **formal systems**, which correspond to **static** TM's, again, a peculiar restriction of the full TM model.

One thus finds that the failure to address Turing machines directly has bred volumes of unproductive debate. Both critics and proponents have engaged in disputes over differing "machine intuitions." Critics have formulated attacks upon TM-surrogates: UTM's and formal systems, while proponents have often responded by defending TM's in essence, but they have done so only implicitly—leaving the critics with their surrogate machine intuitions intact. A more profitable course in examining issues of mind and computation would be for both sides to address explicitly only full-fledged Turing machines, the **real** computers.

I
Minds and Turing Machines

To analyze arguments in the debate, it is first necessary to define clearly the questions at issue. Unfortunately, even this step entails some confusion. Probably the best-known position in support of the computer metaphor of mind is the one dubbed by Searle as **strong AI** (Searle, 1980). Searle describes this position as holding that a suitably programmed computer must have a mind—in the same sense in which humans have minds—if the computer exhibits the right inputs and outputs (Searle, 1980, 1987, 1990, 1992). This characterization is problematic, but the fault is not necessarily Searle's, at least not entirely (see Searle, 1992, Chapter 9, p. 200, and Footnote 2). Often the AI community itself has been too imprecise in articulating its own claims.

One problem with this common view of strong AI is that it conflates two independent propositions. The core assertion of strong AI is that mind is computable, i.e., a mind could result from the actions of a physically instantiated Turing machine. A second proposition holds that the **Turing test** (Turing, 1950) is an adequate means for awarding the label **has-a-mind** to a computing machine. In other words, this second proposition holds that probing a system

exclusively through its "linguistic" input-output behavior can provide sufficient information to justify the attribution of true mental processes to the system (e.g., Dennett, 1990).

Clearly, these two propositions—corresponding to Turing's most celebrated contributions—are independent. A position on one does not entail a position on the other. In particular, it would be possible to argue against the adequacy of the Turing test while maintaining the view that mind is computable by Turing machines. Although many supporters of strong AI also support the Turing test, it would be somewhat of a peripheral issue if TM's could never have minds in the first place. Hence, it is important keep these two propositions separate.

This article is concerned solely with arguments impacting the first proposition, namely, that some type of Turing machine could truly possess a mind. This is will be stated as follows:

Mind is a computation producible by a Turing machine. (P1)

Proposition P1 expresses the essence of the computer metaphor of mind. If P1 is true, then there is a non-empty class of Turing machines that, when physically instantiated, would have (or could develop) minds in the same sense in which humans have (or develop) minds. This is exactly analogous to saying that **addition** could result from a real TM. Hence, P1's claim is that, like addition, mental phenomena are computations that some subclass of TM's could perform.

P1 does not entail any position, pro or con, regarding the Turing test. Even more importantly, it does not refer to computers, programs, formal systems, or universal Turing machines. P1 does not refer to **computers** because the term connotes two related but significantly different concepts: Turing machines and universal Turing machines. P1 does not refer to **programs** because programs are merely descriptions of TM's. It does not refer to **formal systems** because, unlike TM's, formal systems do not provide for receiving inputs from external sources. Finally, P1 does not single out **UTM's** because,

contrary to their name, “universal” machines constitute only a subclass of all TM's—those that are universally programmable.[1]

Given P1, attention may turn to possible arguments against it. The following sections examine two such arguments: Searle's Chinese room and arguments from Gödel's Incompleteness theorems. When viewed in light of P1 and the theory of computation, the flaws of each argument become apparent. Although the theory of computation exposes the flaws, it does not necessarily address the original intuitions of critics who would likely persist in their contention that computation is inevitably devoid of semantic understanding. Therefore, Section III presents a basic account of why TM's can process symbols **non-formally**. Section IV concludes with a view toward further investigation of semantic symbol processing in Turing machines.

II
Attacks on Turing machines?

Since there are two prominent arguments that purportedly refute strong AI, it is natural to examine how each pertains to P1. A straightforward analysis will show that the Chinese room (CR) argument (Searle, 1980) has no direct bearing on P1 because the CR fails to address the entire class of TM's. The second argument is actually a family of arguments centered around Gödel's Incompleteness theorems (e.g., Lucas, 1961; Penrose, 1989). Although somewhat more complicated to analyze, it will be seen that Gödelian arguments are similarly flawed in that they attack only static formal systems, which are an inadequate substitute for dynamic TM's. Both arguments thus fail due to erroneous identifications between TM's and “equivalent constructs” that turn out to be **non-equivalent** for the purposes of answering philosophical questions about minds and computation.

---

[1] Moreover, **strict** UTM's are uninteresting from the point of view of mental processing for a number of reasons. In particular, they are among those TM's that cannot re-program themselves, i.e., they cannot learn.

2.1: The Chinese Room Versus Programmed Computers: Searle claims that the Chinese room (CR) argument refutes strong AI. Does it refute P1? Briefly, the CR argument is as follows. Inside a room is a human **U** (Searle), a program **P**, and Chinese input symbols $\mathbf{c_{in}}$ which are composed into strings. Figure 1(a) depicts this situation. **U** can understand no Chinese and is only able to process the inputs $\mathbf{c_{in}}$ according to program **P**. We may assume that the

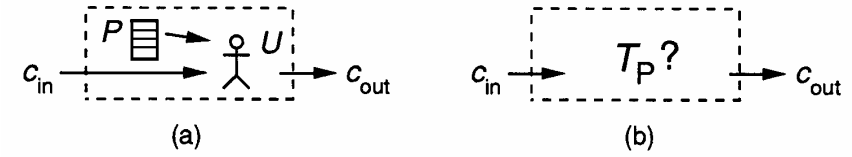

Figure 1: The Chinese room is a UTM. **U** does not understand Chinese, but what about $\mathbf{T_P}$?

CR can convincingly pass a Turing test, and then ask whether it follows that the Chinese symbols are being understood inside the room. At no time can **U** understand the meanings of the symbols $\mathbf{c_{in}}$. Furthermore, argues Searle, there is nothing else about the room which could understand them. The only other entity, program **P**, could be eliminated by having **U** "memorize" it. Doing so would still not enable **U** to understand the symbols $\mathbf{c_{in}}$. Thus, the meanings of the symbols are never understood inside the room.

One conclusion drawn from this argument is that the Turing test is inadequate as a test of understanding: the appearance of understanding need not imply its existence. As noted, however, this issue is not of current concern. A second conclusion is that no "programmed computer" could understand its input symbols because computers always do exactly what **U** does in the room: follow rules for syntactically manipulating symbols without connecting them to any semantic content. Because the semantic processing of symbols is central to mental functioning, this establishes that computers could not have minds.

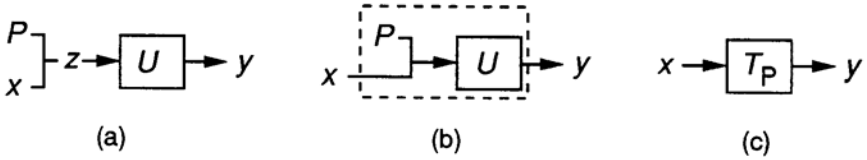

Figure 2: Two types of Turing machines. Part (a) shows a UTM **U** with input **z** composed of a program **P** and a nominal input **x**. Part (c) is an instance of the TM described by **P**. Part (b) is a view of **U** as the machine $\mathbf{T_P}$

TM's, UTM's and Programs: To understand this argument's relationship to P1, it might be useful to review some aspects of so-called "universal" computations. Figure 2 illustrates the relationships between UTM's, programs, and TM's. Fig. 2(a) shows that the total input to every UTM **U** consists of two parts: a program **P** and a nominal input **x**. **P** is a description of some Turing machine $\mathbf{T_P}$ (shown in Fig. 2(c)). **U** interprets program **P** as rules for processing the nominal input **x**. The output of this process, **y**, is the same as the output of $\mathbf{T_P}$ running directly on **x**. Figure 2(b) illustrates the typical operation of a programmed computer in which the program portion of **U'**s input is held constant while the nominal inputs are varied. In this manner **U** is able to simulate the overall input-output behavior of $\mathbf{T_P}$.

It is important to keep in mind the exact sense in which **U** is "universal." **U**, like every other TM, computes one specific function. In general, then, **U** and $\mathbf{T_P}$ are two different TM's that compute two different functions. **U** maps input $\mathbf{z} = \langle \mathbf{P}, \mathbf{x} \rangle$ to the output **y**, while $\mathbf{T_P}$ maps **x** to **y**. Only by restricting attention to **U**'s nominal input **x** (as in Fig. 2(b)) does one observe the reproduction of $\mathbf{T_P}$'s input-output behavior. Internally, **U** operates in a very specific fashion which centers on properly interpreting program **P**, a portion of input **z**. Nominal input **x** is processed only via the instantiation of $\mathbf{T_P}$, not by **U** directly. Hence, to generate $\mathbf{T_P}$'s output on input **x**, it is sufficient for **U** to process **x** purely formally. A UTM, therefore, is simply a TM that treats a portion of its input as a body of rules for formally manipulating the remaining portion of its input.

Consider, for example, the differences between a simple calculator that can perform only addition and a general purpose computer running a program that describes the calculator. There is no software-hardware distinction for the calculator. The calculator itself embodies an algorithm for performing addition. One could never remove the calculator's "program" without losing the whole machine. The computer, on the other hand, embodies a universal algorithm, which is nothing more than an algorithm for interpreting and executing rules. It is clearly not an algorithm for addition. To the computer, a calculator program is just an input, which could be completely replaced without affecting the computer itself. To find out how it would "actually feel" to add numbers, one should consult the calculator because the computer cannot add—it can only instantiate other TM's.

When a computer runs a program, there are at least two identifiable computations, the universal one and that of the TM described by the program, both of which are simultaneously implemented on a single hardware system. It is sometimes said that the program's TM is a "virtual machine," but the truth is that both computations have precisely the same ontological status. Each corresponds to a particular mathematical, or logical, description of the physical system. Thus, neither computation could be considered more or less "real" than the other.[2]

To investigate computational processes properly, therefore, it is advisable to attend only to the processes themselves, ignoring UTM-computers altogether. Unless strictly-universal computations happen to be the objects of interest, UTM's need never enter the picture. They are nothing more than middlemen whose own computations divert attention from the computations that they instantiate. UTM's are mathematical curiosities and engineering

---

[2] This is the formal justification for one-half of the **Systems Reply** to the Chinese room argument. That is, the theory of computation establishes that there are two computations in the CR. Hence, the **possibility exists** that the second computation, the one described by program **P**, is actually understanding the Chinese input symbols. To conclude the second half, that the Chinese is **definitely** being understood, one would either need to believe in the Turing test, or need to know whether the program's TM does what Chinese **brains** do (N.B., not minds).

conveniences, but when it comes to questions about minds and computation they are philosophical quicksand.

The Chinese Room Revisited: The Chinese room argument asserts that executing a program is not sufficient to produce a mind (e.g., Searle, 1992, p. 200). Suppose this conclusion were correct. Would P1 have been scathed? The programmed entity **U** in the CR is exactly a "computer," i.e., a UTM. Thus, at best, the CR only establishes that UTM's can never understand the meanings of their (nominal) input symbols. On the other hand, P1 asserts only that **some** TM's may have such mental qualities. Thus, even if one granted the argument's claims with respect to UTM-computers, there would remain plenty of non-universal TM's for which P1 might still hold.

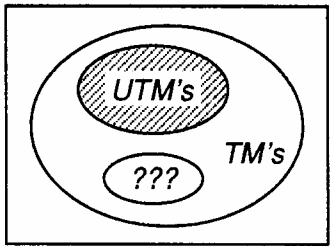

Figure 3: UTM's are a proper subset of all TM's. Even if mind were not a universal computation, it does not follow that it is not in some other Turing-computable class.

Figure 3 illustrates this point, which can be formalized as follows. Consider the following assertion:

Property P is true of every computer. (2)

This statement is ambiguous, and should be replaced by one of the following:

P is true of every Turing machine. (3a)
P is true of every universal Turing machine. (3b)

Statement (3a) is much stronger than (3b). Because UTM's are a proper subset of the set of all TM's, (3a) logically implies (3b), but not conversely. In general, it would be fallacious to form a conclusion like (3a) from arguments that only establish its (3b) equivalent. Although the CR argument only addresses UTM's running programs, it seems to have been widely interpreted as a demonstration that **no TM** could understand the meaning of its input symbols. Clearly, however, such a deduction would be erroneous. P1 is thus immune from any straightforward application of the Chinese room argument.

What does this analysis say about questions of **syntax-versus-semantics**? The preceding comments **do not** dispute Searle's most basic claim—shared by Harnad (Harnad, 1990)—which is that formal manipulation of symbols is significantly different from the semantic understanding of symbols required for minds. Analysis of the CR argument simply shows that it does not address the symbol processing abilities of all (indeed, of most) TM's because the CR examines the internal processing **only** of programmable UTM's. On this point, the universality of UTM's is immaterial because it does not extend below the input-output level, and the argument assumes that performance there looks like the true understanding of Chinese. To establish that all TM's are as numb to certain input symbols as **U**, one would need an argument that specifically addresses the internal processing of all TM's. However, Section III, below, presents arguments to the contrary, showing that TM's are capable of processing symbols in a non-formal fashion.

The Mind-Brain Analogy: Searle points out that the strong AI position is often characterized by drawing an analogy between minds and brains on the one hand, and programs and computers, or software and hardware, on the other, i.e., **mind:brain::program:computer**, or **mind:brain::software:hardware** (Searle, 1980, 1987, 1992). If this analogy is

indeed pervasive, it would not be surprising. Computers and programs are two of the most conspicuous icons of our time. The Chinese room argument is intended to refute this analogy, but such a refutation is unnecessary because the analogy itself is not apt.

Translated into more precise terms, the analogy reads that mind is to brain as the description of a TM is to a UTM, i.e., **mind:brain:: TM-description:UTM**. This borders on the nonsensical. Even if one could make sense of it, there is little point in trying. Turing machines, not simply UTM's, are the objects of interest. The proper analogy to consider, therefore, is the one derived from P1:

mind:brain::computation:Turing-machine.

Could a Turing machine have a mind in the same way that brains have minds?

2.2: Gödelian Truth Versus Formal Systems: The view that Turing machines are merely formal processors, blind to the meanings of their symbols, also underlies critiques based on Gödel's Incompleteness theorems (e.g., Nagel & Newman, 1958; Lucas, 1961; Rucker, 1982; Penrose, 1989; Tymoczko, 1990). All such arguments are based on Gödel's proof that **formal systems** have limitations, which in a certain manner, translate into limitations on Turing machines. Gödel proved that any formal system **F**, sufficient for Peano arithmetic, has a well-formed statement **G(F)** that is clearly seen to be true but is not derivable using only **F**'s axioms and transformation rules. This fact is used as the basis for arguing that certain human powers of thought cannot be reproduced by any formal system or any equivalent TM.

The essence of this argument is a “proof” that runs generally as follows. Suppose that human thought **H** were producible by some formal system **F**, i.e., **H = F**. Since **F** must be sufficient for Peano arithmetic, there must be a statement **G(F)** which is true but which **F** cannot prove via its axioms and derivation rules. Nevertheless, through “mathematical insight” humans **H**

can see that **G(F)** is true. Thus, **H** can prove something unprovable by **F**, showing that **H≠F**: human thought is not producible by any formal system.

The key to this argument is the idea that, thanks to Gödel, humans could always **outmaneuver** any formal system **F**. This is done by forming **F**'s Gödel statement **G(F)** and recognizing its truth. Note that knowing of the existence of some Gödel statement **G(F)** is not the same as knowing that a particular statement **w** is in fact **F**'s Gödel statement, i.e., that **w = G(F).** Gödelian arguments assert that humans are actually capable of establishing the truth of particular statements **G(F)** for systems **F** that are alleged to be equivalent to human thought.

Many point out that such an assumption seems unfounded: if human thought were in fact producible by a formal system, it would undoubtedly be enormously complex, and there is no principled reason to believe that humans could produce its Gödel statement (e.g., Hofstadter, 1979; Rucker, 1982). Others point to related problems regarding the **consistency** of formal systems, which is necessary for applying Gödel's results (e.g., Putnam, 1960; Bowie, 1982; Chalmers, 1990; Davis, 1990; Mortensen, 1990). Although such refutations might have merit, they have not won over many Gödelian critics (e.g., Lucas, 1961; Penrose, 1990; Tymoczko, 1990). Perhaps these refutations do not directly confront the central Gödelian intuition which is something akin to the sense that "I can **easily see** a truth that **provably cannot be derived** by any formal system." A more direct response to this intuition comes in two parts. The first points out that humans are not really so clever. The second points out that TM's could also outmaneuver formal systems, if they were given a fair chance.

Naked Formal Systems: What enables humans to produce Gödel statements for formal systems? Gödel has shown us the key: it is simply necessary to construct a particular self-referential statement about a given system. Surely, humans could reproduce Gödel's "trick" for any given formal system, especially if it were not extremely complex. Let us test this assertion on a very simple

formal system **A**: Peano arithmetic. Of course, it would be too easy to derive **G(A)** if **A** were specified using the usual alphabet of mathematical symbols such as '×' for multiplication or '∃' for 'there exists'. Instead, let us encode the symbols for **A** using arbitrarily chosen Chinese characters: for example, we could replace the symbol '∃' with the Chinese symbol for 'horse'. Call this Chinese encoding of Peano arithmetic $\mathbf{A_C}$.

Now imagine being given $\mathbf{A_C}$ with knowledge of neither the encoding nor even the fact that $\mathbf{A_C}$ represents Peano arithmetic. How could one formulate $\mathbf{G(A_C)}$ and be convinced of its truth? Worse still, suppose one were given an encoded form of a system, **F**, more complicated than $\mathbf{A_C}$? It is absurd to think that anyone could be certain of both forming and recognizing the truth of the Gödel statements for formal systems under such circumstances. Although some might feel that the intended interpretations of the symbols should be provided with the systems, such information is **external** to any formal system. It is information about the relationship between the given formal system and another system—ultimately the "system" of human experience.

The point of this example is simply that humans could only outmaneuver a formal system via Gödel's technique if they were provided with meanings for the symbols of the formal system. Only in such a case could anyone construct meaningful statements (let alone **true** ones) without relying on the system's axioms and transformation rules. This fact is so obvious, it is easily missed. It is easy to assume that **being given a formal system F** further entails being given an interpretation, **Interp(F)**, that assigns meanings to **F**'s symbols.

Semantic Clothing: Given a semantic interpretation for the symbols of a system **F**, it does indeed seem possible to derive a Gödel statement **G(F)**, at least in principle. But if humans require **Interp(F)** in order to form the Gödel statement, it would be unfair to withhold this information from **F**, the system humans claim to be outmaneuvering. **F** should also be given **Interp(F)**. But now this highlights a crucial difference between formal systems and TM's.

Formal systems **do not accept inputs** from external sources, whereas TM's clearly do.

To illustrate the crucial role of inputs, consider the following fallacious argument. I claim that my image can never be captured in any mere photograph. Suppose that, to prove me wrong, you were to take a picture of me. To convince me that you have indeed captured my true image you must give me the photograph. I will then point out that this photograph fails to reflect the obvious fact that I am holding a photograph. Clearly, I can outmaneuver **any** alleged photograph of me because no photograph can portray its own image. I therefore conclude that although photographs might reflect certain likenesses, they can never fully capture my true image. I can always identify a fatal discrepancy in any photograph **that is given to me** (cf. Lucas, 1961; Hofstadter, 1979).

The fact that a person could derive a Gödel statement for system **F**, given an interpretation of its symbols, **Interp(F)**, does not prove that **F** was not an accurate account of the person at the time before **Interp(F)** was provided. In other words, if person **H** were characterized by formal system **F** at time **t**, i.e. if **H = F**, then providing **H** with **Interp(F)** at time **t+1**, would thereby change **H** into $\mathbf{F^+ = F + Interp(F)}$. $\mathbf{F^+}$ has new axioms that may entail **G(F)**, so it would **not** contradict Gödel's theorems for $\mathbf{F^+}$ to derive **G(F)**. Furthermore, the derivation of **G(F)** would in no way imply that $\mathbf{H \neq F}$ at time **t**. There would only be a contradiction if **H** could prove **G(F) without being given Interp(F).**

Gödelian arguments now face the following dilemma. Either (a) humans must be capable of establishing the truth of Gödel statements for formal systems whose symbols are utterly meaningless (to the persons in question), or (b) it must be proved that no formal system $\mathbf{F^+}$ can derive **G(F)**, where $\mathbf{F^+ = F + Interp(F)}$ is a formal system **F** augmented with a suitable interpretation of its own symbols. The first case is impossible because the "self-evident truth" of any Gödel statement requires knowing the meanings of the symbols in question. Establishing the second case, which is not covered by Gödel's theorems, would require a new proof. However, there is good reason to doubt that such a proof exists because Gödel has **shown us how** to construct **G(F)** from a suitable

interpretation of system **F**'s symbols. It is far from obvious that Gödel's process of using **Interp(F)** could not be automated—at least to the same extent that humans are capable of performing it.

As Penrose points out (Penrose, 1989, p. 111-112), there is nothing particularly sacred about the truth of the Gödel statement **G(F)**. **F**'s axioms are also "obviously true" but not derivable within **F**. Hence, there is no more magic required in seeing the truth of **G(F)** than there is in seeing the truth of **F**'s axioms. To be validated, both the axioms and **G(F)** must be interpreted rather than formally derived. In both cases the interpretation process begins by associating meanings with **F**'s symbols. These meanings are essentially correspondences between the symbols and entities in some other system, e.g., a person's knowledge and beliefs about the world. The truths of the axioms and Gödel statement are then ascertained by determining "truth" in the other system.

Learning Machines: The fallacy of Gödelian arguments lies in equating dynamic Turing machines with "static snapshots," i.e., formal systems. Turing machines can interact with external sources of information and, through such interactions, change themselves. Formal systems, on the other hand, are inputless engines for generating a body of theorems. In short, Turing machines can **learn**, whereas formal systems cannot (cf. Arbib, 1987, Section 8.5).

Although most computer programs (Turing machines) that people encounter retain their functionality from one use to the next, making them equivalent to fixed formal systems, not all TM's need be so "faithful." Given inputs, TM's may literally re-program themselves, becoming different machines with new behaviors and abilities. In particular, it is entirely possible that a TM, given an interpretation of its own program, could produce the Gödel statement for that program. There would be no violation of Gödel's theorems because the input interpretation would have first **changed the machine**—just as it would for humans.

III
Formal Symbol Processing Versus Turing Machines

Are the flaws in the Chinese room and Gödelian arguments mere mathematical technicalities? Fundamentally, both arguments are motivated by the same intuition: automatic computation is an inherently meaningless activity, whereas the very essence of mind lies in subjective awareness and the certainty that thought processes are intrinsically meaningful. The CR and Gödelian arguments are simply vehicles that attempt to establish concretely this key difference between computational and mental symbol processing. An essential question, therefore, is whether every Turing machine necessarily processes information in a fashion that is inherently meaningless **to the machine**.

To some it may seem obvious that all TM's are simply formal symbol processors. If this is so, then it should be easy to prove. The first requirement is to define the difference between **formal** and **non-formal** types of symbol processing. The phrases **symbol processing** or **symbol manipulation** are sometimes used as if the processing of symbols were **necessarily** formal, but this is obviously not true. For example, in the CR if **U** were literate in Chinese, then the processing of the input symbol for 'horse', say, would not necessarily be formal. The reason is that, in interpreting the symbol, it would become possible to access **U**'s personal information about horses, information not contained in the symbol itself. If **U** believed that horses are bigger than bread-boxes, for example, this information could play a role in determining **U**'s output.

For the processing of input symbol α_i to be strictly formal, outputs may depend only on objective information intrinsic to α_i itself. This is usually considered information describing α_i's shape (literally its **form**) because that information allows the symbol to be distinguished from other potential symbols. The processing of α_i would be non-formal if (a) the processor were to associate the symbol with subjective information, not intrinsic to α_i's shape, and (b) such information were allowed to influence the course of subsequent processing and, ultimately, the outputs.

Turing Machines and their Transition Functions: To determine whether all Turing machines are restricted solely to formal symbol processing, one must consider the details of how a TM processes its input symbols. A Turing machine **T** is a mathematical object denoted

$$T = (S, \Gamma, \delta), \text{ where}$$

$S = \{ \mathbf{s_0}, \ldots, \mathbf{s_N} \}$ is a set of internal **control states**, which includes the **start state** $\mathbf{s_0}$ but not the **halt state H**

$\Gamma = \{ \boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_M}, \boldsymbol{\gamma_1}, \ldots, \boldsymbol{\gamma_P}\}$ is a set of **tape symbols** containing the subset

$\Sigma = \boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_M}$, the set of **input** tape symbols;

$\delta : S \times \Gamma \rightarrow (S \cup \{H\}) \times \Gamma \times \{$ **Left, Right**$\}$ is the **transition function**.

T has an infinite tape each of whose cells may contain a single symbol from $\boldsymbol{\Gamma}$, and it has a read-write tape head which accesses exactly one cell at a time and which can be moved one cell to the left or right on each application of the δ transition function. The δ function is the heart of a TM. For example, when machine **T** is in control state $\mathbf{s_p}$ and is reading symbol $\boldsymbol{\alpha_i}$, $\boldsymbol{\delta(s_p, \alpha_i) = (s_q, \gamma_j,}$ **Left)** indicates that **T** writes $\boldsymbol{\gamma_j}$ on the tape, moves the tape head one square to the left, and changes to control state $\mathbf{s_q}$. This is how a TM processes symbols.

Is this strictly a formal process? To be so, one must guarantee that the determination of $\boldsymbol{(s_q, \gamma_j,}$ **Left)** from $\boldsymbol{(s_p, \alpha_i)}$ entailed no association of information with $\boldsymbol{\alpha_i}$ that is not intrinsic to the symbol itself. But this is impossible to guarantee. It is entirely possible that in the δ transition, the processing of symbol $\boldsymbol{\alpha_i}$ involved an association with stored information, which in turn influenced the decision to produce the resulting machine configuration $\boldsymbol{(s_q, \gamma_j,}$ **Left).**

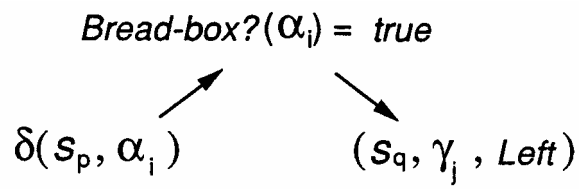

Figure 4: A non-formal δ transition on symbol α_i. The new machine configuration depends on information about α_i that is not an intrinsic property of the symbol.

Figure 4 illustrates how such a non-formal δ transformation might occur. During the transition process the symbol α_i is interpreted yielding, in particular, the fact that α_i is bigger than a bread-box. This information in turn influences the ultimate transition to the new machine configuration. Thus, the transition was not solely dependent on formal properties of the symbol α_i. It also depended on information internal to the machine and on the ability to associate α_i with portions of that information.

Simulated Turing Machines: Weak Equivalence: Why are UTM's formal symbol processors while other types of TM's need not be? A UTM is able to process input symbols on a purely formal basis because it is given the **δ-table** of the TM that it is suppose to simulate. That is, when a UTM, **U**, simulates another TM, **T**, **U** is given a table describing the complete input-output behavior of **T**'s δ function. **U** then performs a **weakly equivalent** (Pylyshyn, 1984) simulation of **T**'s δ function. Weakly equivalent computations need only maintain input-output or black-box equivalence: given the same inputs yield the same outputs. Stronger equivalence entails not only input-output equivalence, but also some degree of equivalence in the manner in which outputs are produced from inputs. Thus, strong equivalence requires some degree of algorithmic equivalence.[3]

[3]Pylyshyn (1984) points out that the notion of strong equivalence has many levels depending upon the level of abstraction at which one describes an algorithm.

For **U** to perform a strongly equivalent simulation of **T**, **U** would not only have to recreate **T**'s δ function's input-output behavior; it would have to do so "in the same way" as **T**. Suppose that the creation of **T**'s δ-table entailed the interpretation of some input symbol. This interpretive process would be completely hidden from **U** because only the result would appear in **T**'s δ-table. As far as **U** is concerned a δ-table is a completely opaque input-output representation of another TM's δ computation.

For a UTM, however, such strong equivalence is never necessary. **U** need only mimic **T** only insofar as necessary to recreate **T**'s global input-output behavior. Because **U** has **T**'s δ-table, it is never necessary for **U** to also recreate the process by which the contents of the table were determined. It is sufficient for **U** to look up the answer—a completely formal process. Thus, a weakly equivalent simulation of **T**'s δ function will always be sufficient, and, therefore, **U** may always treat the input symbols as formal objects only.

On the other hand, when a non-universal machine for **T** is constructed, there is no δ-table. The δ transitions are transitions between physical states of the machine in accordance with the logical states of **T**'s algorithm, to which they correspond. It is impossible to guarantee that, during these transitions, an input signal does not become associated with stored information about its meaning. Such an association is entirely possible. This analysis strongly suggests that P1 is immune, not only to the Chinese room and Gödelian arguments, but to any argument founded on the claim that TM's are incapable of non-formal symbol processing.

Conclusions

The failure of the Chinese room and Gödelian arguments to refute P1 does not, of course, establish its truth. However, most criticisms of the computability of mind are reactions against the process of "merely following an algorithm," i.e., against the rote manipulation of symbols according to a fixed set of instructions. This view of computation is strongly reinforced by UTM-computers and formal systems, both of which faithfully interpret rules and apply them to uninterpreted data. The sole purpose of such activity is to produce a genuine instance of some other TM. Therefore, such computations cannot exhibit key mental traits such as the pursuit of internally determined goals or adaptation through learning from input-output experiences. Understandably, this is a prevalent view of computation, but it is misplaced. It is accurate only with respect to UTM's and other non-adaptive TM's.

Full-fledged Turing machines are not so limited. The true computer metaphor of mind is the **Turing machine metaphor of brain**. The correct proposition to consider is whether human mental processes are within **some** class of Turing machine computations, not simply the universal ones, or ones otherwise prohibited from input-generated learning. General types of TM's are immune from the arguments put forth by critics whose intuitions are based upon surrogate TM's. In particular, it is not true that TM's are incapable of processing symbols in a subjective and dynamically evolving manner.

A Turing machine is governed by its internal programming which may change over time. A TM can be endowed with specific goals. It can receive inputs from an environment, and its outputs can affect its environment. A TM can remember input-output experiences, and it can form generalizations. A TM can associate inputs with current memories and generalizations, enabling it to produce novel outputs. It can develop predictive or causal models of phenomena in its environment, including models of itself. In short, a TM can learn to control its environment in an effort to satisfy its internal goals.

There are many reasons for believing that the control processes of biological machines—including the minds of human brains—are instances of Turing machine computations. These views stem, not from a failure to appreciate the power and intricacy of human thought, but from an appreciation of the power and intricacy of Turing machine computations. Attacking or defending the semantic powers of UTM's or formal systems is a waste of time. The semantic powers of Turing machines are what matter. A more productive exchange of views, therefore, might focus on developing correct intuitions regarding Turing machines, as well as determining what it would take for their computations to be considered truly meaningful.[4]

Department Of Computer Science
University of Massachusetts at Amherst
Amherst, Massachusetts

[4] Special thanks go to Sharad Saxena for many useful and interesting discussions of these issues. I also thank Neil Stillings, Andrew Barto, David Banach, Kevin Staley, John Moore, and Lance Williams for comments on drafts of this work.

References

Arbib, M. A. (1987). ***Brains, Machines, and Mathematics*** (second ed.). New York: Springer-Verlag. Previous edition: 1964 by McGraw-Hill, Inc.

Bowie, G. L. (1982). "Lucas' number is finally up." ***Journal of Philosophical Logic***, 11, 279-285. Slightly revised version reprinted in Jay L. Garfield (Ed.), ***Foundations of Cognitive Science: The essential readings***, New York: Paragon House, 1990.

Chalmers, D. J. (1990). "Computing the thinkable." ***Behavioral and Brain Sciences***, 13(4), 658-9.

Davis, M. (1990). "Is mathematical insight algorithmic?" ***Behavioral and Brain Sciences***, 13(4), 659-60.

Dennett, D. C. (1990). "Can machines think?" In Kurzweil, R. (Ed.), ***The Age of Intelligent Machines***, Cambridge, MA: MIT Press, 48-61.

Harnad, S. (1990). "The symbol grounding problem." ***Physica D***, 42(1-3), 335-346.

Hofstadter, D. R. (1979). ***Gödel, Escher, Bach: An Eternal Golden Braid***. New York: Vintage Books.

Lucas, J. R. (1961). "Minds, machines and Gödel." ***Philosophy***, 36, 120-124. Reprinted in A. R. Anderson (Ed.), ***Minds and Machines***, Englewood Cliffs, NJ: Prentice-Hall, 1964, 43-59.

Mortensen, C. (1990). "The powers of machines and minds." ***Behavioral and Brain Sciences***, 13(4), 678-9.

Nagel, E. & Newman, J. R. (1958). ***Gödel's Proof***. New York: New York University Press.

Penrose, R. (1989). ***The Emperor's New Mind***. New York: Oxford Universe Press.

Penrose, R. (1990). "The nonalgorithmic mind." ***Behavioral and Brain Sciences***, 13(4), 692-703.

Putnam, H. (1960). "Minds and machines." In Hook, S. (Ed.), ***Dimensions of Mind: A Symposium***. New York University Press. Reprinted in A. R.

Anderson (Ed.), ***Minds and Machines***, Englewood Cliffs, NJ: Prentice Hall, 1964, 72-97.

Pylyshyn, Z. W. (1984). ***Computation and Cognition: Toward a Foundation for Cognitive Science***. Cambridge, MA: Bradford Books/MIT Press.

Rucker, R. (1982). ***Infinity and the Mind: The science and philosophy of the infinite***. Boston: Birkhauser.

Searle, J. R. (1980). "Minds, brains and programs." ***Behavioral and Brain Sciences***, 3(3), 417-457.

Searle, J. R. (1987). "Minds and brains without programs." In Blakemore C. & Greenfield, S. (Eds.), ***Mindwaves: Thoughts on Intelligence, Identity and Consciousness***, Chapter 15, 208-233. Oxford, U.K.: Basil Blackwell, Ltd.

Searle, J. R. (1990). "Is the brain's mind a computer program?" ***Scientific American***, 262(1), 26-31.

Searle, J. R. (1992). "***The Rediscovery of the Mind***." Cambridge, MA: Bradford Books/MIT Press.

Sloman, A. (1985). "What enables a machine to understand?" In Josi, A. (Ed.), ***Proceedings of the Ninth International Joint Conference on Artificial Intelligence***, 995-1001, Los Altos, CA: IJCAI, AAAI, Morgan Kaufmann.

Turing, A. M. (1950). "Computing machinery and intelligence." ***Mind***, 59 (236). Reprinted in D. R. Hofstadter and D. C. Dennett (Eds.), ***The Mind's I***, New York: Bantam Books, 1981, 53-66.

Tymoczko, T. (1990). "Why I am not a Turing machine: Gödel's theorems and the philosophy of mind." In Garfield, J. L. (Ed.), ***Foundations of Cognitive Science: The essential readings***, Chapter II-4, 170-185. New York: Paragon House.