Three Fundamental Misconceptions of Artificial Intelligence

Pei Wang
Department of Computer and Information Sciences
Temple University, Philadelphia, PA 19122, USA
Web: *http://www.cis.temple.edu/∼pwang/*
Email: *pei.wang@temple.edu*

Abstract

In the discussions on the limitation of Artificial Intelligence (AI), there are three major misconceptions, which identify an AI system with an axiomatic system, a Turing machine, and a system with a model-theoretic semantics, respectively. Though these three notions can be used to describe a computer system for certain purposes, they are not always the proper theoretical notions when an AI system is under consideration. These misconceptions are not only the basis of many criticisms of AI from the outside, but also responsible for many problems within AI research. This paper analyzes these misconceptions, and points out the common root of them, that is, to treat empirical reasoning as mathematical reasoning. Finally, an intelligent system, NARS, is introduced as an example, which is neither an axiomatic system, nor a Turing machine in its problem-solving process, and does not use a model-theoretic semantics, though is still implementable in an ordinary computer.

Keywords: *limitation of AI, axiomatization, algorithmization, formalization, empirical reasoning*

1 Introduction

The research of Artificial Intelligence (AI) began half a century ago with the realization that the recently invented computer not only could carry out numerical calculation, but also could manipulate sequences of bits (that were used to stand for other things) according to predetermined procedures (programs). In this way, a computer could have certain capabilities of the human mind.

Naturally, people want to know how far computer systems can go in this direction. Can a computer be given all mental capabilities, or there are certain limitations that a computer cannot pass, no matter how it is designed? When Turing (1950) argued for the possibility of intelligent machines, he did not be-

lieve in any of such limitation, and he used a large part of the paper to refute several claims of limitation he anticipated.

Since then, the debates on the possibility of thinking machines, or the limitations of AI research, have never stopped. These debates happen both within the AI community [Kirsh (1991); Hearst and Hirsh (2000)] and outside it (as "the philosophy of AI") [Graubard (1988); Chalmers (2005)]. There is still no sign of any consensus on the overall conclusion.

This paper is not an attempt to provide a survey of all aspects of these debates. Instead, it focuses on three claims which are at the foundation of the debates:

- An AI system is in principle an *axiomatic system*.
- The problem-solving process of an AI system is equivalent to a *Turing machine*.
- An AI system is *formal*, and only gets meaning according to *model-theoretic semantics*.

As usual, these claims often appear in different forms or are implicitly assumed.

In the following, I will start by analyzing each of the claims, to see what it exactly means, in what sense it is correct, in what sense it is wrong, and why it is a misconception of AI. After that, I will analyze the common root of the three misconceptions, and relate them to the mainstream AI research. Finally, a concrete AI system, NARS, is introduced as an concrete example, in which none of the three claims holds.

2 AI and axiomatic systems

The most typical case of taking AI systems as axiomatic systems is the argument that Gödel's Incompleteness Theorem shows a limitation of AI. This opinion was first proposed by Lucas (1961, 1970, 1996), and later popularized by Penrose (1989, 1994).

The argument goes like this: "Gödel's theorem must apply to cybernetical machines, because it is of the essence of being a machine, that it should be a concrete instantiation of a formal system. It follows that given any machine which is consistent and capable of doing simple arithmetic, there is a formula which it is incapable of producing as being true — i.e., the formula is unprovable-in-the-system-but which we can see to be true. It follows that no machine can be a complete or adequate model of the mind, that minds are essentially different from machines." [Lucas (1961)]

Penrose's conclusion is slightly different, but he also believes that Gödel's theorem shows that "human understanding and insight cannot be reduced to any set of computational rules." [Penrose (1994)] Penrose also identified an AI system with an *algorithm*, which will be discussed in the next section.

Now let us first check what Gödel did in his famous 1931 paper [Gödel (1999)]. He started with the axiomatic system of PM [Whitehead and Russell

(1910)], used a code to translate a self-referencing proposition (which states that it is not provable in PM) into a proposition in PM. If this proposition is false, it leads to a contradiction with itself, so it has to be true. Therefore "the proposition that is undecidable *in the system PM* still was decided by meta-mathematical considerations" [Gödel (1999)]. Furthermore, this problem cannot be solved by introducing more axioms, because a new undecidable proposition can be constructed from the new axiom set [Hofstadter (1979)].

No one has doubt about the validity of Gödel's conclusion, but there is a problem in its applicability to AI systems. Gödel's Incompleteness Theorem, in its original form, targeted axiomatic systems with the expressive power of arithmetic calculations. If all AI systems fall into this category, then indeed we can say that there are truths recognizable by the human mind, but not by computer systems. However, the "if" part of the above conclusion needs to be carefully evaluated.

What is the "axiomatic system" in Gödel's paper? It should consist of "a few axioms and rules of inference" [Gödel (1999)]. By definition, the axioms are true under certain interpretations, and are consistent with each other. The inference rules are "truth-preserving", i.e., derive true conclusions from true premises. Within the system, the truth value of a proposition is determined by checking whether there is a proof (i.e., a sequence of inference steps) that derives it from the axioms.

Many people believe that the human mind does not work in this way, so Gödel's Theorem does not apply to it. Lucas was right when he said: "Gödel's theorem applies to deductive systems, and human beings are not confined to making only deductive inferences. Gödel's theorem applies only to consistent systems, and one may have doubts about how far it is permissible to assume that human beings are consistent. Gödel's theorem applies only to formal systems, and there is no a priori bound to human ingenuity which rules out the possibility of our contriving some replica of humanity which was not representable by a formal system." [Lucas (1961)]

However, in that case, why should an AI system (which is supposed to be similar to the human mind) be built as an axiomatic system? According to Lucas, the only other option is an "arbitrary and irrational" system — "In short, however a machine is designed, it must proceed either at random or according to definite rules", and the former is even worse, since "The system will have ceased to be a formal logical system, and the machine will barely qualify for the title of a model for the mind." [Lucas (1961)]

What is missing in the above conclusion is a third possibility — a system that is neither axiomatic nor arbitrary. According to the previous description, we can see that an "axiomatic system", as far as the current discussion concerns, has the following properties:

1. starting with a constant set of (true) axioms,
2. following a constant set of truth-preserving inference rules,
3. having guaranteed consistency among the axioms and theorems.

Such a reasoning system is *closed*, in the sense that all of its knowledge is in the axioms and rules, and it does not get anything *new*.

On the contrary, in AI many reasoning systems are *open*, in the sense that

1. The system's knowledge is obtained from its environment from time to time, so there is no constant axiom set from which all conclusions are derived.

2. The inference rules often make plausible conclusions according to available knowledge, and consequently the conclusions may conflict with new information.

3. The system may have explicit or implicit contradictions among pieces of knowledge and derived conclusions.

Such an open system is clearly not an axiomatic system in the usual sense, whose conclusions are all derived from a constant and consistent set of axioms. It is not arbitrary, however, because the knowledge and the inference rules do have justifications, according to certain beliefs on how the mind works or should work. Examples of this type of system include nonmonotonic reasoning systems [McCarthy (1989); Davis and Morgenstern (2004)] and reasoning systems with learning capability [Michalski (1993); Giraud-Carrier and Martinez (1995)].

Gödel's Incompleteness Theorem is not directly applicable to an open system defined above, because the proof of the Theorem depends on the three properties of axiomatic systems mentioned before. Of course, we are not saying that such an open system is *complete* — quite the contrary, an open system has many undecidable statements, whose truthfulness depends on the knowledge the system is going to get in the future. However, for such a system S we cannot build a Gödel sentence G that is true but cannot be determined to be so by the system itself. The system still has limitation, but the human mind is not necessarily better than it in general.

If the open system S contains a subsystem PM, there is still a Gödel sentence G whose truth value cannot be determined within PM, but it may be decided within S, using the same "metamathematical considerations" as how we decide G to be true — by "jumping out of the system" [Hofstadter (1979)]. As McCarthy pointed out in his review of Penrose: "One mistaken intuition behind the widespread belief that a program can't do mathematics on a human level is the assumption that a machine must necessarily do mathematics within a single axiomatic system with a predefined interpretation." [McCarthy (1990)]

According to Lucas, the above distinction between S and PM does not matter for the conclusion: "Gödel's theorem applies to physical determinist systems, which are sufficiently rich to contain an analogue to simple arithmetic" [Lucas (1970)], so that the incompleteness can be found even at the hardware level. He suggested that "The conclusions it is possible for the machine to produce as being true will therefore correspond to the theorems that can be proved in the corresponding formal system." [Lucas (1961)]

Even though at the hardware level a computer can be in discrete states, and there are rules determining the state change, it is *not* always isomorphic to an

axiomatic reasoning system. Though an axiomatic system can be implemented by a deterministic physical system, the reverse is not always true. For a deterministic physical system, it is not always possible to take the descriptions of its initial state as "axioms", because in the following states some of the descriptions may become false, which is not allowed within an axiomatic system. Lucas never made it clear why a "deterministic system" is always equivalent to an "axiomatic system", but has simply taken it for granted. As discussed above, an open system in which all knowledge is revisable cannot be taken as "axiomatic".

In a later writing, Lucas moved toward the opposite direction, by taking a high-level view of a computer system. He said "the operations of any such computer could be represented in terms of a formal logistic calculus with a definite finite number (though enormously large) of possible well-formed formulae and a definite finite number (though presumably smaller) of axioms and rules of inference. The Gödelian formula of such a system would be one that the computer, together with its software, would be unable to prove. We, however, could. So the claim that a computer could in principle simulate all our behavior breaks down at this one, vital point" [Lucas (1996)]. Again, such a system may not be axiomatic in the usual sense, so his conclusion does not hold.

In all the conceptual confusions mentioned above, a central issue is the notion of "level", which has been fully discussed by Hofstadter (1979). Lucas's response is: "But although there is a difference of levels, it does not invalidate the argument. A compiler is entirely deterministic. Any sequence of operations specified in machine code can be uniquely specified in the programming language, and vice versa" [Lucas (1996)]. What he failed to see is: when a system can be analyzed in multiple levels, even when the levels are deterministically related to one another, different levels may still have very different properties, and a conclusion may be correct in a certain level, but not in the level above or below it. For example, we cannot say that a computer cannot process decimal numbers or symbols (because they are coded by bits), or cannot use functional or logical programming language (because they are eventually translated into procedural machine language).

If a Gödel sentence cannot be directly built for an open system, how about trying something similar? After all, a Gödel sentence cannot be proved in an consistent, finitely-axiomatized system because it says, roughly, "this sentence cannot be proved in this system" (in Gödel's coding, of course). Such a sentence can be represented in the system if its language is rich enough.

This approach will not have the same effect as Gödel's, for two reasons. First, as shown by Whitely's example, the sentence "Lucas cannot consistently assert this sentence" cannot be consistently asserted by Lucas [Whitely (1962)], so this kind of sentence cannot show that the human mind is more capable than a computer. More importantly, Gödel sentence is a valid proposition in PM, representing a mathematical statement, so failing to decide its truth value is a problem, since it demonstrates the incompleteness of PM in its target domain. On the contrary, the above self-reference sentence, without a second interpretation, has little use for a human mind, so failing to consistently assert it has no

practical consequence (except as an example in a debate about the limitation of AI).

In summary, though Gödel's Incompleteness Theorem is a great result in mathematical logic, and has profound impact to metamathematics, it is not directly relevant to AI, except it shows that an axiomatic system is not a proper model of intelligence. An intelligent system, as the human mind, is not an axiomatic system, but an open system, though it may include axiomatic subsystems. AI systems are not complete, but nor is the human mind, therefore Gödel's result does not indicate a fundamental limitation of AI.

Similar analysis can be carried out in other related claims, such as "An AI system must be consistent" and "An AI system can only do deduction (so as to be truth-preserving)" — it is not the case, because an AI system is not axiomatic. However, it does not mean that the system is arbitrary and irrational, because an open system can also be built according to rational principles, though they are different from the principles behind axiomatic systems.

3 AI and Turing machines

The equivalence between an AI system and a Turing machine seems to be more straightforward than the topic discussed above: As far as an AI system is implemented in an ordinary computer, it is equivalent to a Turing machine, which is an abstract model of all computing machinery.

The definition of a *Turing machine* can be found in almost every textbook on theoretical computer science, for example Hopcroft and Ullman (1979). Informally speaking, a Turing machine is a device with a finite set of *states*, and among them there is a single *initial state*, and one or more *final states*. The machine has an infinitely long tape with a sequence of input symbols on it initially. At each step of a running process, the current state within the machine and the current symbol under a "reading head" on the tape will determine whether there will be any state change, output symbol onto the tape, or reading-head move along the tape. A complete running process starts at the initial state, ends at one of the final states. This process is called "computation". Also, a Turing machine corresponds to an *algorithm* that maps the given input to a certain output.

Even though the above definition is well known and widely agreed upon, there are several aspects that are sometimes ignored or confused when the related notions are used:

- The notion of "Turing machine" does not merely refer to the *device*, but refers to a *process* of it, that is, with a given initial state and a set of final states.

- The notion of "computation", in its theoretical sense, does not mean "whatever a computer does", but the above well-defined process [Sloman (2002)].

- The notion of "computation" describes a process in an abstract device, without saying much about how the process should be *interpreted*. On the other hand, the philosophical discussion on "computationalism" often uses the term "computation" to mean "formal symbol manipulation" (e.g., Scheutz (2002)), which is a different issue (to be addressed in the next section).

Now let's see the claims on the limitations of AI based on the above notions.

> "Our idea of a machine is just this, that its behavior is completely determined by the way it is made and the incoming 'stimuli': there is no possibility of its acting on its own: given a certain form of construction and a certain input of information, then it must act in a certain specific way." [Lucas (1961)]

> "The basic problem facing workers attempting to use computers in the simulation of human intelligent behavior should now be clear: all alternatives must be made explicit. ... In problem solving, the issue is not only how to direct a selective search among explicit alternatives, but how to structure the problem so as to begin the search process." [Dreyfus (1979)]

> "Intelligence cannot be properly simulated by algorithmic means, i.e. by a computer, in the sense that we use that term today." "In particular, a conclusion from the argument in Chapter 4, particularly concerning Gödel's theorem, was that, at least in mathematics, conscious contemplation can sometimes enable one to ascertain the truth of a statement in a way that no algorithm could." [Penrose (1989)]

In summary, these claims are all variants of what Turing (1950) called "Lady Lovelace's Objection" — that is, *to solve a problem, a computer must follow a predetermined algorithm*. Since the human mind seems to follow no predetermined algorithm, some people see this as a fundamental limitation of AI.

To analyze these claims, let us first analyze what is a "problem", and what is a "solution" to that problem. Though these terms sound simple, actually there are two very different situations.

In mathematics and computer science, a *problem* P is usually a *set* including many *instances*. For example, "sorting" as a problem includes many concrete sequences to be sorted. A *solution* of this problem, S_P, should be an *algorithm* (or call it "computational procedure" or "Turing machine") described above, which gives a correct answer for each instance of P. In the following we will call them problem and solution of "Type 1".

Outside mathematics, the situation is completely different. In empirical science and everyday life, a "problem" is usually not a well-defined set, but a concrete instance to be treated individually. In the human mind, there is usually no predetermined algorithm to process all such problems of a certain type in

a unified manner. Instead, they are processed in a case-by-case fashion, with whatever method that happens to be recalled or formed. Furthermore, there is often no well-defined distinction between solutions and non-solutions — it is a matter of degree. Some examples are given by Dreyfus (1979): ill-defined games, e.g., riddles (perceptive guess), open-structured problems (insight), translating a natural language (understanding in context of use), recognition of varied and distorted patterns (recognition of generic or use of paradigm case). Indeed, the above problems, by their very nature, have no algorithmic solution. The human mind can solve them in the sense that some (not all) instances get satisfying (not optimal) solutions. We will call them problem and solution of "Type 2".

Why cannot we expect Type 2 problem solving from a computer system? The common answer to this question is "because a computer works by following algorithms". Sure, but algorithms for what? Since an algorithm for problem P_1 is usually not an algorithm for problem P_2, a system with algorithms does not necessarily have an algorithm for every problem it encounters.

Assume there is a learning system S that faces a type of problem P with many instances. Initially S does not solve the problem well. For problem instance P_i, it gave answer A_i^1, which is far from the best answer. However S is equipped with a learning algorithm L, and after some training, when P_i appeared again, it got answer A_i^2, which is better than A_i^1. During this learning process, does S have an algorithm with respect to P or P_i?

According to the definition of "algorithm", the answer is "No", simply because the system's solution to the problem changes over time. The system does follow an algorithm, L, but it is not an algorithm for *the solving of* P, but an algorithm for *the learning of solving* P, which is a different problem.

Penrose thought that as far as a "changing algorithm" is governed by a meta-algorithm, the two together can be seen as a single algorithm [Penrose (1994)]. This is wrong, because an algorithm (or Turing machine) must be defined with respect to a certain problem-solving process, with determined input-output pairs. Even on a computer running programs (each of which follows a well-defined algorithm), it is possible for the system as a whole to do Type 2 problem solving, that is, to provide a satisfying answer to a problem instance without an algorithm for this problem.

For the previous example, the system S can solve some problems of type P (in the sense discussed above), but it has no algorithm for the problem type P, because it handles P_1, P_2, and so on, in a case-by-case manner. Furthermore, it does not even have an algorithm for the problem instance P_i, because its solutions change from time to time. Then how about P_i^j, the jth *occurrence* of P_i? Since S is a deterministic system, some people may suggest that the system's solution to P_i^j follows an algorithm. However, since such an "algorithm" may never be repeated by the system again, it is very different from how the notion of algorithm is defined in theoretical computer science, which implies a predetermined and repeated behavior pattern. Also, we can no longer argue that the human mind does not follow algorithms in this sense, because all the examples listed before only shows that the human behaviors change from situation to situation.

If a system can learn and work in a context-sensitive manner, then for certain problems its behavior does not correspond to a Turing machine, also because there is no fixed initial and final state. For the same problem, in different time and context the system may provide different answers. In this sense, the system can solve problems that are not algorithmic or computable. This conclusion does not contradict the conclusions about the undecidability and incompatibility of certain problems, obtained in theoretical computer science, because here "problem" and "solution" get different meanings. This meaning change is not begging the question, because this is exactly what we mean when saying "the human mind can solve a problem without an algorithm".

Even for such a system, its behavior can still be seen as a Turing machine, if we take its life-long input as a "problem", and its life-long output as the "solution" to the problem. However, this is not the level at which we talk about the problem-solving behaviors of the human mind. If a human being is reborn with exactly the same innate state, and lives through exactly the same experience as the previous life, can different behaviors be observed? It seems that nobody has argued for human non-algorithmic behavior in this sense. Rather, all the arguments are about human behaviors that change with experience and context, which, as argued above, can also happen in a computer system.

Similar conclusions have be proposed as "something can be computational at one level, but not at another level" [Hofstadter (1985)], "cognitive processes that, although they involve more than computing, can still be modeled on the machines we call computers" [Kugel (1986)], and "The irrelevance of Turing machines to AI" [Sloman (2002)]. These conclusions are not widely accepted, again because of the level confusion discussed previously. Just because a computer follow algorithms *at a certain level*, some people think that it can only solve problems for which the system has algorithms, which is an invalid conclusion.

For a computer doing Type 2 problem solving, the traditional analysis of computability and computational complexity cannot be carried out anymore, simply because the system's behavior does not follow a fixed procedure, and the expense of resources also changes from case to case. However, it does not mean that there is no limitation in the system's capability. Actually, Type 2 is a "weaker" manner of problem solving (compared to Type 1), because the system may not be able to find the best solution for certain instances, and may even not be able to find any solution for certain other instances. It is valuable only when Type 1 problem solving cannot be carried out for some reason, and provides a way for AI to go beyond the limitations associated with the notions of algorithm, computation (in the theoretical sense), and Turing machine.

4 AI and model-theoretic semantics

The previous two misconceptions argue against the possibility of AI by proposing something that a computer system cannot do, because of its equivalence to a conceptual device, i.e., axiomatic system and Turing machine, respectively. The opinion to be discussed in this section assumes a computer system can be built

to *behave* just like a human mind, then argues that such a computer system still lacks genuine intelligence, because a computer is a *formal* system that only has *syntax*, but no *semantics*.

The most well-known version of this opinion is Searle's "Chinese Room" thought experiment [Searle (1980)]. Behind the unrealistic scenario described in this argument, the theoretical reasoning is direct and clear: "Because the formal symbol manipulations by themselves don't have any intentionality; they are quite meaningless; they aren't even symbol manipulations, since the symbols don't symbolize anything. In the linguistic jargon, they have only a syntax but no semantics." [Searle (1980)]

To analyze this conclusion, we need to start with the concepts of "syntax", "semantics", and "formal system". To be more concrete, let us see the case of a reasoning system.

A typical reasoning system implements a *logic*, consisting of

1. a language to represent knowledge,
2. a set of inference rules to derive new knowledge from given knowledge,
3. a semantics to link the language to the world, and to justify the rules.

To implement a logic in a reasoning system means to write programs to process the language, to keep the knowledge, to carry out the inference, and to communicate with the world.

In such a system, "syntax" indicates the formats of valid sentences of the language and the patterns of the inference rules; "semantics" indicates the principles of determining the *meanings* of the words and the *truth-values* of the sentences in the language.

A reasoning system can be "formal" (or "symbolic") in two different senses:

1. It has a *formal syntax*, that is, its language is defined by a formal grammar, and its inference rules are all formally defined.
2. It has a *formal semantics*, that is, the meaning and truth in its language are determined according to an interpretation that maps the items of the language into objects and relations in the world.

Now let us explain these two senses in detail.

A formal grammar specifies the formats of valid sentences. For example, in first order predicate calculus, a grammar rule may look like "$s ::= P(c)$", where s is a proposition, P is a predicate, and c is an argument. Such a grammar rule gives one (but not the only one) way to construct a proposition. In the rule, s, P, and c are "symbols" in the sense that they can be replaced by different constants to get different propositions. The grammar rule is "formal" in the sense that it only specifies the format of a proposition, without saying anything about its meaning, or the meaning of the involved symbols s, P, and c.

A formal inference rule specifies the pattern of a single inference step, with its premises and conclusion. For example, in propositional calculus, an inference

rule may look like "$\{p, p \to q\} \vdash q$", where p and q are propositions, and "$\to$" is the implication relation. This rule states that q can be derived from p and $p \to q$. This rule is formal, because p and q can be substituted by any propositions, and the rule remains valid.

For a reasoning system S using a language L, the above grammar rule and inference rule do not appear in L, but in its meta-language, which is part of the specification of S, describing what kind of sentence can appear in L, and how the sentences can be used in inference, at a general level.

A formal semantics for L itself is a quite different matter. According to it, if $Yellow(Tweety)$ is indeed a sentence of L, then the meaning of $Yellow$ and $Tweety$ is determined by an interpretation I, which maps $Yellow$ into an attribute **Yellow**, and $Tweety$ into an object **Tweety**, in a model **M**, which is a set of descriptions of (the relevant part of) the world, consisting of objects, their attributes, and relations. Under this interpretation, $Yellow(Tweety)$ is true if and only if in **M** the object **Tweety** indeed has the attribute **Yellow**. According to this semantics, for the words and sentences of a language to get meanings and truth values, the sufficient and necessary condition is the existence of a proper interpretation. This kind of semantics is called "model-theoretic semantics".

If a reasoning system is formal in both senses (i.e., with formal syntax and semantics), Searle's conclusion directly follows. For such a system, no matter what propositions are in its memory and what inference rules it can use, the system does not necessarily have access to the interpretation that gives *meaning* to them — to the system, they are just meaningless symbols. According to model-theoretic semantics, syntax and semantics are independent of each other — "Syntax is not sufficient for semantics. That proposition is a conceptual truth. ... Computer programs are entirely defined by their formal, or syntactical, structure" [Searle (1984)], so cannot have true understanding, and therefore, intelligence.

However, what if the system is only "formal" in the sense of syntax, but not semantics? That is, the system uses a language with a formal grammar, as well as a set of formally defined inference rules, but does not define meaning and truth according to an interpretation. At least, this is not a self-contradicting idea, because formal syntax and formal semantics are not the same thing.

Model-theoretic semantics came from metamathematics, especially the work of Tarski. It has made great contribution to the research, understanding, and application of mathematics. It is sort of the standard semantics used by mathematical languages. Though Tarski also suggested using it for natural languages, the success there has been quite limited. "As regards the applicability of semantics to mathematical science and their methodology, i.e., to meta-mathematics, we are in a much more favorable position than in the case of empirical sciences." [Tarski (1944)]

What is the major difference between mathematical sciences and empirical sciences, as far as semantics is concerned?

A major difference is that a mathematical theory is *abstract*, and is not especially about any concrete domain. As Russell said, "If our hypothesis is about anything, and not about some one or more particular things, then our

deductions constitute mathematics. Thus mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true" [Russell (1901)]. The last sentence sounds like Searle in his Chinese Room, except that for mathematics, this is desired. When an abstract mathematical theory is to be applied to a concrete domain, an interpretation gives the theory meaning and truth, as specified by model-theoretic semantics. It is this possibility of multiple interpretations that makes mathematics a general tool of thinking that can be used for many purposes.

In empirical sciences and everyday life, most theories and knowledge are *concrete*, in the sense that words in the language used there cannot be freely interpreted. What a word means to a human being depends on its role in the person's experience, so no interpretation is needed.

In her response to Searle's conclusion, Boden (1993) pointed out that the input and output link a computer to the world, and proposed that "meaning and understanding are grounded in causal powers, in *what a creature can do in the world*". This is related to the "Symbol Grounding" problem of Harnad (1990), where he suggested to "ground" the meaning of symbols in sensory categories of the system.

In the study of natural language, there are many theories which explain meaning and truth in ways fundamentally different from model-theoretic semantics, as exemplified by Harman (1982), Lakoff (1988), Ellis (1993), and Langacker (1999). According to these theories, syntax, grammar, semantics, and categorization are no longer isolated from each other, and meaning and truth are closely related to the system's experience and cognition processes. If these theories can be applied to natural languages, why cannot they be used on a language with a formal syntax?

Model-theoretic semantics is not the only choice for a language defined by a formal grammar. There is *procedural semantics*, and "According to this theory the meaning of expressions (in the internal language) are characterized by a certain kind of abstract procedure, which under appropriate circumstances can be used to determine the truth or falsity of propositions and to assess the success or failure of actions" [Woods (1986)]. Another idea is to define semantical notions, such as truth and meaning, with respect to a knowledge base containing relevance experience [Kowalski (1995)] or a database containing background information [Mancini and Bandler (1988)]. Unlike in model-theoretic semantics, in these cases the semantic notions are not defined by an interpretation, and syntax and semantics are no longer independent of each other.

The problem here is that people often implicitly assume that if a language has a formal syntax, it must also have a formal semantics. The term "formal language" merged the two different notions together in people's minds. According to the previous analysis, we can see that a language with a formal syntax does not have to use a model-theoretic semantics. Instead, the symbols in the language can be grounded in something outside the language, and in that case, Searle's conclusion does not apply anymore.

5 AI and mathematical reasoning

The three issues discussed above have a common nature — each of them is about the relation between AI and a metamathematical notion: *axiomatic system*, *Turing machine*, and *model theory*, respectively.

All of the three notions were initially proposed for mathematical reasoning. Roughly speaking, "axiomatic system" is the desired way to organize mathematical knowledge and to make valid inferences; "Turing machine" is the desired way to specify a complete problem-solving process in mathematics; and "model theory" is the desired way to map an abstract mathematical theory into a concrete domain. These notions, as well as the related ones, formed the foundation for modern study of mathematics for decades. There is little doubt about their applicability in mathematics.

Computer science was initially about numerical calculations. Since then, though the domain has been greatly expended into all kinds of "information processing", the theoretical foundation of the field remains dominated by the heritage from mathematics, including the three notions discussed above. Even though in computer science we rarely build fully axiomatic systems, we do hope to start with "correct input", and to produce from them "correct output", step by step. For a problem to be solved, we usually require the solution to be an algorithm that works for every instance of the problem. For the "symbols" in a program, we do take their meaning to be the objects in the world denoted by them. These notions have supported computer science and have achieved great success [Halpern et al. (2001)].

For the above historical reasons, when AI began to grow in the field of computer science, these notions again were accepted into the theoretical foundation.

For example, McCarthy said: "If a computer is to store facts about the world and reason with them, it needs a precise language, and the program has to embody a precise idea of what reasoning is allowed, i.e. of how new formulas may be derived from old. Therefore, it was natural to try to use mathematical logical languages to express what an intelligent computer program knows that is relevant to the problems we want it to solve and to make the program use logical inference in order to decide what to do" [McCarthy (1989)]. Guided by similar ideas, Hayes' "Naïve Physics Manifesto" proposed to put everyday knowledge into the format of mathematical logic, because "Modern formal logic is the most successful precise language ever developed to express human thought and inference" [Hayes (1979)]. One of the largest AI projects, Cyc, has carried out such a practice for a few decades [Lenat (1995)].

Many people believe that the goal of AI is nothing but to find new algorithms, i.e., to "identify a task domain calling for intelligence, then construct a program for a digital computer that can handle tasks in that domain" [Newell and Simon (1976)]. So in this sense, there is little difference between AI and traditional computer applications, except that the "problems" to be solved in AI are those that the human mind can handle well. According to Schank (1991), "AI entails massive software engineering. To paraphrase Thomas Edison 'AI is 1% inspiration and 99% perspiration.' We will never build any real AI un-

less we are willing to make the tremendously complex effort involved in making sophisticated software work".

The influence of model-theoretic semantics can be recognized in the well-known "Physical Symbol System Hypothesis" of Newell and Simon, where the "symbols" still get their meaning through "designation" and "interpretation", even though they are believed to be able to get their "intended interpretation" through a causal relation between a symbol and a process — the symbol "can invoke and execute its own processes from expressions that designate them." Therefore, "the real contribution of logic is not its usual rather sparse syntax, but the semantic theory which it provides"[Newell and Simon (1976)].

Now we can see that the belief that an AI system should be axiomatic, algorithmic, and formal — close to what Hofstadter (1985) called a "Boolean Dream" — is indeed held by many AI researchers. Therefore, the criticisms from Dreyfus, Lucas, Penrose, and Searle are not against a straw man, but against certain influential (even dominating) AI approaches.

As far as these criticisms are taken to mean "a mind is not axiomatic, algorithmic, and formal", they are quite justifiable. As analyzed previously, being axiomatic fails to be open to novel experience, being algorithmic fails to be adaptive and context-sensitive, and being formal fails to ground meaning in experience. Overall, the dependency on these mathematical notions is partly responsible for many failures in AI research in the past.

On the other hand, these criticisms finally say that "a computer cannot be a mind, because it is inevitably axiomatic, algorithmic, or formal", so according the previous discussions, they are wrong. Instead of targeting certain *approaches* toward AI, they targeted the *possibility* of AI, no matter what approach is involved. Since their analysis missed certain important possibilities, their final conclusions are unjustified.

Why the notions of axiomatization, algorithmization, and formalization work excellently in mathematics and computer science, but run into trouble in AI? Simply speaking, this is because *mathematical reasoning* and *empirical reasoning* follow different "logics". Here "logic" is used in its broad sense, meaning regularity, principle, rationale, and so on.

Mathematics and empirical sciences were used to be thought of as following the same logic, and it is just that in mathematics this logic reaches its purest and perfect form. Later, from the study of the philosophy of science, such as the works of Popper (1959), Kuhn (1970), and Lakatos (1999), it is revealed that empirical sciences do not work by deriving theorems from axioms while following well-defined "scientific methods". Even in mathematics, mathematical logic is the logic followed when research results are *validated*, not when they are *discovered*.

Since most AI works are not about the validation of results in mathematics, it should not be a surprise that some mathematical notions cannot be applied. Actually few AI researchers treat AI research as research in mathematics. Instead, they believe that the mathematical tools, though not directly applicable, should not be *abandoned*, but be *extended* or *modified* for AI purposes.

For example, McCarthy (1988) thought "expressing knowledge and reasoning

about the commonsense world in mathematical logic has entailed difficulties that seem to require extensions of the basic concepts of logic", so he proposed a "non-monotonic logic". This logic is different from classical logic because of its nonmonotonicity, but is still very similar to classical logic in the other aspects. Similarly, because for many hard problems tractable algorithms cannot be found, people turn to heuristic algorithms and approximate algorithms, which relax the requirements of classical algorithms, though are still within the scope of computation theory [Simon and Newell (1958)].

The problem with these "revisionist" approaches is that each of them is targeting one issue of classical theory, while ignoring the others. To build a general intelligence system in this way is difficult, if not impossible, because the "tools" built under different considerations won't work together smoothly [Wang (2004b)]. Furthermore, these approaches miss the fundamental difference between mathematical reasoning and empirical reasoning.

One key issue that distinguishes mathematical reasoning from empirical reasoning is that the former assumes the *sufficiency* of its *knowledge* and *resources* with respect to the problems to be solved. A system based on a constant set of axioms and inference rules does not need to *learn*, because all the relevant knowledge is already in the axioms and rules. To prove the truth of a statement, the expense of computational resources (mainly time and space) is irrelevant, as far as they are finite. The system does not attempt to change itself to increase its capability. If it fails to solve a problem that is beyond its capability, the fault is not its, but belongs to whoever gave it the problem.

On the contrary, in empirical reasoning the system is always open to new knowledge and problems. The new knowledge may conflict with old knowledge, and a new problem may be beyond the system's current capability. The system has to resolve the inconsistency, and makes its best guess (based on its experience) on how to solve the problem. All these activities must be carried out within the restriction of available resources of the system. Because of this fundamental difference, traditional theories about mathematical reasoning cannot be used in empirical reasoning, and minor revisions and extensions are not enough. We need something principally different.

A practical reason for many AI researchers to stay with the traditional theories is the lack of alternatives. As McDermott (1987) put it, the logicist position wants "to argue that mathematic logic is a good notation for writing the knowledge down. It is in some sense the *only* notation. The notation we use must be understandable to those using it and reading it; so it must have a semantics; so it must have a Tarskian semantics, because there is no other candidate". Those people who do not like the notions of axiomatization, algorithmization, and formalization have thrown the idea of reasoning system away altogether, and attempt to achieve intelligence through connectionism, evolution, robotics, or dynamics.

6 A non-axiomatic reasoning system

In the previous sections, I have argued that it is quite *possible* for an AI system to be not axiomatic, algorithmic, or formal, in the most natural senses of these notions. In the following I will go further to show that such a possibility has turned into reality in an AI system called NARS (Non-Axiomatic Reasoning System).

Given the nature of this paper, here I will not try to give a comprehensive introduction to the system, nor to propose NARS as the best way to achieve AI. Instead, I only want to use NARS as an example to show that an AI system can go beyond the "limitations" set by the theoretical notions discussed above.

NARS is a *reasoning system*, in the sense that it has its language, semantics, inference rules, memory, and control mechanism. All the technical issues mentioned in the following have been described with much more details in other publications (to be cited in the following). For overviews of the system, see Wang (1995, 2004b), and a comprehensive description of the project has appeared in Wang (2006). The most recent implementation of NARS is linked from my homepage, and can be demonstrated on-line.

NARS uses a language defined by a formal grammar, and a set of inference rules that are also formally defined [Wang (2006)]. Different from the most typical form of mathematical logic, the language and rules of NARS do not belong to the "predicate logic" family, but to the "term logic" family, together with Aristotle's logic. There are two major differences between these two families:

1. In a predicate logic, the basic form of sentence has *a predicate and a list of arguments*, e.g., "$P(a_1, a_2)$". In a term logic, the basic form of sentence has *a subject term and a predicate term linked by a copula*, e.g., "$S \rightarrow P$". Intuitively, this sentence represents a special-general relation, such as "$bird \rightarrow animal$".

2. In a predicate logic, the basic form of inference rule is *truth-functional*, that is, "$\{P\} \vdash Q$" is a valid inference rule if and only if "$\neg P \vee Q$" is true, while P and Q do not need to be related in contents. In a term logic, the basic form of inference rule is *syllogistic*, that is, the two premises must have a shared term, and the conclusion consists the other two terms, e.g., "$\{S \rightarrow M, M \rightarrow P\} \vdash S \rightarrow P$".

NARS does not use model-theoretic semantics, but an "experience-grounded semantics" [Wang (2005)]. According to this theory, the truth value of a statement, like "$S \rightarrow P$", and the meaning of a term, like S, are determined according to the system's experience. Roughly speaking, the "experience" of the system is a stream of input sentences that system gets from its interaction with its environment. At any given moment, for a statement "$S \rightarrow P$" the system may find some positive evidence supporting it, and some negative evidence refuting it, from its experience. The *truth value* of the statement is a pair "$< f , c >$", where f is the *frequency* of positive evidence among total evidence, and c is the

confidence the system has on the frequency judgment, as an increasing function of total amount of evidence. The *meaning* of a term is defined to be its experienced relations with other terms.

According to the above semantics, an inference rule in NARS determines the truth value of its conclusion, by checking how much (positive and negative) evidence the premises provide directly for it. Different combinations of premises correspond to different types of inference, so have different truth-value functions. For example, the basic rules (truth values omitted) are [Wang (1994, 1995, 2006)]:

$$\begin{aligned} \{S \rightarrow M, M \rightarrow P\} &\vdash S \rightarrow P \quad (deduction) \\ \{M \rightarrow S, M \rightarrow P\} &\vdash S \rightarrow P \quad (induction) \\ \{S \rightarrow M, P \rightarrow M\} &\vdash S \rightarrow P \quad (abduction) \\ \{S \rightarrow P, S \rightarrow P\} &\vdash S \rightarrow P \quad (revision) \end{aligned}$$

The memory of NARS can be seen as a network, with terms are nodes, statements as links, and a truth value indicates the "strength" or "weight" of the link [Wang and Hofstadter (2006)]. Under this interpretation, different inference rules correspond to different ways to make new links (and nodes), as suggested by Minsky (1985). This network also shares certain properties with connectionist networks [Wang (1995)].

In the memory, each node and link has a priority value attached, indicating its relative importance and relevance to the system at the current time. The system can absorb new knowledge or answer questions according to available knowledge. When such a task appears, the system uses the knowledge in its memory to process it by inference, which means to add or change the related nodes and links (for new knowledge), or to locate or produce related nodes and links (for new question). In each inference step, the node and links are chosen probabilistically, according to their priority, that is, a node/link with a higher priority has a higher chance to be used. After the step, the priority values of the involved nodes and links are adjusted, according to the immediate feedback about their importance and relevance.

For a given question, the system reports the best answer (e.g., with the highest confidence value) it has found so far, then continues to look for better answers. Since the system at any time usually has many (input and derived) tasks, the time/space resources each of them gets are the result of their competition, and are usually not enough for exploring all possibilities of its processing. For a given task, its processing path and result depend on the knowledge and tasks in the memory, as well as the priority distribution among them. All these factors change over time, depending on what happens in the interaction of the system and its environment [Wang (2004a, 2006)].

Designed in this way, NARS is indeed a reasoning system, with the same major components (language, rules, semantics, memory, and control) as traditional reasoning systems. However, since these components are designed according to the requirement of "adapting and working with insufficient knowledge and resources", their properties are fundamentally different from those of the traditional systems.

NARS, as its name indicates, is not axiomatic [Wang (1994)]. In the system, there is no truth value that cannot be modified by future evidence, including those of the input statements. The inference rules may derive conclusions which conflict with new knowledge. There is no guarantee that the system does not contain implicit inconsistency among its knowledge. Even so, the system is anything but arbitrary, and is designed according to certain beliefs about intelligent reasoning. Furthermore, the system may contain subsystems that are treated as axiomatic in certain ways [Wang (2006)].

The problem solving processes in NARS does not follow problem-specific algorithms [Wang (2004a)]. For each task, the inference procedure is not predetermined, but formed at run-time by the related knowledge that happens to be recalled. There is no fixed initial and final states for a task — it can be accepted at any time, and ended at any time. If the same task is given to the system at different time, its processing is usually different. For instances of the same type of problem, the system's treatments may be different, too. Even so, for certain problems that the system happens to have sufficient knowledge and resources, it can still follow algorithms for them. Furthermore, on a *micro* scale (each inference step) the system still follows predetermined algorithms, and on a *macro* scale (each life-long input/output), the system is still equivalent to a Turing machine. However, these scales should not be confused with the *central* scale, where the system's problem solving processes are defined.

NARS as a whole does not use model-theoretic semantics [Wang (2005)]. As mentioned previously, to the system, the truth value of a statement and the meaning of a term are determined by the system's experience about them. Outside observers can still freely interpret the terms and statements, but that has little to do with the system. With the change of experience and context, meanings and truth values also change in a non-arbitrary way. Though NARS has no human-like sensorimotor mechanism yet, it can still *ground* truth and meaning in its *experience*. Even so, for its axiomatic subsystems, it may use model-theoretic semantics.

Working in this way, none of the proposed "limitations" to AI discussed before applies to NARS.

7 Conclusion

In the discussions about the limitation of AI, there are three fundamental misconceptions:

1. An AI system is in principle an *axiomatic system*.
2. The problem-solving process of an AI system is equivalent to a *Turing machine*.
3. An AI system is *formal*, and only gets meaning according to *model-theoretic semantics*.

There are many implications of these misconceptions, such as:

1. The capability of an AI system is restricted by Gödel's Incompleteness Theorem, while the human mind is not.

2. The processes in an AI system all follow predetermined algorithms, so the system cannot be original or flexible.

3. Since an AI system uses a formal language, its processes only have syntax, but no semantics.

These claims are misconceptions, because they treat *certain ways* to see or use a computer system as *the only ways* to see or use it. The differences among these "ways" often correspond to the difference *levels* or *scales* of description [Hofstadter (1979)]. A notion that correctly describes a system at a certain level or scale may be improper or wrong at another one.

What make things even more complicated is the involvement of concepts like "model" or "emulation". To study the nature of a target system X, a mathematical tool or platform T is often used to build a model M, which shares certain properties with X, so can be used to emulate X for certain purposes. In this situation, it is crucial to clearly distinguish the properties of X and M from those of T in general.

In the current discussion, the X to be studied is an "intelligent system" as exemplified by the human mind, the T is an ordinary computer system, and the M is the AI system built in T according to the designer's understanding of X. The conclusion of this paper is that neither X nor M should be seen as axiomatic, algorithmic, or semantically formal. However, this conclusion does not directly apply to the tools that is used to build M. For example, if M is a virtual machine, it is possible to emulate it in a host virtual machine, which can be described abstractly as a Turing Machine; if M is a reasoning system, it is possible to describe its meta-theory as an axiomatic system, or to understand its meta-language according to model-theoretic semantics. In these cases, we should not confuse a virtual machine with its host machine, or a reasoning system with its meta-theory. This paper is about what properties M (and therefore X) has, but not about what properties T has. The mathematical tools and platforms contributed by traditional theories still play important roles in the building of host machines and meta-theories, though we do not want to build AI systems directly according to them.

Many people outside the field of AI accept these misconceptions, because they take the related notions as "in principle" specifying what an AI system, as a computer system, can do. They fail to see that these notions do not specify all possibilities of a computer system, especially an AI system. Their criticisms have the value of showing the limitation of certain AI approaches, though not all approaches.

Many people inside the field of AI accept these misconceptions (though not all of their implications), because they prefer well-known theories, which have been proved to be successful in certain related fields. They fail to see that those fields have fundamental differences from AI. Their research works make contributions to computer science, though not much to AI.

It has been argued by Wang (1995) that many properties of *intelligence* can be explained as the capability of *adaptation with insufficient knowledge and resources*. The theories about mathematical reasoning cannot be directly applied to AI, because they do not assume these insufficiencies. The development of NARS shows the possibility of building a reasoning system on this new theoretical foundation, and the system has displayed many properties similar to human intelligence.

NARS is not the first computer system that is not axiomatic, algorithmic, and formal. Roughly speaking, an *open* system is no longer *axiomatic*, an *adaptive* system is no longer *algorithmic*, and a *grounded* system is no longer *formal*. There have been many such systems in AI history. However, these practices have not been fully reflected in the theories of AI. As a result, notions of axiomatic system, Turing machine, and model theory are still taken by many people as setting the foundation, as well as limitation, of AI systems.

To say that the notions in mathematical logic and theory of computation do not set limitation for AI, it does not mean that AI has no limitation at all. The problem solving capability of NARS, like that of the human mind, is limited by the system's available knowledge and resources. However, this kind of limitation shows nothing that can be done by the mind but not by the machine.

AI, because of its special nature, should not be simply taken as a sub-domain of computer science. Even though AI systems are eventually implemented in computer systems, AI needs a different theoretical foundation. NARS is a step toward this direction. The project has not been completed yet, though the results so far have shown many interesting features, as well as many novel solutions to traditional problems.[Wang (2006)]

Acknowledgment

I benefited from discussions with Ben Goertzel while writing this article.

References

Boden, M. A. (1993). The impact on philosophy. In Broadbent, D., editor, *The Simulation of Human Intelligence*, pages 178–197. Blackwell, Oxford.

Chalmers, D. J. (2005). Contemporary philosophy of mind: An annotated bibliography, Part 4: Philosophy of artificial intelligence. Web page at `http://consc.net/biblio/4.html`.

Davis, E. and Morgenstern, L. (2004). Introduction: Progress in formal commonsense reasoning. *Artificial Intelligence*, 153:1–12.

Dreyfus, H. L. (1979). *What Computers Can't Do: Revised Edition*. Harper and Row, New York.

Ellis, J. M. (1993). *Language, Thought, and Logic*. Northwestern University Press, Evanston, Illinois.

Giraud-Carrier, C. and Martinez, T. (1995). An integrated framework for learning and reasoning. *Journal of Artificial Intelligence Research*, 3:147–185.

Gödel, K. (1999). On formally undecidable propositions of principia mathematica and related systems I. In van Heijenoort, J., editor, *Frege and Gödel: Two Fundamental Texts in Mathematical Logic*, pages 87–107. iUniverse, Lincoln, Nebraska. Originally published in 1931.

Graubard, S. R., editor (1988). *The Artificial Intelligence Debate: False Starts, Real Foundations*. MIT Press, Cambridge, Massachusetts.

Halpern, J. Y., Harper, R., Immerman, N., Kolaitis, P. G., Vardi, M. Y., and Vianu, V. (2001). On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic*, 7(2):213–236.

Harman, G. (1982). Conceptual role semantics. *Notre Dame Journal of Formal Logic*, 28:252–256.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42:335–346.

Hayes, P. J. (1979). The naïve physics manifesto. In Michie, D., editor, *Expert Systems in the Micro-Electronic Age*, pages 242–270. Edinburgh University Press, Edinburgh.

Hearst, M. A. and Hirsh, H. (2000). AI's greatest trends and controversies. *IEEE Intelligent Systems*, pages 8–17.

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

Hofstadter, D. R. (1985). Waking up from the Boolean dream, or, subcognition as computation. In *Metamagical Themas: Questing for the Essence of Mind and Pattern*, chapter 26. Basic Books, New York.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Language, and Computation*. Addison-Wesley, Reading, Massachusetts.

Kirsh, D. (1991). Foundations of AI: the big issues. *Artificial Intelligence*, 47:3–30.

Kowalski, R. (1995). Logic without model theory. In Gabbay, D. M., editor, *What is a Logical System?*, pages 35–71. Oxford University Press.

Kugel, P. (1986). Thinking may be more than computing. *Cognition*, 22:137–198.

Kuhn, T. S. (1970). *The Structure of Scientific Revolutions*. Chicago University Press, 2nd edition.

Lakatos, I. (1999). *Proofs and Refutations: the Logic of Mathematical Discovery*. Cambridge University Press, Cambridge.

Lakoff, G. (1988). Cognitive semantics. In Eco, U., Santambrogio, M., and Violi, P., editors, *Meaning and Mental Representation*, pages 119–154. Indiana University Press, Bloomington, Indiana.

Langacker, R. W. (1999). *Grammar and Conceptualization*. Walter De Gruyter, Berlin. Cognitive Linguistics Research, 14.

Lenat, D. B. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Lucas, J. R. (1961). Minds, machines and Gödel. *Philosophy*, XXXVI:112–127.

Lucas, J. R. (1970). *The Freedom of the Will*. Oxford University Press.

Lucas, J. R. (1996). Minds, machines and Gödel: A retrospect. In Millican, P. and Clark, A., editors, *Machines and Thought: The Legacy of Alan Turing*, pages 103–124. Oxford University Press, Oxford.

Mancini, V. and Bandler, W. (1988). A database theory of truth. *Fuzzy Sets and Systems*, 25:369–379.

McCarthy, J. (1988). Mathematical logic in artificial intelligence. *Dædalus*, 117(1):297–311.

McCarthy, J. (1989). Artificial intelligence, logic and formalizing common sense. In Thomason, R. H., editor, *Philosophical Logic and Artificial Intelligence*, pages 161–190. Kluwer, Dordrecht.

McCarthy, J. (1990). Review of The Emperor's New Mind. *Bulletin of the American Mathematical Society*, 23(2):606–616.

McDermott, D. (1987). A critique of pure reason. *Computational Intelligence*, 3:151–160.

Michalski, R. S. (1993). Inference theory of learning as a conceptual basis for multistrategy learning. *Machine Learning*, 11:111–151.

Minsky, M. (1985). *The Society of Mind*. Simon and Schuster, New York.

Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: symbols and search. *Communications of the ACM*, 19(3):113–126.

Penrose, R. (1989). *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Oxford University Press.

Penrose, R. (1994). *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford University Press.

Popper, K. R. (1959). *The Logic of Scientific Discovery*. Basic Books, New York.

Russell, B. (1901). Recent work on the principles of mathematics. *International Monthly*, 4:83–101.

Schank, R. C. (1991). Where is the AI? *AI Magazine*, 12(4):38–49.

Scheutz, M. (2002). Computationalism — the next generation. In Scheutz, M., editor, *Computationalism: new directions*, pages 1–21. MIT Press, Cambridge, Massachusetts.

Searle, J. (1980). Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3:417–424.

Searle, J. (1984). *Minds, Brains and Science*. Harvard University Press, Cambridge, Massachusetts.

Simon, H. A. and Newell, A. (1958). Heuristic problem solving: the next advance in operations research. *Operations Research*, 6:1–10.

Sloman, A. (2002). The irrelevance of Turing machine to artificial intelligence. In Scheutz, M., editor, *Computationalism: new directions*, pages 87–127. MIT Press, Cambridge, Massachusetts.

Tarski, A. (1944). The semantic conception of truth. *Philosophy and Phenomenological Research*, 4:341–375.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, LIX:433–460.

Wang, P. (1994). From inheritance relation to nonaxiomatic logic. *International Journal of Approximate Reasoning*, 11(4):281–319.

Wang, P. (1995). *Non-Axiomatic Reasoning System: Exploring the Essence of Intelligence*. PhD thesis, Indiana University.

Wang, P. (2004a). Problem solving with insufficient resources. *International Journal of Uncertainty, Fuzziness and and Knowledge-based Systems*, 12(5):673–700.

Wang, P. (2004b). Toward a unified artificial intelligence. In *Papers from the 2004 AAAI Fall Symposium on Achieving Human-Level Intelligence through Integrated Research and Systems*, pages 83–90, Washington DC.

Wang, P. (2005). Experience-grounded semantics: a theory for intelligent systems. *Cognitive Systems Research*, 6(4):282–302.

Wang, P. (2006). *Rigid Flexibility: The Logic of Intelligence*. Springer, Dordrecht.

Wang, P. and Hofstadter, D. (2006). A logic of categorization. *Journal of Experimental & Theoretical Artificial Intelligence*, 18(2):193–213.

Whitehead, A. N. and Russell, B. (1910). *Principia mathematica*. Cambridge University Press, Cambridge.

Whitely, C. H. (1962). Minds, machines and Gödel: A reply to Mr. Lucas. *Philosophy*, 37:61–62.

Woods, W. A. (1986). Problems in procedural semantics. In Pylyshyn, Z. W. and Demopoulos, W., editors, *Meaning and Cognitive Structure: Issues in the Computational Theory of Mind*, pages 55–85. Ablex, Norwood, New Jersey.