The Semantics Differentiation of Minds and Machines

Chapter 7 from: Ellerman, David 1995. *Intellectual Trespassing as a Way of Life: Essays in Philosophy, Economics, and Mathematics*. Lanham MD: Rowman & Littlefield, pp. 139-154.

Introduction

After several decades of debate, a definitive differentiation between minds and machines seems to be emerging into view. The differentiating criterion emerges from logic as well as computer science itself. Computing machines (e.g., Turing machines) carry out formal processes of symbol manipulation. Symbols are manipulated solely on the basis of their form, not their intended interpretation. Such symbol-crunching processes are usually called "syntactic" or "formal computational."

This leads to the specification of a process that could not be duplicated on a computer, that is, a process that operates on symbols not according to their syntactic form but according to their semantic meaning. Since such processes would use the meaning or intended interpretation of the symbols, they are semantic rather than syntactic in character. One simple example is the process of logical reasoning in the human mind. The best result that could be obtained with a computer is simulation (not duplication) by programming the computer to formally manipulate the symbols in a manner that is correct or appropriate in view of their intended interpretation. Hence the basic tenet of computer science that computers are formal symbol manipulators yields this "semantics differentiation" between minds and computers.

There are other roads that lead to the same principle. One is the philosophy of mind. A symbol does not have an intrinsic meaning; it is not intrinsically "about something." The semantic content or meaning of a symbol is ascribed to it by a mind. My thought of my coffee cup, however, does have an intrinsic aboutness. It is not simply interpreted as being about the

coffee cup (e.g., by a third party); the thought is about the coffee cup. This intrinsic "aboutness" or "directedness" of most mental states is called "intentionality." The semantics of an intentional mental state is builtin; the semantics of a symbol must be externally imputed or ascribed to it. Intentionality cannot be duplicated by formal symbol manipulation processes alone (e.g., by computers) since those processes are, by definition, independent of the symbols' aboutness. Computers lack inherent intentionality.

When approached from the philosophy of mind, the semantics differentiation would be better called the "intentionality differentiation." It has been expounded and ably defended by John Searle [1980, 1981, 1982, 1983, 1984, ...]. A precomputer version of the intentionality thesis was used by Franz Brentano as a mental-physical differentiation [see McAlister 1976]. The notion of intentionality itself descends from the Middle Ages [see Spiegelberg in McAlister 1976, or "Intentionality" in Edwards 1967]. The notion was also central to the thought of Edmund Husserl and the subsequent philosophy of phenomenology [Dreyfus 1982]. Searle's book [1983] seems based on the principle that intentionality is too important to be left to phenomenologists. The book presents a path-breaking treatment of intentionality, which does not require one to learn the exotic proprietary vocabulary of the phenomenological literature.

The Semantics Differentiation

My treatment of the semantics/intentionality differentiation will be based on the approach from semantics, not intentionality. The notion of "intentionality" is foreign to computer science. No new theory of intentionality will be presented here. Nor is a complete theory of intentionality needed for a differentiation of minds and machines. My point is that the rigorous notions of syntax and semantics drawn from mathematical logic, while far short of a theory of intentionality, are nevertheless sufficient to illustrate the differentiation of minds from machines.

The syntax-semantics distinction is familiar to computer scientists, mathematicians, and philosophers from modern logic: formal syntax versus truth-functional or model-theoretic

semantics. A number of semantic theories have been developed for various special purposes: Kripke semantics, functorial semantics, possible-world semantics, Montague semantics, denotational semantics, situational semantics, and so forth. The arguments for the semantics differentiation will be illustrated using the familiar and noncontroversial semantics of propositional and first-order logic.

The semantics differentiation between minds and machines is internal to computer science itself. The operations of a Turing machine are formal symbol manipulations, and all the operations of a general purpose digital computer can be characterized as Turing machine operations (subject to practical limitations such as memory and time). Computers are syntactical engines whose operations always manipulate symbols solely on the basis of their form, not their intended interpretation. A process that did operate on symbols using their intended interpretation could not be duplicated on a computer. Thus the semantics differentiation is based on the computer science characterization of "what computers do."

This semantics differentiation is not "new"; it is essentially a "folk theory" deriving from work in logic (formal syntax, recursive function theory, and model theory) that dates from the early part of this century. It doesn't need to be rediscovered; it needs to be understood. There are several systematic reasons why the semantics differentiation has been misunderstood. Much of my task is to attempt to resolve these misunderstandings.

The Semantics Differentiation is Nonfunctionalist

Semantics is not differentiated from syntax on behaviorist or functionalist grounds. There is a range of cases where semantics and syntax yield the same "behavior." Consider any consistent and complete formal theory such as an axiomatization of propositional or first-order logic. In an axiom system for first-order logic, the semantic notion of validity and the syntactic notion of theoremhood exhibit the same "behavior" in the sense that they specify the same set of first-order formulas. Under the intended interpretation of the logical symbols, the axioms are

valid and the formal rules of inference preserve validity, so all the theorems are valid. By the completeness theorem, all valid formulas can be derived as theorems. Hence the semantics and the syntax of first-order logic yield the same set of formulas (i.e., the same "behavior" or "function"). Yet no one would interpret this completeness result as proving that the semantics and syntax of first-order logic are the same thing. The completeness theorem does not erase the differentiation between semantics and syntax. Indeed, without the differentiation there would have been no equivalence to prove.

It would be a simple misunderstanding to think that the completeness theorem "proves" that validity *is* a syntactic notion; it only proves that there is a functionally equivalent syntactic notion. Yet, time and time again, one finds the analogous misunderstanding in the artificial intelligence (AI) literature. If a mental (semantic) operation can be successfully programmed on a computer (by definition, a syntactic device), then it is claimed this would show that the mental operation was a "formal computable" operation, i.e., was syntactic. Nothing of the sort follows. First-order validity has been perfectly "programmed" in the formal axiomatizations of first-order logic, but that hardly proves validity is a syntactic notion. What it does show is that the differentiation between validity and theoremhood in first-order logic cannot be made on behavioral or functional grounds.

This line of argument can be applied to the warhorse of Turing machine functionalism, the Turing test itself. Suppose that human mental activities could be successfully programmed so that the program would pass the Turing test with flying colors (i.e., the programmed computer would be behaviorally indistinguishable from a person communicating over a teletype). Two points should be made. First, if a computer does pass the Turing test, it does not demonstrate that the human mental activities are solely syntactic or formal computational (for the reasons given above). Second, a successful Turing test does not disprove the semantics differentiation anymore than does the completeness theorem for first-order logic. It is not a behavioral differentiation.

Because the semantics differentiation is nonbehavioral, it places no behavioral or functional restrictions on "what computers can do." AI critics who project behavioral limitations on computers (e.g., "Computers might play good checkers, but could never play championship chess") cannot base their assertions on the semantics differentiation. The differentiation is very liberally disposed towards work in the artificial intelligence field since it places no theoretical limit on the human behavior that can be programmed (and perhaps improved upon) with a digital computer.

The Irrelevance of the Godel Incompleteness Theorem

Since the semantics differentiation was illustrated using a *completeness* theorem (for first-order logic), it should be clear that the Godel Incompleteness Theorem (for systems with the expressive power of arithmetic) is irrelevant to the argument. One does not need the Godel Incompleteness Theorem to separate semantics from syntax. It neither adds to nor subtracts from the semantics differentiation.

Applications of the Godel Theorem to the mind/machine differentiation are often plagued by an order-of-the-quantifiers misunderstanding. The theorem does not show that there is a true proposition that is formally undecidable in any given formal system of sufficient expressive power. It shows that given any formal system of sufficient power, there is a proposition, true in the intended interpretation, that is formally undecidable in that system (assuming consistency). That proposition is decidable in stronger systems (add it as an axiom), but the stronger systems will generate other undecidable propositions.

The undecidability demonstrated by the Godel Theorem is not absolute; it is relative to the given formal system. If there were an "Absolute Godel Theorem" yielding an absolutely undecidable proposition, then that would be relevant to the semantics differentiation. That would show it to be a behavioral differentiation. But there is no such "Absolute Godel Theorem" and the semantics differentiation by itself places no constraint on computer behavior.

Why Johnniac Can't Add

It is virtually impossible to use computers or even talk about them without adopting an intentionalist idiom. This manner of speaking (and mode of thought) imputes the intended interpretation of the programmer to the computer running the program.

Consider a Turing machine that computes the successor function on the natural numbers N. Given the natural number n as input, it computes n+1 as the output. A contiguous block of n+1 1's on the Turing machine tape denotes the natural number n. The structure and the program of the Turing machine are specified by its "quintuples" [e.g., Minsky 1967, 119], which specify the behavior of the machine in terms of the symbol being read and the current internal state of the machine. The machine is programmed so that when it starts reading the leftmost 1 in the input block of 1's, it moves over to the right end of the block, prints an additional 1, and halts.

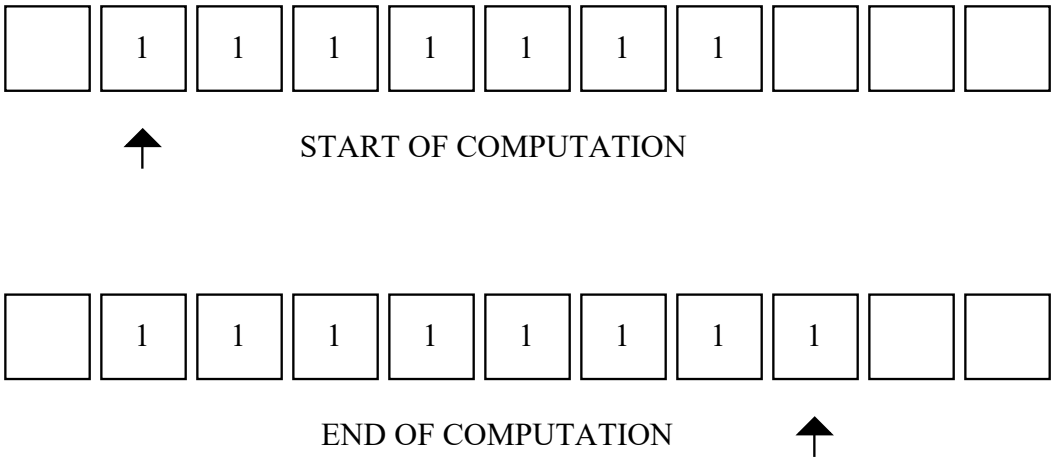

Figure 7.1. Action of Turing Machine to Compute Successor Function

Given a block of 1's representing n, the machine outputs a block of 1's representing n+1, i.e., it computes the successor function.

Or does it? Let $f:N \rightarrow N$ be an effectively calculable isomorphism of the natural numbers onto themselves (e.g., any finite permutation). Now interpret a block of n+1 ones as denoting f(n) instead of n. Then given n, it is denoted by the block of $f^{-1}(n)+1$ ones. The Turing machine

adds a 1 to obtain a block of $f^{-1}(n)+2$ ones that denotes the natural number $g(n) = f(f^{-1}(n) + 1)$. The function g is not the successor function unless f is the identity function.

If the previous Turing machine computed the successor function, do we now have a *different* Turing machine that computes g? No. The Turing machine is exactly the same. It is specified by its "quintuples" and they are completely unchanged. The intended interpretation of the blocks of ones on the tape was never a part of the specification or operation of the Turing machine. It operates on the formal symbols on the tape in a manner independent of their interpretation.

It is only the whole system consisting of the syntactical operations (the Turing machine) plus the semantics (the interpretation of the symbols) that yields a computation of some function on the natural numbers. The Turing machine by itself does not compute the successor function or any of the infinity of other functions obtained by choosing different effectively calculable coding functions $f:N\rightarrow N$. It formally operates on the symbols in a specified way regardless of the coding function used to interpret the symbols.

We thus have the absurd-sounding result that Johnniac cannot add! The apparent absurdity of this conclusion only underscores the entrenchment of the intentionalist idiom in our everyday thought and language. We unthinkingly impute to the computer itself the intended interpretation of the program. But it is precisely in these discussions of the capacities of minds and machines that the semantic interpretation must be bracketed aside from the syntactical symbol manipulation device—no matter how much this cuts across everyday idioms.

The Intentionalist Fallacy

Animism, the imputation of aspects of human mentality to nonhuman entities, is an ancient habit of thought. Primitive tribes took an "intentional stance" [Dennett 1978, 6] toward natural events such as rainstorms. The thunder and lightning expressed the anger of the spirits. John Ruskin called the artistic imputation of emotions to natural events the "pathetic fallacy"

(e.g., "The waves pounded angrily on the rocks"). Neoclassical economists picture capital goods and natural resources as "agents of production" that (who?) "cooperate" together with people to produce the products. "Together, the man and shovel can dig my cellar" or "land and labor together produce the corn harvest" [Samuelson 1976, 536-37]. A shovel is only a tool for the hands while a computer is a tool for the mind. If orthodox economists are unable to resist imputing responsible agency to shovels and land, it is not surprising to find a widespread intentionalist manner of speaking about the most sophisticated machinery ever devised, the modern digital computer.

But our task requires peeling away the popular idiom from the unadorned facts. In our prior example, we saw that contrary to what "everyone knows," computers cannot add. Addition is an operation defined on natural numbers. Computers operate on formal symbols. There is an infinity of ways a programmer could interpret the symbols as referring to natural numbers, and the Turing machine operates independently of all such semantic interpretations.

There was nothing particular about the example. The same sort of differentiation between the syntactic computer operation and its intended semantic interpretation can be applied to any and all computer programs. Computers do not compute the "computable functions." Computers are programmed to formally manipulate symbols in an appropriate manner that can be interpreted as computing the computable functions. Computers do not process semantic information. Computers are programmed to manipulate symbols in a manner that can be interpreted as the processing of information about some subject matter. "The stars run blindly" and so do the so-called "goal-seeking mechanisms." Computers can be appropriately programmed so that by running blindly, certain goals will in fact be pursued.

The imputation of the intended interpretation of the program to the programmed computer could be called the "intentionalist fallacy." The intentionalist fallacy pervades the AI literature. The intentionalist idiom is the *lingua franca* of AI. There is no need to change the idiom. There is a need to understand it as only a manner of speaking or as an "intentional

stance," not as a serious explanatory hypothesis (an hypothesis that, in any case, could be easily refuted by reference to the nonsemantic character of computers). Computers are used by humans to perform computations, to process information, and to seek goals. The syntax of the activity is programmed into the computer so that it will be correct or appropriate in light of the semantics supplied by the human user.

The Amphibious Nature of Programs

There are a number of arguments in the AI literature that attempt to show that computers can have (inherent) intentionality. By far the most common argument derives from the intentionalist fallacy.

In addition to the examples cited above, consider the idea of the "mind as program," i.e., mental processes as instantiations of programs. A program is always "amphibious" in that it has two sides: the program as a formal symbol manipulation process and the intended semantic interpretation of the program. A computer can carry out only the former, while a mind can carry out the latter.

For a simple pre-computer example of this two-sidedness, consider the logical rule of inference *modus ponens*. There is the rule itself as a semantic inference, and there is the syntactic formalization of the rule. For the semantic rule, let P and Q be propositions and let $\Rightarrow$ represent the truth-functional conditional. The semantic rule is:

If P is true and if P $\Rightarrow$ Q is true, then infer that Q is true.

Equation 7.1. Semantic Modus Ponens

This is easily formalized as the syntactic rule:

Given the symbols "P" and "P $\Rightarrow$ Q" as inputs, output the symbol "Q".

Equation 7.2. Syntactic Modus Ponens

The word "deduction," like the word "program," might be used ambiguously to mean either the formal syntactic operation or the underlying semantic rule. The two rules are not

identical. The syntactic rule formalizes and "simulates" the semantic rule but does not duplicate it. It is the semantic rule that gives correctness to the syntactic rule under the usual interpretation of the symbols.

The dual nature of programs is used in one of the exciting methodologies that has grown out of AI, the use of programs as models in cognitive science. The semantic interpretation of the program refers to the mental operations being modeled; the syntactic character of the program allows the model to be run on a computer. These models no more show the mind to operate on formal computational principles than hydraulic or electrical models for the macro-economy (which some economists have constructed) show the economy to run on hydraulic or electrical principles.

The amphibious nature of programs (syntax and semantics) is often abused in AI arguments. When it is argued that programs have intentionality because, for example, they survey alternatives and seek goals (as in a chess-playing program), that refers to the semantic interpretation of the program. When it is said that a computer runs the program, that refers to the program's syntactic side as a formal symbol manipulation routine (independent of its semantic interpretation). Now consider the following argument.

> Programs have intentionality (e.g., they survey possibilities or seek goals).
>
> Computers run programs.
>
> Therefore, computers have intentionality.

The argument is incorrect due to the shift in the meaning of "programs" in the first and second sentences. The first occurrence of "programs" refers to the semantic interpretations while the second refers to the syntactic rules. It is one form of the intentionalist fallacy to impute the semantic interpretation of the program to the program qua symbol manipulation routine.

The Formalization of Semantics

Another line of argument against the semantics differentiation is that semantics can be programmed. This argument might be stated as follows.

> Today, the semantics differentiation has some force because present-day attempts to program semantic relations have been so limited and crude. But eventually sophisticated methods will be developed so that the semantics of thought can be programmed on a computer.

A similar argument holds that the syntax/semantics distinction is relative; the semantics of one level can be formalized in the syntax of a higher level. These arguments fail to understand the nature of the semantics differentiation. I share the optimism about programming or formalizing many of the semantic aspects of thought. The point is that what one then has is a formalization of the notion, a "syntacticalization" of the semantic notion, not the semantic notion itself.

An example can again be found in logic. There are two quite different syntactic systems which might be associated with the semantics of first-order logic. We previously considered the relationship between first-order semantics and a formal axiom system for first-order logic. In a process that might be called "syntactic ascent," the semantic interpretation of first-order syntactic formulas in first-order models can itself be modeled in a formal system of axiomatic set theory.

Proponents of "Strong AI" interpret the semantics differentiation as claiming that the semantic relationship between a symbolic structure and its environment cannot itself be modeled. They argue that a sophisticated computer system could internally represent an external state of affairs. It could then appropriately relate this internal model with the internal symbol structure so the symbols would then exhibit "aboutness." In this manner, strong AI proponents claim that intentionality can be modeled on a computer.

I agree. The semantics differentiation would agree that intentionality can (in theory) be modeled on a computer. Indeed, that modeling process is a computer version of syntactic ascent

as when the semantics of first-order logic is formalized in axiomatic set theory. Suppose that the sentence P(a) in some first-order language L is satisfied in the first-order model B.

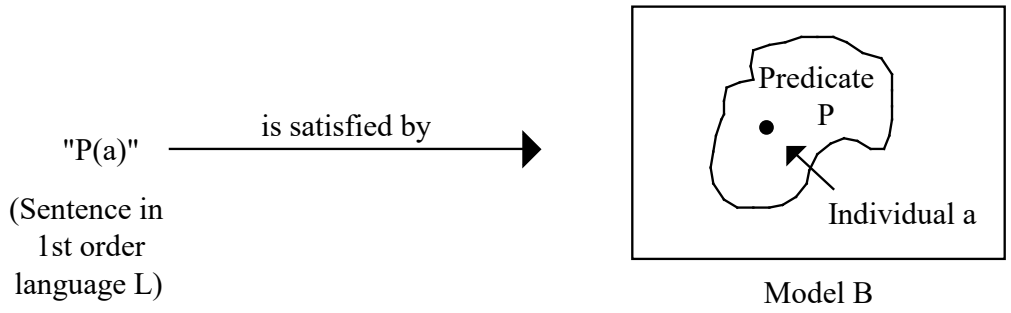

Figure 7.2. Sentence P(a) is Satisfied in Model B

The notion of being a sentence in first-order language L could be represented in a system of axiomatic set theory by, say, a formula WFF("P(a)"). The notion of being a (set theoretic) model for the first-order language could be represented by a formula M(B). Then semantic notion that a sentence P(a) is satisfied by a model B could be represented by some formula S("P(a)",B).

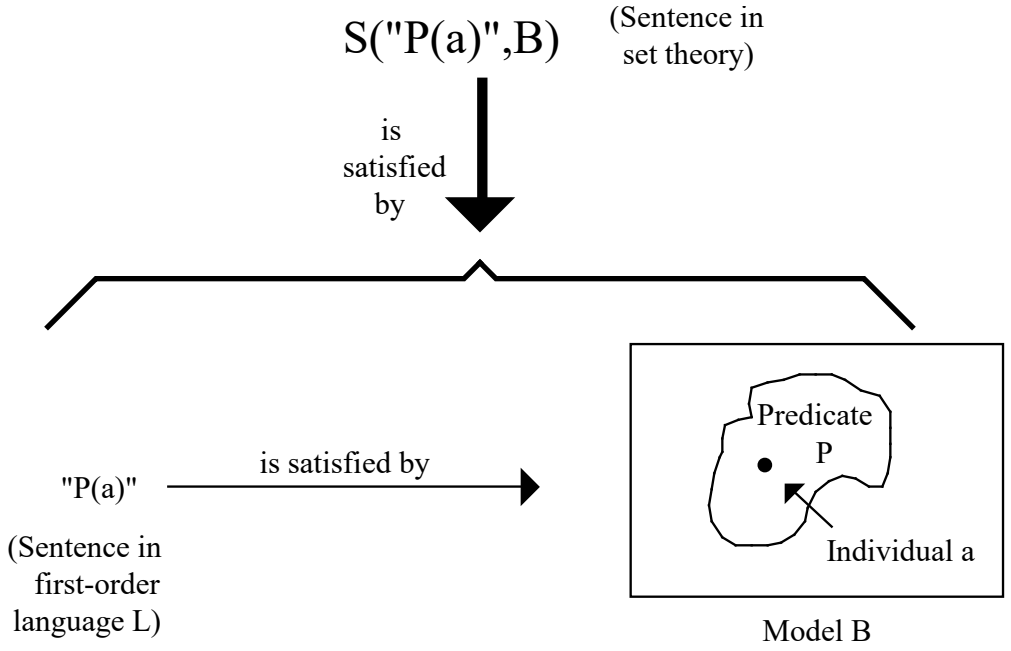

Figure 7.3. Modeling of Semantic Relationship in Set Theory

Relationships between the syntax of a first-order language and its semantic models could then be formally derived within a system of axiomatic set theory. In this manner, the semantics of first-order languages can be modeled in the syntax of axiomatic set theory.

This example of syntactic ascent is rigid and static in comparison with the potential flexible and adaptable computer modeling of the semantic relationship between a perceiving, acting agent and the environment. But both are examples of modeling semantics within syntax, and thus the clear-cut logical example suffices to illustrate the point.

What has been gained by the modeling? Certainly no one would claim that it proves that the semantic relationship between a sentence P(a) and a model B *is* a syntactic notion, only that it can be modeled in the syntax of another formal theory. This is reminiscent of Searle's distinction between duplication and simulation. First-order model-theoretic semantics can be simulated in the syntax of axiomatic set theory, but not duplicated.

The modeling produces a formalization of the semantic relationship, not the semantic relationship being modeled. The distinction between the syntactic machinery (formal set theory in this case) and the intended interpretation (e.g., the semantics of first-order theories) remains the same. The claims of strong AI would require the reproduction (duplication) of the semantic relationship, not its formalization (simulation) in some syntactic system such as a digital computer.

Computer programs can be written *about* anything: playing chess, balancing Aunt Rachel's checkbook, or modeling the semantics of linguistic representations. A program as a syntactic operation is no less formal and no less independent of its content when the intended interpretation happens to be the semantics of certain symbols and representations rather than playing chess or computing the successor function.

Syntax + Robotics = Semantics?

There is another AI argument that might be called the "syntax + robotics = semantics" thesis. Jerry Fodor [1980] gave this "Robot Reply" in his response to Searle's argument. Can intentionality be physically realized by adding physical transducers to symbol crunchers? The argument is that by adding to a digital computer the robotic capabilities to sense and manipulate the environment, the computer can understand the referents of its symbols and thus add a semantic dimension to its syntactic operations. This "Robot Reply" argument is essentially a hardware-oriented variation on the formalization-of-semantics argument considered above.

Robotic abilities certainly do add extra dimensions of power and versatility to a computer. But they do not add a semantic dimension. The computer continues to operate on formal syntactic principles as before, but the range of functions encoded in the syntax can be greatly extended.

For instance, robotic abilities might allow the replacement of certain human inputs of semantic information with purely syntactic links. Suppose a human operator makes certain measurements and then appropriately programs a numerically controlled (NC) machine. An NC machine with robotic sensors could be programmed and calibrated to take the measurements and automatically adjust its control program in an appropriate manner. The analog-to-digital interface between the robotic sensors and the computer could be designed so the relevant physical characteristics of the environment are transformed into the appropriate formal properties of the symbolic structures being manipulated by the computer. The role of the system designer and programmer is pushed back to that of structuring the whole computer-cum-robot system so that by running blindly (i.e., blind to its intended function), it in fact runs in accordance with its intended function.

In simpler terms, a physical connection between a symbol and the environment does not give the symbol an intrinsic aboutness. It does not supply the missing ingredient to transform syntax into semantics. Transducers transmit causes, not meanings. The root of the "syntax +

robotics = semantics" thesis is a confusion between the *cause* of an event and the *meaning* of the event under a certain scheme of interpretation.

> Fred Dretske's gas gauge example illustrates this ambiguity:
>
> Our humble gauge even exhibits the rudiments of intensionality (sic)—representing the amount of gas in my tank. [1983, 82]

Consider the event of the gauge showing "Empty." The meaning of this event under the usual interpretation is, of course, that the tank is empty. The cause of the event is quite distinct. The whole distinction between the mechanism functioning correctly or being "broken" hinges on the correspondence or lack of it between the intended meaning and the cause of the gauge showing "Empty." The physical transducer between the gas tank and gauge does not supply intentionality or meaning to the Empty-event; it only supplies a cause that may or may not correspond with the intended meaning.

If the cause and the meaning of an event were identical, then gas gauges could not malfunction. And the same holds for robots. The robot reply involves the same confusion between cause and meaning as the ascription of intentionality to the gas gauge. The physical transducers of a robotic system will supply a whole new range of causes to affect the symbol-crunching events in the computer. But, as in the case of the humble gas gauge, the causal connection is quite distinct from the intended interpretation or meaning of the symbolic events [for a more information-theoretic treatment of the robot reply, see Ellerman 1986].

Massive Parallelism and All That

Yet another argument holds that advances in hardware and software, such as massive parallelism, will eventually lead to digital computers that have intentionality. Minsky uses this argument in addressing Searle's presentation of the intentionality differentiation.

> I just can't see why Searle is so opposed to the idea that a really big pile of junk might have feelings like ours. He proposes no evidence whatever against it, he

> merely tries to portray it as absurd to imagine machines, with minds like ours—intentions and all—made from stones and paper instead of electrons and atoms. [Minsky 1980, p. 440].

Regardless of one's intuitions about intentionality, it is easy to answer this argument using the semantics approach to the mind/machine differentiation. A Turing machine does not cease being a formal symbol manipulation device as one keeps adding quintuples until it becomes "really big." Adding more and more axioms to a formal syntactic system does not suddenly yield a semantic system. Since a semantic system operates on different principles (i.e., using the meaning of the symbols), it cannot be obtained as a "really big" syntactic system. This is not a question of "evidence"; it follows from the definition of Turing machines or other symbol manipulation systems. Thus the size and sophistication of the computers and programs are quite irrelevant to the semantics differentiation.

Conclusion

Minds crunch symbols according to their intended interpretation. Digital computers crunch symbols solely on the basis of their form. Minds program computers to formally crunch symbols in a manner that is appropriate in view of the intended interpretation of the symbols.

Computers can only carry out formal syntactic processes. The human mind can carry out semantic processes. Thus minds and machines may be differentiated on the basis of two fundamentally different ways of operating on symbols:

1. semantically according to their meaning, and
2. syntactically in a manner independent of their meaning.

In the author's opinion, many AI researchers are "only" trying to obtain what we have agreed is possible, the functional modeling or simulation of human intentionality on a digital computer. That is the success they seek, and they are quite impatient with "philosophical" arguments over whether that is simulation or duplication. While the simulation/duplication

distinction does have philosophical import, it is not "philosophical" in the pejorative sense of being vague since it can be illustrated by examples from logic such as the relationship between first-order semantics and axiomatic set theory. An AI researcher might grant the "theoretical" distinction but consider it "unimportant" because it is nonbehavioral. Indeed, computers are so useful in human society precisely because for behavioral purposes, the semantics differentiation between minds and machines is not relevant.

References

Dennett, Daniel. 1978. *Brainstorms*. Cambridge, Mass.: MIT Press.

Dretske, F. 1983. Precis of "Knowledge and the Flow of Information." *Behavioral and Brain Sciences* 6:55-90.

Dreyfus, Hubert, ed.. 1982. *Husserl, Intentionality, and Cognitive Science*. Cambridge, Mass.: MIT Press.

Edwards, Paul, ed.. 1967. *The Encyclopedia of Philosophy*. New York: Macmillan/Free Press.

Ellerman, David. 1986. "Intentionality and Information Theory." *The Behavioral and Brain Sciences.* 9:1 (March): 143-44.

Fodor, Jerry A. 1980. "Searle On What Only Brains Can Do." *The Behavioral and Brain Sciences* 3: 431-32.

Haugeland, John, ed.. 1981. *Mind Design*. Cambridge, Mass.: MIT Press.

McAlister, Linda, ed.. 1976. *The Philosophy of Brentano*. Atlantic Highlands,N.J.: Humanities Press.

Minsky, Martin. 1967. *Computation: Finite and Infinite Machines.* Englewood Cliffs, N.J.: Prentice-Hall.

Minsky, Martin. 1980. "Decentralized Minds." *The Behavioral and Brain Sciences* 3: 439-40.

Samuelson, Paul. 1976. *Economics.* Tenth edition. New York: McGraw-Hill.

Searle, John R. 1980. "Minds, Brains and Programs" and "Intrinsic Intentionality." *Behavioral and Brain Sciences* 3: 417-24 and 450-56.

Searle, John R. 1981. "Analytic philosophy and mental phenomena." *Midwest Studies in Philosophy* 6: 405-23.

Searle, John R. 1982. "The myth of the computer." *New York Review of Books* 29, no. 7: 3-6.

Searle, John R. 1983. *Intentionality*. New York: Cambridge University Press.

Searle, John R. 1984. *Minds, Brains and Science*. London: British Broadcasting Corporation.