HERBERT A. SIMON and STUART A. EISENSTADT

HUMAN AND MACHINE INTERPRETATION OF EXPRESSIONS IN FORMAL SYSTEMS

ABSTRACT. This paper uses a proof of Gödels theorem, implemented on a computer, to explore how a person or a computer can examine such a proof, understand it, and evaluate its validity. It is argued that, in order to recognize it (1) as Gödel's theorem, and (2) as a proof that there is an undecidable statement in the language of PM, a person must possess a suitable semantics. As our analysis reveals no differences between the processes required by people and machines to understand Gödel's theorem and manipulate it symbolically, an effective way to characterize this semantics is to model the human cognitive system as a Turing Machine with sensory inputs.

La logistique n'est plus stérile: elle engendre la contradicion!
– Henri Poincaré
'Les mathematiques et la logique'

1. INTRODUCTION

This paper considers whether human beings can think in ways that machines cannot. It begins by examining the semantics of human thinking. The question of what constitutes an interpretation of a system of formal expressions became the focus of our inquiry, and our exploration of the meaning of interpretation led us to experiment with computer proofs of Gödel's informal version of his undecidability theorem, or machine verifications of such proofs – one might opt for either description. For our purposes here, Gödel's theorem simply serves as an example on which to test some of our ideas. Any other theorem involving both a rich object language and a rich metalanguage would have served as well, but we happen to have worked with Gödel's.

2. A DISCOVERY IN THE DESERT

Figure 1, below, could well be a transcription of a cuneiform tablet excavated in a Middle Eastern desert. For ease of reproduction, the cuneiform

Synthese **116:** 439–461, 1998.

$\Pi\rho o\delta\upsilon\chi\tau\iota o\nu\sigma$:

(1) $\Xi(\ni\xi\ni) \rightarrow \xi$
(2) $Z \Rightarrow B(\Xi(\ni Z\ni))$
(3) $(\chi o = \xi, B(\xi)) \rightarrow B(\chi o)$
(4) $B(\Xi(\ni\ Z\ni)) \Rightarrow \Phi$
(5) $B(\Phi) \Rightarrow \Xi(\ni Z\ni)$
(6) $B(\Xi(\ni\xi\ni)), B(\Xi(\ni B(\xi)\ni)) \Rightarrow B(\Psi(\ni\xi\ni))$

$\forall\Pi\rho oo\phi\forall$:

X1 $\chi o = \Xi(\ni Z\ni)$
X1 Z
X1 $B(\Xi(\ni Z\ni))$
$B(\Xi(\ni Z\ni))$

X2 $\chi o = \Xi(\ni B(Z)\ni)$
X2 $B(Z)$
X2 $B(B(\Xi(\ni Z\ni)))$
X2 $B(\Phi)$
X2 $\Xi(\ni Z\ni)$
X2 Z
$B(\Xi(\ni B(Z)\ni))$

$B(\Psi(\ni Z\ni))$

Figure 1. Demented Gödel.

symbols have been replaced on a 1-1 basis by arbitrarily chosen Greek letters – except for the arrows and double arrows, which, because of their exceptional beauty, have been retained. We gave the upper part of the text (the part labeled '$\pi\rho o\delta\upsilon\chi\tau\iota o\nu\sigma$') to our computer, after translating it into the format required by the OPS-5 programming language (virtually a symbol-by-symbol translation). When we entered the "Start" command, the computer produced a trace which, when translated back into the language of our transcribed cuneiform original, matched identically the lower half of the text (the part labeled '$\forall\Pi\rho oo\phi\forall$'). (Similarly, when we gave a translation to *Mathematica* with instructions about the order in which to execute the productions, it produced exactly the same output.)

Evidently, the computer, in both cases, interpreted the first six lines as being rules of inference, that is, interpreted the arrows to mean that the

Productions:

(1)	$\Xi(\ni\xi\ni) \rightarrow \xi$	**Ξ elimination**
(2)	$Z \Rightarrow \neg(\Xi(\ni Z\ni))$	**Def of** Z
(3)	$(\chi o = \xi, \neg(\xi)) \rightarrow \neg(\chi)$	**¬ inser for** X_0,
(4)	$\neg(\Xi(\ni Z\ni)) \Rightarrow \Phi$	**¬ elim, replace** Z **by** Φ
(5)	$\neg(\Phi) \Rightarrow \Xi(\ni Z\ni)$	**¬ elim, replace** Φ **by** Z
(6)	$\neg(\Xi(\ni\xi\ni)), \neg(\Xi(\ni\neg(\xi)\ni)) \Rightarrow \neg(\Psi(\ni\xi\ni))$	**Def of** Ψ

"Proof":

X1	$\chi o = \Xi(\ni Z\ni)$	**Supplied by user**
X1	Z	**R1**, $\xi = Z$
X1	$\neg(\Xi(\ni Z\ni))$	**R2**
	$\neg(\Xi(\ni Z\ni))$	**R3**, $\xi = \Xi(\ni Z\ni)$
X2	$\chi o = \Xi(\ni\neg(Z)\ni)$	**Supplied by user**
X2	$\neg(Z)$	**R1**, $\xi = \neg(Z)$
X2	$\neg(\neg(\Xi(\ni Z\ni)))$	**R2**
X2	$\neg(\Phi)$	**R4**
X2	$\Xi(\ni Z\ni)$	**R5**
X2	Z	**R1**, $\xi = Z$
	$\neg(\Xi(\ni\neg(Z)\ni))$	**R3**, $\xi = \Xi(\ni\neg(Z)\ni))$
	$\neg(\Psi(\ni Z\ni))$	**R6**, $\xi = Z$

Figure 2. Mostly demented Gödel.

expression on the left side of the arrow on each line should be replaced by the expression on the right side. Single arrows were interpreted to mean that only whole expressions could be transformed; double arrows to mean that sub-expressions could be replaced within larger expressions. Most important, an examination of the proof shows that almost no other interpretation was required in order to produce the program output. (A little more is involved in Rule 3, but we will postpone discussing it for a moment.)

After recovering from our surprise, we scrutinized the text with great care and finally concluded that it is a formalized proof of some kind of theorem. We would now like to show you how we decoded the tablet and discovered what theorem it represented.

3. DECODING THE TABLET

Figure 2 shows the first step of decoding after we decided (partly by the frequency of its occurrence, and using other cues) that 'B' probably represents the logical symbol '$\neg$'. Already after this minimal change, some interesting patterns emerge. For example, if we substitute 'Z' for 'ξ' in Rule 1, then apply Rule 1 and Rule 2 successively, we get '$\Xi(\ni Z\ni) \Rightarrow \neg\Xi(\ni Z\ni)$'. Interpreting the 'not' semantically, this would seem to mean that from '$\Xi(\ni Z\ni)$' we get 'not $\Xi(\ni Z\ni)$'. Next, Rule 2 has the form of a definition of 'Z' in terms of itself: a self-referential definition. Clearly something like the Liar Paradox was known to the person who produced this tablet. Finally, application of Rule 4 and then Rule 5 gives '$\neg\neg\Xi(Z) \Rightarrow \Xi(Z)$', cancelling double negation in a way that appears a bit non-intuitionist.

Encouraged by this bit of progress, we looked for interpretations of some of the remaining symbols that would exploit the resemblance to the Liar Paradox, arriving finally at the decisions recorded in Figure 3. We tried the substitution of 'provable' for 'Ξ' and 'decidable' for 'Ψ'. Then, we adopted the following interpretation of 'Z'. Let $Z(R, S)$ be the sentence whose name is 'R', and whose meaning is 'The sentence named S is not provable'. Then $Z(R, R)$ will be 'The sentence R is not provable', i.e., 'This sentence is not provable'. Finally, for 'Φ' we substituted an expression asserting that $Z(R, R)$ is the member of a set named 'K'.

What we had now obtained was 'clearly' (clear to us!) a formal version of Gödel's informal proof of his theorem. In the first part of the proof (the part having the "C1" tags) the provability of '$Z(R, R)$' is assumed, and is shown to lead to its non-provability, thereby producing ' "$Z(R, R)$" is not provable' by application of *reductio ad absurdum*. In the second part of the proof (having the "C2" tags), the provability of 'not-$Z(R, R)$' is assumed, and is shown to lead both to 'not-$Z(R, R)$' and to '$Z(R, R)$' (second and last lines), another contradiction. Discharging the assumption, we have ' "not-$Z(R, R)$" is not provable'. Rule 6, applied to the products of the two reductios, gives us: ' "$Z(R, R)$" is not decidable'.

These substitutions can be regarded as interpretations, under the assumption that terms like 'provable' already have some meaning for the (human) interpreter. Of course the computer (as programmed in either the OPS-5 or the *Mathematica* version) is wholly indifferent to the substitution of one uninterpreted symbol for another, hence operates just as well with one of the three figures as with another. The same program, with these replacements of uninterpreted symbols, produces all three proofs interchangeably; it remains the case that the only interpretation required

Rules:

(1) **Prov(‘*x*’) → *x***

(2) ***Z*(*R*, *R*) ⇒ ¬(Prov(‘*Z*(*R*, *R*)’))**

(3) **Co = *x*, ¬(*x*) → ¬(Co)**

(4) **¬(Prov(‘*Z*(*R*, *R*)’)) ⇒ *Z*(*R*, *R*) *ε K***

(5) **¬(*Z*(*R*, *R*) *ε K*) ⇒ Prov(‘*Z*(*R*, *R*)’)**

(6) **¬(Prov(‘*x*’)), ¬(Prov(¬(‘*x*’))) ⇒ ¬(Decid(‘*x*’))**

Proof:

C1	**Co = Prov(‘*Z*(*R*, *R*)’)**	
C1	***Z*(*R*, *R*)**	**R1, *x* = *Z*(*R*, *R*)**
C1	**¬(Prov(‘*Z*(*R*, *R*)’))**	**R2**
	¬(Prov(‘*Z*(*R*, *R*)’))	**R3, *x* = Prov(‘*Z*(*R*, *R*)’)**
C2	**Co = Prov(‘¬(*Z*(*R*, *R*)’))**	
C2	**¬(*Z*(*R*, *R*))**	**R1, *x* = ¬(*Z*(*R*, *R*))**
C2	**¬(¬(Prov(‘*Z*(*R*, *R*)’)))**	**R2**
C2	**¬(*Z*(*R*, *R*) *ε K*)**	**R4**
C2	**Prov(‘*Z*(*R*, *R*)’)**	**R5**
C2	***Z*(*R*, *R*)**	**R1, *x* = *Z*(*R*, *R*)**
	¬(Prov(‘¬(*Z*(*R*, *R*))’))	**R3, *x* = ¬(*Z*(*R*, *R*))**
	¬(Decid(‘*Z*(*R*, *R*)’))	**R6, *x* = *Z*(*R*, *R*)**

Figure 3. Generalized Gödel.

is interpretation of the arrows in the rules as commands to perform the corresponding replacements of letter strings. Nor is justification provided for this interpretation: The program does not reason why; it just acts.

4. WHY THIS IS A PROOF OF GÖDEL'S THEOREM

Two issues must be resolved before we have a right to regard the tablet, in any of its versions, as giving the proof of a theorem, and specifically, a proof of Gödel's theorem. First, we must show that the rules, R1–R6, are tautological: that is, that they will make only tautological substitutions of one assertion for another. Second, we must show that the undefined terms in these rules (in particular, ‘Prov’) can be assigned interpretations

that correspond to their meanings in Gödel's theorem while preserving the tautological character of the rules.

4.1. *Tautological Character of the Rules*

A sequence of transformations of symbol strings is called a "proof" under some interpretation of the strings if the transformations are tautological under that interpretation. We then call the rules of transformation "rules of inference". Without at least some interpretation, there is no way to tell whether rules are rules of inference or not. In Figure 3, we interpret 'Prov' as 'provable', a term that already has a semantic interpretation stored in our (human) memories. A sentence, S, is provable in language L if we can produce a sequence of sentences of L, each of which is an immediate consequence of one or two previous sentences and/or axioms, such that S is the final sentence of the sequence. Given the meaning of 'consequence' and 'axiom', whether a given sentence and proof sequence satisfies this definition can be determined by examining the sentence and the sequence.

'Immediate consequence' means 'obtained by application of a (tautological) rule of inference of L', and 'axiom' means 'tautological sentence of L introduced without proof'.

How it can be determined whether a rule or sentence is tautological will presently be discussed at some length. On this interpretation, Rule 1 asserts that if 'x' is provable (in some language, L) it can be entered as a theorem of the metalanguage (call it 'M') in which the rule is written. (We do not, at least for the present, exclude M from being identical with L.) To many persons this assertion will appear, on the basis of the meaning they attach to 'provable', to be tautological (and historically, it has been so regarded by most logicians). We have not yet given any means whereby a computer could interpret it as tautological, or any explanation (other than historical) as to why people commonly find it so. We will return to this question later.

Certain rules of inference introduce new terms that they define. A definition is tautological because any sentence in a proof that contains the defined term can be replaced by a sentence in which the definiens has been substituted for that term. As '$Z(R, R)$' first appears in Rule 2, we can use that rule as a definition for it, hence tautological. Notice that, in this interpretation, $Z(R, R)$ is defined in M essentially as "any expression that satisfies Rule 2". No guarantee is provided that such an expression exists, but in any specific case, a semantic test will determine whether the name 'S', used in the expression "Sentence S is unprovable", is or is not the name of that expression in the language.

A similarly bland interpretation can be provided for Rule 4, as a definition of 'K'. That is to say, if $Z(R, R)$ is not provable in L, R is a

member of the set K, where we assume that 'set' means a collection of things that satisfy the Zermelo–Fraenkel axioms. We will have more to say presently about the interpretation of these axioms. The blandness of this rule is enhanced by the fact that R has only appeared in the system of rules as an argument of Z, and K has not appeared at all. Rule 5 assumes that a given sentence is not both a member and a non-member of set K. Rule 6 is a definition of the predicate 'Decid', which we interpret as 'decidable'.

In interpreting Rules 2, 4, 5 and 6 we treat a rule that introduces a previously unused term as a definition of that term. Then, the sentence that is produced by substituting the definiendum for the definiens is valid if and only if the original unsubstituted expression was valid: the substitution is tautological, each of these sentences describing the same set of possible worlds. If A is the definition of B, then we determine whether B is true in world W_x by determining whether A is true in W_x, so that B is true in those worlds, and only those worlds, in which A is true. It is precisely by this inverse substitution which replaces definiendum with definiens, that we are able to evaluate whether or not the definiendum is true.

Rule 3 is another matter. We see from its structure that it represents *reductio ad absurdum*. It says that if, in all worlds in which condition C holds, a pair of propositions of the form P and not $\neg P$ are both provable, then in all possible worlds, $\neg C$ holds. If we accept this rule as tautological in any language in which it is incorporated, then we can regard it as tautological in M. As in the case of Rule 1, we will return later to the justification for regarding it as tautological in general.

Accepting provisionally the assertion that all six rules are tautological in M, the sequence of replacements in the lower half of Figure 3 constitutes a proof in M. We do not have to be concerned about the status of axioms, for M has no axioms. The two first theorems are obtained by discharge of conditional statements that have been contradicted. Notice that we cannot make the same statements about Figures 1 and 2, for without an interpretation at least of 'provable', we have no way of discovering whether all the rules are tautological, hence legitimate inference rules. If we were as indifferent to replacements of symbols as our computer (as currently programmed) is, we would have no more reason to regard the display in Figure 3 as a proof of Gödel's theorem than the displays in Figures 1 and 2.

4.2. *The Generalized Gödel Theorem*

A cautious way to describe the theorem of Figure 3 would be somewhat as follows: In any language, L, in which it is possible to define a predicate, call it 'Prov' that satisfies Rule 1, a sentence, call it '$Z(R, R)$', that satisfies

Rule 2, and an argument '*R*' and a set '*K*' that satisfy Rules 4 and 5, the sentence '*Z*(*R*, *R*)' is not decidable *in L*. The proof itself is a proof in *M*, that is, a proof using the inference rules of *M*. As is well known, we can also derive '*Z*(*R*, *R*)' *as a theorem of M*. This can be done, for example, by treating Rule 2 as a definition of '*Z*(*R*, *R*)', thereby allowing it to be applied in both directions, so that from ¬Prov('*Z*(*R*, *R*)'), already proved in *M*, we get *Z*(*R*, *R*). What we have in Figure 3 is a generalization of Gödel's theorem (to any language that satisfies the rules), but also a weakening, for we have not specified the languages for which the theorem holds, nor even shown that such languages exist.

Moreover, there are other interpretations of the terms than the ones we have given above that would guarantee that the expressions of Figure 3 constitute a proof of something. The terms appearing in the rules can be given any interpretation that is compatible with the tautological character of the rules. For example, we can interpret '*X*' as 'true' or 'impredicable' instead of 'provable'. If we interpret '*X*' as the identity predicate, (*X*('*x*') = '*x*'), then Rule 1 becomes '"*x*" → *x*', and Rule 2, '*Z* ⇒ ¬ "*Z*"' so that their successive application produces '"*Z*" ⇒ ¬ "*Z*"', the contradiction known as the Paradox of the Liar. Alternatively, if we interpret '*X*' as 'impredicable', the proof becomes a proof of a form of Russell's antinomy. The affinity among these various antinomies, which is well-known, stems from the self-referential character of Rule 2. In contrast to the indifference (at least so far) of the computer to the various interpretations of terms, for humans, the "generalized Gödel theorem" becomes quite different theorems under different semantic interpretations of these terms.

4.3. *Gödel's Formal Proof*

The formal proof that Gödel actually presents in his 1931 paper includes a demonstration that the language of *Principia Mathematica* (PM) satisfies the conditions of the general theorem when '*Z*' is interpreted as 'provable in PM', hence that the predicate 'provable' is undecidable in PM. To achieve this (hard) part of the proof, Gödel actually provides the definition of this predicate, and uses his celebrated method of arithmetization to show that there is an argument, '*R*', for *Z* that satisfies the conditions of Rules 2, 4 and 5. We must now see what is involved in these additional acts of interpretation.

Before turning to this topic, however, one comment is in order. In everything we have done so far, we have casually mingled expressions defined in *L* (whatever *L* may be) with expressions defined in *M*. In a slightly more precise notation than that of the figures, we would attach to each expression

a label indicating the language to which it belongs (the label designates the expression's "ilk", as "type" and "class" are already reserved for other uses in logic). We have actually supplied such a label in the computer programs, distinguishing among L-theorems, M-theorems and C-theorems (the latter being conditional statements used in the *reductio* episodes). For example, Rule 3 takes a C-statement as hypothesis and, if a contradiction is derived, places an M-theorem, the negation of the hypothesis, in memory; e.g., the M-theorem that 'Z' is not L-provable. Our experience here perhaps gives some hint as to why humans survived for several thousands of years without type (or ilk) distinctions, although they were aware, from Greek times, of the antinomies that lurked in waiting.

5. SEMANTIC INTERPRETATION OF THE OBJECT LANGUAGE

In our urge to find an interpretation of the cuneiform tablet as a primordial Gödel's theorem, we have left our computer behind. Our whole progress depended on successive acts of human interpretation of the demented symbols of Figure 1. For example, we had to find interpretations of expressions like 'Prov("x")' and '$Z(R, R)$' as sentences in some language (and in the specific application of the undecidability theorem to PM, as sentences definable in PM). Moreover, in order to accept Rule 1, for example, as a tautology in the metalanguage, M, we had to accept *semantic* interpretations of 'Prov("x")' and 'x' as 'x is provable in PM' and ' "x" is a theorem in PM'.

This last assertion, that semantic interpretation of Rule 1 is required, is perhaps not immediately evident. In Gödel's metalinguistic proof of his theorem, the sentences of PM are simply uninterpreted strings of symbols, and the definitions of such predicates as 'proof' and 'provable formula' are expressed entirely in terms of syntactic properties of these expressions However, these rules of PM do not provide a logical justification for replacing the sentence, 'Prov("x")' with the sentence 'x', as is done by Rule 1 in the metalanguage, M. This replacement (in M) is tautological, and hence justified, only if we may assert x whenever we may assert Prov('x'), and the legitimacy of this step surely depends on the semantic meanings of these terms.

To see this, we merely need to replace 'Prov' by 'Printed-in-Green'. The rule would then say that if a sentence, 'x', is printed in green then the sentence can be asserted. That all sentences printed in green are true is hardly a tautology. Moreover, if we carried through the proof in this interpretation, we would see that the second *reductio ad absurdum* would give us ' "$\neg R$" is not printed in green'. Thus, the definition (Rule 2) of

'Z' as an expression that is not printed in green guarantees that '$\neg Z$' also not be printed in green, which is an empirical hypothesis (a false one) and not a tautology. In a moment, we will provide a semantic definition (in a metalanguage) of 'provable in PM' and 'theorem in PM' that will clarify this point further.

Up to this point, the computer has shared only one of these acts of interpretation with us: the interpretation of arrows as rules of inference to be executed when their conditions were satisfied (i.e., when the appropriate expressions were already in memory as previously proved theorems). To see how we can get the computer back into the picture, and give it genuine understanding (like the understanding we have) of what the inscriptions mean, we will first have to carry the story of human interpretation to its end.

5.1. *Human Interpretation*

In showing that there is an undecidable sentence in the language of PM, Gödel develops, as we have seen, precise definitions of 'immediate consequence', 'proof' and 'provable' in that language. Until they are interpreted in turn, these definitions are of course also simply strings of symbols that might just as well be written in cuneiform as in Roman letters and mathematical symbols. This absence of interpretation left us with the task of persuading ourselves that the rules of M are tautological, hence properly employed as inference rules, and in particular, that 'provable' thereby receives an interpretation that makes Rule 1 of the metalanguage also tautological. How does this come about?

What transpires in a human head when a visual stimulus corresponding to what we, the observers, call a "rabbit" is presented to an English-speaking person, and that person responds by saying "rabbit"? This is clearly an empirical question: How does the human cognitive system actually work? We will undertake to answer the question with a theory of human thinking that is supported by extensive empirical evidence about the mechanisms of human thought. It will be best if we consider this theory not as an introspection into our own thinking but as based on empirical observation of other people's thinking. This way of looking at it will put its verification on an even footing with the verification of any other empirical hypothesis. It will also remind us that we are not looking for an unattainable logical "proof" of initial premises, but, with Humean modesty, seeking such a degree of conviction as empirical evidence can provide.

We turn to Turing for inspiration. We bring into our minds his description of a Turing Machine, which, you will recall, Turing proposed as a

model for a *human* computor. Better for our purposes, we will consider a von Neumann computer of the familiar kind, which (as long as we can continue to order additional memory from the manufacturer) is at least equivalent to a Turing Machine. We say “at least" because the von Neumann computer, unlike the Turing Machine, has ports through which new symbols and symbol structures can be input. A rabbit (and many other stimuli) can be presented to it and sensed by it. We will be a little fanciful (since at the moment we are trying to model people, not computers) and imagine that these ports have properties similar to those of the human retina.

Stimuli strike the retina; they are recoded in terms of a set (perhaps a vector) of features and then sorted through some kind of discrimination net so that different vectors reach different terminals, where they are assigned internal names. To distinguish these names from other kinds of names that we will have occasion to introduce, we will call them “internal addresses” and we will call the process of reaching an address by discrimination “recognition”. The recognition process, which has been implemented, for example, in the EPAM system (Richman et al. 1995) and in the sensory and perceptual components of many robotic devices, looks like this:

Interpreted by

Simulus → FEATURE EXTRACTOR → *Feature Vector*

Recognized by

→ DISCRIMINATION NET → *Internal Address*

The discrimination process will generally be partial rather than exhaustive. That is, feature vectors will be sorted on only a subset of their features, so that many non-identical, but “similar” vectors may arrive at the same address (e.g., vectors that characterize all lizards). The discrimination net is not a stationary structure but grows with experience, proliferating new branches and developing new routes to existing addresses. Its learning method is simple, but to explain it we have to discuss one other feature. The net may store at an address not only descriptive information about the class of stimuli sorted to that address, but also associations with (pointers to the addresses of) other stimuli that may be experienced when a particular stimulus has been recognized. Suppose that the system has learned to recognize (imperfectly) the kinds of stimuli we call “cats”. Suppose that it has also learned to recognize the printed word ‘CAT’. Then it may associate the word with the corresponding stimuli. On recognizing the word, it may look for a cat in the environment and recognize it; or on recognizing a cat, it may be motivated to output the word.

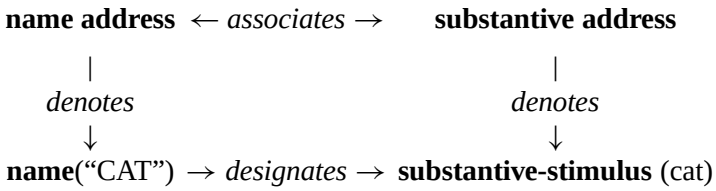
Now suppose that the word ‘LEOPARD’ appears, which is not in the word recognition part of the discrimination net. By sorting it down the net, it will be found that its vector differs from the vector stored at the terminal address (say ‘LEOTARD’). Then a new test can be introduced on the fourth component of the vector, and a new branch created, so that ‘LEOPARD’ will be sorted to a different branch from ‘LEOTARD’. Next, let the word ‘LEOPARD’ appear simultaneously with a stimulus object that is (erroneously) recognized as a cat. The system will now look for one or more differences between the vector stored at the cat address and the vector of the new stimulus, and again a new branch and address (for leopards) will be grown, which will be associated with the name ‘LEOPARD’. Now the system will not only discriminate among the words ‘LEOPARD’, ‘LEOTARD’ and ‘CAT’, but it will also discriminate between a leopard and a cat, and be able to provide each with its correct name. A number of systems having the capabilities just described have actually been programmed and tested. EPAM, one of the most extensive, simulates closely the symbolic processes that humans use to recognize stimuli and to learn to discriminate among stimuli.

A system of this kind can also acquire a natural language vocabulary, associated with its semantics, by being presented with visual scenes and simultaneously with sentences that denote these scenes. (As Siklóssy 1972, has shown with his ZBIE system, such a system can also acquire a natural language syntax.) The details of the process do not matter, and there are many differences of detail among the systems that have been built to recognize and learn to recognize. Just two basic features are important: that the system be able to form arbitrary systems of discriminations among stimulus vectors, and that it be able to associate pairs of related vectors, like the pairings of animal-stimuli with name-stimuli of the example above.

For later reference, we would like to remark that Siklóssy’s system obviously liberates us from Searle’s Chinese Room conundrum by allowing the occupants of the room to look through a window to the outside world in order to learn the denotations of the words that they are to be able to translate. Having acquired the appropriate vocabulary in this way, they can not only make a translation, but also test the truth value of a message, thereby demonstrating that they understand it. If the message delivered to the Chinese room warns that there is a leopard outside the window, the inhabitants of the room can look out the window to see if the leopard is indeed there. They may, for example, see a feline creature outside the window but recognize it as a cat rather than a leopard. We will later make use of these capabilities to explicate machine understanding.

5.2. *Semantics*

The system just described permits classes of external stimuli to be mapped onto internal addresses via the discrimination process. It also permits the same kind of internal mapping of a special class of external stimuli, linguistic stimuli, which are used to denote the other ("substantive") external stimuli. The association of the internal addresses for substantive stimuli with the internal addresses for linguistic stimuli provides the designations for the names. We will use the term 'designates' to refer to the relation between a name and the stimulus it names, and the term 'denotes' to refer to the relation between an internal address and the corresponding external stimulus.



Many details have to be worked out in this scheme (in the course of the organism's learning) before it provides an adequate model for human semantics. For example, it has to provide for optical illusions, but we will not discuss these perceptual issues here.

There is a considerable body of empirical evidence that the human capability for recognizing and associating stimuli employs a mechanism of this kind. Such a mechanism explains how a "proof", for example, may be recognized. A person examines the list of expressions he or she has written down while carrying out a proof in PM, and notices that each step involves adding an expression to the list by applying one of a small number of rules of transformation (inference rules of PM) to one or two of the expressions already in the list. A proof of '$Z(R, R)$ is provable in PM' would be just such a sequence with '$Z(R, R)$' as its terminal member. A person can use these observations to attain a semantic interpretation of Gödel's definition of 'proof in PM'.

Of course, as observers, we take the written proof as empirical evidence for (not *pace* Hume, as a logical proof of) the symbol structures and cognitive processes in the head of the person constructing the proof, inducing or abducting these structures and processes from our observations of behavior. What breaks us out of the circle of meaningless and uninterpreted inscriptions is that we can associate our observations with the mental world of symbolic processes hypothesized to be occurring in the head of the system we are observing. This mental world is induced

from these observations of behavior in the same way that hypotheses about masses and gravitational forces can be induced from observations of the movements of planets.

There is no methodological difference between the task of testing whether the behavior of a person is caused by certain symbolic or neuronal processes occurring in the brain and the task of testing whether any observable physical phenomenon is caused by a hypothesized mechanism that cannot be observed directly (e.g., the behavior of an elementary particle). In both cases, the acceptance or rejection of a theory rests ultimately on empirical evidence, not logical proof, and may prove incorrect in the face of new evidence. Errors may arise from the intervention of instruments (as well as the human senses) between phenomenon and observation, and empirical evidence can show, at most, that the mechanisms postulated by a theory are sufficient to produce a phenomenon, never that they are necessary and that there does not exist an alternative set of mechanisms that could do the job.

A proof is not simply an uninterpreted string, for (as Brouwer insisted) it is associated with a particular process of generation, and a convincing (empirical) theory of both string and process can be constructed from observations of the process. It cannot be emphasized too strongly that this is an empirical judgment (whether called an "induction", an "abduction" or a "retroduction"), not a logical deduction. As we proceed to describe the human processes associated with judgments of analyticity, it will become apparent that these processes can be carried out by appropriately programmed computers, and in particular, computers with the capabilities of the EPAM and ZBIE programs discussed briefly above.

5.3. *Analyticity*

To address the question of how a person "knows" that the rules of the metalanguage or of PM are genuine rules of inference, that is, tautological or analytic, we must elaborate somewhat our model of the mind (Eisenstadt and Simon, 1997). We postulate, again with the support of much evidence, that the system (the mind) contains a large collection of symbol structures that (1) are "indexed" by a discrimination net of the sort described above, which recognizes familiar stimuli by sorting them to their internal addresses, and (2) are operated upon by productions (if-then rules) that, upon matching symbol structures already in memory, generate new structures which they add to memory. The rules of inference we have been discussing are examples of productions, but there may also be productions in memory that are not rules of inference (not tautological). Structures in memory, whether they denote knowledge or rules, may be

typed (or "ilked") by labels of various kinds. In particular, sentences may be typed '*E*' for 'empirical', if they are the products, directly or indirectly, of sensory recognition processes, and '*L*' if they are products of logical processes without empirical contamination.

For our present purposes, we must account (empirically) for three judgments of analyticity: (1) the attribution of analyticity to *modus ponens* as a rule of inference, (2) the attribution of analyticity to *reductio ad absurdum*, and (3) the attribution of analyticity to the axioms of PM. Why do human beings accept these procedures and premises as analytic?

5.4. *Modus Ponens*

Suppose that the sentence, '*A*', is stored in the memory of a person whose logical beliefs and practices we are trying to account for, and also the rule, '$A \rightarrow B$'. Then the rule (a production) will, in fact, execute, its action leaving the symbol '*B*' in memory along with the others. We, the observers, who have an empirical theory of what is going on, interpret this event as follows:

> The rule stored in memory, '$A \rightarrow B$', means that whenever the sentence '*A*' holds, the sentence '*B*' does also. We need not ask where this rule comes from: perhaps from some prior reasoning, perhaps it was abducted from empirical evidence. If this rule holds and if '*A*' is asserted (stored in memory as valid), then '*B*' is also valid, hence may be asserted (as empirically or logically true, as the case may be).

This final assertion, of course, is an operation of the rule of *modus ponens*. The execution of the production can be interpreted as the conversion of a "potential" '*B*' into an "actual" '*B*' – "implicit belief" into a belief. The commitment to the action, whenever the condition, '*A*', is satisfied, is implicit in the presence of the production in memory. Notice that no explicit rule corresponding to *modus ponens* is stored: In the presence of '*A*' and the particular production '$A \rightarrow B$', the system simply acts as if *modus ponens* had been executed upon *A* and ($A \supset B$). Neither '*A*' nor '$A \rightarrow B$' need be tautologies; the tautology lies in the action that occurs: what was already implicit was made explicit. People's experiences of tautology or inevitability reside in the fact that the action is built into the structure of their memory systems: if such a system already believes '*A*' and '$A \rightarrow B$', it will, by virtue of the structure of the production system, come to believe '*B*'.

In a system having this production system architecture, one cannot believe, and become simultaneously aware of believing, *A* and $A \rightarrow B$

without believing B. It is empirically (not analytically) true that tautologies are compelling to human beings, and an empirically supported theory of human information processing explains the mechanism that brings this about. However, to predict what a person will believe, it is not enough to know what working memory elements and productions are stored in memory. To create new beliefs these elements and productions must be evoked and come into the focus of attention before they will be executed, and a complete cognitive theory has to include the appropriate attention-directing mechanisms, which are dependent upon goals, stimuli, and other context. Only God believes all the derivable conclusions as soon as He believes the premises.

Notice that, supplied with a mechanism for classifying statements by type or ilk, a production system can carry out conditional as well as absolute reasoning. To the question, "What if A and $A \rightarrow B$?" it can and will answer "then B"; and if $A \rightarrow B$ is itself already a belief, will give the same answer to "What if A?"

5.5. *Reductio ad Absurdum*

What about the analyticity of *reductio ad absurdum* (Rule 3)? In the production system theory of the mind, elements are placed in working memory (1) when they are adjudged to be true, that is, when they become beliefs, and (2) when they are accepted conditionally, as hypotheses in order to examine their consequences. The set of working memory elements is then used with the rules of inference to infer more facts about the world (beliefs in case (1), and consequences of the hypotheses in case (2)). If both a proposition and its negation are included among the working memory elements, choices can no longer be made by the system between propositions and their negations. The convincingness of *reductio ad absurdum* derives from the embarrassment of the contradictions it creates.

When the system encounters a contradiction (absolute or conditional), what actions can it take to recover its effectiveness? Assume we have a system that has not produced any contradictions, but has collected a set, S, of axioms and theorems in memory that, on the basis of experience, appear to be consistent. Suppose a sentence (a hypothesis) is added to S and now one or more contradictions are derived. To remove them, the system destroys the proof sequence by removing the weakest link, the new hypothesis, thereby restoring (apparent) consistency. This is what Rule 3 accomplishes, and this is the way in which *reductio ad absurdum* uses conditional reasoning to find and remove contradictions.

More generally, any sentence on which the contradiction rests can be removed to restore consistency, but it appears that the new hypothesis is

commonly the first candidate. When a contradiction is found in a system that is already in use, the decision about what to alter can itself be a complex problem whose solution will depend on what criteria beyond consistency may be imposed. For example, different criteria may lead the resolution of contradiction in first order logic in the direction of a hierarchy of types or a restriction on what constitutes a set.

5.6. *Negation*

In our proof of Gödel's theorem, a set, K, is used to avoid having a special rule for the cancellation of double negatives. In a production system, $\neg p$ is placed in working memory only when $\neg p$ is established, and is not simply equivalent to the absence of p. (That is, working memory does not assert $p \vee \neg p$ at all times.) Accordingly, to replace $\neg(\neg p)$ by p in memory would require an inference rule. The expression $\neg(\neg p)$ is not a synonym for p, but a belief that $\neg p$ is *not* in working memory; p may also be absent. This distinction is closely related to intuitionist qualms about axioms that cancel double negatives. We will not discuss here whether there is any justification for these qualms, but find it interesting that this distinction should arise in the usual implementation of production systems.

5.7. *Sets*

'Set' is a concept which people and EPAM-like systems can learn as they learn other concepts: by being presented with examples of sets and non-sets and trying to differentiate them, and/or by being given tests for inclusion and exclusion and learning to execute these tests. Thus, they can learn both extensional and intensional ways of delimiting sets, and that the same set may be characterized in both ways. To avoid the Russell paradox, the instructional process can teach that the concept 'set' does not extend to a collection of objects to which the set itself belongs: by assigning names to objects, it is easy to provide a test for such an illegitimate collection.

Extensional equality of sets is easily introduced, and the remaining axioms of Zermelo–Fraenkel set theory can be introduced using similar, easily programmable procedures, as is shown, for example, on pages 238–240 of Shoenfield 1967. Little more is required than a process for forming sets by sequential additions of members.

5.8. *Proof in PM*

Let us next look at the notion of 'proof in PM', and its implications for Rule 1. We examine a computer in action when it is executing the PM system (reasoning in PM), using only the inference rules (and axioms) defined

for PM. We see it construct a sequence of sentences, each adding a direct consequence to the previous sequence and depositing that consequence in memory, where it can become a premise (condition) for the next rule execution. That is, we see that whenever 'x' is the terminal sentence of such a sequence, then by definition, 'x' is provable [(Prov('x'))] and 'x' is added to memory as asserted by Rule 1. This is the semantic meaning of 'Prov("x")' insofar as the proof of Gödel's theorem is concerned. It is irrelevant whether or not the inference rules of PM itself, which construct the proofs, are analytic or not. The analyticity of Rule 1 stems from the inexorable relation, in the observable system, between the constructability of a proof of a sentence ('Prov("x")') and the deposit of the sentence in memory, 'x'.

Of course, if we are interested in PM not merely as a formal object language used for the proof of theorems in the metalanguage but also as a language for carrying out deductions, then we will also be concerned with the analyticity of the rules and axioms of PM. The rule for propositional logic is *modus ponens*, which can be treated as above. Predicate logic calls for generalization, which allows one to assert that '$F(x)$' holds for all x if it holds for a variable argument, v, so that '$F(v)$'. Generalization is built into the production system just as *modus ponens* is: if '$F(v)$', for variable 'v', is in memory, and instantiation of v by a constant 'c' is required to satisfy the condition of a production that has been evoked, then '$F(c)$' will be deposited in memory

A somewhat similar explanation can be provided for the axioms of PM, which is easiest to see if we change them to inference rules by converting the main "$\supset$" connective to the operator "$\rightarrow$" (see Appendix). For example, in PM it is an axiom that $A \vee B \rightarrow B \vee A$. Suppose we have a production whose condition is $A \vee B$. This condition will be satisfied if either A or B is found in memory, in either order. The system will simply, in its operation, not distinguish between $A \vee B$ and $B \vee A$, hence will treat the axiom about commutativity of disjunction as though it were analytic. The same is true of the communitativity of A & B.

We can, of course, give a kind of truth-table account of the axioms. Suppose that we have an axiom that we translate into a production of the form $C \rightarrow A$. Then for those cases where C is satisfied by the contents of memory (is assigned the letter 'T' by the truth table), A will be deposited in memory. This act will be tautological if A is also already T for these cases. For example, in the case of $(p \supset q) \rightarrow (r \vee p \supset r \vee q)$, the condition is satisfied whenever $\neg p$ is in memory or q is in memory. If $\neg p$ is in memory, the action side will be satisfied if $\neg r$ is in memory (the antecedent fails) or r is in memory (the consequent is T). If q is in memory,

then the consequent of the action side is also *T*. The other axioms can be shown to be tautological in the same way. Notice that this use of truth tables is a little different from the usual one, because '*T*' and '*F*' are not, in our account, uninterpreted symbols, but are interpreted by actions of the production system.

The fact that we can show a close connection between the architecture of our hypothesized production system and the acceptance of certain rules as tautologies should not blind us to the fact that the criterion that determines what rules humans accept as tautological and what ones they do not is an empirical criterion, not a tautology. In the history of logic, it is at least implicitly recognized that the meaning of 'tautological' cannot have a wholly formal base; frequently, "intuition" is called to the rescue. It is empirically observable that there is considerable but not complete agreement among humans about whether a rule is tautological. The many species of intuitionist and non-intuitionist logic which differ precisely on this point witness the incompleteness of the agreement.

The degree of agreement that is achieved about what constitutes a tautology can be explained by the empirical knowledge about the inference process that people can acquire in the manner just described. That recognition of consistency or inconsistency, to say nothing of analyticity, depends on experience is illustrated by the history of logic itself. It is a matter of historical record that the problems of what statements are analytic arose largely after the fact, that is, when the systems of logic developed by Frege and Russell were found by the latter (in the course of completing its formalization) to be internally contradictory. Rules (e.g., the type hierarchy) were then found that removed the known contradictions and whose claims to analyticity rest mainly on that achievement, but no consensus has been reached as to exactly what these rules should be or how it is to be shown that a rule is analytic.

We have "solved" the problem of analyticity only by posing another question: Why should human beings come equipped with a production system possessing (as the evidence shows that it does) the particular built-in features that it has: an inference rule corresponding to *modus ponens* and an aversion to contradiction? The most plausible answer to this question is an evolutionary one: The rules for analyticity would contribute to the fitness of an adaptive system to the extent that they protected the veridicality of the system's picture of its world. The class of possible worlds (worlds consistent with the set of sentences in memory) would be neither broadened nor narrowed by the execution of tautological rules of inference. But our present concern is with the actual design of the human processing system, not with the way in which that design came about.

A computer that had evolved to, had been designed with, or had acquired by learning the same built-in criteria for analyticity as those we humans possess could recognize, from its own behavior, or the behavior of other computers that shared its design, the criteria to which it responded. It would experience the same inability that we humans experience to deny the consequence while accepting the premises, or to regard as possible a world that harbored a contradiction. It is this inability, whether human or programmed into a computer, that demarks analyticity.

5.9. *Can Machines Understand as Humans Can?*

Finally, we come to the crux of this paper's whole enterprise: Can a computer (which has been programmed in the manner just suggested) interpret and understand a concept like analyticity as a human being can? And can it thereby understand Gödel's theorem?

We have shown that for a human being to understand Gödel's theorem, he or she must interpret semantically (that is, must understand) certain of the symbols that appear in the formal statement and proof of the theorem. We have argued that semantic interpretation is an empirical process that is to be explicated by providing a description of the human symbol-processing system, and we have sketched out such a description, modeling the system as a von Neumann computer with sensory capabilities. In that system, we defined the relations (1) of denotation of real-world objects by internal addresses, (2) of association between internal addresses of objects and internal addresses of their names and (3) of designation of objects by names. With this apparatus, we were able to describe a mechanism that provides a semantic interpretation of 'analyticity', and by that means permits an understanding of those terms of Gödel's theorem that require interpretation.

But it thereby becomes clear that computers can also interpret 'analyticity' semantically and thereby achieve an understanding of Gödel's theorem. For since our theory of human thought processes was incorporated in a realizable computer model (most parts of which have actually been realized, for example by EPAM and ZBIE), everything we have postulated a human doing can be done by a computer.

Turning our machine "metaphor" around and using it literally, we can describe how the Turing-Machine-cum-sensors interprets symbol strings and achieves understanding in exactly the same way that a person does. At the final stages of interpretation, the machine has the ability to associate the penultimate associations with sensed objects, which in the case before us will be the mechanisms of reasoning systems, the external outputs of the production rules they employ, the axioms they hold initially in memory

and the proofs and theorems they produce. For any statement that can be made about human understanding there exists a corresponding statement of equal validity about machine understanding. In particular, just as people accept *modus ponens* and *reductio ad absurdum* as tautological, so does the appropriately programmed machine; and the mechanisms of the one that produce this acceptance map directly onto the mechanisms of the other. Hence, if we claim that machines do not understand, we must conclude that human beings do not understand.

6. CONCLUSION

What has our examination of the cuneiform tablet taught us about the relation between the ways in which people and computers might respond to it? Clearly, there is no difference in the ways in which the two can apply "mechanical" processes to it. If the computer "understands" the arrows in the tablet, i.e., interprets them as the arrows of its productions (an EPAM-like recognition), it can execute the rules, verifying the derivations in any of the three versions of Gödel's proof, demented or not. So can a person.

A more interesting question arose with respect to the interpretations that have to be made in order to recognize the derivation as a *proof* that is true in all possible worlds, the further interpretations that have to be made in order to recognize it as a *proof of Gödel's general theorem* (i.e., the theorem of Figure 3), and the still further interpretations that have to be made in order to recognize it as a *proof that there is an undecidable statement in the language of PM*. We have argued that in order for a human to achieve either of the latter two recognitions, he or she must provide a suitable semantics and that an effective way to characterize this semantics is to model the human's cognitive system as a Turing Machine or a von Neumann computer with sensory inputs.

If this method can be used to describe the route used by humans to interpret the text of Demented Gödel, then we have seen that the computer can be given the capability of carrying out these same interpretations. Hence, with respect to understanding Gödel's theorem, or to manipulating it symbolically, there appears to be no difference between people and machines.

This demonstration does not, of course, show that there may not be other differences between humans and machines. We have seen that the computer can operate in several languages simultaneously, as people appear to do, but we have not discussed how the computer might decide to use a metalanguage to analyse an object language, or how it could go about creating the metalanguage. We have seen that the computer can check

derivations, but we have not discussed how it might initially discover these derivations – exhibiting the "insight" and "creativity" that people are said to require for this task. We will leave these discussions to another paper.

7. APPENDIX: SOME FURTHER NOTES ON ANALYTICITY

The task of this appendix is to show how all of the analytic statements required as axioms or inference rules of a logic like *Principia* can be derived from properties of a production system that is capable of heuristic search. A convenient axiomatization to use for this purpose is given by Mendelson, pp. 29–31, for the propositional calculus, and extended to first-order logic on pp. 56–58.

The primitive connectives are: $\supset$ and $\neg$.

The rules of inference [denoted by $\rightarrow$] are:

a. *Modus ponens*, A, $A \supset B \rightarrow B$;
b. Generalization, $A(x) \rightarrow (x_i)A$.

The axiom schemas are:

1. $A \supset (B \supset A)$;
2. $(A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C))$;
3. $(\neg A \supset \neg B) \supset ((\neg A \supset B) \supset A)$;
4. $(x_i)[A(x) \supset A(x)]$, where x is free for x_i in $A(x_i)$;
5. $(x_i)[(A \supset B) \supset (A \supset (x_i)B)]$, where A contains no free occurrences of x_i.

1. We get *Modus ponens* from the basic mechanism of the production system: A and $A \rightarrow B$ in memory produce B.

2. We get *Reductio ad absurdum* (Axiom 3) from the organization of heuristic search:

If a search begins at node A, and without any other "questionable" choices reaches an "undesirable node", then return to A and take a different choice. (If the tree is binary, $(A \rightarrow \text{false}) \rightarrow \neg A)$). If the criterion for 'false' is 'B & $\neg B$', then this is equivalent to Mendelson's 3rd axiom, which can be rephrased as: $(A \rightarrow B)$, $(A \rightarrow \neg B) \rightarrow \neg A$; or $(A \rightarrow B$, $\neg B) \rightarrow \neg A$.

3. We get Axiom 1 from the fact that if A is in memory and we add $B \rightarrow A$, then still A.

4. We get Axiom 2 (transitivity) by composition of A, $A \rightarrow B$, and $B \rightarrow C$. If all three of these are in memory, then A and $A \rightarrow B$ adds B to memory, and B and $B \rightarrow C$ adds C.

5. We can treat Axioms 4 and 5 and Generalizaton as definitional of (x) in terms of $A(x)$, or alternatively, of $A(x)$ in terms of (x).

With these interpretations, the production system will behave as if it possessed the axioms and inference rules of first-order logic.

REFERENCES

Eisenstadt, S. A. and H. A. Simon: 1997, 'Logic and Thought', *Minds and Machines* **7**, 365–85.

Mendelson, E.: 1964, *Introduction to Mathematical Logic*, Van Nostrand, Princeton, NJ.

Richman, H. B., J. J. Staszewski, and H. A. Simon: 1995, 'Simulation of Expert Memory Using EPAM IV', *Psychological Review* **102**, 305–30.

Shoenfield, J. R.: 1967, *Mathematical Logic*, Addison-Wesley, Reading, MA.

Siklóssy, L.: 1972, 'Natural Language Learning by Computer', in H. A. Simon and L. Siklóssy (eds.), *Representation and Meaning*, Prentice-Hall, Englewood Cliffs, NJ.

Herbert A. Simon
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 14213-3890
U.S.A.
E-mail: has@cs.cmu.edu

Stuart A. Eisenstadt
Columbia University
Lamont Doherty Earth Observatory
P.O. Box 1000, 61 Route 9W
Palisades, NJ 10964-8000
E-mail: sae@ciesin.org